A Systematic Evaluation Approach for Data Stream-based Applications

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatikerin

Sandra Geisler

aus Hagen, Deutschland

Berichter: Universitätsprofessor Dr. rer. pol. Matthias Jarke
Universitätsprofessorin Dr. rer. nat. Daniela Nicklas

Tag der mündlichen Prüfung:
21. Dezember 2016

Abstract

The ubiquitous use of mobile devices, sensors, and the linkage between them open up opportunities for new applications using near real-time data processing and analytics. Demands on information systems are high: huge amounts of data have to be processed and results have to be delivered in near real-time. These needs are tackled by the field of Data Stream Management. Processing data streams differs in many ways from static data set processing as much of the data cannot be stored persistently. Likewise, development methods for data stream-based applications have to be specifically adapted. Process modeling has proved to increase the quality of information systems, but there exists no model specifically for data stream applications. Furthermore, the production of high quality applications requires means for a structured, iterative evaluation of the application and its outcomes. Particularly, applications fed by unreliable data sources, such as sensors, are prone to quality losses and errors. Hence, measurement, monitoring, and optionally the correction of data quality problems must take a crucial part in the development and evaluation of data stream applications. Data quality management needs to be domain and application independent and smoothly integrated into a data stream management system. These requirements have not been met satisfactorily so far.

We counter the aforementioned issues by three main contributions. First, we propose a process model specifically tailored to the design, implementation, and, in particular, for the evaluation of data stream applications. To this end, we contribute a thorough analysis of data stream management principles and technologies. We also analyze existing process models in information management and discuss their suitability to data stream applications. Second, we propose evaluation methodologies embedded into the process model. Along these methodologies we design and implement a flexible evaluation framework for data stream applications. Finally, we propose a methodology and framework for data quality management for data stream applications. We first analyze quality dimensions and metrics relevant to data stream applications. We elicitate existing data quality management methodologies and present a methodology for data stream-based applications. As a major contribution we implement a flexible, domain and application independent data quality management framework for relational data stream management systems based on the proposed methodology.

The process model and frameworks have been developed and empirically validated and evaluated in the context of two domains, namely Connected Intelligent Transportation Systems and Mobile Health. Algorithmic solutions for particular problems in the target domains have been devised and applied. Iterative evaluations using the proposed frameworks led to crucial optimizations of the application results.

Zusammenfassung

Die umfassende Nutzung von mobilen Endgeräten, Sensoren und deren Vernetzung ermöglicht die Entwicklung neuer Anwendungen im Bereich der Echtzeitdatenverarbeitung und -analyse. Die Anforderungen an Informationssysteme sind hoch: große Datenmengen müssen verarbeitet und Ergebnisse in Echtzeit erzeugt werden. Das Gebiet Datenstrommanagement beschäftigt sich mit diesen Anforderungen. Es unterscheidet sich sehr von der Verarbeitung statischer Datenmengen, da nicht alle Daten persistent gespeichert werden können. Daher muss es auch spezielle Entwicklungsmethoden für Datenstromanwendungen geben. Der Einsatz von Prozessmodellen kann zu qualitativ besseren Informationssystemen führt, bisher gibt es aber kein Modell explizit für Datenstromanwendungen. Zur Erstellung qualitativ hochwertiger Anwendungen muss man zudem eine fortlaufende, strukturierte Evaluierung durchführen. Insbesondere führen unzuverlässige Datenquellen, wie Sensoren, zu Qualitätseinbußen und Fehlern. Daher müssen Messung, Überwachung und eventuelle Korrektur von Datenqualitätsproblemen bei der Entwicklung und Evaluierung von Datenstromanwendungen eine zentrale Rolle spielen. Das Datenqualitätsmanagement sollte außerdem unabhängig von der Anwendung sein und sich gut in ein Datenstrommanagementsystem integrieren. Dies wurde bisher nur unzureichend gelöst.

In dieser Arbeit gehen wir die genannten Probleme hauptsächlich mit drei Beiträgen an. Zuerst schlagen wir ein Prozessmodell vor, das für das Design, die Implementierung und speziell für die Evaluierung von Datenstromanwendungen geeignet ist. Dazu analysieren wir ausführlich die Grundlagen und Technologien des Datenstrommanagements. Wir analysieren existierende Prozessmodelle für Informationssysteme und deren Eignung für Datenstromanwendungen. Als Zweites werden Methodiken für eine strukturierte Evaluierung erarbeitet und entlang dieser ein flexibles Evaluierungsframework entworfen und implementiert. Als dritten Beitrag beschreiben wir eine Methodik und ein Framework für das Datenqualitätsmanagement bei Datenstromanwendungen. Dazu werden relevante Qualitätsdimensionen und -metriken analysiert, existierende Datenqualitätsmanagementmethoden beleuchtet und schließlich eine Methodik für Datenstromanwendungen präsentiert. Den zentralen Hauptbeitrag stellt das flexible, domänen- und anwendungsunabhängige Framework zum Datenqualitätsmanagement für relationale Datenstrommanagementsysteme dar.

Das Prozessmodell und die Frameworks wurden im Kontext der Anwendungsfelder Vernetzte Intelligente Transportsysteme und Mobile Gesundheit entwickelt, empirisch validiert und evaluiert. Im Rahmen dieser wurden zusätzlich algorithmische Lösungen für spezielle Probleme entwickelt und umfassend evaluiert. So konnten wir entscheidende Probleme und Einflüsse entdecken und Optimierungen vornehmen.

Acknowledgments

Finishing this thesis was one of the most challenging things in my life so far. In particular the last three months of this journey took a lot of will, tenacity, and strength to complete it (which sometimes I still cannot really believe that I finally did). I learned a lot during this thesis project, most importantly that some things are better finished than perfect and that without the support of the many people around me it would not have been possible. To these people I owe huge gratitude which I would like to express in the following.

First of all, I would like to thank Prof. Dr. Matthias Jarke who gave me the opportunity to work on interesting projects in different domains, which contributed to the elaboration of this topic. He helped me to tame the huge amount of material, and gave me crucial hints when needed but without restricting the way I was following. Furthermore, I would like to thank Prof. Dr. Daniela Nicklas who kindly agreed to serve as the second reviewer for my thesis, as well as Prof. Dr. Jürgen Giesl and Prof. Dr. Bastian Leibe for participating as additional examiners in the defense committee and finally Prof. Dr. Hermann Ney heading the committee.

In particular, I would like to thank my group manager, colleague, and mentor PD Dr. Christoph Quix for his persistent support during my time as a student, as a PhD student, and as a research assistant. He has an open ear for all kinds problems, the discussions with him are always fruitful, and he furthered my research skills tremendously. But for the first pushes towards writing publications and starting a PhD Dr. Christa Wessel and Prof. Dr. Anke Schmeink deserve special gratitude. They have been great role models and believed in my skills as a researcher. Furthermore, I would like to thank my dear colleagues at the Big Data and Model Management group, the chair Informatik 5, and the Knowledge-based System Group for their constructive collaborations and support. Especially, I am grateful that (in alphabetical order) Sebastian Brandt, Dr. Jessica Huster, Dr. David Kensche, and Dr. Zinaida Kensche neè Petrushyna have been more than colleagues to me, but got friends over time and helped to keep my chin up. Also, I would like to thank István Koren being such a nice and cheerful office mate enduring my big and little peculiarities specifically during the final phase of my thesis. Also, I like to thank Prof. Dr. Lakemeyer for his consistent motivation by friendly teasing me, but also giving advice. A big gratitude I also owe to our administrators, Reinhard Linde and Tatjana Liberzon, who always helped me out when there were technical barriers, deep despair, or chocolate shortages. Last but not least the good spirits of the chair (in order of appearance), Gabriele Hoeppermanns, Daniele Gloeckner, and Claudia Puhl deserve a big thank you for their support and great work. I also would like to thank all of the colleagues and project partners in the UMIC HealthNet and the CoCar(X) projects for the wonderful and fruitful collaboration. Most importantly, I also would like to thank all of the students whose theses I advised and/or who were involved as student workers in projects during my time at the chair. All of them either contributed to the overall thesis project directly or implicitly.

Finally, huge gratitude deserves my family: my parents Ingrid and Heinz Geisler for believing in me, loving me, and always supporting me in my decisions. I know, that my father would have been very proud if he would have had the chance to witness the end of this chapter of my life. I thank my brother Frank Geisler, who initiated my interest for computer science (well, computers at all), supported the development of my professional skills, especially in database management, and had an open ear for my problems of whatever nature. Furthermore, I thank Dorothee Jasnoch for being a best friend no matter what. Moreover, I thank my (practically) parents-in-law, Franz-Josef and Marita Schiffer, who supported us as a family, often cared for Jakob when time was short, and also took over a lot of organizational things.

Last but not least I want to thank my dear son Jakob and my beloved husband Stefan who I owe the biggest gratitude. Stefan always motivated me when I was down, helped me to never give up, and was bearing all of my quirks. He sacrificed much of his time especially in the last few months to support me and took over many of my duties. Finally, I thank my son Jakob who always was the sun shining on each single day since he was born. No matter how bad a day was, he cheered me up by the small and big things he did and said. I love you both more than I can say.

To my family

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Gartner prognoses that by 2020 a quarter billion connected vehicles will be on the road and in general the number of connected things will reach 20.8 billion worldwide.[1][2] The amount of data produced by these devices will be tremendously large. Research in information systems and management must provide means to tame the data and turn it into business advantages.

Connectivity is a major requirement for many devices as users want to retrieve additional information or send data to gain benefits, for example in terms of services. The research fields of the Internet of Things (IoT) and Industry 4.0 address these demands. Cheap sensors, high bandwidths, affordable mobile Internet connections, and modern mobile devices promote excess data. Users not only demand to extract the most of all data produced by the devices, but also to access the data in near real-time to always get a current view on it. The abundance of real-time data opens up many opportunities for new applications, research fields, and business branches. Real-time trading of stocks or energy, health monitoring, sentiment analysis in social networks, or monitoring of production lines are only a few examples of many, which evolved due to the capabilities of real-time data processing. However, the huge amount of data poses problems to common data management solutions, such as Data Warehouses (DWHs). These solutions have worked fine for offline data analysis. For near real-time querying and analysis of huge amounts of data (also termed Big Data) they are not suited anymore.

Hofstee and Nowka (2013) describe the most demanding Big Data problems by four V's: volume, velocity, variety, and veracity. All four problems are tackled specifically for real-time data by the research field of data stream processing. Data streams are regarded as unbound sequences (i.e., with unknown end and size) of data items usually being produced at a high rate and at a high amount (Babcock et al., 2002a). Data Stream Management Systems (DSMSs) have been proposed to query, aggregate, or filter this data in real-time enabling real-time data analysis (Golab and Özsu, 2010).

After about two decades of research, DSMSs have matured to distributed, powerful data management products, which are used either as single solutions or as part of a Big Data Analytics ecosystem (Dolas, 2015). Hence, real-time analytics also have been adopted in industry and are utilized in a multitude of crucial business cases implementing *data stream applications*. For example, Walmart monitors and analyzes its sales in real-time, and Rolls-Royce monitors production lines using real-time analytics (Marr, 2016). However, to design and implement a complex, efficient, and reliable data stream

[1]`http://www.gartner.com/newsroom/id/2970017`
[2]`http://www.gartner.com/newsroom/id/3165317`

application realizing such business cases is a challenging and time consuming task. As with any other data management or software project, the project costs will multiply if the solution is implemented different as required and expected.

Many steps and problems reoccur for all data stream applications during development. Therefore, it would be beneficial for a developer to be guided through the development process while considering the problems and tasks inherent to data stream applications. Furthermore, the developer has to find out if the application behaves as expected and produces the expected results. Hence, additional guidance for evaluating the application is equally important.

Data analysis in mobile applications faces additional problems when involving connected devices and sensors. The utilized devices may fail or deliver wrong data due to many reasons including battery depletion, environmental influences, low production quality, misconfiguration, or connection failures (Paradis and Han, 2007). Data analysis applications based on such data sources inevitably produce results of lower quality following the *garbage-in garbage-out philosophy* (Gabbay and Guenthner, 2002). Consequently, applications consuming data from potentially error-prone data sources should be provided means to detect and handle varying quality of the processed data. Hence, the evaluation of data stream applications should consider data quality (DQ) problems as well as implementation and system-related issues.

Based on these considerations, we define the research goals and the corresponding contributions of the thesis in the following section.

1.1 Research Goals & Contributions

As discussed in the previous section quality is a crucial demand for applications processing real-time data, in particular, if connected devices and sensors are involved. Hence, the overall goal of this thesis is to make contributions to the improvement of data stream applications and their outcomes. We want to do this by providing support along the whole process of data stream application development. That means, that (1) the development process of data stream applications should be structured to achieve high quality real-time information management solutions, (2) means for a continuous evaluation process of the applications and their results should be provided, and (3) the monitoring and improvement of data quality throughout the whole application should be enabled. These are the three main research goals of this thesis and we will derive the according research questions and describe our contributions to each goal in the following.

1.1.1 Structured Development Process

The definition and modeling of a process is important to make the process repeatable and increase its quality. Approved steps and methods can be proposed to developers to help them concentrate on the task. Thus, we want to elaborate such a **structured process for data stream application development**. In the course of this task we need to answer the following questions.

- **What is a data stream application and how is it structured?**
  We first need to elaborate what are the characteristics of a data stream application and which components it basically comprises. Furthermore, we have to know about typical environments in which data stream applications are executed (i.e., data management architectures) and their specifics.

- **What are important steps in the development process?**
  As data stream applications can be categorized into the area of information management applications, it is worth looking into which abstractions of processes (process models) have been proposed for other information management application types. We then can analyze if these abstractions and the proposed steps are also suited for data stream applications. We have to determine if parts could be adopted and what is missing completely to support data stream applications. In particular we want to focus on the evaluation of applications and their outcomes. We also have to think of, how the steps are connected with each other and if some of these should be repeated and when they should be repeated.

- **What are methods to execute the steps in the development process?**
  The development process was regarded as an abstract description so far. However, to be suitable for practical use we also have to think of which methods can fill steps in such a development process description.

Contributions

We add a thorough analysis of data stream management principles and technologies. Equipped with this knowledge we can derive implications about the development process of data stream applications. Further, we will contribute a deep analysis of existing process models in information management and a discussion of their suitability to data stream applications. Also, we will contribute a discussion of practical methods usable in the data stream application development process. Based on the gained knowledge we will define **a process model for the development of data stream applications**.

1.1.2 Continuous Evaluation

Similar to software development, the development of a data stream application should not be a static process. The outcome should be evaluated iteratively adapting internal and external parameters to achieve the best performance in all senses. Hence, we need **a structured method to evaluate data stream applications continuously**. However, this goal also brings up several research questions which are as follows.

- **How can we determine which parameters influence an application and how?** First we need to know which parameters of a data stream application and its surrounding architecture can be changed. Are there general parameters valid for all applications and systems? Which are the application specific parameters? We have to determine how we can design and execute an evaluation to find the degree of influence of the parameters. Furthermore, it has to be investigated how combinations of parameters influence the application and its results.

- **Do recurring steps and components for evaluation exist?** It is also of interest to know, if a common evaluation setup for stream applications could be found, which can be used independently of domains and systems.

- **How can the results of specialized algorithms in data stream applications be evaluated?**
  To fulfill complex tasks in data stream applications, such as prediction or classification, streaming algorithms have to be used. It is of interest how these algorithms and their performance can be evaluated systematically. In particular,

it is of interest to analyze which algorithm is suited best for which task which parameters influence their results how, and how much do the results influence the overall result of the application.

Contributions

We will provide guidelines and methods for a structured and repeatable evaluation embedded into our process model. These will be exemplified on multiple applications in two different domains. In particular, we will propose an **evaluation framework** for data stream applications, which offers exchangeable components. Finally, we will contribute comprehensive validations on the examples of data stream classification and map matching algorithms.

1.1.3 Data Quality Management

One major issue in mobile applications with connected devices and sensors is data quality. Unreliable data sources make the production of high quality results very difficult. Hence, we urgently need a **flexible, general, and structured way to manage data quality for data stream applications**. This goal raises several research questions.

- **Which aspects according to data stream applications have to be evaluated?**
  We first need to know what are typical quality problems in data stream applications and what are their causes. These comprise data quality problems as well as implementation and system-related problems. Furthermore, it is of interest if these quality problems differ depending on the domain or the application. Finally, we need to elaborate how the quality can be expressed and calculated.

- **What are crucial steps for the measurement of quality?**
  When it is known what should be measured and how will it be expressed, it should be investigated, what high-level steps are required for the process of quality measurement. Again it is worth finding out which DQ methodologies exist and if they are suited for data stream management or can be adapted.

- **How can data quality management be implemented for data stream applications?**
  As soon as the general steps for DQ management are known the question arises how these can be filled towards implementation for data stream applications and DSMSs. In particular, it must be considered, how the implementation can be made independent from domains, dimensions, metrics, and as good as possible from the underlying system. With this in mind, we will contribute the implementation of a DQ management framework for relational DSMSs.

Contributions

We will deliver a thorough analysis of data quality dimensions and metrics in particular for data stream applications. Further, we contribute a detailed analysis of DQ methodologies and propose **a methodology for DQ management in data stream applications**. Finally, we enable a structured and detailed evaluation of data quality in stream applications by developing **a DQ management framework for data stream applications** based on the provided methodology.

The elaboration of the research questions and the presentation of the contributions are structured as described in the following section.

1.2 Outline

The remainder of the thesis is organized as follows:

Part I: Foundations

In this introductory part we explain the fundamentals of our application domains. Furthermore, a detailed introduction to the principles of data stream management and processing are given.

Chapter 2: We introduce the field of Connected Intelligent Transportation Systems (C-ITS) and its foundations and challenges as it is one of our two main application domains. In particular, basic terminology, the important principles of traffic disruptions, categorizations of traffic states, and the concepts of queue-ends are detailed. Furthermore, the most important measured traffic parameters are delineated. Stationary and mobile data sources producing these traffic parameters are introduced and described. Subsequently, traffic models, which are crucial for simulation tools, are introduced and categorized. Finally, traffic applications from the C-ITS domain are explained. Parts of this chapter have been previously published in (Geisler et al., 2012).

Chapter 3: Chapter 3 introduces the most important concepts of the second application field of the thesis, namely Mobile Health (mHealth). A general architecture of mHealth systems, its components, and main challenges of mHealth applications are identified. Subsequently, the most common data sources of these applications are identified, which mainly comprise sensors. Types of medical sensors are briefly introduced, followed by a brief description of Body Sensor Networks (BSN) and sensor fusion principles. Finally, examples for data stream-based mHealth applications are given to show potential application fields.

Chapter 4: This chapter forms the basis for the understanding of data streams and their main concepts. The crucial challenges in data streams are presented and different architectures types tackling these challenges are introduced while delimiting our field of research. Moreover, we detail important foundations of data stream languages emphasizing semantics and operators. Furthermore, the foundations of data quality management in the context of data stream research are highlighted. Finally, the field of data stream mining, in particular data stream classification, is covered. Parts of this chapter have been previously published in (Geisler, 2013; Geisler et al., 2016, 2011).

Part II: Process Model & Applications

The second part of the thesis covers the explanation of the proposed process model for data stream-based applications. Furthermore, its application in the domains C-ITS and mHealth is demonstrated and evaluation results for each of the applications presented.

Chapter 5: The chapter explains our process model for the design, implementation, and evaluation of data stream-based applications. It starts with the discussion of previous process models and methodologies in information management and data mining. Afterwards, we detail general approaches to the design of data stream applications. We then present the overall view on the process model and go through each component in the corresponding sections. For each step the purpose, the challenges, the desired outcome, and possible ways to realize the step (methodologies) are presented.

Chapter 6: After we explained our process model we show the usage of the model in two applications in the C-ITS domain in this chapter. First, the project context of the applications, the Cooperative Cars project, is detailed. Subsequently, for each of the chosen applications, Queue-end Detection (QED) and Traffic State Estimation (TSE), we follow the process model and use one of the proposed methodologies. For each step in the process model the outcome is shown and the results are discussed. Parts of this chapter have been previously published in (Geisler et al., 2011, 2010; Geisler and Quix, 2012; Geisler et al., 2016, 2012).

Chapter 7: Chapter 7 utilizes the process model in the course of two mHealth projects and applications. We introduce the context of each project. It presents a different approach to the stream application development as the prerequisites differ to the C-ITS applications in the previous chapter. Again for each application we go step-by-step through the process model and show the results. Parts of this chapter have been previously published in (Geisler et al., 2016).

Part III: Data Quality Management & Structured Evaluation of Algorithms

Chapter 8: In this chapter we first discuss the related work in the field of data quality management in data streams. Subsequently, we propose a general data quality management methodology for data streams and delineate the implementation of our DQ management framework. Finally, we evaluate the DQ framework in the applications we described and implemented in Chapters 6 and 7. Parts of this chapter have been previously published in (Geisler et al., 2016; Geisler and Quix, 2012; Geisler et al., 2011).

Chapter 9: In the course of the C-ITS applications implemented in Chapter 6, we used a simple Map Matching algorithm. Previous evaluation results showed that improvement of the algorithm is beneficial to the overall accuracy. Hence, we describe the development of our own Map Matching algorithm in this chapter after discussing related work. Furthermore, the evaluation of the algorithm for the C-ITS applications and the corresponding results are highlighted. Finally, we present the results of our algorithm achieved in a Map Matching contest in the ACM SIGSPATIAL GIS Cup 2012 (Ali et al., 2012) in comparison to state-of-the-art algorithms. The work presented in this chapter has partly been developed and evaluated in the scope of a Master Thesis by Mitre (2012) supervised by us.

Chapter 10: The last chapter of this part highlights the evaluation of data stream classification algorithms on the example of the C-ITS applications. We first present the results of the comparison of different data stream algorithm types. We then investigate the influence of over- and undersampling on the algorithms and present a dynamic balancing algorithm for data streams. Finally, we also delineate the results of our experiments investigating the influence of concept drift. Parts of this chapter have been previously published in (Geisler et al., 2012; Geisler and Quix, 2012).

Part III: Epilog

This part concludes the thesis by discussing the contributions in a narrow and broad perspective. Finally, an outlook is given on possible avenues for further research on the topic.

Some of the material covered in this thesis was developed in in Bachelor, Diploma, and Master theses supervised by me (Chen, 2009; Soleimankhani, 2010; Weber, 2011; Mitre, 2012; Jongiran, 2013; Claßen, 2013). I would like to thank the people involved for their support and contribution.

Part I

Foundations

Chapter 2

Application Domain I: Connected Intelligent Transportation Systems

In this part we introduce two important domains which pose challenges for real-time data processing - Connected Intelligent Transportation Systems (C-ITS) and Mobile Health (mHealth). These are just two examples for many other domains which have (near) real-time data analytics requirements. Although we are not restricted to these domains, they are interesting and fruitful domains with common and diverse properties, which support the development and evaluation of our contributions. Many examples in the following chapters are also based on the two example domains. Hence, a thorough understanding of the most important terms and concepts used in these domains is indispensable.

There exist several definitions for the term Intelligent Transportation System (ITS). The U.S. Department of Transportation, for example, defines that ITS "apply well-established technologies in communications, control, electronics, and computer hardware and software to improve surface transportation system performance."(Sussman et al., 2000) The definition names the most important components of an ITS, but it remains vague, as it is not clear what the term "well-established" means. The Journal of Intelligent Transportation System issued by Taylor & Francis gives a more detailed and research oriented definition: "Intelligent transportation systems [...] are characterized by information, dynamic feedback, and automation that allow people and goods to move efficiently. They encompass the full scope of information technologies used in transportation, including control, computation and communication, as well as the algorithms, databases, models, and human interfaces."[1]

In this chapter, we are especially interested in the implementation of ITS and the components mentioned in the last definition by means of mobile data sources, termed Connected ITS (C-ITS) (European Comission, 2016) or Vehicle-2-X (V2X) or Car-2-X (C2X) applications where X stands for other vehicles and infrastructure (V2V: Vehicle-2-Vehicle, V2I: Vehicle-2-Infrastructure, likewise it is C2C: Car-2-Car, C2I: Car-2-Infrastructure). These represent an particularly interesting domain as they incorporate streaming data and static data from a multitude of data sources. The streaming data is constituted of high volumes of geospatial and other data produced in a short period of time. These promise a better resolution and coverage in terms of location

[1] `http://www.tandfonline.com`

and enable real-time ITS applications.

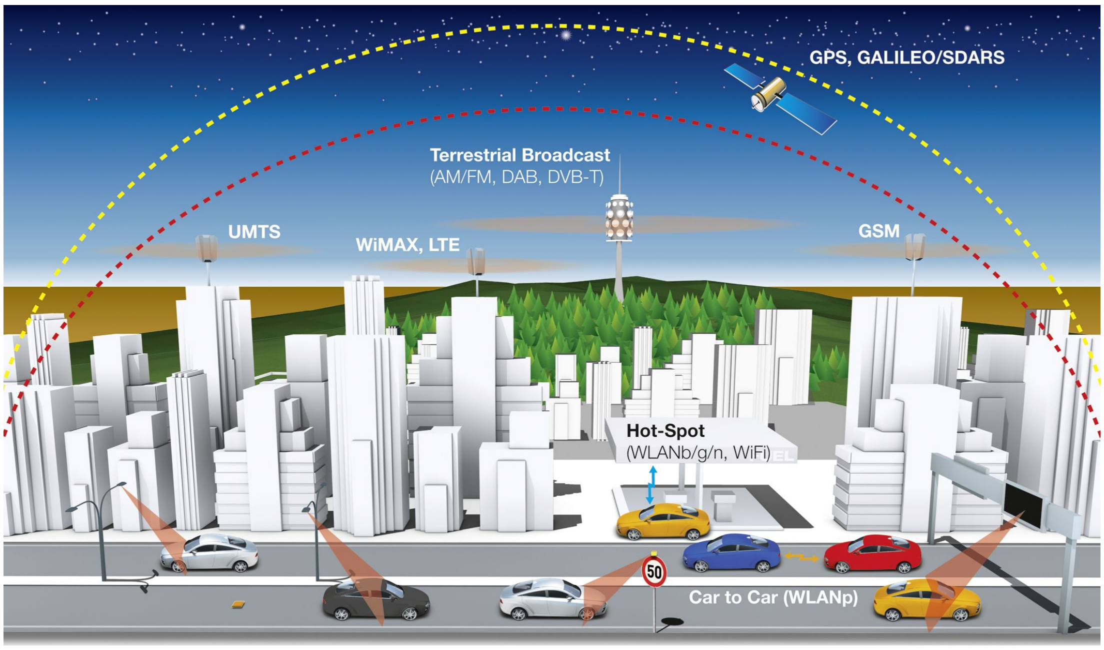


Figure 2.1: Connected Intelligent Transportation Systems[2]

In Figure 2.1 the possible components and participants of a C-ITS are depicted. Typically, a *positioning system*, such as GPS or GALILEO is required to determine the position of a vehicle. This can also be done via cellular networks, but a satellite driven system usually has a better positioning precision. As both are not perfectly correct, the measured positions might not exactly lie on a road. Hence, to make assumptions about traffic parameters, the positions need to be matched to a road network. The algorithms estimating the match are termed *Map Matching* algorithms. An introduction to Map Matching is given in Chapter 9. Furthermore, the C-ITS uses one or more wireless technologies which comprise cellular networks (e.g., LTE, UMTS, GSM), wireless networks with standardized common protocols (IEEE 802.11 b/g/n), and specific protocols for automotive applications (IEEE 802.11p). Similar to the components in Figure 2.1 the ETSI (European Telecommunication Standards Institute) summarizes the components in subsystems: personal ITS subsystem (e.g., mobile phone), vehicle ITS subsystem (including sensors and in-vehicle networks), central ITS subsystem (central ITS station and network with gateways, routers etc.), and Roadside ITS subsystems (e.g., roadside network with gateways, routers etc.) (ETSI EN 302 665, 2010).

In the following, we first explain the most important basic terms and concepts related to transportation. Subsequently, common data sources used in the traffic domain for ITS and C-ITS are introduced followed by a categorization and explanation of existing ITS projects. Finally, the applications used in this work's case studies are explained.

[2]`http://www.continental-corporation.com`

2.1 Traffic Engineering Terminology

Queues in traffic are events which have negative effects regarding many perspectives. Besides being annoying for the individual drivers (in 2015 German drivers spent on average 29.57 hours in congestion)[3], they also have negative economical and environmental influences. Cargo is not reaching its destination on time, working persons are not productive and even miss appointments. Furthermore, the amount of exhaust gases increases. Finally, traffic jams also pose a big safety risk in form of queue-ends. Accidents with vehicles driving into a queue-end are happening frequently and often result in injured persons or even fatalities. Hence, queue-end detection and corresponding warnings are an interesting application in C-ITS, as they could prevent accidents by increasing the driver's awareness (Bogenberger and Dinkel, 2009). This application is especially challenging for C-ITS as the queue-end moves quickly and a real-time tracking of the queue-end is necessary. Real-time traffic state estimation and tracking is offered by several commercial and public service providers. However, it is still a challenge to offer an accurate real-time estimation based on diverse data sources, though traffic state estimation seems like a relatively simple task to implement at first sight.

In the following, we will first explain the concepts crucial for traffic states, in particular queues, and queue-ends. Then, we will explain the most common and important traffic parameters that are relevant for the selected applications. The data sources which can be used to measure those traffic parameters will be detailed in the subsequent section.

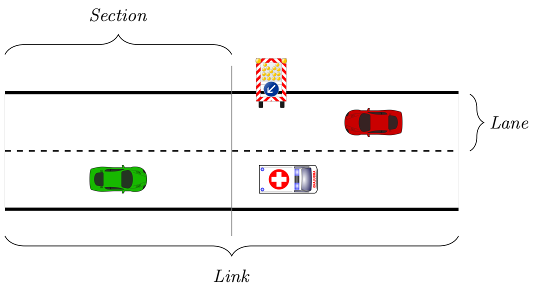
2.1.1 Basic Terms and Road Infrastructure

The basis of almost all traffic applications is a *road network*, which spans a certain geographical area (cf. Figure 2.2(a)). A road network consists of a set of roads or *links*, each labeled with a unique identifier. Links have a start point, an end point, and a direction, i.e., two directions of the same road are two individual links. Links can be classified according to different aspects, e.g., to their type, such as highway, urban roads, or rural roads, or according to the admitted velocity. Each link consists of one or more *lanes*. Each link can further be divided into *sections*, which are equally or differently sized parts of the link. On a link *vehicles* are driving. Figure 2.2(b) illustrates how these terms are related to each other. There exist different classifications for vehicles, e.g., according to their type (car, truck, motorcycle, bus etc.), their speed (slow moving etc.), or their purpose (e.g., public and individual transport). Additionally, a road has an *infrastructure*. The infrastructure consist of dynamic and static road signs, and units to measure, process, and communicate. This infrastructure varies, depending on the type of the road.

The typical infrastructure in the German highway network is divided into several regional sub-networks which all have the same hierarchy: a Traffic Control Center (TCC), a sub control center, and a section station with data measurement and output facilities and a control unit (Kirschfink, 2000). The section station gathers the data from the measurement units, aggregates them and manages the traffic control systems. The measurements are executed lane-wise and comprise the measurement of traffic volume and speed of all vehicles and trucks separately. Traffic control systems can be categorized into Rerouting Systems, Line Control Systems, Systems for temporary Hard

[3]`http://ec.europa.eu/transport/facts-fundings/scoreboard/compare/energy-union-innovation/road-congestion_en#2015`

(a) Example Road Network (Extract from Openstreetmap)



(b) A Simple Link

Figure 2.2: Visualization of Basic Terms

Shoulder Running, and Ramp Metering systems.[4] *Rerouting Systems* are dynamic road signs proposing alternative routes to drivers in cases of road blockages or congestion and inform the drivers about the cause. *Line Control Systems* are variable road signals, which adapt speed limits, warnings, and overtaking bans for lanes and links to the current traffic and weather situation. *Hard Shouldering Systems* enable the opening of hard shoulders (lanes on the very right of the highway for broken down vehicles) in cases of heavy traffic. *Ramp Metering* systems control the flow of vehicles entering a highway when traffic volume is high.

A Traffic Control Center (TCC) monitors and controls the highway infrastructure. In particular a TCC has the following tasks:[5]

- harmonization of traffic flows,
- road condition warnings,
- proposals for alternative routes,
- determination and analysis of traffic situation,
- monitoring and controlling of traffic control systems,
- monitoring of tunnels, and
- other services, e.g., finding free spots for trucks on parking lots.

Not all traffic control and sub control centers in Germany use the same software to manage and analyze their data and to control the infrastructure. However, there exists an initiative called NERZ e.V.[6], in which eight federal states of Germany have committed to maintain and develop the ERZ Software (Einheitliche Rechnerzentralen-software für Verkehrsleitsysteme, Engl.: unified traffic center software for traffic control systems). The software consists mainly of a data broker, which loosely connects data sources and traffic applications (beck et al. projects GmbH, 2006). Data brokers can also be coupled to support load balancing and failure safety.

[4] `http://www.svz-bw.de/vba.html`
[5] `http://www.svz-bw.de/vrz.html`
[6] `http://www.nerz-ev.de`

2.1.2 Principles of Traffic Disruptions

Traffic States

Traffic states or *Levels of Service (LOS)* are a classification of the current traffic situation and occupancy of a road section. There have been several approaches for modeling these levels. For example, Kerner (2004) defines three states (free, synchronized, and congested traffic), while the Highway Capacity Manual of the U.S. Transport Research Board proposes six states ranging from A to F (A = free flow, B = reasonably free flow, C = stable flow, D = approaching unstable flow, E = unstable flow, F = forced or breakdown flow). The Federal German Highway Research Institute defines a classification which has been widely implemented in German Traffic Management Centers (BASt, 1999). It distinguishes four levels of traffic: free, dense, slow-moving and congested traffic. For each level and a number of lanes a threshold for the mean speed and the local density are defined. For example, for a highway with two lanes the traffic state is *slow-moving*, when the mean speed is above or equal 30 km/h and below 80 km/h and the density is below or equal 50 veh/km. The ERZ software (beck et al. projects GmbH, 2006), which is widely used in Germany, has a more fine-granular classification with six levels: free flow, vivid, dense, slow-moving, partially congested, congested (cf. Figure 2.3).

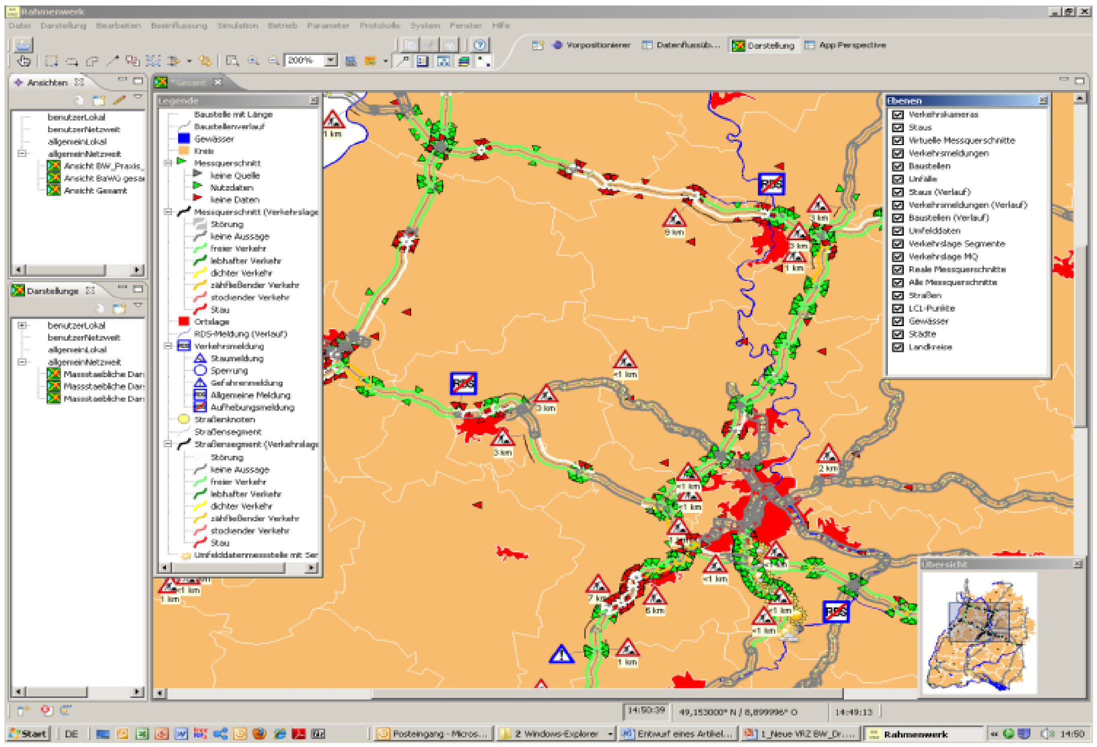


Figure 2.3: Screenshot from the ERZ Software Visualizing Traffic States[7]

The determination of the traffic state for a section of the road network is called *Traffic State Estimation* (TSE). Methods for traffic state estimation and prediction are discussed in Section 2.4.1.

[7]`http://www.nerz-ev.de`

Traffic Disruptions

The types, causes, and movements of traffic disruptions are a huge research area themselves. There exist different approaches on how to describe the phenomenon of congestion. Based on the observation of individual vehicles' movements, queues can be categorized into *short speed drop* (vehicle brakes, but accelerates very quickly again), *stop and go* (in a section the vehicle brakes multiple times and accelerates again), *wide congestion* (driving at declined speed for a long time), and *stationary traffic* (vehicle remains at the same position for a longer time) (Bogenberger and Dinkel, 2009). Disruptions and instabilities in the overall traffic flow are a combination of several influential factors. The capacity of a road is one aspect. The rule of thumb for the capacity of a lane is 1800 veh/h based on the recommended time gap between two vehicles. Normally, the very right lane has a lower capacity due to the higher number of trucks. Another aspect is the speed of the vehicles. Flowing traffic is categorized into *free traffic* where vehicles can choose speed and lane freely (*target speed*) and are not influenced by other vehicles. In *bound traffic* other vehicles influence the speed of an individual vehicle, which in the worst case may lead to synchronized traffic where all vehicles drive at the same speed (Kerner, 2004; Mensebach, 2004). If the traffic volume reaches the capacity of the lane, the traffic is bound. Furthermore, the distance between vehicles makes a difference, because vehicles have to adapt their speed if the distance to the vehicle in front is getting too small. The overall braking time consists of reaction time for braking and accelerating, and the braking and acceleration time of the vehicle to adapt the speed to (Treiber and Kesting, 2010). All of these factors are of course connected. If the capacity of the road is exceeded by the traffic demand, vehicles cannot freely choose their speed anymore and this will lead to instabilities and a breakdown of the traffic flow. Causes for instabilities and capacity excess could be bottlenecks, high traffic volume, and disruptions in traffic flow (hard braking actions, overtaking trucks etc.). Instabilities can be categorized according to the number of affected vehicles, size of disruptions, direction of propagation of the disruption (upstream, downstream, or both), and wavelength of disruptions (because disruptions propagate in waves). Disruptions can move and grow in one direction, which is called *convective instability* (Treiber and Kesting, 2010). The dynamics of disruptions can be classified by six patterns: pinned localized clusters, moving localized clusters, triggered stop-and-go, oscillating congested traffic, homogeneous synchronized traffic, and homogeneous congested traffic (Treiber and Kesting, 2010).

Queue-ends

There also exist different categorizations and definitions of a queue-end. A *queue-end* in this work designates the spatial position where a vehicle enters the disruption upstream. In the literature also the temporal end of the queue (disruption is resolved) or the downstream queue-end (vehicle leaves the last disrupted section) are defined. Furthermore, soft and hard queue-ends are distinguished. *Soft queue-ends* are characterized by synchronized traffic flowing upstream towards the disruption resulting in "soft" deceleration. *Hard queue-ends* cause a high deceleration, because traffic is flowing freely towards the queue-end. Bogenberger and Dinkel (2009) summarize characteristics of hard and soft queue-ends which are, amongst others: lane speed, speed difference on lanes, lane changes, lane occupancy, vehicle convoys, and type of speed drop. Furthermore, queue-ends can be visible or hidden, depending on the road topology (Bogenberger and Dinkel, 2009). This results in four different classes of queue-ends which pose different levels of safety risks for approaching vehicles.

2.1.3 Traffic Parameters

"Traffic can be seen as a process which is described by measured traffic parameters and which is continuous in space and time" (Hoyer, 2003). The parameters which are used for different traffic applications can be gathered in different ways. Raw data can be measured by several kinds of detectors, especially stationary and mobile detectors, but it can also be obtained from other sources such as traffic planning data. In this section common traffic parameters and their measures are shortly explained.

Traffic parameters can be categorized according to different aspects as follows:

- **Type:** The traffic data can be raw, i.e., directly observed by a detector or read from other sources, such as topological information. This data can be further aggregated to decrease the level of granularity, such as aggregation of values per minute to values per hour. Furthermore, data can be calculated or derived from other data to gain new information.

- **Reference Point**: The different parameters can be distinguished according to the geographic granularity they cover. Local data, for example, can be measured at a single geographic position on the road (in stationary detection also called gauging section), such as the local velocity. Other data references a part of the road (section-based data) or a specific network.

- **Detection Mechanism**: The technique how data is measured determines several criteria of traffic data, such as the unit or the level of granularity, the accuracy, and the reliability. In general, the following categories for data sources are distinguished:

  - *Stationary Detection*: Detectors measure traffic data at a fixed position.
  - *Mobile Detection*: The vehicles themselves have technical equipment to report their geographic position and other data.
  - *Traffic Planning Data*: Data available from traffic planning, such as the type of road, location of detectors, or the traffic capacity of a road are written down in traffic plans.
  - *Detectors for Stationary Traffic*: These detectors are used to obtain information about parking cars, for example detectors at parking garages.
  - *Environment Measurements*: These detectors do not produce traffic data directly, but the information can be used in different applications to enrich or derive traffic information.
  - *Traffic News*: This includes messages issued by the police or automobile clubs and also messages about road works.
  - *Historical Data*: Measurements and aggregated data from the past, which have been permanently stored, can be used to enhance the quality of current measurements by resolving erroneous or missing measurement data. Of course also long-term analyses can be based upon historical data.

  The data sources are described in detail in Section 2.2.

- **Measurement Site:** Not only the point of reference is important in traffic applications, but also the kind of roads which are addressed in the application. Techniques for applications, such as traffic estimation and forecasting, differ according to the road type they address. A coarse distinction can be made between urban areas and highways, but there also exist more fine grained categorizations, using categories such as motorways, residential roads, and urban main roads.
- **Individual Car / Traffic Flows:** The traffic data can also be categorized according to the point of view on the traffic. For individual cars data such as acceleration, lurch, or velocity can be measured. Furthermore, traffic can be seen in whole and described by parameters such as traffic volume or density.

In Appendix A.1 the most important parameters measured for individual vehicles (cf. Table A.1)and traffic flows (cf. Table A.2) are listed. For individual vehicles distance (m), occupancy (true or false), speed and velocity (m/s or km/h), travel time (h), and vehicle type are the most used measures. For traffic flows these comprise traffic volume (vehicles/h or vehicles/s), traffic density (vehicles/km), average travel time (min), and traffic capacity $(\#\text{vehicles} \cdot (\text{km/h}))^2$. In the following section the data sources delivering these and other measures are explained.

2.2 Data Sources

We now highlight different types and dimensions of data sources used in the course of C-ITS applications.

2.2.1 Stationary Detection

Stationary detection mechanisms are the most common and established data sources for gathering traffic data. They are installed permanently in road surfaces, at road sides, are integrated into traffic signaling systems, or are affixed to road infrastructure components, such as street lights. In Germany, one of the most used devices are *induction loops*, which can measure different parameters, such as presence, occupancy time, vehicle type, or speed. We will not go into detail of stationary detection, as we are not that much interested in the stationary devices, but in the data they are measuring. We have summarized the most important stationary detection sensors and methods in Table A.3 in Appendix A.2 along with a brief technical description, and the measured parameters based on Hoyer (2003); Treiber and Kesting (2010).

2.2.2 Mobile Detection

In contrast to the stationary detection of traffic parameters where detectors are installed at fixed positions, in mobile detection sensors and detectors are carried in the vehicles, i.e., the vehicles are themselves moving sensors in the traffic flow. This data can further be categorized into Floating Car Data (FCD) and Floating Phone Data (FPD). For FCD one mobile identifier is connected with one vehicle. Hence, the vehicle itself is connected via cellular networks or via Wireless LAN mostly to a proprietary system provided by the car producer. The FCD system is deeply integrated into the vehicle and has access to the vehicle's sensors. For FPD individual phones belonging to individual people, not vehicles, are registered. In the following we will describe both kinds of data sources and discuss advantages and disadvantages in detail.

Floating Car Data

For FCD manual and automatic detection can roughly be distinguished. In manual detection so called traffic jam detectors, such as employees of the ADAC and private persons, report traffic jams and other traffic events using a mobile phone. However, the manual detection lacks accuracy due to a rating subjectivity (e.g., rating of the traffic state) (Steinauer et al., 2006), and events and all-clears may be reported irregularly. For automatic acquisition of FCD vehicles report first and foremost their measured position and possibly aggregated data to a central unit. Therefore, vehicles, which serve as FCD probes, require four things in general: a positioning facility, a communication facility, sensors to measure parameters of interest, and a processing unit preparing the data for transfer and is able to receive data. The four components of the system can be implemented in different ways, which will be discussed in the remainder of this section.

Communication The transfer of FCD can technically be done in several ways, e.g., via a mobile phone using cellular networks, such as LTE, the navigation system of the vehicle, Wireless LAN, Terrestrial Broadcasting, Satellite Broadcasting, Bluetooth, or a beacon system (Treiber and Kesting, 2010; Huber, 2001a),(ETSI EN 302 665, 2010). Beacons can be infrared senders and receivers with which the vehicle can communicate over a special on-board device. The beacons are installed at roadsides and the vehicles send the measured data to the beacon as they pass it. The beacon system has two disadvantages. First of all the system can only report with a certain delay. Events are not detected until the vehicle passed the last beacon of the respective section, i.e., it can report the event in a section only when it already passed it. This makes it unsuitable for real-time reporting of events. The second disadvantage, the costs of additional infrastructure elements, led to the non-acceptance of the beacon technique (Steinauer et al., 2006).

For Wireless LAN a set of standards has been elaborated for Vehicular Ad-hoc Networks (VANETs) in the US and Europe, which is called WAVE (Wireless Access for Vehicular Environment, IEEE 1609 standard). It subsumes the IEEE 802.11p standard on the physical and MAC layer (Ahmed et al., 2013). WAVE can operate at high data rates of up to 27 MBit/s in a specific frequency band range which is dedicated for VANET applications. It enables V2V as well as V2I applications. VANETs usually are composed of network devices in the vehicle, which can communicate with the in-vehicle infrastructure and the external infrastructure, road side infrastructure, an access network, and a core network (Ahmed et al., 2013). Advantages of VANETs are, amongst others, the easy deployment, use of mature technology, and native support of V2V communication. Disadvantages are scalability issues, unbounded delays, and lack of deterministic QoS guarantees, and the required infrastructure for reliable connections (Araniti et al., 2013).

Cellular networks pose another possibility for communication in C-ITS. A long time cellular networks have not been considered for ITS applications as the bandwidth was limited and latency was unacceptable, especially for safety applications. Nowadays, UMTS and especially LTE changed this situation. Data rates of up to 300 MBit/s can be reached. Furthermore, the existing infrastructure with a high spatial coverage, market penetration, capacity, and acceptance rate make cellular networks attractive for C-ITS (Araniti et al., 2013).

The ITS communication architecture proposed by the ETSI allows to combine several technologies to make advantage of the positive aspects of multiple communication ways and to decrease the negative effects of using them individually (ETSI EN 302 665, 2010).

Message Standards To make C-ITS a success, the penetration rate of equippend vehicles has to be high. Hence, standardization of communication protocols is an important task as vehicles of all vendors and types should be able to communicate. For this purpose multiple message standards have been proposed which will be introduced briefly in the following.

One data communication standard for FCD is GATS, the Global Automotive Telematics Standard, which was invented by Mannesmann Autocom und Tegaron Telematics in the late 90s (Behrens et al., 2010). It is an open industry standard for the communication between service centers of telematics providers and on-board devices, such as navigation systems, which are also called GATS-devices, if they support the GAT standard. It is mainly intended for highways, not for urban areas. In this standard GPS is designated for positioning and cellular networks for communication. For data transfer the SMS (short message service) of the GSM network is used in both directions. The standard defines communication protocols with certain transmission criteria, i.e., a message is only send, if a certain criterion is fulfilled (event-based). Events are already identified in the car, e.g., if the car drives into a traffic disruption, the car recognizes it and sends a message. A two-step process generates the messages. First, the GPS locations and velocities are summarized in a time variation curve. The time variation curve is aggregated to the average travel speed and variance of speed, which is also called the micro profile (Steinauer et al., 2006). Additionally, a driving profile is generated, i.e., important positions on the way are stored and when an event arises, the series of locations (trajectory) is transferred with the event. This chain of locations can enable a more accurate process to map positions to a road network. One disadvantage of the GAT Standard is the data transfer using SMS. Due to its size (140 Byte per message) the transfer rate is not very high. Hence, event messages have to be short or several messages in a row have to be send. Also the delay imposed by sending SMS makes it unsuitable for events which have to be send immediately to other drivers, such as emergency braking events. Furthermore, the GAT-B standard requires maps in the vehicle, which means that map updates may not be done frequently to include new or changed sections and road works, which influences the quality and reliability of gathered FCD (Steinauer et al., 2006).

A very frequently used standard is Alert-C of the RDS-TMC (Radio Data System Traffic Message Channel) (ISO 14819-1, 2013; ISO 14819-2, 2013) which is used by a multitude of information providers, e.g., it is commonly used in navigation systems. The TMC system consists of several subsystems and messages are transferred via radio broadcast (public and commercial providers). The transfer rate of RDS-TMC is relatively low; it just reaches 60 bit/s which is not suitable for time critical applications such as queue-end warnings. Furthermore, the vehicles can only receive messages, but cannot send messages, which make it unusable for V2X applications.

In 2014 ETSI published two standards for messaging in V2X applications: a standard for *Cooperative Awareness Messages* (CAM, (ETSI EN 302 637-2, 2014)) and a standard for *Decentralized Environmental Notification Messages* (DENM, (ETSI EN 302 637-3, 2014)). CAMs are sent out on a regular basis by the individual participants in the ITS. They contain general information about the participant, e.g., the vehicle type and size, and current status information, such as a timestamp, the current position and motion state, and the currently activated systems. Furthermore, CAMs of vehicles can include a specific message container depending on the role of the vehicle, such as a dangerous goods transport. These containers include additional information according to the role. The information in the CAMs can be used for ITS applications in different ways, such as collision warnings or selective geospatial reception of messages (GeoBroadcasting). The dissemination interval, depending on the participant and channel traffic, can vary between 1 Hz and 10 Hz (ETSI EN 302 637-2, 2014). The ETSI standard for CAMs also includes specifications for quality and confidence requirements of the sent information, such as the speed accuracy and the corresponding confidence value for this accuracy. The DENM standard, defined in (ETSI EN 302 637-3, 2014), describes dissemination, reception, and format of traffic event messages. DENM are sent by ITS applications in cases of road hazards or traffic disruptions, or by the ITS participants themselves, e.g., when a vehicle brakes hard. The messages can be exchanged via V2V or V2I communication using the DENM protocol in a relevant geospatial area (e.g., only in downstream direction before a road hazard). Furthermore, the relevancy of the message for each vehicle is checked by the corresponding ITS application. DENM can also be relayed, e.g., to warn approaching vehicles even when the sending vehicles already left the event area. A DENM contains information about the event, such as the event type, position, a timestamp, and the duration. In addition to the general event information, containers for specific event types, such as accidents or roadworks, are defined, which contain event specific data. Additionally, there exist container types for the event location and miscellaneous data. In the DENM protocol messages can report a new event, but also update, cancel, or negate an existing event (ETSI EN 302 637-3, 2014). The CAM and DENM formats have been defined in the standard documents in the Abstract Syntax Notation One (ASN.1) language[8], which is used to describe data structures in a unified way.

Floating Phone Data

Cellular networks have been invented in the early 80s and introduced to end consumers in the 90s. Since then, the coverage and penetration of cellular networks and mobile devices has incredibly risen. In 2014, statistically each German inhabitant had 1.4 SIM cards to connect to a cellular network resulting in a 139% penetration rate and 112.63 million active subscribers (Crowley-Nicol et al., 2015). The two most widespread standards and technologies used today to operate cellular networks are GSM (Global System for Mobile Communication) and UMTS (Universal Mobile Telecommunication System). GSM as a standard of the second mobile telecommunication generation (2G) provides mainly voice connections and the Short Message Service (SMS). Later on, data transmission standards EDGE or GPRS have been added to the GSM standard. The successor of GSM, UMTS being the 3rd generation technology (3G), integrates voice connections as well as data transmission into one standard. With the later extensions HSDPA (High Speed Downlink Package Access) and HSUPA (High Speed Uplink Package Access) the possible data rates have been increased substantially. Finally, the 4th generation standard, LTE (Long Term Evolution), has been set into operation 2010 as UMTS also reached its capacity limits (Sauter, 2011). In 2014 in Germany 13 Mio. of 112,63 Mio. active SIM cards used LTE where LTE reached a population coverage of up to 80%. Due to the high data rates and prevelance of smartphones, the data transfer rates also increased a lot (393 GB of data have been transferred via all cellular networks in 2014 (Crowley-Nicol et al., 2015)).

[8] `http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=x.680`

GSM, UMTS, and LTE have a lot of technical differences whose details will not be explained here. An important aspect all of them have in common is that each network is organized in cells. A *cell* is defined as the area an antenna tower, or so called Node B (UMTS), or base transceiver station (short: base station, GSM), covers. The size of a cell depends, amongst others, on the technology and frequencies used and on the actual and expected number of users. Hence, cells in urban areas are usually smaller than in rural areas. In UMTS, for example, cell sizes can vary between a few hundred meters up to 180 kilometers (Sauter, 2011).

Due to the excellent penetration rate in many countries, cell phones have been found to be very useful as a data source. In this regard, the location of the phones is of particular interest. From the location of a phone its track could be followed through a road network or from an accumulation of phones traffic events could be derived. Hence, the determination of a phone's location, or *positioning*, is detailed in more depth in the following.

There are several ways to determine the position of a mobile device based on the cellular network's properties and technologies. These positioning technologies can be categorized according to various aspects. First, techniques can be categorized into *network-based* and *handset-based* positioning (Izadpanah and Hellinga, 2007), depending on where the measurements and the estimation of the position takes place – in the network infrastructure or on the mobile phone itself. Usually, the handset-based techniques use a GPS receiver for their measurements. Smartphone operating systems, such as Android, Windows, or Apple iOS, and apps such as Waze[9], collect the GPS data of the smartphones anonymously and use them for their traffic services (Jeske, 2013).

Furthermore, techniques can be active or passive (Valerio et al., 2009). In active techniques additional actions are taken to estimate the handset's position. One example is *Enhanced Observed Time Difference* in UMTS networks where Node Bs send a signal to the handset which estimates its location calculating the intersection of the signal hyperbolas. Passive techniques do not require any additional signaling traffic. Already available information from the network's operation is used. In ITS mostly passive techniques have been used as the sole data from network providers can be used without any further effort. Mostly, data from Local Area Update (LAU) and handover events are used, which will be explained in the following as these are the most frequent events in a network (Tettamanti et al., 2012).

A phone that is switched on reports its location every time it enters a new Location Area (LA) comprising several cells. This procedure is called *Location Area Update* (LAU). Given the size of cells this is a very imprecise location method, but it is sufficient for the network to contact the phone when a call or SMS comes in. When the phone is in an active call and moves between two cells, a *handover* procedure is initiated by the current base station to ensure the call is not interrupted. To decide, when a handover is necessary, in GSM networks the handset regularly sends measurement reports including signal strength and quality of the current base station and stations in near vicinity every 480 seconds. In UMTS networks the reports can also be sent event-based (Gundlegard and Karlsson, 2009) and can be sent more frequently. There is also a technical difference between GSM and UMTS in how the handover works. In UMTS *hard handovers* and *soft handovers* are distinguished. While in hard handovers the handset is only connected to one Node B to transmit the call and data, in soft handovers the transmission is carried out over several Node B's. Hence, with hard

[9]`https://www.waze.com`

handovers the connection can shortly be interrupted while this is not the case for soft handovers. Hard handovers are also used in GSM (Sauter, 2011). For positioning the handover type makes a difference - while with hard handovers only one position is determined, in soft handovers each time the handset is connected to a new Node B or disconnected from a Node B a position can be determined. Hence, soft handovers offer more information, but can also add ambiguity and complexity. Experiments by Gundlegard and Karlsson (2009) show that the average error in positioning is lower for UMTS (6.3 meters) than using GSM handovers (23.1 meters) in a suburban area. They have also shown that travel times and speed inferred from UMTS handovers are more accurate than for GSM handovers. In general, the positioning using handovers is more accurate than using LAU events. Traffic parameters are derived by calculating the traversal time, that is the time between two events (Janecek et al., 2015).

The time a cell phone is registered at a base station between two handovers is called *Cell Dwell Time* (CDT) and can also be used to derive road traffic-related parameters such as congestion (Steenbruggen et al., 2011a). The handover event can be detected either by a client software on the handset, e.g., Nokia Mobile Quality Analyzer (Tettamanti et al., 2012), or a handover report is created by the network provider. Hence, the handover data is very often used to derive other information as no additional equipment has to be installed or means have to be taken for positioning. But it also has some drawbacks: the phone has to be in an active call and a call maybe terminated before a handover happens. Furthermore, there may be a time difference between two consecutive calls which does not allow inference about the actual path of the phone through several cells (Caceres et al., 2012). However, there are also approaches which try to track the location from devices in active calls and idle devices to increase the network coverage. For example, Janecek et al. (2015) propose an approach which retrieves the identifier of the cell the handset is connected to in cases of data transfer, SMS, calls, and location area updates.

In general, it is not easy to map phones to vehicles. A phone maybe carried by pedestrians, cyclists, or multiple people in a vehicle or train. This multitude of possibilities could distort derived traffic information. For example would a pedestrian area nearby a road falsify an assumption about the number of vehicles on this road. Furthermore, specific events could also influence the call behaviour, e.g., a peak in SMS messages and calls after 12 am on January 1 or during a football game. From the handover and LAU events single position estimates can be determined or a number of events in a certain area per time (handover or LAU rate, respectively) can be identified. Also, an increase or drop in the number of calls or SMS messages could be related to the change of certain traffic parameters (Steenbruggen et al., 2011b). Furthermore, the observation of the measure Erlang (1 Erlang = 60 minutes of phone use per person) can reveal information about the traffic situation (Steenbruggen et al., 2011a). Basyoni and Talaat (2015) proposed an approach to cluster cell phones of one vehicle using data swarm clustering. Hence, it is possible to identify single vehicles even when they carry more than one handset.

Overall, FPD are a useful source to derive traffic parameters. Parameters which can be derived from FPD comprise vehicle and link speed (Fontaine and Smith, 2007; Gundlegard and Karlsson, 2009)), travel times (Janecek et al., 2015; Wunnava et al., 2007; Gundlegard and Karlsson, 2009), Origin-Destination (OD) matrices (Caceres et al., 2008), traffic volume, density, and flow (Valerio et al., 2009), incident detection (Steenbruggen et al., 2011b), traffic state (Pattara-atikom et al., 2007), vehicle type (Basyoni and Talaat, 2015) and mobility patterns.

2.2.3 Additional Data Sources

Besides data acquired by the mentioned stationary and mobile traffic detection mechanisms, other data sources are of high value for ITS applications. These include weather data from weather stations or corresponding weather services. Furthermore, data about roadworks, incidents, and speed checks can be useful. This could either be retrieved from public institutions, such as the police, or reported by private persons (e.g., as implemented in Waze[10]). Also archived historical traffic data can be very useful, e.g., speed profiles, to predict future traffic states. Finally, raw or processed traffic data is also offered by many free and commercial services and marketplaces. For example, the German Federal Highway Research Institute (BaST) is maintaining the Mobility Data Marketplace (MDM)[11] where round about 500 data providers and 250 data consumers are registered. Also other companies, such as PTV in collaboration with TomTom[12] offer archived and live traffic data.

2.3 Traffic Modeling

Traffic is a complex ecosystem which is not easily understood and reproduced in a realistic way. Many mathematical models have been proposed to describe movements of individual vehicles, traffic flows, and driver behaviour in traffic. To emulate traffic, especially "the density-flow relation, the spontaneous formation of jams and the spatio-temporal evolution of such jams" have to be modeled in a realistic manner (Schreckenberg et al., 2001). *Fundamental diagrams* are mostly used to compare simulations with reality. They depict the relationship between traffic flow and traffic density and velocity and density, respectively. Fundamental diagrams are used to expose important characteristics of a traffic situation. In this section, we will briefly introduce the most important ones. These models are the basis for simulation software used in traffic research and planning.

Traffic models can be mainly categorized by their resolution detail, namely, into microscopic and macroscopic models. *Microscopic models* emulate the behaviour of individual vehicles, while *macroscopic models* are based on flows of traffic. Microscopic models offer a higher variety of possibilities, as many traffic parameters are directly accessible (Mazur et al., 2005). Additionally, macroscopic dimensions can also be derived from microscopic models by aggregation for clearly defined spatial extensions (e.g., a link or a section). However, microscopic models require a high processing effort in simulation, but allow for real-time analysis of traffic (Treiber and Kesting, 2010). *Mesoscopic models* combine the former models, e.g., when macroscopic measures influence a microscopic model.

2.3.1 Microscopic Models

For microscopic models different mathematical approaches are used. The most common methods are *car-following models* or *follow-the-leader models*, which are based on coupled ordinary differential equations with time as continuous independent variable (Treiber and Kesting, 2010). Based on the vehicle's own velocity and the distance to the vehicle ahead and its velocity, the velocity and position is adapted accordingly (acceleration function). An example is the Optimal Velocity Model (Bando et al., 1998),

[10]`http://www.waze.com`
[11]`http://www.mdm-portal.de/en`
[12]`http://vision-traffic.ptvgroup.com/de/produkte/traffic-data/live-daten/`

where the vehicles try to reach an optimal velocity, compared to the vehicle ahead, by adapting their velocity by acceleration or deceleration and taking into account the drivers response delay and the sensitivity of the driver by a weighting factor. Other common models are the Intelligent Driver Model (IDM) (Treiber et al., 2000) or the car-following model by Newell (Newell, 1961).

A subcategory of car-following models are *cellular automata*. While the former models are continuous at least in time but can also be continuous in space, cellular automata (Schreckenberg et al., 2001; Nagel and Schreckenberg, 1992) subdivide the road into cells of a fixed size where a cell is either empty or occupied by one vehicle with a discrete velocity (measured in cells per time step). Furthermore, the model advances in time steps, i.e., space and time are modelled discrete. The use of discrete variables makes the models computationally less expensive. All vehicles behave in each time step using four consecutive steps: (1) the vehicle increases its velocity v by 1 while keeping a certain distance of cells to the next vehicle and not exceeding a maximum velocity, (2) the vehicle decelerates, if the distance to the next vehicle gets to low and adapts its speed to the velocity of the vehicle ahead, (3) the velocity v of the vehicle is reduced by one with probability p (simulates unforseeable human behaviour), (4) the vehicle moves forward v cells. The first approaches are for single lanes, but there exist also multi-lane approaches (Nagel et al., 1998). The influential Nagel-Schreckenberg cellular automaton approach has been extended in several approaches to include more realistic traffic behaviour. An important extension has been made by Kerner et al. (2002) emulating the behaviour described in the three-phase traffic theory (Kerner, 2004). Recent models are all based on one of these two models and extend them in various ways.

Other driver models (used for example in the simulation software VISSIM[13]) are psycho-physical car-following models based on Wiedemann (Wiedemann, 1974) consisting of two components. The psychological component is integrating the speed and distance the driver desires. The physical component implements perception thresholds and deficient vehicle control. Four states depending on the distance to the car ahead and on the velocity are distinguished: free driving, approaching, following, and hazard. The model can be intensively parameterized, e.g., deviations in acceleration and distance, maximum acceleration, and distance to vehicle ahead in standstill. Also the probability of a temporary attention drop of drivers or a sudden distortion can be configured (Leonhardt, 2012).

Additionally, for complex simulations with multiple lanes, *lane changing behaviour models* also have to be defined. Lane changing is usually modelled with multiple steps, consisting of a tactical phase where the driver considers and prepares the lane change by deceleration or acceleration, and the operational phase, where the driver rates the possibility of the immediate change and decides, and the realization phase, where the driver executes the change (Kesting et al., 2007; Treiber and Kesting, 2010), if possible. Lane changing models are discrete decision models, which underlie defined rules mimicking driver decisions. The rules and parameters used vary depending on the model. There exist several methods to approach this decision process from simple implementations of the rules as process flows (Gipps, 1986) to complex agent-based systems (Hidas, 2002). Furthermore, lateral driving behaviour inside a single lane can be modelled, e.g., when bicycles overtake other road users.[13]

[13]`http://vision-traffic.ptvgroup.com/de/produkte/ptv-vissim`

2.3.2 Macroscopic Models

Macroscopic models emulate traffic as a flow comparable with fluids or gases. Hence, many approaches are derived from fluid dynamics (Treiber and Kesting, 2010; Mazur et al., 2005). Macroscopic approaches model traffic as the "continuous time and space evolution of the density and mean velocity of the flow of vehicles" (Bellomo and Dogbe, 2011). They use partial differential equations with time and space as continuous independent variables (Treiber and Kesting, 2010). These models are often based on static models, such as the fundamental diagram, to define certain relationships of parameters. In general, static models describe "the pairwise relationship between the measures density, flow, velocity, or occupancy."(Treiber and Kesting, 2010). Further models which can be used for macroscopic modeling are cellular automata or coupled iterated maps where time is modeled in a discrete manner (Treiber and Kesting, 2010). We do not go into much detail about macroscopic models, as we only use microscopic models in this work. A comprehensive survey on macroscopic models can be found in (Bellomo and Dogbe, 2011).

2.4 Traffic Applications

After we have discussed the most important fundamentals of transportation, in this section we describe common applications of C-ITS. According to the ETSI, traffic applications can be categorized into road safety applications, traffic efficiency applications, and other applications (ETSI TS 102 637-1, 2010). Examples for *road safety applications* are notifications in cases of an emergency vehicle approaching, emergency braking, wrong way driving vehicle, certain traffic conditions (queue-end, traffic jam), adverse weather conditions, roadworks, or human presence on the road (ETSI TS 101 539-1, 2013; ETSI TS 101 539-3, 2013). Depending on the application, notifications can be triggered either automatically based on sensor readings and algorithmic processing, or manually. Examples for traffic *efficiency applications* are notifications regarding traffic information, speed limit notification, driving assistance, or cooperative navigation (ETSI TS 102 637-1, 2010). Other services may comprise points of interest, parking management and automatic access, electronic commerce, fleet management, or loading zone management (ETSI TS 102 637-1, 2010).

All of these applications have different requirements regarding data and functionality. Hence, for each of the application a requirements analysis must be done to determine which detection methods and parameters are useful for the application.

The applications used for examples and case studies throughout this work, namely Traffic State Estimation and Queue-end Detection and corresponding existing approaches, will be described in more detail in the following.

2.4.1 Traffic State Estimation

Traffic state estimation (TSE) or also termed *traffic state detection* or *traffic state analysis* derives the current traffic status of a road section for a certain point in time. The process is an estimation, as the traffic state cannot be measured directly, but has to be derived from measured and observed traffic data (Van Hinsbergen et al., 2012). The result of the traffic state estimation is a complete traffic state description which reflects the current traffic status by estimated traffic flow variables, which can exceed the number of measure parameters (Wang et al., 2007). The tempo-spatial traffic state description is the basis for the creation of a traffic state report and also crucial for traffic forecasting and network-based data fusion and control (Hoyer, 2003). There are different ways to describe a traffic state (cf. Section 2.1.2). The description as well as the estimation techniques may differ depending on the context. Mainly approaches for urban and non-urban traffic are distinguished, but due to the increase of mobile data sources and probes (cf. Section 2.2.2) also systems for all road types exist. There is a multitude of free and commercial services which offer traffic state reports (Dirscherl, 2016). Especially, real-time reports are of high interest to users today. Navigation systems (in-vehicle or on mobile devices) often already integrate traffic state information into the routing information. Also other services outside of navigation systems combine maps, navigation, and traffic state reports. Popular services are Google Maps/Traffic, Waze, Nokia Here, TomTom Traffic, or Bing Maps. Most of these use mutliple data sources from static and mobile detection. For the algorithmic part of traffic state estimation different approaches have been proposed which are explained in the further.

Threshold Methods

Threshold methods are the simplest form of traffic state estimation. In general, the input of such a system are values measured for example by successive gauging sections (stationary detectors) on highways. The measured and aggregated values are then compared with certain thresholds to identify the traffic state. For each gauging section with an inductive loop a threshold of 50 km/h is defined. If the speed of a single car falls below this value, the section from the last exit to the next exit is assumed to be congested. However, the congestion is only reported when the measured values constantly lie below this threshold for ten minutes. The method can only distinguish two traffic states and can only report the congestion when it is really observed by a detector (ddg, 2000).

One example for a more sophisticated threshold system is the VKDIFF approach. For each gauging section i a dimensionless value VK is calculated by the following equation:

$$\mathrm{VK}(i,t) = [(\frac{V_{\mathrm{free}} - V_m(i,t)}{V_{\mathrm{free}}})^2 + (\frac{k(i,t)}{k_{\mathrm{max}}})^2]^{\frac{1}{2}} \tag{2.1}$$

where $V(i,t)$ is the velocity at a time t and the gauging section i, $k(i,t)$ is a measure for the traffic density (vehicles per kilometer) which has to be estimated from the data. V_{free} and k_{max} are constants. VK can be said to be the deviation of the average velocity from the free flow velocity and the ratio of the current density and the maximum density. To estimate the traffic state the $\mathrm{VK}_{\mathrm{DIFF}}$ measure is calculated using two successive gauging sections which are subtracted from one another.

If the $\mathrm{VK}_{\mathrm{DIFF}}$ value is high, the two VK values have an opposing development, which indicates a hazard. For the $\mathrm{VK}_{\mathrm{DIFF}}$ value different thresholds exist indicating the traffic state. According to Hoyer (2003) this method can be used for lane control systems in traffic control centers and sub control units.

Fuzzy Logic

A drawback of simple threshold methods is the low number of states which are detectable and more sophisticated threshold methods have been found some of which utilize fuzzy logic. Fuzzy logic has been widely used for many different traffic applications. With fuzzy logic the thresholds are not fixed at a distinct value, but described by fuzzy sets. For each fuzzy set a membership function assigns an affiliation value to each input element indicating how much a value belongs to the fuzzy set. This allows for a more flexible and sophisticate description of thresholds and traffic states, respectively. To implement the fuzzy logic an inference machine is used which represents a decision process. Components of the inference machine are rules consisting of logical operators. Result of the fuzzyfication process is a distinct decision (this process is also called defuzzyfication) which delivers a concrete value for the output parameter, e.g. the traffic state.

A proposal invented at Siemens uses the $\mathrm{VK}_{\mathrm{DIFF}}$ and two trend factors as input to a fuzzy system (Hellendoorn and Baudrexl, 1995). The rules employed are based on a dynamic fundamental diagram and the system takes into consideration several impact factors as fuzzy variables, such as the time of day, the weather, or the season. A famous approach used in the XFCD system by BMW is based on fuzzy logic (Huber, 2001b). Huber (2001b) identified important traffic parameters which indicate a congestion. The first parameter is a delay parameter which interprets a strong decrease of velocity in a short time period. The second parameter is an indicator which can rate a slow velocity profile, i.e., it can detect if the vehicle is driving slowly over a certain period of time (similar to the threshold method). Additionally, flash lights are observed as they are a strong indicator for a vehicle driving towards the rear end of a traffic jam (Huber, 2001b). Furthermore, the state of the right direction indicator is also included in the decision process, as it may indicate that the vehicle is turning off the road. The processing is done inside the vehicle, such that the vehicle can report only the event and does not have to send all of the detailed data to a server to do external processing, saving communication costs. A different approach which uses fuzzy logic to estimate the traffic state is described in (ddg, 2000). Here traffic data is first summarized to parameter vectors which are intended to describe the traffic state. The traffic parameters include velocity, traffic flow, and density. The method uses a data stream approach where the incoming records represent a stream of the parameter vectors. A window is defined which only contains the parameter vectors of a certain time period (e.g., one minute) and of a certain location interval. The measured values are therefore obtained as a moving time function (ddg, 2000). Finally, the parameter vectors are used as an input for a fuzzy logic system which classifies the parameter vectors into the different traffic states using a membership function for each dimension of the parameter vector. For each parameter different results for the single states are output. These results can be summed up and standardized to retrieve the overall traffic state result for each single location. To get the traffic state for a road section the location vectors with the traffic states are extrapolated with a Gaussian filter over the location dimension.

Fuzzy sets are also used in combination with other approaches, e.g., in the FOTO system by Kerner (2004).

Section-based Balancing

In these methods, at the beginning and the end of a section with known length velocity and traffic flows are measured (ddg, 2000). The measured parameters are balanced between start and end point of the section. If at the end of the section the traffic flow decreases but at the beginning the same amount of vehicles flows in, a congestion in the section can be assumed. The congestion can be detected before it reaches the second detector at the end of the section.

Kalman Filtering

Kalman Filters are algorithms for producing an optimal estimate for linear dynamic physical processes and systems. Physical parameters are estimated to approximate the true state of the linear system at a time k. It is assumed that the system is observed by measurements. These are described by an observation model or function which also includes measurement noise (Gaussian with zero mean). The system itself is described by a state transition model/function. Additionally, it accepts input modelled by a control function and finally, noise is added to reflect process and system errors. The result is a vector representing the system state at a time k based on the previous state at time $k - 1$. Extended Kalman Filters expand this idea to cover non-linear dynamic systems by modeling observation and transition by differentiable instead of linear functions. Extended Kalman Filtering is the most popular method for traffic state estimation (Van Hinsbergen et al., 2012; Wang and Papageorgiou, 2005). In traffic state estimation Extended Kalman Filters are used in combination with stochastic state models for macroscopic modelling of the traffic flow. It calculates the output values based on the measured input values and the macroscopic model parameters. The output parameters are continuously compared to the real measured values and the model parameters are adapted. The output parameter is the section-related profile of the traffic state parameters (Hoyer, 2003). For example, a popular approach by Wang and Papageorgiou (2005) uses a spatio-temporally discretized macroscopic flow model to model the traffic on a highway. It utilizes aggregated variables for traffic density, space mean speed, and traffic flow and is described in terms of differential equations. Errors are represented by zero-mean Gaussian white noise. The measurements are represented by modeling detectors for traffic flow and mean speed and also include zero-mean Gaussian white noise. The Extended Kalman Filter is then represented by an equation which includes the application of the above macroscopic model applying corrections based on the measurement model. Due to its popularity new approaches for Extended Kalman Filtering in TSE are proposed once in a while, e.g., by Van Hinsbergen et al. (2012). Other groups of algorithms where TSE is regarded as a filtering problem and which try to outperform Kalman Filters are particle filters (Wu et al., 2015; Bi et al., 2013) or Probability Hypothesis Density (PHD) filters (Canaud et al., 2013).

Cellular Automaton

The cellular automaton method is an approach which models dynamic systems solely with discrete integer variables (Treiber and Kesting, 2010). For an explanation, please see Section 2.3.1. The traffic state can then be derived from the model by feeding it continuously with measured data and observing the resulting state. This approach can be used in urban and extra-urban areas and has been successfully deployed in the autobahn.NRW project by the University of Duisburg now called Verkehr.NRW.[14]

Fundamental Diagrams

In the MARZ description (BASt, 1999) fundamental diagrams are used to determine the traffic state. Fundamental diagrams can describe the state of traffic flows (Kuhne, 2011; Schnabel and Lohse, 1997) by showing the relationship between traffic flow, velocity, and density. In (BASt, 1999) the assignment of the data from the fundamental diagram to corresponding traffic states is done by using decision tables dependent on the number

[14] `http://www.verkehr.nrw.de`

of lanes, velocity, and density from smoothed 1-minute values (Hoyer, 2003). In this model four different states are used to describe the traffic state: free, dense, tough-flowing and congested traffic. In the City-FCD approach (Steinauer et al., 2006), also fundamental diagrams are used. Thresholds in the fundamental diagram which have been derived from simulations determine the traffic state ranges. For highways four states have been used (free flow, dense traffic, tough-flowing traffic, traffic jam). The thresholds are dependent on the link considered and expectation values defined for it. The states are furthermore not strict, but can overlap (Steinauer et al., 2006). For rural roads City-FCD distinguishes three states (capacity available, no capacities, traffic jam) and on urban streets only “no traffic jam” and “traffic jam” are distinguished.

Three Phase Theory

The three phase theory is an empirical model by Kerner (2004). This approach distinguishes three traffic phases on extra urban roads: free flow, wide moving jam, and synchronized flow. The approach assumes that traffic can be described by spatio-temporal patterns as traffic is always related to location and time. Therefore, according to Kerner (2004) a traffic phase is a traffic state considered in space and time which can be described by spatio-temporal features which are only specific to that traffic phase. In general, congested traffic is seen as complementary to free flow and the average vehicle speed is lower than the minimum possible speed in free flow. A traffic jam is bordered by two fronts, the upstream jam front and the downstream front. The wide moving jam phase describes a traffic jam which ”maintains the mean velocity of the downstream front of the jam as the jam propagates” (Kerner, 2004), i.e., the velocity of the downstream front is not influenced by the state of the following jam zone. The vehicles in the upstream front only drive at a very low speed or halt, while the vehicles in the downstream front always accelerate almost from halt (stop-and-go). Therefore, the jam and its fronts move along the road. Additionally, the traffic volume leaving the traffic is independent from the traffic volume driving into the traffic jam, the jam is only getting longer. In the synchronized flow phase the downstream front is often fixed in a bottleneck (Kerner, 2004). Hence, the downstream front separates the synchronized flow upstream from free flow downstream. In this phase the traffic volume leaving the traffic jam is dependent from the vehicles entering the congestion.

Kerner (2004) developed two models, the ASDA (Automatische Staudynamikanalyse, Engl: Automatic Tracking of Moving Jams) model and the FOTO (Forecasting of Traffic Objects) model based on the three phase theory, which automatically detect and track congested spatio-temporal traffic patterns on freeways. The FOTO model is used to identify the traffic phases and to track synchronized traffic, while the ASDA model tracks the moving jam propagation. In the models traffic is described by means of macroscopic models. Each object, which can be a wide moving jam or a synchronized traffic flow, has properties such as the object width, locations of the upstream and downstream front, velocities of the upstream and downstream fronts, and an object lifetime. The creation of the objects is done by the FOTO model, which firstly identifies the traffic phases based on measurements of two successive detectors locally. It uses Fuzzy Logic to interpret the measured parameters and determine the traffic state. As input parameters mainly the traffic flow rate and the vehicle’s speed are taken. The values for the membership functions have been determined empirically with measured data. Secondly, both models recognize the moving fronts of the phases at the gauging sections. After front detection the properties for the size and position of the corresponding traffic objects can be derived. Finally, the models track the identified objects. With the models it is possible to predict the development of the traffic jams, e.g., when two wide moving jams or two synchronized flows are merging (a mixture is not possible) even when the jams are not directly observed. So not only TSE but also traffic flow prediction can be done.

Traffic State Estimation using Floating Phone Data

As described in Section 2.2.2 for FPD either the GPS readings are transferred or the phones are detected in the cellular network. Hence, the parameters used in TSE differ accordingly. While GPS readings might also submit speed values (Habtie et al., 2015), in the cellular network other parameters such as the number of active calls (Caceres et al., 2012), the Cell Dwell Time (Pattara-atikom et al., 2007), or handoff events (Alessandri et al., 2003) on the link are used to derive the traffic volume.

Systems are proposed which utilize the Cell Dwell Time in cellular networks to estimate the traffic state on urban roads (Pattara-atikom and Peachavanish, 2007; Pattara-atikom et al., 2007). The authors assume that high Cell Dwell Times are an indicator for traffic hazards. To acquire the Cell Dwell Time on mobile phones a software had been installed on the phones which collects the data. Besides of the measured Cell Dwell Time, date and time, the MCC, the MNC, the LAC and the cell ID are collected in idle and active mode of the phone each time a handoff takes place. The collected data are then send to a server and classified by firstly smoothing the gathered data by averaging the data with historical values and the moving average technique. In the next step the data are classified into three different congestion states (free-flow traffic, moderate traffic and heavily congested traffic) by using a three level feedforward backpropagation neural network. Input for the neural network are the time, the LAC, the cell ID and the Cell Dwell Time as vectors. The output is the level of congestion. The LAC and cell ID have been used along with the CDT to identify the geographic location of the cell and in conclusion determine the size of the cell. The neural network can then learn the comparative cell size from the relationship between CDTs and observed congestion levels. In the second system by Pattara-atikom et al. (2007) the same data collection and smoothing method are used. In addition to the moving average smoothing also a weighted exponential average method is used. For classification into the different traffic states (which are also the same) the authors use a simple threshold technique and fuzzy logic instead of the neural network. For the simple threshold technique empirical thresholds for the average CDT at a time t are defined for each traffic state. The membership functions for the fuzzy logic system are defined similarly.

Another approach utilizing data from mobile phones uses a microscopic traffic model that allows for the representation of active mobile phones in freeway traffic (Alessandri et al., 2003). In the model a freeway is partitioned in sections, defined by the cell sizes of the cellular network. The approach was tested by using a microscopic freeway simulator and a cellular network simulator which simulates mobile phone activity and tracks handoffs to provide the density of active mobile phones in a cell and flows from one cell to another. It is assumed that the distribution of cars along a freeway is uniform. The traffic simulator feeds data (traffic flows of vehicles entering the section and traffic flows leaving a specific section) into a traffic predictor which is implemented with the macroscopic model to estimate traffic density and velocity for comparison with the Extended Kalman Filter. The vehicle positions generated by the macroscopic model are then passed to the cellular network simulator. Furthermore, the data of the traffic simulator is fed into the Extended Kalman Filter which uses the macroscopic model and a dynamic model of the active mobile phones. The Extended Kalman Filter estimates for each section the following state variables by using the mobile phone densities and the handoffs for the corresponding cells: density of vehicles, mean velocity, and percentage of active mobile phones. The simulation setup does not implement a specific cellular network technology. The estimated values are compared with the mean velocity and density which are output by the traffic simulator. The authors obtained good results with the Extended Kalman Filter and a constant rate of 5% active cell phones. Velocity is estimated with a maximum error of 5 km/h while density differed up to a maximum of 100 veh/km.

A recent approach by Habtie et al. (2015) uses positions, speeds, and timestamps of GPS measurements of handsets recorded every second. About 450 probes are emulated by the SUMO traffic simulation. The problem of unwanted probes (such as peasants) is assumed to be solved by subscription - only handsets subscribed to the service are taken into account. The speeds are aggregated using a window of ten minutes. An artificial neural network is used to predict the average speed on a link. Finally, the traffic state estimation is done using a simple threshold method and three traffic states.

2.4.2 Queue-end Detection (QED)

Queue-end or rear-end collision detection and warning systems aim at the prevention of this highly dangerous hazard. One specific challenge of these systems is the very low processing delay allowed. While QED was a difficult task using stationary sensors, the advent of in-vehicular sensor networks and V2X communication opens up new opportunities to prevent rear-end collisions. In the following we discuss some solutions to the problem.

Huber (2001b) studied the opportunities in traffic information collection using XFCD. He analyzed which in-vehicle information, e.g., collected over CAN bus and sensors, can be used to detect certain local traffic events. One example he investigated is the queue-end detection using a deceleration parameter, an indicator for low speed level, right turn signals, the road type, and warning flashers. The event determination is implemented using fuzzification and a rule base. To reach 90% probability that a vehicle arrives at an incident in a one minute interval a penetration rate of 6.9% is required already. The high penetration rate is not really suitable for local real-time hazard warnings. The system does not use any learning for fuzzification and rules – these are manually defined. Chan et al. (2003) present a system for real-time queue tracking based on the average speed detected on road sections by identifying three traffic zones and analyzing the arrangement of these zones. Though, the authors base their work on stationary detection and have to rely on larger sections (minimum of 500 m between loop detectors on a comprehensively equipped highway), the idea of identifying the parts of the queue seems appealing. Other works on queue-end detection use methods from the field of artificial intelligence. Khan (2007) presents a simulation study using stationary sensors as data source. They analyze the data using artificial neural network models, predicting the current queue length based on accumulated numbers of cars and trucks at fixed locations. Although they postulate real-time processing in their information system, the used algorithm is not capable of online learning, i.e., it cannot adapt to concept changes. As mentioned earlier the introduction of V2V opens up new opportunities to prevent rear-end collisions. Early warnings can be send to vehicles to make them aware of the situation ahead. Usually, sensor readings, for example from emergency brake lights or radar, are used to indicate a hazard. A warning is then issued by the corresponding vehicle. An approach which is based on V2V communication and in-vehicle sensors is presented by Chen et al. (2011a). This approach assumes that each participating vehicle is equipped with a GPS receiver and a distance sensor in the front, which enables the vehicle to locate its own position, measure the distance to the vehicle in front of it, and the speed of the vehicle ahead. A warning message is propagated in a warning group of vehicles when an emergency braking event took place. The group is built based on distances between the vehicles.

2.4.3 Data Stream Applications in ITS

Data stream management systems and data stream mining algorithms have been studied intensively, and some approaches apply these techniques to traffic management applications. In the product MineFleet (Kargupta et al., 2010) data stream mining is integrated into vehicle embedded systems for fleet monitoring to analyze vehicle health, emissions, or driver behavior. Another approach detecting driver behavior is presented by Horovitz et al. (2007), which uses a combined approach of unsupervised data stream clustering and fuzzy logic to detect drunken driver behavior. Liu et al. (2006) propose a distributed traffic stream mining system for the determination of congestion level using Frequent Episode Mining. On a central server frequent patterns are determined based on historical data and are then distributed to stationary detectors, which use the pattern to classify data from the sensors. Linear Road (Arasu et al., 2004b) is a well known benchmark in the area of DSMS, but it focuses on the performance of the DSMS and not on the characteristics of the traffic application. A contest at the International Conference on Data Mining (ICDM) in 2010 (Wojnarski et al., 2010) evaluated the quality of data mining algorithms for traffic applications. However, the challenge focused only urban traffic and did not take into account different parameters of the DSMS or the traffic application. Biem et al. (2010) implemented a data stream application for travel time and shortest path estimations using GPS probes from taxis and trucks in the city of Stockholm. A recent approach combines data stream management, complex event processing, and crowd sourcing for traffic state estimation (Artikis et al., 2014). The authors use the STREAM system, the Event Calculus System for CEP, and a custom model and implementation for crowd sourcing suitable for a data stream setting. Kuka et al. (2013) present a system for context modeling for autonomous vehicles where sensor readings of static and mobile sensors are fused in the DSMS Odysseus for moving object prediction. Quality also plays a crucial role here and is embedded into Odysseus. The system is implemented along the JDL Data Fusion Process model (Hall and Llinas, 1997). Only the last approach by (Kuka et al., 2013) is concerned about data quality and evaluation, while all others lack structured evaluation and quality assessment.

2.5 Conclusion

In this chapter we discussed one example application domain for this work. Traffic data for C-ITS applications may be produced in real-time while the data of many users has to be processed. Real-time applications, such as real-time traffic states and incident warnings are highly demanded by road users. Hence, the applications have to produce results in (near) real-time as well. In particular, safety applications require low latency, fast processing, and high quality results. As depicted in this chapter, data sources vary in granularity, availability, and quality. The presented traffic applications rely heavily on unreliable data sources, such as sensors. Further, many parameters influence the results of a traffic application, such as the data source types, the density of probes, or the weather and road conditions.

There is a huge variety of algorithms utilized for the implementation of these applications. To develop effective traffic applications, a thorough evaluation of the techniques employed is necessary. The output quality of the traffic application is not only dependent on the choice of the DSMS and of the processing algorithm; there are many parameters that have to be considered to achieve a certain quality level in the derived traffic information. Recurring general questions, such as how much data is required to produce reliable application results, or which data sources are beneficial for the results, need to be tackled in a structured way. But also domain specific problems, such as real-time Map Matching or parameterization of data stream mining algorithms, have to be evaluated. However, a structured approach to test and find the optimal parameters and algorithms for real-time C-ITS applications is missing. Furthermore, a general and flexible way to measure and rate the quality of the application results is lacking.

We address these issues in the next part by providing a process model for the structured design and evaluation of stream-based applications in Chapter 5. The model is applied on two C-ITS case studies in Chapter 6 comprising a structured preliminary evaluation. A detailed data quality assessment for the traffic applications is carried out in Chapter 8 using the proposed data quality framework. Also, a new online Map Matching algorithm is evaluated in Chapter 9 using the proposed methods, followed by the detailed evaluation of different data stream algorithms in the two case studies in Chapter 10.

Chapter 3

Application Domain II: Mobile Health Systems

Mobile Health, short mHealth, as the second application field for our research has many interesting aspects and properties, which demand for quality-oriented real-time processing. Mobility in the health domain is in particular an expression of the desire for independence, autonomy, and self-control. Wireless technologies, specifically powerful smartphones, have pushed forward the mobility aspect and opened up new opportunities for applications. Especially, in the health domain much of the data is produced at high rates by unreliable data sources while at the same time high reliability and data quality are necessary. Typical problems are sensor misplacement or distortion, in-subject variability, signal distortion by the mobile communication unit, or intentional data quality degradation for performance improvement (Kumar et al., 2013).

mHealth is currently an intensively discussed and researched topic. With the advent and the further development of mobile devices, the publications about mHealth applications also increased rapidly (Fiordelli et al., 2013). Many countries in the world are actively conducting projects or offering services in the area of mHealth (Global Observatory for eHealth, 2011). According to the World Health Organization (WHO) mHealth is defined as "medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices".(Global Observatory for eHealth, 2011) The European Commission put special emphasis on mHealth since 2014: they published a green paper on the topic (European Comission, 2014), initiated a public consultation to elicitate the ideas and issues of the public according mHealth applications, and published a legal guideline for "developers of lifestyle and well-being apps" (European Comission, 2015).

The distinction to other fields, such as Electronic Health (eHealth), telemedicine, and telehealth is not easy (Bashshur et al., 2011). The area of mHealth can be seen as a subarea of eHealth. Telehealth and telemedicine differ from mHealth in that they particularly focus on the remote support or care by health professionals. mHealth applications do not necessarily have that aspect and focus more on the mobility of the user, which is often not a property of telehealth applications (Pankratov and Znamenskaya, 2014). Special advantages of mHealth in contrast to the other fields mentioned before comprise the high coverage, reachability, and penetration rate. This is due to the ubiquitous use of mobile devices, cellular networks, and Wireless LAN, which is especially helpful in developing countries and at locations, where many people live at the countryside. In these cases access to cellular networks is available, but a fixed phone network does not exist (Global Observatory for eHealth, 2011; Economist Intelligence

Unit, 2012; Bashshur et al., 2011). Furthermore, mobile devices are personalized, users are skilled in the use of the devices, and no or minimal additional training to handle the device is needed for application users. Hence, the initial conditions for user acceptance are promisingly good. The mobile aspect as such opens up many more opportunities to realize applications in contrast to desktop applications.

Besides of eHealth, telemedicine, and telehealth, a common term in the context of mHealth is *Pervasive Health*. It is defined as "healthcare to anyone, anytime, and anywhere by removing locational, time and other restraints while increasing both the coverage and the quality of healthcare" (Varshney, 2007). While the definition and term of Pervasive Health is describing somehow the goal of the applications, mHealth is more focused on the technical implementation. But in principle the architecture of pervasive health applications (Varshney, 2007) is almost identical with the mHealth architecture described in the following. Pervasive Health applications and mHealth applications are often the same or similar (Pankratov and Znamenskaya, 2014).

3.1 General mHealth Architecture

The general architecture of a mobile health application, as depicted in Figure 3.1, consists of a set of mobile and/or stationary sensors (optional), a mobile processing device with internal sensors, a communication network, and optionally a server processing complex and resource-intensive tasks. Additionally, other external data sources, such as weather data or archived data, can be integrated with the health data to make the output richer or to infer new information. Depending on the application's complexity more or less of these components are required. There are simple SMS-based applications, which are already considered as mHealth applications. An example is an SMS-based prenatal care application with which women get important information and advices via SMS at certain times in their pregnancy (Global Observatory for eHealth, 2011). But there also complex applications, which include many sites, sensors, and algorithms, for example as elaborated in the HealthNet running application (Quix et al., 2013) described in Section 7.1.

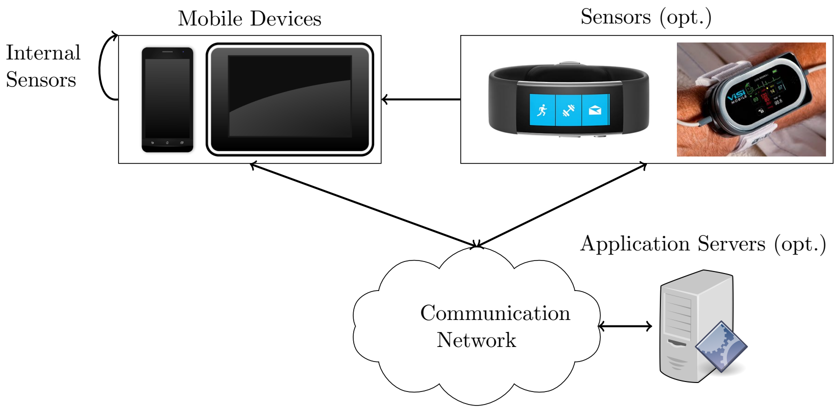


Figure 3.1: General Architecture of mHealth Systems (sensor images taken from Microsoft[3]and San Diego Union Tribune)[4]

Sensors as an optional component in mHealth architectures got ubiquitous in our daily lives. Smartphones and other cellular phones include a lot of internal sensors, such as gyroscopes, GPS, cameras, microphones, light, and temperature sensors. But also the rest of our environment gets more and more populated by sensors: smart homes monitor and control your personal space, sensors in cars make driving easier, safer, and more comfortable, and environmental sensors give us information about air pollution, weather, or animal trails. The Internet of Things (IoT) utilizes sensors for the idea of making items from our daily life smart and connected, such that new service and safety applications can be implemented in a user-friendly and unobtrusive way.

In particular, mHealth already has proved to be very helpful in personal and professional health care. Prime examples are systems for patients suffering from chronic diseases. Many of them require regular condition monitoring and instructions for actions in critical or atypical situations. These use cases can be assisted by mHealth systems. Another typical example is fitness applications, which monitor and advise laymen and athletes during their workout. Mobile fitness and activity trackers have gained huge popularity due to the ability to couple them with a smartphone and to analyze the data.[5]

Applications for chronic diseases and fitness management are also the ones, which are mostly targeted by application developers (research2guidance, 2014; Fiordelli et al., 2013). Furthermore, mobile systems for the assistance of elderly in their daily life help to keep them independent as long as possible. This is particularly interesting in the light of the current staff shortage of certified personnel in elderly care in Germany. This topic is also addressed by research in the area of Ambient Assisted Living (AAL).

3.2 Properties of mHealth Applications

For data management mHealth application scenarios also pose many challenges, as a lot of data can be created by the utilized sensors and devices, the processing power on the mobile devices may be limited, or processing has to be distributed. Disruptions in Internet and cellular network connections as well as failing sensors have to be taken into consideration in the data management and system development. Further challenges relate to data privacy and security.

The amount of mHealth applications already commercially available is overwhelming. In 2014 the biggest app stores provided more than 100.000 mHealth apps (research2guidance, 2014).

There are several criteria, which help to categorize mHealth applications, some of which are:

- **Application Context**: The environment for which the application is conceptualized. There exist systems for personal home care and personal fitness, doctors' practices, hospitals, health care centers, and emergency units. The systems may be designed for indoor or outdoor usage, or both.

- **Type of Mobile Devices**: We can distinguish off-the-shelf devices and proprietary devices. The type of devices can be either off-the-shelf cellular phones or

[3] `https://www.microsoft.com/microsoft-band/en-us`

[4] `http://www.sandiegouniontribune.com/business/biotech/sdut-soteras-visi-mobile-vital-sign-tracker-okd-2012aug21-story.html`

[5] `http://www.livescience.com/42144-activity-monitors-popularity.html`

tablets, wearables including smart watches, or other mobile proprietary devices, such as mobile monitors.

- **Sensor Types**: The kind of sensors used for the applications varies a lot. The technology used for sensing as well as the sensed data can be different. Some systems rely on invasive sensors (sensors for whose installation or measurement the body is penetrated), while most of them are non-invasive.
- **Type of Sensing**: Khan et al. (2013) distinguish the kind of sensing, i.e., how much users are involved in the sensing process and architecture. *Participatory Sensing* involves the user, i.e., the user initiates and executes the sensing process. For example, a patient initiates a blood pressure measurement or a runner starts a run measurement. In opportunistic sensing, the application decides if, when, and how long the sensing is executed. Both kinds can further be distinguished into personal, social, and public sensing (Khan et al., 2013). Personal sensing concentrates solely on the acquisition of information of the current user. Social sensing collects social information and shares it with other participants in a social network. Public sensing acquires data for the public benefit, such as sensing traffic information or weather data.
- **System Architecture**: The architectures are very diverse. Many applications only run locally on mobile devices using solely the intrinsic features of the phone. Some applications rely on bigger architectures using peer-to-peer, client-server, or cloud-based architectures.
- **Communication Channels**: Depending on the features of the mobile device, different communication channels may be used by the applications. Some only rely on GSM, others are dependent on access to the Internet, enabled by 3G/4G or Wireless LAN. Additionally, in many applications external sensors rely on communication with the mobile device by using short-range communication, such as Bluetooth or ZigBee.
- **Purpose**: As mentioned before, the goals of the applications are manifold. Amongst others the application fields comprise health and fitness monitoring and self-management, disease surveillance and control, enhancement of clinical diagnosis, prevention, therapy, health promotion, community mobilisation, education and learning, training for health professionals, drug and treatment adherence, emergency response services and information, management of patient care, and decision support for health professionals (Global Observatory for eHealth, 2011; Mechael and Sloninsky, 2008).

3.2.1 Challenges

Challenges for mHealth applications are manifold. Depending on the functionality and equipment used, battery consumption is an issue. The usage of Bluetooth, GPS, Wireless LAN, sensors, and the display of information stresses the battery. Furthermore, the sensors and devices must be unobtrusive and not burden the users - and they have to be as portable as possible to fulfill the mobile character. As in almost all health applications the (wirelessly) exchanged data is typically sensitive, such that means for data security and privacy have to be provided. Mobile health applications often also have real-time requirements, especially when sensors are involved, as data is produced in real-time and, depending on the applications, users expect results in (near) real-time.

This poses challenges for data processing and management. Furthermore, physiological intra- and inter-subject variability make health applications a challenge for the utilized algorithms as they have to be adaptive to different patients and different conditions of one patient. If data is coming from several sources describing the same event, sensor fusion has to be executed to represent the full picture. This is not trivial, as we will discuss in Section 3.3.4. Finally, the wireless communication is an issue. The application needs a fallback solution, if the connection is breaking down, e.g., in areas where there is no cellular network signal.

In medicine, a plethora of medical devices is used for diagnosis and therapy. These devices mostly use proprietary protocols and data formats to export the measured data (and this is still the case for some of the new devices). However, the urgent need to exchange medical data between professional health services brought up standardized medical data formats. Especially, the advent of Electronic Health Records (EHR) pushed this development. Hence, many medical devices are capable today to output their data in a standardized way. The most important standards are ISO 11073/IEEE 1073 for data exchange between medical devices, HL7 (Health Level 7) as a set of standards for exchange of a broad range of health information, and DICOM (Digital Imaging and Communications in Medicine) for medical images (Schmitt et al., 2007). For mHealth in HL7 recent efforts are made to standardize short messages used in mobile communication, such as SMS (Datta et al., 2016). These standards can be important for the creation of mHealth applications when these applications are planned to send their data to the interfaces of other systems, such as Hospital Information Systems. Hence, the interoperability with other systems may be another challenge to be tackled by mHealth systems.

All of the challenges mentioned above are important to keep in mind during design and implementation of an mHealth application and the corresponding data management (when selecting the data sources, in the design of the algorithms, data processing and analysis, and so on). In Chapter 5 we introduce a methodology for the design and implementation of streaming applications addressing many of these issues.

3.3 Data Sources

In this section we will discuss which kinds of data sources are relevant for health care and specifically for mHealth applications. The most important data sources are sensors. They are intensively used in health applications to describe the condition of a patient along with the context she is in. Sensors can be combined to sensor networks, where the sensors communicate with each other. In the health domain, medical devices often do not adhere to general open standards, but implement proprietary interfaces and protocols. Others adhere to standards commonly used in hospitals as briefly sketched in the previous section.

3.3.1 Sensor Data

The National Cancer Institute Thesaurus (NCIT)[6] defines sensors as follows: "A device that responds to a stimulus, such as heat, light, or pressure, and generates a signal that can be measured or interpreted." Technically speaking, a sensor is a small device, which measures a certain phenomenon. The device consists of a sensing unit and optionally a transducer to convert the signal into a more convenient format (Hoffmann, 2011). The

[6]https://ncit.nci.nih.gov

raw output of a sensor is an electric signal (including noise), which can be described by frequency, amplitude, and form and time of their occurrence (Hoffmann, 2011). Properties of such a signal are that it is typically continuous, it is limited in energy or band, respectively, and it fulfills the Markov property (Sathe et al., 2013). The signal can be further processed by using amplification, filtering, digitalization, linearization, noise recognition, and filtering. *Smart sensors* are chips extended, amongst others, with microcontrollers and memory. They have some intelligence integrated on a single chip to convert the sensor signal into a more convenient format, and possibly process it in many other ways, e.g., executing data mining techniques (Eren, 2014). A sensor can furthermore be extended by communication interfaces and batteries, to enable wireless and mobile transmission. The sensor can then be used as a node in a sensor network. The sensor output can range from "simple scalar numerical or categorical values, to complex data structures" (Sow et al., 2013). Typically, a time series is produced, which includes timestamped data. In Figure 3.2 a sensor node including typical components is depicted, where RF stands for Radio Frequency and ADC is an analog-to-digital converter (Chen et al., 2011b).

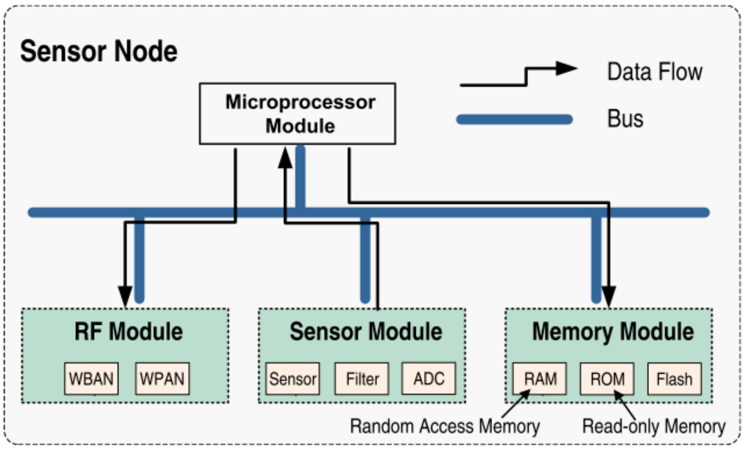


Figure 3.2: Architecture of a Typical Sensor Node (Chen et al., 2011b)

3.3.2 Medical Sensors

Medical sensors or *biosensors* measure biosignals as phenomenons to describe the current condition and changes in conditions of patients. Biosignals can be characterized according to their properties. The properties can be structural parameters, e.g., length or volume, or functional parameters, e.g., temperature or pressure (Hoffmann, 2011). The type of biosignal measured can be categorized into acoustic, biochemical, electrical, electromagnetic, mechanical, optical, thermal, and spatial signals (Eren, 2014; Hoffmann, 2011). The sensors can be invasive or non-invasive, and can be used for living objects and non-living objects, such as tissues and body liquids. The sensors differ in modality, i.e., the technology the sensor is based on, size and costs, and the utilized communication technology (Ko et al., 2010). Sensors do not necessarily have to be devices directly measuring the target object - people can also serve as sensors by observation or environmental sensors can be used to analyze additional context information (Sow et al., 2013).

3.3.3 Body Sensor Networks

The outputs of several sensors can also be combined to describe complex situations. Sensors can deliver their output to another sensor to combine the measurements or send them to a central processing unit. These tasks are covered by Wireless Sensor Networks (WSN). Body Sensor Networks (BSN) or Body Area Networks (BAN) are a specific kind of WSN tackling the special challenges posed by a human body. For example, they are very small in size, have to deal with motions, comprise a small number of sensors, and have high requirements regarding data protection (Yang et al., 2014b). A BSN consists of several body sensor nodes (sensors with communication unit, battery, low-level processing unit), an optional local central master node, and a personal server (i.e., either a proprietary device or an off-the-shelf mobile device) (Chen et al., 2011b; Poon et al., 2015; Yang et al., 2014b). The sensor nodes may be worn on the body, inside the body, or somewhere near (in the range of two meters) (Chen et al., 2011b). The master node has more processing, memory, and power resources than the other nodes to fulfill its tasks and it can help to organize the data collection from the other nodes (Espina et al., 2014). Intra-BAN communication connects the sensors with each other, typically using a star topology and short-range communication, such as Bluetooth or radio. However, the used topology is dependent on the targeted application (Espina et al., 2014). The local processing unit can fuse and preprocess the sensed data and transmit it to the personal server also using short-range communication. The personal server can then transmit the data via WAN to the next instance, such as dedicated access points, cloud servers, or remote application servers (Chen et al., 2011b). Sensors and in particular BSNs can produce high amounts of data in a very short time, which poses big challenges to the data processing and management. Hence, efficient means have to be found to get the most of the data while providing (soft) real-time requirements and accurate analysis results in the target applications, and keeping energy consumption and communication costs low.

BSNs can be used for monitoring of patients with chronic diseases (prognosis, progression, adherence to therapy plan, and so on), in hospital monitoring, home monitoring, care for the elder, self-monitoring, wearable and other health-related robotics, personalized health, and sports applications (Yang et al., 2014b). It can also be used for therapy systems, e.g., for Closed Loop Systems, which automatically adapt the therapy to measured parameter values (building a loop) and measure the effect of the changes (Hoffmann, 2011; Yang et al., 2014b).

Three types of BSN network architectures can be distinguished whose use is dependent on the target application: Standalone BSNs, Pervasive Sensor Networks, and Global Healthcare Connectivity (Espina et al., 2014). Standalone BSNs only consist of the BSN, its nodes, and a central processing unit. Pervasive Sensor Networks integrate contextual information with BSNs to accomplish an overall picture of a situation as complete as possible. Finally, Global Healthcare Connectivity includes a connection to the Internet to transmit the information to other services, such as hospitals. This can be done to (1) offer special services based on the measured data, e.g., heart pacer maintenance, (2) to complete the gathered information for the mobile application, e.g., using Electronic Health Records (EHR), or to (3) store the data for later use or deeper analysis.

3.3.4 Sensor Fusion

Sensor Fusion or Multi-Sensor Fusion describes a multi-step process and technologies for correlating and combining data from one or several sensors (or sources in general) to describe an event, the condition of an object, or a process, also taking into account the context of the object (Khaleghi et al., 2011; Yang et al., 2014a). Hall and Llinas (1997) describe sensor fusion as a transition between measured parameters and a final decision or inference, which reflects the process of abstraction from raw signal data to a final outcome, such as detection of a dangerous blood glucose level for a diabetes patient.

Based on the application requirements and hence, on the reason why multiple kinds of sensors are used, competitive, complementary, and cooperative sensor fusion models are distinguished (Yang et al., 2014a). *Competitive fusion* strives to increase the quality of the data by using several sensors observing the same parameter of an object. Sensor readings can be compared and reliability increased, while a higher fault-tolerance is achieved. But it can also lead to conflicting and confusing results, if the sensors readings differ too much. *Complementary fusion* uses several different sensors measuring different, but correlated aspects of an object to enhance fault tolerance and resolve ambiguities. Finally, *cooperative fusion* combines the data of several sensors to derive information, which could not be obtained using only a single type of sensor. The type of fusion model determines on which level in the data processing the sensor fusion takes place (Yang et al., 2014a). Using the competitive model, fusion can already be done on the raw data level, because sensor readings have the same dimensions and context. This is also called *direct data fusion*. Algorithms used on this level can be simple averaging or detection and estimation methods (Hall and Llinas, 1997; Yang et al., 2014a). As mentioned for competitive fusion, this setting is helpful to increase data quality, but can also be used for self-calibration of the BSN (Yang et al., 2014a). If direct data fusion is done in a master node in the BSN, it has the advantage of lower communication costs, as less data has to be transmitted to other processing components. On the other hand, it burdens the battery and more processing power and memory is required.

If the sensors are not measuring the same parameters, sensor fusion has to be done either on feature or on decision level. For the fusion on the feature level, features, which represent the sensor data well, are extracted from each kind of involved sensor. To this end, feature detection and extraction methods are used. For continuous sensor signals, features may describe the characteristics of the signal, i.e., analysis in the time or frequency domain, or both domains correlated. Subsequently, relevant features (selected by using common analysis methods, such as the Receiver Operating Characteristic (ROC)) are combined to a feature vector, which is processed by a data mining algorithm, such as pattern recognition or classification algorithm (Hall and Llinas, 1997; Yang et al., 2014a). But also fuzzy logic or probabilistic algorithms can be used. Depending on the problem, application, and corresponding requirements at hand, the algorithms for sensor fusion vary a lot. Khaleghi et al. (2011) describe the multitude of algorithms along a taxonomy categorizing the quality problems tackled by the algorithms. For BSN neural networks and probabilistic methods are particularly important (Yang et al., 2014a).

The fusion on decision level uses already processed, e.g., aggregated data, where each data source describes the event or object condition already on a very abstract level. The results are then combined by using inference and other probabilistic techniques, fuzzy logic, or classification algorithms (Hall and Llinas, 1997). Big data techniques, such as MapReduce, can support the fusion process for huge amounts of sensor data

(Yang et al., 2014a).

As we have described in this section, sensor fusion is an important means to integrate data from different sensors and can be done on different levels depending on the sensors and requirements. In the presented framework, sensor fusion could be implemented on different levels and we will see for the C-ITS as well as the mHealth case studies how this can be done.

3.4 Stream-based mHealth Applications

Many mHealth applications, which are enabled by data stream management, are related to the monitoring of vital parameters to support chronically ill patients. In particular, the most frequent case studies are about cardiovascular diseases, diabetes, and psychiatric conditions. But also monitoring of the general well-being is an important area, such as the health monitoring of soldiers, athletes, firefighters, or astronauts. After we have reviewed sensors as most important data sources and sensor fusion as way to process the data, we discuss briefly mobile and non-mobile health monitoring applications, which have been implemented with the help of data stream technology. We will not go into much detail, but we will consider which kind of applications are possible and what is required.

An early mHealth application based on data streams is presented by Chen et al. (2004). The application is used to monitor ECG and heart rate. It is able to filter out disturbances by movements using accelerometers. In this implementation, patients wear a mobile ECG sensor unit and accelerometers. All sensor nodes send their data wirelessly (via ISM and Bluetooth) to stationary and mobile end devices. These devices send the data to a server, which runs T2, a successor of the DSMS Tribeca (Sullivan, 1996). On the server, the data can be analyzed by permanent queries. From the ECG data the heart rate is extracted in the DSMS, the average heart rate is determined, and irregularities are detected. A disadvantage in the used stream processing is the fixed sizes of considered heart cycle windows (distinct parts of the ECG data), as inter- and intra-subject variabilities have to be considered in the data. Lindeberg et al. (2010); Stoa et al. (2008) analyse ECG measurements from surgical experiments with pigs to detect myocardial ischemia (blood supply for the heart muscle is insufficient as occurring before heart attacks). The authors adapted the DSMS Esper[7] to implement windows whose sizes are dynamically adaptable to the length of a complete cardiac cycle (distance between to heart beats). The system uses signal processing techniques, such as Fast Fourier Transform, to detect ischemia.

Brettlecker and Schuldt (2007) propose the distributed data stream architecture OSIRIS-SE, which is particularly tailored to mobile applications. It also provides specific means in case of the failure of nodes in the distributed system to avoid data loss. As an important use case an mHealth monitoring application is presented. The application makes use of ECG sensors, blood pressure sensors, and a webcam to detect critical conditions in patients suffering from chronic heart diseases. The web cam is used to detect red areas in the face of the patient. The ECG sensor sends its data to a PDA, which carries the OSIRIS-SE software. All other sensors send their data to a laptop also equipped with the OSIRIS-SE software. The laptop is used as a base station for sending out alarms, when one of the sensors measures or detects values in a critical range. The alarms are sent to the mobile device of a medical professional (also

[7] `http://www.espertech.com/products/esper.php`

running the software).

A completely mobile health monitoring data stream mining application without any server is presented by Haghighi et al. (2009). A quite simple mobile phone (Nokia N95) is used to analyze data from a blood pressure sensor (systolic and diastolic blood pressure and heart rate) and context information. This context information is represented by geometrical objects in multidimensional spaces, the context spaces, extended by fuzzy logic to abstract to situations. A Light Weight Clustering algorithm (a specific data stream mining algorithm) is used, whose parameters are dynamically adapted in real-time to detect abnormal blood pressure conditions such as hypertension or hypotension.

The framework ARTEMIS, which is based on the commercial DSMS IBM InfoSphere, has been used for the monitoring and analysis of astronauts (McGregor, 2015). Originally, the system was designed for neonatal intensive care monitoring. In a space craft astronauts can carry different wearable sensors, amongst others, ECG sensors measuring the ECG signal, breathing movements, and derived heart and breathing rates, pulse oximeters for measuring oxygen saturation and heart rate, and blood pressure cuffs. The system uses methods from clinical decision support for analysis, such as knowledge bases, inference, and data mining techniques. For example, the heart variability has been analyzed. The heart variability can be used in combination with other context information to detect irregular conditions, which can occur during space flights, such as fatigue or depression (McGregor, 2015). The stream analysis framework is deployed in the spacecraft and at mission control to analyze and visualize the data, which is replicated and transmitted to mission control.

3.5 Conclusion

In this section we introduced the most important concepts of mHealth applications. Although acknowledged standards exist for data exchange, in (m)Health applications the wealth of data sources, data types, and formats pose many challenges to consuming information systems. Even more than in the domain of C-ITS in many mHealth applications sensors and real-time data processing play a crucial role. Sensor fusion integrates data from different sensors, but is a complex task and might be prone to errors. However, while using sensor data at the same time the quality of the application results must be high to be suitable in a health setting. We gave examples of applications, which are already implemented using streaming technologies. Similarly, as in the case of the C-ITS domain the need for finding the optimal parameterization, algorithms, and circumstances for optimal application results is given. Both domains also share their requirements regarding real-time processing and data quality assessment. However, also for the field of mHealth a structured approach for designing and evaluating applications is missing. There is a strong need for data quality management of these applications (Kumar et al., 2013), as data quality plays a crucial role in the health domain. Hence, we present the design and implementation of two case studies in the mHealth domain in Chapter 7 using the proposed process model from Chapter 5 for data stream applications. Finally, we evaluate the application of the data quality management framework using the case studies in Chapter 8.

Chapter 4

Foundations: Data Stream Processing

Chapters 2 and 3 presented monitoring applications from the C-ITS and mHealth domain, which rely in particular on the use of sensors and real-time data. This data typically occurs in the form of continuous *data streams*. Other important examples for monitoring applications are weather observation and environment monitoring in general, monitoring of assembly lines in factories, or RFID monitoring. But there are also other typical applications producing data streams (further termed *stream applications*), such as social media analysis, stock price analysis, or network traffic monitoring, which all can produce even millions of samples per second. All these applications share characteristic properties, which are especially challenging for a data management system processing the generated data. The data is produced at a very high frequency, often in a bursty manner. This poses real-time requirements on applications processing the data and may allow them only one pass over the data. Data streams are typically unbounded, i.e., it is not known, if and when a stream will end. Also the mapping of recorded data to a time domain is important to rate the timeliness of the data and to make it interpretable in that dimension. This is also connected to another problem in data streams - namely disorder of data, which is likely to happen when the protocol used for transmission cannot guarantee to sustain order or network latencies occur (Srivastava and Widom, 2004). Depending on the application, data from multiple sources also have to be fused or integrated and analyzed subsequently to get a comprehensive picture of a situation. For example, data from several health sensors (such as ECG, temperature, blood pressure) may be integrated to derive that a patient is in a critical situation. Another challenge is caused by the pure mass of data. Due to limited system resources in terms of storage, memory, and CPU time, algorithms analyzing the data cannot store and process the entire data, but can process the data only once (*the one-pass property*) utilizing the available resources. It may be also required to combine streaming data with historical or static data from a common database. Typically, data is prone to quality defects. In particular, sensor data likely includes errors introduced by the imprecision of measurement techniques, loss of packet data, environmental influences, internal electronic failures, depleted batteries, or communication interruption (Paradis and Han, 2007). The more complex the architecture is the higher is the potential for errors. If a wireless sensor network is used, data quality may be additionally hampered due to communication errors and failures between the nodes of the network (Mahmood et al., 2015).

To tackle the aforementioned challenges in data stream processing, a specific system type, namely *Data Stream Management Systems* (DSMSs), has evolved. The properties of the stream applications discussed before lead to a long list of requirements for DSMSs. In contrast to common data management systems and based on the nature of stream applications, DSMSs have to react to incoming data and deliver results to listening users frequently (Stonebraker et al., 2005). This is also termed as the *DBMS-Active, Human-Passive model*, while the *DBMS-Passive, Human-Active model* is implemented by common Database Management Systems (DBMS) (Carney et al., 2002). In DSMSs a reactive behaviour is realized by continuous queries, which are registered by a user in the system. Once registered the queries are executed on the incoming data incessantly. But applications may at the same time require to allow ad-hoc user queries (Abadi et al., 2003a) or the definition of views (Golab and Özsu, 2010). Data flowing through a DSMS must not only be processed and forgotten, but the system also has to react to changes induced by the data items, which may lead to recalculation of already produced results. This requires an appropriate change management in the system (Abadi et al., 2005; Arasu et al., 2003a).

Handling unbounded data streams while having only a limited amount of memory available and being restricted in CPU time for processing the data is one of the main challenges for a data stream management system. The creation of incremental results for critical data processing operations and the application of window operators as discussed in Section 4.2.1 are only two examples of how these issues are solved in DSMSs. As already mentioned, most of the applications pose real-time requirements to the data processing system (Stonebraker et al., 2005). This implicitly comprises the requirement to be scalable in terms of data rates, which in turn demands techniques for load balancing and load shedding. However, the system must also be scalable in terms of queries as some application contexts can get complex and require the introduction of several queries at the same time. Therefore, the demand for and the adaptability to newly registered, updated, or removed queries is obvious (Chandrasekaran et al., 2003). This also brings up the need for multi-query optimization, e.g., by sharing results of operators in an overall query plan. Query plan modification during query processing is not only desirable for query optimization, but also to serve Quality of Service demands under varying system resource availabilities (Raman et al., 2003).

Additionally, a DSMS must address the imperfectness of data. Unordered data has to be handled adequately, and may be tolerated in controlled bounds. Means to recognize and rate the quality of the data processed are also crucial to make assumptions about the produced answers or results (Tran et al., 2010). Finally, as Stonebraker et al. (2005) demand, a DSMS also has to have a deterministic behaviour, which outputs predictable and repeatable results to implement fault tolerance and recovery mechanisms. This contrasts with non-deterministic components in a DSMS, such as a component randomly dropping tuples to compensate a high system load (Zdonik et al., 2004).

4.1 Architectures

Due to the system requirements for DSMSs their architectures differ in several aspects from the traditional relational DBMS architecture. Querying has to be viewed from a different angle as data is pushed into the system and not pulled from the system (Chandrasekaran et al., 2003). In Data Stream Management Systems queries are executed continuously over the data passed to the system, also called *continuous* or *standing queries*. These queries are registered in the system once. Depending on the system, a query can be formulated mainly in two ways: as a declarative expression, mostly done in an SQL-dialect, or as a sequence or graph of data processing operators. Some of the systems provide both possibilities. A declarative query is parsed to a logical query plan, which can be optimized. Similar to DBMS this logical query is afterwards translated into a physical query execution plan (QEP). The query execution plan contains the calls to the implementation of the operators. Besides of the actual physical operators, query execution plans include also *queues* for buffering input and output for the operators. A further auxiliary element in QEPs are *synopsis structures*. DSMSs may provide specific synopsis algorithms and data structures, which are required when an operator, e.g., a join, has to store some state to produce results. A synopsis summarizes the stream or a part of the stream. It considers the trade-off between memory usage and accuracy of the approximation of the stream. Additionally, *load shedders* can be integrated in the plan, which drop tuples on high system loads. In most systems, execution plans of registered queries are combined into one big plan to reuse results of common operators for multiple queries. The physical query plan may be constantly optimized based on performance statistics, for example. In Figure 4.1 the query processing chain is depicted.

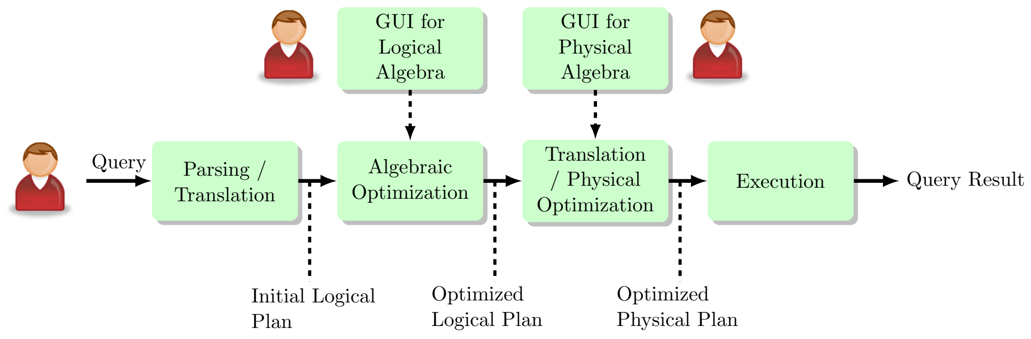


Figure 4.1: Query Processing Chain in DSMSs (Krämer and Seeger, 2009)

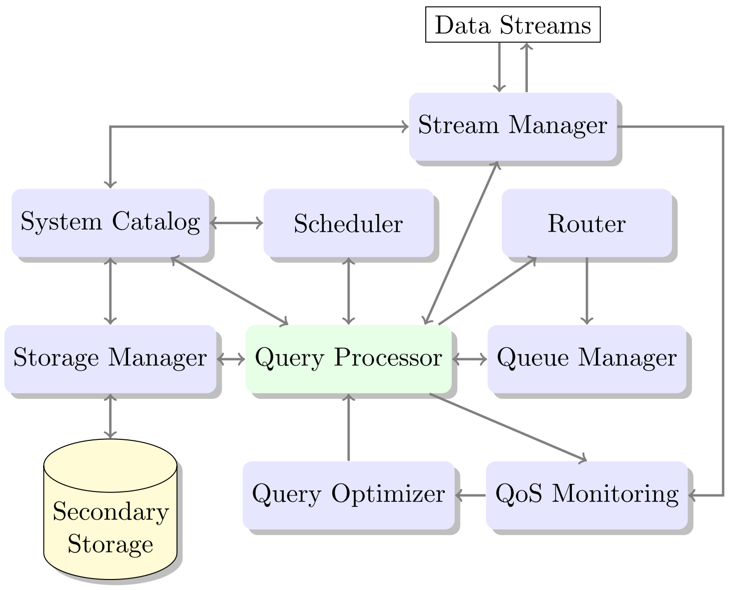


Figure 4.2: A Generic Architecture of a DSMS Merging Ideas from (Ahmad and Çetintemel, 2009; Golab and Özsu, 2010)

In Figure 4.2 a generic architecture of a DSMS merging ideas from (Ahmad and

Çetintemel, 2009; Golab and Özsu, 2010) is shown. First of all a DSMS typically gets data streams as input. Wrappers are provided, which can receive raw data from a source, buffer, and order it by timestamp (as e.g. implemented by the *Input Manager* in the STREAM system (Srivastava and Widom, 2004)) and convert it to the format of the DSMS. Systems, which adopt a relational data model, represent data stream elements as tuples, that adhere to a relational schema with attributes and values. The *Stream Manager* coordinates the creation and processing of streams and stream elements. After reception the tuples are added to the queue of the next operator according to the query execution plan. This can be done e.g. by a *Router* component as implemented in the Aurora system (Abadi et al., 2003b). The management of queues and their corresponding buffers is handled by a *Queue Manager*. The Queue Manager can also be used to swap data from the queues to a secondary storage if memory resources get scarce. To enable access to data stored on disk many systems employ a *Storage Manager*, which handles access to *secondary storage*. This is used, when persistent data is combined with data from stream sources, when data is archived or swapped to disk. In addition it is required when loading meta-information about, inter alia, queries, query plans, streams, inputs, and outputs. These are held in a system catalog in secondary storage.

While the queue implementation decides which element is processed next, a *Scheduler* determines which operator is executed next. The Scheduler interacts closely with the *Query Processor*, which finally executes the operators. Many systems also include some kind of *Monitor*, which gathers statistics about performance, operator output rate, or output delay. These statistics can be used to optimize the system execution in several ways. The scheduler strategy can be influenced, e.g., prioritizing specific sub-plans. Furthermore, the throughput of a system can be increased by a *Load Shedder*, i.e., stream elements selected by a *sampling* method are dropped. The Load Shedder can be a part of a Query Optimizer, a single component, or part of the query execution plan. Furthermore, the statistics can be used to reoptimize the current query execution plan and reorder the operators. For this purpose a *Query Optimizer* can be included.

DBMSs and DSMSs share some of their query optimization goals – both try to minimize computational costs, memory usage, and size of intermediate results stored in main memory. But obviously the priorities for these goals are different for the two system types. DBMS mainly try to reduce the costs of disc access (Elmasri and Navathe, 2004) while DSMSs mainly have to reduce memory usage and computation time to be fast enough. Of course these different goals stem from the different data handling strategies (permanent storage vs. real-time processing). In a DSMS a query is also not only optimized before execution, but it is adaptively optimized during its run time. This enables the system to react to changes of input streams and system and network conditions.

4.1.1 Distributed Systems

Distribution is also a very important aspect of many DSMSs today. To divide up tasks and data processing load, many DSMSs offer the possibility to define nodes (i.e., processes, which are running on separated machines or in a cluster). Borealis (the successor of Aurora), was one of the first systems to provide a distributed architecture. Distributed streaming systems differ in multiple aspects. First, the topology of the node networks may be different. There are centralized, hierarchical systems in which one master node distributes the workload to worker nodes, such as Apache Storm[1] or Apache S4.[2] Apache Storm is maintaining a master node, called the *Nimbus*. The Nimbus distributes parts of the process graph to be executed (the *topology*) (i.e., code) and hence, work load, to slave nodes (the *Supervisors*) inside a Storm cluster network. A *Zookeeper* cluster coordinates the communication between Nimbus and Supervisors and keeps states of the network to make it fail-proof. Similarly, the stream processor Apache S4 runs a *platform* for coordination and distribution of tasks to the *S4 nodes*. The S4 nodes run in *clusters* (a group of nodes), which execute the applications (operator graphs). Other network topologies are designed in a peer-to-peer fashion, such as in the OSIRIS-SE system (Brettlecker and Schuldt, 2007), where each node of the network hosts one or more streaming operators including their state (an aggregation of the so far seen data). If a node fails, the other nodes take care of migrating the operators and their states to another node, using replicated metadata. Peer-to-peer systems are more complicated to implement and maintain and hence, are not found so often.

Another aspect in which distributed systems may vary is how workload is distributed (statically or dynamically (Schneider et al., 2012)) and what is distributed (parallelism on operator/data-level or on query/graph-level (Castro Fernandez et al., 2013; Heinze et al., 2014)). Some systems allow only a simple, manually designed distribution, but this is cumbersome and error-prone (Schneider et al., 2012), such that most of the contemporary distributed systems have a system-integrated automatism. Distribution of data to the different nodes on operator-level is, for example, done by Apache S4 by dispatching data based on its key.[3] Data can also be distributed to nodes randomly as done in Aurora. Apache Storm offers more complex operators, which can dispatch the tuples based on different criteria, e.g., based on attributes, i.e., that tuples with the same values for the defined attributes will be dispatched to the same thread in a node (field grouping).[4] Dynamic rebalancing, i.e., scaling up and down in terms of operators and resources, are supported by many systems as it is an important requirement for today's data volumes. This is mainly done based on costs, such as operator selectivity (ratio of operator input and output) (Schneider et al., 2012). The distribution on query level is creating algorithms for the distribution of operators to machines (e.g., Bolts in Apache Storm). These algorithms can be distinguished according to their optimization goal, execution model, and run time (Heinze et al., 2014).

4.1.2 Big Data Streaming Architectures

In the last years streaming architectures have evolved especially towards their scalability. Heinze et al. (2014) termed this the third generation of streaming systems, because they are based on systems such as Aurora or STREAM, but go beyond them providing properties which allow for even higher input and lower latency. These systems put more effort in offering parallelization, elasticity, and distribution and apply techniques from NoSQL and cloud-based systems, such as MapReduce, to increase their efficiency. A nice overview of the evolution of many data streaming systems can be found in (Heinze et al., 2014). Those streaming systems nowadays do not exist on their own, but are often part of a Big Data Analytics ecosystem (Dolas, 2015).

Big Data Analytics ecosystems combine components from large-scale data mining

[1] `http://storm.apache.org`
[2] `http://incubator.apache.org/s4/`
[3] `http://incubator.apache.org/s4/doc/0.6.0/overview/`
[4] `http://storm.apache.org/releases/current/Tutorial.html`

frameworks, data collection systems (e.g., Apache Flume[5]), distributed message queueing middlewares (e.g., Apache Kafka[6]), NoSQL database management systems and processing frameworks, DSMSs, and other data management solutions (Ranjan, 2014; Dolas, 2015). Dolas (2015) distinguishes three general layouts for such ecosystems with emphasis on streaming data. Firstly, a linear streaming architecture basically consists of several components lined up. The components may comprise the data sources, a data collection system, a messaging middleware, and a real time processing system (such as Apache Storm[7]). From here, events or results can either directly be provided to consumers or the results are persisted in a permanent big data storage platform, such as Hadoop.

Secondly, to account for streaming and batch data sources in parallel, the general Lambda Architecture presented by Marz and Warren (2015) tries to make the data available using views. The architecture combines batch and stream processing by implementing a batch, a speed, and a service layer. In the batch layer data is stored in an HDFS store holding the master copy of this data. The batch layer uses the common MapReduce paradigm to prepare precomputed views (so called *batch views*) for the data which are stored in a NoSQL database in the service layer. The service layer provides a query interface which enables to query the views. As soon as a view is precomputed by the batch layer, it is injected into the service layer database. Hence, data is missing in the service layer, namely the data which has arrived after the last upload of a view. To fill this gap an additional speed layer is introduced, where only recent data is processed and provided to the service layer, such that a query can access both kinds of data. Reprocessing of the data is possible and the input data remains unchanged (Kreps, 2014). Drawbacks are, that the data processing code has to be implemented twice - for the batch and the stream part, and that there is an architectural overhead (Kreps, 2014).

The Kappa Architecture (Kreps, 2014) proposes to just use a little trick to make reprocessing possible without implementing application code for stream processing twice. For the stream processing a common stream processing engine, such as Apache Storm, is used which runs a job on the current data and stores results in an output table in a service database. This table can be queried by end users or applications. A messaging middleware, such as Apache Kafka[8], is logging the streaming data for a certain amount of time. If data has to be reprocessed because the application code has changed, a parallel job processes the data logged with Kafka and stores it in a second output table. This is done until the new job has caught up with the current time and finally the old output table is substituted by the new one.

There exist also some streaming extensions to MapReduce batch processing systems. Queries can be handled by batch processing systems in an extremely efficient manner. However, data stream processing in real-time and low latency updates are mostly lacking. To tackle this issue Apache Spark, for example, is extended by Spark Streaming using the abstraction of Discretized Streams, short DStreams (Zaharia et al., 2012). DStreams consist of so called resilient distributed datasets (RDDs) which are the smallest batch unit to be processed in parallel by Spark operators. However, this is somehow a very limited and untypical stream processor, as RDDs are built as fixed interval batches (similar to tumbling windows, cf. Section 4.2.1), which do not allow

[5]`https://flume.apache.org`
[6]`http://kafka.apache.org`
[7]`http://storm.apache.org`
[8]`http://kafka.apache.org`

for data element by data element processing (Beggs, 2015). Another "workaround" for near real-time insertions in NoSQL systems, such as Hadoop, is offered by SQLStream's s-Server.[9] It provides an adapter, which can ingest data into Hadoop at a very high scale (448 MB/s).[10]

4.1.3 Complex Event Processing

Parallel to DSMSs, the terms Event Processing, Complex Event Processing (CEP), or Event Stream Processing have evolved. These terms are referring to a concept which is closely related to the notion of data streams. The corresponding systems are specialized on the processing and analysis of events to identify higher level events, such as predicting a political development from Twitter tweets or detecting a serious condition from vital parameter readings. DSMSs have a broader application scope, but CEP applications can also be rebuild with DSMSs (Etzion and Niblett, 2011). Cugola and Margara (2012) distinguish DSMSs and CEP systems as data processing and event detection systems denoting the different focuses. The focus disparities result, among others, in differences in architectures, data models, and languages between CEP systems and DSMSs.

CEP as well as extended Big Data architectures are beyond the scope of this work. They open up interesting features, which may be helpful to implement complex applications. However, we will concentrate solely on the use of DSMSs for which we presented important features of their architectures. In the following sections we will discuss the specific query languages, data models, and semantics used in DSMS. Subsequently, the basics of data quality management and data stream mining are explained, to built the foundation for the understanding of the rest of this work.

4.2 Query Languages

Basically, two main types of query languages for DSMSs can be distinguished: declarative languages (mostly relational, based on SQL) and imperative languages which offer a set of operators (also called box operators) to be assembled to a data-flow graph using a graphical user interface or code. The imperative languages often also include operators which represent SQL operations. SQL-based languages are widely used though SQL has many limitations for querying streams (Law et al., 2004). Systems which include a declarative SQL-based language are, for example, STREAM (Continuous Query Language, CQL) (Arasu et al., 2006), PIPES (Krämer and Seeger, 2009), or StreamMill (Expressive Stream Language, ESL) (Thakkar et al., 2008). A system which solely uses SQL standard queries without any extra operators is SQLStream s-Server.[11] Imperative languages are supported for example by the Aurora/Borealis system (SQuAl) (Abadi et al., 2003b), Apache Storm[12](amongst others: Java, Python), Apache S4[13] (Java), or System S/InfoSphere Stream[14] (SPADE/SPL) (Gedik et al., 2008).

Because a stream is potentially unbounded in size, it is neither feasible nor desirable to store the entire stream and analyze it. Some of the operations known from traditional query languages, such as SQL, might wait infinitely long to produce a result, as the operation would have to see the entire stream to generate a result (defined as *blocking*

[9] `http://www.sqlstream.com/blaze/s-server/`
[10] `http://www.sqlstream.com/products/stream-processing-architecture`
[11] `http://www.sqlstream.com`
[12] `http://storm.apache.org`
[13] `http://incubator.apache.org/s4`
[14] `http://ibm.com/software/data/infosphere/streams`

operations) (Babcock et al., 2002a). The missing support of sequence queries, i.e., retrieving sequential data, is one crucial limitation known from relational databases and SQL (Law et al., 2004). One simple yet powerful way is to first extract only a desired portion of the stream and use this portion (called window) in the remainder of the query. Therefore, a very important requirement for a DSMS query language is the provision of *windows* (Cherniack and Zdonik, 2009; Arasu et al., 2003b; Patroumpas and Sellis, 2006). Windows are operators, that only select a part of the stream according to fixed parameters, such as the size and bounds of the window. Hence, they provide an approximation of the stream, but are at the same time implementing the desired query semantics (Babcock et al., 2002a). A window is updated based on fixed parameters (Patroumpas and Sellis, 2006) and internal matters of the system (e.g., in principle, a result can be updated whenever a new element arrives or whenever time proceeds) (Jain et al., 2008). We will detail the different types and parameters of windows in Section 4.2.1.

Besides the definition of windows in the language, also an approved set of query operations accompanied by established semantics is beneficial for a DSMS query language (Arasu et al., 2003b). This can be accomplished, e.g., by using a common (standardized) query language based on relational algebra and its operators for operations on finite tuple sets (commonly called *relations*). The advantage is the reuse of operator implementations and transformations for query optimization (Arasu et al., 2003b). But such an approach also risks to be complicated, because the closure under a consistent mathematical structure, such as bags in relational algebra, is crucial to enable nested queries and algebraic optimization. Cherniack and Zdonik (2009) investigated the property of DSMS query languages to be closed under streams. A language is closed under streams if its operators get streams as an input and if the output of the operators are streams as well. They state that most languages provide stream-to-stream operators by either implicitly using operators for windowing and conversion to streams or special stream-to-stream operators. Only CQL provides the possibility to explicitly formulate the conversion of relations to streams by specific relation-to-stream operators. These operators can either add output elements to a result stream when a new element compared to the last time step is in the result set (Istream operator), or when an element has been removed from the query result relation compared to the last time step (Dstream operator), or all elements which are present in the result set in the current time step (Rstream) are added to the stream (Arasu et al., 2006, 2003b). But to avoid an inconsistent query formulation producing results not closed under streams, CQL also offers many default query transformations. In fact, Arasu et al. (2003b) showed that a stream-only query language (only using stream-to-stream operators) can be build based on the set of CQL operators. We will detail the types of operators used in continuous query languages in Section 4.2.1. To illustrate the use of data stream operators we will now give examples from the C-ITS domain introduced in Chapter 2.

Example 4.1 Suppose we want to monitor the number of speeders in a reduced speed area. We would like to retrieve the number of V2X messages which have been sent in the last minute and which contain a speed greater than 30 km/h. Additionally, we also want to retrieve an update every 10s. In CQL the query from Listing 4.1 could be formulated to fulfil our information requirement.

The same query can be formulated in an imperative query language by assembling box operators. In Figure 4.1 a query formulated in Aurora's SQuAl is depicted. The Filter operator is similar to the selection in relational algebra - it retrieves all tuples with speed greater than 30.0. Furthermore, an Aggregate operator is connected to the

Filter operator, which can be parametrized with the aggregate function and details for an implicit window integrated in the operator.

Listing 4.1: Query 1

```
SELECT Istream Count(*) FROM
  V2XMessage[Range 1 Minute Slide 10s]
WHERE Speed > 30.0
```

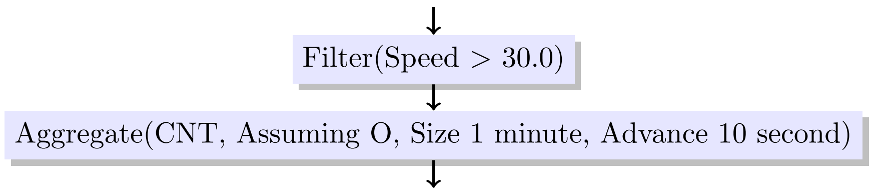

Filter(Speed > 30.0)
Aggregate(CNT, Assuming O, Size 1 minute, Advance 10 second)

Figure 4.3: Query 1 Formulated in an Imperative Language

Another requirement to DSMS query languages is the extension of the language by custom functions and operators to include more complex actions, such as data mining or custom aggregates (Law et al., 2004). Most of the languages provide a possibility for custom extensions. Examples for systems allowing for extensions are System S by IBM (Gedik et al., 2008), Aurora (Abadi et al., 2003b), or StreamMill (Thakkar et al., 2008). In Apache Storm so called *Bolts* are derived Java classes, which can contain any Java code processing the incoming tuples,[15]. However, Apache Storm can be extended to also allow for high-level declarative querying with SQL by using the SQLstream for Storm[16] API for Storm. SQLstream builts Bolts based on a continuous SQL query and integrates them into the processing graph (topology) or lets them run in parallel.

4.2.1 Data Models and Semantics of Data Streams

Before we go into detail on continuous queries and operators used in the queries, we have to understand what is exactly meant by the term "data stream". In this section we will also break data streams down to their indivisible components and examine the data models applied in DSMSs.

Data Models

Depending on the desired application realized with the DSMS and the data sources to support there are different demands on the adopted data model and the semantics of the query language. In the literature, the relational model is very dominant, presumably due to the well-defined semantics, the established set of relational algebra operators, and the very well studied principles of DBMS. Also, from a user's perspective, the step from SQL towards its streaming extensions seems to be quite small. However, there are also data sources which do not conform to the flat table concept for discrete data offered by the relational data model (Maier et al., 2005). There exist specialized systems and corresponding query languages, which deal with those. There are systems processing

[15]`http://storm.apache.org/releases/current/Tutorial.html`
[16]`http://www.sqlstream.com/storm`

XML data using XML-QL (Chen et al., 2000), XPath (Peng and Chawathe., 2003) or XQuery (Botan et al., 2007), RDF streams (Barbieri et al., 2009), continuous signal data, object streams, graph-based models (Feigenbaum et al., 2005; Sun et al., 2007; Angel et al., 2012), logical models (Zaniolo, 2012) or spatio-temporal data streams. Finally, there exist also generic systems and models, which allow the use of multiple languages, such as the Odysseus framework (Bolles, 2009) or the PIPES system (Krämer and Seeger, 2009). Overviews of systems, models, and query languages have been presented by Geisler (2013); Golab and Özsu (2010); Heinze et al. (2014). However, in this work we will concentrate on declarative relational models and query languages, but are not limited to it. In the following semantics for the relational streaming model is presented.

Representations and Semantics of Data Streams

A data stream S can be understood as an unbounded multiset of elements, tuples, or events (Law et al., 2004; Babcock et al., 2002a; Demers et al., 2005; Krämer and Seeger, 2009). This means, each tuple can occur more than once in the stream which is denoted by a multiplicity value. Each tuple $(s, \tau) \in U$, where U is the support of the multiset S, has to adhere to the schema of S. The *support* U of a multiset S is a set which consists of the elements which occur in the multiset S at least once. Multiset and support can be defined as follows (Syropoulos, 2000; Singh et al., 2007):

Definition 4.1. *(Multiset and Support of a Multiset) A* multiset *is a tuple $\mathcal{A} = (B, f)$, where B is a set and f is a function $f : B \to \mathbb{N}$, assigning a multiplicity to the tuple (number of occurrencies of the tuple in the stream). The* support *of $\mathcal{A}$ is a set U which is defined as follows:*

$$U = \{x \in B | f(x) > 0\},$$

i.e., $U \subseteq B$.

The schema of a stream is constituted of attributes $A_1, \ldots, A_n$. Each tuple contains also an additional timestamp τ from a discrete and monotonic time domain. Most definitions of data stream semantics do not consider the timestamp as a part of the stream schema (Arasu et al., 2006; Krämer and Seeger, 2009; Abadi et al., 2003b) and therefore, it is always separately listed in the tuple notation.

We use the following formal definition of a data stream (Geisler, 2013) which is based on the definition of Arasu et al. (2006):

Definition 4.2. *(Data Stream) A data stream S is an unbounded multiset of data stream elements (s, τ), where $\tau \in T$ is a timestamp attribute with values from a monotonic, infinite time domain T with discrete time units. s is a set of attribute values of attributes $A_1, A_2, \ldots, A_n$ with domains $dom(A_i), 1 \leq i \leq n$, constituting the schema of S. A stream starts at a time τ_0. $S(\tau_i)$ denotes the content of the stream S at time τ_i, being*

$$S(\tau_i) = \{< (s_0, \tau_0), m_0 >, < (s_1, \tau_1), m_1 >, \ldots, < (s_i, \tau_i), m_i >\}$$

.

Example 4.2 The schema of the V2X message stream from Example 4.1 would be written according to this definition as:

$$\texttt{V2XMessage(Timestamp, TS, AppID, Speed, Acceleration, Latitude, Longitude)}$$

In this example the timestamp `Timestamp` has been generated by the DSMS and `TS` is the creation timestamp defined by the application. The different kinds of timestamps are detailed in Section 4.2.2.

Depending on the data model, the attributes $A_1, \ldots A_n$ can contain values of primitive data types, objects or XML data. For example, Krämer and Seeger (2009) consider tuples to be drawn from a composite type (in the relational case this is the schema) and its attributes can contain objects. This generic definition allows them to be open to different data models.

Data streams commonly adhere to an append-only principle, i.e., data once inserted in the stream will not be removed or updated (Babu and Widom, 2001). But to enable updates in the streams, there are also systems which do not only support insertion of tuples, but also updates and deletions, for example, in Borealis (Abadi et al., 2005), or STREAM (Arasu et al., 2006). In the PIPES (Krämer and Seeger, 2009) and STREAM systems, streams are separated into *base streams* and *derived streams* to denote the origin of the stream. Base streams are produced by an external source and derived streams are produced by internal system operators. Furthermore, in PIPES stream notations are distinguished based on the level in the query processing chain. Streams from external sources are termed *raw streams* and adhere to the attributes plus timestamp notation described above (according to which the tuples are ordered in the stream). Streams on the logical or algebraic level are termed *logical streams*. The tuples of a logical stream contain attributes corresponding to the stream schema, a timestamp τ and a multiplicity value. The multiplicity value indicates how often the tuple occurs at time τ in the stream, which implements the bag semantics explicitly. Finally, the PIPES system also introduces a *physical stream* notation. The notation is used to represent streams in query execution plans, which in addition to the attributes include a validity time interval with start and end timestamp.

While above we have described the representations of streams and tuples, the semantics or denotation of a stream (i.e., the underlying mathematical concept) can be separated from these representations (Maier et al., 2005). So far we used the rough semantics of an unbounded multiset for a stream. This rough semantics raises the questions of how tuples are organized in the stream, which tuples are included, and how a stream evolves with the addition of tuples. A stream denoted as a *multiset* or *bag* of elements, allows duplicate tuples in the stream (Arasu et al., 2006; Bai et al., 2004; Krämer and Seeger, 2009). The denotation as a *set*, i.e., without duplicates, is also possible. Furthermore, a stream could also be interpreted as a sequence of states (Maier et al., 2005), where a relation transitions from one state to another (on arrival of tuples or progression of time). For example, the insertion of a new tuple in a stream denoted as a bag can be recursively defined by describing what will be the following state, i.e., the "result bag".

Inclusion of Persistent Relations

The sole querying of streams is often not sufficient to implement certain applications. Take the traffic state estimation application as an example. To improve our assertion about the traffic state we could also integrate information from other streams, such as Floating Phone Data, or from persistent sources such as historical data about the traffic state at the same time last week or last year on the same street. This would involve to join data of streams with other streams and to join streams with persistent data from "static" data sources. In (Cherniack and Zdonik, 2009) this ability is called *correlation.*

Most languages support joins between streams and relations, because they are easy to implement. In principle, each time new tuples arrive in the stream these are joined with the data in the relation and the operator outputs the resulting join tuples. The join between streams is a little bit more complicated. Most languages require at least one stream to be windowed (Cherniack and Zdonik, 2009) which results in the former situation of joining a stream with a finite relation. Some offer also specific operators for joins without windows.

The use of persistent relations in queries requires them to be represented in the query language. In SQL-based languages these are noted in the same way as streams (it is maybe more correct to say that the streams are represented in the same way as relations). Representations for relations can also be related to the notion of time. A relation at time τ then consists of a finite, unbounded bag of tuples which is stored in the relation at time τ (Arasu et al., 2006). In CQL, a time-related relation is called an *instantaneous relation*, and analogously to streams, *base relations* and *derived relations* are distinguished.

Continuous Queries and Algebraic Operators

Now that we have clarified the main constituents of continuous queries in DSMSs, namely, streams, tuples, and relations, we will proceed to explain continuous queries and operators for languages based on the relational algebra. So what is the difference between a continuous query and an ad-hoc query? One goal of Tapestry (Terry et al., 1992), one of the first data management systems processing continuous data, was to give the user the impression that a query is continuously executed (which is not possible). To achieve this in DSMSs, the result of a continuous query at time τ is equal to the result of the query executed at every time instant before and equal to τ (Terry et al., 1992; Arasu et al., 2006). That means it takes into account all tuples that arrived up to τ. Depending on the language the result of the query can be a stream or a finite set of tuples. For example, in CQL the result can be either, while other languages only support streams.

There are three main models how data is processed in a DSMS, i.e., when continuous queries are executed. When a *time-driven model* (Jain et al., 2008) is used, a query will be updated with progression of time (on every time step of the system). In a *tuple-driven model* a query is evaluated on the arrival of each tuple, unless the query includes some temporal restriction, such as a time-based window (Jain et al., 2008). The *event-driven model* allows to define events or triggers on whose firing the query is executed, e.g. in OpenCQ (Liu et al., 1999). Of course these could be also temporal events or an amount of tuples seen so far, but these could be also user-defined events, such as a fired alert or an incoming e-mail.

One main problem for operators processing streams is the fact that streams are unbounded. Especially, *blocking operators*, i.e., operators which do not produce a result tuple before they have seen all tuples on their inputs (Babcock et al., 2002a) are problematic. The set of these operators comprises aggregations, groupings, but also set operations, such as `NOT IN` or `NOT EXISTS` (Law et al., 2004; Gurevich et al., 2007). For example, if we would like to calculate the average over the speed of the incoming C2X messages, a classical average operation has to wait until the stream of messages ends (but we do not know when the stream ends) to produce the desired result. In contrast, *non-blocking operators* produce results periodically or on arrival of new tuples, i.e., incrementally (Law et al., 2011). *Partially blocking operators* are operators which can produce intermediate results but also a final result in the end (Law et al., 2004).

Obviously, a language for continuous queries can then only be based on non-blocking operators (Law et al., 2004; Babcock et al., 2002a). But the set of non-blocking operators is neither in relational algebra nor in SQL sufficient for all expressible relational queries (Law et al., 2004). Another type of challenging operators are *stateful operators* (Zaharia et al., 2012; Krämer and Seeger, 2009; Maier et al., 2005) (in contrast to *stateless operators*). These operators, e.g., joins, require to store a state for their operation, which for streams is unbounded in size. Hence, one remedy to these problems is to approximate processing the stream as a whole as good as possible. One simple yet powerful approach is the partitioning of the stream into small portions, so-called *windows*. Each window is a finite bag or set of tuples and can be processed also by the common relational blocking operators. A second possibility is to provide incremental implementations of these operators, which are able to update the result with new tuples and output "intermediate" results. Finally, an approximation of the stream in form of a summary or *synopsis* can be used to operate on. In the following we will detail window operators and their semantics, as these are intensively used in this work.

Windows

In continuous query languages based on SQL, windows are a crucial extension to the algebraic set of operators. It depends on the language which types of windows are supported. A window is always built according to some ordered *windowing attribute* which determines the order of elements included in the window (Patroumpas and Sellis, 2006; Maier et al., 2005). The type of window, i.e., how it is determined which elements are valid in the current window, according to (Patroumpas and Sellis, 2006) can be described by its measurement unit, the edge shift, and the progression step. The *measurement unit* can be either a number of x time units (*time-based window*) or tuples (*tuple-based window*) declaring that the elements with timestamps within the last x time units or the last x elements are valid for the window at the point in time of the query. In the following different types of windows and their properties are described. We define a time-based window of size l (Geisler, 2013) similar to the definition in (Patroumpas and Sellis, 2006) in Definition 4.3. We assume that the stream elements are ordered by timestamp τ before a window operator is applied.

Definition 4.3. *(Time-based Window) A* time-based window W_{l_T} *(with window size* $l_T \in \mathbb{T}$*) over a stream* S *at time* $\tau_i \in T$ *is a finite multiset of stream elements*

$$W_{l_T}(S(\tau_i)) = \{< (s_k, \tau_k), m_k > \mid (s_k, \tau_k) \in U, \tau_i - l_T \leq \tau_k \leq \tau_i, \tau_k \geq \tau_0\},$$

where m_k *is the multiplicity of the tuple in the subset and* U *is the support of stream* $S(\tau_i)$*.*

The definition for a tuple-based window of size l_N is similar:

Definition 4.4. *(Tuple-based Window)*

Let $S(\tau_i) = \{< (s_0, \tau_0), m_0 >, \ldots, < (s_n, \tau_n), m_n >\}$, $\tau_i \geq \tau_n \geq \tau_0 \wedge \tau_j \geq \tau_{j-1}\ \forall j \in \{1, \ldots, n\}$*, be the content of stream* S *at time* τ_i*. Then a* tuple-based window W_{l_N} *(with window size* $l_N \in \mathbb{N}$*) over stream* S *at time* $\tau_i \in T$ *is a finite multiset of stream elements*

$$W_{l_N}(S(\tau_i)) = \{< (s_k, \tau_k), m_k > \quad | \quad (s_k, \tau_k) \in U, j \leq k \leq i, \\ \exists m'_j, m''_j \;\; m'_j \geq 0, m''_j > 0, \\ m_j = m'_j + m''_j, \quad m''_j + \sum_{\ell=j+1}^{i} m_\ell = l_N\}$$

where $(s_n, \tau_n) \in U$, $\forall \tau_j \in U \;\; \tau_n > \tau_j$, *and* U *is the support of stream* $S(\tau_i)$.

This means that we go backwards in the stream from time τ_i on and search for the last point time τ_n at which elements arrived. We collect exactly N tuples in the stream. We assume that the last tuple which (partially) fits into the window is $< (s_j, \tau_j), m_j >$. Then only the multiplicity portion m''_j which still fits into the window will be added to the window.

In the Aurora system (Zdonik et al., 2004; Abadi et al., 2003b) value-based windows as a form of generalization of time-based windows are presented. These windows implement the idea of having a different windowing attribute instead of a timestamp - the attribute just has to be ordered. Furthermore, the value-based window should return only those tuples whose value of the particular attribute is within a specific interval (hence, value-based windows). A similar concept are predicate-windows (Ghanem et al., 2006) suited for systems which work with negative and positive tuples. *Partitioned windows* (Li et al., 2005; Arasu et al., 2006) are applicable to time- or tuple-based windows and follow the idea of dividing the stream into substreams based on filter conditions and of windowing them separately. Afterwards, the windows of the substreams are unioned to one result stream (Abadi et al., 2003b).

The *edge shift* of a window describes the motion of the upper and lower bounds of the window. Each of them can either be fixed or moving with the stream. For example, in the most common variant, the *sliding window*, both bounds move, while for a *landmark window* one bound is fixed and one is moving.

Finally, the *progression step* or *periodicity* defines the intervals between two subsequent movements of a window. This again can either be *time-based* or *tuple-based*, e.g., the window can move every 10 seconds or after every 100 arrived tuples. When the contents of windows in each progression step are non-overlapping, this is termed a *tumbling window*, i.e., size and sliding step have an equal number of units. Windows can also be *punctuation-based.* A notification tuple sent with the stream indicates the window operator that it should evaluate. We will explain punctuations in Section 4.2.2.

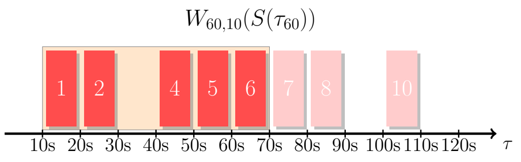
In the following, one of the most commonly used forms of a moving window, a sliding time-based window, is defined as:

Definition 4.5. *A* sliding time-based window *with window size* $l_T \in \mathbb{T}$ *and slide value* $v \in \mathbb{T}$ *over a stream* S *at time* $\tau_i \in T$ *is a finite multiset of stream elements*

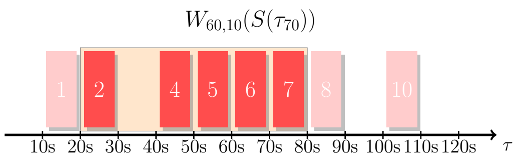
$$W_{l_T,v}(S(\tau_i)) = \{< (s_k, \tau_k), m_k > \quad | \quad (s_k, \tau_k) \in U, \exists j \in \mathbb{N} : \tau_0 + j \cdot v \leq \tau_i, \\ \tau_0 + (j+1) \cdot v > \tau_i, \tau_i \geq \tau_0 + l_T, \\ \tau_0 + j \cdot v - l_T \leq \tau_k \leq \tau_0 + j \cdot v, \tau_k \geq \tau_0\}$$

where U *is the support of stream* $S(\tau_i)$

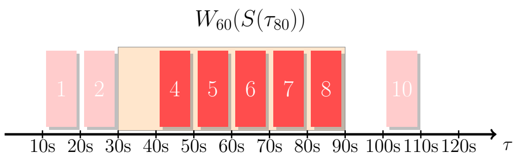
Example 4.3 In our traffic example we would like to retrieve the number of V2X messages with `speed` greater than 30 km/h from the last minute every 10s. We realize this by defining a time-based sliding window W_{l_T}, v of size $l_T = 60\text{s}$ and with a sliding step of $v = 10\text{s}$ over the `V2X message` stream. In Figure 4.4(a) the content of the window after 1 minute is shown. This is the first point in time, where the query with the window delivers results (hence, $\tau_i \geq \tau_0 + l_T$). The window contains the elements 1,2,4,5,6. After 10 more seconds the window slides, element 1 is dropped from the window and element 7 is added (Figure 4.4(b)). After another slide after 10 seconds element 2 is dropped and element 8 is added to the window (Figure 4.4(c)).


$W_{60,10}(S(\tau_{60}))$
1
2
4
5
6
7
8
10
10s 20s 30s 40s 50s 60s 70s 80s 90s 100s 110s 120s
τ

(a) Window after 60s


$W_{60,10}(S(\tau_{70}))$
1
2
4
5
6
7
8
10
10s 20s 30s 40s 50s 60s 70s 80s 90s 100s 110s 120s
τ

(b) Window after 70s



(c) Window after 80s

Figure 4.4: Example of a Sliding Window with Size of 1 Minute and Slide Step of 10 Seconds

Depending on the language windows can either be implicitly included in an operator (see the definition of the Aggregate operator in Example 4.1, Figure 4.1) or defined as a separate operator in the query language. In SQL-based declarative languages the window construct included in the SQL:2003 standard is extended, e.g., by a `SLIDE` keyword to enable the definition of a sliding step.

4.2.2 The Notions of Time and Order

We already mentioned before that time plays an important role in DSMSs. In all DSMSs the processed tuples have some kind of timestamp assigned from a discrete and monotonic time domain. The timestamps allow to determine if a tuple is in order or not and enable the definition of time-based windows (Srivastava and Widom, 2004).

Time

The prominent status of timestamps can already be seen from several stream definitions in corresponding semantics. The timestamp is always handled as a specific attribute which is not part of the stream schema (Patroumpas and Sellis, 2006; Arasu et al., 2006; Krämer and Seeger, 2009). A monotonic time domain T can be defined as an ordered, infinite set of discrete time instants $\tau \in T$ (Patroumpas and Sellis, 2006; Arasu et al., 2003b). For each timestamp exists a finite number of tuples (but it can also be zero).

In the literature, there exist several ways to distinguish where, when, and how timestamps are assigned. First of all, the temporal domain from which the timestamps are drawn can be either a *logical time domain* or *physical clock-time domain*. Logical timestamps can be simple consecutive integers, which do not contain any date or time information, but serve just for ordering. In contrast, physical clock-time includes time information (e.g., using UNIX timestamps). Furthermore, systems differ in which timestamps they accept and use for internal processing (ordering and windowing). In most of the systems *implicit timestamps* (Babcock et al., 2002a; Zdonik et al., 2004), also called *internal timestamps* or *system timestamps* are supported. Implicit timestamps are assigned to a tuple, when it arrives at the DSMS. This guarantees that tuples are already ordered by arrival time when they are pipelined through the system. Implicit timestamps assigned at arrival in the system also allow for estimating the timeliness of the tuple when it is output (Abadi et al., 2003b). Besides a global implicit timestamp (assigned on arrival), there exists also the concept of new (*local*) timestamps assigned at the input or output queue of each operator (time of tuple creation). This could also be implicitly expressed by the tuple's position in the queue (Zdonik et al., 2004). In contrast, *explicit timestamps* (Babcock et al., 2002a; Zdonik et al., 2004), *external timestamps* (Bai et al., 2004) or *application timestamps* (Srivastava and Widom, 2004) are created by the data stream sources and an attribute of the stream schema is determined to be the timestamp attribute. Depending on the semantics of a data stream, tuples can include more than one timestamp, e.g., denoting the start and end of an event (Demers et al., 2005).

An interesting question is how timestamps should be assigned to results of binary operators and aggregates to ensure semantic correctness. Babcock et al. (2002a) propose two solutions to assign a timestamp to results of a join. The first option is to use the creation time of a join output tuple when using an implicit timestamp model. The second option is to use the timestamp of the first table involved in the join in the `FROM` clause of the query, which is suited for explicit and implicit timestamp models. For aggregates similar considerations can be made. For example, if a continuous or windowed minimum or maximum is calculated, the timestamp of the maximal or minimal tuple, respectively, could be used. When a continuous sum or count is calculated, the creation time of the result tuple or the timestamp of the latest element included in the result can be used. If an aggregate is windowed there exist additional possibilities. The smallest or the highest timestamp of the elements in the window can be used as they reflect the oldest timestamp or most recent timestamp in the window, respectively. Both maybe interesting, when timeliness for an output tuple is calculated, but which one to use depends obviously on the desired outcome. Another possibility would be to take the median timestamp of the window.

Order

Many of the systems and their operators rely on (and assume) the ordered arrival of tuples in increasing timestamp order to be semantically correct (Srivastava and Widom, 2004). For example, in the STREAM system (using a time-driven execution model) time can only advance to the next time instant, when all elements in the current time instant have been processed (Arasu et al., 2006). This has been coined as the *ordering requirement* (Srivastava and Widom, 2004). But as already pointed out, this can not be guaranteed especially for explicit timestamps and data from multiple sources. In the various DSMSs basically two main approaches to the problem of disorder have been proposed.

One approach is to tolerate disorder in controlled bounds. The Aurora system, for example, does not assume tuples to be ordered by timestamp (Abadi et al., 2003b). The system divides operators into *order-agnostic* and *order-sensitive* operators. The first group of operators does not rely on an ordering of elements. The order-sensitive operators are parametrized with a definition how unordered tuples should be handled. The definition contains the attribute which indicates the order of the tuples and a *slack* parameter. The slack parameter denotes, how many out-of-order tuples may arrive between the last and next in-order tuple. All further out-of-order tuples will be discarded. The order can also be checked for partitions of tuples, specified by an additional `GROUP BY` clause in the order definition. A general concept of a slack parameter, called *adherence parameter* has been presented for the STREAM system (Arasu et al., 2004a; Babu et al., 2004). The adherence parameter is a measure for how well a stream "adheres" to a defined constraint. The authors define a set of *k-constraints* one of which is the ordered-arrival-k-constraint. This constraint conforms to the slack parameter's ordering semantics. The second way to handle disorder is to dictate the order of tuples and reorder them if necessary. While the use of implicit timestamps is a simple way of ordering tuples on arrival (Srivastava and Widom, 2004), the application semantics often requires the use of explicit timestamps. *Heartbeats* (Srivastava and Widom, 2004) are tuples sent with the stream including at least a timestamp. These markers indicate to the processing operators that all following tuples have to have a timestamp greater than the timestamp in the heartbeat. Heartbeats are only one possible form of *punctuation* (Tucker et al., 2003). Punctuations, in general, can contain arbitrary patterns which have to be evaluated by operators to true or false (Tucker et al., 2003). Therefore, punctuations can also be used for approximation. They can limit the evaluation time or the number of tuples which are processed by an otherwise blocking or stateful operator. Other methods for reordering tuples in limited bounds use specific operators, e.g., the BSort operator in Aurora (Abadi et al., 2003b).

4.3 Data Quality Management

The quality of utilized data is one of the most important factors determining if an application is successful or not. Poor quality can lead to dissatisfied users, financial losses – in the worst case it can also risk people's health and lives (Redman, 2004; Wang and Strong, 1996). Decisions based on data of poor quality will inevitably be wrong in most cases. While the availability of huge masses of data and data streams enables new applications, common sources producing these data streams (such as sensors) pose new problems to data quality assessment as they may be failure- and error-prone. Common problems especially with sensor data are the loss of packet data, environmental influences, internal electronic failures, depleted batteries, or communication interruption (Paradis and Han, 2007). Due to the high volume of data produced continuously by sensors, traditional data quality procedures cannot be applied, while the various types of sensors may pose further problems (Campbell et al., 2013).

Especially, critical applications such as systems for tornado warnings (Tran et al., 2010), heart attack prediction (Lindeberg et al., 2010), or traffic safety applications require a high reliability of the data. In the ITS context safety applications require high quality output which demands for data with corresponding quality. Consequently, the determination, monitoring and improvement of data quality (DQ) is a crucial issue in sensor- and data stream-based systems. Furthermore, for big data applications, in which the data is often also processed in a stream-like way, the interest of industry in tools for DQ is rising (Judah and Friedman, 2014).

But what exactly is data quality? According to Juran (1999), quality is (1) the set of properties of a product which satisfy user needs and (2) when a product is free of deficiencies or errors, which is also termed "fitness for use" (Juran and Godfrey, 1999b). This definition has been adopted for data by Redman (2004) defining "data to be of high quality if they are fit for their intended uses in operation, decision making and planning." Hence, Redman (1996) rates the quality of data according to the grade of how the data satisfies user needs, which makes it on first glance hard to quantify, hard to measure, and highly dependent on the application. But what this definition is supposed to convey is that information system designers have to analyse each application and the data's planned use to determine which aspects of data quality are important. Also in ITS literature it is recommended that data quality design should be closely connected to the "application requirements and cost implications" (Tarnoff, 2002). Furthermore, not only the application, but also the view on the data and the role of the user can make a difference (Jarke et al., 1999). Wang and Strong (1996) emphasized that data quality cannot be rated according to a single aspect. Even for a single application field and a single set of data, multiple facets must be considered to assess the quality of the corresponding data. Hence, Wang and Strong (1996) introduced the concept of a data quality dimension being "a set of data quality attributes that represent a single aspect or construct of data quality." We simplify this definition for our purposes as follows:

Definition 4.6 (Data Quality Dimension, Data Quality Metric, and Data)**.** *A* **data quality dimension** *is considered as an atomic attribute which has a clear definition and is measurable using one or more well-defined metrics. A* **data quality metric** *is a function mapping one or more data items to a numeric* DQ value*. Under the term* **data** *we subsume all kinds of value items which are produced or processed - no matter if they are measured as raw values or derived or calculated values.*

Now that we explained the foundations of DQ management, we will review and discuss existing DQ management models in the following.

4.3.1 Data Quality Management Models

As mentioned in the last section, DQ dimensions are single attributes describing a single quality aspect of the data at hand and are measured using metrics. A crucial question is, how we can get an application's data quality requirements and define corresponding dimensions. In fact, an overall management process is required to not only determine the dimensions, but also to find out how to measure, to control, and to improve data quality throughout the usage of the data. Batini and Scannapieco (2006) define a DQ methodology as "a set of guidelines and techniques, that [...] defines a rational process for using the information to measure and improve the quality of data of an organization through given phases and decision points."

Data quality management models have taken ideas and concepts from general quality management methodologies, where products, customers, and processes are in the focus. In principle, data can also be seen as a product (Wang, 1998) whose properties might differ from properties of other resources, such as intangibility (Redman, 1999), but at the same time they are as valuable for businesses relying on them as other resources are. Hence, it is comprehensible that models such as Total Quality Management have been adopted, but adapted to the specifics of data.

Total Quality Management (TQM) is a term which has developed since the 60's as a pure quality end control of products in factory processes until today to an enterprise-wide bundle of processes and concepts to measure and improve quality (Koch, 2011). It involves all departments, groups, employees, and collaborators and is not restricted to factories or production. It implements, depending on the version of the TQM model, an action cycle (Plan-Do-Check-Act (PDCA), Plan-Do-Study-Act (PDSA), Deming, or Shewhart (Moen and Norman, 2006)) which iterates through these action phases to manage product quality. Wang (1998) adapted the TQM model to a Total Data Quality Management (TDQM) methodology, regarding data as a product. The methodology includes a TDQM cycle which is based on the Deming cycle defining steps suited for data quality management. The cycle is composed of four successive steps:

1. **Define:** Determination of DQ requirements and dimensions
2. **Measure:** Definition and execution of DQ metrics.
3. **Analyze:** Analysis of DQ problems and corresponding costs.
4. **Improve:** Application of techniques for DQ improvement.

As a competitive or successor model of Total Quality Management the concept of *Six Sigma* has evolved. Originally, a concept developed by Motorola, which only allowed a certain percentage of parts per minute to be defective, it became an enterprise quality management strategy later on (Schroeder et al., 2008). The Six Sigma model has four major elements: (1) parallel meso-structure (quality measures are developed in parallel to normal business on multiple organizational levels), (2) improvement specialists (groups of trained employees in each project to be improved), (3) structured method (use of a method for improvement, such as the define, measure, analyze, improve and control (DMAIC) model), and (4) performance metrics (set of metrics which are suited to measure certain parameters of the process and user satisfaction) (Schroeder et al., 2008). This concept has also been applied to data quality management. For example, in the transportation field it was used in the project ROSATTE (Schützle, 2009) for managing data quality for road safety attributes, such as positions of warning signs. In ROSATTE specifically the DMAIC model has been adopted, being implemented mainly as a manual approach, where process improvement is emphasized (Schützle et al., 2010).

There are also models which have been specifically elaborated for data quality management. Lee et al. (2002) proposed the AIMQ methodology, which consists of a model to organize determined data quality dimensions into four categories, a questionnaire to measure the values of these dimensions from users and weight them. Furthermore, the authors aggregated the results for each of the four dimension categories to four values. These values are finally analyzed by a benchmarking tool which compares the values with data quality values of competitors' and a best practice solution (gold standard). For DQ management in Data Warehouses (DWH) Jarke et al. (1999) proposed the DWQ methodology based on the Goal Question Metric (GQM) paradigm proposed by Basili and Rombach (1988) originally for software engineering processes. What is special about the DWQ methodology is that data quality management is integrated into the DWH architecture and described by a meta-model. The meta-model is formulated in Telos (Mylopoulos et al., 1990), a knowledge representation language, which includes concepts and relationships between them. In the meta-model goals for data quality can be defined. For each goal several quality dimensions can be assigned. Each of the dimensions is assessed by measurement agents which store the results also in the DWH repository. For each of the goals, quality queries (the question in the GQM approach) can be formulated in the meta-model and executed over the repository to retrieve the measurement results. This integrates the quality management tightly into the DWH, but makes it at the same time highly flexible.

Batini et al. (2009) analyzed multiple data quality methodologies in literature and identified the following general process steps:

1. **State Reconstruction:** analyzes the application's processes, services, and their costs and also data quality problems. Can be seen as a requirements analysis for data quality management design. The result of this phase are the DQ dimensions to be measured.

2. **Assessment/Measurement:** actions to acquire the values of the identified data quality dimensions. In *Measurement* values are determined unbiased, while in *Assessment* the measured values are additionally compared with a reference value.

3. **Improvement:** process to counteract when quality is poor. The according means can either be process-driven or data-driven, describing the level on which the improvement is made.

This also complies with the three steps Juran (1993) describes for quality management in general: quality planning, quality control, and quality improvement (Godfrey, 1999). Batini and Scannapieco (2006) base their Complete Data Quality Methodology (CDQM) on these three steps. The outcome of the state reconstruction is here the requirements analysis and a processes/organizations matrix, describing who participates in which process and with which role. In the Assessment phase the definition and measurement of DQ dimensions is done, while in the Improvement phase improvement actions are considered and documented in a data/activity matrix.

All the aforementioned approaches have one crucial disadvantage: they all require a lot of manual effort especially for the design, analysis and improvement tasks. This is not feasible for a data stream setting, where data passes by rapidly and is volatile. But some of the ideas of these classical approaches are very useful and approved and can be adopted for a DQ management in DSMSs. For example, the metadata-driven approach of DQ management in the DWQ project automates at least a part of the analysis phase.

4.4 Data Stream Mining

Data stream mining algorithms provide the ability to analyze the data while it is streaming by. Achieving this and at the same time being efficient the algorithms have to fulfill certain properties (Domingos and Hulten, 2003; Lee et al., 2015). These properties include:

- the one-pass property (every record can only be seen once),
- the amount of memory for the model has to be limited,
- the amount of time to process each element has to be constant,
- distribution between multiple processes has to be taken into consideration,

- any-time learning must be possible,
- the algorithm should be able to adapt to concept drift, i.e., it should update the model whenever the process creating the data or the underlying data distribution changes (Tsymbal, 2004; Aggarwal, 2015a).

The requirement of a constant processing time has to be discussed. A new class of algorithms, called *anytime mining algorithms* (Kranen, 2011) do not require to deliver a result in a constant time, but allow mining on demand (Aggarwal, 2015a). These algorithms only use the time available between two stream elements to create and refine an initial result in the available time slot.

Many variants of mining algorithms for data streams from different parts of the field have been introduced, such as clustering algorithms (Silva et al., 2013; Aggarwal and Yu, 2008) or finding frequent item sets (Jiang and Leung, 2013; Cheng et al., 2008). In this thesis we will concentrate on data stream classification algorithms as these were most important in our application scenarios. The framework introduced in this thesis is not limited to classification algorithms and can be extended, e.g., to use clustering algorithms. In the following we will briefly introduce some basic principles of data classification, explain the specifics of data stream classification, and discuss different categories of data stream classification algorithms. Besides decision trees other classification schemes have been adapted to or have been newly implemented for data streams. These include Support Vector Machines (Laskov et al., 2006) or fuzzy-rule classification (Angelov and Zhou, 2008). Furthermore, Gama and Rodrigues (2009) pointed out that neural networks are well suitable for data streams without any changes, using stochastic sequential training. We extensively use decision tree algorithms suited for data streams in our experiments. Hence, we will explain these in detail in this chapter. Furthermore, the important aspects of concept drift and class balancing are explained briefly, as these are also studied in the experiments.

4.4.1 Data Classification

Classification is the problem of assigning one of a set of *class labels* to a data record based on a set of *features*. A *classifier* is built with a (supervised) learning technique using a set of (labeled) *training data*. The output of this *learning phase*, the classifier, is represented by a model which is utilized to predict a class for unknown data records (Aggarwal, 2015b; Han and Kamber, 2006). Commonly, the set of training data used to build the model is carefully segregated from a set of *test data* which is used to evaluate the accuracy of the model (percentage of correctly predicted elements) in the *testing phase*. To determine which features are characterizing the data records of the set or stream very well in terms of the targeted class labels, several feature selection methods exist. These are either integrated into a classification algorithm (Wrapper Models) or are independent means and determine the features by certain criteria (Filter Models), such as the Gini Index, Entropy, or the Fisher's Index (Aggarwal, 2015b). The Gini Index calculates for a single value of the feature for all classes how often it results in each class compared to the overall occurrence of this class label and sums this up (and using normalization), to see how discriminative the feature value is for the class label. In the end, weighted averaging over the index values is done to determine the degree of balance in the data set for that feature. For Entropy this is done similarly.

After discriminating features have been identified a mining algorithm can be applied on those. There exist different categories of classification algorithms which mainly are:

probabilistic methods, decision trees, rule-based methods, instance-based learning, and Support Vector Machines (SVM) (Aggarwal, 2015b). In classification, the performance of models and corresponding algorithms is typically rated according to the ratio of class errors in comparison to overall classified elements. In two-class problems usually training elements are termed as *positives* (P), if they are labeled with the desired or more important class and as *negatives* (N), when the opposite is true. Types of errors are then *false positives* (FP) (the algorithm tells you, you have a heart attack, but you have not) or *false negatives* (FN) (the algorithm tells you, you do not have a heart attack, but actually you have). The importance of each of these errors depends on the application. Correctly classified elements are termed *true positives* (TP) and *true negatives* (TN), respectively. This schema is known as *confusion matrix analysis* (Batista et al., 2004).

Common metrics to rate a learned model based on error rates are *precision*, *recall* (or sensitivity), specificity, f-measure, and accuracy. These are defined as follows:

- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- Specificity = $\frac{TN}{FP+TN}$
- Accuracy = $\frac{TP+TN}{P+N}$
- F-measure = $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

In this chapter, we will only elaborate the principles of the algorithms relevant and specific for data stream mining. These will be detailed in the corresponding subsections.

4.4.2 Data Stream Classification

Unlike in the traditional setting in data stream classification the amount of training and test data is not a problem – data superabounds. However, algorithms cannot rely on having the complete training set in memory. Therefore, new training and evaluation methods to work on streams have to be found. *Incremental learning* approaches these new challenges. It is characterized by using the derived knowledge from the prior seen data for incoming new data and it is integrating new data into that knowledge without accessing the old data (He et al., 2011; Rutkowski et al., 2014b; Hoens et al., 2012). In data stream mining it is assumed that training and test data are represented as streams and may also be mixed in one stream (Lee et al., 2015; Aggarwal, 2015a). Two procedures for stream mining proved to be suitable (Bifet et al., 2010a,b).

The *Interleaved Test-Then-Train* or *Prequential* approach uses each incoming stream element first for testing the model and then for training, allowing a very fine-granular evaluation. In the prequential method accuracy can also be evaluated only over classification results of a window containing the last n elements (Gama et al., 2009). The *Holdout* method periodically evaluates the model with a set of elements held back, giving a snapshot of the current accuracy of the model. Furthermore, the type of algorithms can be distinguished into *incremental* and *decremental* learners (Aggarwal, 2015a). While incremental learners change their model constantly based on the recent incoming data, decremental learners are using windows to forget data which is no longer in the window.

In this work we mainly use the interleaved test-then-train approach and incremental decision trees and Bayes networks as classification algorithms. Hence, we will describe only those in detail.

Decision Trees

Decision trees can be considered as one of the most popular and also a very efficient type of classification algorithms. A decision tree is a directed graph with nodes and edges, where each inner node represents a *splitting rule* on one or more attributes of the data (e.g., `Average Speed` <= 20) determining to which of its child nodes the data is routed next until the item reaches a leaf node. Each leaf node is labeled with a class into which the data is categorized (e.g., `slow-moving traffic`). The used attributes and classes can be either categorical (discrete) or numerical (continuous) (Han and Kamber, 2006). There are algorithms building binary trees as well as algorithms creating n-ary trees.

Most learning algorithms recursively build a tree as described above by using a greedy top-down tree induction (Han and Kamber, 2006). Top-down tree induction tries to find a set of splitting rules which discriminates the available data best. For each new node the splitting rule can be found by optimizing a certain goal. For decision trees common goals are the error rate (how many elements would end up with a false class label?), information gain, Gini index, and many more (Han and Kamber, 2006). There exist also specific goals for binary trees (Lee et al., 2015). If a new node would only process data from the same class, class purity is reached and the node is created as a leaf node with the corresponding class label (Rutkowski et al., 2014b).

During tree induction it can happen that a tree adapts too much to the training set it reads (*overfitting*). This is reflected by an increasing error rate on unseen examples. Also, a bigger sized tree might result in performance drops, which may be not justified by a small gain in accuracy. Hence, decision tree algorithms may apply two different kinds of *pruning*. In *Prepruning* node creation is stopped based on a criterion such as a maximum tree size or a threshold for an information gain delta (Han and Kamber, 2006; Lee et al., 2015). *Postpruning* substitutes branches of a completed tree by a leaf based again on specific criteria such as the cost complexity (function of number of leaves and error rate) (Han and Kamber, 2006). Popular and commonly used greedy decision tree algorithms are ID3, C4.5, and CART (Han and Kamber, 2006; Lee et al., 2015). Basically, these differ in the type of tree produced (binary or non-binary), the strategies for node splitting (Rutkowski et al., 2014b), and strategies for pruning.

Decision tree algorithms for streaming are either based on a greedy or on a statistical strategy (Lee et al., 2015). Furthermore, there are algorithms which use single learners and others use ensemble learning. We will briefly discuss some algorithms from each of these categories.

Well-known approaches for statistical techniques are the Very Fast Decision Tree (VFDT) (Domingos and Hulten, 2000) and algorithms based on it. VFDTs are based on Hoeffding Trees which use a statistical measure called Hoeffding bound. The bound determines how many examples are required at each node to choose a split attribute. It guarantees that with a given probability the same split attribute is selected as the attribute which would have been chosen if all training elements were known already. The algorithm first determines the two best attributes by using a relevance measure such as information gain or the Gini index. Afterwards, if the difference between the two attributes' relevance values excels the calculated Hoeffding bound for the examples seen so far, the best attribute is chosen for the split.

This algorithm has been further extended to Concept-adapting VFDTs (Hulten et al., 2001). The algorithm detects concept drift in time-changing data streams and grows alternative branches for subtrees with an insufficient accuracy. When the alternative subtrees deliver better accuracy values than the old ones, they substitute them in the tree. The training data is processed using a sliding window which decreases the impact of older data (Aggarwal, 2015a).

There are also streaming variants of classical greedy algorithms. Based on the ID3 algorithm several streaming variants (ID4, ID5, ID5R) have been proposed (Lee et al., 2015) for solving two-class problems. ID4 cuts down all children of a node if it detects that the discriminating attribute of this node does no longer have the best information gain. ID5R improves this algorithm by not discarding but restructuring subtrees beneath a node with such an attribute by pulling up nodes below. This algorithm guarantees that ID3 would produce the same tree on a set of data seen so far.

Other approaches use hybrid variants combining ideas from the greedy and VFDT techniques. For example, Rutkowski et al. (2014b) introduce a streaming algorithm based on VFDT and CART. CART creates a binary tree and uses the Gini index to decide on node splitting. In principle the algorithm is similar to the VFDT, but calculates the Gini gain for deciding which attributes are the most discriminating two attributes and uses not the Hoeffding bound, but the Gaussian approximation to decide if the best attribute is sufficiently better than the second one to justify a split. They also use a tie breaking parameter to force a split after a certain amount of elements although the two best attributes have not been sufficiently different. The Gaussian Decision Tree algorithm based on ID3 and VFDT (Rutkowski et al., 2014a) is similar to the above described algorithm, but is applicable to two-class problems only and it uses prepruning. The prepruning tests if one class dominates the other in the so far seen examples. If yes, further examples are awaited before the split condition is tested.

There have also been approaches which use ensemble techniques for learning. Ensemble algorithms learn and use multiple models in parallel for classifying an element using a basic model type (the *base learner*), e.g., a Hoeffding Tree. In addition, each example in the training set is weighted to determine its importance and the different models can also be weighted to denote how reliable their classification results are. Algorithms differ in how they weight the examples and the models (Bifet and Kirkby, 2009a). The most famous variants are *bagging* and *boosting* algorithms. Bagging algorithms train their different models on different training sets which have been assembled by randomly drawing (with replacement) a set of examples from a common training set with the same size as this training set. The votes of all models have the same weight, i.e., they are unweighted. Boosting algorithms combine several so called weak learners (very simple models) to build a strong learner. This is done in an iterative process, where models are chained and successive models learn from the failures of previous ones. The models are weighted according to their overall accuracy representing their reliability. Boosting algorithms train their models in an iterative way. They feed training examples into a model which classifies them. A set of these examples is forwarded to the next model. This set consists of an equal number of falsely and correctly classified examples. The next model is only trained with examples which have been classified contradictorily by the former models. For example for data streams Hashemi et al. (2009) propose a one-versus-all-classifier technique, where for each class a binary classifier is trained, i.e., the tree only decides between the class it is specialized on and all other classes. For each element all classifiers decide if the element is in their own class and the classifier with the highest confidence value gets the vote. For data streams online bagging and online boosting algorithms have been proposed (Oza and Russell, 2001; Oza, 2005).

4.4.3 Class Balancing

There are certain cases, where the imbalance of data elements in the classes is very high. However, situations which occur seldom in the overall data may be particularly important for the application at hand. For example, a heart attack is more important to be detected reliably as opposed to normal heart activity. Class imbalance has an impact on performance of learning algorithms as shown, for example, by Prati et al. (2015). If the class imbalance is 1/99 percent, then on average a 20% performance loss could be observed, while 20/80 had a loss of 5% on average. The loss is significant if the minority class is only present in 10% of the cases (Prati et al., 2015). While Support Vector Machines are quite immune to class imbalance, decision trees suffer particularly from class imbalance and class overlap. Decision trees may create many nodes to learn the minority classes, but these might be sacrificed in postpruning again. Also, more leaves will be labeled with the majority class than with the minority class (Batista et al., 2004). Furthermore, Batista et al. (2004) showed that resampling methods positively effect the performance of learning algorithms with imbalanced data sets. Hence, we will discuss these algorithms briefly as our case studies involve imbalanced data sets. In the next section we will also discuss concept drift as the combination of class imbalance and concept drift is one core challenge in incremental learning (Hoens et al., 2012).

To stress the importance of a class for a learning algorithm, there exist two main approaches, namely example weighting and example resampling (Lee et al., 2015). Example weighting assigns higher weights to more important classes or examples or assigns costs to certain types of errors. Hence, it does not change the set of training data, but gives algorithms a hint what to learn more intensively. However, not all algorithms are suited for these techniques (Prati et al., 2015). Resampling techniques change the training set sizes for the different classes, e.g., artificially adding more examples to underrepresented minority classes and removing examples from overrepresented majority classes. Due to alteration of the training set this influences learning, either prolongation of learning or having less examples to learn (Domingos, 1999). The used algorithms are the same or similar to the ones in the common data classification scenario (Lee et al., 2015). However, there are some stream specific algorithms and we will describe some of the algorithms from both categories, which we use in this work as examples.

Resampling techniques can be categorized into undersampling, oversampling, and hybrid techniques, combining the two former ones. The oversampling techniques with the best performance are random oversampling and SMOTE (Batista et al., 2004; Prati et al., 2015). Random oversampling picks examples from the minority class of the data set randomly and clones these to create balance. SMOTE (Chawla et al., 2002) executes interpolation on the minority class examples to artificially create new data elements. There are extensions such as LN-SMOTE, Borderline SMOTE, and Safe-Level SMOTE which extend SMOTE by interpolating over the k-nearest neighbours of the minority examples (Maciejewski and Stefanowski, 2011). In ADASYN (He et al., 2008) hard to learn examples from the minority class are getting a higher weight than others. For these more artificial examples than for other examples of the minority class are generated.

Undersampling techniques are less successful then oversampling techniques (Prati et al., 2015). However, there are some popular algorithms, which are frequently used. Random undersampling again just removes randomly picked examples, which might include important information for the learning algorithm. Algorithms based on Condensed Nearest Neighbour (CNN) reduction try to find for randomly picked examples from the training data set those, which lie at the boundaries of the element's neighbourhood. This means that an NN algorithm determines the nearest element that has a different class and uses this for the reduced result set (of course only majority class examples are removed (Batista et al., 2004). This indicates that elements with a high potential of differentiation are used and less discriminating ones are removed. Adaptions of the original algorithm, such as the well-known algorithm by Tomek (1976), improve the way how examples are picked from the source training set (ordered, close to the boundaries of classes etc.). All of these algorithms aim to reach as good results with the reduced set as with the complete data set (Tomek, 1976).

Hybrid techniques which use over- and undersampling in the same algorithm also proved to achieve good performance results (Maciejewski and Stefanowski, 2011; Batista et al., 2004). These combine for example SMOTE with CNN or Tomek links.

Data stream specific approaches to the class imbalance problem mainly concern algorithms for sampling data. To represent a data stream, which is unbounded in size, as good as possible is a complex problem, which also concerns the selection of training examples from this data stream. *Reservoir sampling* (Vitter, 1985) is an acknowledged method for this problem (Hall et al., 2009). Reservoir sampling maintains a set of size n (the reservoir) which is initialized with the first n elements of the stream on start up. The algorithms randomly pick data from the data stream and substitute elements in the reservoir. Its properties are that the data is processed in one pass and true randomness (Vitter, 1985). Algorithms mainly differ in the way the samples are selected and substituted, and if the size of the reservoir changes (Al-Kateb et al., 2007). (Babcock et al., 2002b) introduce two different algorithms to sample from a sliding window over a stream. One approach is *chain sampling*, i.e., reservoir sampling, but with removing elements which expired in the tuple-based window and adding the newly arriving element instead. Those samples would only represent recent stream history, which is suitable for data stream mining especially in view of concept drift, but will only work for tuple-based windows (Babcock et al., 2002b). For time-based windows they propose *priority sampling* which assigns randomly a priority to each new window element. In the sample the non-expired element with highest priority will be included. A further common approach is Bernoulli sampling and its extensions (Gemulla et al., 2006). Each new element is inserted in the sample with a certain probability. The sample size is random as it is determined by the binomial distribution and has no upper bound. Finally, also specific ensemble learning algorithms help to cope with class imbalance (Hoens et al., 2012). For example, decision tree algorithms which grow multiple trees specialized on distinct classes and intensify learning on wrongly classified elements (Hashemi et al., 2009).

4.4.4 Concept Drift

Concept drift is a specific problem of data stream mining. Data streams are potentially unbound in size. A hypothetical function creating elements for the stream may change over time. This can lead to (1) a change in the a-priori probability of a certain class, (2) a change in the probability for the membership of a data element in a given class (class distribution), or (3) given a certain element, that the probability to be in a certain class might change (posteriori probability) (Hoens et al., 2012). These cases all may lead to a performance decrease of a classifier as it has learned to classify elements according to former content of the data stream. This change in data distribution is called *concept drift* (Tayal, 2005; Hoens et al., 2012). As the hypothetical function and the assumed change in distribution are not known (Hoens et al., 2012) the only way to tackle concept drift is to detect it and adapt the classifiers accordingly. (Tsymbal, 2004) distinguishes different types of concept drift. First, concept drift can develop suddenly or gradually. Second, real and virtual concept drifts can be distinguished, depending on which of the three probabilities may change, but both lead to a necessary change in the model (Hoens et al., 2012). Concept drift in combination with class imbalance is even more problematic as a concept drift in the minority class might never be detected as elements occur rarely (Hoens et al., 2012). Algorithms coping with detected concept drift are adaptive learning algorithms, which are able to change the model when drift occurs (e.g., the CVFDT algorithm (Hulten et al., 2001), ensemble learners which create alternative models to compensate drift (e.g., the one-versus-all ensemble decision tree algorithm by Hashemi et al. (2009)), or algorithms adapting the training set (e.g., FLORA by (Widmer and Kubat, 1996)) (Hoens et al., 2012). We will not go into details of concrete algorithms, but will describe these where they have been used in this work.

4.5 Conclusion

This chapter introduced data streams and the most important basic concepts. Emphasis was put on architectures, systems, and query languages for relational DSMS. These basics help to understand the specific requirements for a process model to design, implement, and evaluate data stream applications. We will discuss existing process models for the design of information systems and present our own data stream-specific and quality-oriented process model in Chapter 5. Furthermore, we introduced the field of data quality management and corresponding management models for information systems. None of the existing models are directly applicable to the data stream setting as often manual effort is involved. However, basic ideas can be used to create a methodology specifically for data streams, whose design we will present in Chapter 8 along with the description of a data quality management framework implementing the methodology. We further introduced the field of data stream mining. We highlighted classification algorithms and the specific problems of class imbalance and concept drift. It turns out that it depends in general very much on the application at hand which algorithms can be used to produce satisfying results. A structured way has to be found to elaborate which algorithms suit which task in a data stream application best. To this end, we also discuss the applicability of process models for data mining in Chapter 5 and show, that they are not sufficient for the data stream case. Furthermore, we use the proposed evaluation framework from Chapter 5 and the data quality management framework from Chapter 8 to experiment with different data stream mining algorithms in the field of C-ITS in Chapter 10.

Part II

Part II: Process Model & Applications

Chapter 5

Process Model for Development and Evaluation of Data Stream Applications

The design and implementation of stream-based applications involve many aspects. They differ from the design of non-streaming information systems in many ways and we will discuss each design aspect in detail. One aspect is data management. Data has to be retrieved and processed. The input streams have to be modeled and corresponding continuous queries to be defined to model the process flow of data within the application. As queries can be registered and de-registered during run time, and streams might change their schema, the classical conceptual modeling techniques might not work anymore (Roussopoulos and Karagiannis, 2009). Applications are more and more data-driven and hence, data models and management methods should be too (Brodie and Duggan, 2014a,b).

Furthermore, if learning is involved, also aspects of data mining have to be considered, modeled, and implemented taking data stream specifics into account. Also, data quality must be an integral part of the design and DQ requirements should be defined right from the start, because stream data sources are often prone to errors (e.g., readings from sensors). For data streams, data quality monitoring and improvement must be executable at any time, on different levels, and in a timely manner, which makes it even more challenging. Finally, the evaluation of stream applications is very important. It has to be tested, if the application is doing what it is ought to do and if the quality measures actually reflect the quality of the measured instances. Due to those and further characteristics of data streams, a new or at least adapted process model, which takes these aspects into consideration, has to be defined. So far no stream application specific process model has been proposed - only partial aspects have been covered in literature.

In general, a *process model* can be defined as a description of which tasks have to be done to reach a goal (e.g., to design and implement a stream application). A *methodology* describes steps and means how these tasks can be carried out (Mariscal et al., 2010).

In this chapter we will first discuss existing process models for the design of information systems, data mining, and data stream management. Subsequently, we will propose our approach to a process model specifically designed for the creation and evaluation of data stream applications. We will discuss each of the steps in the process model in depth: Requirements Analysis, Design & Implementation, and Configuration

& Evaluation. For each of them, we will describe aspects to be considered, the outcome of the steps, and methodologies, how these steps could be executed in practice. We can only recommend methodologies - of course other methodologies might be applicable as well. The core of the Design & Implementation step is the proposal of a framework, which comprises the main components for the implementation, configuration, and evaluation of data stream applications.

5.1 Related Work

Process models for the creation of information systems have a long tradition and evolved over time. In the following we discuss the models which have been proposed for Information Systems in general. Subsequently, process models specialized on data mining and knowledge discovery applications are described. Although no complete process model for data stream applications is known, approaches for specific aspects of the design of data stream applications will be covered.

5.1.1 Process Models for Information Systems

A classical database application comprises usually one or more databases, software accessing these databases, and possibly a Data Warehouse. This kind of architecture strictly separates the data management design from the operational or process design. Data is sitting more or less statically in the database, applications send queries to retrieve the data for their purposes and produce results based on it. For both aspects research has come a long way and approved methodologies exist. One of the first structured process models for databases has been described by Teorey and Fry (1982). It is divided into four steps: requirements formulation and analysis, conceptual design (creating entity-relationship (ER) models), implementation design (transferring the ER-model to a specific database), and physical design (efficient implementation of the data processing) (Teorey and Fry, 1982; Kemper and Eickler, 2009). The approach also describes a methodology to fulfill the steps. These steps can be iterated to improve the quality of the resulting system. Because this process model is mainly tailored to relational database systems, the corresponding methodology steps, e.g., the requirements specification, focus on finding and modeling entities, attributes, and relationships between entities. These are covered in the information structure requirements. An interesting approach by Zachman (1987) describes a process model which utilizes three concepts to describe an information system, namely *Material* (i.e., data), *Function* (i.e., process), and *Location* (i.e., network model, distribution). These three aspects are also expressed by the questions "What, How, and Where". Along these aspects a process model for the creation of information systems is developed. They present alternatives to fill the steps by methodologies using a second dimension, which represents a view, e.g., the view of the designer, owner, or builder of the system. Each view could be regarded as one step in the process model. This process model is highly interesting as it also could apply to data stream applications though it has been created in times where relational databases came up. The distinction between the modeling of data, processes, and locations makes also sense for data streams, as all three aspects have to be considered individually, but integrated in the end.

Another very famous process model is the CASE method introduced by Barker (1990) consisting of seven phases: Strategy, Analysis, Design, Build, Documentation, Transition, and Production (Langer, 2007). It also focuses on relational information systems. All phases are build on the Strategy phase in which a Strategy Document is built. This document is structured along several aspects, such as logical data modeling, business process modeling, system boundaries (e.g., interfaces to other systems), and project management aspects (e.g., budget). The following phases refine this document and the information system is designed and implemented along the documentation. This method mixes the technical aspects of creating an information system with general project management tasks. It has many advantages to have a holistic view on a project and consider technical and project management aspects in a process model. The technical design aspects of this approach in some extent can also be useful for data stream applications. However, the project management aspects will not be covered here.

In times of big data the quite static modeling of data management in beforehand and the creation of a conceptual model, which might change once in a while, is no longer applicable. Schema-before, schema-after, and schema-never, are concepts which might not be timely anymore for new application concepts. Instead continuous modeling is advised (Roussopoulos and Karagiannis, 2009). This is especially true for data stream applications. Data sources, continuous queries, and operators are orchestrated to a stream application and might change dynamically by registration and de-registration during run time, depending on the system. Furthermore, process and data are not strictly separated anymore. Due to the data-driven nature of today's applications Brodie and Duggan (2014a) opt for the use of ensemble models, i.e., using different models for different kinds of information systems. Hence, the modelling of data stream applications has to be different from the classical relational modelling although we focus on relational streams here.

For Data Warehouses (DWH) Jarke et al. (1999) proposed to model activities for the development of DWHs in a meta-model. They employ a formal modelling process where all components of a DWH, including static components, processes (dynamic components), and data quality are stored and managed in a metadata repository. A process model, called Information Engineering, has been defined by Martin and Finkelstein (1988). They distinguish four stages: information strategy planning, business area analysis, system design, construction with encyclopedia (repository with information about all phases, dictionary and plans, models, and design tools) constituting the heart of the approach. Finkelstein (2006) proposed a business-driven variant of Information Engineering, which includes active participation of business experts in analysis and design. It consists of three technology-independent phases (strategic business planning, data modelling, process modelling) and two technology-dependent phases (system design, implementation). The Information Engineering process model is very business-oriented and focuses on business processes. This is totally fine for the information systems and their ecosystems, where task and data can be clearly separated. For data-intensive, data-centric applications, where business and data processing are fusing, these process models might not be adequate. Another process model focusing on the business view of an information system, is the Architecture for Integrated Information Systems (ARIS) (Scheer, 1992) and its extension, the ARIS-HOuse for Business Engineering (HOBE) (Scheer and Nüttgens, 2000). Similarly to the Information Engineering approach these focus on modeling business processes. The concept is to create information systems by gathering requirements, model business processes, and assemble existing software components (Scheer, 2013), accordingly. For data fusion, i.e., sensor fusion, a process model has been proposed by (Hall and Llinas, 1997). It concentrates mainly on sensor fusion and the tasks specific to sensor fusion. It can be used to implement

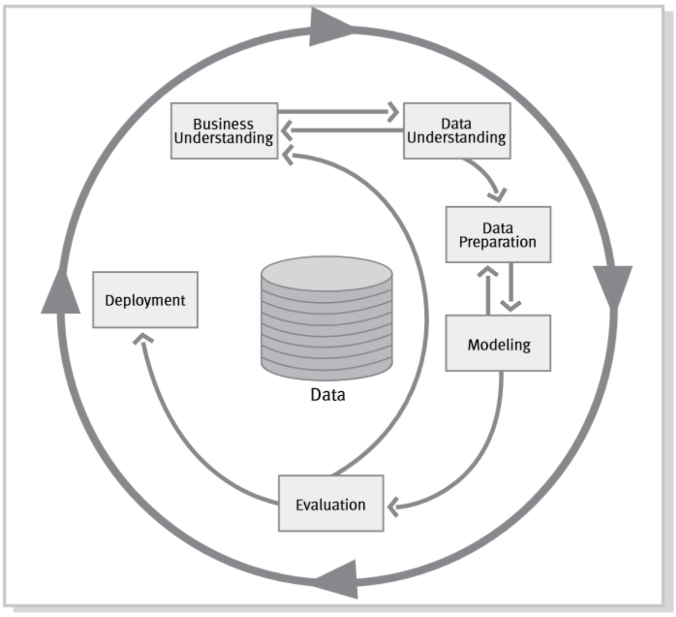


Figure 5.1: CRISP-DM Process Model (Chapman et al., 2000)

data stream applications dealing with object and context modeling (Kuka et al., 2013), but cannot be generalized for data stream applications per se. The quality aspect is sketched briefly by the process refinement component, which should monitor the performance of the fusion, identify information for the improvement of the fusion results, and allocate other sensors and sources. The authors also propose techniques which can be used for each step.

5.1.2 Process Models for Data Mining

Data stream application design is closely related to the design of data mining applications and also overlaps in multiple cases. Hence, it is fruitful to investigate methodologies and process models in the area of data mining. One of the most famous and most often used process models for data mining is the CRISP-DM (CRoss-Industry Standard Process for Data Mining) (Mariscal et al., 2010). It was proposed in 1996 by mainly four companies, amongst others Daimler Benz, and has been officially published in 2000 (Shearer, 2000; Chapman et al., 2000). As depicted in Figure 5.1, it comprises six steps, which are conducted in cycles iteratively: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment (which has no point of return).

It emphasizes the importance of a thorough analysis of requirements and knowledge of the task, data, and domain. It mainly addresses static data sets and also considers data quality, fusion, integration, and formatting of data. Hence, some of its concepts can be useful for a process model for data stream applications, but have to be modified to be applied in a streaming scenario. Most of the objectives defined for the CRISP-DM model are also true for the design of data stream applications, such as “insurance of the quality of results, reduction of skills, integration of experience for reuse, general purpose, robust, and tool and technique independent” (Mariscal et al., 2010). Many other process models are based on CRISP-DM, such as the Guerilla Analytics model presented by Ridge (2015). Guerilla Analytics is a data analytics approach which assumes (Ridge, 2015)

- a very dynamic project environment (change in data, requirements, resources)
- team with limited resources (time, tools)

and requires reproducible, transparent, and traceable results. It consists of the steps Extract, Load & Receive, Analytics, Consolidate, Reporting, and Work Products. This approach is very interesting with respect to data streams as it is more iterative than the classical CRISP-DM. In Guerilla Analytics the process has to react to new requirements and data fast, it has to deal with multiple data inputs. Hence, the process allows to get back to any of its steps at any time. Furthermore, it emphasizes the evaluation of the analyses results which corresponds to the CRISP-DM model. Guerilla Analytics has many characteristics which apply to data streams, such as fast changing data and multiple inputs. Hence, some of the ideas in the process model could be interesting for a process model for the creation of data stream applications.

The second very famous but less used model is the Knowledge Discovery in Databases (KDD) approach by Fayyad et al. (1996). It describes several steps for knowledge discovery, but is concentrating very much on the actual data mining process and is less general described to reflect the data management and analysis process required for data stream applications. Hence, we will not further describe it here.

5.1.3 Discussion

In a nutshell, there exist approved and famous process models for information systems in a broad sense (including data mining and knowledge discovery). Most of them address relational and static data sources. For data stream applications several specifics have to be considered, which have not been covered so far by existing process models. Furthermore, the evaluation and data quality aspect has to be integrated which is considered only in some of the discussed process models or as separate process models (cf. Chapter 8). Hence, we propose a process model for data streams in the following, which is based on approved methods, but extends them by aspects for stream processing, evaluation, and data quality.

5.2 A Process Model for Data Stream Applications

We propose a process model that can be used to evaluate and create streaming applications for arbitrary domains. For complex streaming applications a testing and evaluation phase is necessary, as the efficiency and effectiveness of a stream application is dependent on many factors. These must be revealed before the actual application can be set to a productive state. In traffic applications, for example, the output of an application might be dependent on the number of vehicles and the quality of the data. Or a health application might be dependent on the type of data available to get desired answers. Hence, we propose an evaluation-based meta process model which consists of:

1. **Evaluation Phase**: In this phase the dependencies and resulting configuration of an application are investigated.
2. **Configuration & Productive Phase**: Based on the results of the evaluation phase the application is configured in the productive phase. The application

owner has a good understanding of the conditions under which results are produced.

Each of the phases can be organized along a detailed process model. This is described in the next section.

In general, we distinguish three ways how a data stream application may be approached and created.

1. **Completely data-driven, explorative:** A huge amount of data is produced, e.g., in social media, by sensors, computer networks, or during stock trade. The data is explored to get new insights and discover new aspects about the domain, but there is no concrete task to be solved. Examples are applications using Data Lakes (Dixon, 2010, 2014), where raw data is gathered along with its metadata and the information needs are developed and issued as queries by users over time. Hence, we call this kind of application completely data driven - it deals with the data available and offers space to explore the data and experiment with it.

2. **Partly data-driven, fixed goal:** In many cases the goal of an application is very clear, but the data sources dictate the data and parameters available to the application. Many of the data sources are only available in a limited scope or not available at all, e.g., for a specific geographic area or in a certain time period, or for a specific group of people. In these cases the data determines what and how results are produced and maybe new ways to reach the goal have to be found based on the available data. Sometimes, the goal can even be only partially reached or only in a limited way.

3. **Application-driven, fixed goal:** In these cases, the application goal is also very clear and the developer is not limited in her choice of data sources to reach the goal. Hence, the application design particularly includes also the selection and modelling of data sources. Constraints regarding, amongst others, costs, volume, or frequency have to be considered of course. This will be detailed in the following description of the process model.

Depending on the chosen way the steps and their order in the process model may differ. An overview of the proposed process model is depicted in Figure 5.2.

The process model for the evaluation phase, as most of the other information system design process models, starts with the requirements analysis.

5.3 Requirements Analysis

One of the foremost activities which have to be done to create any information system is the analysis and description of the target domain and application. Data management is never done for its own sake - it always serves a higher goal in a specific domain. Hence, a deep understanding of the field including its characteristics, processes, users, and quirks (all of which influence each other and belong together) is needed. RA and/or business modeling is an integral part of almost every process model, but it might differ in what is analyzed. Though in data stream applications data management and processes are not as strictly separated as in classical database applications, it makes sense

1. to have a look at which tasks have to be fulfilled without being biased by implementation details,

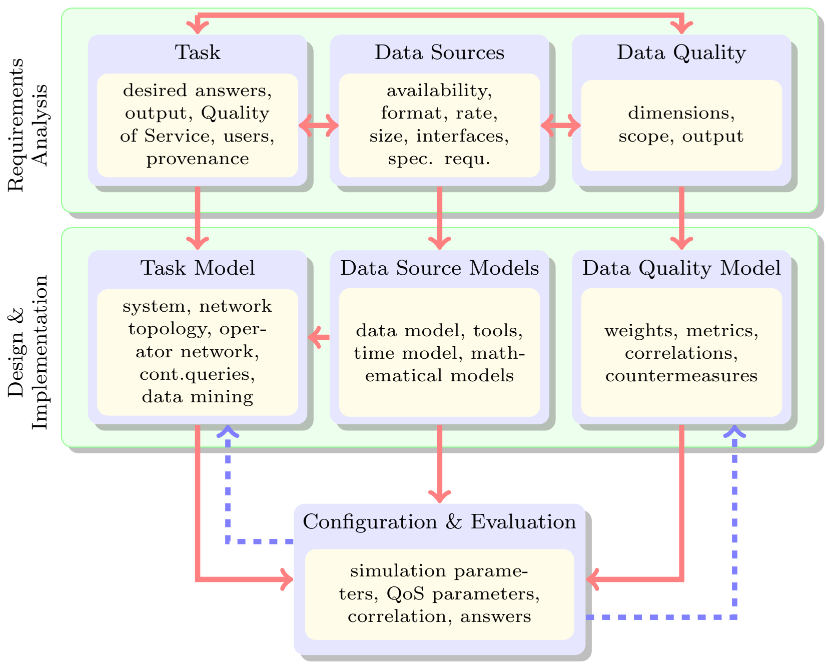

Requirements Analysis
Task
desired answers, output, Quality of Service, users, provenance
Data Sources
availability, format, rate, size, interfaces, spec. requ.
Data Quality
dimensions, scope, output
Design & Implementation
Task Model
system, network topology, operator network, cont.queries, data mining
Data Source Models
data model, tools, time model, mathematical models
Data Quality Model
weights, metrics, correlations, countermeasures
Configuration & Evaluation
simulation parameters, QoS parameters, correlation, answers

Figure 5.2: Process Model to Design and Implement Quality-affine Stream Applications

2. to see which data sources are required, which data sources are available, and what properties do they have, and

3. to define the quality requirements for the application to integrate quality measures and means right from the start.

Hence, we consider three different types of requirements in the process model that are vital for the design of data stream applications. According to the three approaches to an application defined in the beginning of the section the order of the steps in the requirement analysis may differ. A completely data-driven approach starts with the requirements for the data sources as all data is fed into the system and the task is not clearly defined. Task and data quality requirements may follow later and maybe less detailed than for other approaches. For the partly data-driven approach both steps - data source requirements and task requirements - are valid points to start at. The task-driven approach should definitely start with the task requirements which influence data source requirements substantially.

The requirements analysis does not only have an impact on what will be implemented in the end, but also which means are used for this. Some criteria, such as the data model, might limit the choice of a DSMS, the libraries used, or the data mining algorithms. The different parts of the RA influence each other and there will be several iterations of writing and updating the RA document. In a completely data-driven approach the available data influences the tasks. In a task-driven approach the task will influence the requirements of data sources. Depending on the data context the data sources will influence the data quality requirements as well as the task does. But also the requirements posed on the data quality have influence on the data sources, e.g., when a high accuracy for positions is required a data source using GPS coordinates might be preferred over mobile phone data.

In the following we will detail each step in the requirements analysis.

5.3.1 Task Requirements

It is quite common in process models for software and information systems engineering that the data management part is separated from modeling the business processes that should be reflected or carried out by the application to be built. A very well-known example is the ARIS process model (Scheer and Nüttgens, 2000). Data stream applications are networks fed by one or more producers and which in the end deliver output to consumers. Depending on the functionality of the system this could be the DSMS itself, a web application showing the data on a map, a database, or any other software. Hence, this is the point where the RA should start - namely at the end of the chain, i.e., we propose a goal-oriented approach for the task of RA, similar to the Goal-Question-Metric method by Basili (1992), a top-down approach for the definition of metrics for software measurement. First, specific goals for the measurement are fixed and questions are defined to help to refine the goal. Subsequently, metrics are elaborated, which help to answer the questions (Koziolek, 2008).

For stream applications, the following steps are suggested for an analysis of the task.

1. **Goal Definition:** Task modeling starts with the definition of the overall goal of the data stream application. Which problem does the user want to be solved? What information does she need? For example, does she want to have near real-time visualization of animal movements on a map or real-time estimations for energy prices which are fed into an ordering software? A description via UML use-cases is a possible implementation in a methodology.

2. **Output Definition:** The output is somewhat the "data goal" of the stream application. In the end the application produces again data that will be used in other applications, e.g., is visualized, further processed, or archived for later use. Hence, it must be defined what the output is, what is done with it afterwards, and who the consumers are. Here already a very detailed description of the output can be done. This includes for example, the format and the granularity of the output. For the animals movement example this could be position data with coordinates and additional data about the animal itself, e.g., a calculated speed or a heart rate.

3. **Quality of Service:** For the output it should be also very clear, which requirements the application should fulfill for the delivery of the output. At which frequency should the output be created and is there any delay allowed? What happens with outdated information? Are there any requirements regarding buffering or throughput? Load, latency, or bandwidth are only some of the possible QoS measures (Lakshmanan et al., 2008).

4. **Provenance:** For the choice of the DSMS it must be further elaborated, if data provenance is an important requirement of the application and for the users. For example in health monitoring applications this may be a crucial aspect. For streams provenance can be tracked on different levels. In stream applications the DSMS often already include some basic functionality for implementing provenance, e.g., adding a timestamp when the stream element has entered the DSMS, a timestamp by which operator the stream element has been created (please see Section 4.2.2 for a detailed discussion of timestamps), or the name of the operator which created the stream element. Beyond this basic information created

usually by one operator, there exist approaches to track provenance over several operators in the network (Glavic et al., 2013; Misra et al., 2008). For example, if aggregates or joins have been included as operators the resulting stream elements carry information, how they have been created.

5. **Users:** Finally, also the users and their usage of the final system and the data must be defined.

Outcome

The output of this step should be a detailed documentation of the task requirements, which can be integrated in an overall requirements document (the *requirements specification*). The documentation will be used for reference in the further requirements analysis process and in the design phase for selecting system components and building the operator network or designing the queries, respectively.

Methodology

Requirements analysis (RA) has been studied very extensively and we will not go into much detail about how RA can be done. Expert interviews, literature research, UML- and scenario-based variants of RA and documentation are widely known (Pohl, 2010). Interviews are conducted with project partners, domain experts, and potential users of the application. Especially, semi-structured interviews are useful, as they follow a predefined interview guideline. The guideline helps to acquire the desired information, but still leaves room for open discussions, which may bring up important insights (Hove and Anda, 2005). Furthermore, literature research is indispensable, as the domain has to be understood in depth and experiences from former and current projects are very valuable.

The results of the requirements analysis are usually documented in one big document (the *requirements specification*). There are basic elements which are always included, such as the status quo of the context, the unique numbering and prioritization of each requirement, and the distinction between functional and non-functional requirements. Visualizations, such as Use-case diagrams, help the readers to easily understand which functionality the application should provide and build a solid basis for discussions.

5.3.2 Data Source Requirements

One side of a data stream application has been covered already, namely the desired output of the application. The other side, the input, is equally important. In the introduction of this chapter we distinguished three ways to approach the development of stream applications. These are categorized along the availability of data sources. Hence, it makes sense to distinguish between the description of available data sources (status quo) and the requirements of desirable data sources. The existing data sources should be described first to see what is already there. Based on these descriptions requirements for new data sources are elaborated.

For the description of data source requirements approaches exist, which mainly refer to relational databases (Wang et al., 1993; Zachman, 1987; Barker, 1990). In these approaches requirements are retrieved from the users and modeled using a conceptual modeling paradigm, such as the Entity-Relationship (ER) model. Of course there exist also modeling approaches for other structured and semi-structured data sources. For

XML, modeling with DTD, XML Schema, UML, or the specific conceptual modeling languages for XML, such as XSEM (Necasky, 2007), can be done. For RDF or OWL sources ontology modeling techniques and visualizations can be used.

The data sources for a data stream application can be very diverse. Hence, a general description, which applies to different kinds of data sources should be used. All information which is important for the further handling in the DSMS must be gathered. The general description must include properties amongst others, the format, the size, interfaces, or the update frequency.

Outcome

The outcome of this step should be a detailed description of the general and semantic properties of each of the data sources, existing or required.

Methodology

For the RA analysis and documentation of data sources, we propose to use predefined forms, which help to remember the most important aspects for each data source and to collect that information in a uniform way.

We propose a form, called Data Source Documentation (DSD), found in Appendix B. It can be used for the documentation of existing data sources, but also for the elicitation of requirements for new data sources (or *required data sources*). We distinguish two aspects of requirements: functional and non-functional or semantic requirements. Functional requirements describe properties of data sources not connected with the content or meaning of the data, but about general information, such as the size or the format. Non-functional requirements describe properties of the data with respect to its semantics and content. Hence, the part General Description as well as the Schema Description part should be taken into consideration for them.

For required data sources the description might include a lot of blanks, because not all information is known or not relevant at the moment. But surely a structured description of the data sources will help to discuss them with project partners and customers. They may have a different view on it or an idea where to get the desired data. Definitely the requirements for the data sources also depend on the task requirements and description. In the design step the DSD can be utilized to model the data sources in a meta-data model.

5.3.3 Data Quality Requirements

As we have stressed earlier, data quality is a very important aspect in stream applications and helps to interpret the intermediate and final results of an application. Hence, the analysis of which DQ aspects should be considered should be an integral part of a requirements analysis for stream applications. In the following we propose aspects considerable for the data quality RA. There exist several approaches which guide software developers through the definition of DQ requirements.

The Goal-Question-Metric paradigm describes an RA for software measurement (Basili, 1992). It has to be stressed, that the GQM paradigm does not distinguish dimensions and metrics as done in this thesis, but combines them and terms the combination metric. The top-down approach defines goals and corresponding questions, whose answers, measured by metrics, should help to reach the goals. Wang et al.

(1993) propose a very detailed methodology for RA for data quality where they distinguish subjective and objective quality dimensions (both are DQ attributes). While subjective dimensions depend on the context and definition (e.g., timeliness), objective dimensions do not leave room for interpretation (e.g., the creation time of data). A conceptual model of the application, i.e., an Entity-Relationship model, is created and annotated with subjective and objective dimensions resulting in a data quality model. If there are several application views the resulting DQ models are integrated into one big schema. The Quality Function Deployment (QFD) is a management approach originally invented in the 70's in Japan. It is used for the design of products in many business branches, such as manufacturing of any kind. Hauser and Clausing (1988) proposed a tool called the "House of Quality" for QFD, which combines several QFD matrices to an overall matrix which looks like a house in the end. The basis of the house are user requirements (rows) and actions (columns) which help to fulfill the requirements. An additional column contains a number to rate the relative importance of the requirement. The content of the cells includes a value, which indicates the effect of the action in fulfilling the requirement. Besides of the requirements other rows can be added, such as the costs for the action (Hauser and Clausing, 1988). Redman (1996) applied the QFD matrices approach to determine data quality requirements. His approach transforms user requirements into data quality requirements and performance requirements in five steps.

As the design of the operator network has not been done yet, the approaches by Wang et al. (1993) and Hauser and Clausing (1988) are not applicable, but will be useful in the design phase. The approach by Redman (1996) is too static and operates on a too high level, to be adopted for our purposes. Hence, we will not further consider it.

In our approach, we again follow similarly to the GQM approach a top-down approach by defining the goals for the quality measurement first and afterwards considering, which dimensions should be measured, to reach this goal.

- **Goal/Output:** Similar to the goal of an application, it must be also defined which goal(s) the DQ measurement is aiming at. For example, one goal could be to detect failures in a production process or to analyze how much data of a sensor gets lost during a certain time period. After defining the goal(s) it has to be considered with which kind of data and dimensions these goals can be achieved. A concrete example could be a traffic application that detects a hazard on the road. The goal is to output the degree of confidence, that the hazard is actually existent, to filter out unreliable information and improve the user experience.

- **Quality Dimensions:** Once the goal has been defined, it is important to think about how the goal could be achieved and which data and DQ dimensions could help to reach it. Data quality can be measured on many levels (please refer to Section 4.3) and can have many manifestations. Examples for DQ dimensions are completeness, accuracy, or timeliness. We do not want to anticipate design and implementation in this step. At this stage it is not important to know exactly for which data which quality should be measured how. However, it makes sense to name and describe what kind of data quality dimensions are important for

  1. the domain,
  2. the application in question,
  3. and specifically for the goals described before.

The earlier such definitions are discussed and thought through, the better they are understood and agreed upon. A requirement analysis description should always include a glossary of domain terms which helps to see if project partners have the same understanding of the domain terms (Balzert, 2010). The DQ dimension description is exactly this: an agreement on the understanding what the dimensions mean. The interpretation of the dimensions and the metrics might vary from domain to domain and context to context, but also from object to object whose DQ is reflected by that dimension. Hence, the detailed definition has to be done in the DQ Model designed in the Design and Implementation step. It makes also sense to see if there might be already other applications in that context measuring DQ. For the concrete example of the confidence value for the hazard application the dimensions which are required to determine the confidence value have to be considered. Amongst others, surely timeliness is important as well as the amount of data used to detect the hazard.

Outcome

Basically, the outcome of this step should be:

- a description of the quality goals,
- a first description of quality dimensions.

Methodology

The quality descriptions of the outcome can be integrated into the overall requirements specification in a separate chapter for DQ requirements.

5.4 Design & Implementation

After the requirements analysis has been done, it is time to design the stream application. The design process could in turn lead to new requirements or the need to find out more about certain requirements. In accordance with the RA we propose the consideration of three main aspects. The first aspect is the modeling of the data sources. The availability of the right data is crucial for an application. Secondly, the modeling of the data processing to achieve the goals defined in the RA takes some time. Finally, the design of a data quality model helps to integrate DQ management into the data management process. In the following we will describe the three aspects in detail.

5.4.1 Selection and Modeling of Data Sources

Data stream applications need input data to be processed. Hence, the applications are dependent on the data sources they can use. Data sources are usually connected via adapters to a DSMS to convert and feed the data to the processing operators. The adapters can also regulate the data flow. Data sources can vary according to several properties. From the creation of the data until the reception and processing by the DSMS multiple aspects have to be considered. For the data source modeling we distinguish two aspects:

1. **Data Provisioning**
   The data sources, whose requirements have been described in the RA, are now selected. Depending on the data and task requirements, a corresponding DSMS must be selected. For example, if the most important data sources are RDF, XML, or binary streams, a different system may be selected than for data sources which can be broken down to relational streams and tuples. If provenance is required, this also has to be offered by the DSMS at the desired level. Furthermore, the decision whether a simulation is used or a real-world data source is used has to be made. We present a detailed discussion on this issue in Excursion 5.4.1. Additionally, costs of communication, latency, and any other issues regarding data transfer have to be considered and documented.

2. **Data Modeling**
   After the selection of the desired data sources, these have to be described as detailed as possible. After the general description each data source can be described semantically using the most appropriate modeling approach, or by using a meta-model such as the Generic Role-based Meta-model (GeRoMe) by Kensche et al. (2007).

Excursion 5.1: Real-world or artificial data?

A thoroughly created RA constitutes the basis for the right choice of data sources. From the requirements it must also be clear which kind of data is needed. Often the data sources are given by a project or the customer. Especially in research, it can be part of the problem to find out which data sources could help to solve the task. If all or some data sources are not given, the requirements may lead the way. For example, if several experiments have to be done and the parameters must be fully controllable, it is questionable if this can be done in a real world scenario. If the scenario contains parameters which cannot be controlled (e.g., natural observations, dynamical systems), a simulation or otherwise artificially created data sources can be used. On the one hand, if the application must be tested within a scenario as realistic as possible, or simulations are not existent or difficult to create, the use of real world data sources is advised, but not always possible. There may be online sources available, but they might not contain all the data needed for an experiment or application. Or there are fixed data sets available, but these do not change and only represent a snapshot of the scenario needed – limited in time and space and might not be fully reproducible. On the other hand, simulations also involve several difficulties and issues which might not be desirable. Simulations are very complex and often only creatable by very costly software. One has has to fully understand the underlying mathematical and physical models. Furthermore, all configuration possibilities must be known. This means a high effort in learning how to use the software.

Again not all desired data might be available or may not be creatable in simulation. And finally, saying it with Magritte: "Ci n'est pas une pipe." (This is not a pipe. He meant, that it is only a painting of a pipe, and not a real pipe.) In the end only the real world is real and a simulation is only an approximation of this world, not exactly the world itself. The creation of simulation scenarios is a too complex topic to be discussed here in detail. It depends highly on the domain and application. We will describe the consideration, selection, and modeling of a simulation data source for the traffic systems domain in Chapter 6. For the design and implementation a very detailed model and experimental setup plan and documentation has to be made, to exactly know the relevant parameters and the mathematical and physical models behind the simulation. After designing the simulation, this may serve as a data source and its output can be handled as any other data source. A combination of both, real world data, and artificial data is also possible. Interpolation of data or creation of data based on empirical knowledge might also be a feasible solution. But this has to be considered carefully, because data are falsified and experiments based on it can easily be criticized. For the case of missing data features another possibility is the integration of data from different data sources. But this is also not easy and the pros and cons have to be carefully weighed. For further reading, we recommend the thorough and acknowledged discussion of which experimental setting is applicable to a given problem at hand for information systems found in (Benbasat et al., 1987).

Outcome

The outcome of this step is

- a selection of data sources,
- a detailed description of the general properties of each data source (format, name, size etc.), which completes the description from the RA,
- a detailed description of the underlying specifics of the data source, e.g., mathematical model, simulation parameters, if applicable,
- and a data model for each data source.

Methodology

As a follow-up of the requirements analysis, the Data Source Description found in the Appendix B can be extended and completed to design and describe the required and existing data sources. As explained earlier, it depends very much on the type of data source, which modeling means are used to visualize and describe the data of the data source.

5.4.2 Task Modeling & Implementation

After data sources have been selected, modeled, documented, and prepared for data processing, the actual design and implementation of the task process model has to be done. In the RA one important thing was to have a look at the output - i.e., the actual data which is produced in the end. It has to be mentioned right from the beginning, that many design and implementation issues and restrictions (as with any other information and software systems) are dependent on the system and underlying language at hand. For example, while several systems offer the possibility to design operator networks hiding query language details from the designer, many rely on the manual definition of queries or even require implementation of code. We refer the reader to Chapter 4 to compare the different systems and languages.

Hence, it makes sense to have a sharp look at the requirements and the chosen data sources and data model, and search for aspects which allow only for the selection of specific systems. Partly, this has been done in the last step. But that consideration should not only involve the required data models, but also time and processing models. For example, when should the system in general proceed with processing? When a new element arrives or on the next time tick? Additionally, Quality of Service (QoS) has to be considered for the DSMS selection. Previously, requirements for QoS have been defined, which have to be fulfilled by the system and the final application. The target DSMS has to provide means to measure and adapt QoS in these regards.

Furthermore, the corresponding adapters for the data sources must be designed and implemented if they are not already included in the system. Finally, these are customized for the concrete data sources. In the next step the operator network, corresponding queries, and other specialized processing steps, such as data mining, have to be defined. It is advised to test the stream application after adding a new step to see if the resulting stream is delivering results as expected.

Outcome

The outcome of this step should be:

- a selection of the DSMS,
- an implementation of adapters for the corresponding system and data sources,
- an implementation and first configuration of the operator network,
- a detailed description of the network, e.g., using an operator network diagram,
- descriptions of applied data mining algorithms and preliminary configuration,
- and a detailed description of the distributed network, if applicable.

Methodology

Depending on the system at hand this design step could be visualized by an operator network diagram which describes the changes of the stream schemata at each single step. Not only the purpose of each step but also the incoming and outgoing streams must be described. This includes, amongst others, the description of windows, aggregates, and sampling. Additionally, also the expected data rates and values should be documented. Such a network description does not only document the data processing, but helps again to communicate with project partners and customers and to compare with results in tests of the running application. In the textual description the detailed queries and algorithms can be explained. If the chosen system is distributed a network diagram can be constructed.

To implement the stream applications, we propose a framework which includes the following main components:

- A **Data Stream Management System** which includes a **Data Processor** to enable the processing of data streams and queries,
- Comprehensive **Data Mining** facilities, either integrated in the DSMS or the DSMS should offer means for extension (e.g., enable use of a library). Data Mining is a crucial part of many data stream applications and hence, a framework should offer the corresponding functionality to achieve this requirement.
- **Data Quality Management** components to enable easy integration of DQ monitoring into the data processing,

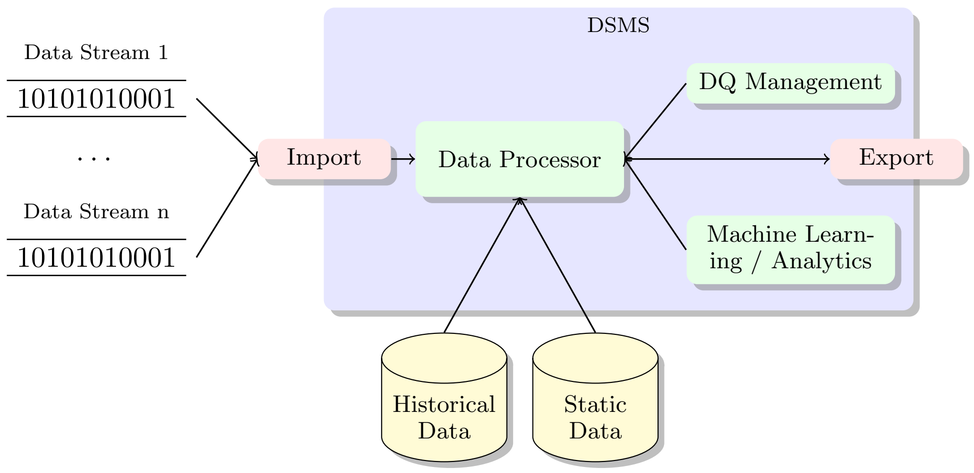


Figure 5.3: A Framework for the Creation and Evaluation of Data Stream Applications

- **Adapters** for streaming as well as non-streaming data sources,
- **Export** functionality to provide data to other systems and users.

An overview of the proposed framework is depicted in Figure 5.3

5.4.3 DQ Modeling & Implementation

Data quality management should be an integral part of each stream application. Sensor data has often a minimized data quality, due to hardware, software, environmental, or communication problems (Paradis and Han, 2007). Results based on this data are influenced by its quality. To rate the value of the produced information, its quality has to be determined. After the initial requirements for DQ for the application have been inquired, it is time to model it for the concrete stream application. By now, data sources and data management processes should be clear, and streams, operators, and data processing plans defined and implemented. Now, for each of the involved data objects it must be determined if data quality will be measured and which dimension will be measured how. Not later than at this point, each dimension needs a concrete textual definition. Because dimensions might be reused in different contexts they can have different meanings (Redman, 1996; Wang et al., 1993). Hence, it is important to distinguish dimensions using unique names and describe their purpose in detail. For each combination of dimension and data object, a metric must be defined, which allows for the measurement of the dimension. For each combination of dimension, data object, and metric a weight can be defined.

Outcome

The outcome of this step should be

- a mapping of dimensions and metrics to data objects described by a DQ model,
- a detailed documentation of dimensions and metrics,
- a weighting schema for dimensions and data objects.

Table 5.1: Metrics Matrix

	Dimension1	**Dimension2**	**Dimension3**
Streams			
Stream1		Metric2	
Attributes			
Attribute1	Metric1	Metric2	
Attribute2	Metric3		Metric4

Table 5.2: Weighting Matrix

	Stream Object	**Dim1**	**Dim2**	**Dim3**
CombDim1	Attribute1	Value	Value	Value
CombDim2	Stream2	Value	Value	Value

Methodology

In the discussion of the DQ requirements we already mentioned the House of Quality (HoQ) (Hauser and Clausing, 1988) as an appropriate means to elaborate and document requirements. But in the process model we propose, it was to early to apply it in the requirements step, as the concrete operator network and streams have not been known. At the current step we can now proceed with designing the DQ. As described in the previous section, the HoQ consists of several matrices which can be combined to create an overall DQ overview (the house). These can easily be understood by partners, customers, and users from other domains and help to agree on the DQ part of data processing. For data stream applications, similar to the HoQ, the following matrices are proposed to understand the user's and application's needs:

The *Metric Matrix* has a row for each stream object, such as whole streams or attributes of streams in the relational case. Each dimension is represented by a column. For each stream a separate matrix can be used to keep it manageable, but the different levels of objects, such as the whole stream, windows, and attributes, should be denoted by different categories in the matrix. In the cells the corresponding metric can be noted, subsequently. This allows for a quick overview of what dimensions are measured for which element and how. An example structure is given in Table 5.1.

A *Weighting Matrix* documents combinations of dimensions for each element and the weighting of each of the dimension in the result. An example structure is shown in Table 5.2.

Finally, in an *Action Table* and **Counteraction Table** for each stream object and dimension, actions, and counteractions, respectively, can be documented. Both have the same layout. An example for the Action Table is given in Table 5.3

This knowledge can be additionally modeled in a DQ model extending the operator

Table 5.3: Action Table

	Dimension1	**Dimension2**	**Dimension3**
DQ_Att1	Action1 + Rule1	Action2 + Rule2	
DQ_Att2		Action3 + Rule3	

network model of the data stream application by dimensions attached to the streams and attributes in the network model. After documenting the DQ design in the matrices, this knowledge can be implemented in a machine readable format easily to make it (re)usable for a later DQ processing. The design can be seen as a kind of documentation in this case and of course both representations should be kept synchronized. For the implementation a conceptualization, such as an ontology or a database, can be used. Important features for the conceptualization are relationships between objects and reusability of, amongst others, dimensions, metrics, and the data objects themselves. We describe our implementation of the DQ management in Chapter 8 in detail.

5.5 Configuration & Evaluation

After the stream application, including data quality management, has been implemented, this setup can be used to carry out experiments. A detailed experiment plan helps to organize these activities. A description of the purpose and the expected results needs to be elaborated. As data sources (e.g., simulations), the stream application, DSMS, and the DQ management can be configured in many ways, all configuration parameters must be documented for each experiment. The evaluation of the results must also be documented and the resulting data ideally has to be recorded to allow for a comprehensible end-to-end evaluation. Of course the extent of the experiments depend on the kind of application and task. Parameters which might be configurable are from the following categories:

- Simulation parameters
- System parameters, e.g. fixed sampling size, QoS parameters
- Data management parameters, e.g., window sizes, aggregates
- Algorithmic parameters, e.g., for parameters for data mining algorithms
- Data Quality parameters, e.g., weights

Experiments which rely on random variables should be carried out several times with different seeds for the random variables. For the results of these runs the mean can be calculated, which is in the end more representative than individual measurements. If the evaluation results are not meaningful enough, this indicates issues in the processing chains, or certain goals have not been reached. For example, if the performance is low, this might lead to a reconfiguration or even to changes in the task, data source or, DQ model. This again requires a new set of experiments.

Outcome

The outcome of this step should be

- a detailed documentation of experiment setups and configurations,
- a rating of the application results based on the DQ measurements,
- and a result data set for evaluation.

Methodology

For the documentation of the experiments it is advised to manage something like a laboratory book: each experiment is noted down with its date, configurations, data in, data out, and everything noteworthy for the experiment. Depending on the data at hand, coded tests, similar to unit tests, can be written (Ridge, 2015). These compare the expected data with the actual outcome. For data streams this is not always easy, as the results might not exactly be the same for each run. Influences, such as system load or communication speed, might cause slight differences in the results. For example, windows always shift at the same time, but they might include different data, if this does not arrive exactly as it has arrived in the last run.

5.6 Conclusion

We have presented a proposal for a process model for the development of data stream applications. Existing process models do not satisfy all the needs of data stream applications. In particular, the demand for evaluation and data quality assessment for data stream applications has only been addressed unsatisfactorily. We distinguished three different types of how to approach the development of stream applications. The different steps of the process model, their relationships to each other, and possible recurrences have been delineated. Furthermore, for each step the desired outcome and potential methods to execute the step have been explained. But how can a process model be evaluated? Commonly, this is done by applying it to case studies and gathering experiences from developers using it (Wirth and Hipp, 2000). Also, case studies are an appropriate and acknowledged means for the evaluation in information system research (Runeson and Höst, 2009; Benbasat et al., 1987). Case studies must fulfill certain properties to be suitable for the problem at hand (Runeson and Höst, 2009). To be suitable for the evaluation of the proposed process model, we must show its applicability, flexibility, and generalizability to different cases. Hence, the case studies should

- have different application goals in different domains,
- have different starting points for the choice of data sources (fixed, flexible),
- comprise typical data sources for the domains including both, simulated and real-world data,
- vary in requirements, e.g., according to latency and data quality,
- and involve complex and domain specific algorithms.

To this end, in the next two chapters we will show the application of the model to four different case studies from two different domains. First, we apply the model to the applications of Queue-end Detection and Traffic State Estimation in the C-ITS domain in Chapter 6. Both case studies are very well suited to study the effectiveness of the process model and the proposed methodologies, particularly the evaluation framework. They use real-time Floating Car and Floating Phone Data, but for different purposes and with different properties. QED is a traffic safety application where latency and quality are crucial. The results serve as incident warnings and hence, a high confidence of the produced information is required. TSE is dealing with more data from many vehicles and covers a larger area, but is more relaxed according to latency and quality. Both applications require complex algorithms for their implementation and many system and environmental factors may influence the overall result of the applications. Furthermore, crucial questions, such as how much data is required for satisfying application results, must be elaborated, for which a structured evaluation using the proposed evaluation framework is helpful.

In Chapter 7 the process model is applied in two case studies from the mHealth domain. Both involve real-time sensor readings from sensors worn at a human subject's body. Data is coming from a single or multiple users and sensors. The quality of the data varies a lot as subjects move and mobile communication and positioning may be prone to failure. The applications are less complex than the C-ITS case studies but show the easy use, flexibility, and generality of the model and also require quality assessment.

Chapter 6

Validation of the Process Model in the C-ITS Domain

We already gave a detailed introduction to C-ITS in Chapter 2 and stressed that Vehicle-2-X applications became a very important topic in the area of public and private transportation in the last decade. We will evaluate the process model and evaluation framework elaborated in the previous chapter intensively in the context of the project Cooperative Cars (CoCar)[1] and its successor Cooperative Cars eXtended (CoCarX)[2]. In the following, we will first describe the projects to build the foundation for a requirements analysis. Afterwards we will describe a walkthrough of the process model step by step explaining our realization of the steps.

6.1 The Cooperative Cars (eXtended) Project

There exist a multitude of research projects and initiatives, which investigated or are still investigating opportunities of C-ITS using a multitude of technologies. Popular examples of recent and completed projects comprise the Cooperative C-ITS Corridor[3], the European Comission C-ITS platform[4], CIMEC[5], Car 2 Car Communication Consortium[6], Drive C2X[7], MYCAREVENT[8], and CVIS.[9]

CoCar and its successor CoCarX were subprojects of the project *aktiv* (Adaptive and Cooperative Technologies for Intelligent Traffic) funded by the German Federal Ministry of Education and Research Germany (Bundesministerium für Bildung und Forschung, BmBF). Key players from the area of transportation and communication, such as Daimler AG, Ford, Volkswagen AG, MAN Nutzfahrzeuge AG, Vodafone Group R&D Germany, Ericsson, and PTV AG, but also research institutes, such as Fraunhofer FIT, Informatik 5 at RWTH Aachen University, and BASt (Bundesanstalt für Straßenwesen, German Federal Highway Research Institute), were involved in different stages of the project. The projects investigated the suitability of UMTS technologies

[1]`http://www.aktiv-online.org/english/aktiv-cocar.html`
[2]`http://www.aktiv-online.org/deutsch/aktiv-cocarX.html`
[3]`http://www.c-its-korridor.de`
[4]`http://ec.europa.eu/transport/themes/its/c-its_en.htm`
[5]`http://cimec-project.eu`
[6]`https://www.car-2-car.org`
[7]`http://www.drive-c2x.eu/project`
[8]`http://www.fir.rwth-aachen.de/forschung/forschungsprojekte`
[9]`http://www.ecomove-project.eu/links/cvis`

and their extensions (LTE) for the direct, targeted transmission of traffic data arising from both, stationary and vehicle-based sensors. Furthermore, the efficiency and effectiveness of mobile communication for real-time traffic applications has been investigated, e.g., to see if the transmission delay is small enough to enable warnings for queue-ends or collisions at crossings (Fiege et al., 2011).

In the course of the projects a complex modular system architecture has been elaborated to implement not only the requirements mentioned before, but several other services, such as infotainment, flexible billing, or security and privacy aspects (Fiege et al., 2011). The overall CoCar system architecture is depicted in Figure 6.1.

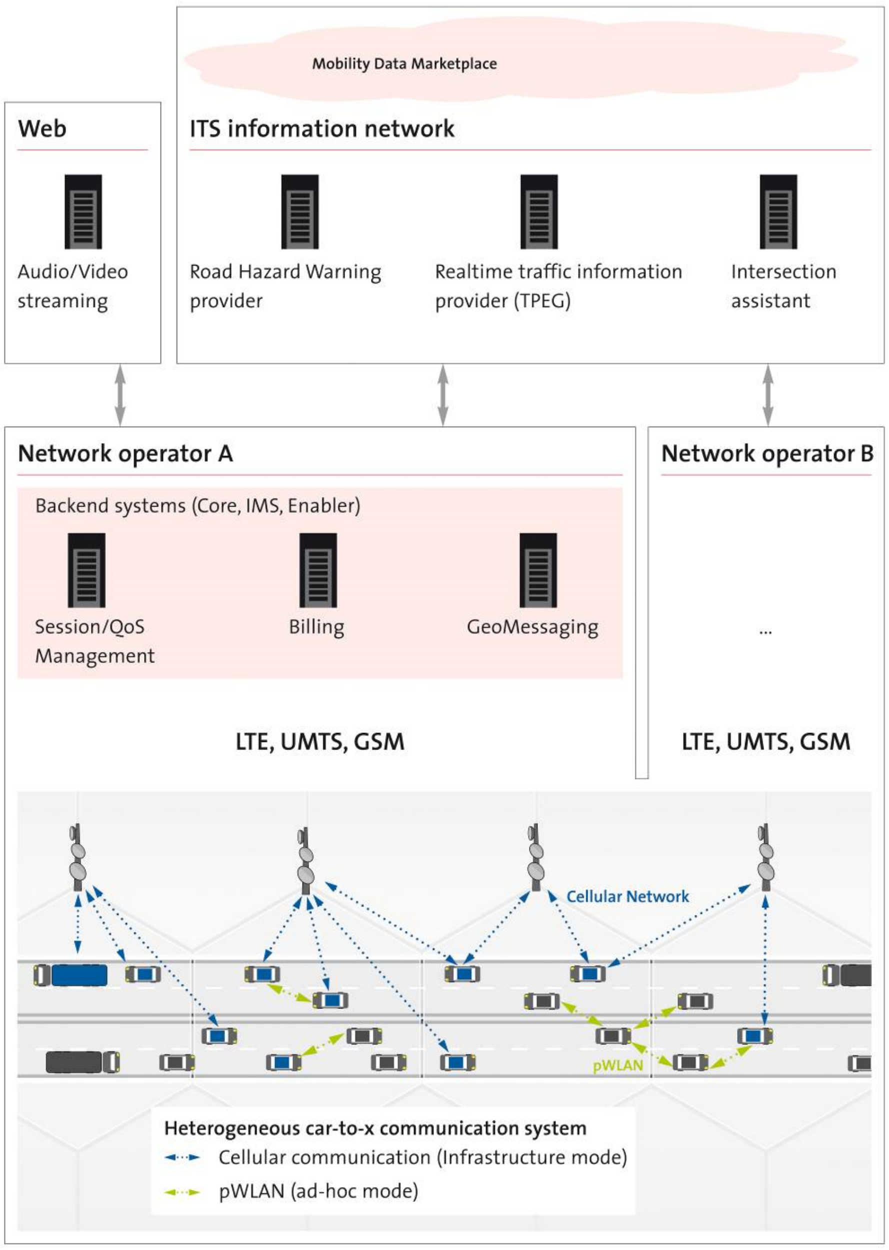

Mobility Data Marketplace
Web
Audio/Video streaming
ITS information network
Road Hazard Warning provider
Realtime traffic information provider (TPEG)
Intersection assistant
Network operator A
Backend systems (Core, IMS, Enabler)
Session/QoS Management
Billing
GeoMessaging
Network operator B
...
LTE, UMTS, GSM
LTE, UMTS, GSM
Cellular Network
pWLAN
Heterogeneous car-to-x communication system
Cellular communication (Infrastructure mode)
pWLAN (ad-hoc mode)

Figure 6.1: CoCarX Architecture Model (Fiege et al., 2011)

In the lower part of the figure vehicles drive along a road and communicate with each other and the infrastructure via cellular networks and pWLAN (ad-hoc wireless network according to IEEE standard 802.11p, specifically designed for transportation scenarios).[10] The provided services comprise hazard warnings, multimedia streaming, and information services. These are enabled by multiple backend systems. The project envisioned data retrieval from a marketplace, where service providers offer and buy traffic data. It has been implemented as Mobility Data Marketplace[11] (MDM) and is operational today.

In the CoCarX project vehicles disseminated data via messages. There were two kinds of messages: CAM (Cooperative Awareness Messages) which are send out on a regular basis and DENM (Dedicated Event Notification Messages), which are used for hazard warnings. In the project a custom JSON format, the CoCar or CoCarX message, has been used intensively for test purposes. But also other data sources, such as Floating Phone Data, have been investigated according to their usefulness for certain traffic applications.

Two applications have been picked to showcase the near real-time data processing, management, and creation in the the CoCar architecture.

1. **Queue-end detection (QED):** Queue-end detection has been picked, as rear-end collisions are one of the most frequent accidents with fatalities as the drivers perceive the queue-end often too late. Queue-end detection is therefore an important traffic application which could reduce the severe consequences caused by such accidents significantly. Furthermore, queue-end detection requires a low latency and very frequent updates which have to be tackled by a system with real-time capabilities.

2. **Traffic State Estimation (TSE):** Traffic state estimation has been selected, as it is a very common application, very well known, and comprehensible for end users. Doing TSE in real-time poses some extra challenges to the system as a high volume of data has to be processed in a short time period. Interesting research questions open up, such as which data sources are relevant, how much data is sufficient for a reliable estimation, and how frequent does the state change.

In the following we will start with the classification of the two applications according to the three ways of approaching data stream applications explained in Section 5.2. Subsequently, we will elicitate the requirements on the three levels of tasks, data sources, and data quality. However, we will not present an exhaustive requirements analysis, but will only summarize the most important points.

6.2 Queue-end Detection

As explained in Chapter 2 queue-end detection is the process of determining positions or areas in a road network where the end of a queue poses a hazard to approaching vehicles. In this work we consider hard queue-ends located upstream of the approaching vehicles and we are especially interested in hidden queue-ends (cf. Section 2.4). A queue-end is thereby the rear-end of a queue where vehicle speed drops rapidly. We realize the implementation of the queue-end detection using a partly data-driven approach with a fixed goal (cf. Section 5.2). We know exactly what the outcome of the application

[10] `https://standards.ieee.org/findstds/standard/802.11p-2010.html`
[11] `http://www.mdm-portal.de`

should be, but we are only partly free to select appropriate means and data sources due to constraints from the project context.

In the following we describe the requirements for the application of locating such a queue-end. The requirements have been raised in the course of the project with the project partners and additional literature research.

6.2.1 Task Requirements

We explained in Section 5.3 that the task requirements analysis in our process model comprises three steps, namely the goal description, the output definition, and descriptions of Quality of Service and provenance requirements. In the following we will detail those points for the QED application.

- **Goal Description:** For the queue-end detection our goal is to get the precise position (in a range of 100 to 400 m) for each queue-end present in a road network. This position is send in a message to vehicles in near vicinity of the queue-end over the CoCarX Geocast Service to warn vehicles approaching this position. The Geocast Service is able to disseminate messages into a certain spatial area (Fiege et al., 2011). The message with the queue-end warning should only be sent if the information's reliability is sufficiently high. The scenario is depicted in Figure 6.2. Further goals are to research if the data from V2X messages is sufficient to ensure reliable results for the detection or if further data sources are required. Furthermore, the question arises which type of information from the messages is crucial for the application. Also the overall required data volume for the application and a sufficient ratio of equipped vehicles and non-equipped vehicles to represent the traffic situation correctly is of interest. Finally, we require the application to learn and detect by itself when a queue-end is present or not based on former and current data.

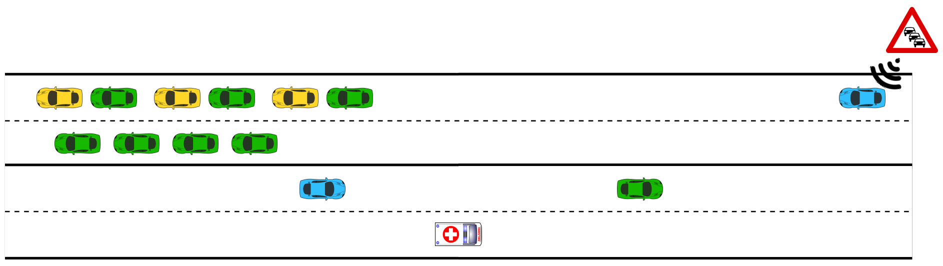


Figure 6.2: Queue-end Detection Application

- **Output Definition:** The output of the application should be a data element, event, or message, which contains the data from Table 6.1.

- **Quality of Service:** The output should be sent as soon as information is available (push basis). The system should revise the current situation at least every 30 seconds for each part of the road network. The latency in the system from receiving data from vehicles to creating a warning should be in the range of a few milliseconds to ensure quick information of approaching vehicles. If information is outdated, the information should be discarded. There are no requirements

Table 6.1: Required Output for QED Scenario

Name	**Description**	**Data Type**
Latitude	The latitude value of the queue-end position.	Float
Longitude	The longitude value of the queue-end position.	Float
Timestamp	A timestamp including milliseconds which indicates when the queue-end has been detected. This should be the creation date of the message or event announcing the queue-end.	Integer
Confidence	A confidence value, which indicates how reliable the detected event is.	Float

regarding buffering. The throughput should be high as bursts of messages from vehicles can be expected.

- **Provenance:** For the research purpose of the system provenance has not been taken into account. But for a real system this definitely has to be considered.

- **Users:** End users of the data are first of all researchers who elaborate the parameters and conditions beneficial for optimal results of the application. The targeted end users are vehicle drivers, which get warning messages from the application making them aware of the queue-end as a hazard.

6.2.2 Data Source Requirements

After we identified the rough goals of the queue-end detection application we now can analyze which are potential data sources. As we mentioned already, there are some constraints from the project according to the data sources. In the course of the CoCarX project one research question is to see if a queue-end detection can be successfully implemented using only the individual V2X messages created by the vehicles in the CoCarX infrastructure (FCD). Hence, the first and provisionally only data source is the stream of CoCar messages.

In the CoCarX project two types of CoCar messages are used: CAM (Cooperative Awareness Messages) which are send out on a regular basis and DENM (Dedicated Event Notification Messages) which are send on an event basis. For testing purposes instead of the standardized CAM and DENM mainly messages in a custom JSON format are used in the project. These messages only contain the required basic information. In the CoCarX project three types of event messages and one type of periodic messages have been used:

- **Emergency Break Light (EBL):** These messages are send by a vehicle every time it brakes very hard.

- **Warning Light Announcement (WLA):** A vehicle sends a WLA every time it turns its warning light flashers on.

- **Emergency Vehicle Warning (EVW):** Special purpose vehicles, such as ambulances or police cars, send these messages regularly when they are in an operation.

- **Vehicle Probe Data (VPD):** Awareness message, which transmits the current position and status of the vehicle.

In the following we exemplify the use of the Data Source Documentation from Appendix B for the CoCar messages.

General Data Source Description

Table 6.2: The Properties of the C2X Message Data Source

Property	Value
Functional Requirements	
Name	Vehicle Probe Data
Version	Unknown
Origin	Vehicles equipped with CoCarX technology. The messages are sent to a central server, which routes them to the QED application.
Format / data model	Each element is sent as a JSON object.
Update Frequency	On event basis. EVW messages are periodically sent in operation.
Schema	Semi-structured schema. But the fields in the elements do not change.
Size	The single elements are very small. The size of the stream depends on the covered area, the traffic volume, and the penetration rate of CoCars.
Costs and Availability	No costs, as this is project internal data. The data is not available as only one or two CoCar prototypes are existent at implementation time of the application. A very simple custom simulation was available at the start of the project.
Communication requirements	The data is accessible over TCP with all advantages and disadvantages TCP brings with it. That is, simple communication protocol and high reliability as a connection-oriented protocol. The overhead depends on how messages are sent (in bulk or individually).
Non-functional Requirements	
Time Range	The data is immediately send by the vehicles and routed. No specific time range has to be considered.
Spatial extension	The data is related to a spatial area. Single elements include a GPS position. The spatial extension from which messages come may span a road network of the size of a city or smaller, but depends highly on the use case, the system architecture, and the available resources.
Timestamps	The data is timestamped with a UTC timestamp with second precision.
Purpose /Description	The CoCarX messages are event and periodic messages which either warn of hazards or send regularly the position and status information about the vehicles to the CoCarX server.

Table 6.2: The Properties of the C2X Message Data Source

Property	Value
Models	The messages are created depending on their type. EBL messages are created when a certain negative acceleration is measured in the vehicle. WLA are created when the warning lights are turned on. The same is true for EVW messages for the light bars on the special purpose vehicles, such as ambulances. The periodicity of the VPD messages is configurable. The optimal frequency still has to be found.
Specifics	The messages have to be converted from the JSON format to the targeted format of the anticipated DSMS. For a relational-based DSMS, this means, the messages' nested structure has to be flattened to be usable as message tuples.

Schema Description

The CoCar messages are represented as nested JSON objects. They have the following general structure:

Listing 6.1: Constituents of a CoCar Message

```
<General Part><Application Part>
```

The general part is present in all CoCar messages. The application part is filled depending on the type of the message. It remains empty for VPD messages and has extra information for the other message types. An example VPD JSON object is exemplified in the following.

Listing 6.2: CoCarX Message Structure

```
{
  "actionID": "3ab1c05b2a56261a",  # unique identifier
  "cell_id": 123456,               # mobile network cell
  "ts": 1237898126,                # timestamp in s
  "ttl": 8,                        # time to live in s
  "wgs84": {                       # WGS84 coordinates
     "lat":50.790596,              # latitude
     "lng": 6.064266,              # longitude
     "elev": 123,45                # elevation in m
  },
  "heading": 175.64,               # direction in Grad
  "speed": 4.55,                   # speed in m/s
  "accel": 2,5,                    # acceleration in m/s^2
  "VPD":{}
}
```

A detailed description of the data can be found in Appendix B.3.

It is obvious that for a queue-end detection application we need data from a medium to high amount of vehicles over a longer period of time. For all available traffic states the application needs to be prepared and tested.

Based on the CoCarX messages and given the goal of detecting a queue-end based on these messages, we can discuss if the given attributes of the CoCar messages could be helpful for queue-end detection. As we are searching for a hard queue-end, speed and acceleration are expected to be of high importance for the detection. Of course also the position, heading, and the timestamp are required to locate vehicles and estimate the currency of the event. Furthermore, the information if a vehicle turned on its warning light flashers might be a strong indicator for a queue-end, as this usually only done by drivers on a highway when they approach end of a queue or in an emergency (break down, driver is not feeling well, other hazard on the road). If vehicles brake hard this could also be an indicator of a hazard on the road. But a single vehicle braking or turning on its warning light flashers is not enough to detect a queue-end. A certain amount of vehicles testifying this situation is required.

Though the studies should first only comprise CoCarX messages, we further consider using Floating Phone Data as it poses an interesting data source for traffic data (cf. Section 2.2). Such a data source should include a position of the handset (longitude and latitude) as exact as possible and frequency and coverage should also be as high as possible.

Based on these first considerations about the existing and required data sources, requirements for the corresponding required data quality can be described.

6.2.3 Data Quality Requirements

In this section we will discuss which DQ dimensions are important for the domain of C-ITS. Adhering to the methodology, we will then define the overall DQ goals for the application and which dimensions are particularly important for QED.

There exist already a set of approved and common dimensions which are generally applicable. However, each application may introduce new aspects of DQ which are specific for this application or domain. As Redman (2004) points out, there exist "hundreds of dimensions of data quality, a relatively few dimensions are most important in practice." Hence, it is crucial to determine the ones relevant for the target application.

There exists a plethora of classifications which structure and describe DQ dimensions, such as the Total DQ Management (TDQM) classification (Wang and Strong, 1996; Strong et al., 1997), the Redman classification (Redman, 1996), or the Data Warehouse Quality (DWQ) classification (Jarke et al., 1999). The classifications categorize the dimensions according to different aspects. Fan and Geerts (2012) identify in particular timeliness, completeness, and consistency as central issues in data quality. Specifically for ITS, several sources of deficiencies are inherent to the way data is acquired. Static as well as mobile sensors are subject to errors and failure. Roadway-based sensors (such as inductive loops) can introduce errors from 2% to 10% in volume and speed measurements caused by, for example, environmental conditions, incorrect placement, calibration problems, bad maintenance, and operational failures. These causes also apply to non-intrusive sensors, such as radar sensors (Margiotta, 2002). Hence, the monitoring of data quality right from the acquisition of the data is crucial to make proper estimations. Also problems in road network coverage, sampling rate, and aggregation interval are quality issues (Margiotta, 2002). Dalgleish and Hoose (2008) define three categories of errors in traffic data. *Systematic errors* (constant deviations from a reference value), *random errors* (error values with a certain distribution around a mean), and *blunders* (unforeseeable errors, e.g., human fault). There have been several works, which also try to analyze quality issues in traffic applications. In Table 6.3 the proposed dimensions are summarized.

Now that we have reviewed general DQ dimensions for ITS applications we will analyze the DQ requirements for the application at hand.

Table 6.3: DQ Dimensions in ITS Literature

	(Turner, 2002)	ITS America (Turner, 2002)	(Tarnoff, 2002)	(Schützle, 2009)	(Baumgartner et al., 2010)
Accuracy	✓	✓	✓	✓	✓
Completeness	✓			✓	✓
Timeliness	✓	✓	✓	✓	✓
Coverage Breadth	✓		✓		✓
Coverage Depth		✓			
Accessibility	✓				
Confidence		✓			✓
Availability		✓		✓	✓
Consistency		✓		✓	✓

- **Goal and Output:** As we have described for the application goal in the task requirements, the application should deliver a confidence value for each detected queue-end, such that a message will or will not be sent to the end users. This should decrease the probability that users do not trust the application and get annoyed quickly. Finally, false alarms are better than a missing alarm in case of a real queue-end. As the detection of a queue-end is expected to be a non-trivial task which will be based on several data attributes of the CoCarX messages, the confidence value should be a compound value. The confidence value should represent the quality of the process to detect the queue-end and at the same time reflect the quality of the used data. It should be possible to weight the single components contributing to the confidence value's calculation as some aspects might be more important than others for the final result.

- **Quality Dimensions:** Besides confidence, other dimensions are important for the QED application. We plan to employ an algorithm which determines the position of a queue-end based on the given data. Hence, we also need quality metrics which rate the accuracy of the algorithm. In this section we introduced several metrics, which can be used as DQ dimensions in the DQ management. Furthermore, we need the basic dimensions data volume and timeliness. Data volume is required to research if the number of items used to make the result is related to the overall result quality. And timeliness is needed to see if latency is kept in an acceptable range. Completeness can be used to see if data is missing, e.g., if a sensor does not deliver data which results in null values. Other dimensions which are related to the semantics of the data have to be determined later, when all data attributes for the application are known.

6.2.4 Selection and Modeling of Data Sources

After raising the requirements and describing the available data sources, further considerations have to be made. The CoCarX messages being the most important data source in the short run for this application is problematic. There exist only very few prototypes which are far not enough to deliver data to test the intended application. Hence, the data has to be artificially created to ensure sufficient data volume. Two ways can be considered, namely the adaption of recorded real-world data, and the use of a traffic simulation software extended by a mechanism to create CoCarX messages. We discussed the pros and cons of using real-world data and using a simulation already in Chapter 5. As we also consider to use Floating Phone Data at the same time, it is also very hard to find data sources which provide both: traffic data with V2X messages and data of cell phones. In the case of the QED application we decided to use a traffic simulation as we wanted to

- have full control over the traffic situation and the parameters influencing it,
- repeat the situation arbitrary times in the exact same way, and
- use predefined and common microscopic models.

There are powerful macroscopic and microscopic simulations (cf. Chapter 2 for further information about traffic models) which are used to research traffic phenomena. These tools have proved to replicate traffic situations in a realistic manner. We pose the following requirements to a tool to simulate CoCarX messages and FPD:

- microscopic model for simulation as each individual vehicle should be able to disseminate messages depending on the situation,
- creation of artificial maps as well as import of real maps must be possible,
- extendability to integrate own code and functionality,
- configurable traffic volume,
- configurable composition of different vehicle types,
- configurable number of persons in a vehicle for potential simulation of multiple cell phones (Floating Phone Data),
- simulation of cellular networks,
- simulation of multiple kinds of motor vehicles and trains, as well as pedestrians and bicycles,
- accessible output of data for each simulated road user,
- at least position, timestamp, speed, and acceleration should be provided for each road user

We evaluated the traffic simulations VISSIM by PTV AG[12], the open source platform SUMO[13], and the simulation framework VanetMobiSIM[14]. VISSIM[15] is a microscopic simulation software and a flagship of the PTV AG. It provides microscopic, mesoscopic, and hybrid traffic models. For the microscopic models the Wiedemann psycho-physical car-follow models have been implemented (cf. Section 2.3). It has many plug-ins to extend the original product and may be extended by using a COM interface and Python code. For research purposes a free full version license can be ordered. The newest version is 8.0, the version at the beginning of the CoCar project

[12]http://vision-traffic.ptvgroup.com/de/produkte/ptv-vissim

[13]`http://sumo.dlr.de/wiki/Main_Page`

[14]`http://vanet.eurecom.fr`

[15]`http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/`

was 5.10. Simulation of Urban Mobility (SUMO)[16] is an open-source traffic simulation software developed by the Aerospace Center Germany (Deutsche Luft- und Raumfahrt, DLR). At the project start it had version 0.10.3 and moved forward today to 0.27.1. It "is a microscopic, inter- and multi-modal, space-continuous and time-discrete traffic flow simulation platform." (Krajzewicz et al., 2012) SUMO supports the import of many formats for roadway networks and enables the simulation of different road user types. Networks of whole cities can be simulated and other features, such as on-line control, i.e., changing parameters during simulation, and emission simulation are integrated. VANETMobiSIM is an open-source simulator framework which extends the CanuMobiSIM user mobility simulation with a traffic simulation (Härri et al., 2006). For mobile user simulation different tools, such as n2, are supported. It provides macroscopic and microscopic traffic simulation. The microscopic simulation utilizes multiple models, such as Fluid Traffic Model or models which can also take into account other vehicles' movements (Intelligent Driver Model with Intersection Management and the Intelligent Driver Model with Lane Changes) (Härri et al., 2006). There exist also multiple models specifically dedicated to pedestrians. The current software version is 1.1, which has not changed since the beginning of the project. Hence, we can assume, that its development is discontinued. In Table 6.4 the comparison of all three simulation applications is presented.

Table 6.4: Comparison of Traffic Simulation Software

Feature	**SUMO**	**VanetMobiSIM**	**VISSIM**
Microscopic Model	✓	✓ also macroscopic	✓ also mesoscopic and mixed (meso + micro), but only since VISSIM 8.0
New Maps	✓	✓	✓
Map Import	✓ many formats supported.	From TIGER database.	✓ only from VISUM (ANM format), with custom code also from OpenStreetMap[17]
Network Size	✓	?	✓
Extendability by Code	✓ plug-ins/tools written in Python	C++ and OTcl	COM interface, C2X module programmable using Python
FCD or FPD Simulation	Only Bluetooth or WLAN output, not usable as event in the simulation.	n2[18] (VANET simulation) or other mobile network simulators	Only simple C2X simulation, but programmable and usable in the simulation.
Configurable Traffic Volume	✓	✓	✓

[16]`http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/`
[17]`https://www.openstreetmap.de`
[18]`http://nsnam.sourceforge.net/wiki/index.php/User_Information`

Table 6.4: Comparison of Traffic Simulation Software

Feature	**SUMO**	**VanetMobiSIM**	**VISSIM**
Unlimited Number of Road Users	✓	?	✓
Configurable Composition of Vehicle Types	?	?	✓
Simulation of Multiple Vehicle Types, Peasants, Bicycles, public transport	No dedicated models for bicycles, peasants, and trains, only workarounds	Only cars, trucks, pedestrians, buses	✓
Time Steps	In seconds	?	In seconds
Persons in Vehicles	—	—	✓
Accessible Realtime Output	✓ (in newer versions, before: file output)	only file output	✓ over Python code possible
Information in Output	Limited, e.g., acceleration missing	Limited	✓

A detailed description and further comparison of all three tools according to the aspect of simulating FPD can be found in (Chen, 2009). We decided to use the VISSIM simulation software for the CoCar project and for the QED application, because it fulfills all posed requirements and offers a detailed simulation of many vehicle types with corresponding models which is not the case for all of the other simulations. It is parametrizable in many ways and enables extension by code to integrate own functionality. Furthermore, a lot of information can be output by the simulation and it provides the identification of queues and their length.

After deciding for VISSIM as a simulation software, we need to model how it can be utilized as a data source for our needs. Several components need to be considered:

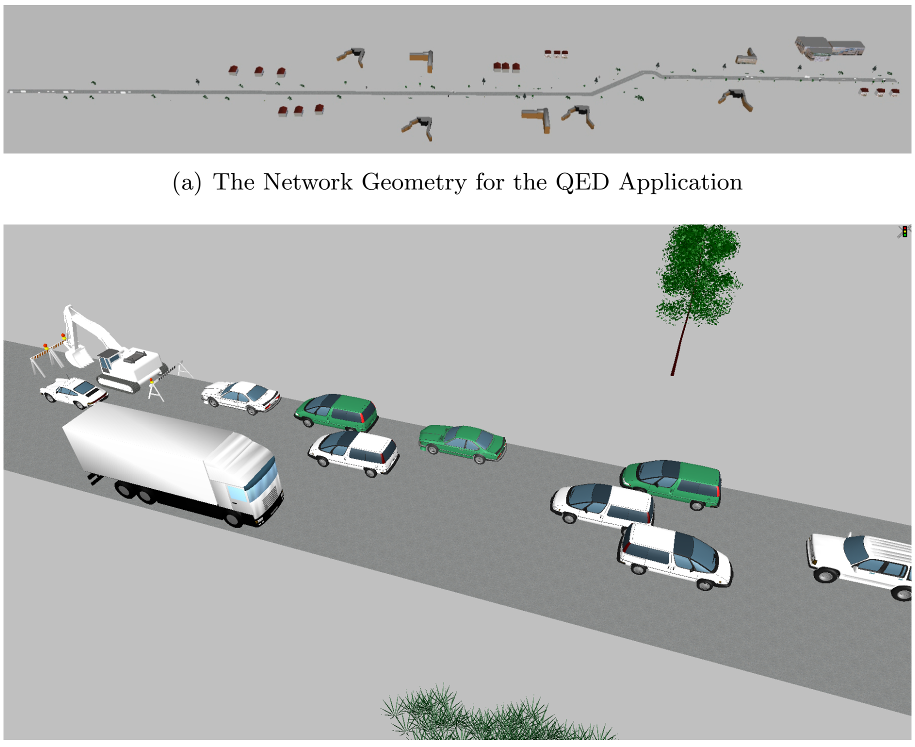
- design of a road network for simulation,
- abstractions for queue-ends,
- design and implementation of the V2X messages,
- output interface for V2X messages, and
- design of parameterization of simulation runs for the network

We will describe all of these tasks in the next sections.

Road Network Design

In VISSIM road networks consist of links and connectors. A link is a curve composed of a sequence of geometry points with a start and an end point. A link is unidirectional and consists of a number of fixed lanes. A link has at least one lane. Vehicles can change lanes within a link. Two links are connected via connectors (also curves) with each other. With connectors also junctions are modeled. For each link a driving behaviour can be defined (e.g., speed limits, no overtaking). Each link and each connector have a unique identifier.

As we aim at developing an application for detecting a queue-end, we pick the typical traffic situation and network geometry of a hidden queue-end. Hence, we start with an artificial network consisting of a highway with two links, each comprising two lanes. Both links have a length of 5 km. The speed limit is unrestricted. The links are almost straight, with one sharp turn as depicted in Figure 6.3(a). One link contains a hazard (a construction site with an excavator) narrowing the street to one lane, shown in Figure 6.3(b). A reduced speed area encloses the construction site, restricting the maximum speed to 70 km/h.


(a) The Network Geometry for the QED Application

(b) Close-up of Road Works in the QED Scenario

Figure 6.3: The Network and Setup of the QED Scenario in VISSIM

Abstraction for Queue-ends

The next challenge is to consider how a queue-end could be represented in such a scenario and how we can identify it. We are interested in the position of the queue-end as exact as possible to send warnings to vehicles in near vicinity by GeoCast. For this we need the geographical coordinates, i.e., longitude and latitude, or a reference location, where the queue-end is located. For a queue-end an exact geographic location is not really necessary. Particularly on highways vehicles typically approximate a hard queue-end fast and hence, warnings should be sent early to vehicles about 1-2 km before.

As a result, it is sufficient to locate a queue-end only on a section of a link and we aim at the division of links into sections with unique identifiers per link. As we need the links and sections as spatial objects, e.g., for Map Matching, we use a spatial database to store the road network as curves and divide its links into sections, i.e., curves of a fixed length. The last section of the link contains the remainder of the overall length of the link. We first used the spatial data features of SQL Server 2008 to implement such a spatial database. Due to performance problems we migrated the database to PostgreSQL using PostGIS as spatial feature extension during the project. PostGIS adds spatial data types and functions to the database and is compliant with the OGC specifications for GIS. We import the spatial representation of the links in the VISSIM network into the spatial database as line strings (sequence of points in a two-dimensional space represented as a string).

The division into sections is done using stored procedures. The schema of the relational database comprises tables connected via foreign keys for the networks, sections, and links (including their spatial representation). For the example network from Figure 6.3 an example division into sections is depicted in Figure 6.4.

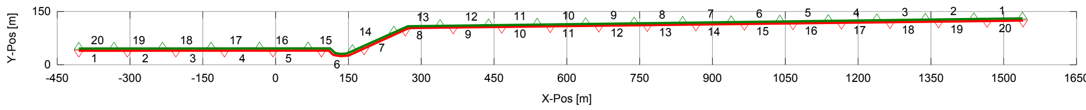


Figure 6.4: Division of the QED Network into Sections

Our idea is to collect data for all vehicle positions which are located on a specific section and, based on this data, determine whether a queue-end is present on that section or not. Hence, it can be interpreted as a classification problem with two classes. To provide automatic detection of a queue-end a possible approach is to use supervised classification algorithms (cf. Section 4.4.1). A requirement for these algorithms is the provision of examples to enable learning. To create examples for queue-end detection it has to be known, if a queue-end is present on a section or not. We utilize the queue counter feature of VISSIM for the determination of the queue-end. Queue counters can be positioned on links of a VISSIM network and measure the average and the maximum queue length starting from the position of the counter. The criteria when a vehicle is considered as queuing can be configured in VISSIM. In our configuration of the simulation a vehicle starts queuing when it is slower than 15 km/h and has a maximum distance of 20 m to the vehicle in front. It stops queueing if it is faster than 30 km/h. The queue lengths are measured every ten seconds. We extend VISSIM with Python to collect this information in code and create corresponding data elements. The information comprises the event's timestamp, the length of the queue, and the counter id. The positions of the counters are also stored in the spatial database and a stored procedure can be used to determine the position where the queue-end is located. Now that it is known where actually queue-ends are located on a map, the modeling of V2X messages has to be done.

Design of V2X Messages

The next challenge is to extend VISSIM with V2X capabilities. We use and extend the C2X Python library and the COM interface provided by VISSIM to create V2X messages. The different types of messages required and simulated have been detailed in Section 6.2.2. In VISSIM a special C2X vehicle type is defined. Hence, the ratio of C2X equipped vehicles in the overall traffic volume can be defined. Only those vehicles are able to send messages. Based on its state in simulation the C2X vehicle sends a message in the following cases:

- **VPD:** Vehicle Probe Data messages are created every 10 seconds. They include the current status information of the vehicle sending the message, such as position, speed, or acceleration.
- **EWL:** The conditions for sending an EWL message is checked in every timestep. If the acceleration of the vehicle is below -5 m/s^2 and speed is above 0 m/s, a message is issued.
- **EVW:** The conditions for sending a WLA message are checked in every timestep. If the speed is below 10 m/s and acceleration below 0 m/s^2, a WLA message is issued. The message is only issued every 10 seconds to avoid sending too many messages.
- **WLA:** Vehicles which are marked as emergency vehicles or road work vehicles issue a message every 10 seconds.

We implemented the corresponding Python classes for message creation and sending. Messages are encoded as JSON objects (as it is the case in the real-world CoCarX scenario) and are appended to a TCP server, because this was the supposed setup in the real-world CoCarX project. A TCP client is also provided which is able to receive the estimated queue-end from the QED application.

Ground Truth Messages

We have mentioned, that we like to employ classification algorithms to solve the problem of QED. We abstract QED to a two-class problem as we try to determine for a road section on a link based on aggregated traffic data, if the section contains a queue-end or not. For supervised classification algorithms we require to know the ground truth, i.e., if the section really contains a queue-end or not in the given situation. With the help of the queue-end counters we can determine the queue-end and can send corresponding messages with the queue-end position likewise via TCP.

Simulation Parameterization

After creating a network and enabling C2X message creation and sending, the parameterization of the simulation has to be done. We will briefly explain which parameters are important for the QED application and will be used and varied throughout this work. In Table 6.5 the parameter list and their descriptions are given.

After we have modeled the C2X message data source by using the VISSIM traffic simulation and providing a TCP server to send the messages, we can proceed with modeling and implementing the QED application using a DSMS.

6.2.5 Task Modeling & Implementation

As detailed in Chapter 5 the next step in the creation of data stream application is the modeling and implementation of the data processing part. The first task is to select a DSMS which fits the data management requirements.

Table 6.5: Simulation Parameters

Parameter Name	**Description**	**Example Value**
Car-follow Model	The algorithm which is used to model the behaviour of vehicles.	Wiedemann79
Traffic Volume	The overall number of vehicles per hour independent of their type.	2500 veh/h
Truck Ratio	The ratio of trucks in the overall number of vehicles.	12%
C2X Penetration Rate	Ratio of C2X vehicles (cars and trucks) in the overall number of vehicles.	10%
Section Length	The length of the equally sized sections in a network link.	100 m

Selection of a DSMS

In the case of the QED application we deal with JSON object streams which are send via a TCP server. In prospect of further applications and data sources in the CoCarX project we decided to use a relational DSMS. Other data formats, such as JSON, can be transformed to the flat relational model with low effort. As no other special data formats, such as RDF data or XML data are expected, the relational model is sufficient. Further requirements are that we want to be able to extend the DSMS with own code to implement special features such as DQ management and integrate custom algorithms. We decided to use the Global Sensor Network (GSN)[19] system (Aberer et al., 2006), as it provides all these features and additionally, it is very easy to implement new stream applications. In the following we describe the most important principles of GSN.

The Global Sensor Network System

GSN provides a flexible, adaptable, distributable, and easy to use infrastructure. It wraps functionality required for data stream processing and querying around existing relational database management systems, such as MySQL or Microsoft SQL Server. We decided to use the MySQL in-memory database to ensure efficient processing. GSN mainly consists of the following components:

- **Stream Element:** Data processed by the system has to be in the form of a stream element. A stream element can be viewed as a data record in a table (as in relational databases) which has a certain schema, defined by the wrapper or virtual sensor producing or forwarding the element, as its output structure.
- **Wrapper:** A *wrapper* is a component which receives data from data sources. It consists mainly of a Java class which determines its behavior. An example for a wrapper is a JSON wrapper, which enables to retrieve JSON messages (the V2X messages) via TCP. Each wrapper has a shortcut to identify it.
- **Virtual Sensors:** A *virtual sensor* is a component processing the data inside the GSN system. Once a data stream element has been retrieved by a wrapper, it is forwarded to the virtual sensors attached to the wrapper. Each virtual sensor

[19] `http://sourceforge.net/projects/gsn`

also can have virtual sensors attached to it retrieving data from it. In this way, wrappers and virtual sensors build a dependency graph. Each virtual sensor consists of an XML configuration file and a Java class controlling the behavior of the sensor. An example for an XML configuration is given in Listing 6.3 (virtual sensor MemoryMonitorVS delivered with GSN).

Listing 6.3: Virtual Sensor Configuration File

```
 1 <virtual sensor name=”MemoryMonitorVS” priority=”11”>
 2   <processing class>
 3   <class name>
 4     gsn.vsensor.BridgeVirtualSensor
 5   </class name>
 6   <output structure>
 7     <field name=”HEAP” type=”double” />
 8         <field name=”NON_HEAP” type=”double” />
 9   </output structure>
10   </processing class>
11   <description>None</description>
12   <life cycle pool size=”10” />
13   </addressing>
14   <storage history size=”1” />
15   <streams>
16     <stream name=”input1”>
17           <source alias=”source1” sampling rate=”1”
18                   storage size=”1m” slide=20s>
19             <address wrapper=”memory usage”>
20               <predicate key=”sampling rate”>1000</predicate>
21             </address>
22             <query>SELECT HEAP,NON_HEAP,
23                                 PENDING_FINALIZATION_COUNT,
24                                 TIMED FROM wrapper
25             </query>
26           </source>
27           <query>SELECT HEAP,NON_HEAP,
28                         PENDING_FINALIZATION_COUNT,
29                     TIMED FROM source1
30           </query>
31         </stream>
32   </streams>
33 </virtual sensor>
```

This virtual sensor retrieves data from a wrapper, which monitors the system memory. After the definition of the sensor name and its priority (line 1), the corresponding class (lines $3-5$) and its output structure, i.e., the schema of stream elements produced by the sensor, (lines $6-9$) are defined. The streams section (lines $15-32$) defines the streams the virtual sensor acquires data from (the example includes only one, `source1`). For each stream the corresponding source has to be defined, which can be either a wrapper or another virtual sensor (lines $19-21$). Depending on the wrapper or sensor several parameters can be configured (line 20) for it. In this example the number of milliseconds between two readings of the system memory size is configured. Afterwards, the query retrieving the data from the wrapper is given (lines $22-25$). The data can be windowed by defining a corresponding storage-size (number of tuples or time period) in line 18. Also the sliding step of the window can be determined (line 18) using the slide parameter. Finally, in line 17 a sampling-rate can be defined which indicates how many elements from the stream should be dropped for load shedding, where 1 means none, and 0.5 means 50%). Lines 27 – 30 denote the overall query which determines the output of the virtual sensor. Here data from one or multiple source streams defined before can be joined, aggregated, or otherwise used in one statement. The `SELECT` clause of this overall query has to include all attributes which are listed in the output structure.

Design of Data Stream Processing for QED

We now have a closer look at how the data is processed inside the DSMS. An overview of the data flow in the system and the implemented components is depicted in Figure 6.5.

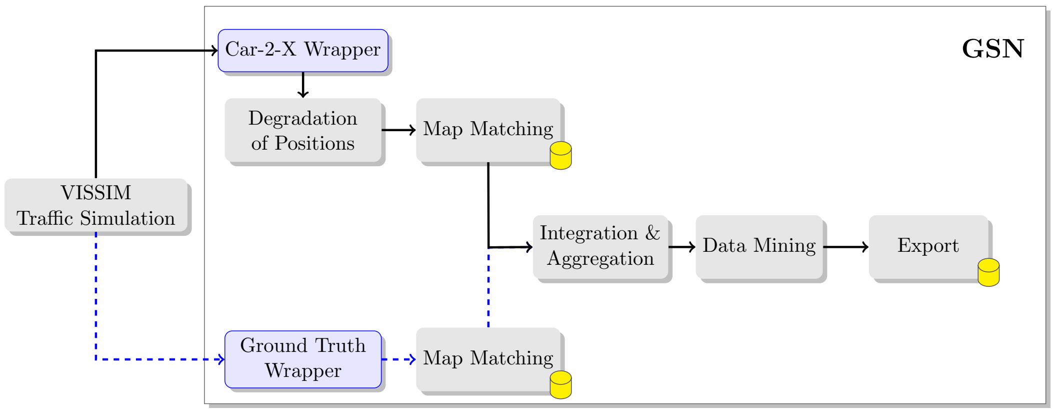


Figure 6.5: The QED Application using GSN Wrappers and Virtual Sensors

As we have detailed in the previous section, GSN provides two structures to work with data streams: wrappers and virtual sensors. The wrappers manage the connections to the data sources, receive the data, and convert it to GSN stream elements, which are in fact relational tuples.

Using Definition 4.2 for data streams, the schema of the stream produced by the wrapper receiving the CoCar messages looks as follows (we only include the attributes which are of importance to the case studies):

$$\texttt{CoCarMsg}(\texttt{Timed}, \texttt{TS}, \texttt{Lat}, \texttt{Lng}, \texttt{Speed}, \texttt{Accel}, \texttt{ApplicationID})$$

where `Timed` is the implicit timestamp given by GSN, `TS` is the explicit timestamp, `Lat` is the latitude of the vehicles position, `Lng` is the longitude of the position, `Speed` is the speed of the vehicle at that time, `Accel` is the acceleration, and `ApplicationID` denotes the type of the message, which can either be an Emergency Brake Light message (EBL), a Warning Light Announcement message (WLA) or a Vehicle Probe Data message (VPD). In the queue-end detection scenario the ground truth messages contain the position of a queue-end and their schema is:

$$\texttt{QueueEnd}(\texttt{Timed}, \texttt{TS}, \texttt{Lat}, \texttt{Lng})$$

where `Timed` and `TS` are defined as before, and `Lat` and `Lng` represent the position of the queue-end. The ground truth messages for the queue-end still have to be matched to a link and section of the network.

One common issue in mobile data detection is, that the measured positions contain some error whose amount depends on the used positioning technique. This error influences the accuracy of traffic information which has been shown, e.g., by Geisler et al.

(2010), and therefore, has to be taken into consideration. In case of the V2X messages we can assume GPS positioning accuracy as in the CoCarX project a GPS device in the vehicle is used for positioning. To approximate reality as close as possible the exact positions in the messages created by VISSIM have to be degraded. To that end, the stream elements created from the CoCar messages are forwarded to a degradation virtual sensor. We use a normal distribution to model the error. When working with real data sources this step is obsolete, of course. The accurate positions are replaced by the degraded positions in the stream elements and forwarded – the schema and all other data remain unaltered.

The next virtual sensor matches the positions of the V2X messages to the road network, which is termed *Map Matching*. Map Matching is the process of finding the closest link and point to the position where the road user actually is located. We use a simple Map Matching technique, where the link and the section with the shortest perpendicular distance to the measured point are selected utilizing the spatial representation and functions of the road network in the spatial database. More sophisticated methods also consider the trajectory of a vehicle and the topology of the network. The V2X messages do not contain any identifying information about the vehicle sending the message because of privacy reasons. Without information about the trajectory of the vehicle, we can only utilize this simple Map Matching technique. We also experimented with our own more complex Map Matching algorithm, which is described in detail in Chapter 9. For the first implementation of the application only the simple technique is used.

The Map Matching also contributes to a realistic scenario as it introduces an error common in traffic applications, too. For the positions in the ground truth messages of the scenarios, it is mandatory that they are accurate and therefore, these are not degraded. The ground truth messages then have the following schema:

$$\texttt{QueueEnd}(\texttt{Timed}, \texttt{TS}, \texttt{Lat}, \texttt{Lng}, \texttt{LinkID}, \texttt{SectionID}, \texttt{HasQueueEnd})$$

The attribute `HasQueueEnd` will be always 1 as the stream will only contain tuples for the sections with a queue end. However, to determine the links and sections the ground truth positions are located on, in the queue-end scenario the corresponding stream elements are forwarded from the wrapper to a second map matching sensor. This sensor works in the same way as the Map Matching sensor. It introduces no error, because the positions lie exactly on the sections and are matched precisely.

In the Map Matching sensors, we add the new information of link and section number to the stream element. Hence, the schemas of the forwarded stream elements have to change:

$$\texttt{CoCarMsg}(\texttt{Timed}, \texttt{TS}, \texttt{Lat}, \texttt{Lng}, \texttt{Speed}, \texttt{Accel}, \texttt{ApplicationID}, \texttt{LinkID}, \texttt{SectionID})$$

After the position of each message has been matched to a section, we need to prepare the data for data stream mining. The overall goal of the mining process is to determine a class value for a section, characterized by a set of data. In our approach, we aggregate CoCar messages with positions lying on a particular section for a certain time window. This means, we calculate the following parameters from the CoCar data stream elements for one section number and a time-based window over the last x time units:

- average speed (`AvgSpeed`)
- average acceleration (`AvgAccel`)
- number of Emergency Brake Light messages (`EBLNo`)

- number of Warning Light Announcement messages (`WLANo`)

In the QED scenario no VPD is used. We solely rely on the event messages of types EBL and WLA. To complete the training data set for the data stream mining, we have to join the aggregated data with the ground truth stream elements, which define the true class value of each data set. Both streams are joined over the section and the time window. The join is obsolete if no training is performed.

We now formulate the queries used to aggregate and join the data from the CoCar messages stream and the ground truth streams using the semantics from Section 4.2 and the extended relational algebra semantics (Garcia-Molina et al., 2009). First, we define the queries retrieving the data from the two windowed streams:

$$\begin{aligned}
\texttt{vQueueEnd} &= \pi_{\texttt{LinkID,SectionID,HasQueueEnd}}(\\
&\qquad \gamma_{\texttt{LinkID,SectionID,HasQueueEnd}}(W_{120,10}(\texttt{QueueEnd}))) \\
\texttt{vCoCar} &= \pi_{\texttt{Speed,Accel,ApplicationID,SectionID,LinkID,TS}}(W_{120,10}(\texttt{CoCarMsg}))
\end{aligned}$$

The γ-operator in this extended relational algebra expression performs the grouping over the input relation. In this example, the sliding window over the `QueueEnd` stream is grouped by `LinkID`, `SectionID`, and `HasQueueEnd`. In GSN, the results of these queries are represented by views and therefore, can be used as relations in the integrating query. The integrating query for the queue-end example looks as follows in relational algebra (note, that we do not need to specify a window here anymore):

$$\begin{aligned}
&\pi_{\texttt{AvgSpeed,AvgAccel,HasQueueEnd,WLANo,EBLNo,LinkID,SectionID}}(\texttt{vQueueEnd} \\
&\quad ⟕ (\gamma_{\texttt{SectionID,LinkID},AVG(\texttt{Speed})\rightarrow\texttt{AvgSpeed},AVG(\texttt{Accel})\rightarrow\texttt{AvgAccel}}(\texttt{vCoCar}) \\
&\quad ⟖ (\sigma_{\texttt{ApplicationID}='WLA'}(\gamma_{\texttt{SectionID,LinkID},Count(*)\rightarrow\texttt{WLANo}}(\texttt{vCoCar})) \\
&\quad ⟖ (\sigma_{\texttt{ApplicationID}='EBL'}(\gamma_{\texttt{SectionID,LinkID},Count(*)\rightarrow\texttt{EBLNo}}(\texttt{vCoCar}))))))
\end{aligned}$$

This query results in the creation of stream elements which contain the desired aggregated values per section and the ground truth class for training. The stream elements are now ready to be fed into the data stream mining algorithm. The resulting schema looks as follows:

$$\begin{aligned}
&\texttt{MiningElement}(\texttt{Timed}, \texttt{AvgSpeed}, \texttt{AvgAccel}, \texttt{HasQueueEnd}, \\
&\quad \texttt{WLANo}, \texttt{EBLNo}, \texttt{LinkID}, \texttt{SectionID})
\end{aligned}$$

We already mentioned, that we only use classification in our case study. As depicted in Figure 6.5 with dashed and solid arrows, we can setup the scenario according to the two classical supervised learning modes: training and classification. If a classification without training is required, the obsolete virtual sensors can easily be removed. We integrated the data stream mining framework Massive Online Analysis (MOA) into GSN by encapsulating it into a virtual sensor. MOA is based on the well-known mining framework Weka[20]. In the mining virtual sensor, the incoming data stream elements are converted into MOA compatible instances.

After the mining algorithm classified an instance, the statistics of the classifier are updated and the classifier is trained on the instance. The statistics of the classifier comprise values of a *confusion matrix*. For N classes a confusion matrix contains NxN values for each combination of predicted class and true class (cf. Section 4.4.1). Then,

[20]http://www.cs.waikato.ac.nz/~ml

a corresponding message is generated and can be exported, e.g., archived in a database, send to the traffic simulation, or in the real-world scenario to vehicles in near vicinity. A detailed description of the data mining process and the different algorithms we experiment with are described in Chapter 10. In the first implementation of the application we used the Hoeffding Tree (cf. Section 4.4) for classification. The stream elements produced by the mining sensor have the following schema for queue-end detection:

$$\texttt{ClassifiedElement}(\texttt{Timed}, \texttt{AvgSpeed}, \texttt{AvgAccel}, \texttt{HasQueueEnd}, \texttt{WLANo}, \texttt{EBLNo}, \texttt{LinkID}, \texttt{SectionID}, \texttt{ClassQueueEnd})$$

where `ClassQueueEnd` is the predicted class.

When a queue-end is determined (`HasQueueEnd` = 1) a hazard warning message is sent by the export sensor in JSON format via TCP to the traffic simulation to visualize the proximity between the positions of the real queue-end and the estimated queue-end. The visualization of the estimated and real queue-ends in the traffic simulation based on the created messages is depicted in Figure 6.6.

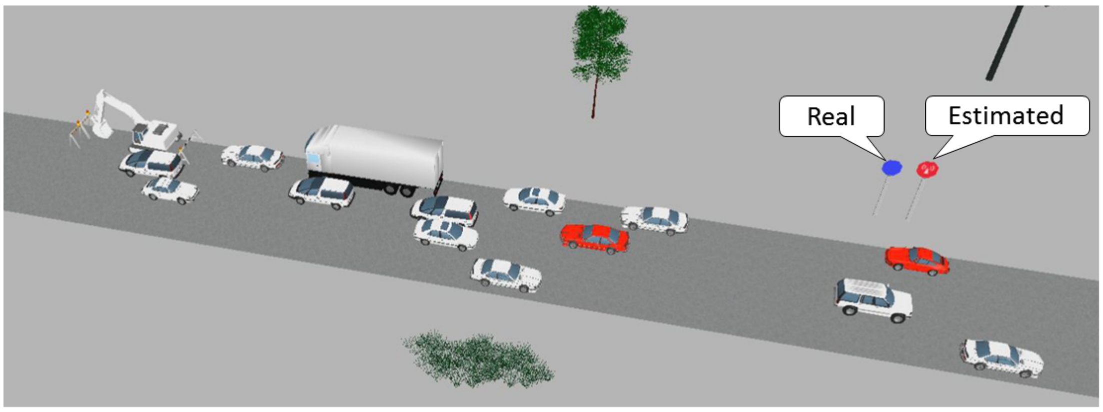


Figure 6.6: Visualization of Simulated and Estimated Queue-ends

After we have setup the data processing in the DSMS the DQ design and implementation of the application will be executed in the next step.

6.2.6 Data Quality Design and Implementation

The design and implementation of DQ management is a very important step in the process model. Some methodologies have been mentioned in the corresponding description of the process model. In this work we present our own methodology which is specifically suited for DSMS and data stream applications. The methodology provides a framework consisting of a metadata model to describe DQ management concepts in general and allows to be extended to specific applications by defining dimensions, metrics, and actions. Furthermore, a framework is provided which extends DSMSs with DQ management capabilities, i.e., the measurement of the specified dimensions for the data stream elements to which they have been assigned. We will defer the description of the complete framework to Chapter 8 and exemplify its use with the two ITS applications described in the chapter at hand.

At this point we will just show one example for the use of the matrices proposed in the process model for one wrapper of the QED application in Table 6.6.

Table 6.6: Metrics Matrix for the Car-2-X-Wrapper

Attribute	**Accuracy**	**Consistency**	**Timeliness**
Speed	0.9 (constant)	$0 <= SPEED \wedge SPEED <= 180$	
Acceleration	0.95 (constant)	$accel < 5 \wedge accel > -7\,5$	
TS			currentTime - ts
LAT		$-3250 < LAT \wedge LAT < 1300$	
LNG		$LNG < 130 \wedge 25 < LNG$	

In the following section, we will proceed with a first evaluation of the so far implemented QED stream application to answer first questions and possibly conduct some reconfiguration of the system.

6.2.7 Evaluation Setup and Results

In this evaluation, we focus on the effect of various application-related parameters on the accuracy of the detection method. We analyze three parameters, which seem to be most promising in the queue-end scenario: the penetration rate of CoCars, the traffic volume, and the length of the sections. In addition, we study one system-related parameter: the window size of the queries in the DSMS, i.e., the amount of data from the past which is taken into account by the data stream mining methods. For the queue-end detection time-based windows have been used to select data from the last defined period of time, e.g., the last minute.

For the evaluation, a default configuration of these parameters has been determined. We set the penetration rate to 5%, the traffic volume to 2500 veh/h, and the section length to 100 m. The window size was chosen to be 120 s. To investigate the influence of one parameter, this parameter is varied and all other parameters keep their default value. The following evaluation metrics of the data mining results have been calculated for each run, whereby we define two classes to be identified by the classifier: sections with a queue-end, denoted as *positives*, and sections without a queue-end, denoted as *negatives*. *True positives* are the correctly identified queue-ends, whereas *false positives* are instances erroneously classified as sections containing a queue-end. Correspondingly, *true negatives* and *false negatives* are defined (cf. Section 4.4.1).

- **Overall accuracy:** Determines the ratio of instances with correctly classified classes to the number of all instances classified so far.

$$\frac{\text{True Positives } + \text{ True Negatives}}{\text{Positives } + \text{ Negatives}} \tag{6.1}$$

- **Sensitivity:** Sensitivity is the ratio of correctly identified queue-ends to the number of all real queue-ends.

$$\frac{\text{True Positives}}{\text{True Positives } + \text{ False Negatives}} = \frac{\text{True Positives}}{\text{Positives}} \tag{6.2}$$

In this scenario, we are highly interested in the sensitivity, because it is more dangerous to leave a real queue-end unrevealed as to forecast a non-existent queue-end.

- **Specificity:** Specificity is defined as the ratio of the number of correctly classified sections without queue-end to the number of all Negatives.

$$\frac{\text{True Negatives}}{\text{True Negatives } + \text{ False Positives}} = \frac{\text{True Negatives}}{\text{Negatives}} \tag{6.3}$$

Each evaluation starts with a new tree, i.e., the classifier has not seen any training instances so far. To ensure comparability, all components in the framework have to work in a reproducible manner, i.e., all randomized elements have to be set to a fixed seed. For example, each traffic simulation run configured with the same parameters is identical, i.e., the raw data produced by the vehicles is always the same as well as the traffic state in each time step. To test the comparability, two identical evaluation runs with default values have been made before starting the actual tests. The runs had almost identical accuracy gradients and differed no more than 1% in value. Differences in values can be explained by minor time shifts induced by TCP transfer in GSN and differences in time between start of the simulation and start of the GSN system. That means, the time windows might not be at the exact same position in the time line as they had been in the run before and therefore, might not result in identical averages. For the experiments we turned off the degrading to analyze the algorithm in the ideal situation. Comparisons between runs with and without degrading showed that it has an influence, especially on the sensitivity but the overall trend was the same. Each simulation run had a duration of 45 minutes. The average time for mining one element was about 0.6 ms, while the algorithm mined up to 11014 elements per run. In the following, we describe each parameter that has been evaluated before we show and discuss the most interesting results of the experiments.

Window Size

The window size determines how much data of the past is included in the aggregated data of the instances. A good window size would be very close to the average time for which the queue-end stays in one section. For example, if the queue-end is 50 s in section A and 50 s in section B, then with a window size of 100 s (or more) the system would not be able to make a distinction between section A and B. On the other hand, if the window size is too small, the mining algorithm will not get enough information to determine that a queue-end is in a particular section. Based on some initial observations on the time needed for a queue-end to go from one section to the next (we observed times between 10 s and 110 s for the default traffic volume), we decided to vary the window size parameter between 10 s and 300 s. The results for accuracy and sensitivity are shown in Figure 6.7 and Figure 6.8, respectively.

The results for accuracy are not helpful to identify a good window size as there are much more examples for negatives than positives, especially for the smaller window sizes (4% positives in the 10 s run, 19% in the 300 s run). Therefore, a window size of 10 s has the best accuracy although there are only very few true positives (cf. Figure 6.8).

The noisy start phase of about 400 s should not be taken into account as the traffic jam has not been established yet and the mining algorithms could not be trained on many positives up to that point. The results for sensitivity show, that the bigger the window gets, the better is the sensitivity. This can be explained by the residence

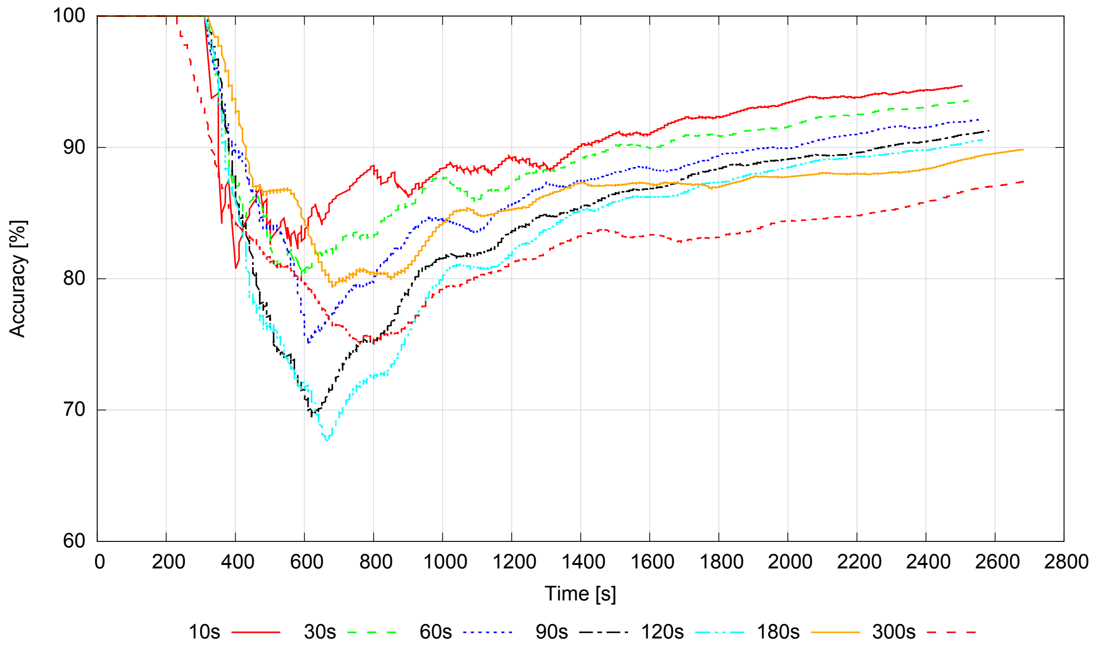


Figure 6.7: Accuracy for Varying Window Sizes

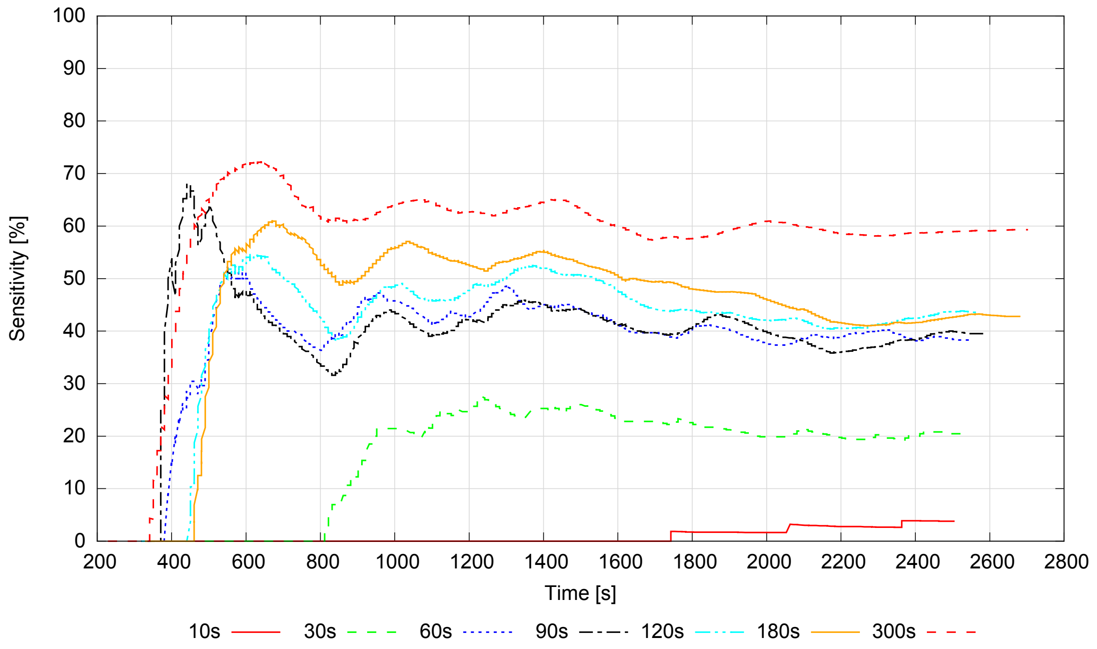


Figure 6.8: Sensitivity for Varying Window Sizes

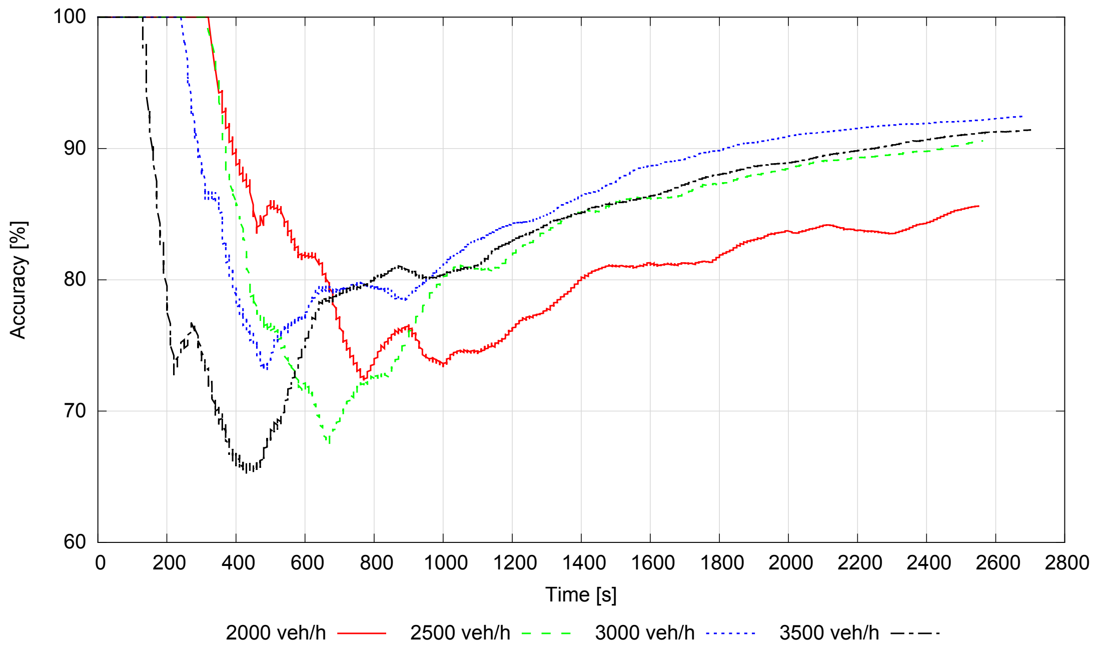


Figure 6.9: Accuracy for Varying Traffic Volumes

time of a true queue-end in the windows. As soon as an actual queue-end is reported for one section, it is included in a window. For bigger windows, an actual queue-end occurs correspondingly longer in the sliding windows (for 300 s it would occur 30 times, for a 10 s window only once). Hence, the number of examples for positives is higher. On the other hand, the specificity is decreasing the bigger the window gets. We decided to keep 120 s as the default window size, at it seems to be a good compromise between sensitivity and specificity and also corresponds to the maximum of the observed duration of a queue-end to stay in one section.

Traffic Volume

The traffic volume is given in vehicles per hour. It is expected that, when the traffic volume increases, more vehicles will be present in one section. This will also lead to a higher volume of CoCars per section and enhance the number of messages per section. Additionally, a queue will grow faster when the amount of traffic is higher, which, in turn, results in a higher rate of vehicles actively braking and switching on their warning flashers. It can be assumed that a higher amount of messages will lead to a more reliable approximation of the actual traffic situation in one section. Therefore, we expect the accuracy to increase when the traffic volume is rising. According to current statistics published by the German Federal Highway Research Institute on German highways traffic volume averages 1500 veh/h calculated from the daily traffic volume (including the low-traffic night)[21]. However, traffic queues occur at higher volumes during the day. Therefore, we experimented with 2000, 2500, 3000, and 3500 veh/h (additionally, 1500 veh/h did not lead to any queueing in our traffic scenario).

The accuracy results depicted in Figure 6.9 fulfill our expectations: the accuracy rises with an increased traffic volume. Skipping the noisy start phase, it can be observed

[21]`http://www.bast.de/DE/Verkehrstechnik/Fachthemen/v2-verkehrszaehlung/Aktuell/zaehl_aktuell_node.html`

that the accuracy is above 80% for all tested traffic volumes, even above 90% for a traffic volume of more than 2500 vehicles per hour. As expected, the highest traffic volumes deliver the best results. The number of seen examples increases in each run (which is also dependent on the duration of the run). While in the 2000 veh/h run the classifier trained on 2634 instances, in the 3500 veh/h run the classifier observed 8033 examples which corroborates the assumption, that the number of messages rises with increasing vehicle volume. However, the analysis of the specificity and sensitivity evaluation showed, that the biggest portion of the correctly classified instances are constituted by true negatives. The specificity reaches accuracies of 98% at maximum in the 3000 veh/h run and shows the same pattern (rising values with rising volumes), while the sensitivity only reaches about 49% in the 3000 veh/h run and no clear trend in the runs can be observed. It can be seen in the training data that the sensitivity gets rapidly worse when the queue reaches the bounds of the network. The trained algorithm gets in this case contradictory information about how a queue-end looks like in the data (queue-end sections are no longer distinguishable from congested sections, because the queue-end counter reports only the last section of the network as the queue-end) and hence, returns no true positives anymore. This seems to be a sign that the used mining attributes allow a distinction between a section with queue-end and a congested section.

To avoid that the queue reaches the network bounds during simulation time, we extended the links of the network to 10 km length. We experimented with 2500, 3000, 3500 and 4000 veh/h. For these experiments the accuracy and specificity showed no clear trend, but as in the 5 km experiments the 3000 veh/h run was slightly better than the others. For the sensitivity the 4000 veh/h run turned out to be slightly better than the other runs, but also deteriorates at the end of the simulation to the same level than the others.

An analysis of the ratio of the actual positive to negative examples reveals, that with increasing traffic volume and simulation time and hence, increasing queue length, obviously the portion of sections from which messages are received and are containing a queue-end in a time window decreases (from 14% in 2000 veh/h run to 9% in 4000 veh/h run), while the fraction of sections which do not contain a queue-end increases. The latter group are most likely sections with congested traffic, but do not contain the queue-end. This leads to a gradual deterioration of sensitivity from the middle to the end of the simulation time.

Penetration rate

One of the most common and interesting questions for traffic applications based on data from mobile sources is: how many vehicles must be equipped with the technology to deliver acceptable results for an application? Huber revealed in his analysis (Huber, 2001b), that a general assumption for FCD penetration rates cannot be made, but that it has to be evaluated in the context of the information system, the traffic application, and the required output quality at hand. The penetration rates he identified in the analyzed studies vary between $1 - 5\%$. We took these values as a basis and made experiments with 1%, 3%, 5%, 7% and 10% penetration rate. It is expected that a higher penetration rate leads to a higher accuracy, as it can be assumed that a higher amount of messages per section is produced. The results for this experiment were inconclusive – no clear trend could be identified in the available simulation time.

In Figure 6.10 the accuracy results are shown. Surprisingly, the experiment showed that all penetration rates converge to a value between 87% and 93%, which would

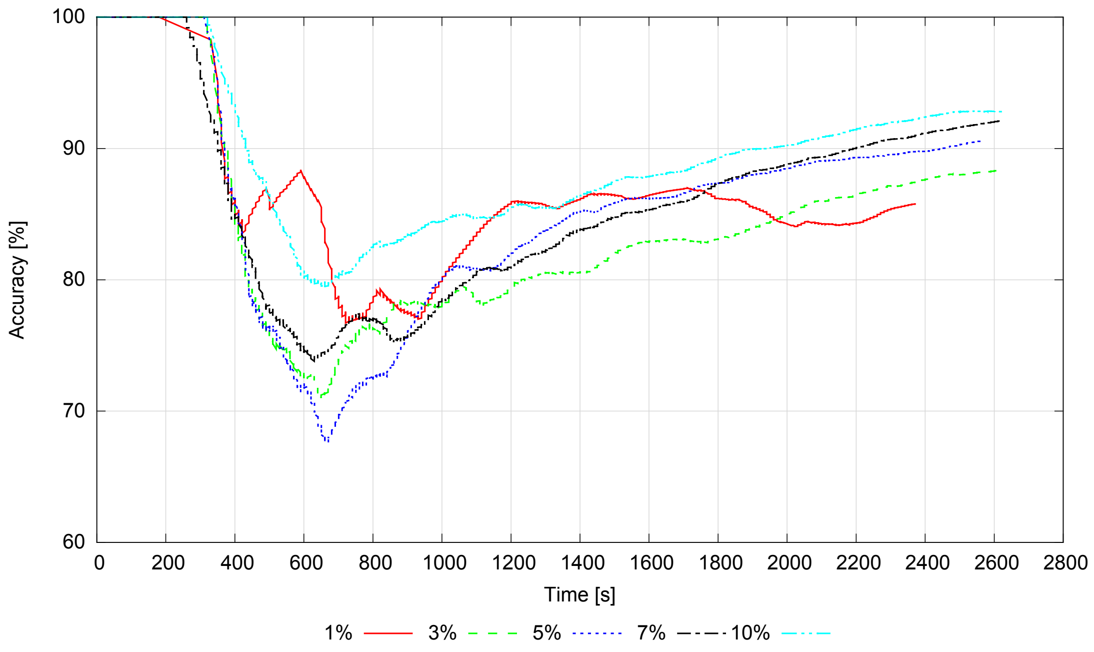


Figure 6.10: Accuracy for Varying Penetration Rates

lead to the assumption, that the penetration rate has no substantial influence on the accuracy. Again, the specificity is the most influential indicator, showing the same pattern as the overall accuracy, while the sensitivity is only partially conclusive: it can be observed, that the higher penetration rates of 5%, 7% and 10% deliver better results than the lower rates. Though 5% and 7% perform better in the beginning, 10% has a steeper learning curve and seems to catch up in the end.

Section Length

The section length influences the amount of data that passes through the system, because smaller sections lead to a higher amount of mining instances. Furthermore, with smaller sections the real queue-end can be approximated to a higher degree if classified correctly, because the mean distance between the real queue-end and the forecasted queue-end (at the middle of the section) is smaller. Due to the lack of empirical data, we selected section lengths in a way, that results in a tolerable mean error of distance to the real queue-end (at most half of the section length). We experimented with 30 m, 50 m, 100 m, 150 m, and 300 m. We expected that a maximum in accuracy can be identified for a certain section length for the following reasons. Too small sections produce a too small amount of messages which are not sufficient to determine whether there is a queue-end or not. Too long sections are expected to be harmful to accuracy, because they can contain too many messages, which do not indicate the correct class. We present the sensitivity results in Figure 6.11. In contrast to our expectations, it can be observed that for values between 30 m and 150 m there is no substantial difference neither in accuracy nor in specificity. However, sensitivity results get better with increasing section length. As a consequence, we should switch to a section size larger than our current default (100 m) when deploying the system and make additional experiments with higher section lengths to see if we reach a maximum.

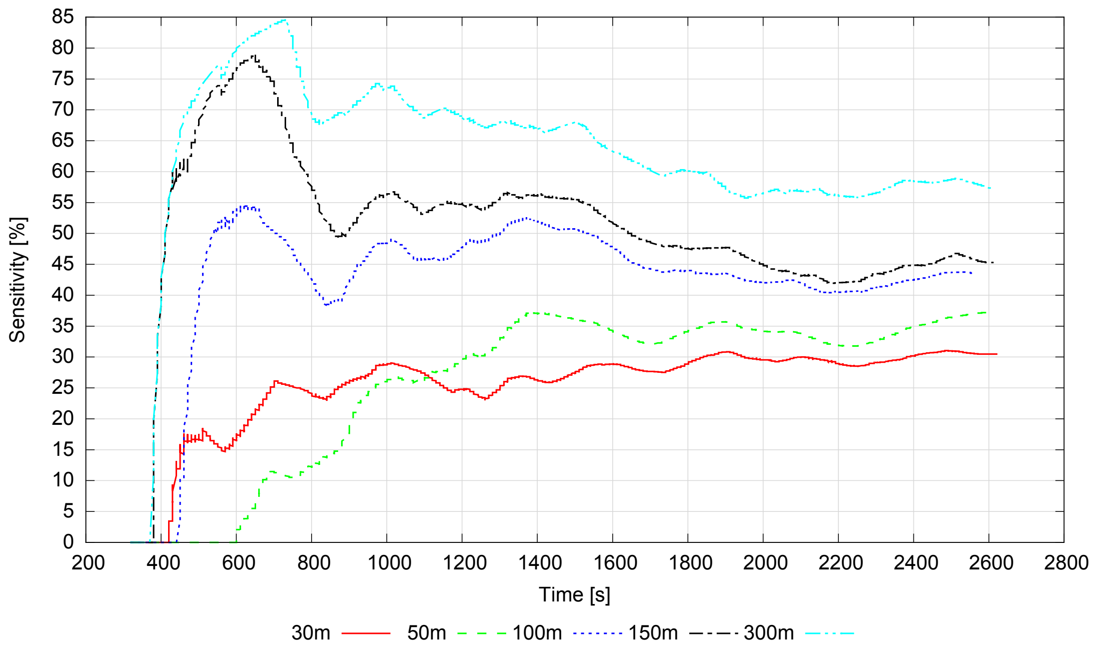


Figure 6.11: Sensitivity for Varying Section Lengths

Discussion of Results

In conclusion, the first results of the experiments show trends of influential parameters and reveal promising accuracy values also for smaller penetration rates. However, the results are not sufficient to draw final conclusions as we need to evaluate different traffic scenarios and fine tune the data stream mining algorithm. In particular, the low sensitivity rates and inconclusive results need further investigation. Balancing positive and negative examples seems to be a promising direction for improving sensitivity, but it requires further investigation such that the overall accuracy is not decreased. We will further investigate these issue in the next chapters. In combination with the DQ management a more detailed analysis is possible by then.

6.3 Traffic State Estimation

In the following we first discuss the goals of the Traffic State Estimation (TSE) application and the task requirements. Hence, we again follow the process model, but only stress the differences to the QED application as both share similar properties and hence, a similar setup for the TSE application can be used. We approach the TSE application with a partly data-driven approach as in the case of the QED as we have given the data sources, but we are free in the modeling and usage of the sources.

6.3.1 Goal and Task Requirements

Traffic State Estimation (TSE) is determining the current traffic situation on a road or highway based on current and historical traffic data (cf. Chapter 2). It is the second application evaluated in the CoCarX project regarding the question if we just can use CoCarX messages for this and which kind of CoCarX messages are sufficient (only event messages or is probe data also required?). A further question is if additional data from cellular probes (Floating Phone Data) is beneficial for the accuracy of the results.

Table 6.7: Required Output for QED Scenario

Name	**Description**	**Data Type**
SectionID	The ID of the section whose traffic state is determined.	Integer
Traffic State	One of the four traffic state classes.	String
Confidence	A confidence value, which indicates how reliable the detected state is.	Float

In our approach, traffic state estimation is the determination of a class representing the current traffic state or *Level of Service (LOS)* for each section of a link. There have been several approaches for modeling these levels (see Section 2.1.2). In this approach, we use a traffic state model, which has been described by the Federal German Highway Research Institute in (BASt, 1999) and has been widely implemented in German Traffic Management Centers. It distinguishes four levels of traffic: free, dense, slow-moving and congested traffic. We use these four levels as the classes for data stream mining.

Hence, the output of this application can be defined as given in Table 6.7.

The QoS requirements are not as strict as for QED. Real-time traffic state estimation usually operates in ranges of half a minute to a few minutes. Though, due to the data volume a high throughput should be provided to keep latency relatively low. Provenance is not required. End users could be users of navigation software and devices or visitors of websites which show the current and prospective traffic situation.

The next section describes the requirements for the additional required data sources.

6.3.2 Data Source Requirements

For the TSE application also the V2X messages from the CoCarX project have to be used. Furthermore, an interesting source we wanted to examine in the project are Floating Phone Data (FPD). FPD are anonymously collected positions of mobile phones. From these positions other traffic data, such as the vehicle speed, can be derived. The derivation of traffic information from anonymously located handsets (i.e., mobile devices which are able to connect to a cellular network) is a vibrant research field. Most approaches using Floating Phone Data include the following steps (Smith et al., 2004):

1. Localization of the handset as exact as possible.
2. Map matching to determine the position in a road network.
3. Derivation of traffic information using the estimated positions, e.g., link speed.

Of course, the derived information is not exact. The precision of the estimated values is influenced by several factors. There has already been a lot of related work done investigating the feasibility of using mobile phone data for traffic parameter estimation and we discussed the measurable parameters and techniques in Chapter 2. All in all the reviewed works agree in the feasibility and the potential FPD can have for traffic applications. The feasibility is also shown by commercial systems, such as TomTom HD Traffic^{TM} (TomTom, 2010).

In previous works (Geisler et al., 2010; Chen, 2009) we also investigated factors influencing the accuracy of FPD. These insights can be used in designing FPD as a data source for the TSE application. We require the FPD to include the position of the handset, a timestamp, and some message identifier. For the sake of realism no further information should be included. In the next section we will describe, how this data is simulated using VISSIM.

6.3.3 Data Quality Requirements

DQ requirements for TSE do not differ much from the ones for QED. We also need some confidence value for the overall result, and we need accuracy values for speed and positions, and ratings for data volume and timeliness.

6.3.4 Selection and Modeling of Data Sources

For the V2X message nothing changes for the TSE application. The messages will be issued by vehicles on events or periodically as probe data. The generation of Floating Phone Data is also simulated in the VISSIM traffic simulation. Based on our previous work we use several factors which influence their generation. The number of persons in a vehicle is fixed by the traffic simulation. We use a probability function to assign a handset to those persons. That means, every time a vehicle is spawned in the simulation, the persons in the vehicle get assigned a mobile phone with a certain probability. A list of those vehicles and handsets is kept until the vehicle disappears. A second probability function determines when a handset is sampled, i.e., if it issues a message including its position. The positions are degraded according to a distribution error function to simulate the varying quality of positioning for cellular networks.

For the TSE application we use an artificial, but more complex road network than the one used in the QED scenario. It is depicted in Figure 6.12.

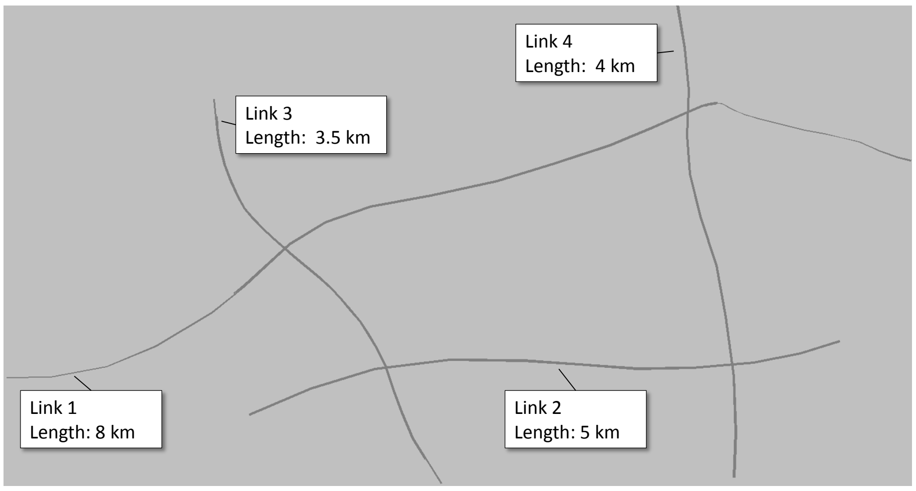
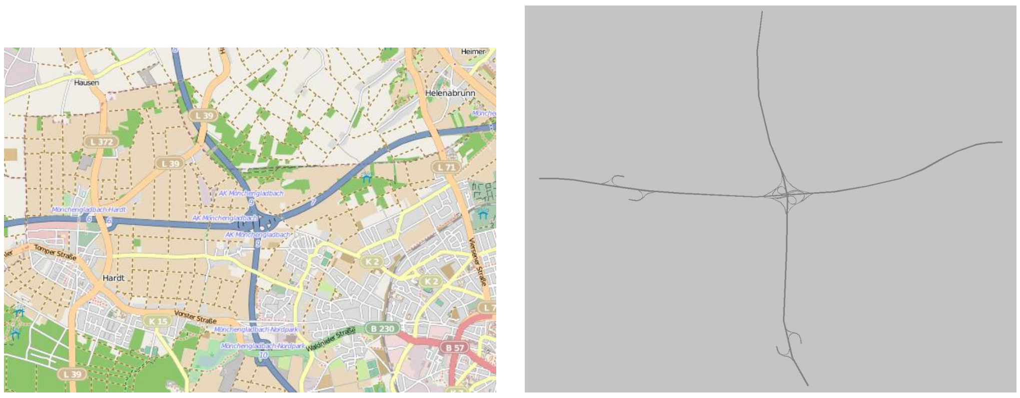


Figure 6.12: Artificial Network for Traffic State Estimation

The network consists of four links with two lanes each and covers an area of approximately 4 km by 8 km. For the links there is no speed limit defined by default. One difficulty when working with classification algorithms in traffic applications is, as we have seen from the QED application, the balancing of training examples. There have to be sufficient examples for each of the traffic levels, such that all are well learned by the algorithm. This is challenging in simulation-based studies, as we have to prompt the simulation to produce these states in a very realistic manner. Congestion can be provoked in VISSIM by several means. One possibility is to use a stop line to block a lane. This forces the vehicles to stop before this line, or if possible, change the lane, but leads to less realistic driver behavior (no zip merge). This can be avoided by using lane reduction, i.e., a link with two lanes is connected with a link with one lane, thereby also modeling a blocked lane. To create slow-moving traffic, reduced speed areas (a defined part of a link with a speed limit) in combination with varying traffic volumes can be utilized.

To provoke an even more realistic scenario, we import a real road network from the area of Mönchengladbach, where several highways cross. The network is exported from OpenStreetMap[22] (OSM), but VISSIM does not provide a direct import of OSM networks. Hence, we have written our own tool which converts the network from OSM to ANM which can then be imported into VISSIM. The former map in OSM is shown in Figure 6.13(a) and the resulting network in VISSIM is shown in Figure 6.13(b). It includes the road network topology and important things such as signaling.



(a) Road Network in OpenStreetMap (b) Road Network in VISSIM

Figure 6.13: The Imported Road Network

Furthermore, we implement the definitions of the traffic levels to generate ground truth class values in the traffic simulation. We utilize the definitions from (BASt, 1999), where for each level and a number of lanes a threshold or a range is defined for the mean speed and the local density at a stationary sensor. For example, for a highway with two lanes the traffic state is *slow-moving*, when the mean speed is above or equal 30 km/h and below 80 km/h and the density is below or equal 60 veh/km. In VISSIM it is possible to measure traffic data on a section basis, e.g., measuring the mean speed for all equal sized sections of a link. These values have been used to calculate the ground truth classes for traffic states online and they are send as ground truth messages to the DSMS. The sections in VISSIM comply with the sections in our spatial database.

6.3.5 Task Modeling and Implementation

We again utilized GSN as DSMS and could reuse the sensors from the QED application. Differences exist in the ground truth messages send by VISSIM.

These are constituted of the following fields:

[22]`https://www.openstreetmap.de`

$$\texttt{TrafficStateGT}(\texttt{Timed}, \texttt{TS}, \texttt{LinkID}, \texttt{SectionID}, \texttt{TrafficState})$$

In the DSMS, the CoCar messages and ground truth messages are integrated and aggregated using a 60 s window which slides every 10 s. For the data stream mining evaluation in this scenario, we used the prequential method described in Section 4.4 with a window size of 20 elements.

As link and section are already known, no additional map matching to get the specific section is necessary. All other steps are the same: the traffic data is aggregated and integrated with the ground truth messages. The elements are fed into the data stream mining sensor and either used for training or for classification to determine which traffic state is present on the section at hand. Finally, the results can be exported. The exported data elements are used for the visualization of traffic states on a real-time map.

For evaluation and demonstration purposes we implemented a third map which is imported from OSM to VISSIM. It contains several highways and primary roads around Düsseldorf, Germany, as 1394 links in an area of about 10 by 5 km. The network implemented in VISSIM is depicted in Figure 6.14.

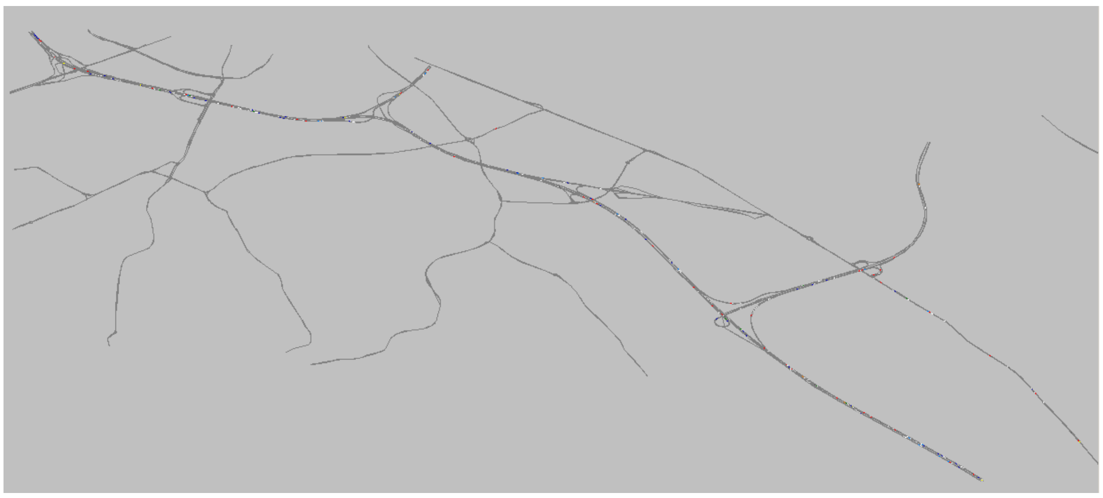


Figure 6.14: The Highway Crossings around Düsseldorf imported into VISSIM

The visualization of traffic states is depicted in Figure 6.15, where green is free-flow, yellow is slow-moving traffic, and red is congested.

6.3.6 Evaluation and Results

The goal for the evaluation is to test how well the application is able to estimate traffic states for all road sections based on the given traffic data. Furthermore, it should be determined which parameters influence the results and if additional data sources are beneficial. In the evaluation of the TSE application, we record the number of combinations of correct and predicted classes (traffic states) of the classification result in a confusion matrix which is shown in Table 6.8.

The columns are the predicted classes and the rows the correct classes. The values a_{ii} on the diagonal represent the number of correctly identified traffic states for each class. All other cells represent the number of elements which have been falsely predicted to class `p<Class>`, but should be `c<Class>`. In the example table, a_{12} is the number of

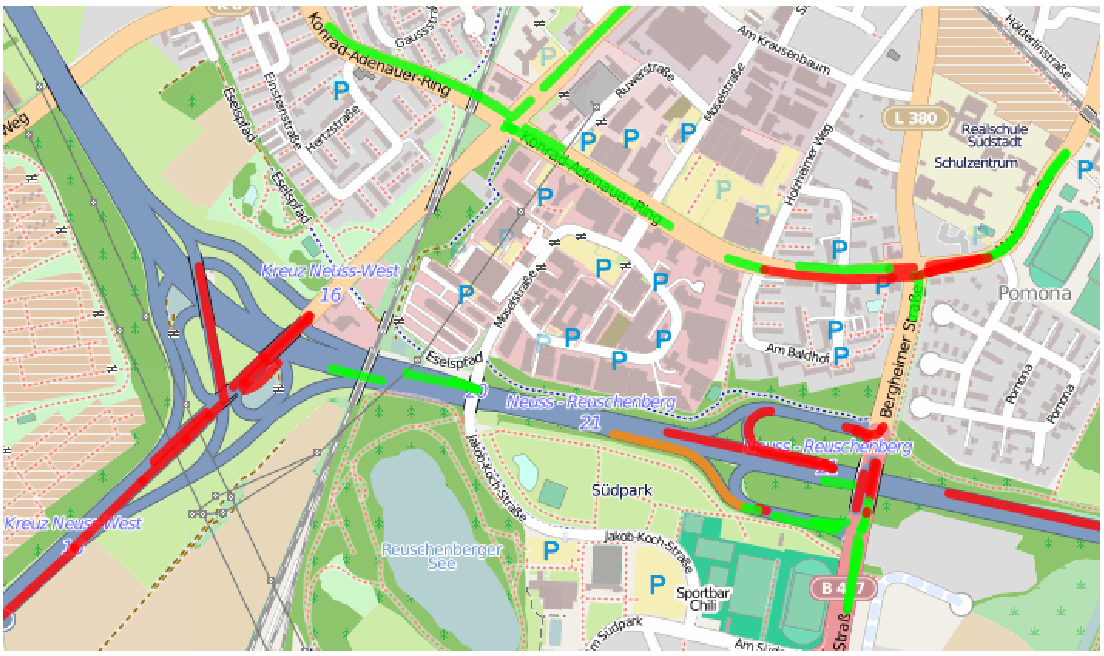


Figure 6.15: Traffic State Visualization in OSM

Table 6.8: Confusion Matrix for Traffic States

	pFree	pDense	pSlow	pCongested
cFree	a_{11}	a_{12}	a_{13}	a_{14}
cDense	a_{21}	a_{22}	a_{23}	a_{24}
cSlow	a_{31}	a_{32}	a_{33}	a_{34}
cCongested	a_{41}	a_{42}	a_{43}	a_{44}

elements which have been predicted with class `dense`, which were actually class `free`. With the recorded values we can now calculate the mining accuracy. For each class the number of correctly classified elements in this class is divided by the overall number of classified elements. To reward, when the algorithm was only slightly wrong, but not completely wrong, we introduce a constant factor 0.6 with which we multiply the count of similar traffic state classes and add the result to the number of correctly classified elements. The resulting equations for the accuracy in each class are defined as follows:

$$\begin{aligned}
FreeAcc &= \frac{a_{11} + (0.6 \cdot a_{21})}{a_{11} + a_{21} + a_{31} + a_{41}} \\
DenseAcc &= \frac{a_{22} + (0.6 \cdot (a_{12} + a_{32}))}{a_{12} + a_{22} + a_{32} + a_{42}} \\
SlowAcc &= \frac{a_{33} + (0.6 \cdot (a_{23} + a_{43}))}{a_{13} + a_{23} + a_{33} + a_{43}} \\
CongestedAcc &= \frac{a_{44} + (0.6 \cdot a_{34})}{a_{14} + a_{24} + a_{34} + a_{44}}
\end{aligned}$$

The overall accuracy is calculated by:

$$Acc = \frac{(a_{11} + (0.6 \cdot a_{21})) + (a_{22} + (0.6 \cdot (a_{12} + a_{32}))) + (a_{33} + (0.6 \cdot (a_{23} + a_{43}))) + (a_{44} + (0.6 \cdot a_{34}))}{\sum_{i,j=1}^{n} a_{ij}}$$

The evaluation process comprised several simulation runs with varying parameters for the traffic simulation and the data stream mining algorithms. Each run started with a 420 s setup phase in which the network was populated with traffic. Afterwards, in the evaluation phase of 1800 s the data stream processing and mining was carried out to analyze the traffic state estimation accuracy. For each run a classifier was learned from scratch. If not varied in a test, we used for the segment length 400 m and 10% CoCar penetration rate as default values. The concept-adapting Hoeffding Option Tree with Naïve Bayes prediction strategy was used as the default classifier as it delivered the best results in comparison to other algorithms. The detailed results of the evaluation of data stream mining algorithms are presented in Chapter 10.

Inclusion of Probe Data

In the first tests we made, we only used EBL and WLA messages and the corresponding data for aggregation and mining. As these messages are only created in situations with congested traffic, the class `congested` has been classified in these tests in the mean with an accuracy over 96% while other classes had a very bad accuracy. Therefore, the simulation was extended by introducing an additional CoCar message type, the Vehicle Probe Data (VPD) message type. These messages are sent by the vehicles periodically, i.e., every 10 s, and only contain basic information such as position, speed, and acceleration. This enabled the mining algorithms to learn the other classes with a higher accuracy. Furthermore, the preliminary traffic volume of 2500 veh/h turned out not to be sufficient to learn dense traffic very well, but only free flow. This was due to the fact, that with 2500 veh/h only at the location where the link was narrowed congested traffic was created and the traffic was flowing freely otherwise. To produce also situations with dense and slow-moving traffic, the traffic volume was increased to 3500 veh/h. However, this led to false classification of many elements with class `dense` to class `free` and the few examples for the class `free` were not sufficient to learn this class with an acceptable accuracy. Therefore, different traffic volumes were used on different links to balance the number of examples for each class. Still, the algorithms were not able to distinguish properly between the classes `dense` and `free`. As the ground truth also includes the density as a distinctive property for the different traffic states, we added the number of VPD messages as a new input attribute to the training examples. This led to an increase in accuracy for the classes `free`, `dense` as well as `slow-moving` as can be seen in Figure 6.16 for dense traffic. The diagram also shows the increase in accuracy between 2500 veh/h and 3500 veh/h. Hence, in further experiments we used VPD messages and the VPD message count as an attribute.

Section Length

As seen from the queue-end detection case study, the section length can have an impact on the mining accuracy. Therefore, we also experimented with varying section lengths in this scenario. Section lengths between 100 m and 1000 m have been tested, geared to the distances between inductive loop detectors on German highways. It turned out that longer section lengths are less beneficial for the accuracy as depicted in Figure 6.17. While lengths between 800 m and 1000 m produce results of around 80%, lengths of 200 m and 400 m deliver the best results of approximately 87%. This is due to the fact, that in longer sections the probability is higher that they include more than one traffic state class. Thus, a section length of 400 m is used as a default parameter in further tests. These results may also depend on the traffic volume in the network and therefore,

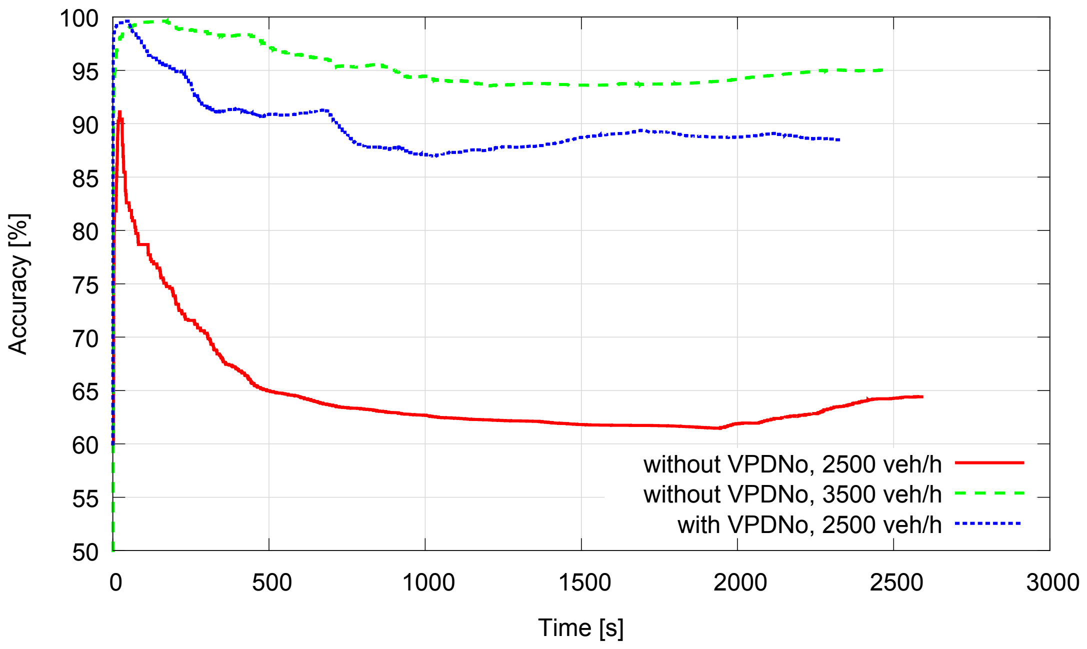


Figure 6.16: Accuracy for Dense Traffic with Varying Traffic Volume and with and without VPD Message Count as Attribute

further tests with varying traffic volumes and section lengths have to be carried out.

CoCar Penetration Rate

It seems to be plausible that, with an increasing penetration rate, more information is available in each section and therefore, the mining accuracy should be higher. In the results of the queue-end detection case study, the CoCar penetration rate turned out not to have too much impact on the mining accuracy. To see if this is also the case for the traffic state estimation, we tested CoCar penetration rates between 5% and 20%. Figure 6.18 indicates that the above assumption has also been proven wrong for traffic state estimation. The results vary slightly between 88% and 91%, but no clear trend can be identified. However, an increased penetration rate inevitably leads to a higher network coverage and hence, is always rated as beneficial.

Applicability to Realistic Maps

Using artificial maps for the training of a model is beneficial, as traffic situations can be controlled to create sufficient examples for every kind of traffic state class. However, these networks may not reflect traffic situations as they would occur in a realistic road network. Hence, we rebuild a German highway interchange nearby Mönchengladbach, where the highways A52 and A61 are crossing. The network covers a 3 km by 3 km area and includes all links, slip roads and exits, as depicted in Figure 6.13(b) with the corresponding OpenStreetMap excerpt[23] in Figure 6.13(a).

In the simulation runs the traffic volume of each link was varied to simulate realistic traffic situations under varying loads in the network. We tested traffic volumes of 2000 veh/h to 5000 veh/h, which were the same for all links in the same run. For the

[23]`http://www.openstreetmap.org`

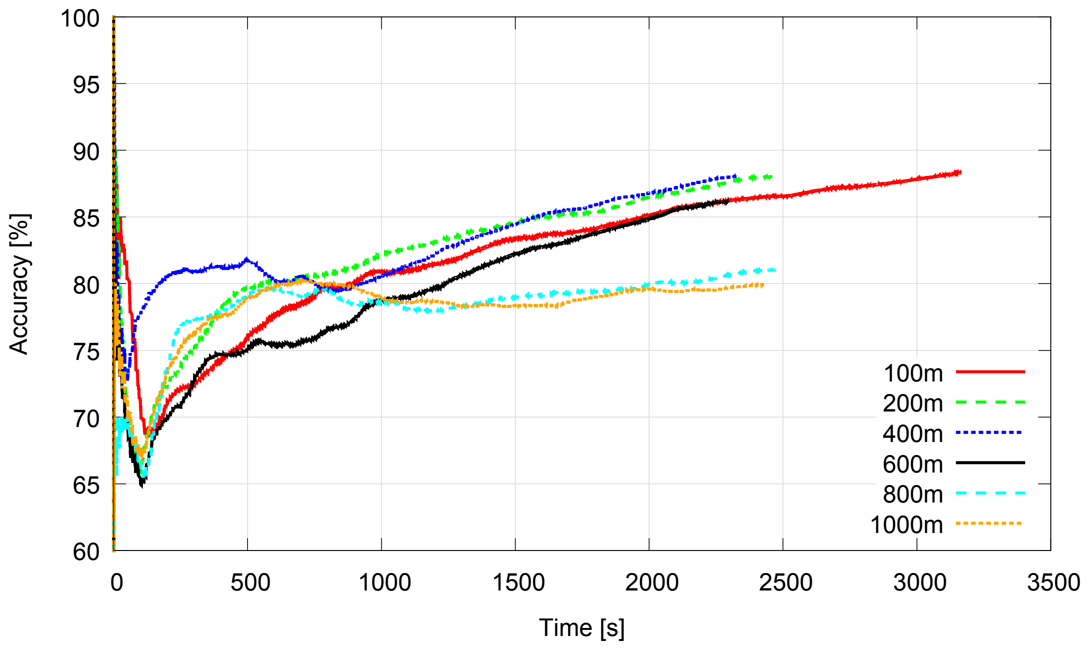


Figure 6.17: Accuracy for Varying Section Lengths

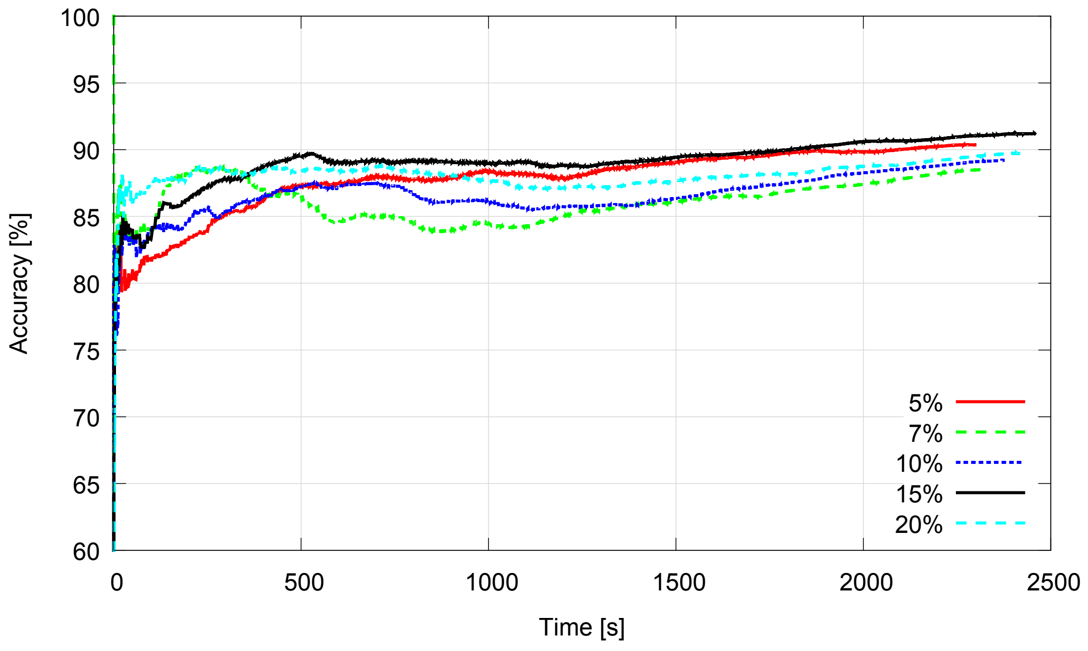


Figure 6.18: Accuracy for Varying CoCar Penetration Rates

tests, a trained model with very good accuracies in the tests with the artificial network was used. The model was only used for classification and was not further trained during these runs. In Figure 6.19 the results show, that for a very low traffic volume of 2000 veh/h the model can estimate the traffic state with a very good (98%) accuracy. A very high traffic volume of 5000 veh/h reaches an acceptable value of (87%) accuracy. 3000 veh/h and 4000 veh/h reach 83% to 84%. This can be explained by a higher rate of dense and slow-moving traffic occurring in these simulation runs. The accuracy for these classes was also lower in the artificial network, as it is easier to predict the 'extreme' situations (free and congested traffic) than to predict dense and slow traffic.

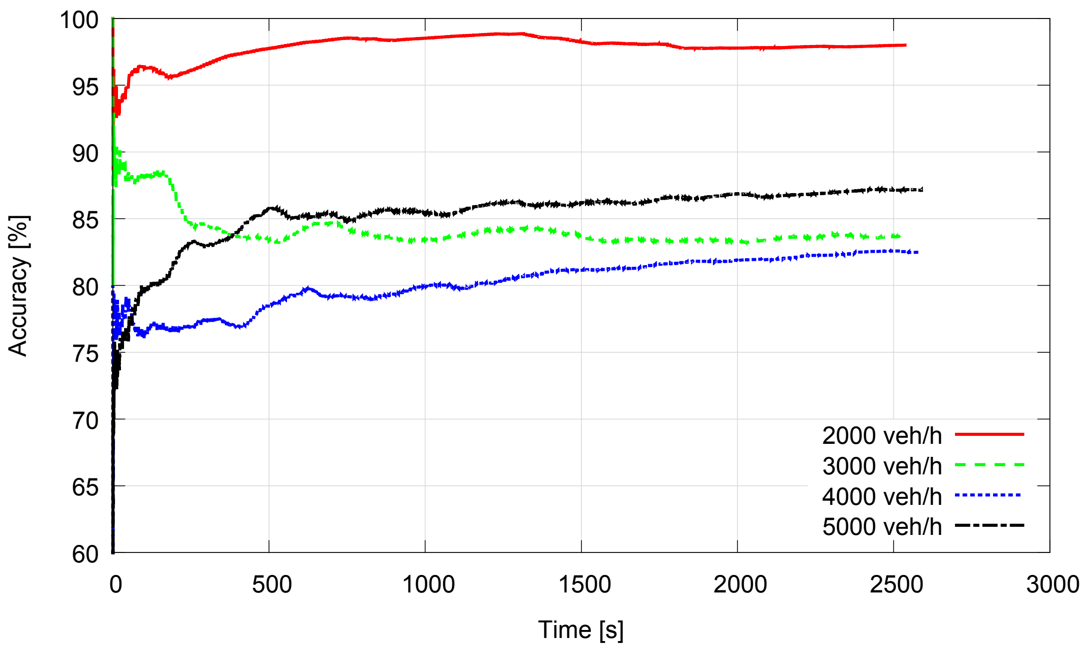


Figure 6.19: Accuracy Using a Trained Classifier on a Realistic Map for Simulation

As in the case study of queue-end detection, for traffic state estimation we also observed that imbalanced training data harms the results of the classifier. Therefore, the artificially road network is constructed in such a way that training data for the four traffic classes are generated almost equally. The last experiment has shown that the classifier trained on the artificial network delivers also very good results on a real road network.

In general, the results for traffic state estimation are more conclusive than for queue-end detection. We were able to identify a turning point for the section length (400 m) and higher traffic volumes generate a higher accuracy. In addition, if the classifier is trained in multiple runs, we can see that after the second run, the classifier is very stable and the accuracy changes only very little at a level of 90%.

Performance

Performance of data processing is a crucial aspect in data stream management systems as a huge amount of data has to be processed in a short time. Hence, we measured the processing time of the data stream elements for each sensor using the timeliness dimension. This means, that for each V2X message the time between its creation in the traffic simulation and the processing time in the sensor is calculated. When data is aggregated over a window, the timeliness of the oldest stream element in the window is used as the timeliness value. Figure 6.20 shows the timeliness over simulation time for the Düsseldorf case study smoothed with a Bézier function. In this performance experiment in a 40 minutes simulation run, about 19000 CoCar messages have been processed, aggregated to nearly 50000 data stream elements which have been subsequently mined by a data stream mining algorithm. It can be seen from Figure 6.20 that the sensors before aggregation only need a few milliseconds to process the stream elements, where the processing time is slowly increasing with simulation time. In the aggregation sensor, data of the last 120 seconds has been aggregated, hence the timeliness values are higher per se.

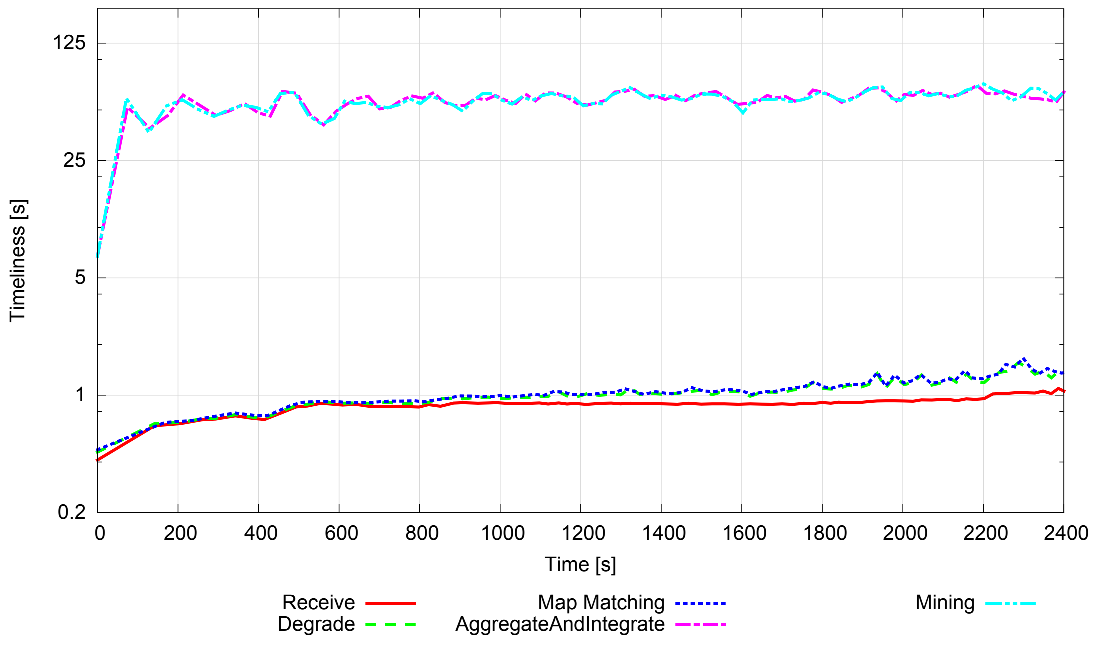


Figure 6.20: Timeliness of Stream Elements Measured over Simulation Time

6.4 Conclusion

We have presented the design, implementation, and evaluation of two C-ITS applications along our process model. The guidance by the process model was very helpful in many aspects. The description of data source requirements and their modeling supports the detailed and structured documentation of the data sources. The task requirements and modeling is similar to requirements engineering in software engineering projects and helps to reflect about the goals of and expectations towards the application. Task, data sources, and data quality requirements could easily be held separately. However, the description of DQ requirements and DQ modeling is sometimes mixed up and it took some time (and changes in the description of the process model) to find a good way to separate them. There is still room for improvement, for example, with respect to writing the Data Source documentation, which was a bit tedious and led to a long document. This could be alleviated by the use of special tools and graphical representations of the data source and DQ models (as indicated in Chapter 8). The evaluation framework has been filled by components suitable for the implementation of the C-ITS applications. This worked for the two case studies very well - they utilize the same data sources, but for example different outputs. The guidance of the framework helped to organize the components accordingly.

Still, the process model provides a substantial guidance in developing data stream-based applications as the planned and structured procedure helps to order thoughts and development steps. A preliminary structured evaluation of the applications was carried out by changing several parameters in the system and in simulation, but without using dedicated data quality management. To make more complex evaluations, to employ real DQ monitoring, and to implement automated DQ improvement, we need DQ management integrated into the data stream management process. We present the description of the DQ management implementation for the case studies in detail in Chapter 8, where the data quality methodology and framework are first introduced in detail and afterwards evaluations are carried out. A detailed evaluation regarding Map Matching algorithms and the proposal of our own online Map Matching algorithm are delineated in Chapter 9. The analysis of the performance of the data mining algorithms, i.e., which parameters are the best, which algorithm suits best for which applications, does balancing and handling of concept drift help, are presented in Chapter 10.

Chapter 7

Validation of the Process Model in the mHealth Domain

In this chapter we will demonstrate the independence of our approach from the domain and its general flexibility according to the properties of the applications. In the last chapter we showed, that for C-ITS applications the process model conveniently can be used to design, implement, and evaluate the applications. The specific circumstances in the C-ITS domain required the use of simulation data to test influencing parameters. We were more or less free in the choice of data sources, while the task was fixed. In the case studies we carry out in the mHealth domain the situation is different: we have real-world data to analyze but the tasks are less strictly defined. In the following we first describe each project context and then follow the methodology to design, implement, and evaluate the according stream applications.

7.1 The UMIC HealthNet Project

The HealthNet project was part of the research excellence cluster Ultra-Highspeed Mobile Information and Communication (UMIC).[1] HealthNet was a joint research project of several chairs of the RWTH Aachen University from electrical engineering, mechanical engineering, and computer science. In the project a wearable textile platform was designed, which enables an unobtrusive measurement of health parameters, such as ECG, pulse, or skin temperature. The measured parameters are processed to summarize and predict further parameters which are of interest for sports, fitness, or health care applications. The HealthNet architecture is presented in Figure 7.1.

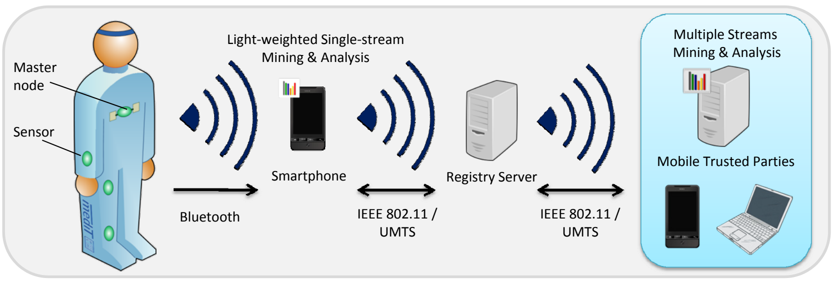
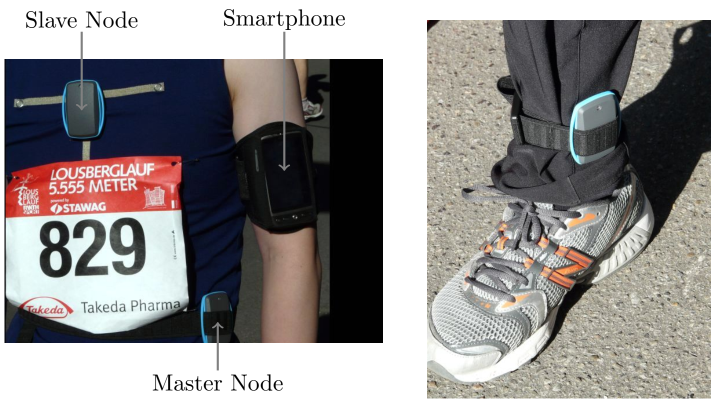


Figure 7.1: Architecture of the HealthNet System (Quix et al., 2013)

[1] https://www.umic.rwth-aachen.de

The textile platform is a shirt in which conductive yarns have been used for signal reception and transmission. Furthermore, the user wears a body sensor network (BSN) called IPANEMA (Kim et al., 2009). The BSN is very flexible as it may consist of multiple slave nodes carrying different kinds of sensors and a master node which communicates with the slave nodes and sends the gathered data to another entity, such as a mobile device, via Wireless LAN. In the HealthNet project the BSN sends the data to a smartphone, where a mobile application processes it. The setup worn by the user is depicted in Figure 7.2.



(a) Master and Slave Sensors, Mobile Device at Upper Body (b) Slave Sensor at Leg

Figure 7.2: HealthNet Body Sensor Network

On the smartphone the data is aggregated and displayed to the user. Additionally, the first steps of a multi-step prediction algorithm are executed which tries to predict parameters. The mobile application architecture is modular, such that many different applications can be implemented using the architecture. It comprises a central HealthNet controller organizing data distribution, a cache, and a windowing mechanism for sensor data, and a data transmission unit. Optionally, the data is forwarded by the mobile device to another mobile device (e.g., a trainer device) or a server, where a detailed analysis of the data and prediction is executed. The mobile device offers four transmission modes to authorized receivers: on request, periodically, direct transmission, and manual transmission. The architecture of the mobile application is shown in Figure 7.3.

The analysis of the data uses a multi-step anytime classification algorithm with incremental learning and a distributed sensor clustering algorithm for aggregating similar sensor data (Quix et al., 2013; Hassani and Seidl, 2012; Kranen et al., 2010). As a proof of concept the architecture has been used to implement a runner application. We made several interviews with runners and trainers from the RWTH Aachen University Sports department to elicitate the overall requirements according to the runner application. Each runner of a team wears the setup of shirt, BSN, and smartphone as depicted in Figure 7.2. On the phone runners can observe their own data, such as position or heart rate, while running. They can switch between different kinds of views, such as data view, or map view. A trainer can observe the data of all team members on a mobile device with the corresponding trainer application as presented in Figure 7.4. Humidity, pulse, distance, speed, and temperature have been measured, and the estimated arrival time has been predicted. The application has been showcased and tested in the Lousberg Run, a local running event in Aachen, in 2011 and 2012. We recorded the data of the Lousberg Runs for later analysis. The data is used to find new stream applications.

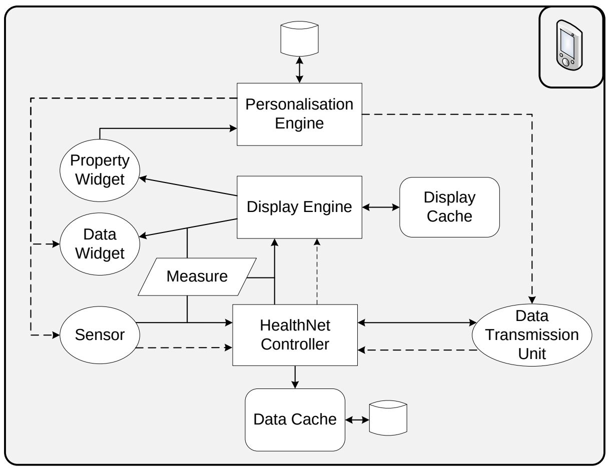
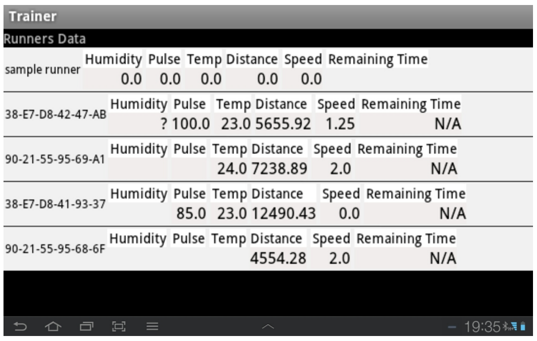


Figure 7.3: Architecture of the Mobile Health Application

In the following we will detail the process of application design and implementation.



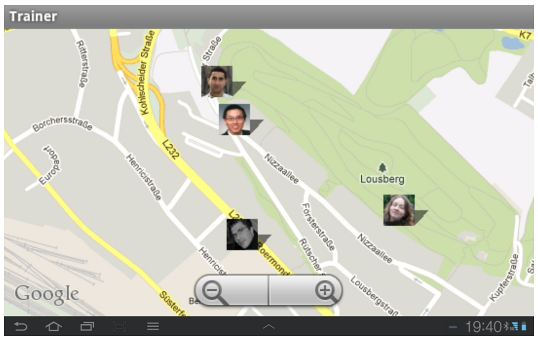
Trainer

Runners Data

	Humidity	Pulse	Temp	Distance	Speed	Remaining Time
sample runner	0.0	0.0	0.0	0.0	0.0	
38-E7-D8-42-47-AB	?	100.0	23.0	5655.92	1.25	N/A
90-21-55-95-69-A1			24.0	7238.89	2.0	N/A
38-E7-D8-41-93-37		85.0	23.0	12490.43	0.0	N/A
90-21-55-95-68-6F				4554.28	2.0	N/A

19:35

(a) Data View



(b) Map View

Figure 7.4: HealthNet Trainer Application

7.1.1 Goal and Task Requirements

In the HealthNet application no clear goal regarding data analysis has been given. We know, that it is helpful to observe the data of the runners for the trainers and the runners themselves, but there is still much room to think of potential streaming applications. Hence, we can say, that the stream application is *completely data-driven and explorative*. We are quite limited to the data delivered by the mobile device. Based on literature research and on the interviews with the sports people we identified two potential applications considered to be useful using the combination of shirt, BSN, and mobile device:

- **Emergency & Physical Overload Detection:** The application detects a physical overload and warns the sports person and optionally a trainer to avoid injuries and states of exhaustion. If an emergency is detected an automatic emergency message can be issued to the trainer or an emergency instance. Especially in runs and trainings with amateur athletes this regularly happens. A very good example is the Lousberg run 2010, where 23 people where treated by health professionals due to the hot weather, four had to be brought to hospital, and one even had to be reanimated in the finish area. They were suffering from circulatory collapse and feeling of faintness (Zander and Eimer, 2010). Hence, the detection of such events in beforehand or right away would be beneficial for runners. In the following we will term this kind of application *Overload Application*.

- **Real-time Monitoring & Automated Training Advices:** Based on the observed health data, the application can give training advices regarding the pace. Furthermore, the data could be recorded to be used for follow-up analysis of trainings and runs. A real-time comparison with runs in the past on the same track could also be useful according to the interviewees. This application will be referred to as *Monitoring Application*.

For both applications the output may look similar. For the first application we propose to send a message with the estimation of the runner state (emergency or overload) and either the raw data or the aggregated data (averages over the last X minutes) and a confidence value can be included in the message. For the second application the raw data or aggregated data should be output to allow online analysis and/or storage in another data management system or file. Regarding Quality of Service, it is crucial for the case of the Overload Application that the throughput and results are produced near real-time to enable fast help and countermeasures. Also the confidence value has to be high enough to avoid false alarms. Provenance is also a crucial issue in a productive system especially for the Overload Application.

7.1.2 Data Source Requirements

We are limited in the choices of our data sources for the two applications. The central data source are data streams sent by each runner's mobile device. So far, for the Overload Application we only can rely on the health parameters measured with the sensor shirt and the mobile device. The data source delivers data elements which have the schema described in Table 7.1.

The described data source comprises five data sets (one for each runner) from the Lousberg run 2011, where each data set has on average only 335 data elements. The sampling rate is 10 seconds to keep communication costs low and spare battery life. The runs have an overall duration of 63 minutes on average. The data has been recorded in csv files. Additional data sources for both applications could be humidity and temperature of the environment from weather stations, track information from map services, and historical data of former runs. Also information manually entered by the user, such as recovery pulse, or ranges for pulse in training could be helpful.

7.1.3 Data Quality Requirements

Regarding data quality two extremes clash. On the one hand, in the HealthNet project and the running applications we deal with very unreliable sources. Due to the movement of the runner, the sensors may lose contact and the produced signal is distorted. Furthermore, the applications are mobile and based on cellular networks. Hence, the UMTS signal may be weak or lost during the run and GPS fixes may also be incomplete. Figure 7.5 shows the UMTS coverage on the track of the Lousberg run. The color scale is ranging from green (best signal strength) to red (no signal).

The figure shows, that the UMTS signal has only seldom full strength and that there are some spots where there is no signal at all. This leads to elements arriving too late on connected applications, such as the trainer app. On the other hand, health applications,

Table 7.1: Data Element Schema Produced by the HealthNet Architecture

Field	**Description**	**Data Type**	**Example**
id	ID of the message	Integer	121
runner_id	The unique hexadecimal ID of the runners device.	String	38-E7-D841-93-37
acc_x	Acceleration sensor value representing the x-axis.	Float	1.424
acc_y	Acceleration sensor value representing the y-axis.	Float	-2.424
acc_z	Acceleration sensor value representing the z-axis.	Float	-0.48
temperature	Measured skin temperature of the runner.	Float	24.0
humidity	Current skin humidity of the runner.	Float	1.0
ecg	Current ECG value measured with a three lead placement of ECG sensors.	Integer	1303
heartrate	Currently measured heart rate of the runner.	Float	68.0
longitude	Longitude value of the current GPS position.	Float	6.079763174
latittude	Latitude value of the current GPS position.	Float	50.78586817
speed	Currently measured speed of the runner in km/h.	Float	1.5
distance	The covered distance since start of the measurement in meter.	Integer	4317

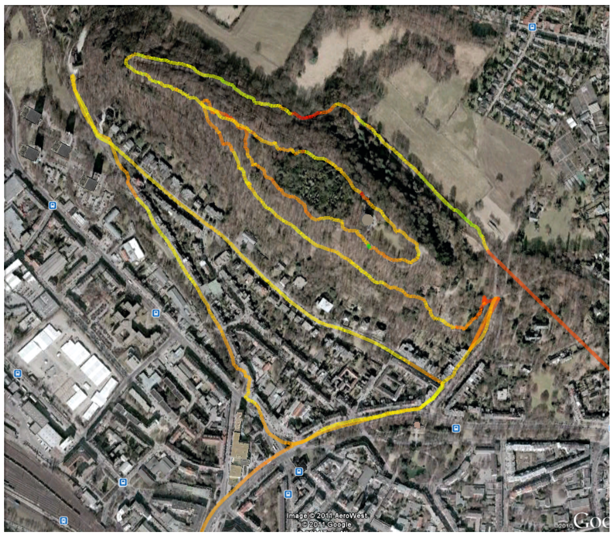


Figure 7.5: UMTS Coverage on Lousberg

such as the Overload Application require reliable information and statements. Hence, data quality management is also useful in these applications to rate the data used to create a certain information. Similar to the C-ITS applications there are some quality dimensions generally relevant to the domain and some relevant only to specific applications.

We did a literature research to find DQ dimensions specifically important in the health domain. The papers analyze studies according to the dimensions which have been rated as the most important and frequent ones. Table 7.2 summarizes our findings.

Table 7.2: DQ Dimensions in the Health Domain

	(Mawilmada et al., 2012)	(Hogan and Wagner, 1997)	(Thiru et al., 2003)	(Liaw et al., 2013)
Timeliness	✓		✓	✓
Relevance	✓			✓
Completeness	✓	✓	✓	✓
Accuracy	✓	✓	✓	✓
Consistency	✓		✓	✓
Precision	✓		✓	
Reliability	✓		✓	✓

Hence, it can be said, that completeness and accuracy are most important, but also timeliness, consistency, and reliability are relevant. For the Overload Application surely timeliness, consistency, and completeness (for the entire stream element, but also for individual attributes, such as gps) can be considered. For the monitoring application it is not really clear, what is important, as it depends on what exactly is monitored and what is done with the data. For sole monitoring the general dimensions already named are sufficient.

7.1.4 Data Source Design and Implementation

As we use real data with a given schema, the design of data sources is very limited. No simulation is involved and no artificial data is added. To make experiments with the data from the five runners in the Lousberglauf with a real-time data management system, we need to replay the data in real-time preserving the original time periods between the elements. The data was originally stored in csv files. We import the data into five tables, one for each runner, in a PostgreSQL database. We wrote a simple Replay Tool which is capable of replaying data from one or more tables in a database. The data of each table is streamed over a separate TCP port. Additionally, it can be configured if the original timestamps or current timestamps should be used (but still the period between the messages is preserved). Figure 7.6 shows the simple user interface of the Replay application.

7.1.5 Task Implementation

We again use the GSN system for data processing as we also work with flatly structured data elements easily transferable into a relational schema. The setup of the application is presented in Figure 7.7.

For each of the simulated mobile devices we provide an instance of the same wrapper which receives the data elements and converts them into data stream elements. The

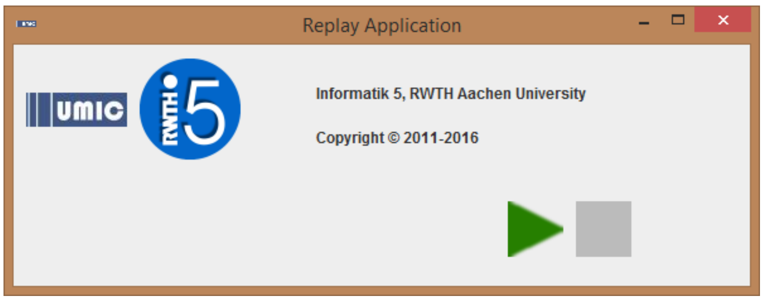


Figure 7.6: A Simple Replay Application

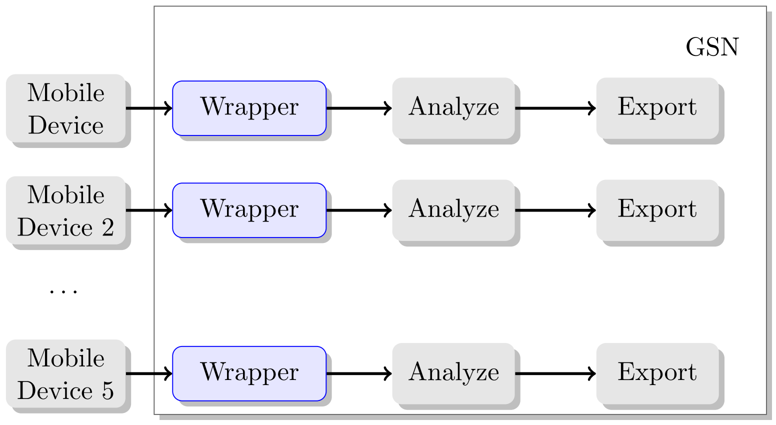


Figure 7.7: The Setup of the Lousberg Scenario

elements are analyzed depending on the application (represented by the Analyze virtual sensors). If the analysis of a common value would be required, the data from all five runners could be integrated by a single virtual sensor located after the analyze sensors. Finally, one or more export sensors deliver the output of the application to a consuming instance, e.g., a web service. We did only first a simple implementation for monitoring to preliminary check the data and its quality. Based on these findings we will proceed to the detailed implementation.

7.1.6 Data Quality Implementation & Evaluation

For the two applications (Overload and Monitoring) the heart rate definitely is a crucial parameter. From the interviews and literature research we know that runners try to keep their heart rate in a certain range in training to reach optimal training results and to improve their performance. During contests the heart rate is interesting to avoid exhaustion and critical situations. Hence, we first start with analyzing the heart rate and its consistency. The heart rate normally lies in certain physiological ranges. We define the following metric to rate the plausibility (or consistency) of the heart rate value according to physiological ranges:

- $HR >= 60 \wedge HR <= 160 : Plausibility = 100$
- $HR >= 0 \wedge HR < 60 : Plausibility = \frac{HR}{60}$
- $HR > 160 \wedge HR <= 260 : Plausibility = \frac{HR}{100}$

- $HR < 0 \vee HR > 260 : Plausibility = 0$

This metric creates values between 0 and 1 representing the percentage. Similarly, we define a dimension and metric for speed consistency. The speed is calculated based on distance and time and given in km/h. For the position we check, if the position fields contain values or not, represented by the latitude field. The attached dimensions are exemplified for one runner in Figure 7.8.

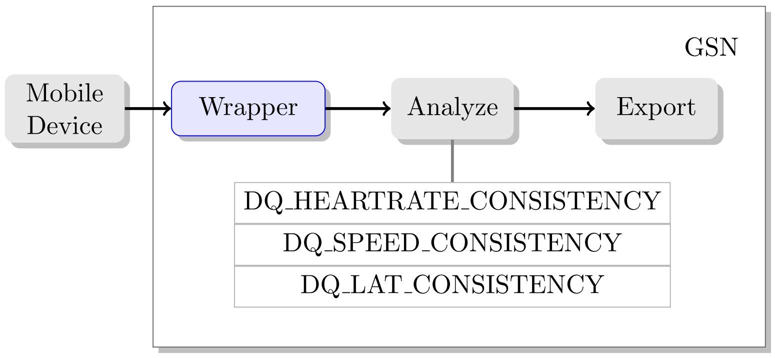
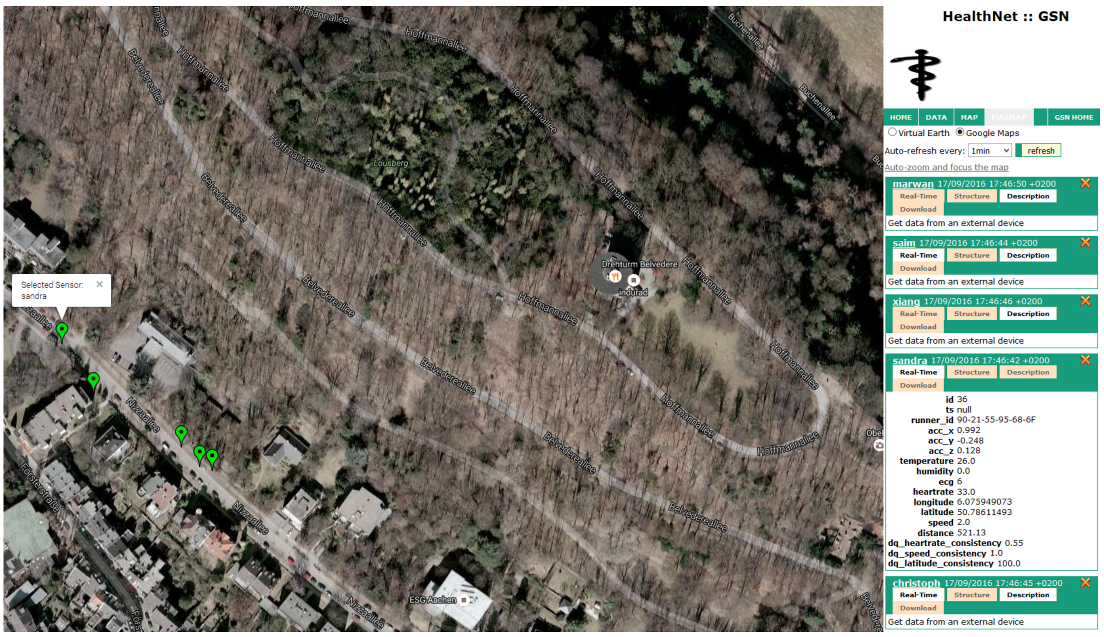


Figure 7.8: The Setup of the Lousberg Scenario incl. DQ Dimensions

Figure 7.9 shows the representation of the replayed data in GSN including the data quality and visualized on a map.



○ Virtual Earth ● Google Maps
Auto-refresh every: 1min refresh
Auto-zoom and focus the map

marwan 17/09/2016 17:46:50 +0200
Get data from an external device

saim 17/09/2016 17:46:44 +0200
Get data from an external device

xiang 17/09/2016 17:46:46 +0200
Get data from an external device

sandra 17/09/2016 17:46:42 +0200

```
                       id 36
                       ts null
                runner_id 90-21-55-95-68-6F
                    acc_x 0.992
                    acc_y -0.248
                    acc_z 0.128
              temperature 26.0
                 humidity 0.0
                      ecg 6
                heartrate 33.0
                longitude 6.075949073
                 latitude 50.78611493
                    speed 2.0
                 distance 521.13
dq_heartrate_consistency 0.55
dq_speed_consistency 1.0
dq_latitude_consistency 100.0
```

christoph 17/09/2016 17:46:45 +0200
Get data from an external device

Figure 7.9: The Lousberg Run Replayed in Real-time in the GSN System

The detailed implementation of the DQ processing and the evaluation using the DQ values are presented in Chapter 8. In the next section we will show a second example for the implementation of a streaming mHealth application along the process model.

7.2 The MAS Project

The MAS project (Nanoelectronics for Mobile AAL (Ambient Assisted Living) Systems[2]) developed an AAL system with a communication platform and sensor devices for collecting vital parameters of a patient. The goal of the project was to detect problems in the cardiovascular system very early by monitoring the health status of a patient. Hence, critical situations might be avoided and a more individual treatment can be enabled. The system architecture is similar to the HealthNet architecture described in Section 7.1. In MAS, modern mobile sensors have been used, such as an in-ear pulsoxymeter. Various vital parameters are measured by the sensors, for example, oxygen saturation, or heart rate. The sensor data is passed to a master node which sends the data via Bluetooth to a smartphone. The smartphone does some preprocessing on the data and sends it over an Internet connection to a server. On the server side, the data can be analyzed and a medical expert can investigate critical situations in more detail. As the sensors might not produce correct data due to movement of the patient or a slight incorrect placement of the sensor, incorrect sensor data has to be taken into account. It is therefore important to measure the quality of the sensor data in such a context. Other problems, such as failing Internet connection, has to be taken into account also.

7.2.1 Goals and Task Requirements

Similar to the HealthNet project we use a fixed data source consisting of a stream of sensor data from one person wearing the pulsoxymeter. Hence, we also adopted the completely data-driven, explorative approach as the task was not given in beforehand. The only application goal was to monitor the person and gather the data in real-time for online monitoring and later analysis. The Quality of Service requirements comprise the near real-time processing of the data. Provenance may be an issue in a later productive system, when transparency of the result origin is important.

7.2.2 Data Source Requirements

The data source is given by the measurements carried out in the MAS project. The measurements were made with a pulsoxymeter which measures heart rate and oxygen saturation by sending light beams through the skin (e.g., on the finger or on the ear) and measuring the difference in the reflected light received by the oxymeter. The data has been recorded with a new pulsoxymeter device developed in the MAS project. The data has been recorded over a time period of about 40 seconds. In this time, 2990 PPG measurements have been produced, which result in an average data rate of about 100 elements per second. The data source schema is structured as given in Table 7.3. The data was recorded in a csv file.

7.2.3 Data Quality Requirements

As the sensors might not produce correct data due to movement of the patient or a slight incorrect placement of the sensor, incorrect sensor data has to be taken into account. It is therefore important to measure the quality of the sensor data in such a context. We concentrate in this case study on the PPG data as it is used to determine other parameters, such as the heart rate. Although there are differences in the PPG

[2]`https://www.fit.fraunhofer.de/de/fb/life/projects/mas.html`

Table 7.3: Data Element Schema Retrieved from the MAS PPG Sensor

Field	Description	Data Type	Example
pk	The ID of the measurement.	Integer	22
timed	The timestamp of the measurement in ms.	Integer	1380039350867
pulse	The current heart rate of the patient.	Integer	62
spO2	The current value of the oxygen saturation of the patient.	Integer	98
ppg	Photoplethysmogram value measured by the pulsoxymeter. It represents the absorption of light by the skin send out by the pulsoxymeter. For each cardiac cycle a peak is shown in the PPG.	Integer	155

data of various patients (intersubject variabilities), the PPG of one person should have a regular waveform. Each peak represents a cardiac cycle. Hence, the regularity should be reflected in a consistency or regularity DQ value. Furthermore, the timeliness of the messages could be very interesting to determine the throughput and response capacity of the system.

7.2.4 Data Source Design and Implementation

As the data source was fixed, we only needed to recreate a real time scenario with the real-world data where several runs for tests can be realized. Hence, we again use the Replay application to simulate the real measurements. We described the schema already in Section 7.2.2. The data is imported into a PostgreSQL database in one table. The Replay application uses the timestamps in the data to exactly reproduce the time periods between the measurements. We again use the GSN system as we also work with flatly structured data elements easily transferable into a relational schema.

7.2.5 Task Implementation

To implement the application, which analyzes the uniform periodicity of the PPG measurements, we designed a simple network consisting of a wrapper and virtual sensors in GSN depicted in Figure 7.10.

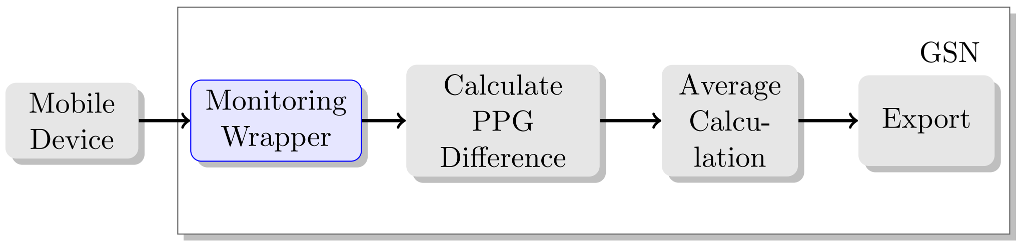


Figure 7.10: The Virtual Sensor Network of the PPG application

Our approach to estimate the uniform periodicity of PPG values calculates the difference between two consecutive sensor measurements. If the average difference over a window of 50 elements is high, this indicates low data quality. This rule can be expressed by the formula in Equation 7.1

$$\text{consistency} = 100 - (\max(\frac{\sum_{i=2}^{k=i+49} \text{PPG}_i - \text{PPG}_{i-1}}{50} - 5; 0) * 10) \qquad (7.1)$$

We integrate a tolerance value of 5 and a minimum value of 0. To get a percentage value we multiply the result by 10 and subtract it from 100.

The mobile device of the user sends the measurement data of the pulsoxymeter via TCP to the *Monitoring Wrapper* which converts the data elements into data stream elements processable by GSN. The first virtual sensor (*Calculate PPG Difference*) implements the first window on the incoming data stream elements with a window size of 2 and calculates the difference between the current and the former PPG value. The difference is added as a new field into the schema of the data stream. The next virtual sensor (*Average Calculation*) just retrieves the data from the previous virtual sensor and enables the second windowing in the DQ management to calculate a consistency value.

7.2.6 Data Quality Implementation and Evaluation

As we have already mentioned in the requirements, the task is to find out, how regular the periodicity of the PPG measurements is. This regularity is a sign for a healthy cardiovascular system and good quality of the sensor measurements, as related parameters calculated from the PPG value are more reliable. For the DQ processing we implement the consistency measure as described in Equation 7.1. The detailed implementation of the DQ processing and the evaluation is presented in Chapter 8.

7.3 Conclusion

In this chapter we presented the application of the process model to two case studies from two projects in the domain of Mobile Health. The case studies were simpler than for the C-ITS domain, but still they showed, that the process model was also applicable without problems though the applications had different requirements and starting points for the design. The data sources delivered real-world data, where parameterization is difficult. Hence, evaluation cannot be steered as in the C-ITS case, but experiments must be designed to consider the given circumstances, such as slow or fast moving runners. Both cases are supported by the process model, reflected by the design of more or less options for parameterization and iterations. Even unclear application goals as in the case of the HealthNet project could be mapped by the process model. Additionally, the evaluation framework could also be applied to the specific requirements of the mHealth applications. Some of the components had to be realized differently, but core components, such as the DSMS, have been kept. It can be concluded, that the process model and the framework are well applicable to different domains and case studies with different demands.

Part III

Data Quality Management & Structured Evaluation of Algorithms

Chapter 8

Data Quality Management

We have stressed in Chapter 4 and Chapter 5 that data quality makes a difference in data intensive applications and it should be considered in the planning of an application in which sense and degree DQ should be monitored and evaluated. To support the DQ monitoring, evaluation, and optionally countermeasures, the data stream architecture must provide a corresponding framework. In this chapter we propose a framework to enable DQ management in DSMS. The framework allows for an easy definition, configuration, and control of data quality values.

In the following section, we will discuss, which requirements are posed to a DQ framework for DSMS and the solution we propose to fulfill these requirements.

8.1 Requirements

Data quality management is not only an issue in DSMS. Data quality is one of the most important success factors of an application. Poor quality may lead to dissatisfied users, financial losses – in the worst case it can also risk people's health and lives (Redman, 2004; Wang and Strong, 1996; Judah and Friedman, 2014). Especially in the transportation context, safety applications require high quality output based on data with corresponding quality. In our work, we adopt the definition of data quality as "fitness for use" as this notion of quality applies to products as well as to data (Juran, 1999; Redman, 1996). As a consequence, DQ cannot be defined by a single measure, rather multiple facets must be considered to assess the quality of the corresponding data (Wang and Strong, 1996). The complexity of DQ definition and measurement requires a well-defined methodology to establish DQ management techniques in a system. These methodologies are often adopted from product quality management models.

However, existing DQ methodologies (Batini et al., 2009), such as the TDQM (Total Data Quality Management, (Wang, 1998)) model, are not applicable in the context of data streams as the analysis and improvement steps do take not into account the dynamic and continuous nature of a DSMS (cf. Section 4.3). Actions to improve DQ in a DSMS need to be performed during runtime while data is being processed; thus, the system should be able to invoke methods *to improve DQ automatically*, if the DQ measurements are not within the desired range. Though the concrete actions cannot be adopted as such, it is also reasonable for a DQ framework in DSMS to require *a structured process for DQ* in which the main phases of design, measurement, analysis, and improvement are included.

DQ measurements can be very complex and might require *more expressivity* than offered in a simple SQL query. For example, in traffic information systems, positions from GPS sensors (Global Positioning System) have to be matched to a particular part of a road (Map Matching). Such applications require *application-specific DQ metrics and DQ dimensions*, because the rating of the matching confidence can only be rated by the method itself. Redman (2013) even describes DQ as subjective – its definition may vary from case to case and user to user. Therefore, we aim at a DQ framework which is *easily extensible and is not restricted to certain quality dimensions.* Since DQ is multi-dimensional (i.e., an element might have several DQ values in different DQ dimensions), the computation of a DQ value based on individual DQ values should be possible. This computation should be done by *user-defined evaluation formulas.* Hence, users can adjust the computation of the composite DQ value according to their needs and personal preferences. In principle, a general method has to be found, which enables to implement any dimension and any metric independent of the application field.

Furthermore, DQ management in data streams should be *holistic* and not only focused on the quality of the output stream. It is important to *monitor DQ throughout the whole DSMS*: incoming tuples may already contain quality information which needs to be maintained as data "flows" through the DSMS. This also includes that *semantics of query operations* (which might require a recalculation of the DQ values, e.g., by aggregation, joins or similar operations) has to be considered in the DQ management. Additionally, new DQ dimensions have to be added which either can be computed based on existing DQ values, or by application-specific functions, or by semantic rules. Thus, to provide quality information in each data processing step, a solution must facilitate the extension of data stream elements with DQ values.

The knowledge about the quality of the utilized data helps to rate the significance of calculation results based on this data. Besides the sole observation of decreasing or low DQ, it is desirable to take *countermeasures to improve the quality* if possible. Hence, a DQ framework should also include a comfortable way to define rules which specify the DQ values that are checked regularly. Furthermore, it should be possible to define actions which should be executed when a rule is violated.

A data analyst may also be interested in data quality measures which involve the windowing of the original data, e.g., an average value of a parameter over the last five minutes or 50 elements. The windowing may only be required for data quality analysis and should not be carried out when this DQ measurement is not needed. Hence, *a flexible solution to create a data window for analysis purposes* should be provided.

DSMSs belong to the group of real time systems (Stonebraker et al., 2005), which are operating in narrow time frames and memory bounds. Hence, the integration of DQ processing needs to *satisfy real time requirements* and may not pose significant additional overhead to memory usage or CPU time. Also a *maximal degree of automatism* is required. There is no time for manual processes which redefine the data flow in a DSMS. Furthermore, the DQ management should be an optional feature – if DQ information is not required or desired, it should be possible to *turn off the DQ features* without affecting the data processing.

Finally, to allow *easy customization and extension*, the DQ framework must be metadata-driven, i.e., the processes for measuring DQ should be configured by metadata that is managed in a central component. This enables also the separation of the definition of the DQ-related functions and the processing of data streams.

In the following section, first related work for DQ frameworks in DSMS will be discussed. Subsequently, the overall methodology will be presented in Section 8.3. Finally, the proposed data quality framework for DSMS will be described in detail.

8.2 Related Work

DSMSs can implement means to monitor the system performance during query processing and adapt the system configuration if certain quality criteria are not met. This is usually summarized under the term Quality of Service (QoS). For example, if the system is overloaded and the output delay or the throughput is low, a DSMS can drop tuples with sampling techniques (e.g., in the Aurora system (Abadi et al., 2003b)). In the QStream system (Schmidt et al., 2004), two descriptors for each output stream are defined – a content quality descriptor which includes the dimensions inconsistency and signal frequency defined by Schmidt (Schmidt, 2006), and a time quality descriptor consisting of values for data rate and delay. The quality dimensions are calculated throughout the whole query process. All operators in the query execution plan comprise functions used to calculate the value of each quality dimension when new tuples are processed. At the end of the processing chain quality values are output for the result streams of each continuous query. In Borealis (Abadi et al., 2005), the QoS model of Aurora has been refined to rate QoS not only for the output, but also for each intermediate operator. To this end, each tuple includes a vector with quality dimensions, which can be content-related (e.g., the importance of an event) or performance-related (e.g., processing time for an event up to this operator). The vectors can be updated by operators in the query execution plan and a scoring function is provided (Abadi et al., 2005).

A crucial limitation of the previous approach is that the quality dimensions in the vector are equal for each stream, which does not allow for an application-specific DQ rating of the stream content. To achieve a more fine-granular rating of DQ on attribute, tuple and window level and to also include application-specific DQ dimensions, Klein and Lehner (2009) extended the PIPES system (Krämer and Seeger, 2009) with modified operators. DQ information becomes thereby a part of the stream schema. Klein et al. distinguish four different types of operators based on the operator's influence on the stream data (modifying, generating, reducing or merging operators). Changes on the data can in turn result in updates of the DQ of an attribute, tuple, or window. In addition, the size of their DQ windows (the part of the stream for which data quality is evaluated) is dynamically adaptable based on an interestingness factor, e.g., the window size is decreased when there are interesting peaks in the stream data. A drawback of the approach is the deep integration of DQ features into the operators. The implementation of operators has to be changed substantially to include the quality information. Fiscato et al. (2009) integrated data quality into the data model of streams as defined in the STREAM system (Arasu et al., 2003a). They extended the definition of a stream by three DQ dimensions (weight, recall, and utility) which are fixed. Using this definition the authors integrated DQ into the data model, but without providing flexibility. Kuka and Nicklas (2014b) extend the DSMS Odysseus by features for probabilistic processing and quality processing. They extend their data stream model by creating tuples with three parts: one for the metadata of the stream element with an validity time interval and an uncertainty value for the existence of the tuple. The second part includes quality values for continuous values calculated by Gaussian Mixture models and an online Expectation Maximization algorithm. The third part includes the common relational representation of the tuple for discrete values. Continuous values are represented by a reference to the second part. Similar to the system by Klein and Lehner (2009), relational operators of the Odysseus system are adapted to the specific data stream model, which integrates the DQ management deep into the system, sacrificing flexibility and generality. Kuka and Nicklas (2014a) propose an ontology-based quality management extension to the Odysseus system. The quality assessment is integrated into a map operator which adds quality values to each data stream element based on metadata stored in an ontology. However, quality measures can only be defined by relational predicates with Boolean result or with intervals, which restricts the possibilities of DQ values.

8.3 Data Quality Management Methodology for DSMS

Common methodologies in DQ management have been approved in industry and research. Hence, it is reasonable to utilize this knowledge and extract the aspects which are suited for a data stream setting. We chose to adapt the aforementioned TDQM cycle (see Section 4.3.1) to data streams. We adopt the four phases *Design, Measure, Analyze, and Improve*, but add an additional step in the cycle. Furthermore, for the data stream setting, we distinguish between steps which have to be done manually during design time and steps which are automatically executed during runtime. For the four phases the following actions are desired:

1. **Define:** Similar to other models, in this phase, the requirements for the applications to be implemented are acquired. Based on the requirements, DQ dimensions and corresponding metrics are defined. Furthermore, mappings between the dimensions, metrics, and data stream components, e.g., attributes have to be specified. These steps are done during design time. During the runtime of the system no definition step is required. Also, rules for the automatic analysis and improvement can be defined in this step.

2. **Measure:** Based on the defined mappings between dimensions, metrics, and stream components, the corresponding DQ values can be measured. This has to be done during runtime of the system, as the DQ values have to be as current as the underlying data.

3. **Analyze:** The measured DQ values are also evaluated during runtime. The online analysis enables the system and users to react timely to low DQ values, which is a big advantage compared to classical DQ methodologies. The rules for the analysis are defined during design time to enable an automatic monitoring of DQ values.

4. **Improve:** There are two levels of improvement possible. First, measures defined during design time, e.g., the activation of additional data sources, are automatically executed at runtime, when DQ monitoring in the Analysis phase has determined a decrease in DQ values. This online improvement can help to fix problems, which have been identified during the Define phase, as soon as they occur. The effects of the improvement, if existing, can in turn be detected in the Measurement phase. Secondly, of course also offline improvement of processes, data sources, etc. can be done when reviewing the measured DQ values. Finally, the meta-data defined in the Define phase can be updated to improve the runtime DQ measurement, analysis and improvement processes, e.g., a missing dimension can be added or a new countermeasure can be defined.

The complete cycle is depicted in Figure 8.1.

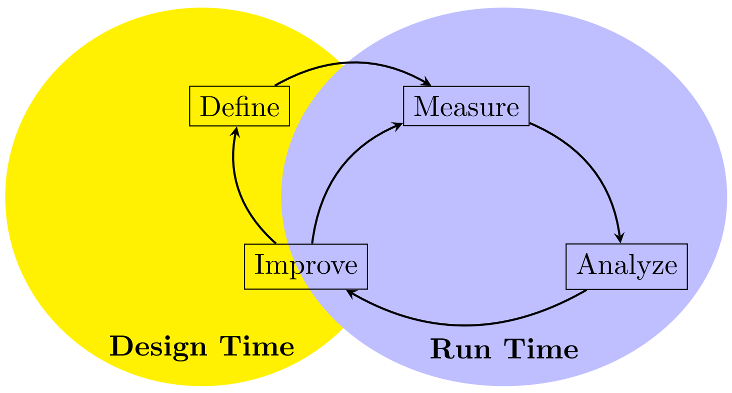


Figure 8.1: DQ Management Methodology for Data Streams

In the subsequent section, an overview of the architecture of the DQ framework is described. All architecture components are classified into the four phases of the methodology.

8.4 Ontology-based Data Quality Framework Architecture

Based on the requirements, we identified three major aspects in data stream applications and data stream processing, for which the DQ of elements and attributes have to be calculated. First, the content of the data stream and its elements should be rated according to their semantics. This can be done by *semantic rules*, which describe the acceptable ranges for values of one attribute or values of combinations of several attributes in one data stream element. Such rules are *semantic* because they use the domain specific semantics of a data item. For example, a rule could define an upper bound for speed measurements of vehicles. If a vehicle exceeds a speed of 280 km/h, the DQ value for this attribute should be very low. DQ measurements of this type are done by a *Content-based Quality Service*. Second, to rate the DQ in each step of the data processing, also the query processing operators (e.g., join, aggregation) in a DSMS have to be taken into account, as they can alter the DQ values of stream elements flowing through the system. Hence, a framework should incorporate means to analyze the queries and recalculate quality values accordingly. We assign these tasks to a *Query-based Quality Service*. Finally, in data stream applications customized and application dependent data processing and analysis is implemented which may also add or change DQ values. These tasks are subsumed under the component *Application-based Quality Service*. Additionally, a *Data Quality Monitor* observes the data quality values and invokes actions if necessary to counteract low quality. The metadata of these services and of the quality monitor is stored in the DQ ontology; the overall architecture is depicted in Figure 8.2.

We propose a DQ framework which extends a DSMS, i.e., the DQ Framework is an (optional) add-on to the DSMS. We adhere to relational DSMS and the well-known multiset or bag semantics for data streams we introduced in Chapter 4. Hence, a single data stream is defined by a set of attributes (the schema) and consists of data stream elements or tuples adhering to the schema. When incorporating DQ, the set of data stream attributes can be extended by a set of DQ attributes, where each attribute repre-

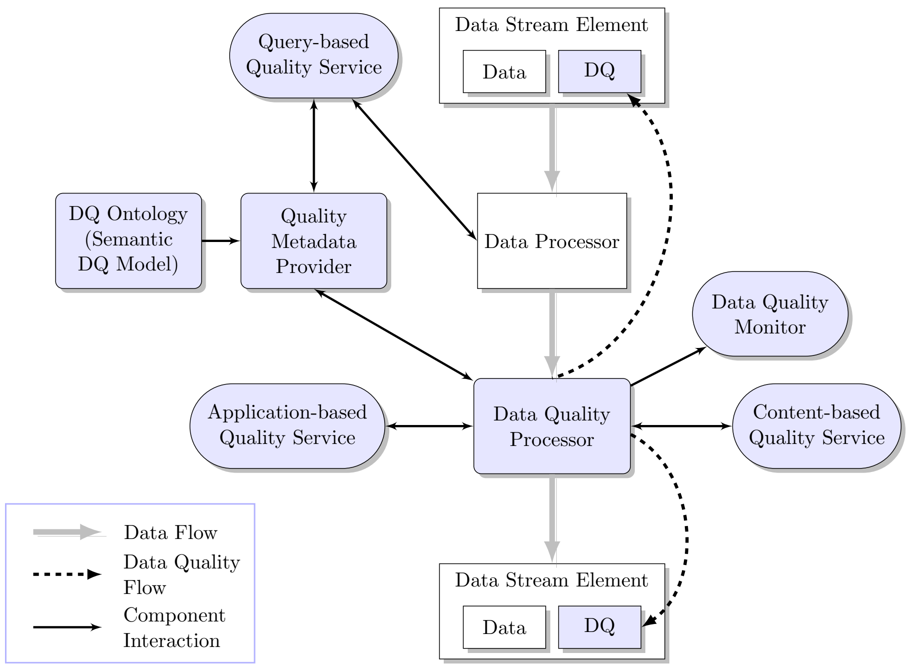


Figure 8.2: Architecture of the Ontology-based DQ Framework

sents a DQ dimension. We define these streams as *quality-affine data streams* as follows:

Definition 8.1 (Quality-affine Data Stream)**.** *A* **quality-affine data stream** S_{dq} *of a stream* S *is a (possibly infinite) multiset of data stream elements* (s, τ, q)*, where* $\tau \in T$ *is a timestamp attribute with values from a monotonic, infinite time domain* T *with discrete time units* $\in \mathbb{N}$*.* s *is a set of attributes* $(A_1, A_2, \ldots, A_n)$ *with domains* $dom(A_i), 1 \leq i \leq n$*, constituting the schema of the stream* S*.* q *is a set of attributes* $A_{dq_1}, A_{dq_2}, \ldots, A_{dq_m}$ *with domains* $dom(A_{dq_j}), 1 \leq j \leq m$*, representing the measured data quality dimensions. Together* s *and* q *constitute the schema of* S_{dq}*.*

Syntactically, quality attributes are not distinguishable from data attributes. Semantically, they are handled differently and must be recognizable during data processing. For instance, a Car-2-X message data stream (the messages include information from vehicle sensors) could be extended by DQ information in the following way:

$$\texttt{Message}_{dq} = (\texttt{Timed}, \texttt{ID}, \texttt{Latitude}, \texttt{Longitude}, \texttt{Speed}, \texttt{Acceleration}, \texttt{RoadID}, \\ \texttt{Timeliness}_{dq}, \texttt{SpeedAccuracy}_{dq})$$

$\texttt{Timeliness}_{dq}$ rates the age of the message and $\texttt{SpeedAccuracy}_{dq}$ rates the error of the measured speed. The aforementioned quality services manage the DQ attributes in a data stream element. The metadata is managed in an ontology that provides a *DQ ontology* (or *Semantic DQ Model*). The *Quality Metadata Provider* acts as an interface to this ontology. The *DQ Processor* invokes the *Content-based* and *Application-based Quality Services* as required by the definitions in the ontology. The *Query-Based Quality Service* interacts directly with the *Quality Metadata Provider* and the *Data Processor*, as queries have to be modified in the data processing steps. All colored boxes are DQ components and are added by the framework to the DSMS. The components will be detailed in the following sections.

8.5 Semantic Data Quality Components

The DQ Ontology represents the metadata which is required for the DQ measurement, including DQ dimensions and DQ metrics, linked to concepts in the domain of data stream applications. This allows the evaluation of DQ for tuples, windows, and attributes. We decided to use an ontology for structuring this semantic knowledge. The ontology enables us to define domain knowledge, DQ dimensions, and metrics separately, and relate them to each other in a modular way. In the following, we will first discuss and derive suitable DQ dimensions and metrics. Afterwards, we will detail the structure of the ontology and describe the components of the framework processing the ontology.

8.5.1 DQ Dimensions in DSMS

DQ dimensions have been defined as features of data items that can be measured, and *DQ metrics* are coined as methods performing these measurements (i.e., assigning a DQ value to a data item, see definition 4.6). As it has been emphasized in several DQ methodologies and also in our approach (cf. Section 8.3), the first important step in DQ management is the identification of application requirements and the derivation of relevant DQ dimensions and metrics. There exist already a set of approved and common dimensions which are generally applicable. However, each application may introduce new aspects of DQ which are specific for this application or domain. As Redman (2004) points out, there exist "hundreds of dimensions of data quality, a relatively few dimensions are most important in practice." Hence, it is crucial to determine the ones relevant for the target application. We already listed some of them in the DQ requirements of the corresponding domains in Chapters 6 and 7.

There exists a plethora of classifications which structure and describe DQ dimensions, such as the Total DQ Management (TDQM) classification (Wang and Strong, 1996; Strong et al., 1997), the Redman classification (Redman, 1996), or the Data Warehouse Quality (DWQ) classification (Jarke et al., 1999). The classifications categorize the dimensions according to different aspects. Fan and Geerts (2012) identify in particular timeliness, completeness, and consistency as central issues in data quality.

Klein and Lehner (2009) proposed a classification of DQ dimensions for data streams, which is relevant for our work. In Table 8.1, we list a non-exhaustive set of DQ dimensions, which we think are of importance for DQ rating in a DSMS. In particular, we selected dimensions which we deemed relevant for traffic and health applications (based on (Klein and Lehner, 2009), Table 6.3 and Table 7.2). Other dimensions may be also relevant and our system is not restricted to the selected dimensions. For example, Kuka and Nicklas (2014a) use the Semantic Sensor Network (SSN) ontology[1] which provides quality dimensions specifically for sensor data. Using an extensible ontology, we are able to model an arbitrary set of dimensions and corresponding metrics. We distinguish two categories of dimensions. *Application-based DQ dimensions* rate the application-specific semantics of data. In addition, performance related issues are considered by *system-based DQ dimensions*. For example, Quality of Service (QoS) for various performance aspects of a system has to be tracked and taken into account in query processing. There are also dimensions which are in both categories. Furthermore, the DQ for a dimension can be measured on different levels. It can be measured system-wide, e.g., an output rate, on operator level, e.g., the selectivity of an operator,

[1] `https://www.w3.org/2005/Incubator/ssn/ssnx/ssn`

or on window, tuple, or attribute level.

Table 8.1: Example DQ Dimensions

DQ Dimension	**Informal Description**	**Example Metric**	**Application-based / System-based**
Completeness	Ratio of missing values or tuples to the number of received values/tuples	The number of non-null values divided by all values including null values in a window	Application-based
Data Volume	The number of tuples or values a result is based on, e.g., the number of tuples used to calculate an aggregation	Quantity of tuples in a window	System- and Application-based
Timeliness	The age of a tuple or value	Difference between creation time and current system time	System and Application-based
Accuracy	Indicates the accuracy of the data, e.g., a constant measurement error or an estimation of result quality	An externally calculated or set value, e.g., the result confidence of a data mining algorithm	Application-based
Consistency	Indicates the degree to which a value of an attribute adheres to defined constraints, e.g., if a value lies in certain bounds	Rule evaluation, check of constraints	Application-based
Confidence	Reliability of a value or tuple, e.g., the confidence to have estimated the correct traffic state	A weighted formula that is calculated from values for other DQ dimensions	Application-based
Drop Rate	Indicator for the system performance.	The number of tuples dropped during stream processing due to latencies.	System-based

In the next section, the DQ ontology is proposed in which the identified dimensions can be represented and can be put into the context of DQ management for data streams.

8.5.2 Data Quality Ontology

An ontology is a well suited tool to (1) model the knowledge about DQ concepts as well as about domain concepts and their relationships to each other, (2) modularize the knowledge and make it reusable, (3) be changed by users due to its human-readability and availability of a wide range of tools, and (4) be used as a configuration in an automatic process of evaluating DQ.

Several ontologies have been proposed regarding DQ[2], [3], [4], DQ management cleaning (Preece et al., 2008; Brüggemann and Grüning, 2008), The DQ Management Vocabulary Specification [5], and data streams [6], (Kolozali et al., 2014). But many are not accessible or only sparsely described in papers. And those who are accessible could not fulfill our needs completely. Several concepts of DQ are repeated in multiple ontologies,

[2] `http://bioportal.bioontology.org/ontologies/IDQA`

[3] `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32575`

[4] `http://www.iso.org/iso/catalogue_detail.htm?csnumber=50798`

[5] `http://semwebquality.org/dqm-vocabulary/v1/dqm`

[6] `http://iot.ee.surrey.ac.uk/citypulse/ontologies/sao/sao`

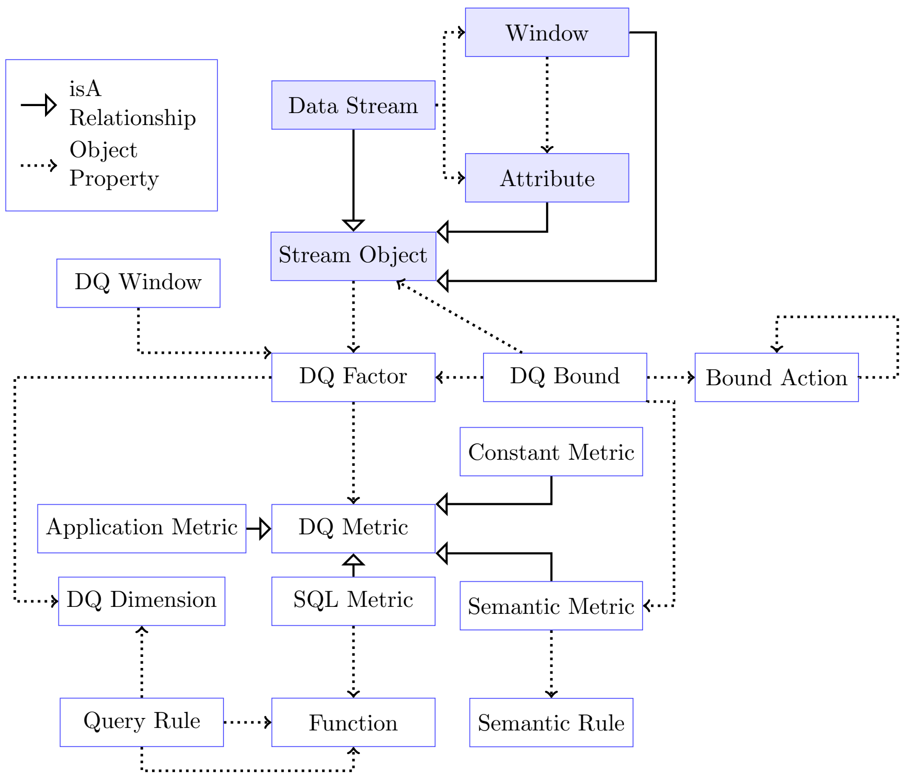


Figure 8.3: Visualization of the DQ Ontology

such as DQ dimensions, metrics, and so on. But some are also unique to the application at hand.

In respect to the overall framework architecture, the DQ Ontology has been modeled to link the components in data stream applications (tuples, windows, attributes) where DQ evaluation is desired with the corresponding quality dimensions and metrics. The DQ Ontology can be also seen as a metamodel for DQ information; it has been inspired by the DWQ metamodel for DQ management in data warehouse systems (Jarke et al., 1999). It differs to the DWQ metamodel in that it includes concepts from the data stream domain and does not include data warehouse related concepts. Furthermore, it is designed along the trisection of the approach. Figure 8.3 delineates the structure of the DQ ontology. Colored concepts describe the scope in data stream applications, e.g., data stream windows or attributes of data stream elements. White concepts illustrate the DQ assessment components including metrics to rewrite SQL queries, evaluate semantic rules, as well as applying user-defined operations to applications. Dashed arrows denote object properties (i.e., relationships) where the arrow points into the direction of the “used” object, e.g., a *DQ Window* has a *DQ Factor*. The solid arrows represent isA-relationships.

The concept of *DQ Factors*, which has been adopted from the DQ evaluation methodology for data warehouses (Jarke et al., 1999), provides the linkage of *Stream Objects*, *DQ Metrics* and *DQ Dimensions* to DQ assessment tasks in the scope of data stream components. The metrics are categorized according to their purpose, way, and type of calculation. A *Semantic Metric* is related to one or multiple semantic rules. A *Semantic Rule* can be expressed by arbitrary mathematical expressions, e.g., Boolean expressions or arithmetic expressions (depending on the power of the mathematical expression parser used to evaluate the expressions during online quality assessment in the system). For example, we can define a Semantic Rule instance, which limits reasonable speed values by an expression $speed < 280$. To characterize intrinsic properties of the data (e.g., according to EU directives we fixed the intrinsic measurement error of a speedometer to 0.9), a *Constant Metric* can be defined whose instance will include constant values for a DQ dimension.

An *Application Metric* provides an interface for user-defined functions and corresponding DQ dimensions. The calculation of an Application Metric is defined in the user-defined code in the DSMS. Finally, the *SQL Metric* accounts for two cases: (1) a DQ dimension is introduced, which is calculated using SQL operators (e.g., completeness), and (2) DQ values change during the processing of SQL queries (e.g., when joins or aggregations are used in the query). Each SQL Metric instance is related to one or more *Function* instances, which can identify either the SQL operator changing the quality value (input) or the SQL expression used to calculate the new DQ dimension value (called output or replacement function). The rewriting of SQL queries with the input functions using the defined output functions is defined by a *Query Rule* instance. These rules link DQ dimensions, SQL operators, and functions to replacement functions. The rewriting of queries will be described in Section 8.6.

The ontology, as it was described so far, is generic, i.e., domain independent – it could be applied to any domain. To implement the DQ requirements of the desired applications, for the modeled concepts instances have to be created. Figure 8.4 outlines an example usage of an SQL metric to assess the completeness of an attribute in a relational tuple of a window. The stream attribute `Speed` is assigned with a DQ factor that links the DQ dimension `Completeness` with an SQL metric. `Completeness` is defined by an expression consisting of arithmetic operators and SQL functions (`count/all`). The variables `count` and `all` are references to functions. Each function has an *SQL Mapping*, which denotes the SQL expressions inserted into the query to calculate the variable value. The placeholder # in the definition of the mapping is replaced by the corresponding attribute name (in this case `Speed`). The expressive power of the SQL mappings is limited to the expressions which are allowed in the SELECT clause of an SQL query; more complex expressions could be realized with an application metric.

Another requirement described in Section 8.1 is the possibility to define bounds for DQ values and corresponding countermeasures if the value is not within the desired bounds. Hence, a concept *DQ Bound* is implemented in the ontology which relates the tested DQ Factor, a Semantic Rule describing how to check the bound, a stream object for which the DQ Factor is defined, and one or more *Bound Actions* to be executed when the bound is violated. For each bound action one or more *Counter Bound Actions* can be defined. These are executed when the DQ value is again in a tolerable range. For each bound an *Incubation Value* is definable, which determines how often the measured DQ value has to go beyond or fall below the bound before taking an action. For example, assume that we monitor the number of elements used to calculate the average speed using a DQ Factor. If there are only a few elements, we activate another data source to increase the accuracy of the average speed value. Hence, as depicted in Figure 8.5 a bound for a DQ factor `Data_Quality_Factor_Volume`, a stream `aggregateAndIntegrate` (representing the aggregated and integrated data) with an incubation time of 5, and a corresponding bound rule are defined. The bound rule checks if more than 50 C2X messages have been used in the current time window, i.e., the rule would be `dq_car2xno_datavolume` ≤ 50. The Bound Action `addMobilePhones` will be invoked, when the rule has been evaluated five consecutive times to `true`. When

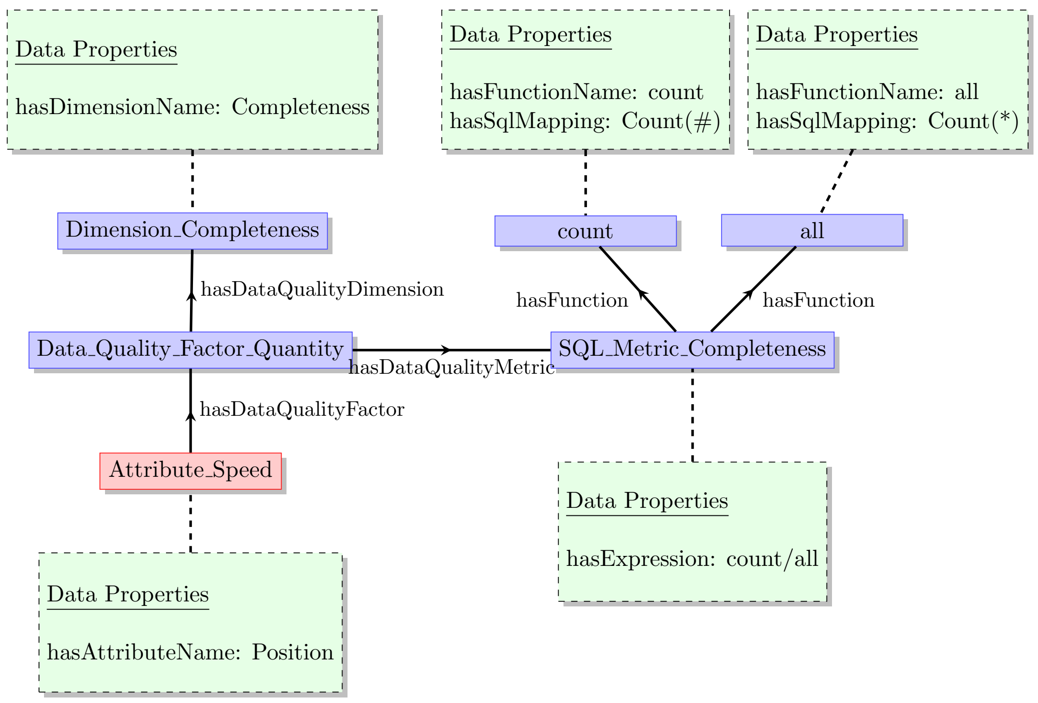


Figure 8.4: Illustration of a SQL DQ Metric in the Ontology

the rule has been evaluated five consecutive times to `false`, the method defined in counter action `removeMobilePhones` is called.

Finally, a flexible solution to create windows only for data quality analysis was demanded in the requirements. Although there is already a concept *Window* in the ontology, a new concept *DQ Window* is introduced for this purpose. The first concept is part of the application logic while the second is only required for DQ analysis purposes. The DQ Window is also connected to the DQ Factor to indicate the parameters of DQ to be measured and to connect to a corresponding stream object. Each instance of this concept has two attributes - size and slide - which determine the size of the DQ window and the slide step (both can be value-based or time-based). In the following, we describe how the ontology is loaded during initialization of the system and provided to the DQ components of the framework. The definition of the attributes, sensors, dimensions, metrics, and rules has to be done only once for an application. Due to the modular design of the ontology it is possible to reuse instances for dimensions, metrics, and so on. The ontology has been implemented in OWL, is available at `http://dbis.rwth-aachen.de/projects/DQStream`, and has been submitted for indexing to Swoogle and Watson. The DQ information is provided by the Metadata Provider and processed by the DQ Processor and corresponding services.

8.6 Data Quality Processing

Subsequent to the *Define* phase in the extended TDQM cycle (cf. Section 8.3) the *Measurement* phase has to be prepared and executed. The ontology contents created in the *Define* phase are utilized to configure the *Measurement* phase, or more specifically speaking, to configure the system for runtime. This means, that for each of the stream components the assigned DQ dimensions and metrics are integrated into the DSMS pro-

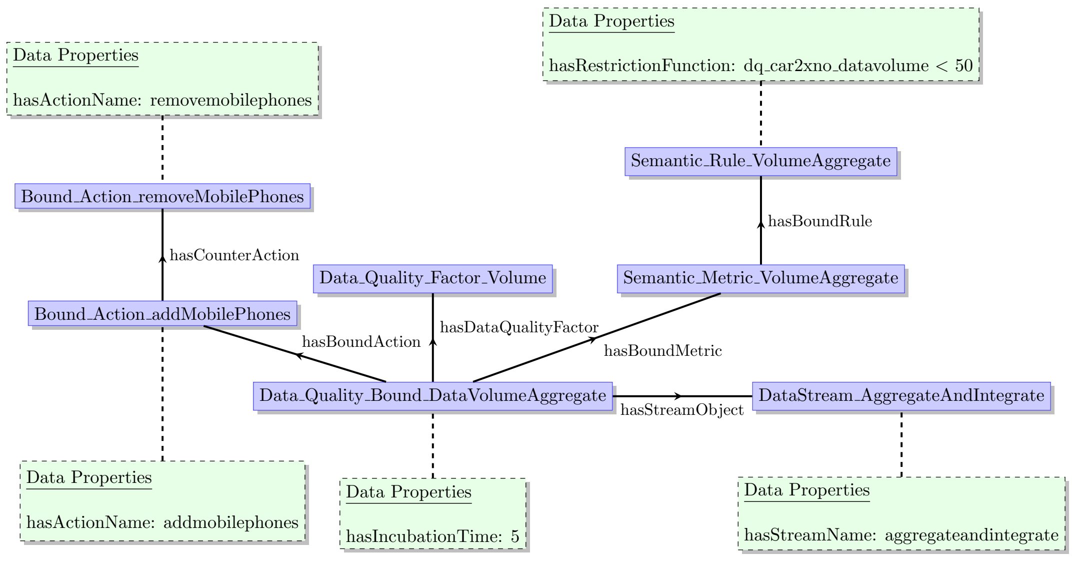

Data Properties
hasActionName: removemobilephones
Bound_Action_removeMobilePhones
hasCounterAction
Bound_Action_addMobilePhones
Data Properties
hasActionName: addmobilephones
Data_Quality_Factor_Volume
hasDataQualityFactor
hasBoundAction
Data_Quality_Bound_DataVolumeAggregate
Data Properties
hasIncubationTime: 5
Data Properties
hasRestrictionFunction: dq_car2xno_datavolume < 50
Semantic_Rule_VolumeAggregate
hasBoundRule
Semantic_Metric_VolumeAggregate
hasBoundMetric
hasStreamObject
DataStream_AggregateAndIntegrate
Data Properties
hasStreamName: aggregateandintegrate

Figure 8.5: Bound Definition for DQ Control in the Ontology

cesses and data flow, such that the DQ measurements can be executed during runtime. The components responsible for the preparation of the measurements are explained in the following. The actual measurements are carried out by the Data Quality Services described in Section 8.6.

As the framework focuses on the optional extension of a relational DSMS with DQ assessment features, all underlying components that contribute to the data processing and are part of the used DSMS are subsumed under the term *Data Processor*. This comprises the data manipulation and execution of queries (rewritten and original queries) in the system. The Data Processor acquires the stream tuples and propagates them to the system. When DQ processing is enabled, it delivers tuples to the DQ Processor, which will calculate DQ values according to the loaded metrics. Corresponding to the three aspects identified for data processing, three different DQ assessment services have been designed. The DQ Services are divided into two groups. First, the *offline* DQ processing is executed in the initialization phase of the system, which performs the rewriting of queries corresponding to the SQL Metrics. Second, the *online* DQ services perform real time evaluation of semantic rules and integrate DQ measurement results from applications.

The *Quality Metadata Provider* represents the interface between the DQ evaluation information (defined in the DQ ontology) and the DQ Processor, which will be outlined subsequently. The DQ Processor loads the ontology into main memory during initialization of the DSMS or when a continuous query is registered or changed. The DQ Processor retrieves the DQ information from the Quality Metadata Provider, and enables the real time DQ processing in the DSMS. The processor analyzes the registered data stream components and looks up the corresponding DQ dimensions and metrics in the Quality Metadata Provider. For each DQ dimension and attribute a field is added to the data stream elements at hand (i.e., the stream is converted to a quality-affine stream). The processor identifies which metric is used for each DQ attribute. It delegates the processing of the Constant Metrics, Semantic Metrics, and Application Metrics to the corresponding service components (Application-based Quality Service and Content-based Quality Service). The SQL Metrics are processed by the

Query-based Quality Service, which directly communicates with the Quality Metadata Provider. The Quality Services are detailed in the subsequent section.

8.6.1 Query-based Quality Service

Since the relational DSMS allows data processing and filtering by relational operators, this also has to be taken into account for DQ measurement. Parsing and analysis of queries allows identification of operators that have influence on DQ values in data processing steps, such as aggregation, join, or selection. Furthermore, new DQ attributes which can be calculated by SQL expressions, can also be inserted by rewriting the initial query. For that reason, in the initialization phase of the DSMS or when registering a new query, the *Query-based Quality Service* rewrites the continuous queries. These queries are executed by the Data Processor of the DSMS and process DQ information in stream elements. By this, we enable the system to handle data streams and quality-affine data streams in the same way. For instance, if the number of Car-2-X messages used to calculate the average speed for a road is relevant for DQ measurement, an SQL Metric `speed_datavolume` would be defined for the attribute speed. Analogue to the example from Figure 8.4, a variable `count` and an SQL Mapping `COUNT(#)` are defined, which denote how the query should be rewritten to calculate the value of an attribute `SpeedDatavolume`. Based on this metric the original query *Q1* (see Listing 8.1) would be rewritten to a query *Q2*.

Listing 8.1: Query rewriting example

Example 8.1 (t)

```
Q1: SELECT RoadID, AVG(Speed)
    FROM message
    GROUP BY RoadID

Q2: SELECT RoadID, AVG(Speed),
    COUNT(Speed) AS SpeedDatavolume_DQ
    FROM message
    GROUP BY RoadID
```

8.6.2 Content-based Quality Service

In contrast to the rewritten queries, this service is executed as data stream elements arrive and when corresponding metrics and dimensions are available. The *Content-based DQ Service* computes DQ values based on data values in the stream elements' fields. These calculations are based on the evaluation of semantic rules of Constant Metrics and Semantic Metrics in the ontology. The rules are expressed by mathematical expressions using the name of attributes as variable identifiers. The total results of the calculations are inserted into the corresponding DQ attributes. The Semantic Metrics also allow defining rules using multiple attributes from one stream element. Equation 8.1 exemplifies a semantic rule that has been defined to identify speed values below or above reasonable bounds.

$$\texttt{SpeedConsistency}_{dq}(\texttt{Speed}) = 0 > \texttt{Speed} \wedge \texttt{Speed} \leq 280 \qquad (8.1)$$

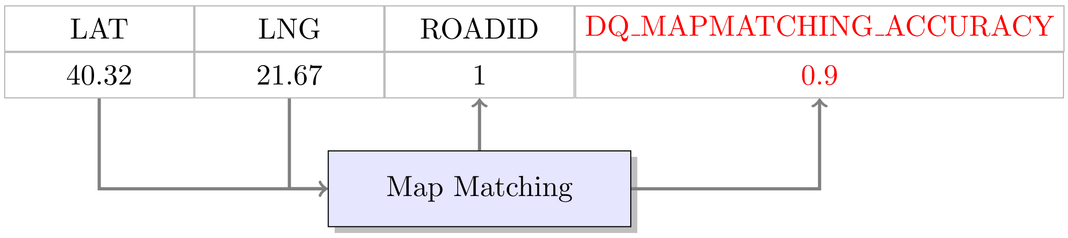


Figure 8.6: Illustration of User-defined Operations

8.6.3 Application-based Quality Service

The *Application-based Quality Service* provides an interface for user-defined functionality to add DQ values to every data stream element. This allows the definition of arbitrary user-defined DQ assessment methods (as long as they can be coded in Java). To take into account the complexity and unknown parameters of user-defined operations, the evaluation of DQ is dedicated to the particular application. For instance, a data mining classification algorithm could provide its accuracy (i.e., the correctly estimated elements compared to all classified elements) as DQ attribute. Figure 8.6 sketches the application of a user-defined operation: A Map Matching algorithm expects GPS coordinates as input and matches these to the corresponding road of a road network. Besides the matching result, the user-defined function also provides quality information according to its computations and configuration, respectively.

8.6.4 Data Quality Monitor

The DQ monitor observes the quality of the processed stream objects, and invokes bound actions and corresponding counteractions as required. Hence, it implements two of the steps in the DQ methodology from Section 8.3, namely the *Analysis* and the *Improvement* phase. In the example from Section 8.5.2, the number of C2X messages available in a time window, represented by the DQ value `dq_car2xno_datavolume` will be monitored. For each stream object, the semantic rules are evaluated in the same way as the semantic rules used in the Content-based Quality Service. The only differences are, that the resulting values are not added to a stream element and that the rules must evaluate to `true` or `false`. If a rule returns `true` the corresponding bound is regarded as violated and a violation counter is increased. The incubation value determines how many consecutive violations are tolerated before an action is taken. When the maximum incubation value is reached, a method (e.g., `addMobilePhones`) is executed whose name has also been defined as bound action in the ontology together with the DQ bound. The method is invoked using reflection in Java. When the rule has not been violated as many consecutive times as the incubation value determines, the specified method for the counter action (e.g., `removeMobilePhones`) is executed in the same way as the bound action.

The framework has been implemented on top of the Java-based DSMS Global Sensor Networks, which has already been used for the implementation of case studies in Chapters 6 and 7. In the following, we will briefly present the implementation of the described components for the GSN system.

8.6.5 Proof of Concept

GSN comprises a flexible architecture: wrapper components are used to abstract from data sources, and virtual sensors are applied to combine data processing and filtering. Based on the configurations the system arranges all virtual sensors in a dependency graph, which determines the data flow of stream elements. We described GSN and its components in detail in Section 6.2.5. To take into account DQ processing, the semantic descriptions for the DQ assessment are loaded from the ontology at system start. Based on this information, the virtual sensor configurations are rewritten recursively, including the queries and the output structure of each virtual sensor in the dependency graph. Aside from the offline DQ assessment that is rewriting the virtual sensor configurations, the evaluation of semantic rules and the interface for user-defined operations has been implemented into the Java base class for all virtual sensors. The extension allows adapting the output structure of stream elements and enables online DQ evaluation of rules by a math expression parser. Likewise, it provides an option to contribute DQ values to every stream element from user code.

8.7 Evaluation

The evaluation of the DQ framework was carried out to test several aspects. First, we aim at a proof of concept in the domain of traffic applications to show the usefulness and adaptability of the system. For this, we used the queue-end detection scenario presented in Section 6.2. Second, it has to be demonstrated, that the framework does not influence the real-time processing capabilities of the DSMS. Therefore, the CPU usage, memory consumption and time complexity of the DQ framework has been examined. Also, the effectiveness should be demonstrated: we varied the source quality of the data and tested, if the changes have been reflected in the derived DQ values of the output streams. Finally, we evaluated the DQ handling in two additional case studies in a second and totally different application field, namely health monitoring, to show the flexibility and adaptability of the framework.

8.7.1 Case Study Queue-end Detection

Queue-ends should be detected with high reliability as otherwise road users will ignore the information if there are too often false positives. As the DQ of the real-time messages may fluctuate due to the issues discussed in Section 8.5.1, e.g., caused by measurement inaccuracies, low data volume, outdated information, missing values, misleading information, or inconsistencies, the reliability of the results of the data processing cannot be ensured at all times and in all aspects. It is desirable to disseminate only those messages to road users that correspond to a high level of DQ. Therefore, an overall confidence value based on multiple DQ values is calculated, which reflects the reliability of the queue-end estimation. Road users can then configure an individual filter such that they receive only queue-end messages above a certain confidence level. We already described the steps of the queue-end detection scenario and their implementation in GSN in detail in Section 6.2. For each of the steps we identify DQ dimensions which influence the overall reliability. The DQ ontology was extended to model the required DQ dimensions and metrics for this scenario. In Figure 8.7, the DQ dimensions for all components are shown. As an example, we describe the DQ dimensions of the `Car-2-X Wrapper` in more detail in the following.

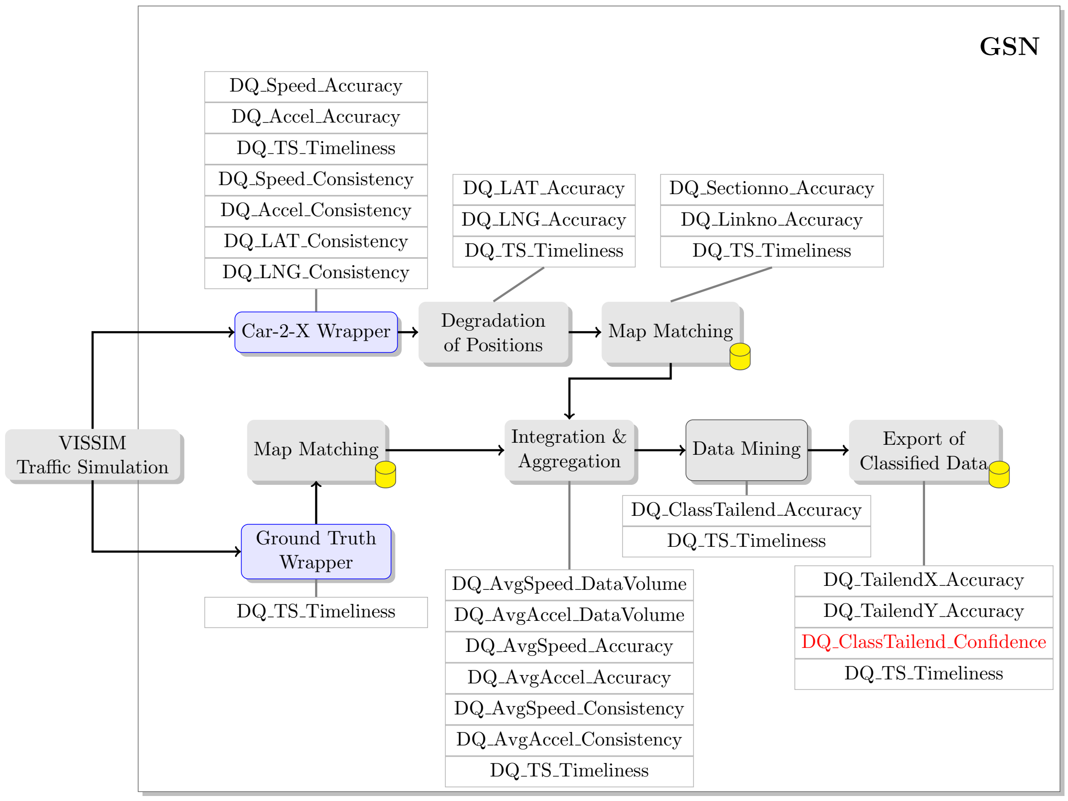


Figure 8.7: Data Flow of the Queue-end Detection Scenario

In real life, the inaccuracies introduced by the speedometer and the acceleration sensor have to be considered which we do in the simulation by introducing a statistical measurement error. The DQ measurement for these values can be based on general statistics or legal requirements (e.g., a speedometer must have an *accuracy* of 90% in the EU). Therefore, we use constant DQ metrics here. An important DQ dimension especially in real-time applications is *timeliness*. A timestamp (`TS`) is attached to each incoming message which will be used to calculate the age of the messages in the system. This is computed by an SQL DQ Metric. The data fields `SPEED`, `ACCEL`, `LNG`, and `LAT` will be assessed with a Semantic DQ Metric that calculates the DQ value using semantic rules for the dimension *consistency*. The rules express the reasonableness and rationality of field values, and model domain-specific integrity constraints. The consistencies for the fields `LNG` and `LAT` are measured according to restrictions of the geospatial boundaries in the road network. The total *confidence* value, which should indicate the reliability of the predicted hazard in the disseminated message, is handled as a common DQ dimension. It is also calculated by a Semantic DQ Metric and rule for which Equation 8.2 over various DQ values is defined in the ontology:

$$\begin{aligned}\text{DQ\_Confidence} = &\frac{1}{16} \cdot (2 \cdot (1 - (1 - e^{-\frac{1}{400} \cdot \text{DQ\_TS\_Timeliness}}))\\ &+ 2 \cdot \text{DQ\_Sectionno\_Accuracy} + 2 \cdot \text{DQ\_Linkno\_Accuracy}\\ &+ \text{DQ\_AvgSpeed\_Accuracy} + \text{DQ\_AvgAccel\_Accuracy}\\ &+ \text{DQ\_AvgSpeed\_Consistency} + \text{DQ\_AvgAccel\_Consistency}\\ &+ (1 - e^{-\frac{1}{5} \cdot \text{DQ\_AvgSpeed\_DataVolume}}) + (1 - e^{-\frac{1}{5} \cdot \text{DQ\_AvgAccel\_DataVolume}})\\ &+ \text{DQ\_TailendX\_Accuracy} + \text{DQ\_TailendY\_Accuracy}\\ &+ 2 \cdot \text{DQ\_Classtailend\_Accuracy}\end{aligned} \tag{8.2}$$

The confidence formula is dependent on the application scenario and domain and can therefore be easily adjusted by changing the corresponding element in the ontology.

Results

After adapting the DQ model to include the DQ dimensions and relationships for the present data stream application, we ran traffic simulations on the artificial road network of 5 km length presented in Section 6.2 and processed the data created by the simulation in the GSN system with the components described above. The volume of vehicles is set to 3000 vehicles per hour. 10% of the vehicles were equipped with V2X communication technology. Each simulation run had duration of 30 minutes. The experiments were executed on a machine with 8GB main memory, a 2.4GHz Intel Core I5 dual core processor, and Windows 7 64bit operating system. The first simulation run is considered as a reference run using the parameters and setup as described before.

Effect of DQ Changes in Derived DQ Information and Confidence Values

To assess how accurate the DQ information and the resulting confidence value reflect changes in the quality of the processed data, we simulated these changes in the data. First, we evaluated the effect of changes in the data volume of the incoming data on data quality factors by varying the sampling rate for the message wrapper. In Figure 8.8 the varied sampling rate (1.0 means 100% elements are forwarded, 0.25 means 75% of the elements are randomly dropped) and its influence on the data volume quality dimension for the average speed of the Integration & Aggregation component are shown. The sampling rate directly influences the data volume as data is dropped and hence less data is forwarded and aggregated, subsequently. Obviously, an average speed value for a road section is more accurate the more data (i.e., messages) is used to calculate it. Hence, the data volume is an important DQ dimension to rate the average speed.

The influence of this DQ factor is also reflected in differences in the varying confidence values (cf. Figure 8.9). As this DQ factor has only a weight of $\frac{1}{16}$ in the overall confidence value, the difference is not very significant.

In a second experiment we compared accuracies of positions and their influence on the overall DQ. We varied the error of positioning for the messages using a normal distribution for GPS localization accuracy with a deviation of 5 meters (which is the case for Floating Car Data, FCD) and cellular network localization using a deviation of 75 meters (which is the case for Floating Phone Data, FPD). This should have an impact on the map matching accuracy and of course also on the confidence values. Figure 8.10 shows the accuracy and confidence values for the two different cases with increasing message amount. It is easily observable, that the accuracy and confidence value of map matched messages using FCD positioning are both higher than the values for messages with FPD positioning. The difference for the map matching accuracy is higher, because again the confidence value is composed of several DQ values and the map matching accuracy influences it only with $\frac{2}{16}$.

System Performance

One crucial requirement for the extension of a real-time data processing system is that the DQ assessment should not, or only slightly, influence the system performance.

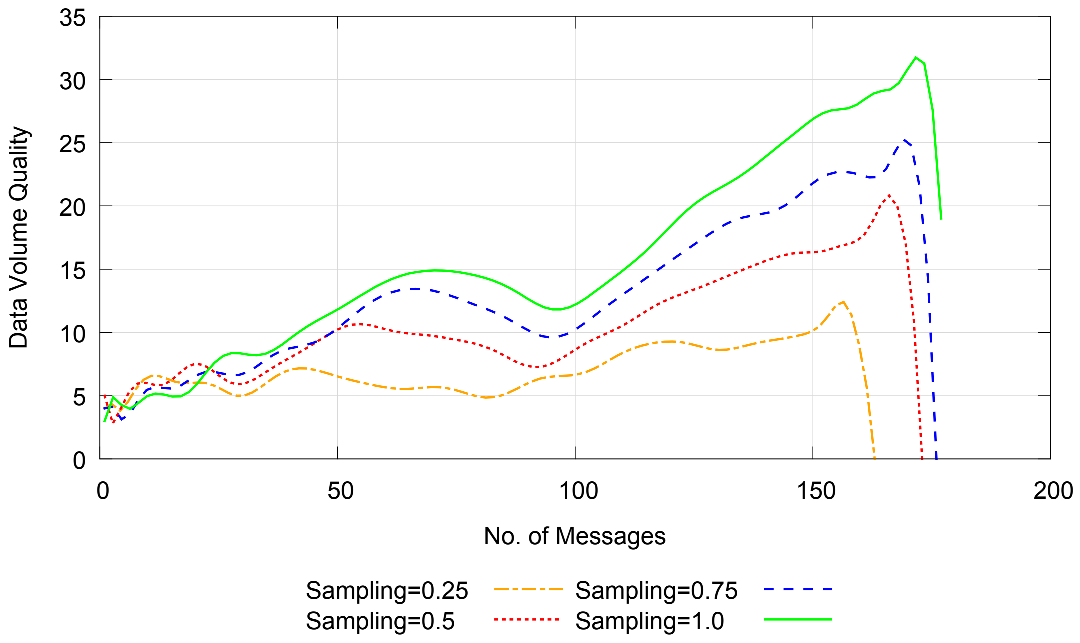


Figure 8.8: Data Volume Quality for Varying Sampling Rates

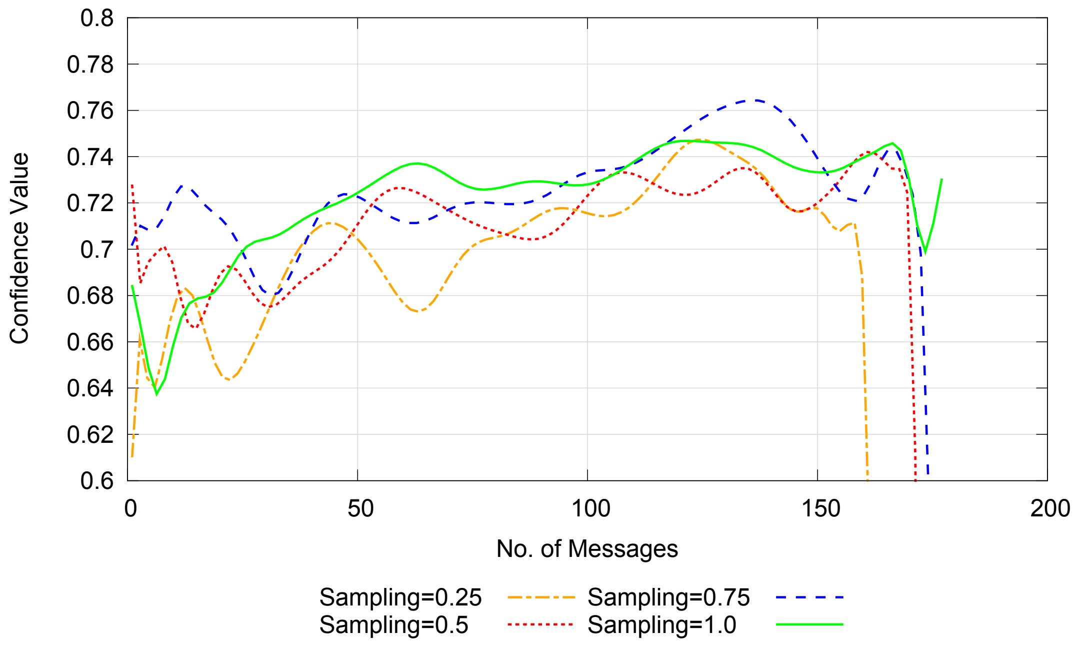


Figure 8.9: Confidence Values for Varying Sampling Rates

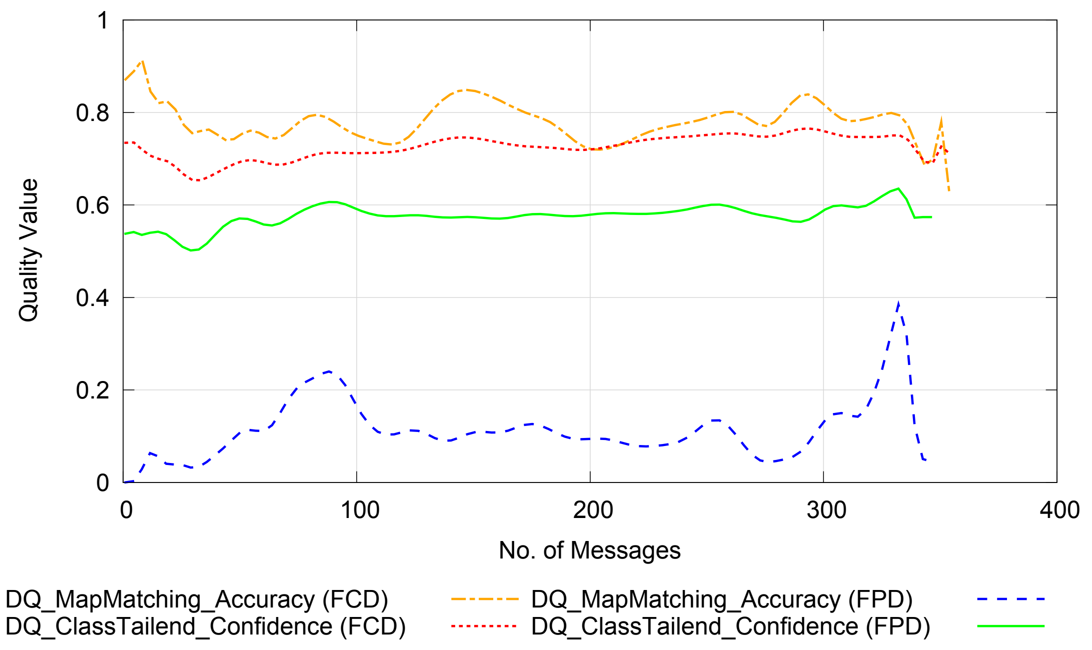


Figure 8.10: Influence of Positioning Accuracy on Map Matching Accuracy and Confidence Value

Therefore, we did several experiments comparing simulation runs with and without DQ assessment.

Figure 8.11 shows the average timeliness for each virtual sensor of the scenario presented in the previous section. The timeliness is the absolute time between the creation of an element and its processing in the sensor. For this experiment we used a window size of 30 seconds and a sliding step of 10 seconds. The diagram shows the fast processing time of elements in the system. The high increase in timeliness can be explained by the windowing (the average timeliness of elements in the window is 15 seconds).

We were also interested whether a DSMS using the DQ framework requires significantly more CPU power or main memory than a DSMS without DQ framework. Figure 8.12 shows, that the CPU usage is only higher during the initialization phase of the system. During initialization, the ontology is loaded and DQ information is processed by a reasoner. Also, the query rewriting is in this phase. After the initialization phase no substantial difference can be noticed.

The same applies also for the memory consumption (cf. Figure 8.13) which is significantly higher during initialization due to loading the ontology. However, at the end of the simulation run the amount of memory is about the same in both situations. We assume that the slow decrease of the memory consumption is caused by the Java VM Garbage Collection.

Effect of Data Quality Monitoring and Control

The monitoring of quality values is important to take counter measures when data quality is insufficient for reliable results. On the other hand, the performance of the system has to be kept up. To see, how effective the DQ monitoring and the corresponding actions can be used, we monitored the number of CoCar messages aggregated

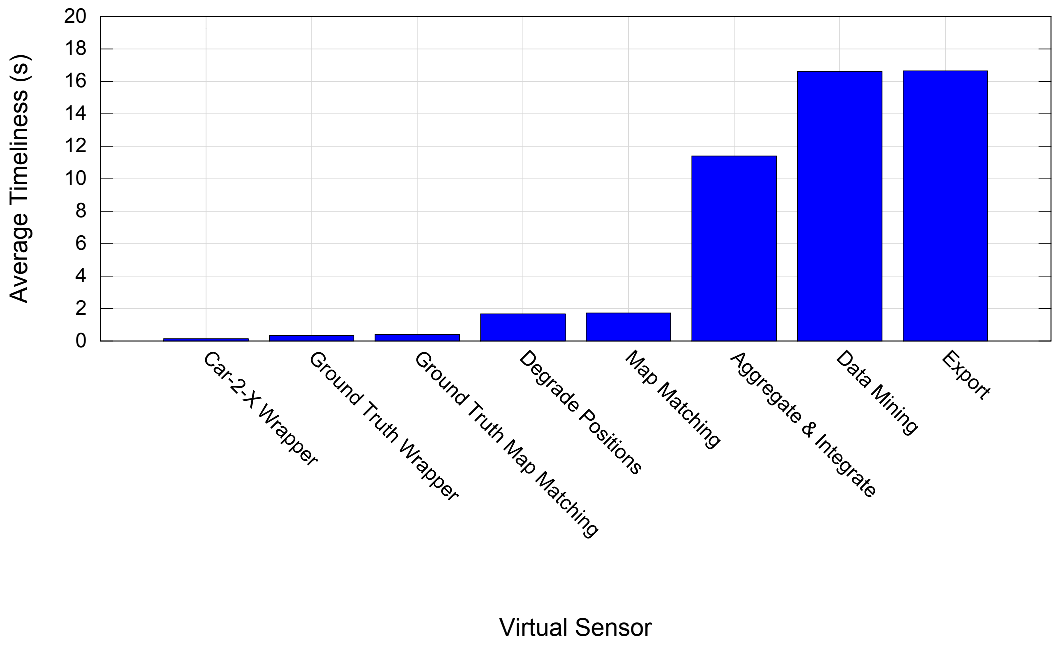


Figure 8.11: Average Timeliness for each Virtual Sensor

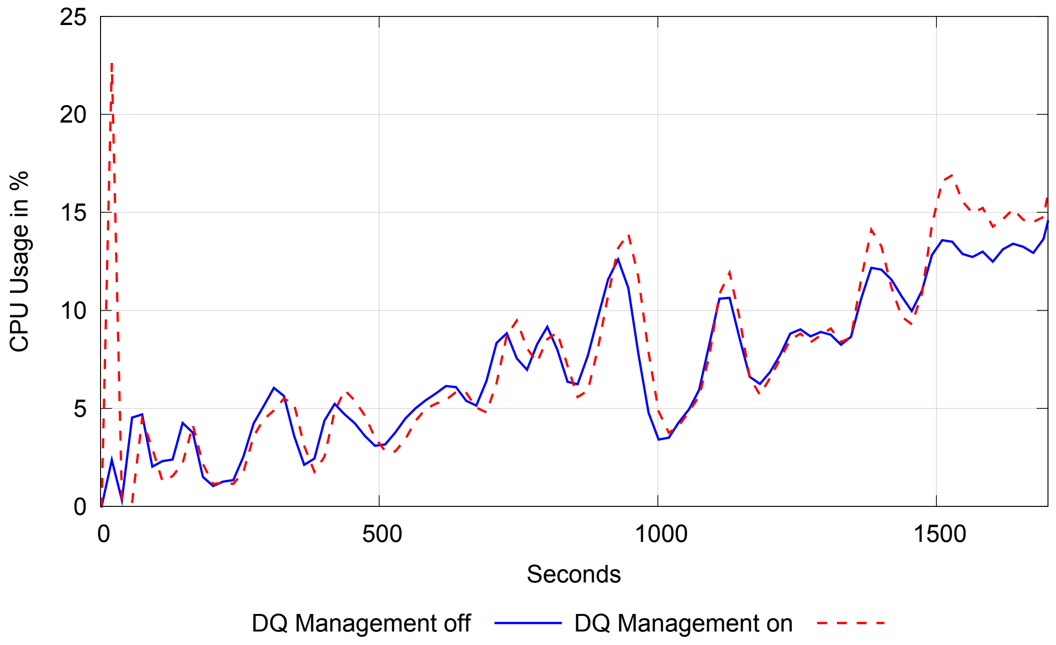


Figure 8.12: Comparison of CPU Loads between Runs with and without DQ Assessment Enabled

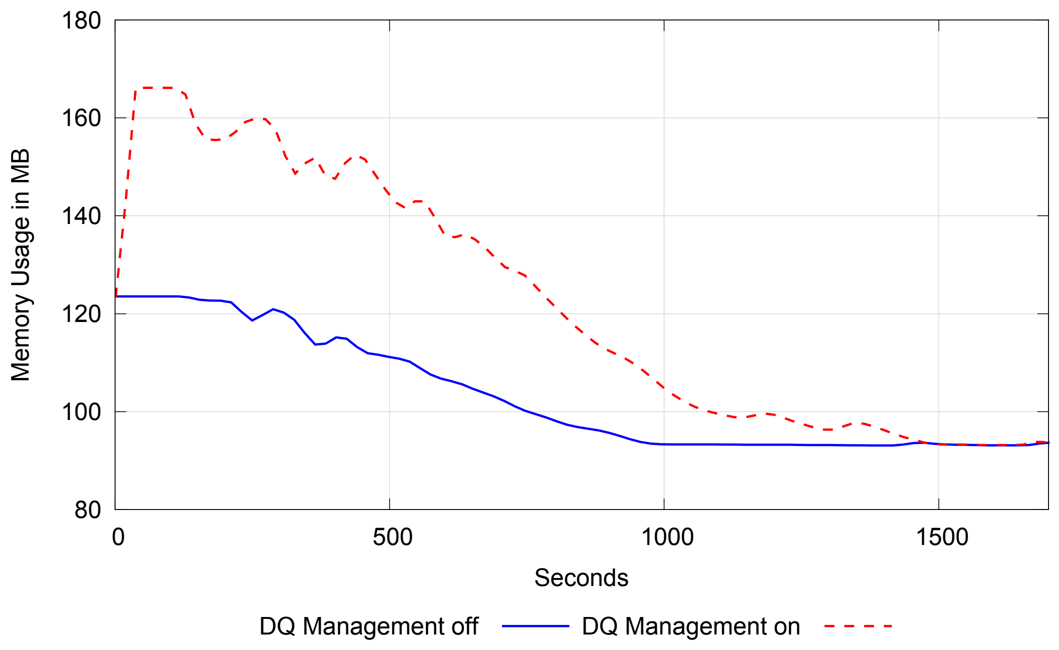


Figure 8.13: Comparison of Memory Usage between Runs with and without DQ Assessment Enabled

to average and count values for data mining in one time window (cf. the example in Section 8.5). This data quality value is important as it reflects how accurate the aggregation results are. The more data is aggregated, the better reality can be approximated and in conclusion a better estimation of the current traffic situation can be made. For this experiment, we varied the penetration rate (meaning the percentage of CoCars in the overall traffic) in the traffic simulation. In the first 1000 seconds of a simulation run 5% CoCars are simulated. After that time, the penetration rate is decreased to 1%, such that after that point in time less CoCar messages are expected.

We defined a DQ Bound with a rule which evaluates to `true` when less than 50 CoCar messages have been observed in a time window. When the DQ Monitor observes that the rule has been "true" for five consecutive times, additionally, data from mobile phone positioning is integrated, aggregated, and fed to the data mining. For the mobile phone positioning we simulated the position of 100 phones in the traffic application simultaneously and calculate the speed of the mobile phones from two consecutive positions of each mobile phone in the DSMS (in detail described in (Geisler et al., 2010)). These speed values and the number of mobile phone observations are also used in the aggregation and mining process. When the number of CoCar messages in the aggregation time window is greater than 50 (i.e., the rule is evaluated to "false") for five consecutive times, mobile phone data is no longer acquired and again only the CoCar messages are used in the aggregation to save performance and communication costs. Figure 8.14 shows that the data mining accuracy (number of true positives and true negatives divided by all classified elements) is slightly higher, when mobile phone data is included in periods, where only a low number of CoCars is available. On the other hand, accuracy is not influenced when enough CoCar messages are available. Mobile phone data is used from the beginning as only few cars are in the network in the first 15 minutes. After approximately 900 seconds, there are enough CoCar messages such that mobile phone data is turned off. After about 1200 seconds, the decreased penetration rate shows an effect and the mobile phone data is used again in the data mining process.

The confidence values for the run with switching mobile phone data on and off are up to 3% lower and more fluctuating than in the run without mobile phones. This can be explained by conflicting data quality value changes. On the one hand data mining accuracy increases when mobile phones are added; on the other hand, other values, such as the accuracy of matched positions and speed, are lower because, mobile phone positioning and the resulting computed speeds have also a lower accuracy.

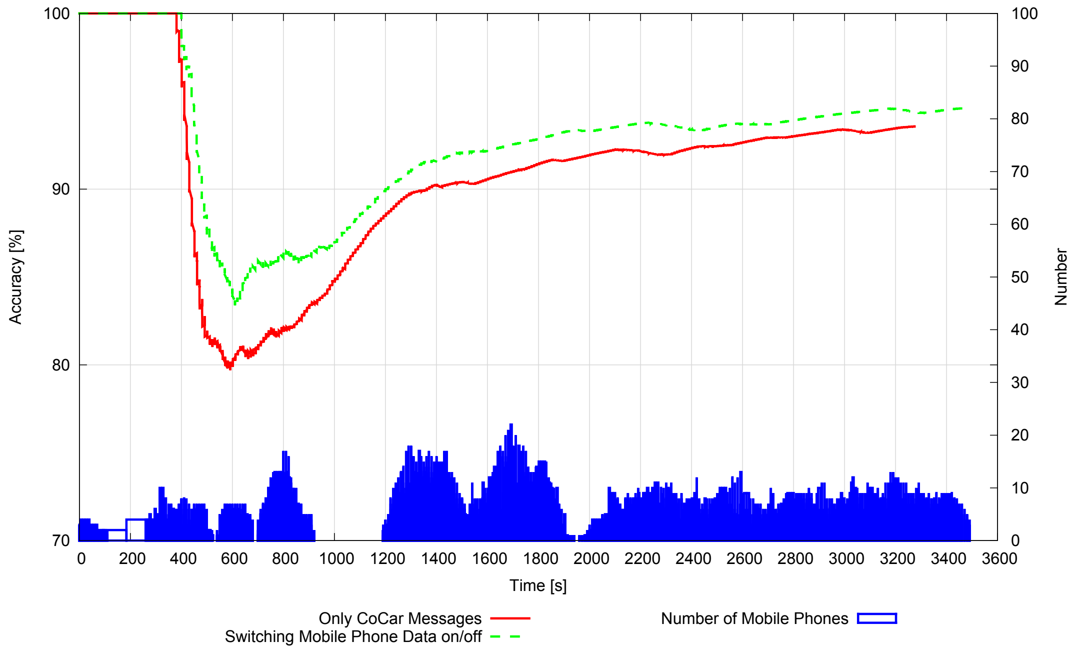


Figure 8.14: Data Mining Accuracy with and without Switching Mobile Phone Data on/off

8.7.2 Case Study Traffic State Estimation

We demonstrated the usefulness and flexibility of our framework in a second case study which utilized the scenario of traffic state estimation (cf. Section 6.3). The traffic simulation was run on a real map (a part of the city of Düsseldorf). We further extended the ontology by additional instances for this second case study. In line with the queue-end detection scenario, an overall confidence value for the information was calculated. The results were visualized in a real-time web application, depicted in Figure 8.15, showing the most important values of the DQ assessment during runs of the traffic simulation.

8.7.3 Case Studies Health Monitoring

We applied the data quality framework to the domain of health monitoring in the context of the HealthNet project (cf. Section 7.1) and the MAS project (cf. Section 7.2). We present the implementation and the results of the evaluation of both in the following.

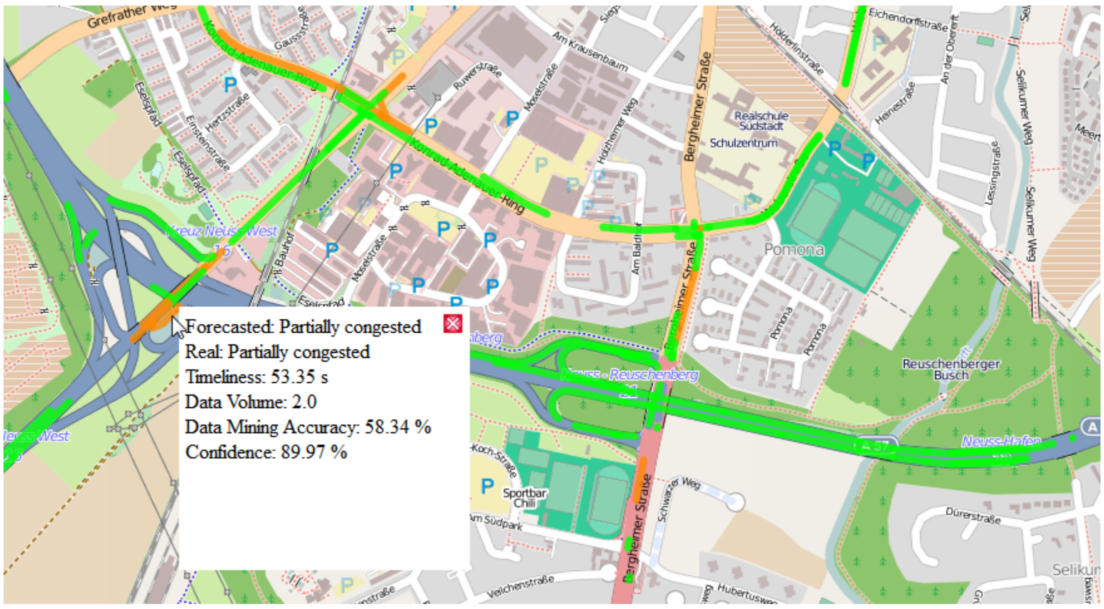

Forecasted: Partially congested
Real: Partially congested
Timeliness: 53.35 s
Data Volume: 2.0
Data Mining Accuracy: 58.34 %
Confidence: 89.97 %

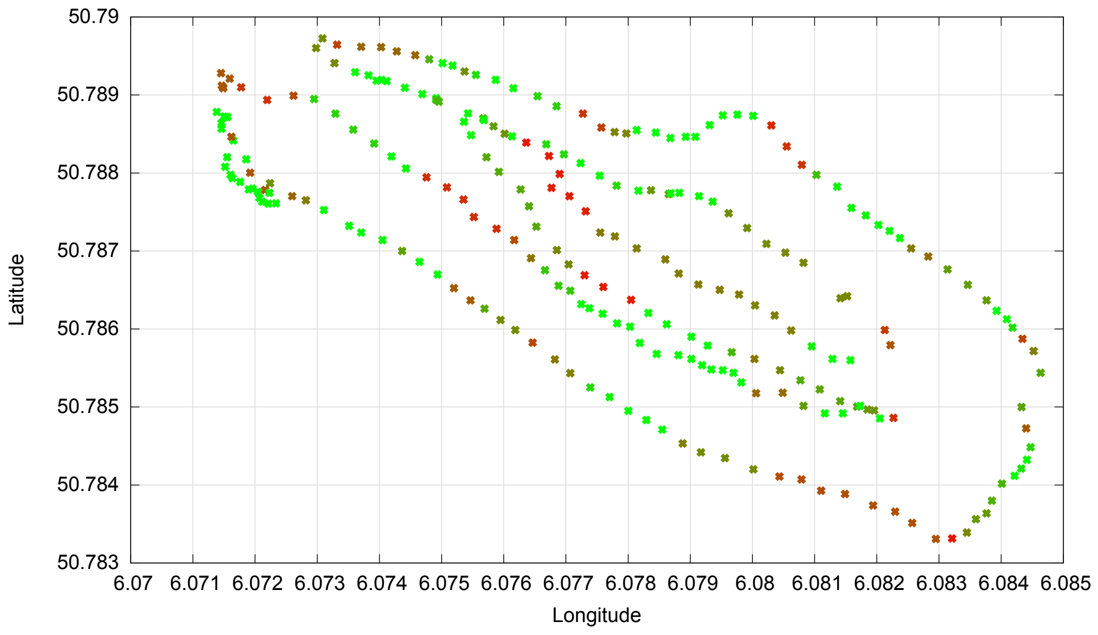
Figure 8.15: Visualization of DQ Information for the Traffic State Estimation Scenario

Evaluation in the HealthNet Project

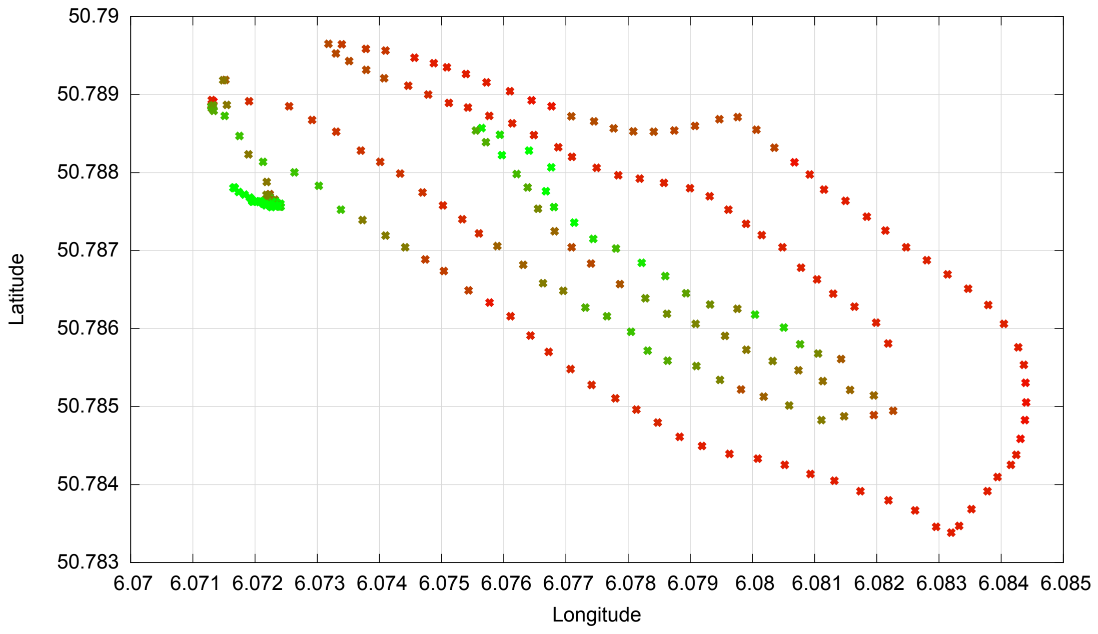
As mentioned in the application development description of this case study, we replayed the data of five runners from the Lousberg run 2011. After the replay we evaluated first the consistency of the heart rate. We plotted the heart rate quality against the position to get a first visual impression of how many values were in acceptable ranges and how many were not. Furthermore, we wanted to know if there are certain sections on the track which cause the quality of the heart rate (due to bad contact of sensors) to drop. The heart rate consistency is represented by a color scale between green (light green represents 100%) and red (light red represents 0%). Figure 8.16 shows the results for two runners.

We can derive several observations from the figures. First, the quality range of the heart rate value differs a lot between the runners. Apparently, the Runner 1 had much better sensor contact than Runner 2. This let us doubt that further investigation into the emergency application based on this data is fruitful the data are not reliable enough to for example train an algorithm to detect an overload situation. Furthermore, from the data we see that nearby the start / finish area the signal was very good for both runners, because they moved only slowly waiting for the start or recovering after the run. Furthermore, the heart rate consistency of both runners also drop when they reach steep bumpy downwards sections on the track. Steep upwards sections do not seem to make a difference. This may be explained by more movement and higher speed when running downhill. Hence, we also analyzed if the heart rate quality is related to the speed of the runner. We took Runner 1 as an example. Figure 8.17 shows the relationship between heart rate consistency and speed.

The comparison of both lines shows that there is some correlation between speed and heart rate consistency. It can be seen that a higher speed results in lower heart rate consistency and the other way around. Especially in the extreme situations (such as standing around beside the track, going steep uphill or downhill) this can be observed.



(a) Runner 1



(b) Runner 2

Figure 8.16: Heart Rate Consistencies of the Two Runners

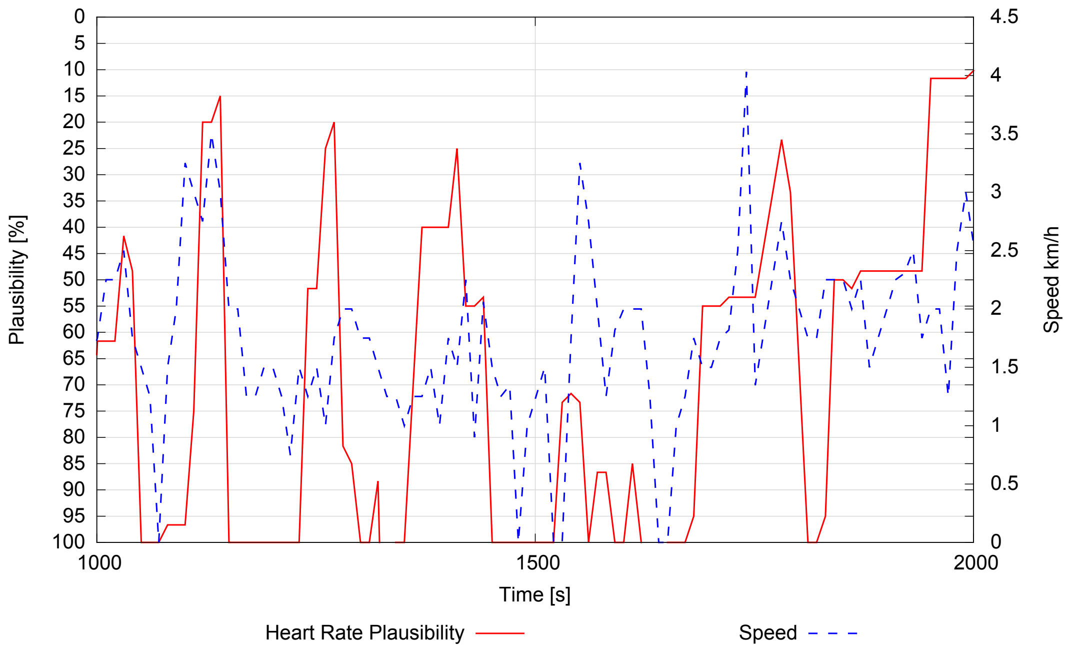


Figure 8.17: Relationship between Heart Rate Plausibility and Speed

8.7.4 Evaluation in the MAS Project

Our approach to estimate the data quality in the MAS project tries to measure a "regularity": it calculates the difference of two consecutive sensor measurements of PPG values. If the average difference over a window of 50 elements is high, this indicates low data quality. This rule can be expressed by the formula already detailed in Equation 7.1 in Chapter 7.

Having a close look at Equation 7.1, it gets apparent, that we need to window the incoming data twice. First, we need a window to get two consecutive PPG values and calculate the difference between them. Second, we need a window of 50 elements for the average calculation. The first sensor, `Calculate PPG Difference`, is used to implement the first windowing of two elements as described before to calculate the PPG difference from two consecutive readings. The second sensor technically just forwards the data and the two DQ dimensions `DQ_PPG_DATAVOLUME` and `DQ_PPG_DIFF_AVERAGE` are introduced here.

In the last sensor, the consistency is calculated as defined in Equation 7.1 and defined in the ontology as a Semantic Metric. The second window of 50 elements required for the final calculation of the consistency is implemented using the feature of defining a window in the ontology for analysis purposes (as described in Section 8.5.2). The instantiation of the windowing in the ontology is depicted in Figure 8.19.

Results

We tested the health monitoring scenario using real world data recorded with the oximeter device over a time period of about 40 seconds. In this time, 2990 PPG measurements have been produced, which results in an average data rate of about 100 elements per second. At the beginning of the measurements the finger with the device was held still and at the end (after about 2250 elements) the finger was moving

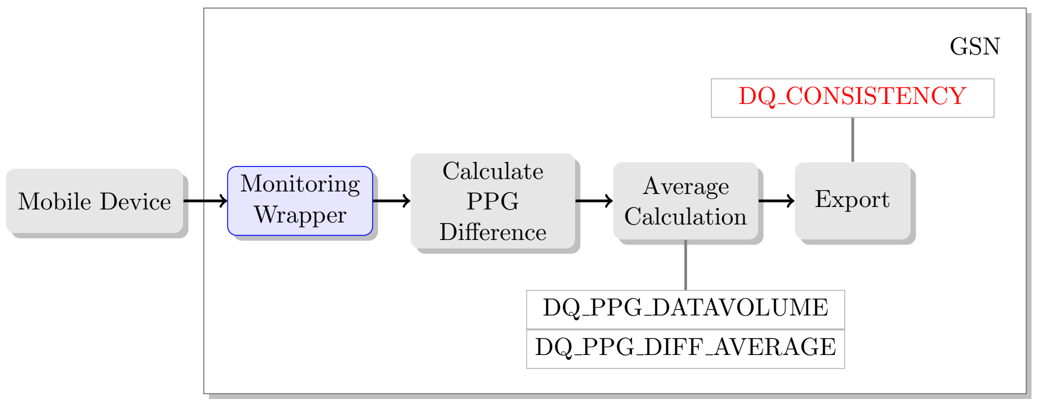


Figure 8.18: Setup of Virtual Sensors and Attached DQ Dimensions

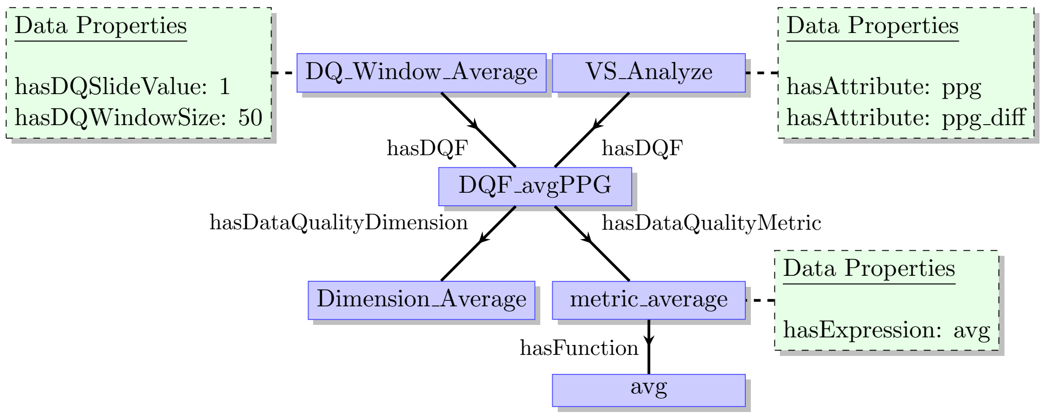


Figure 8.19: Ontology Extended by Instances for Windowing in the Health Monitoring scenario

which should result in bad consistency values as the measurements are no longer as regular as they are expected to. The results of the consistency assessment are shown in Figure 8.20. The evaluation period is extremely short, but this is a great example to show how fast our framework is able to detect decreasing (and likewise increasing) data quality. With only a little marginal delay, which is also attributed to the windowing, the irregularity is detected and the consistency degrades rapidly.

8.8 Conclusion

In this chapter, a flexible, holistic DQ framework for relational DSMSs has been presented. The framework is based on our DQ management methodology for DSMS. Existing methodologies were not applicable to the data stream setting, as they do not consider that DQ assessment, control, and improvement has to be continuous and automated as much as possible. We could show that due to the definition of DQ dimensions, metrics, and mappings to data streams and their attributes in an ontology, the DQ framework is very flexible and easily adaptable to different applications, tasks, and domains. We used the framework to evaluate parameters and their influence on the overall result of the applications and it turned out, that both, simple and complex DQ metrics, can be implemented. The three-fold approach for measuring DQ gives the developer many options for DQ assessment, even custom code can be used to mea-

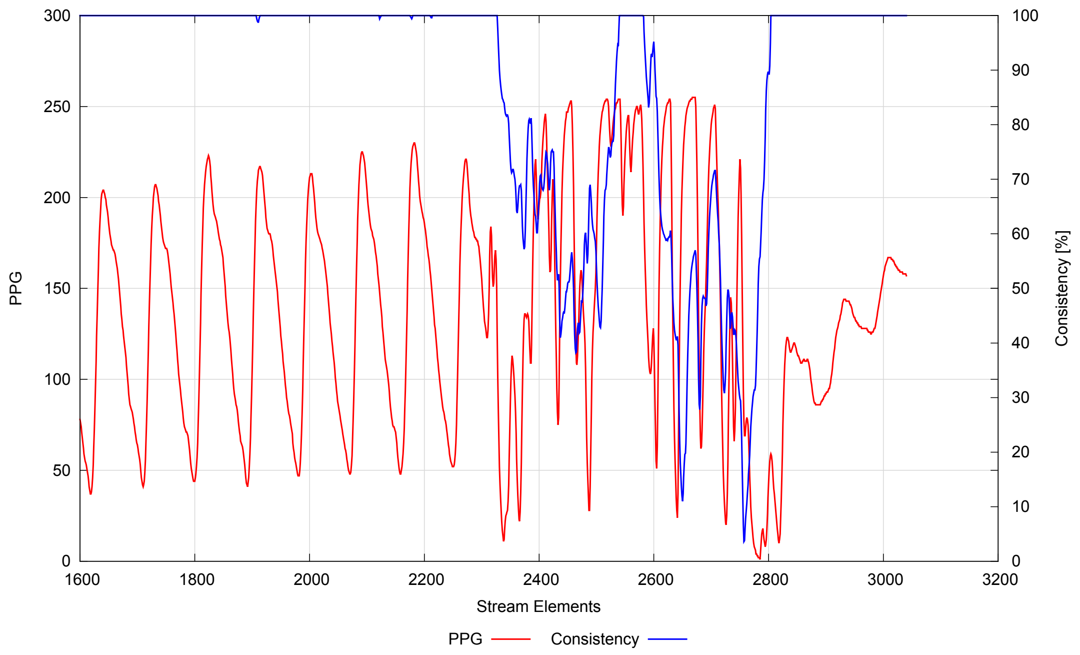


Figure 8.20: Consistency Measurements for the Health Monitoring Case Study

sure quality parameters. Furthermore, we showed that DQ management reduces the performance of the system only marginally, which is very important for DSMSs. The smooth integration with the DSMS by handling DQ values as the other values in the stream element, is beneficial as DQ values can be reused in continuous queries and the DQ management can also be turned on and off as required. However, the implementation of the framework is in some aspects dependent on the underlying DSMS. In particular, the decoupling of the query rewriting component from the specifics of the system and extraction to a modular library could help to proceed towards a general solution. For future work, other query dialects and operators could be considered by using inheritance and plug-ins for the rewriter.

Chapter 9

Real-time Map Matching

In Chapter 5 we introduced an evaluation framework for data stream applications and implemented it for two case studies from the domain of C-ITS in Chapter 6. In this chapter we will use it to evaluate a new map matching algorithm as a problem specific to traffic applications.

Map matching is the basic process to map a measured geographic position to a road network. Amongst others, map matching is crucial for traffic applications based on V2X applications, such as traffic state estimation, or driver assistance systems. V2X communication can enable traffic applications or improve their quality, but it raises privacy concerns and might also increase communication costs. Thus, map matching methods which require frequent position updates or which allow to build a trajectory of the car, are not applicable in this context. In addition, map matching and data analysis have to be done in real-time which means that only a minimal amount of data can be used for map matching. Depending on the application, also a high amount of positions has to be matched in a short period of time which also demands for a very efficient algorithm. In this chapter, we present a new topological map matching algorithm, that incorporates the aforementioned three requirements. These requirements have not been addressed in combination by algorithms so far. The proposed approach needs only two consecutive GPS position fixes and the vehicle heading to estimate a position match; neither a previously matched position or road section nor turn restriction information are required. First, we are building a set of candidate sections using probabilistic map matching. From this set, the segment with the highest total weighted score is chosen. The weighted score is calculated by combining four criteria, taking into account, e.g., the angle between the heading of the vehicle and the candidate section heading.

We evaluate the algorithm in the context of the QED and Traffic State Estimation applications. The evaluation show that the algorithm is robust against changes in position and heading accuracy, fulfils real-time requirements, and that it has an accuracy comparable to existing algorithms. In the following the related works of the approach are presented and compared.

9.1 Related Work

Map matching algorithms can be categorized along multiple aspects. First, *online* or *real-time* and *offline* or *postprocessing* algorithms can be distinguished (Levin et al., 2012; Quddus et al., 2007). Online algorithms map match each position of a vehicle directly after its creation, while offline algorithms utilize and match the complete trajectory of a vehicle at the end of a trip. Furthermore, algorithms can also be categorized according to the degree map information is utilized Quddus et al. (2007). Depending on how many positions of the trajectory of the vehicle are used, local, incremental, and global algorithms can be distinguished (Lou et al., 2009; Goh et al., 2012). *Local* algorithms only use the current position, while *incremental* algorithms utilize a small portion of the trajectory of the vehicle. In contrast, *global* algorithms take the complete trajectory seen so far in consideration. *Geometric MM* only considers the shape of single links (point-to-point, point-to-curve, and curve-to-curve matching). *Topological MM* additionally includes information about the relationship of several links, such as connectivity, adjacency, and orientation, and also previously matched positions or links (Quddus et al., 2007; Velaga et al., 2009). *Probabilistic MM* algorithms(Ochieng et al., 2003; Zhao et al., 2003) define a region (an ellipse, circle, or other shape) around a position fix, based on the error of the measuring device. The region is used to identify candidate sections among which the section with the highest score according to defined criteria is chosen for position matching (Quddus et al., 2007). Furthermore, algorithms can be distinguished in the frequency of processable updates. According to Lou et al. (2009) *low-sampling* algorithms have updates only once in two minutes or less frequent, while high-sampling algorithms have one reading every 10-30 seconds. Moreover, algorithms differ in the additional information required besides position fixes. Some algorithms require the speed of the vehicle, the previous matched position or link, the vehicle heading, or positioning errors. Also the number of required position fixes varies, where most of the algorithms require two fixes. More advanced approaches use more complex techniques such as Kalman Filters, Fuzzy Logic, neural networks, particle filters, or Hidden Markov Models. Many of the proposed algorithms are combinations of multiple MM methods, e.g., topological algorithms using probabilistic MM or geometric MM as initial matching method (limiting the sections or nodes to finally be matched to). As we are targeting online map matching of positions attached to the stream elements, we review some current online algorithms in Table 9.1.

Table 9.1: Categorization of Different Online Algorithms

Work	**Algorithm Type & Used Information**	**Local / Global / Incremental**	**Low / High Sampling Rate**	**Online / Offline**
Lou et al. (2009)	Geometric and Toplogical. Uses temporal and speed constraints, candidate sections, shortest path distance, fixed windowing of the trajectory, and weighted costs	Global	Low	Online, incremental

Table 9.1: Categorization of Different Online Algorithms

Work	**Algorithm Type & Used Information**	**Local / Global / Incremental**	**Low / High Sampling Rate**	**Online / Offline**
Liu et al. (2012)	Based on simplification of maps and road network (use only intersection and start and end points). Use also only two points and weights: use intersection of Euclidean distance radius and topological distance to determine candidate sections for next match. Candidates are ranked by calculating weighted measures using projection distance between second fix and link, angle between the line of the two fixes and the link, traversing distance from start point of the link to the line between the fixes, and traversing angle between the line between the fixes and the line between the start point of the link and the second fix.	Local	High Sampling Rate (up to 30s)	Online
Quddus and Washington (2015)	Topological. Weight-based algorithm using the A* algorithm for shortest path calculation between two points. Weights perpendicular distance, difference between vehicle heading and link direction, distance along the shortest path, and distance along trajectory. It considers link connectivity and turn restrictions at junctions.	Incremental / Global	High (1-60s)	Online

Table 9.1: Categorization of Different Online Algorithms

Work	**Algorithm Type & Used Information**	**Local / Global / Incremental**	**Low / High Sampling Rate**	**Online / Offline**
Hunter et al. (2014)	Topological. Uses a Bayesian probability function for the current vehicle state with prior knowledge about the previous trajectory. For an observed GPS point multiple current candidate states for the vehicle are calculated. For each state a number of candidate paths to reach this state are calculated using the A* algorithm. A probabilistic framework with a discrete filter is used to determine the most probable trajectory.	Global / Incremental	High (1s) & Low (1 min)	Online
Goh et al. (2012)	Utilize an online Hidden Markov model. Use probabilistic weights for GPS coordinates, vehicle speed, speed limit, road width, inferred vehicle heading directions, and topological constraints	Incremental with a variable sliding window.	High (3s) & Low (5 min)	Online

In the following we will discuss pros and cons of the different solution types.

9.1.1 Discussion

For the map matching in a streaming application definitely an incremental online algorithm has to be found as we cannot wait until the complete trajectory of the vehicle has been seen. Hence, offline and global algorithms are not suitable for our case. The sampling can vary a lot depending on e.g., the degree of traffic volume, but we focus more on techniques which can cope with high sampling. Many algorithms use the vehicle heading as an additional information, but the value of this information depends on its source. If the vehicle heading is determined based on the estimated positions, it decreases the accuracy of the final match. The usage of a dead reckoning (DR) system or the GPS heading improves this situation. Also additional topological information such as disallowed turns on intersections proved to be beneficial in reducing the set of possible links to which a vehicle may have driven.

Geometric solutions are quite suitable for online map matching as they do only require one or two position fixes, are easy to implement, and are quite fast. However, they have a very low accuracy as they lack additional topological information. Topological algorithms use more information but may have problems in situations where sampling is low and at intersections. Probabilistic methods have a better accuracy as they consider more information and build up a set of candidate links or sections by using a confidence region. Building this region is costly in terms of time and hence, can only be considered for a subset of the positions to be matched, such as at junctions or at the start (Ochieng et al., 2003; Greenfeld, 2002). The latter algorithms mostly require to know the previous link which is not possible in our scenario. We do not have any prior information about the vehicle. The same is true for many of the more complex algorithms where prior information is required. Hence, we came up with our own simple weighted probabilistic approach which only utilizes two consecutive position fixes and the vehicle heading. We will detail the algorithm and its evaluation in the remainder of this chapter.

9.2 Real-time Map Matching Algorithm using Two Positions

The proposed algorithm consists of four major consecutive steps as depicted in the overview in Figure 9.1. Each step will be detailed in the following.

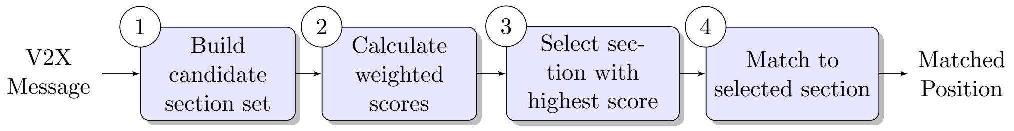

V2X Message
1
Build candidate section set
2
Calculate weighted scores
3
Select section with highest score
4
Match to selected section
Matched Position

Figure 9.1: Map Matching Process

9.2.1 Input

The input of the algorithm is taken from a V2X event message sent by a vehicle. There are various standardized message formats as described in Chapter 2. Usually, these message formats can be adapted to include additional required information. Our algorithm trusts, that the event message includes two consecutive positions P_1 and P_2 (measured using GPS or cellular network positioning) and a heading h_{DR} from a calibrated in-vehicle dead reckoning (DR) system using, for example, a gyroscope and an odometer. The integration of GPS positions and the DR heading, is beneficial as it minimizes the errors inherent to both technologies (Quddus et al., 2003; Ochieng et al., 2003). The integration of GPS positions and DR heading can be done using Extended Kalman Filters (Zhao et al., 2003) and is used today in some high-end in-vehicle navigation systems. In our approach, we also use both pieces of information to complement each other in the score calculation for candidate sections. We match the second of the two consecutive position fixes, P_2, to road network, while P_1 is required in the decision process for the section to match to. One specificity of the input is, that the processed event messages do not carry any information about the identity of the vehicle, i.e., it cannot be determined, which message has been sent by which vehicle.

9.2.2 Select Candidate Sections

Efficient map matching algorithms do not consider all available sections of a road network, but restrict their computationally complex search to an initial and potentially smaller set of *candidate sections* s_c. When there is no trajectory information available, a first set of sections has to be found, one section has to be selected from this, and the position fix has to be matched to this section. This process of matching the initial position is termed *initial map matching* (Quddus et al., 2007; Velaga et al., 2009; Ochieng et al., 2003). Algorithms which are based on the creation of a trajectory, use prior matches and the trajectory for the map matching of subsequent position fixes of the same vehicle. As the provided V2X messages do not include the vehicle's identity, our algorithm does not use the trajectory of a vehicle as an input to the map matching process. Hence, our approach is similar to initial map matching methods used in other approaches. The selection of candidate sections and can be regarded as a projection mm_i over a set of sections s and is defined as follows:

$$mm_i : s \rightarrow s_c \tag{9.1}$$

The selection of candidate sections heavily influences the final map matching accuracy. If the section the vehicle is actually travelling on, is not in the candidate set, then the position will definitely be matched with a greater error (dependent on the size of the sections) and further matching steps will base on this wrong result. In our work, we compare two methods identifying the candidate sections - Closest Point Identification and (CPI) and Error Region Identification (ERI).

Closest Point Identification

One method to select a first set of sections is to first find the closest node (start or end point of a link) calculating the perpendicular distance to the position fix. All road sections that are connected to this node – including the section the node is on – are included in the candidates set s_c (Quddus et al., 2003; Greenfeld, 2002). Though this method is easy to implement and efficient, because only a subset of points in the network have to be considered, it is not very accurate as this method depends very much on the length and shape of the sections. Hence, we adapted the method and calculate the distance between all shape points of the sections and the position fix. Every section which is connected to a shape point A_i is put in the set s_c. This initial map matching method also has several drawbacks. First, it has to go through all the links of the network and calculate for each of the points the distance to the position fix, which might pose a performance problem, depending on the size and shape of the sections. Second, the MM accuracy may vary a lot depending on the accuracy of the network representation. Finally, sections with many shape points will be preferred over sections with fewer shape points by the MM method. An advantage is the easy implementation.

Error Region Identification

To overcome the drawbacks of the CPI method techniques from probabilistic map matching have been proposed for the candidate sections identification (Quddus et al., 2007; Velaga et al., 2009; Ochieng et al., 2003). In this method, an error rectangle, ellipse, or circle, whose size is based on the error of the positioning hardware, is drawn around the position fix P_1. Depending on the approach, every section which intersects with, is tangent to, or is inside this geometrical error shape, will be included in s_c (Velaga et al., 2009; Quddus et al., 2007; Ochieng et al., 2003). In our approach, we use a circular region with a radius of 3-drms (distance root mean square error) as it is expected, that 99% of the actual positions of GPS measurements lie in this radius

(Quddus et al., 2007). In the evaluation of the algorithm in Section 9.4, we assess both candidate section identification techniques as part of the algorithm and compare the results. In the ERI method we use the CPI method as a fallback solution, if no section lies within the error region.

9.2.3 Weighted Score Calculation

The next step in the map matching process is to select the section for which we have the highest confidence to be correct. This confidence value is determined by using a set of four weighted scores: (1) similarity in direction, (2) proximity, (3) segment intersection, and (4) relative position. None of them are sufficient for themselves to determine the best candidate section, but have to be combined. These criteria have been proposed by Quddus et al. (2003) and we will summarize them in the following.

Score for Direction Similarity

A good indicator for a correct match to a section is a high similarity in the direction of the vehicle and the section (Greenfeld, 2002). For this, the heading of the vehicle and the heading of the section have to be determined. As the vehicle heading calculated from two consecutive GPS positions is not reliable enough (Quddus et al., 2007), we use the heading measured by the DR system determined at the second of the two consecutive position fixes. To have a common reference point for both, the vehicle and the link heading, the *azimuth* is calculated for both of them. The azimuth is the angle between the northing and the direction vector of the corresponding object. We calculate the difference between the two azimuths to determine the direction similarity:

$$\Delta\theta = \theta_{DR_i} - \theta_{s_j} \tag{9.2}$$

where θ_{DR_i} is the azimuth of the vehicle direction for the V2X message i and θ_{s_j} is the azimuth of the candidate section s_j. The smaller the angle, the more similar the two directions are. We normalize the difference to angles between $[-180°; 180°]$ as the maximum value for the angle is the complete opposite direction which would be $-180°$ or $180°$, respectively. We use the following normalization function to produce $\Delta\theta'$ (Quddus et al., 2003):

$$\Delta\theta' = \begin{cases} \Delta\theta & -180° \leq \Delta\theta \leq 180° \\ 360° - \Delta\theta & \Delta\theta \geq 180° \\ 360° + \Delta\theta & \Delta\theta \leq -180° \end{cases} \tag{9.3}$$

The score for direction similarity is then calculated as follows:

$$S_{Dir} = W_{Dir} cos(\Delta\theta') \tag{9.4}$$

The cosine is used to give more weight to smaller values, and finally to produce negative values for more than 90° difference (Greenfeld, 2002; Velaga et al., 2009). It produces values in the range of $[-1; 1]$. $W_{Direction}$ is the relative weight given to this score.

Score for Proximity

Another important factor in determining the best candidate section, is the proximity of the position fix to this section. The closer the section to the position, the more likely, that it is the correct candidate section (Greenfeld, 2002; Velaga et al., 2009). Hence, we calculate the perpendicular distance of each candidate section to the position fix, i.e., the smaller the distance, the higher the confidence, that it is the correct section. This is expressed by the following formula for the proximity score:

$$S_{Prox} = \frac{W_{Prox}}{D} \tag{9.5}$$

where D is the perpendicular distance and W_{Prox} is the weight for this score. If the perpendicular line does not intersect with the section curve, the distance to the nearest section node is calculated. During the evaluation we noticed that, if the perpendicular distance from the position fix to the candidate section is very small, the value of the score gets very high in comparison to the other scores. Hence, we use the following cases for D:

$$D = \begin{cases} 1 & D < 1 \\ D & otherwise \end{cases} \tag{9.6}$$

Score for Segment Intersection

In our approach, we only have the two position fixes P_1 and P_2 available. These two positions indicate a kind of mini trajectory of position fixes (not matched positions), which can also be compared to the candidate sections. The intersection of this trajectory with a section curve also leverages the confidence, that it is the correct section (Greenfeld, 2002; Quddus et al., 2003). This confidence is expressed in the size of the the intersection angle. We define the score as follows:

$$S_{Intersec} = W_{Intersec} cos(\Delta\beta) \tag{9.7}$$

where $\Delta\beta$ is the inner (acute) angle between the trajectory curve and the section curve and $W_{Intersec}$ is the weight for this score. The direction of the trajectory curve is not considered, as the heading from the DR system is more accurate.

Score for Relative Position

Finally, Quddus et al. (2003) point out, that the angle between the closest node or shape point of a candidate section and the position fix is also an indicator, if the section is the actual section the vehicle is on (Quddus et al., 2003; Ochieng et al., 2003). The score is calculated as follows:

$$S_{RelPos} = W_{RelPos} cos(\alpha) \tag{9.8}$$

where α is the angle between the position fix and the candidate section curve and W_{RelPos} is the weight of this score.

9.2.4 Section Selection

The total score for each candidate section is calculated by simply summing up the four scores (Quddus et al., 2003):

$$TS = S_{Dir} + S_{Prox} + S_{Intersec} + S_{RelPos} \tag{9.9}$$

Former research showed, that more weight has to be given to the heading compared to the relative position (Quddus et al., 2003; Greenfeld, 2002). In turn, the relative position should be more emphasized than the segment intersection and the proximity criteria. The last two are both considered as proximity scores and hence, get the same weight. Briefly, this relationship of the weights can be summarized with (Quddus et al., 2003):

$$W_{Dir} = aW_{Prox}$$
$$W_{RelPos} = bW_{Prox}$$

where a and $b > 1$ are corresponding weighting factors to balance the relationship. These are dependent on the network and the used sensors and have to be fixed using sample data (Quddus et al., 2003). Finally, the candidate section with the highest score is used to match the position fix P_2 to.

9.2.5 Determine Vehicle Position on Section

After the target section has been determined, the position is matched to this section using simple geometrical matching. This is achieved by doing a perpendicular projection of the position fix to the section. If the perpendicular does not intersect the section, again the closest node of the section is used as the match. This simple matching is accurate and efficient enough for our purposes. There exist more complex matching methods (Quddus et al., 2003), but have not been considered in this approach.

9.3 Implementation & Evaluation Setup

The efficiency and effectiveness of the map matching algorithm is evaluated in the two C-ITS applications, namely QED and TSE (cf. Chapter 6). Map matching is a crucial aspect in the aforementioned scenarios, as we aggregate the data not only over time, but also over location. If a high percentage of positions is matched incorrectly, this may mislead the data stream mining algorithm, as data, which is actually not related, is aggregated to one section. Finally, this would result in a wrong information about the section, e.g., an undetected queue-end or a wrong traffic state for a section. Hence, we also compared the overall accuracy of the applications using the proposed map matching algorithm and using a very simple map matching technique.

9.3.1 Evaluation Setup

As mentioned before, we utilize the existing evaluation framework to assess map matching algorithm in the context of V2X applications. For this purpose, we added new virtual sensors and exchanged the original map matching sensor as depicted in Figure 9.2. The goal is to first determine the ground truth (actual position and section in the network) and compare it later with the estimated position and section.

We will detail the involved components in the following.

9.3.2 Traffic Simulation

V2X messages are produced by the traffic simulation at certain conditions, for example, when a vehicle brakes very hard (acceleration below a certain value). For the assessment of the algorithm, traffic is simulated on two different road networks (one for each

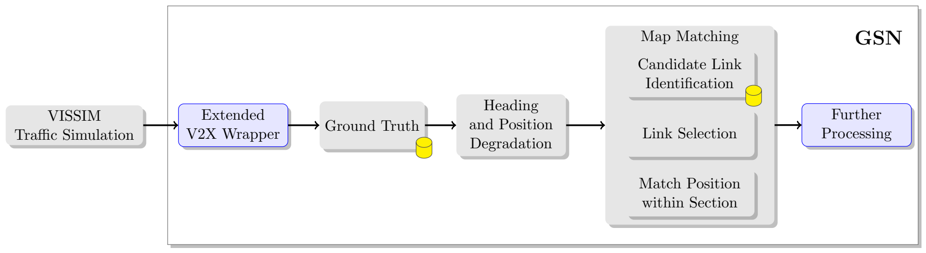

GSN
VISSIM Traffic Simulation
Extended V2X Wrapper
Ground Truth
Heading and Position Degradation
Map Matching
Candidate Link Identification
Link Selection
Match Position within Section
Further Processing

Figure 9.2: Steps for Map Matching in GSN

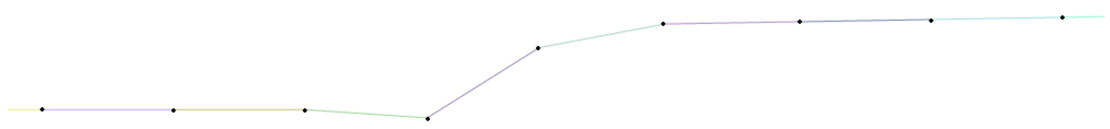
scenario). For the queue-end scenario we use the simple, artificial network consisting of two links in opposite direction and about 5 km long with 100 meter maximum section length, pictured in Figure 6.3. Both links have two lanes, but on a part of one link the lanes are merged to one lane to enforce a queue.

For the traffic state estimation scenario an excerpt of a real map from OpenStreetMap has been imported into the traffic simulation software. The network includes 1394 links, consisting of 7030 sections of 400 meter length max around Düsseldorf. It is depicted in Figure 6.14.

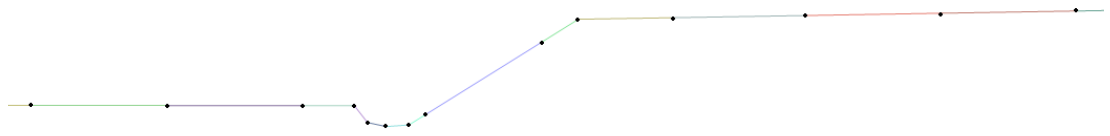
On both networks traffic is simulated which contains cars as well as trucks (12%). The percentage of vehicles equipped with V2X software can be configured and is by default 5%. For each vehicle data, such as current position, speed, and acceleration can be retrieved in each time step. Each simulation run was executed in real-time, i.e., one time step equals 100 ms. As the algorithm requires two consecutive position fixes and the identity of a vehicle is not revealed in an event message, the consecutive positions have to be included in one message. Hence, in the traffic simulation the current and the previous position from the time step before are kept for each vehicle in each time step. The positions are totally accurate, i.e., no positioning error is included and the current position is used as ground truth in the evaluation of the algorithm. The vehicle heading is not available in the traffic simulation. Therefore, we emulate the heading in the DSMS based on the two consecutive positions.

9.3.3 Spatial Database

For the map matching in the DSMS we also need a digitization of a map to match to. As we use a traffic simulation, the road network simulated on is already available in digital form, i.e., links are represented in a 2D graph with nodes and lines. Links in VISSIM are one-directional. Two-directional links have to be represented by two links in opposite directions. To make more fine granular statements about traffic conditions than on a link basis, we divide the links in the spatial database into smaller chunks of road sections of equal size (e.g., 400 meters). For the map matching we implemented a two-level subdivision. While we need equal sized sections for data aggregation (cf. Figure 9.3(a)), for map matching a more fine-granular representation of the section curves is required. On this second level each section is a line between two shape points (cf. Figure 9.3(b)). That means a first-level section of 400 meters with n shape points (including start and end point) is further divided into n-1 sections. Both levels are separately stored in two relations in the spatial database to enhance performance. We will refer to the 1st-level sections in the following as *sections* and to the 2nd-level sections as subsections. The division is done in beforehand not at run-time.



(a) Equal-sized Sections, Stored with Start and End Point Only



(b) Subsections Used for Map Matching

Figure 9.3: Two-level Subdivision into Sections

9.3.4 Ground Truth and Heading Emulation

The V2X messages are received by the DSMS using a wrapper. The data stream elements produced by the wrapper have the following (simplified) structure:

$$\texttt{V2XMessage}(\texttt{Timed}, \texttt{TS}, \texttt{Lat}, \texttt{Lng}, \texttt{PrevLat}, \texttt{PrevLng}, \texttt{Speed}, \texttt{Accel})$$

where `Timed` is the timestamp when the element was created by the DSMS, `TS` is the actual creation timestamp at the traffic simulation, `Lat` is the latitude of the current position fix, `Lng` the corresponding longitude, `PrevLng` and `PrevLat` the longitude and latitude of the previous position, `Speed` and `Accel` are the vehicle's speed and acceleration at that time step.

As the goal is to match the positions to sections and links in the road network to finally aggregate the data related to them, we have to identify the real sections and links the positions have been measured on. We mentioned before, that the positions included in the V2X messages are exact as they have been directly retrieved from the traffic simulation. Hence, we can use a simple perpendicular projection in the DSMS to identify the correct section and link for both positions in each message in the spatial database.

Furthermore, we have to emulate the heading which in reality would be measured by an in-vehicle DR system. There are two possible ways, given the data we have. (1) With the two exact consecutive positions P_1 and P_2 at hand, the vector of the line between them can be used to calculate the heading. (2) When we know the subsection the current position P_2 is located on, we can use the heading of this subsection from the spatial database. For the same reasons as the DR heading was considered to complement GPS positions (Zhao et al., 2003; Quddus et al., 2003) the second option is preferred over option (1). For example, when the two positions do not lie on the same subsection, the positions' vector depicted as blue arrow in Figure 9.4 does not represent the heading correctly, while the subsection vector (red arrow) is what is desired.

After retrieving the corresponding subsection curve from the database the azimuth is calculated, i.e., the angle between the north direction and the vector. This converts the heading vector $\vec{AB}$ to a scalar, where $A(x_1, y_1)$ is the start and $B(y_1, y_2)$ is the end point of the subsection. The scalar makes the degradation, required in a later step, easier. The azimuth is calculated as follows:

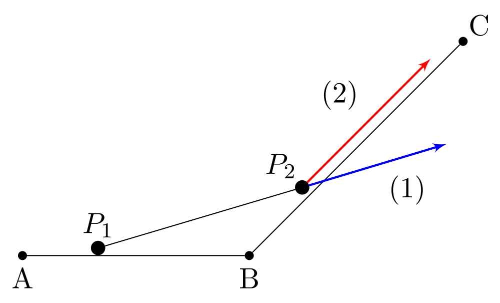


Figure 9.4: Determination of Vehicle Heading

$$Az_{\vec{AB}} = \tan^{-1} \frac{x_2 - x_1}{y_2 - y_1} \tag{9.10}$$

To normalize the azimuth to a value range of $[-180°; 180°]$ as the worst case is the opposite direction, the following rules are applied:

$$Az_{total} = \begin{cases} Az_{\vec{AB}} + 180° & (y_2 - y_1) < 0 \\ Az_{\vec{AB}} + 360° & Az < 0 \end{cases} \tag{9.11}$$

Finally, the ground truth values for the two consecutive positions, the link, section, and subsection, the message has been created on, and the vehicle heading have been assembled. This information is added as attributes to the produced stream element resulting in:

$$\texttt{V2XMessage}(\texttt{Timed}, \texttt{TS}, \texttt{Lat}, \texttt{Lng}, \texttt{PrevLat}, \texttt{PrevLng},$$
$$\texttt{Speed}, \texttt{Accel}, \texttt{LinkID}, \texttt{SubsectionID}, \texttt{Heading})$$

9.3.5 Degradation of Positions and Heading

In a simulation study it is of great importance, that reality is approximated as close as possible to make achieved results useful. Hence, errors inherent to measurement devices, such as GPS receivers and DR systems have to be integrated into the so far exact data. This is also a big advantage in simulation studies. Errors can be varied, i.e., distributions with different types, means, and deviations can be tested and the influence on the algorithm can be assessed. For this purpose we implemented a degradation virtual sensor which is retrieves stream elements from the ground truth sensor in the DSMS.

In a DR system, devices can be error-prone due to several factors, e.g., environmental influences or scale factor errors (Ochieng et al., 2003). Because there are many factors involved, an appropriate way to model these heading is a zero-mean Gaussian distribution. The standard deviation can be configured for the virtual sensor in the XML file, such that this can be varied in the experiments.

For the positions GPS accuracy was assumed by default. This means the error for both, latitude and longitude, have to be modeled as independent Gaussian distributions. We assume a maximum error of 7.8 meters which includes 95% of GPS measurements and results in a standard deviation value of 2.81 for both deviations. This can also be configured and varied in the virtual sensor. This enables us to also experiment with other positioning techniques such as positioning in cellular networks. In general, we modeled the errors for the DR system and GPS system independently to open up a greater variety of configuration possibilities.

After degradation the degraded values are also added as new attributes to the stream element produced by the virtual sensor. The produced stream elements include all information required for map matching itself and its assessment, i.e., degraded positions and heading and the corresponding ground truth data.

$\texttt{V2XMessage}(\texttt{Timed}, \texttt{TS}, \texttt{Lat}, \texttt{Lng}, \texttt{PrevLat}, \texttt{PrevLng}, \texttt{Speed}, \texttt{Accel}, \texttt{LinkID}, \texttt{SubsectionID}, \texttt{Heading}, \texttt{DegHeading}, \texttt{DegLat}, \texttt{DegLng}, \texttt{DegPrevLat}, \texttt{DegPrevLng})$

9.3.6 Map Matching

The stream elements of the degradation sensor are consumed by the virtual sensor executing the actual map matching for both points P_1 and P_2. Note, that only the wrapper and this sensor are necessary when we use real world data. The other sensors are just for the assessment of the map matching performance. In the map matching sensor the three stages of the MM algorithm as described in Section 9.2 are implemented. For candidate selection, both methods, closest point identification (CPI) and error region identification (ERI) have been implemented. For the CPI method a spatial query is executed, which gets the start or end point of all subsections with the shortest perpendicular distance to the point at hand. All subsections which include this point are put in the candidate section set. For the ERI method, also a query on the spatial database is formulated. In the query, a circular buffer of specified size is built around the point at hand and retrieves all subsections which intersect with the buffer. If no subsection intersects, the CPI method is used as fall back solution. The weighting scores to rate the candidate subsections have been implemented according to the principles in Section 9.2 in Java code. We briefly summarize the specifics of their implementation here. Each of the scores is calculated for each candidate subsection and the current point P_2.

- S_{Dir}: The vehicle heading is already included in the V2X message. The headings of each candidate subsection are calculated the same way. Finally, for each subsection the difference between vehicle and subsection heading is calculated.
- S_{Prox}: The perpendicular distance from position fix P_2 to each subsection is calculated using a query to the spatial database. If the perpendicular does not fall onto the subsection, the distance to start and end point are calculated and the smallest chosen.
- $S_{Intersec}$: If a subsection and the segment between P_1 and P_2 do not intersect, the score is 0.
- S_{RelPos}: The relative position of P_2 to the subsection at hand is calculated by determining the closest end point A of the subsection to the current position and calculating the azimuth of the segment between P_2 and A. The difference of the azimuth of the subsection heading and this azimuth is the result for this score. But as we want to have as less accesses to the spatial database as possible, an alternative calculation was made using the cosine law. In principle, we can assume, that the three points A, B, and P_2, where A and B are the end points of the subsection, build a triangle (see Figure 9.5).

The length of each side of the triangle can be easily determined based on the Pythagorean theorem as all coordinates of the points are known. Then the cosine law can be applied

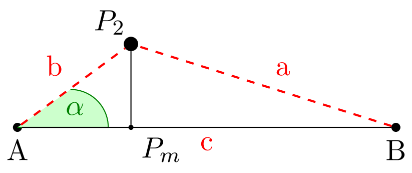


Figure 9.5: Calculating the Relative Position Using the Cosine Law

$$a^2 = b^2 + c^2 - 2bc \cdot \cos\alpha \tag{9.12}$$

and the acute angle is determined by:

$$\alpha = \arccos(\frac{b^2 + c^2 - a^2}{2bc}) \tag{9.13}$$

Determine Position on Section

Once the best candidate subsection has been identified, the position has to be matched to a point P_m on this section. As mentioned before, a perpendicular projection is applied. To spare access to the spatial database, we use the coordinates A and B of the subsection and the position P_2 to calculate the relative position r of P_m as follows:

$$r = \frac{(y_A - y_{P_2})(y_A - y_B) - (x_A - x_{P_2})(x_B - x_A)}{(x_B - x_A)^2 + (y_B - y_A)^2} \tag{9.14}$$

with the following meanings for the values of r:

$$r \begin{cases} = 0 & \text{then } P_m = A \\ = 1 & \text{then } P_m = B \\ < 0 & \text{then } P_m \text{ on backward extension of AB} \\ > 1 & \text{then } P_m \text{ on forward extension of AB} \\ > 0 \wedge < 1 & \text{then } P_m \text{ lies between A and B} \end{cases} \tag{9.15}$$

Finally, the coordinates of the matched point are determined as follows:

$$x_{P_m} = x_A + r(x_B - x_A)$$
$$y_{P_m} = y_A + r(y_B - y_A)$$

The coordinates of the matched point are also added as attribute to the stream element.

9.4 Evaluation

We evaluate the performance and accuracy of the implemented algorithm in the following. There are different parameters which can influence the algorithm's performance and which we will vary in the evaluation, including:

- Weighting schema
- Candidate section identification method

- Road network
- Heading error standard deviation
- Position error standard deviation
- Maximum section length

To rate the accuracy of the algorithms we define two metrics.

1. **Correct Section / Link Identification:** The ratio of correctly identified sections or links to the overall number of V2X messages.
2. **Horizontal Accuracy:** Distance between the ground truth position of the vehicle and the matched position in meters, where average, maximum, and minimum horizontal error are distinguished.

9.4.1 Results

The experiments have been carried out with a default set of parameters for the aforementioned factors. We used as simulation software VISSIM and the Düsseldorf network as default road network. It comprises 1394 links and 7030 sections with a maximum length of 400 meters. In the Düsseldorf network we do not distinguish between sections and link accuracy as the number of links and sections is the same for 400m section length. The traffic simulation is executed always in the same way with 10000 V2X messages sent. As the map matching accuracy always stabilized after 2000 messages, only the first 2000 messages are included in the evaluation. The heading degradation has a default value of 30°.

Weighting Schema

The weighting schema defines values for all four scores in the algorithm. The evaluation aims at finding an optimal combination of weights. Findings in prior research showed that heading and relative positions are more important than distance or intersection scores (Quddus et al., 2003). As distance and intersection scores are both proximity scores, they get the same weight. This results in the following schema, where $a, b > 1$ are the weighting factors:

$$\begin{aligned} W_{Head} &= a \cdot W_{Prox} \\ W_{RelPos} &= b \cdot W_{Prox} \end{aligned}$$

a represents the importance of the heading score S_{head} when compared to the proximity scores S_Dist and S_Intsec. b represents the importance for the relative position score S_{RelPos}. We start with base weights $a = 5$ and $b = 2$ based on (Quddus et al., 2003).

Ratio of Weights

We first try to find a proper ratio of a to b, to see if the heading score or the relative position score should have a higher weight. The results with different combinations of a and b values are shown in Figure 9.6 where the heading degradation is 30 degrees and Figure 9.7 where the heading degradation is 50 degrees. Both are carried out on the Düsseldorf road network.

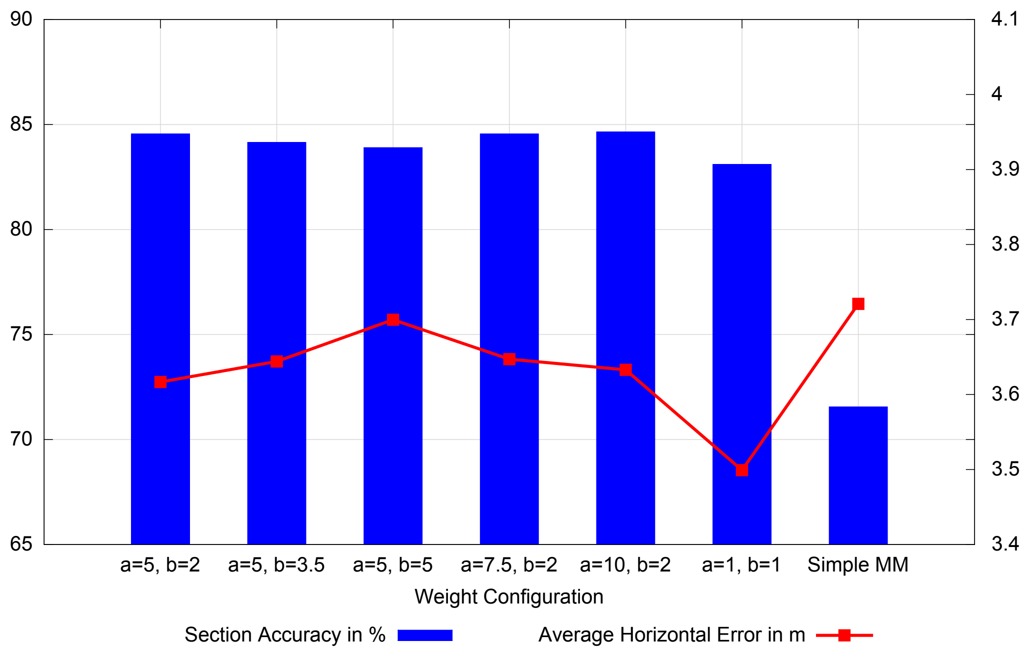


Figure 9.6: Ratio of a to b with 30° Heading Degradation on the Düsseldorf Network

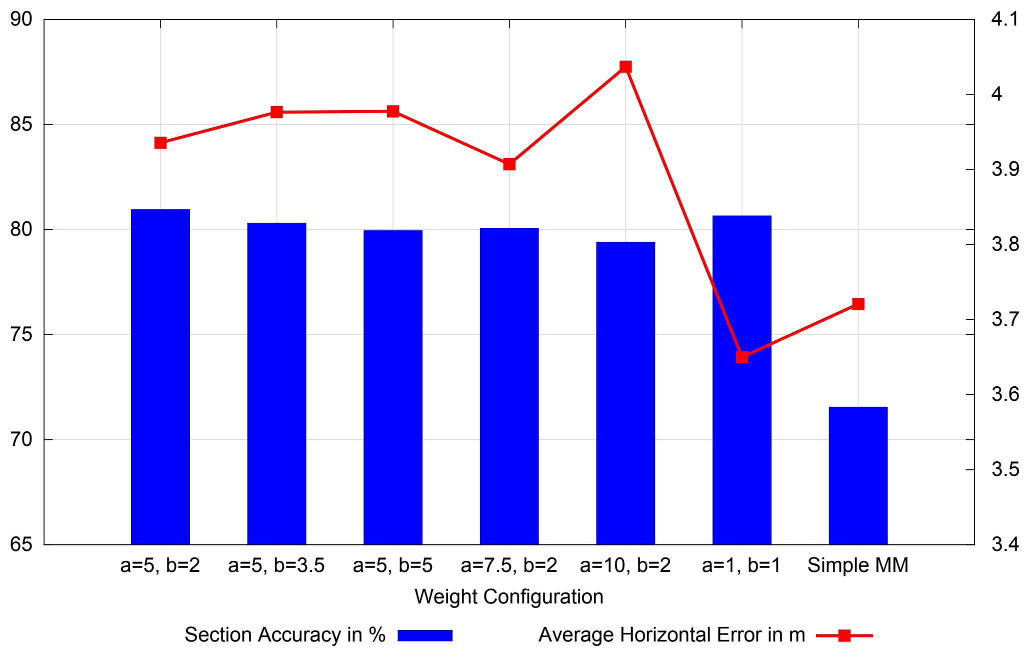


Figure 9.7: Ratio of a to b with 50° Heading Degradation on the Düsseldorf Network

We start with the recommended weights and then increase a and b separately by 1.5 in each step, i.e., the other weight is kept constant and vice versa. Finally, we test how accuracy and error change, when both weights are 1 (i.e., no weights). We also always compare with the simple map matching method (i.e., smallest perpendicular distance to the next link). It can be noticed from the results that the section accuracy does not vary much when the weight ratio is varied. For both heading degradation values, the difference is just 1.55% which is 31 sections from 2000. The increase of b with constant a results in a decrease of the section accuracy and an increase of the average horizontal error. The increase of a with constant b has only a minor effect on the section accuracy and causes an increase in the average horizontal error. Based on these results, the ratio between a and b is optimal for $a = 5$ and $b = 2$. Compared with the simple map matching algorithm the new algorithm provides a significant improvement.

Magnitudes of Weights

After confirmation of the optimal ratio between the weights, we try to find the optimal magnitudes for the weights. In Figure 9.8 and Figure 9.9 the results of the experiments with different magnitudes for a and b are presented.

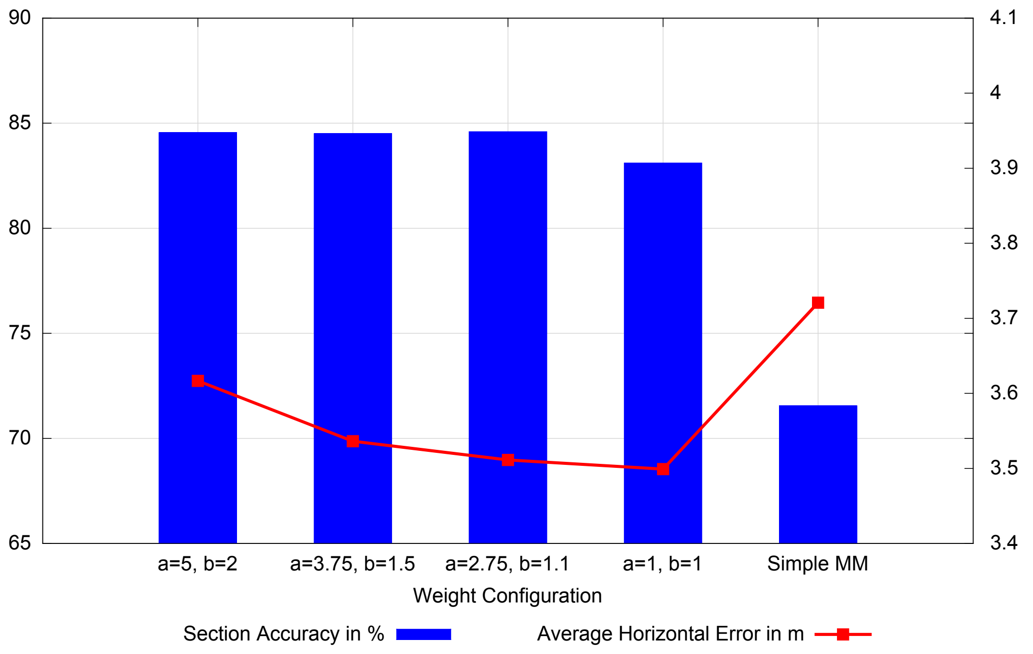


Figure 9.8: Variation of Weight Magnitudes for 30° Degradation and the Düsseldorf Network

Combinations of $a = 3.75$, $b = 1.5$, and $a = 2.75$, $b = 1.1$ with a fixed ratio of $\frac{a}{b} = \frac{5}{2}$ are tested and compared with original weights combination and the simple map matching algorithm.

For 30° heading degradation the section accuracy does not really change, while the average horizontal error decreases with decreasing values. For 30° heading degradation the section accuracy and horizontal error fluctuate and no clear trend is visible. For both heading degradation values, the combination of $a = 3.75$ and $b = 1.5$ provide the best results. Hence, this combination will be used in subsequent experiments.

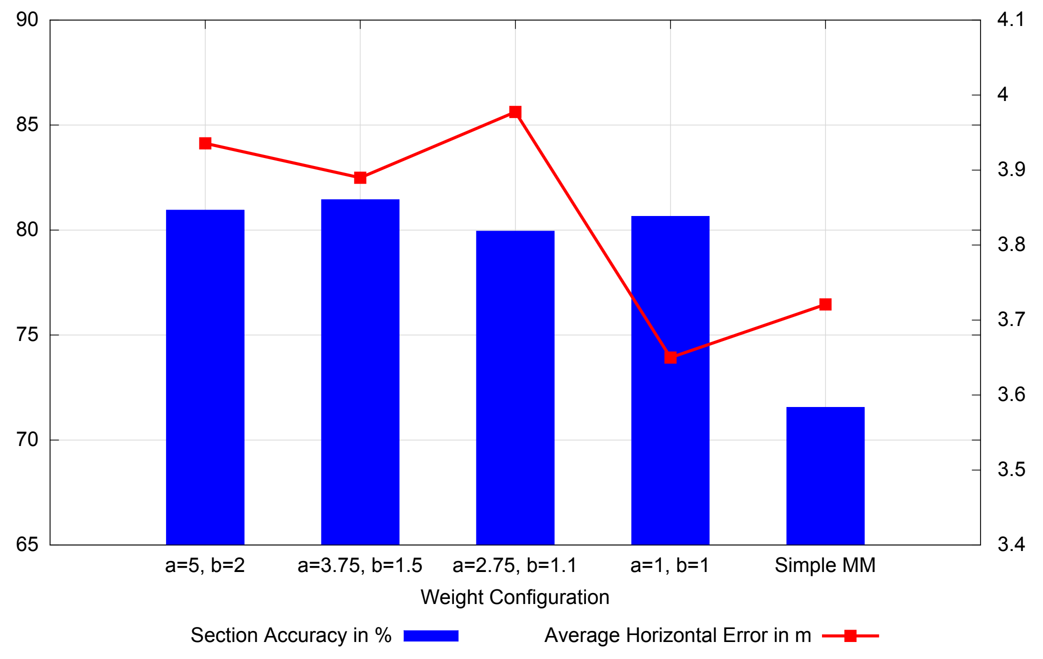


Figure 9.9: Variation of Weight Magnitudes for 50° Degradation and the Düsseldorf Network

Variation of Heading Degradation

We are also interested how the degradation of the heading accuracy influences the section and link accuracy as well as the average horizontal error. We vary the angular degree of degradation between 10° and 50° for the simple network and between 10° and 60° for the Düsseldorf network. The results of experiments with $a = 3.75$ and $b = 1.5$ for the simple network are presented in Figure 9.10 and for the Düsseldorf network they are presented in Figure 9.11.

Both figures reveal the expected results: with increasing degradation the section and link accuracies decrease and the average horizontal error increases. It can be remarked that even with 50° or the proposed map matching algorithm degradation still performs way better in terms of section accuracy than the simple map matching algorithm. In the simple network it also outperforms the simple algorithm in terms of average horizontal error, while it is worse for 50° and 60° degradation in the Düsseldorf network. The good results of the algorithm in the simple network can be accounted to the fact that it is less complex than the Düsseldorf network and not so many pitfalls (such as intersections etc.) are present in this network.

Variation of Candidate Identification Method

As mentioned in Section 9.3 we implemented two different ways to identify candidate sections: building an error region around the fix (Error Region Candidate Identification, ERCI) or calculating the closest node (Closest Node Candidate Identification, CNCI). The comparison of both methods with the simple map matching method is presented in Figure 9.12. We used the Düsseldorf network for simulation, $a = 3.75$ and $b = 1.5$ as weights, and a heading degradation of 30°.

The results impressively show, that the ERCI method outperforms the CNCI method

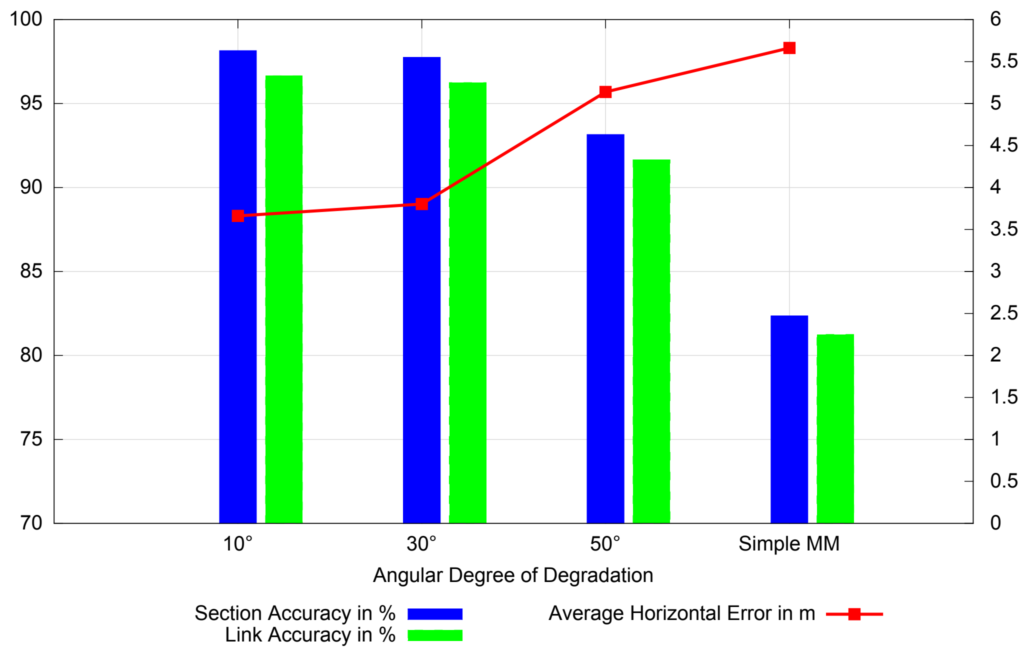


Figure 9.10: Variation of Heading Degradation for the Simple Network

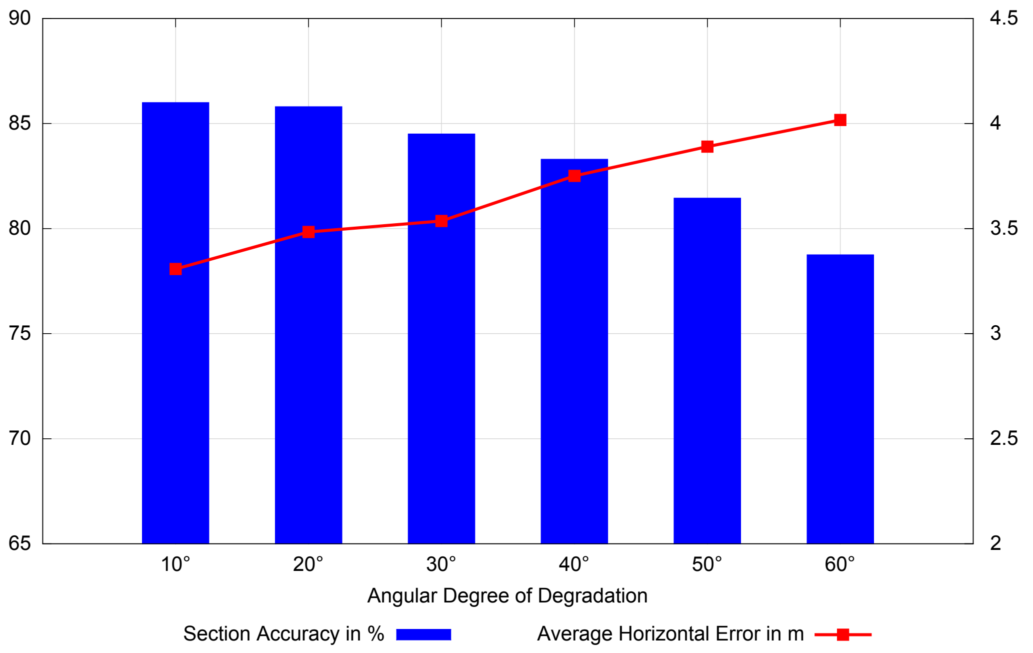


Figure 9.11: Variation of Heading Degradation for the Düsseldorf Network

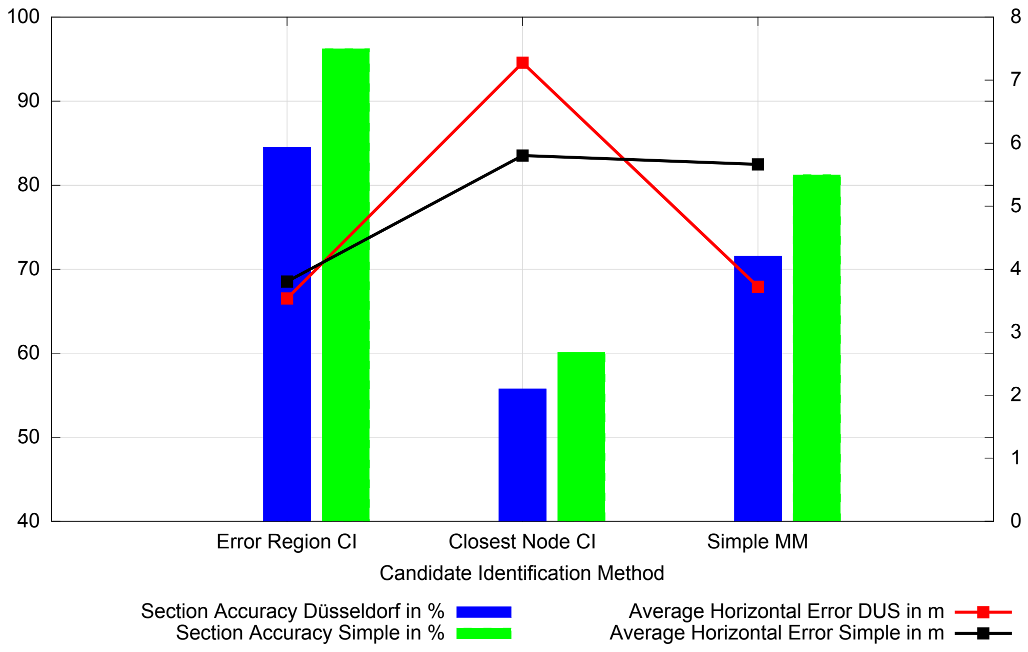


Figure 9.12: Variation Candidate Identification Method in the Düsseldorf Network

and the simple map matching method in both networks. The bad performance of the CNCI method can be accounted to the way how links and sections are represented in the spatial database. As links and sections are unidirectional, the CNCI method picks the wrong section with 50% chance, because start and end point are just exchanged for the sections of both directions.

Position Degradation

Depending on the type of positioning system, positions are tracked with more or less error. The GPS error usually has a standard deviation about 2.81 mfor each coordinate. We expect the positioning error to have effect on the overall performance of the map matching. Hence, we experiment with 2.81 m, 4 m, and 6 mstandard deviation for the positioning error. For degradation we use the Degrade virtual sensor from the QED and TSE applications, respectively. We again use $a = 3.75$ and $b = 1.5$ as weights and a heading degradation of 30°. The results for the simple network are shown in Figure 9.13 and for the Düsseldorf network in Figure 9.14.

From the results we see, that, as expected, the section accuracy decreases with increasing positioning error. The average horizontal error increases with the increasing positioning error for the simple network and the new map matching algorithm from 3.66 m for 2.81 m standard deviation to 7.49 mfor 6 m standard deviation. For the Düsseldorf network and the new Map Matching algorithm it increases from 3.54 m for 2.81 m standard deviation to 7.97 m for 6 m standard deviation. For the simple Map Matching algorithm a similar trend is observable, but the new algorithm is much more robust to changes in the positioning error.

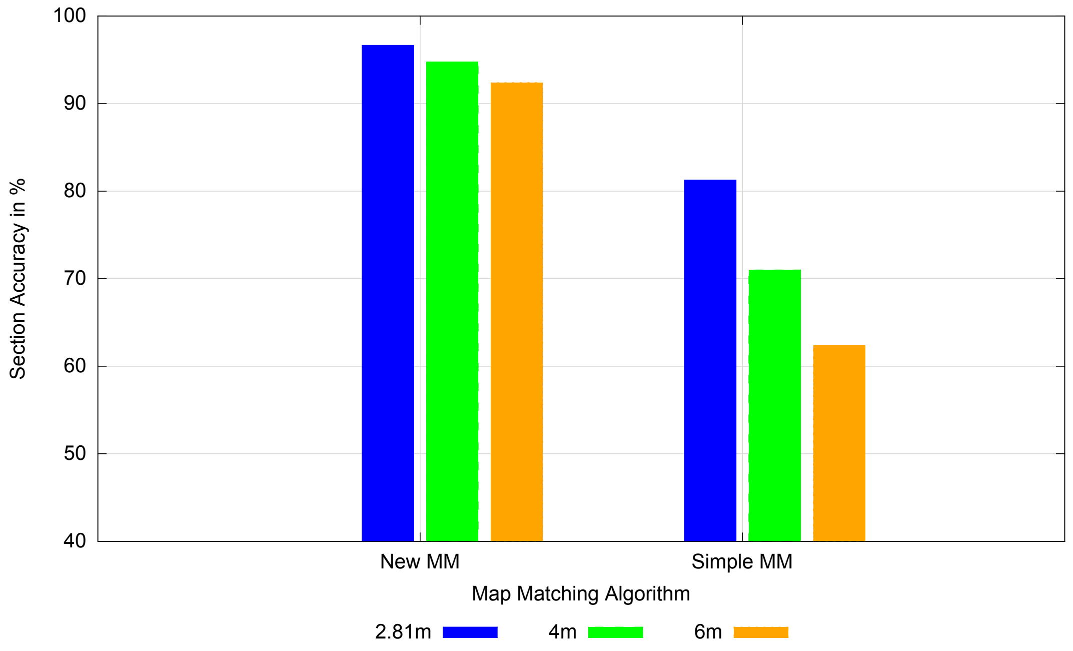


Figure 9.13: Variation of Position Degradation in the Simple Network

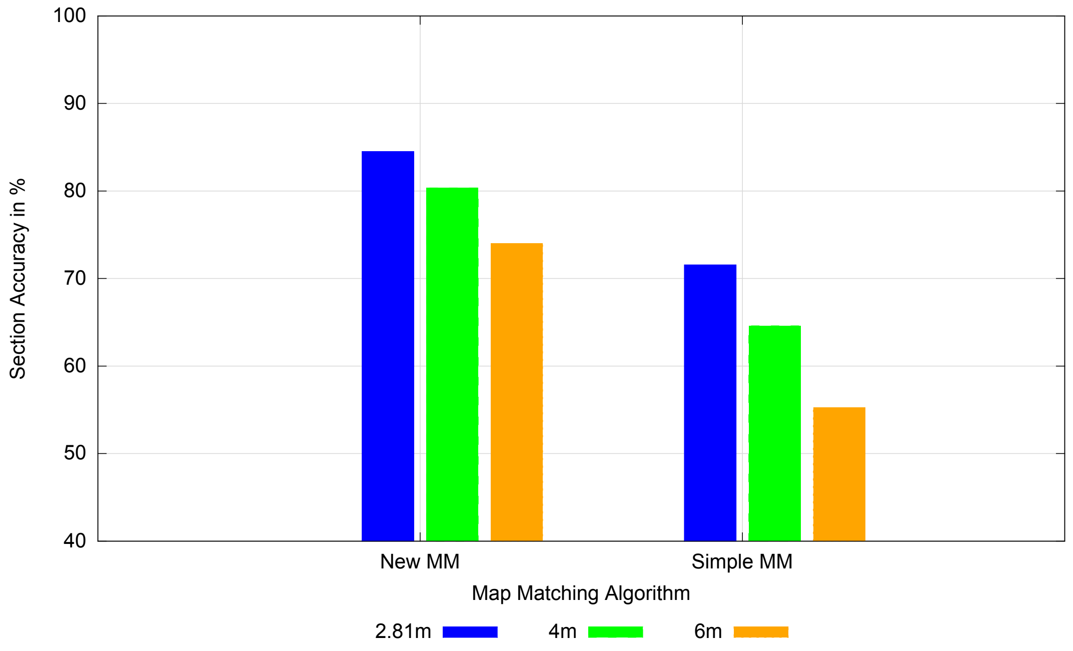


Figure 9.14: Variation of Position Degradation in the Düsseldorf Network

Variation in Section Length

Finally, we want to know if the length of the sections on the links makes a difference in section accuracy and average horizontal error. Hence, we varied the division into sections in the spatial database and made experiments with maximum section lengths of 400 m, 200 m, 100 m, and 50 m. We used the Düsseldorf network for simulation, $a = 3.75$ and $b = 1.5$ as weights, and a heading degradation of 30°. The results are depicted in Figure 9.15.

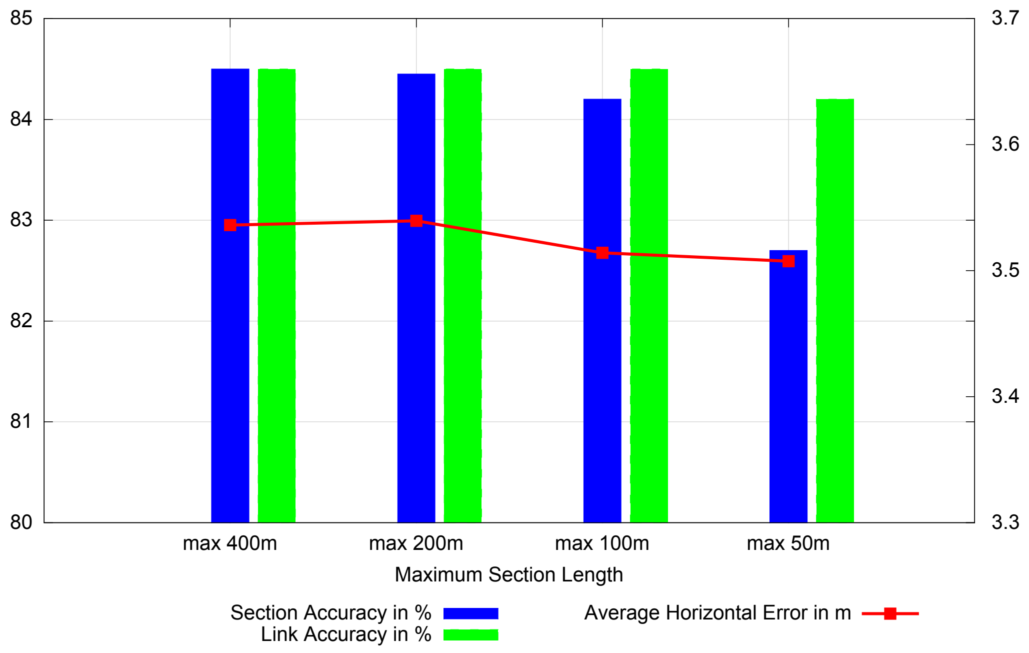


Figure 9.15: Variation of Section Length in the Düsseldorf Network

It can be observed that the section accuracy decreases faster than the link accuracy with decreasing section length. This is explainable when the average section length is analyzed. For a maxmimum of 400 m the average section length is 18.7 m, while for a maximum of 50 m the average is 14.6 m. This means, that the average section length only varies slightly, though the maximum section length has been changed as the Düsseldorf network seems to include only very small sections. Nevertheless, the section length influences the section more than the link accuracy while the average horizontal error decrease is 0.03 mbetween 400 m maximum section length and 50 m maximum section length.

9.4.2 Evaluation in the ACM GIS Map Matching Cup

To compare our map matching algorithm with other recent algorithms, we participated in the ACM SIGSPATIAL GIS Cup 2012 (Ali et al., 2012). In the contest the algorithm was tested on a road network from Washington State, USA with 535.452 nodes and 1.283.540 edges (this corresponds to links and sections in our networks) with their corresponding geometry (sequence of nodes constituting the link). The algorithm was run on several real world data sets with varying numbers of GPS readings and high sampling frequencies (between 1 and 10 seconds). For each correct match contestants earn a point weighted by the confidence of the rating algorithm about the correctness.

Table 9.2: Comparison of Runtimes

	i5 MM	(Tang et al., 2012)	(Levin et al., 2012)
10 Files, 1s Sampling	275,467 s	1,646 s	18,877 s
10 Files, 5s Sampling	205,363 s	1,259 s	14,384 s
10 Files, 10s Sampling	197,851 s	1,339 s	14,279 s
100 Files, 1s Sampling	589,078 s	2,213 s	39,095 s

Table 9.3: Comparison of Correct Link Percentages

	i5 MM	(Tang et al., 2012)	(Levin et al., 2012)
10 Files, 1s Sampling	84,73 %	97,45 %	98,28 %
10 Files, 5s Sampling	85,06 %	98,79 %	95,74 %
10 Files, 10s Sampling	85,56 %	95,16 %	97,37 %
100 Files, 1s Sampling	84,73 %	97,45 %	98,28 %

If the match is false, one point weighted with the same confidence is subtracted. Hence, the overall rating is calculated as follows:

$$\frac{\text{sum of weight. corr. matches} - \text{sum of weight. incorr. matches}}{\text{run time}} \tag{9.16}$$

Furthermore, the initial map loading time was also rated. In the end, our algorithm achieved the 11 place of 31. In the following we present our results compared to the first and fifth best algorithms of the contest. Table 9.2 presents the run time results, while Table 9.3 shows the percentage of correctly matched positions.

The results show that our algorithm still can be improved in terms of run time. It is a magnitude of 10 to 100 slower than the fifth best algorithm. In our own experiments with the Düsseldorf network we reached 16 ms per map matching of a position using the PostgreSQL database. The percentage of correctly matched positions differs from the other approaches between 10 to 14%. These are promising results which still can be improved. But all in all we are very satisfied with the 11 place out of 31. Unfortunately, the results of other participants are not available.

9.5 Discussion and Conclusion

In this chapter we presented a new weighted topological algorithm with two alternative methods for candidate section identification for matching a V2X position to a road network. The algorithm just needs two consecutive positions and the heading of the vehicle to match the position. The algorithm is structured into four consecutive steps: (1) First a candidate set of sections is assembled. (2) For each candidate section weighted scores are calculated. (3) The candidate section with the highest score is selected for matching. (4) A simple map matching procedure matches the position to the selected section. After determining the optimal ratio and magnitudes of the weights, we used our evaluation framework to evaluate different factors which influence the performance of the algorithm. The performance is measured by the section accuracy and the average horizontal error. We found that the heading error, the positioning error, and the average section length influence the performance of the algorithm. Furthermore, we identified the Error Region Candidate Identification method as the best method for selecting the candidate sections. Most importantly, we proved that the new algorithm outperforms the simple map matching algorithm. In the ACM SIGSPATIAL GIS Cup contest we compared our algorithm with other algorithms and showed that we are among the first 12 algorithms according to matching accuracy and run time, but that there is still improvement necessary. Finally, we determined that the algorithm is suitable for V2X real-time Map Matching as map matching a position only takes 16 ms on average with the PostgreSQL database including the PostGIS extension.

Chapter 10

Data Stream Classification

In Chapters 6 and 7 the data processing of data streams for C-ITS and mHealth applications using examples has already been detailed. We explained that we use data stream classification algorithms to learn to derive further information based on the incoming data. So far only the Hoeffding Tree algorithm from the MOA framework has been used in the applications for evaluation. But there are still many questions open regarding the choice of the data stream mining algorithm. The Hoeffding Tree might not be the best choice for all applications and in all situations. Hence, we conducted several experiments to find out, which algorithm suits which task best, what parameters specific to the mining influence the algorithms, if balancing makes a difference, and which kind of balancing is best, and which algorithms perform best for concept drifts.

10.1 Classification Algorithms

In the MOA mining framework various stream mining algorithms are readily available, most of them based on the basic algorithm of Hoeffding Trees detailed in Section 4.4. To find out, how well the algorithms are suited to be used in the application of traffic state estimation, several tests with varying algorithms have been made. One distinction of the implementations of the algorithms is in their prediction strategy. The prediction strategy is the method to decide at the leaves in the decision tree, which class for the element to classify is assigned. There are three different prediction strategies implemented in MOA: Majority Voting, Naïve Bayes Leaves and Adaptive Hybrid (Bifet and Kirkby, 2009a). While majority voting assigns the class, for which the most examples arrived at this leaf, the Naïve Bayes Leaves uses the Bayes theorem and the probabilities of the attribute values to determine the probability of each class and selects the class with the highest probability. The Adaptive Hybrid strategy is a combination of both. For each strategy the accuracy is calculated and the Naïve Bayes decision is only used if its accuracy was higher before. In Figure 10.1 the accuracy results of all three strategies for a Hoeffding Tree are shown. On average, the Naïve Bayes and the Adaptive Hybrid strategy were very similar in their results, but both outperformed the Majority Voting strategy.

Ensemble algorithms often deliver better results than single learners as they can learn more diversely. However, they are computationally more expensive. Implementations of various stream ensemble algorithms are also included in MOA. We analyzed all of them in our evaluation framework using the Hoeffding Tree as the base learner. The ensemble algorithms comprise the online boosting and bagging algorithms by Oza and

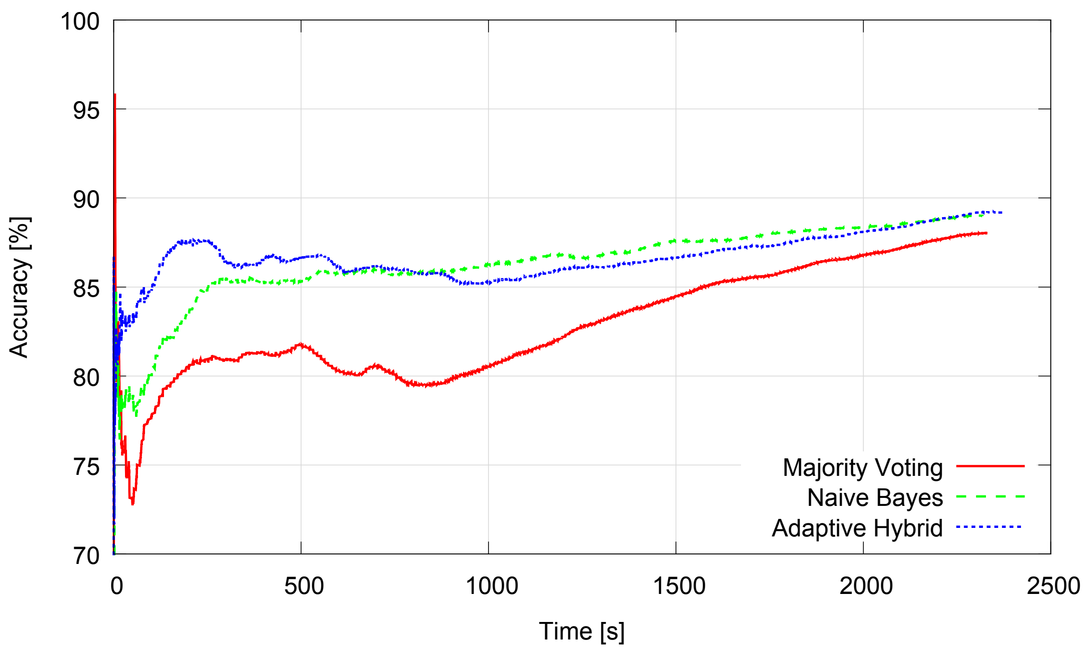


Figure 10.1: Accuracy for Hoeffding Trees with Varying Prediction Strategy

Russell (Oza and Russell, 2001; Oza, 2005), the Online Coordinate Boosting algorithm presented in (Pelossof et al., 2009), and an Option Tree algorithm introduced in (Bifet and Kirkby, 2009a). Option trees grow new alternative branches at so called option nodes. Examples are pipelined to all branches and leaves connected with the option node and the overall result is determined by combining the results of the nodes beneath the option node. We do not go into the details of these algorithms, but refer the interested reader to the cited papers. Figure 10.2 shows the results for the tested ensemble algorithms. The boosting algorithm is slightly better than the other algorithms with accuracies of up to 92%. On average, it delivers also only a slightly better accuracy than the best single classifier algorithm in Figure 10.1.

So far, a new classification model has been used for each simulation run. Each simulation run was 1800 s long. If the model is trained over a longer period of time, it is assumed that the accuracy will also be increased, as the model has more examples to learn on. Therefore, multiple simulation runs have been carried out, reusing the trained model in each run and thereby incrementally extending it. The results shown in Figure 10.3 corroborated our assumption. The overall accuracy of the mining increased from approximately 88% to 91%, which denotes an improvement, but not as high as expected.

10.2 Class Balancing

In the experiments before, we identified the unbalanced ratio of examples for negatives and positives as a problem in particular for the QED case study. This is a problem very well known in machine learning, but often not the only one which contributes to a bad learner performance (Batista et al., 2004). We decided to analyze the problem and assess the influence of the unbalanced ratio of negative and positive examples for the three measures accuracy, sensitivity, and specificity. In the queue-end detection application we are especially interested in the true positives, as these reflect the cor-

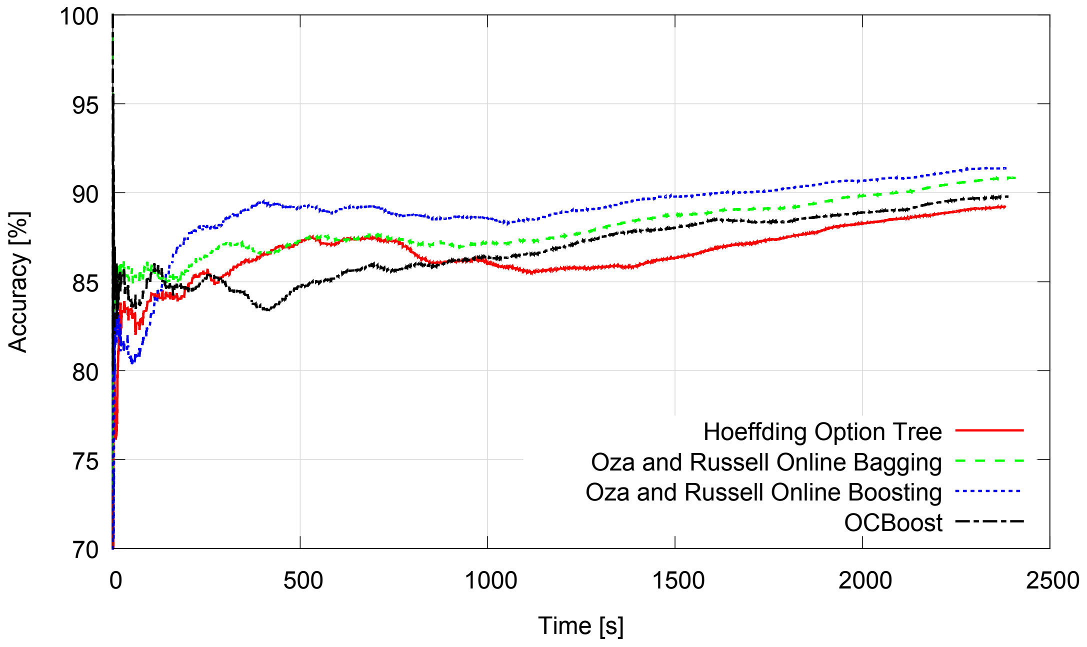


Figure 10.2: Accuracy of Multiple Ensemble Algorithms

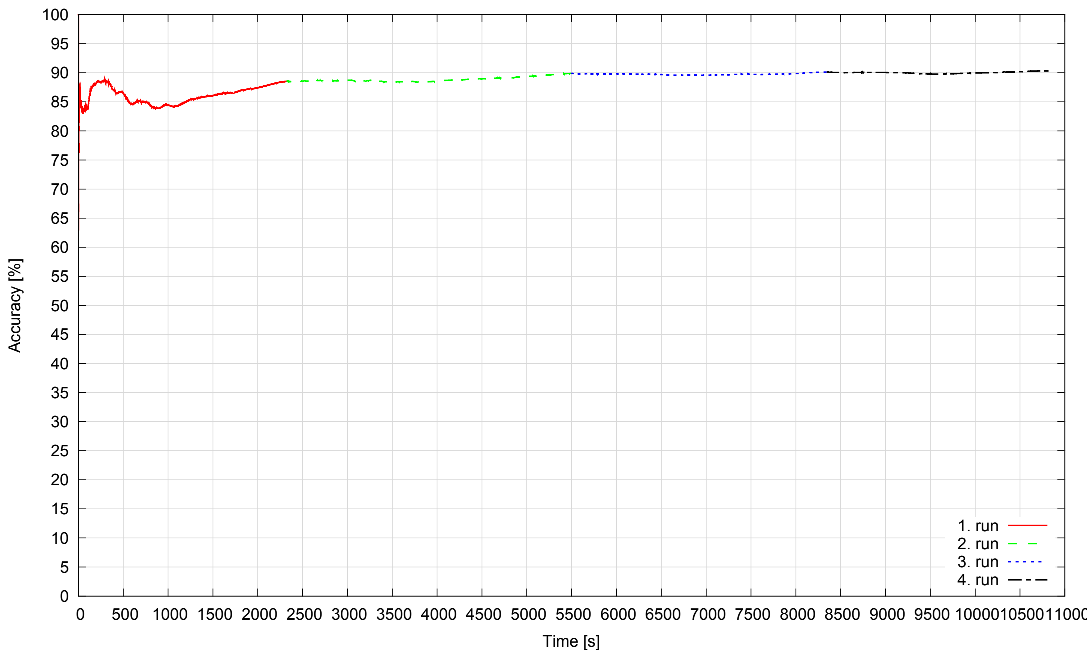


Figure 10.3: Development of Accuracy in Successive Runs

rectly identified queue-ends. Hence, the sensitivity is the most important evaluation measure we have to look at. In the following, we did several experiments with different balancing techniques (undersampling as well as oversampling).

10.2.1 Undersampling Techniques

First, we implemented a very simple undersampling algorithm suited for streams. For each negative example we draw a random number from an equal distribution of numbers between 1 and 100. A negative example is dropped when the random integer is less or equal to the percentage of negatives which should be dropped (if 75% should be dropped all elements with integers below 76 are dropped).

We used as parameter values penetration rate to 5%, the traffic volume to 2500 veh/h, the section length to 100 m, and the window size to 120 s for all runs. We tested percentages of elements between 0% and 90%. Figure 10.4 shows the results.

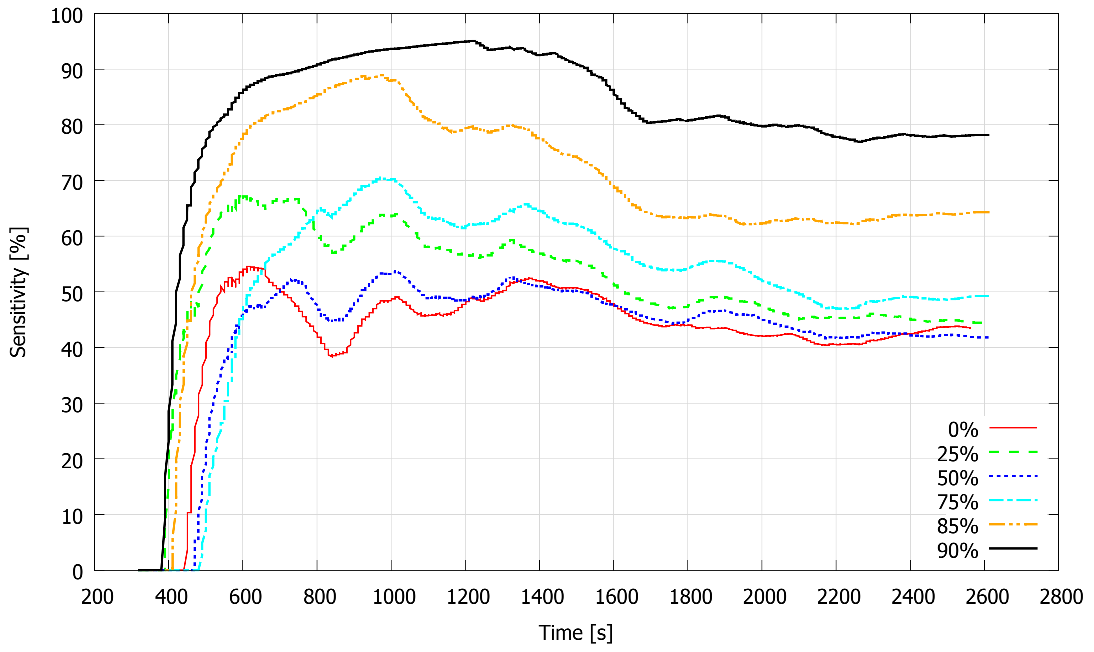


Figure 10.4: Sensitivity for Random Undersampling with Varying Undersampling Rates

It can be derived from the results that dropping a high number of negatives is very beneficial for the sensitivity. But the results for the specificity reveal, that this has the opposite effect on the specificity (only reaches a little more than 50% for the 90% run), though the ratio of positives to negatives is almost equal (1.2:1). A good trade-off seems to be 85%, where the ratio is 0.85:1. Here the sensitivity reaches values of about 65% while the specificity is around 82%.

We also rerun the experiment with three different seeds for the probability distributions in the simulation and calculated the average. As simulation parameters we used a road network with 5km, a penetration rate of 5%, a traffic volume of 3000veh/h, a section length of 100m , and a time-based window of size 120s. As these provide the best results for sensitivity, specificity, and accuracy, we used them as the default parameters in the following experiments. Figure 10.5 shows, that the results are similar and tendencies are the same compared with Figure 10.4.

For the same application we also implemented and tested other undersampling

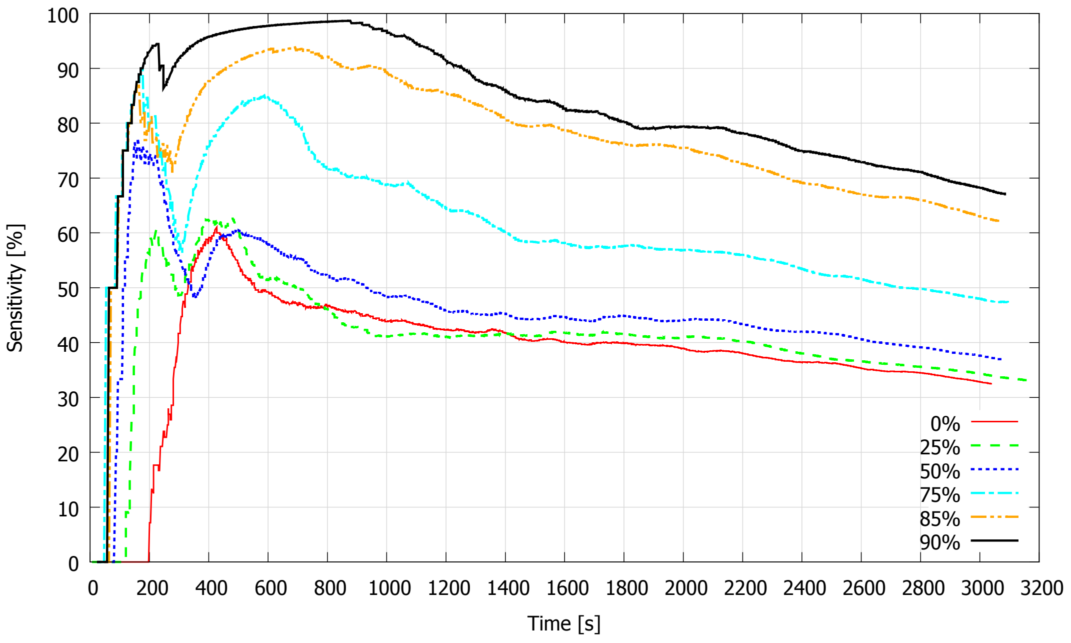


Figure 10.5: Sensitivity for Random Undersampling Algorithm Using Different Rates and Averaging over Different Seeds

techniques. We implemented a variation of the original Bernoulli Sampling algorithm (Gemulla et al., 2006). The variation does not consider a set for the sampling, but is also working in an incremental fashion. It is similar to the random sampling technique, but uses float numbers instead of integers to draw from. The results for the sensitivity are depicted in Figure 10.6.

The results also resemble the results from the previous undersampling technique. With increasing dropping rate the sensitivity also increases.

Finally, we also implemented a version of the Reservoir Sampling algorithm. We do not maintain an actual reservoir, but drop an item, if the condition for including the element in the "intended" reservoir is fulfilled. That is, an item is dropped with probability $\frac{M}{N}$ where M is the reservoir size parameter and N is the number of elements seen so far. Figure 10.7 shows the results for the sensitivity values.

The results for sensitivity show, that with increasing reservoir size the sensitivity value drops. This is a logical consequence as the probability to keep an element is higher with a higher reservoir size. While sensitivity drops, accuracy and specificity are increasing as can be seen from Figure 10.8 and Figure 10.9.

A good compromise cannot be found based on these values as either accuracy and specificity or sensitivity fall below 50%, e.g., for a reservoir size of 50. Hence, the reservoir sampling method seems not to be a good alternative.

10.2.2 Oversampling Techniques

Besides undersampling techniques we also experimented with oversampling techniques. We implemented a simple form of random oversampling for which a window with the last x elements is maintained in a first-in first-out manner. From the window new elements are artificially generated (i.e., copied). A parameter determines with which ratio new items are generated. A value of 50% means generation of one new item for

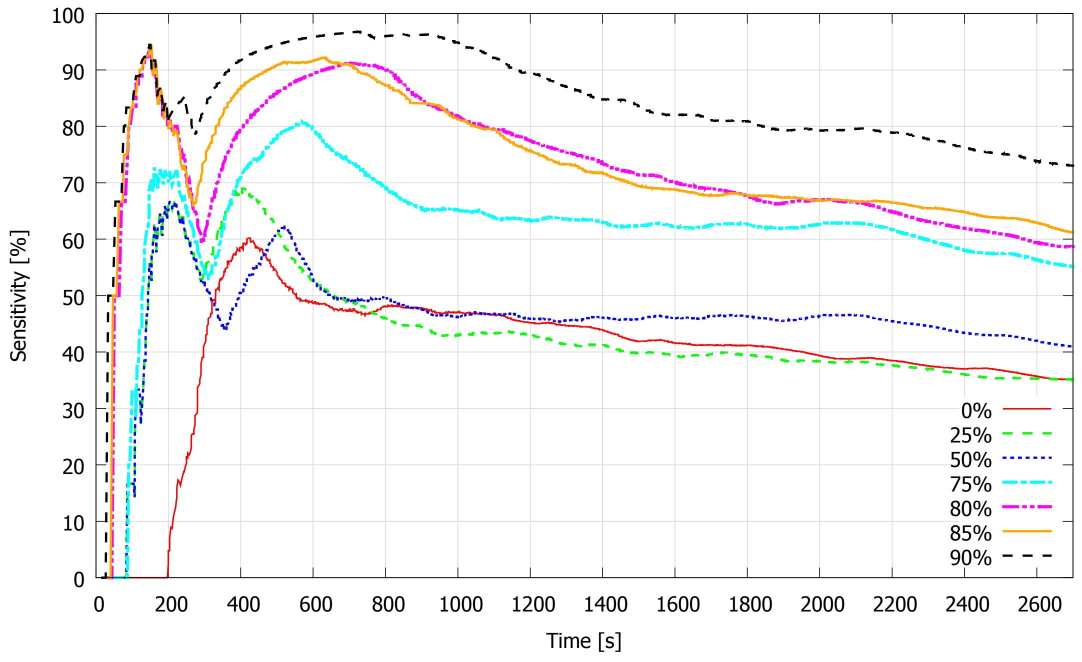


Figure 10.6: Sensitivity for Bernoulli Undersampling with Varying Sampling Rates

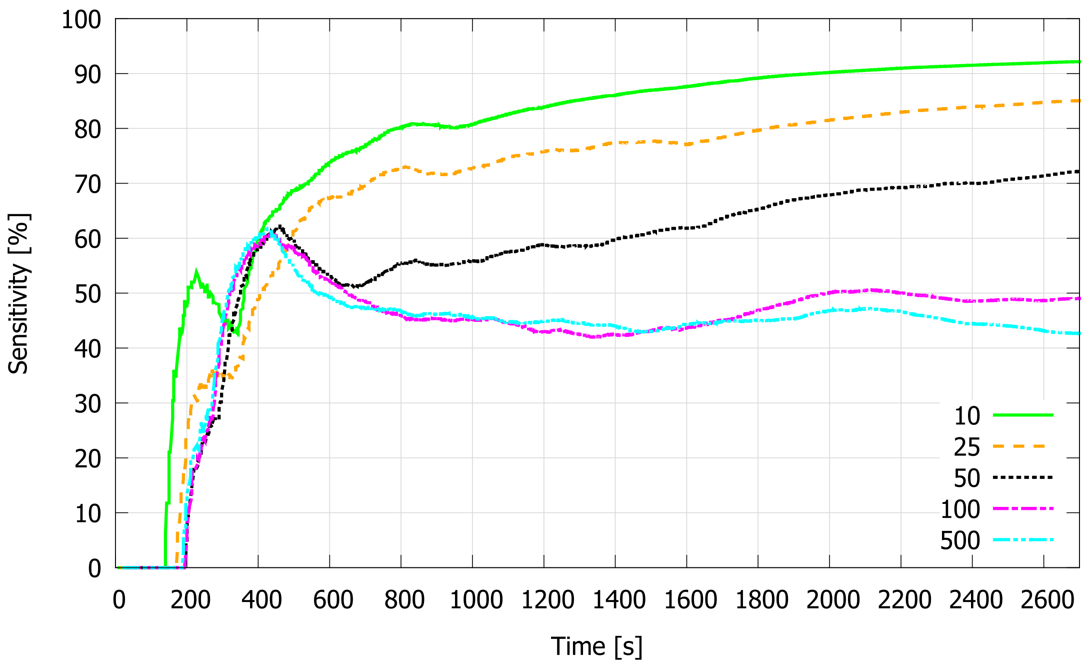


Figure 10.7: Sensitivity for Reservoir Undersampling with Varying Reservoir Sizes

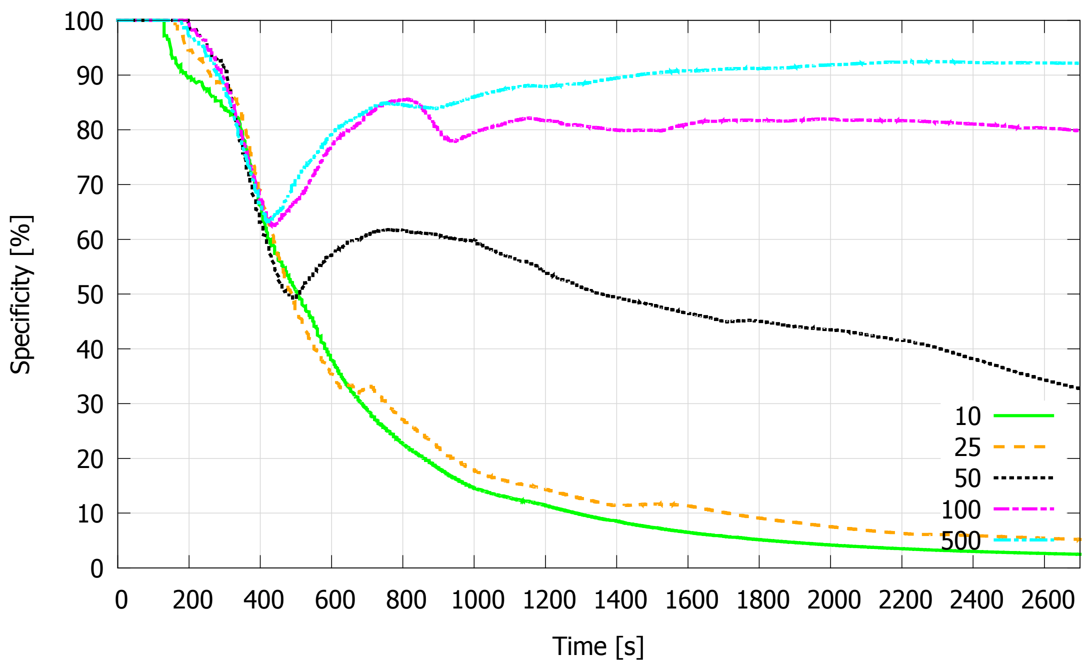


Figure 10.8: Specificity for Reservoir Undersampling with Varying Reservoir Sizes

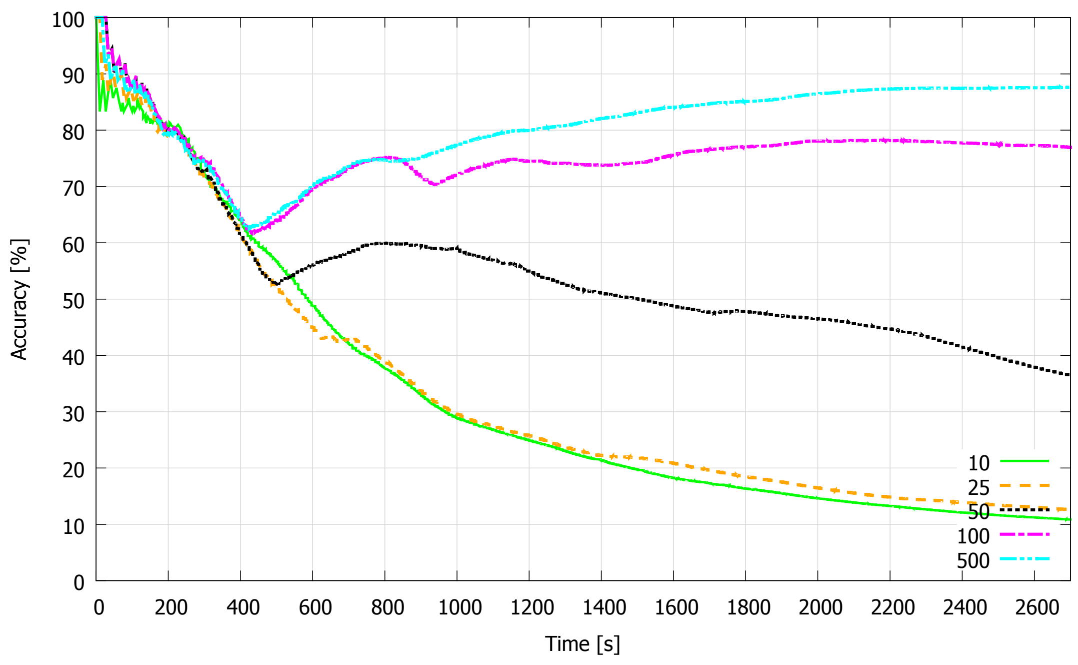


Figure 10.9: Accuracy for Reservoir Undersampling with Varying Reservoir Sizes

each pair of old items. If the value is greater than 100% more than one new item is created per each old item. We experimented with different oversampling ratio values and a window size of 10 elements. The results are presented in Figures 10.10 to 10.12.

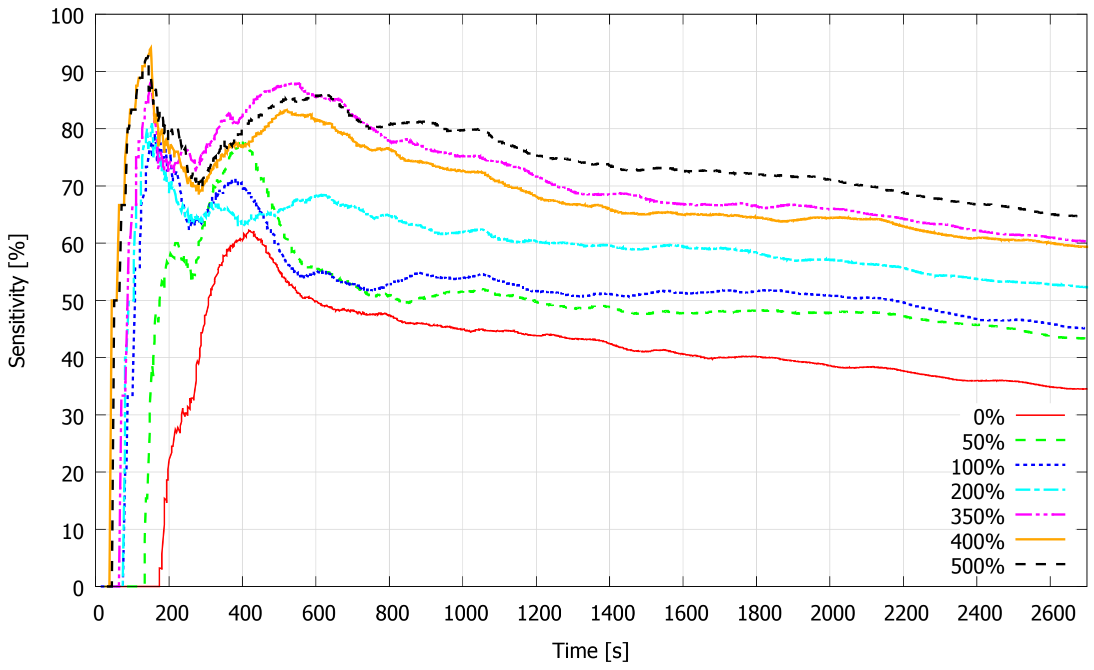


Figure 10.10: Sensitivity for Random Oversampling with Varying Oversampling Ratio Values

Figure 10.10 shows, that with increase of the examples for the minority class (the positives) also sensitivity is increasing while specificity and accuracy are decreased. However, 200% (i.e., two new elements are created per one old element) or 350% seem to be good compromises to balance sensitivity and specificity.

10.2.3 Dynamic Balancing

What could be seen for some of the results from the previous sections is, that sensitivity deteriorates the longer the simulation runs. One reason could be that more examples of the minority class are produced. To approach such a change in the class distribution, we implemented dynamic variants of the sampling algorithm described before. The algorithms use the statistics recorded by the data mining algorithm, i.e., the overall examples and the examples of each class seen by the algorithm so far. The dynamic undersampling algorithms calculate the drop rate for each class c using Equation 10.1.

$$\text{dropRate}_c = \frac{\text{elementsSeen}_c - \text{elementsSeen}_{\text{min}}}{\text{elementsSeen}_c} \tag{10.1}$$

The number of seen elements for the minority class min is substracted from the elements seen of class c and divided by those number. As an example, let $\text{elementsSeen}_c =$ 9 and $\text{elementsSeen}_{\text{min}} = 3$ for class c being the negatives (no queue-end). Then the dropping rate is 0.75.

For the oversampling algorithms the oversampling rate for each class c is calculated using Equation 10.2.

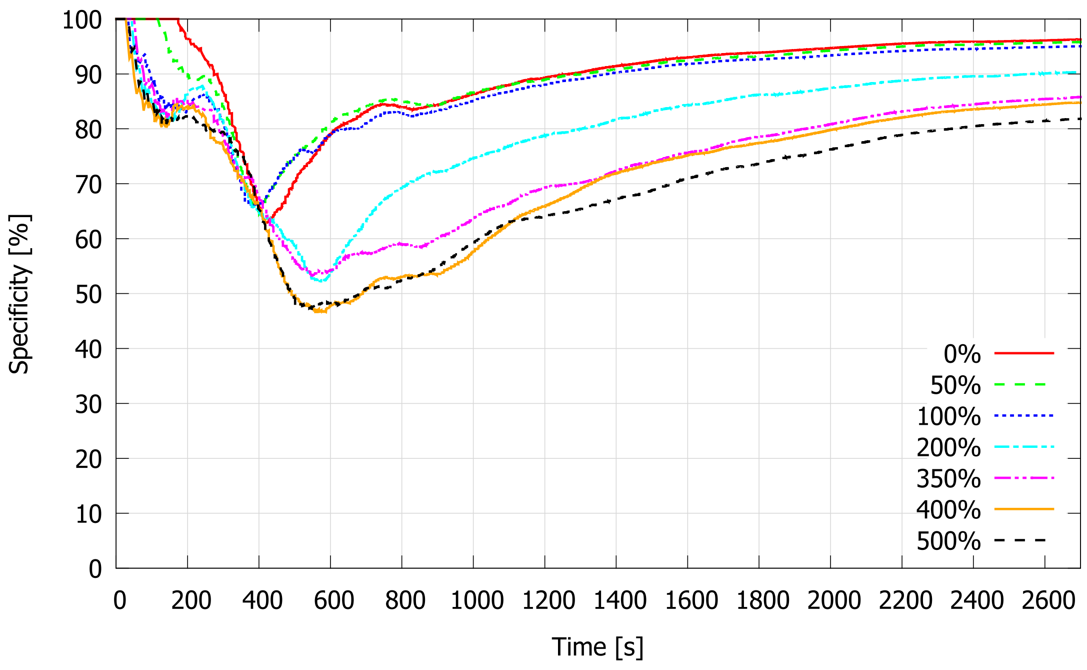


Figure 10.11: Specificity for Random Oversampling with Varying Oversampling Ratio Values

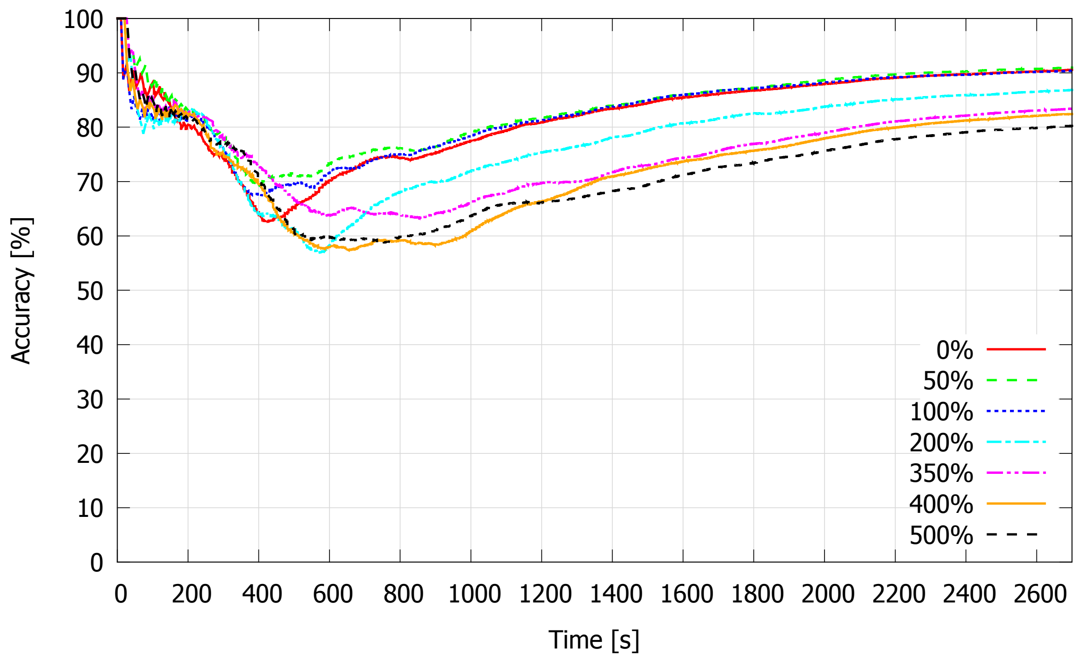


Figure 10.12: Accuracy for Random Oversampling with Varying Oversampling Ratio Values

$$\text{oversamplingRate}_c = \frac{\text{elementsSeen}_{\text{maj}} - \text{elementsSeen}_c}{\text{elementsSeen}_c} \tag{10.2}$$

Reusing the example before, the oversampling rate would be 2 (200%) given that class c are the positives (queue-end).

We made experiments with each of the dynamic variants, using the parameter configuration from beforehand. The results for sensitivity, specificity, and accuracy are shown in Figures 10.13 to 10.15.

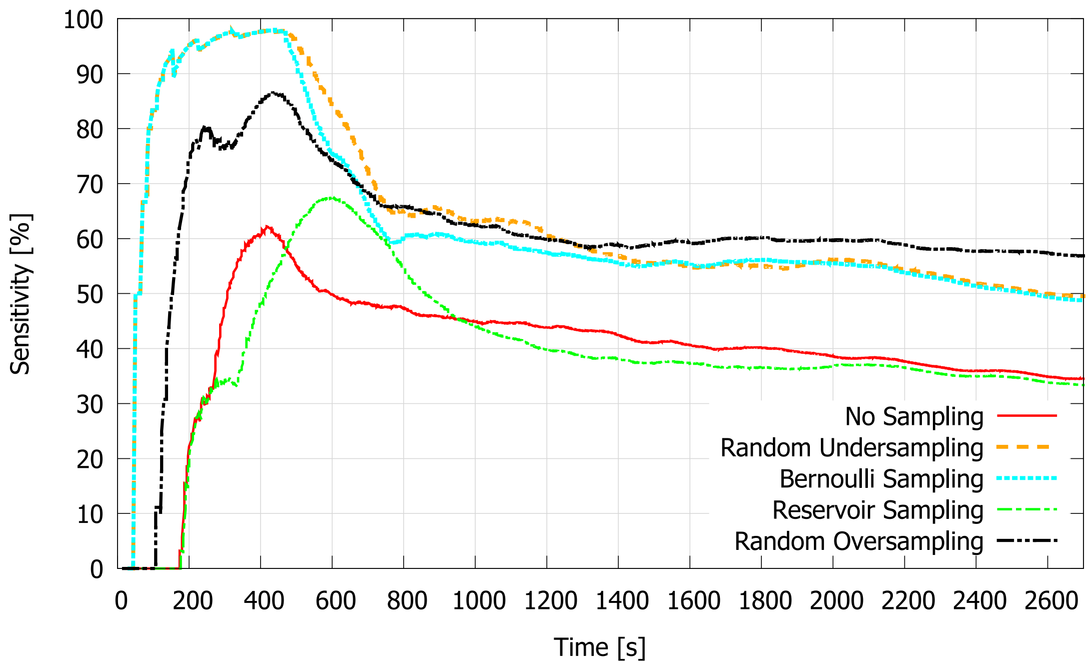


Figure 10.13: Sensitivity for Dynamic Balancing with Varying Sampling Algorithms

The results in Figure 10.13 show, that for sensitivity Random Undersampling and Random Oversampling perform the best. But in comparison to the static random variants, the results of the dynamic variants are worse with about 8 to 10% in the end. But the results are more stable for the dynamic variants and do not deteriorate that much over time. For specificity and accuracy the results are very good and also keep stable.

In the previous experiment, we have investigated the influence of the dynamic balancing variants on the measures sensitivity, specificity, and accuracy for the QED application. The QED application was considered as a two class problem as elements were only classified into queue-end or no queue-end classes. For the TSE application this is different as already discussed in Chapter 6. In the TSE application four classes are distinguished, which makes balancing more difficult. We wanted to know if the dynamic balancing is working better for the TSE application and more balanced data sets. Here, we only use the accuracy as a measure. The equations to calculate the accuracy for the classes have been introduced in Chapter 6 in Section 6.3.6. We experimented with the Düsseldorf network with varying traffic volumes, window size of 120s, 5% penetration rate, and 100m section size. Simulations were done with three different seeds for the random components for each algorithm and the result represents the average of the three seeds. Figure 10.16 shows the accuracy results.

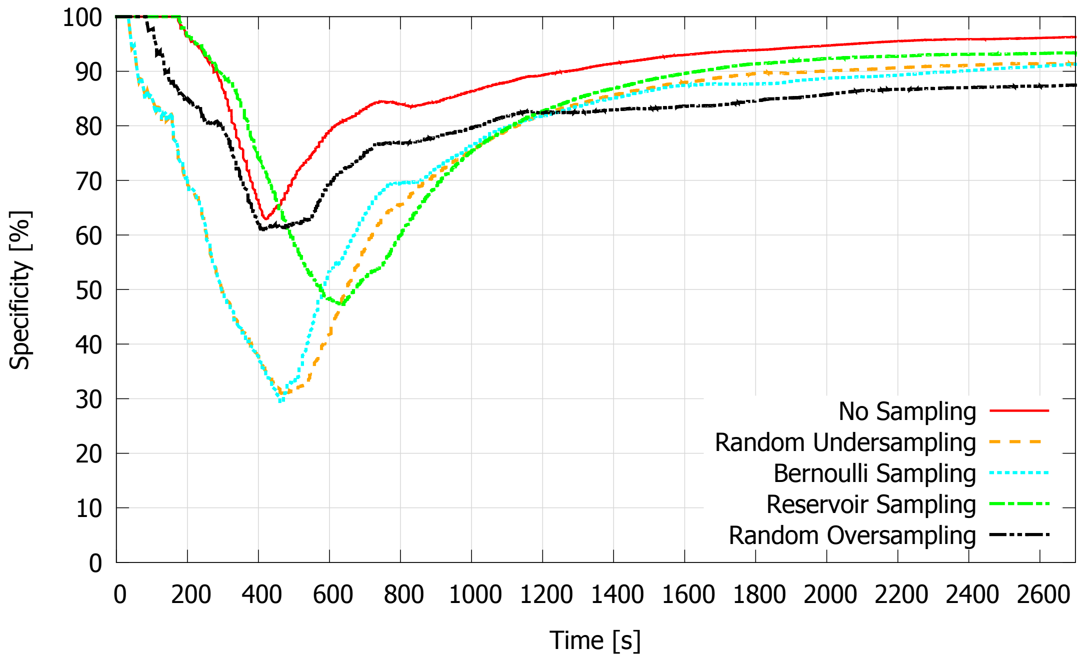


Figure 10.14: Specificity for Dynamic Balancing with Varying Sampling Algorithms

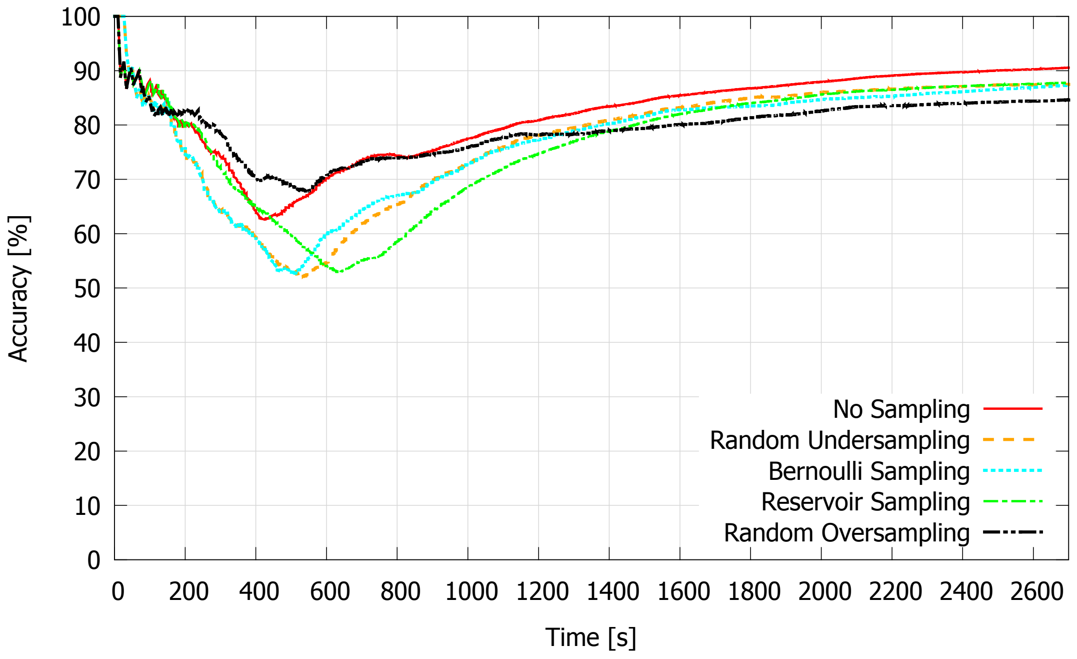


Figure 10.15: Accuracy for Dynamic Balancing with Varying Sampling Algorithms

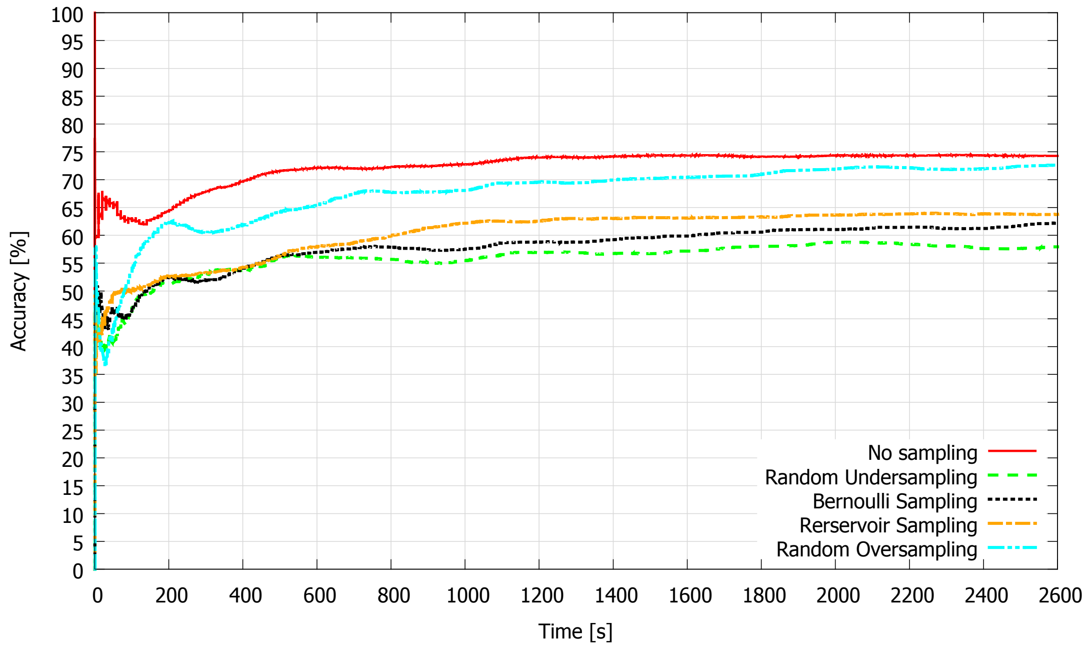


Figure 10.16: Accuracy for Dynamic Balancing with Varying Sampling Algorithms for the TSE Application

In contrast to the QED application, for the TSE application no sampling and Random Oversampling outperfom the other algorithms. In comparison to the results in Section 6.3.6 and Section 10.1 the balancing is harmful to the accuracy. The accuracy is about 10% to 15% lower than without balancing.

10.3 Concept Drift

In Chapter 4 we have discussed that class imbalance and concept drift are the main causes which harm results of data stream mining algorithms. Hence, we also analyzed algorithms adapting to concept drift according to their accuracy. First, we wanted to see, how the concept-adapting algorithms perform comparably well to the other algorithms without an actual concept drift being present. Hence, we used first the "normal" QED scenario and network to evaluate them, as it does not exhibit a concept drift. Subsequently, we will describe the implementation of two scenarios exhibiting concept drift and present the corresponding results.

MOA provides four different algorithms. They implement the online bagging algorithm by Oza and Russell using an Adaptive-Size Hoeffding Tree (ASHT) as the base learner. The ASHT is limited in the number of splitting nodes. When the maximum number is reached, nodes are deleted. (Bifet and Kirkby, 2009a) argue, that smaller trees can adapt faster to changes than bigger trees, while the latter ones are more accurate for stationary streams due to the higher number of examples seen. Furthermore, they combine the Oza and Russell bagging with a concept detection algorithm called ADWIN, which has been proposed in (Bifet and Gavalda, 2007). ADWIN maintains a window which can adapt its size to the degree of change in the stream, meaning that the window always represents the average of the streams numeric attributes. If the current window content does not represent the average anymore, concept drift has taken place and the window adapts its size. Furthermore, an implementation of a single classifier combined with a concept drift detection method from (Gama et al., 2004) has been integrated as well as an adaptive version of the Hoeffding Option Tree. Details about all implemented algorithms in MOA can be found in (Bifet and Kirkby, 2009b). Figure 10.17 reveals that all of the concept-adapting algorithms perform with almost identical accuracy. Compared to the ensemble techniques in Figure 10.2 they do not provide substantial improvements in accuracy – actually, the Oza and Russell online boosting is better in the average.

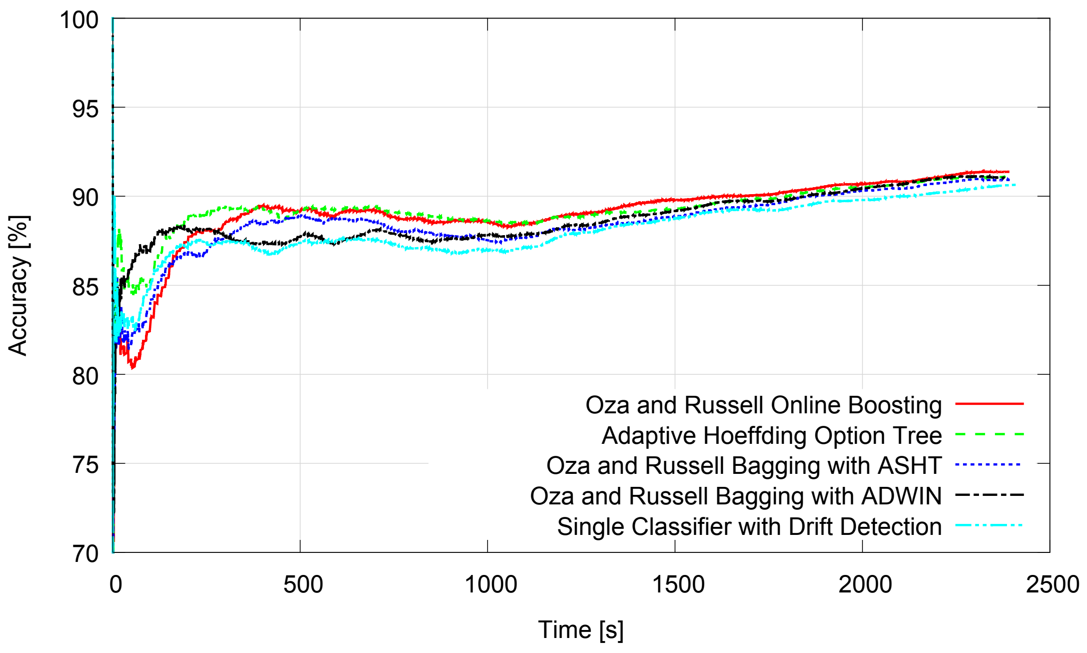


Figure 10.17: Accuracy of Multiple Concept-adapting Algorithms

10.3.1 The Winter Scenario

The scenarios described so far did not exhibit any concept drift. Hence, we created an event which produces a typical concept drift for traffic applications, namely the onset of winter, artificially. We utilize the QED scenario and road network and adapt it towards this winter scenario. The change in driving behaviour for an onset of winter has been analyzed in detail in literature. In Table 10.1 we summarize the most crucial differences we identified in literature between winter and summer conditions. Furthermore, we explain how we adapted the simulation to simulate this change. We defined specific "winter" vehicle types (for V2X vehicles as well as for unequipped vehicles) in VISSIM which fulfill the requirements named in Table 10.1. The penetration rate of CoCar vehicles is 5%.

The winter scenario causes a sudden concept drift as the concept changes in a very short period of time. In Figure 10.18 the results for the winter scenario are presented.

In the figure the change of the underlying conditions and the point in time where the concept drift gets apparent in the data is indicated by two black lines. In comparison the results of the single classifiers deteriorate when the concept drift occurs. Hence, it can be said the ensemble classifiers perform more stable than the single classifiers.

Table 10.1: Parameter Configuration of the Winter Scenario

Property	**Transfer to Simulation**
Vehicle Speed	We reduce the speed limits on the links by between 30 km/h as this has been observed for German highways (Durth et al., 1989) and 50 km/h. Hence, summer cars have a speed distribution between 88 km/h and 130 km/h, while the summer truck speed distribution is ranged between 75 km/h and 100 km/h. In winter cars drive speeds between 58 km/h and 68 km/h, while trucks drive between 40 km/h and 45 km/h.
Reduced Braking Distance	This is fixed to 15m in VISSIM by default. To simulate increased distances we changed this value based on a field study (Durth and Hanke, 2004) on German highways between 50 m and 80 m, respectively.
Average Deceleration	Based on (Maurer, 2007; Mitunevičius et al., 2009) we set the desired deceleration of the winter cars to -2.7 m/s^2 and -3.3 m/s^2 and for winter trucks between -0.7 m/s^2 and -1.3 m/s^2, respectively. For summer cars the desired deceleration is between -4.7 m/s^2 and -5.3 m/s^2, for summer trucks it is between -1 m/s^2 and -1.5 m/s^2.
Reduced Road Capacity	This is simulated in our scenario by reducing the traffic flow in two steps. We start with 3000 veh/h consisting of summer vehicles, after 1000s we reduce to 2500 veh/h consisting of summer vehicles, and finally we have 2500 veh/h only winter vehicles.

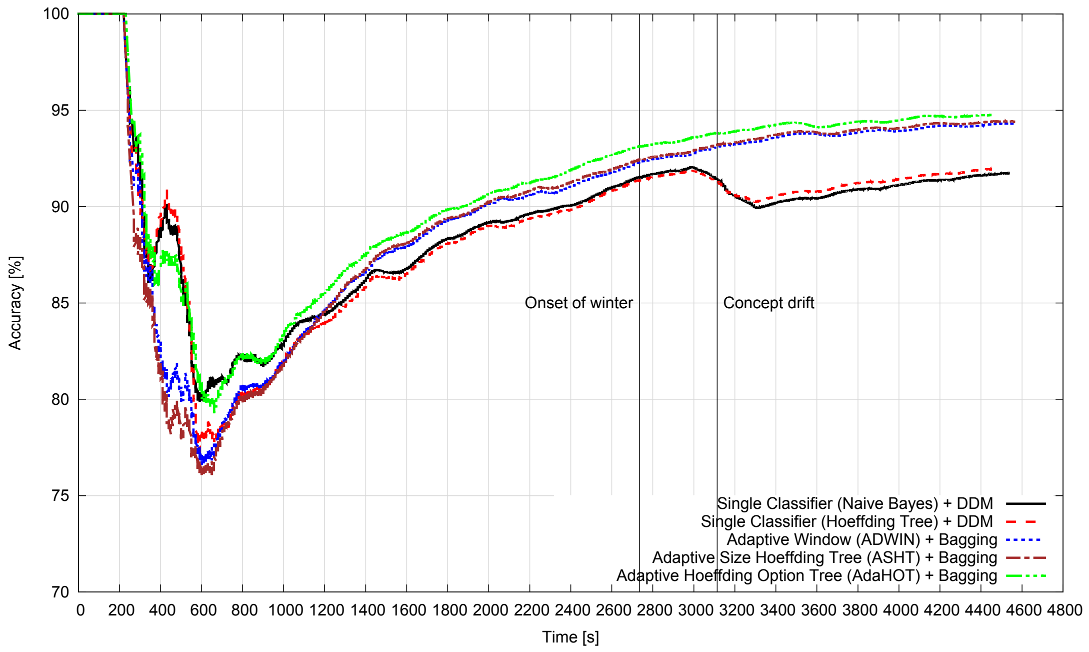


Figure 10.18: Accuracy for the Winter Scenario Comparing Different Concept-adapting Algorithms

10.3.2 The Football Match Scenario

In the former scenario a good example for a sudden concept drift has been given. Due to the weather conditions the overall driving behavior changes. Other good scenarios for sudden concept drifts are events which cause many vehicles to drive from one area into another in a very short period of time, while this is not the case for other periods. That means, the capacity of a road or a road network is exceeded. Such an event is a football match which occurs only once a while in a certain area of a city. In a very short period of time, many vehicles drive to the parking lots, and drive back after the match. This should cause a rapid change in the concept. We simulated a football match for the QED application by increasing the traffic volume stepwise. We use the 10km QED artificial road network, a simulation time of 3600s, and a V2X penetration rate of 5%. We varied the learning algorithms as before. The following increases in traffic flow were made during simulation run time:

- 0s - 1200s: 2500 veh/h
- 1200s - 1800s: 3000 veh/h
- 1800s - 2400s: 4000 veh/h
- 2400s - 3600s: 4500 veh/h

The results of the experiment are shown in Figure 10.19.

Unfortunately, only a slight drift could be created by the selected parameter configuration (most probably at 1900s). As already seen in the winter scenario the single classifier performs slightly worse than the others, while the ensemble algorithms are almost the same. No real difference could be determined.

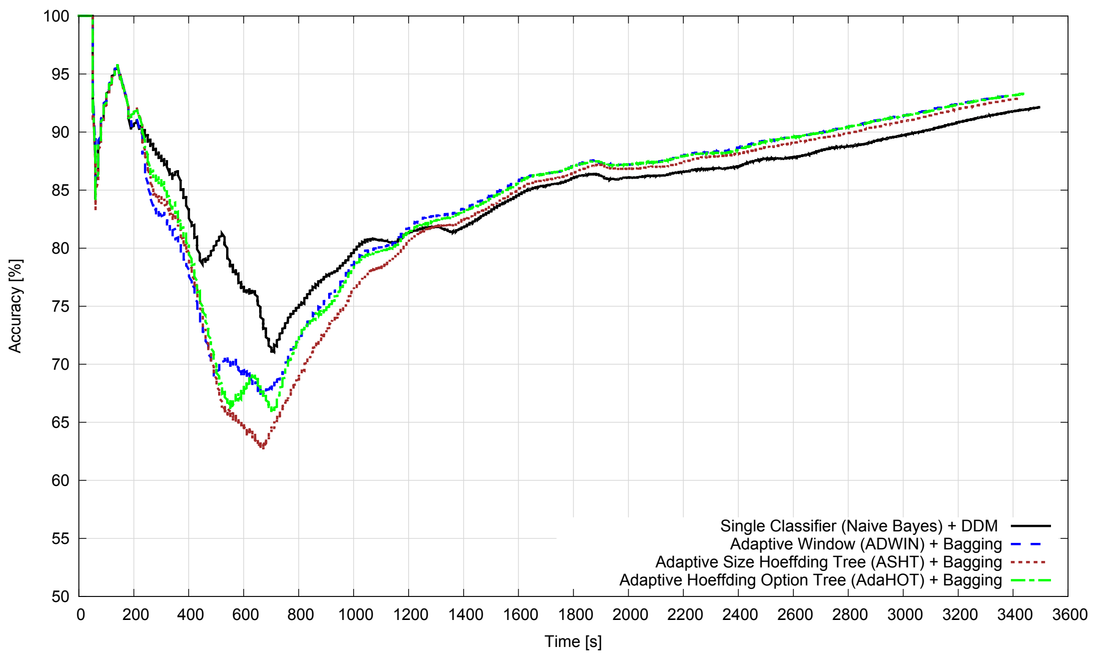


Figure 10.19: Accuracy for the Football Match Scenario Comparing Different Concept-adapting Algorithms

10.4 Conclusion

In this chapter we used our evaluation framework to study the use of different variants of data stream classification algorithms. The experiments showed that for the voting strategies Naive Bayes and Adaptive Hybrid outperformed Majority Voting. Furthermore, we identified in the non concept-adaptable ensemble the boosting algorithms to perform best. We found out that the classification result of the Hoeffding Tree stabilizes at some point in time even when the simulation runs for a very long time. For balancing we identified undersampling techniques as performing best compared to oversampling algorithms and among these the random under- and oversampling algorithms outperformed the others. The implementation of a dynamically adapting balancing variants did not show any improvement. Instead the results got worse. A windowed implementation of this dynamic balancing is interesting to research, but is subject to future work. Finally, we presented two scenarios invoking sudden concept drifts. For the winter scenario we identified the ensemble concept-adapting algorithms to perform better than the single concept-adapting classifiers. The second scenario did not show an as obvious concept drift as in the first scenario, but here also the single classifier performed the least good.

Part IV

Epilog

Chapter 11

Conclusion & Outlook

In this chapter, we will review the research goals and questions formulated initially, and we discuss how our contributions help in answering these questions. We conclude by outlining future work.

11.1 Conclusion

In this thesis we presented a holistic contribution to the development and evaluation of high-quality data stream applications. The contribution consists of three major parts, namely (1) a structured guidance for developers to design, implement, and evaluate data stream applications with a focus on data quality, (2) assistance for the structured evaluation of such applications, and finally (3) flexible, automated, domain- and task-independent data quality assessment throughout the whole process of data stream management. In the following, we will briefly describe how we reached these goals in the thesis.

Structured Development Process

A detailed research of data stream management principles, DSMS architectures, and corresponding query languages led to the identification of important characteristics and components of data stream applications and their development process. The thorough review of existing process models for information system and data mining application development revealed that none of them are suitable for data stream applications which demand quality assessment and structured evaluation. Based on the aforementioned knowledge we elaborated a quality-oriented process model tailored to data stream applications in Chapter 5. The process model comprises steps, and method proposals for the structured design, implementation, and evaluation of data stream applications with a special emphasis on data quality. Furthermore, we distinguished three different ways of approaching the data stream application development which are all covered by our process model. These influence the content and order of the steps in the process model. To evaluate the model and demonstrate the applicability, flexibility, and generalizability we utilized four case studies from actual research projects. We selected case studies from domains in which applications use real-time and sensor data as these are typical examples for data stream management settings, namely Connected Intelligent Transportation Systems (C-ITS) and Mobile Health (mHealth). For all case studies, we examine applying the process model (cf. Chapters 6 and 7). Although the domains, preconditions, and requirements for the case studies were quite different, we could cover most of the specifics and problems using the process model.

In conclusion, we contribute a general guidance for data stream application developers with which they can organize and document the design, implementation, and evaluation of data stream applications in a structured way.

Continuous Evaluation

The evaluation of a data stream application is a major and complex task. Parameters and conditions influencing the application results are manifold and have to be determined. We proposed a structured way to do this guided by appropriate tools. To this end, we first reviewed important characteristics and applications of the two selected domains C-ITS in Chapter 2 and mHealth applications in Chapter 3. This led to a deep understanding of important steps in the development and evaluation process of corresponding data stream applications. We proposed an evaluation framework, which can be instantiated for the application at hand. It guides the developer in selecting which and what kind of constituents may be useful for the implementation but also for the evaluation of the applications. We showed the implementation of the framework in the four case studies for C-ITS in Chapter 6 and for mHealth in Chapter 7. We found that due to the characteristics of data stream applications several iterations of evaluation in very different directions are required. The iterations are necessary to find out which parameters are influencing the results and how. For many of the parameters we could find optimal values and we reused these in the next evaluation iterations. Furthermore, it proved to be beneficial to test parameters one by one though some of them were connected to one another. An in-depth evaluation was done for algorithms specific for the domain of C-ITS. The improvement of application results by a devised online Map Matching algorithm was evaluated and shown in a structured way in Chapter 9. A further detailed evaluation was carried out for data stream mining algorithms, including solutions for the problems of class imbalance and concept drift in Chapter 10. The structured evaluation found the best combination of algorithms for the task at hand.

The combination of process model, evaluation framework, and data quality management framework contributes a powerful toolbox for developers to conduct structured and iterative evaluations of data stream applications.

Data Quality Management

The measurement of data quality throughout the whole application and data stream management process provides a powerful tool to identify problems and potential optimizations. To achieve this goal we first presented and analysis of important requirements for DQ management for data streams in Chapter 8. Based on this knowledge and a discussion of existing DQ management methodologies, we designed a methodology which provides the most important steps for DQ management for relational DSMSs. Along the methodology we designed a flexible, holistic DQ management framework. We showed its applicability to and usefulness for different domains and applications using the four case studies presented in Chapters 6 and 7. Previous approaches for DQ management in data streams either focused on system-related aspects or were very tightly integrated into the DSMS which hampers the extension of DQ management with new DQ metrics or DQ dimensions. Our approach provides flexibility by storing the DQ metadata in an ontology which is easily adaptable. Detailed evaluations for the case studies in Chapter 8 showed the usefulness of the approach to identify DQ issues.

We offer developers a powerful and flexible tool to define, measure, monitor, and improve data quality in their data stream applications independent of the domain or application.

11.2 Outlook

Though the work in this thesis has provided a substantial step towards the improvement of the design and evaluation processes for data stream applications, there are still things which are of interest for future research.

In its current manifestation, the proposed process model assumes basic standalone data stream management architectures. More complex data ecosystems for data analytics have evolved and DSMS are part of them (Ranjan, 2014; Dolas, 2015). An important research aspect would be to elaborate the suitability and adaptivity of the model to such ecosystems. The concepts of the Lambda architecture (Marz and Warren, 2015) or a combination of an architecture with Streaming Data Warehouses (Golab and Özsu, 2010) pose interesting challenges to the evaluation and data quality assessment of applications implemented in such a setting.

Process models are mostly evaluated by applying them. Someone has to use them for the targeted purpose and, in the end, rate whether it fulfills the expectations and requirements. We presented the application of the proposed process model in different case studies. Several process models for information systems, data mining, and data quality management either spawned directly from industry or were evaluated in several big industrial or research projects (Hauser and Clausing, 1988; Wirth and Hipp, 2000; Schroeder et al., 2008). To alleviate the evaluation of our process model to the next level, it needs to be evaluated by more developers from research and industry in more domains. Another form of evaluation could be to apply the process model to do a backwards analysis: the process model is applied to describe and evaluate existing architectures, such as for the probabilistic and quality-oriented architecture in (Kuka and Nicklas, 2014b). If all aspects of the architecture and implemented applications can be mapped by the process model, it is suitable to guide the design and evaluation of the applications.

A further open research challenge is the use of user interfaces for data quality modeling and implementation. We have designed a metadata model for the definition of data quality dimensions, metrics, and their relationship to data objects. The developer has to design the ontology manually in an ontology editor. This is comfortable, but requires some knowledge about ontologies. A more user-friendly way would be a dedicated data quality user interface. If possible, this should be integrated into a visual design tool for data stream applications as provided already by some DSMS, such as IBM InfoSphere[1]. In a drag-and-drop fashion dimensions could be attached to data streams and their attributes and metrics could be defined customly or selected from a library. In the background a corresponding metadata model would be updated to reflect the changes. A crucial and difficult aspect is to define the logic behind this as it has to consider correlations between attributes and operations and many special cases. The combination with a data quality dashboard similar to visualization tools for big data analytics (Zhang et al., 2012) would provide a powerful, but user-friendly toolbox

[1]`http://www-01.ibm.com/software/data/infosphere/streams`

for developers.

The challenge of developing effective, efficient, and high-quality data stream applications is a huge task which cannot fully be tackled by this thesis alone. However, we made a series of contributions to offer developers a tool set to guide them through the whole process of design, implementation, and evaluation of data stream applications emphasizing the importance of data quality in every step.

Part V

Appendix

Appendix A

Traffic Parameters & Data Sources

A.1 Traffic Parameters

In Table A.1, basic traffic parameters, i.e., parameters which directly can be used to describe a traffic situation and which correspond to a *single vehicle* are listed (Treiber and Kesting, 2010; Schnabel and Lohse, 1997; Hoyer, 2003; Steinauer et al., 2006; Mensebach, 2004).

Table A.1: Basic Traffic Parameters for Individual Cars

Parameter name	Description	Unit
Displacement	Way from the starting point to the current point (the objects overall change in position). Therefore, it is a vector quantity.	m or km
Distance	Total distance between two points on the road (how much ground has an object covered). Therefore, it is a scalar quantity.	m or km
Speed	The rate at which an object covers distance (scalar quantity).	m/s or km/h
Velocity	Displacement per time unit (the rate at which an object changes its position). In contrast to speed, velocity includes a direction and is therefore a vector quantity. The direction is simply the one in which the object is moving.	m/s or km/h
Average Velocity	Total displacement of an object in an interval divided by time.	m/s or km/h
Acceleration	Change of velocity per time unit.	m/s^2
Jar	Change of acceleration per time unit.	m/s^3
Vehicle Type	Indicates the kind of vehicle passing by a gauging section.	Enumeration of vehicle types
Occupancy Time	The time a car occupies a detector.	s

Table A.1: Basic Traffic Parameters for Individual Cars

Parameter name	Description	Unit
Presence	Detects if a car is present or not.	Boolean
Travel time	The time required by a vehicle to travel from a start point to an end point including the halt times.	h
Travel velocity	The average velocity of a vehicle travelling from a start point to an end point.	km/h
Halt time	The time a vehicle halts travelling from a start to an end point.	h

Further parameters according to an individual car which might be of interest for different traffic applications are measures describing the braking behaviour of a car, such as emergency braking or detection of a hidden queue-end. In the literature, mainly Service Braking (average brake delay of up to 3 m/s^2), i.e., braking in common situations such as at traffic lights, and Emergency Braking (average brake delay of up to 6 m/s^2), i.e., in case of a critical situation are distinguished (Mensebach, 2004). In general, the time until a car stops consists mainly of the reaction time and the braking time. Furthermore, the braking time includes the response time, the time until full braking pressure is reached and the effect time. There may be also some delay caused by environmental circumstances, such as black ice, which has to be added to the braking time.

In Table A.2 important parameters according to traffic flows (in contrast to individual vehicles) are explained. The measures are classified according to their spatial reference, i.e., if the measurement is made for a single position (local) or a section (section-based).

Table A.2: Basic Traffic Parameters (Traffic Flows)

Parameter Name	Description	Unit	Local / Section-based
Traffic Volume (Intensity)	The quotient of the number of vehicles passing the gauging section and the time span T. This can be additionally divided in the volume of cars and volume of trucks. (Hoyer, 2003).	vehicles/h or vehicles/s	L
Traffic Density (Concentration)	Quotient of the number of vehicles and the distance.	vehicles/km	S
Local velocity	Velocity measured at a certain gauging section in an interval.	m/s or km/h	L

Table A.2: Basic Traffic Parameters (Traffic Flows)

Parameter Name	Description	Unit	Local / Section-based
Average local velocity	The average velocity of all vehicles in a time interval measured at a certain gauging section. This can be additionally divided in the average velocity of cars and average velocity of trucks. (Hoyer, 2003).	m/s or km/h	L
Momentary Velocity	Velocity of all vehicles at a fixed point in time on a fixed road section.	m/s or km/h	S
Net Time Gap	Time difference between two cars driving one after another passing the same gauging section. The time is measured between the back end of the first car and the front end of the second car.	s	L
Gross Time Gap	Time difference between two cars driving one after another passing the same gauging section. The time is measured between the front end of the first car and the front end of the second car.	s	L
Net Distance Gap	Distance between two cars driving one after another passing the same gauging section. The distance is measured between the back end of the first car and the front end of the second car.	m	L
Gross Distance Gap	Distance between two cars driving one after another passing the same gauging section. The distance is measured between the front end of the first car and the front end of the second car.	m	L
Traffic Capacity	—	$\#\text{vehicles} \cdot (\text{km/h})^2$	S
Average Travel Time	Is the average travel time of n vehicles travelling from a start point to an end point.	min	S
Time to Collision (TTC)	An indicator for traffic safety. Calculated from the distance and the speeds of two vehicles driving in a row.	Net distance/Speed difference	S

A.2 Stationary Detection Devices

Table A.3: Stationary Detection Mechanisms

Name	Description	Parameters
Induction Loops	Depending on the setup they are used at traffic signals for demand actuation, at parking garages for counting and on highways for vehicle detection, speed measurements and determination of the vehicle type. Actually, they are the most common mean to measure local traffic data on urban roads and on highways (Hoyer, 2003; Treiber and Kesting, 2010). Induction loops can be installed under or evaporated onto the surface in different setups. Depending on the application one-loop, two-loop and three-loop setups can be distinguished.	Presence, counting, speed, occupancy time, presence, traffic volume, vehicle type, time gap
Laser / LiDAR sensor	A sender sends out light pulses which are reflected by the vehicle and sent back to a receiver. A processor analyses the differences between the sent out signal and the received signal.	Traffic volume, speed, vehicle type, time gap, occupancy time, presence
Radar	Sends out electromagnetic waves and measures the differences in the reflected signals (Doppler effect).	Traffic volume, speed, vehicle type, time gap, occupancy
Light barrier / Light sensors	Light barriers detect when a vehicle interrupts the light beam sent out by the device. Light sensors measure the difference in light, when a vehicle passes the sensors.	traffic volume, speed
Microwave Systems	Microwave sensors send out waves to a certain road area with constantly changing frequencies. Signals are reflected back to the sensor and vehicles are detected.	traffic volume, occupancy, average speed, and vehicle type.
Ultrasonic Systems	Emit sound waves very quickly and are reflected by the vehicles to a sensor. The difference of the signal is measured (Doppler effect).	Traffic volume, speed, vehicle type, time gap, occupancy, presence.
Infrared Systems	Passive and active systems.	traffic volume, speed, vehicle type, time gap, occupancy, presence

Table A.3: Stationary Detection Mechanisms

Name	Description	Parameters
Video Image Processors	Video detection via video cameras and infrared cameras	Traffic volume, speed, vehicle type, time gap, occupancy time, presence, number plate recognition, trajectories, traffic volume, lane changes.
Pneumatic road tubes	Similar to induction loops. One or more tubes are fixed with mastic tape and webbing onto the road surface crossing one or more lanes. When vehicles drive over them, the pressure change indicates an event (McGowen and Sanderson, 2011).	Counting, speed, traffic volume, vehicle type
Pressure sensors	Piezo or fibre sensors in the road surface to detect vehicles.	Speed, traffic volume, vehicle type, presence, occupancy, time gap

Appendix B

Data Source Documentation

B.1 General Description

Authors:
Version:
Date:

Table B.1: General Part of the Data Source Documentation

Property	**Value**	**Description**
Functional Requirements		
Name		The name of the data source.
Version		The version of the data source.
Origin		Describes where the data came from and who owns it.
Format / data model		Either a standardized format or a custom one. If custom, please use the format description field for further information.
Format description		A description of the custom format.
Update Frequency		Describes how often data items in the data source are updated. If this is done frequently, this should contain a number, e.g., 1 MHz or 100/ms. If data is produced infrequently, then the estimated minimum and maximum period between two updates should be documented.
Schema		Describes if a schema is used (unstructured, semi-structured, structured) and if a schema is used, to which standard it complies. The schema itself should be described in the later part of this documentation.
Size		Describes the estimated size of single items (e.g., a tuple or object) or the whole data set.

Table B.1: General Part of the Data Source Documentation

Property	Value	Description
Costs and Availability		The costs for retrieving the data, i.e., data provider costs, communication costs. Is the data available all the time or is the access limited?
Communication requirements		Is the source online? Which protocol is used to retrieve data? Which interfaces? Is there a delay introduced?
Non-functional Requirements		
Time Range		If for the data in the data source a temporal aspect is important, the period in which the data has been recorded has to be documented.
Spatial extension		Documentation of the geographical area or spatial extension of the recorded data, e.g., Munich, Germany, a living room of 2m x 2m.
Timestamps		Describes if data is timestamped and which field(s) contains the actual timestamp. If applicable, different types of timestamps must be distinguished, e.g., the creation timestamp, last update timestamp and so on.
Purpose /Description		Overall description of the contents of the data source.
Models		Some data sources and are created based on specific models, such as mathematical or physical models. This especially the case for simulation data and data produced in data mining. The process of creating the data should be very clear to rate the data at hand.
Specifics		Are there any data transformations needed to make value of the data? Is implicit knowledge included? If so, which knowledge?

B.2 Schema Description

This section is to be filled for all structured and semi-structured data sources which have schema (ideally) created with a standardized schema description language (e.g., SQL, DTD, RDF, XML Schema, OWL). A correct and comprehensive description of a schema depends very much on the type of the schema. In general, we recommend to describe the schema along logically separated units, such as tables or XML nodes. Additionally and if applicable, the relationships between units must be described. Depending on the schema type a diagram is helpful to depict units and relationships. Important aspects for the later use of the data are the properties of the single data items (e.g., fields in tables, XML properties) which contain the actual data.

These also have mainly functional and semantic aspects which should be described. For each field the following properties could be useful, but might also vary depending on the data source and its format:

Table B.2: Schema Description of Data Source Documentation

Property	Value	Description
Name		The name of the data field.
Description		The semantics of the field should be described.
Data Type		The data type of the values in the field.
Numeric precision		If numeric data is contained, what is the precision?
Value Range		In which range does the data lie? What are possible values?
Granularity		Has the data been aggregated? And if so, how?
Frequency		How often is this data updated?
Geospatial Information		If it is a geospatial field, what is the system used to define it (e.g., GALILEO, GPS)?
Temporal Information		Is this a timestamp and what is its precision?
Optional		Boolean value indicating, if the field has to contain a value. The value can be yes or no.

B.3 Case Study Queue End Detection

General part:

Property	Value
Name	**actionID**
Description	A unique identifier for each message.
Data Type	A string with letters and digits.
Frequency	The value is unique for each object.
Optional	No

Property	Value
Name	**cell_id**
Description	A unique identifier for a mobile network cell.
Data Type	Integer
Optional	Yes

Property	Value
Name	**ts**
Description	The timestamp of the object.
Data Type	Integer
Temporal Information	Seconds since 1.1.1970 UTC
Optional	No

Property	**Value**
Name	**ttl**
Description	Time to live.
Data Type	Integer
Temporal Information	In seconds.
Optional	No

Property	**Value**
Name	**wgs84**
Description	WGS84 coordinates as separate JSON-Object
Data Type	Object
Geospatial Information	GPS coordinate according to the WGS84 system. Consists of latitude (lat), longitude (long), and elevation (elev)
Optional	No

Property	**Value**
Name	**heading**
Description	The direction of the vehicle in Grad.
Data Type	Float
Numeric Precision	Two decimal digits.
Optional	No

Property	**Value**
Name	**speed**
Description	The current speed of the vehicle in m/s.
Data Type	Float
Numeric Precision	Two decimal digits.
Optional	No

Property	**Value**
Name	**accel**
Description	Acceleration of the vehicle in m/s^2.
Data Type	Float
Numeric Precision	One decimal digits.
Optional	Yes

EBL Part

Property	**Value**
Name	**vtype**
Description	The vehicle type.
Data Type	String
Value Range	car, truck, motorcycle
Optional	No

Property	Value
Name	**tsms**
Description	Milliseconds as addition to the timestamp.
Data Type	Integer
Temporal Information	In milliseconds.
Optional	Yes

Property	Value
Name	**L**
Description	The length of the vehicle in cm.
Data Type	Integer

Property	Value
Name	**W**
Description	The width of the vehicle in cm.
Data Type	Integer
Optional	Yes

Property	Value
Name	**H**
Description	The height of the vehicle in cm.
Data Type	Integer
Optional	Yes

Property	Value
Name	**offL**
Description	GPS antenna length offset from front in cm.
Data Type	Integer
Optional	Yes

Property	Value
Name	**offW**
Description	GPS antenna width offset from left in cm.
Data Type	Integer
Optional	Yes

EVW Part

Property	Value
Name	**lightbar**
Description	Status of the light bars of the vehicle
Data Type	String
Value Range	not equipped,disabled,enabled,engaged
Optional	No

Property	Value
Name	**sirene**
Description	Status of the sirene
Data Type	String
Value Range	not equipped,disabled,enabled,engaged
Optional	No

Property	Value
Name	**restype**
Description	Type of event.
Data Type	String
Value Range	unknown,special right of way, non-emergency, pursuit, safeguard danger, on work, blocking lane, blocking direction, slow driving, hindering passing, convoy
Optional	Yes

WLA Part

Property	Value
Name	**lightbar**
Description	Status of the light bars of the vehicle
Data Type	String
Value Range	not equipped,disabled,enabled,engaged
Optional	No

Bibliography

Abadi, D. J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., and Zdonik, S. B. (2005). The Design of the Borealis Stream Processing Engine. In Stonebraker, M., Weikum, G., and DeWitt, D., editors, *Proc. 2nd Biennal Conference on Innovative Data Systems Research (CIDR)*, pages 277–289, Asilomar, CA, USA.

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Erwin, C., Galvez, E. F., Hatoun, M., Maskey, A., Rasin, A., Singer, A., Stonebraker, M., Tatbul, N., Xing, Y., Yan, R., and Zdonik, S. B. (2003a). Aurora: a data stream management system. In (Halevy et al., 2003), pages 666–666.

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, S. B. (2003b). Aurora: a new model and architecture for data stream management. *VLDB Journal*, 12(2):120–139.

Aberer, K., Hauswirth, M., and Salehi, A. (2006). A middleware for fast and flexible sensor network deployment. In (Dayal et al., 2006), pages 1199–1202.

Aggarwal, C. C., editor (2013). *Managing and Mining Sensor Data.* Springer Science & Business Media.

Aggarwal, C. C. (2015a). A Survey of Stream Classification Algorithms. In (Aggarwal, 2015c), chapter 9, pages 245–273.

Aggarwal, C. C. (2015b). An Introduction to Data Classification. In (Aggarwal, 2015c), chapter 1, pages 1–36.

Aggarwal, C. C., editor (2015c). *Data Classification - Algorithms and Applications.* Taylor & Francis.

Aggarwal, C. C. and Yu, P. S. (2008). A framework for clustering uncertain data streams. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE)*, pages 150–159.

Ahmad, Y. and Çetintemel, U. (2009). Data Stream Management Architectures and Prototypes. In (Liu and Özsu, 2009), chapter D, pages 639–643.

Ahmed, S. A., Ariffin, S. H., and Fisal, N. (2013). Overview of wireless access in vehicular environment (WAVE) protocols and standards. *Indian Journal of Science and Technology*, 6(7):4994–5001.

Al-Kateb, M., Lee, B. S., and Wang, X. S. (2007). Adaptive-size reservoir sampling over data streams. In *Proceedings of the IEEE 19th International Conference on Scientific and Statistical Database Management (SSBDM)*, pages 22–22.

Alessandri, A., Bolla, R., and Repetto, M. (2003). Estimation of Freeway Traffic Variables Using Information from Mobile Phones. In *Proceedings of the 2003 American Control Conference*, volume 5, pages 4089–4094. IEEE.

Ali, M., Krumm, J., Rautman, T., and Teredesai, A. (2012). ACM SIGSPATIAL GIS Cup 2012. In (Cruz et al., 2012), pages 597–600.

Angel, A., Sarkas, N., Koudas, N., and Srivastava, D. (2012). Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *Proceedings of the VLDB Endowment*, 5(6):574–585.

Angelov, P. and Zhou, X. (2008). Online learning fuzzy rule-based system structure from data streams. In *Proceedings of the IEEE 16th International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 915–922.

Araniti, G., Campolo, C., Condoluci, M., Iera, A., and Molinaro, A. (2013). LTE for Vehicular Networking: A Survey. *Communications Magazine, IEEE*, 51(5):148–157.

Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J. (2004a). STREAM: The Stanford Data Stream Management System. Technical report, Stanford InfoLab. Available at: http://ilpubs.stanford.edu:8090/64.

Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Nishizawa, I., Rosenstein, J., and Widom, J. (2003a). STREAM: the stanford stream data manager (demonstration description). In (Halevy et al., 2003), page 665.

Arasu, A., Babu, S., and Widom, J. (2003b). An Abstract Semantics and Concrete Language for Continuous Queries over Streams and Relations. In *Proceedings of the 9th International Conference on Data Base Programming Languages*, pages 1–11.

Arasu, A., Babu, S., and Widom, J. (2006). The CQL continuous query language: semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142.

Arasu, A., Cherniack, M., Galvez, E., Maier, D., Maskey, A., Ryvkina, E., Stonebraker, M., and Tibbetts, R. (2004b). Linear road: a stream data management benchmark. In (Nascimento et al., 2004), pages 480–491.

Artikis, A., Weidlich, M., Schnitzler, F., Boutsis, I., Liebig, T., Piatkowski, N., Bockermann, C., Morik, K., Kalogeraki, V., Marecek, J., et al. (2014). Heterogeneous Stream Processing and Crowdsourcing for Urban Traffic Management. In *Proceedings of the 17th International Conference on Extending Database Technology (EDBT)*, pages 712–723.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002a). Models and Issues in Data Stream Systems. In Popa, L., editor, *Proc. 21st ACM Symposium on Principles of Database Systems (PODS)*, pages 1–16, Madison, Wisconsin.

Babcock, B., Datar, M., and Motwani, R. (2002b). Sampling from a moving window over streaming data. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 633–634.

Babu, S., Srivastava, U., and Widom, J. (2004). Exploiting k-constraints to reduce memory overhead in continuous queries over data streams. *ACM Transactions on Database Systems (TODS)*, 29(3):545–580.

Babu, S. and Widom, J. (2001). Continuous Queries Over Data Streams. *SIGMOD Record*, 30(3):109–119.

Bai, Y., Luo, R. C., Thakkar, H., Wang, H., and Zaniolo, C. (2004). An Introduction to the Expressive Stream Language (ESL). Technical report, WEB Information System Laboratory, UCLA, CS Department.

Balzert, H. (2010). *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Springer-Verlag. In German.

Bando, M., Hasebe, K., Nakanishi, K., and Nakayama, A. (1998). Analysis of optimal velocity model with explicit delay. *Physical Review E*, 58(5):5429.

Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., and Grossniklaus, M. (2009). C-SPARQL: SPARQL for Continuous Querying. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 1061–1062. ACM.

Barker, R. (1990). *CASE Method: Tasks and Deliverables*. Addison-Wesley Longman Publishing Co., Inc.

Bashshur, R., Shannon, G., Krupinski, E., and Grigsby, J. (2011). The taxonomy of telemedicine. *Telemedicine and e-Health*, 17(6):484–494.

Basili, V. (1992). Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Technical Report UMIACS-TR-92-96, University of Maryland, Department of Computer Science.

Basili, V. and Rombach, H. (1988). The TAME project: Towards Improvement-oriented Software Environments. *Software Engineering, IEEE Transactions on*, 14(6):758–773.

BASt (1999). *Merkblatt für die Ausstattung von Verkehrsrechnerzentralen und Unterzentralen (MARZ)*. Bundesanstalt für Straßenwesen. (in German).

Basyoni, Y. and Talaat, H. (2015). A Bilevel Traffic Data Extraction Procedure via Cellular Phone Network for Intercity Travel. *Journal of Intelligent Transportation Systems*, 19(3):289–303.

Batini, C., Cappiello, C., Francalanci, C., and Maurino, A. (2009). Methodologies for data quality assessment and improvement. *ACM Computing Surveys (CSUR)*, 41(3):16.

Batini, C. and Scannapieco, M. (2006). *Data quality: concepts, methodologies and techniques*, chapter Methodologies for Data Quality Measurement and Improvement, pages 161–200. Springer.

Batista, G., Prati, R., and Monard, M. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29.

Baumgartner, N., Gottesheim, W., Mitsch, S., Retschitzegger, W., and Schwinger, W. (2010). Improving Situation Awareness In Traffic Management. In *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*.

beck et al. projects GmbH (2006). VRZ3 Kernsoftware Tutorial. Technical report, beck et al. projects GmbH.

Beggs, R. (2015). 5 Reasons why Spark Streaming's Batch Processing of Data Streams is not Stream Processing. SQLStream Blog. http://sqlstream.com/2015/03/5-reasons-why-spark-streamings-batch-processing-of-data-streams-is-not-stream-processing. Date accessed: 28.10.2016.

Behrens, G., Kuz, V., and Behrens, R. (2010). *Softwareentwicklung von Telematikdiensten: Konzepte, Entwicklung und zukünftige Trends*, chapter Kurze Protokollübersicht, pages 32–45. Springer-Verlag.

Bellomo, N. and Dogbe, C. (2011). On the Modeling of Traffic and Crowds: A Survey of Models, Speculations, and Perspectives. *SIAM review*, 53(3):409–463.

Benbasat, I., Goldstein, D. K., and Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS quarterly*, pages 369–386.

Bi, J., Chang, C., and Fan, Y. (2013). Particle Filter for Estimating Freeway Traffic State in Beijing. *Mathematical Problems in Engineering*, 2013:1–7.

Biem, A., Bouillet, E., Feng, H., Ranganathan, A., Riabov, A., Verscheure, O., Koutsopoulos, H. N., and Moran, C. (2010). IBM InfoSphere Streams for Scalable, Real-Time, Intelligent Transportation Services. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 1093–1104.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 7th SIAM International Conference on Data Mining*, pages 443–448.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010a). MOA: Massive Online Analysis. *Journal of Machine Learning Research*, 11(May):1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., and Seidl, T. (2010b). MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. In *Proceedings of the 2nd International Workshop on Handling Concept Drift in Adaptive Information Systems (HaCDAIS)*, pages 3–16.

Bifet, A. and Kirkby, R. (2009a). *Data Stream Mining - A Practical Approach.* University of Waikato.

Bifet, A. and Kirkby, R. (2009b). *Massive Online Analysis - Manual.* The University of Waikato.

Bogenberger, K. and Dinkel, A. (2009). Sicherheitswirkungen von Verkehrsinformationen. *Berichte der Bundesanstalt für Straßenwesen*, F84:1–79.

Bolles, A. (2009). A flexible framework for multisensor data fusion using data stream management technologies. In *Proceedings of the 12th International Conference of Extending Database Technology (EDBT) &17th International Conference on Database Theory (ICDT), Workshops*, pages 193–200. ACM.

Botan, I., Kossmann, D., Fischer, P., Kraska, T., Florescu, D., and Tamosevicius, R. (2007). Extending XQuery with window functions. In *Proceedings of the 33rd International Conference on Very Large DataBases (VLDB)*, pages 75–86. VLDB Endowment.

Brettlecker, G. and Schuldt, H. (2007). The OSIRIS-SE (stream-enabled) Infrastructure for Reliable Data Stream Management on Mobile Devices. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1097–1099. ACM.

Brodie, M. L. and Duggan, J. (2014a). Big Data is Opening the Door to Revolutions: Databases should be next. ACM SIGMOD Blog.

Brodie, M. L. and Duggan, J. (2014b). What versus Why. Towards Computing Reality. ODBMS.org: Big Data and Analytical Data Platforms - Articles / Expert Articles.

Brüggemann, S. and Grüning, F. (2008). Using domain knowledge provided by ontologies for improving data quality management. In *Proceedings of the i-KNOW Conference 2008*, pages 251–258.

Caceres, N., Romero, L., Benitez, F., and del Castillo, J. (2012). Traffic Flow Estimation Models Using Cellular Phone Data. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1430–1441.

Caceres, N., Wideberg, J., and Benitez, F. (2008). Review of traffic data estimations extracted from cellular networks. *Intelligent Transport Systems, IET*, 2(3):179–192.

Campbell, J. L., Rustad, L. E., Porter, J. H., Taylor, J. R., Dereszynski, E. W., Shanley, J. B., Gries, C., Henshaw, D. L., Martin, M. E., Sheldon, W. M., et al. (2013). Quantity is Nothing without Quality: Automated QA/QC for Streaming Environmental Sensor Data. *BioScience*, 63(7):574–585.

Canaud, M., Mihaylova, L., Sau, J., and El Faouzi, N.-E. (2013). Probabilty hypothesis density filtering for real-time traffic state estimation and prediction. *Networks and Heterogeneous Media (NHM)*, 8(3):825–842.

Carney, D., Centintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., and Zdonik, S. (2002). Monitoring Streams - A New Class of Data Management Applications. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, pages 215–226, Hong Kong, China. VLDB Endowment, Morgan Kaufmann.

Castro Fernandez, R., Migliavacca, M., Kalyvianaki, E., and Pietzuch, P. (2013). Integrating scale out and fault tolerance in stream processing using operator state management. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 725–736. ACM.

Chan, T. N., Lee, C., and Hellinga, B. (2003). Real-time Identification and Tracking of Traffic Queues Based on Average Link Speed. In *Proceedings of the Transportation Research Board 82nd Annual Meeting.*

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., and Shah, M. A. (2003). TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In

Proc. 1st Biennal Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (2000). CRISP-DM 1.0 Step-by-step Data Mining Guide. Technical report, CRISP Consortium.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Chen, C.-M., Agrawal, H., Cochinwala, M., and Rosenbluth, D. (2004). Stream query processing for healthcare bio-sensor applications. In *Proceedings of the 20th International Conference on Data Engineering (ICDE)*, pages 791–794. IEEE.

Chen, J., DeWitt, D. J., Tian, F., and Wang, Y. (2000). NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In Chen, W., Naughton, J. F., and Bernstein, P. A., editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 379–390, Dallas, Texas. ACM.

Chen, L.-W., Peng, Y.-H., and Tseng, Y.-C. (2011a). An infrastructure-less framework for preventing rear-end collisions by vehicular sensor networks. *IEEE Communications Letters*, 15(3):358–360.

Chen, M., Gonzalez, S., Vasilakos, A., Cao, H., and Leung, V. (2011b). Body Area Networks: A Survey. *Mobile networks and applications*, 16(2):171–193.

Chen, Y. (2009). Data Quality Management for Traffic Monitoring Systems based on Wireless Location Technology. Master's thesis, RWTH Aachen University.

Cheng, J., Ke, Y., and Ng, W. (2008). A survey on algorithms for mining frequent itemsets over data streams. *Knowledge and Information Systems*, 16(1):1–27.

Cherniack, M. and Zdonik, S. (2009). Stream-Oriented Query Languages and Architectures. In (Liu and Özsu, 2009), chapter S, pages 2848–2854.

Claßen, E. (2013). Evaluation of sampling techniques for data stream classification. Master's thesis, RWTH Aachen University.

Crowley-Nicol, C., Heitzer, C., Preusser, P., Reifenberg, M., Thiele, S., Weiß, M., and Werscheid, S. (2015). Jahresbericht 2014. Technical report, Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen.

Cruz, I. F., Knoblock, C., Kröger, P., Tanin, E., and Widmayer, P., editors (2012). *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL GIS)*, Rodondo Beach, CA, USA.

Cugola, G. and Margara, A. (2012). Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Computing Surveys (CSUR)*, 44(3):15.

Dalgleish, M. and Hoose, N. (2008). *Highway traffic monitoring and data quality.* Artech House Publishers.

Datta, G., Graham, M., Botts, N., Rhodes, H., and Owens, M. (2016). HL7 Mobile Health Projects Overview. Technical report, HL7 Mobile Health Work Group.

Dayal, U., Whang, K.-Y., Lomet, D. B., Alonso, G., Lohman, G. M., Kersten, M. L., Cha, S. K., and Kim, Y.-K., editors (2006). *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*. ACM Press.

ddg - Gesellschaft für Verkehrsdaten mbH (2000). Von der Meßwerterfassung bis zur automatisch generierten Verkehrsmeldung. Technical report, ddg - Gesellschaft für Verkehrsdaten mbH.

Demers, A., Gehrke, J., Hong, M., Riedewald, M., and White, W. (2005). A General Algebra and Implementation for Monitoring Event Streams. Technical report, Cornell University.

Dirscherl, H.-C. (2016). Diese Verkehrslage-Dienste gibt es für Autofahrer. PC-Welt Ratgeber & Tipps. http://www.pcwelt.de/ratgeber/Stau-Warnung-Google-Maps-Tomtom-Verkehrslage-Echtzeitverkehrsinformationen-373385.html. Date accessed: 28.10.2016.

Dixon, J. (2010). Pentaho, Hadoop, and Data Lakes. James Dixon's Blog. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes. Date accessed: 28.10.2016.

Dixon, J. (2014). Data Lakes Revisited. James Dixon's Blog. https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited. Date accessed: 28.10.2016.

Dolas, S. (2015). Design patterns for real time streaming data analytics. In *Proceedings of the 2015 Strata + Hadoop World*.

Domingos, P. (1999). Metacost: A General Method for Making Classifiers Cost-sensitive. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 155–164. ACM.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 71–80. ACM.

Domingos, P. and Hulten, G. (2003). A General Framework for Mining Massive Data Streams. *Journal of Computational and Graphical Statistics*, 12(4):945–949.

Durth, W. and Hanke, H. (2004). *Handbuch Straßenwinterdienst*.

Durth, W., Hanke, H., and Levin, C. (1989). Wirksamkeit des Straßenwinterdienstes auf die Verkehrssicherheit und die Wirtschaftlichkeit des Verkehrsablaufes. *Forschung Strassenbau und Strassenverkehrstechnik*, (550).

Economist Intelligence Unit (2012). Emerging mHealth - Paths for Growth. Technical report, PwC.

Elmasri, R. A. and Navathe, S. B. (2004). *Fundamentals of Database Systems*. Addison-Wesley, 4th edition.

Eren, H. (2014). Measurements, Instrumentation, and Sensors. volume 1, chapter 1. CRC press.

Espina, J., Falck, T., Panousopoulou, A., Schmitt, L., Mülhens, O., and Yang, G.-Z. (2014). *Network Topologies, Communication Protocols, and Standards*, chapter 5, pages 189–236. In (Yang, 2014).

ETSI EN 302 637-2 (2014). Intelligent Transport Systems (ITS); Specification of Cooperative Awareness Basic Service.

ETSI EN 302 637-3 (2014). Intelligent Transport Systems (ITS); Specifications of Decentralized Environmental Notification Basic Service.

ETSI EN 302 665 (2010). Intelligent Transport Systems (ITS); Communications Architecture.

ETSI TS 101 539-1 (2013). Intelligent Transport Systems (ITS); V2X Applications; Part 1: Road Hazard Signalling (RHS) Application Requirements Specification.

ETSI TS 101 539-3 (2013). Intelligent Transport Systems (ITS); V2X Applications; Part 3: Longitudinal Collision Risk Warning (LCRW) Application Requirements Specification.

ETSI TS 102 637-1 (2010). Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements.

Etzion, O. and Niblett, P. (2011). *Event Processing in Action.* Manning Publications Co.

European Comission (2014). Green Paper on mobile Health (mHealth). Technical report, European Comission.

European Comission (2015). Digital Agenda for Europe - A Europe 2020 Initiative - mHealth. https://ec.europa.eu/digital-agenda/en/mhealth. Date accessed: 28.10.2016.

European Comission (2016). C-ITS Platform. Technical report, European Comission.

Fan, W. and Geerts, F. (2012). Foundations of data quality management. *Synthesis Lectures on Data Management*, 4(5):1–217.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, 39(11):27–34.

Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., and Zhang, J. (2005). On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2):207–216.

Fiege, G., Gasser, T., Gehlen, G., Geisler, S., Jodlauk, G., Phan, M.-A., Quix, C., Rembarz, R., Wiecker, M., and Westhoff, D. (2011). ITS Services and Communication Architecture. Deliverable D03, Cooperative Cars eXtended.

Finkelstein, C. (2006). Information engineering methodology. In *Handbook on architectures of information systems*, pages 459–483. Springer.

Fiordelli, M., Diviani, N., and Schulz, P. J. (2013). Mapping mHealth Research: A Decade of Evolution. *Journal of medical Internet research*, 15(5).

Fiscato, M., Vu, Q. H., and Pietzuch, P. (2009). A quality-centric data model for distributed stream management systems. In *Proceedings of the 7th International Workshop on Quality in Databases (QDB)*, pages 1–10.

Fontaine, M. D. and Smith, B. L. (2007). Investigation of the Performance of Wireless Location Technology-Based Traffic Monitoring Systems. *Journal of Transportation Engineering*, 133:157–165.

Gabbay, D. M. and Guenthner, F. (2002). *Handbook of philosophical logic*. Springer.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with Drift Detection. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA)*, page 286. Springer.

Gama, J. and Rodrigues, P. (2009). An Overview on Mining Data Streams. *Foundations of Computational Intelligence*, 6:29–45.

Gama, J., Sebastião, R., and Rodrigues, P. P. (2009). Issues in evaluation of stream learning algorithms. In Elder IV, J. F., Fogelman-Soulié, F., Flach, P. A., and Zaki, M. J., editors, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 329–338. ACM.

Garcia-Molina, H., Ullman, J. D., and Widom, J. (2009). Extended Operators of Relational Algebra. In *Database Systems: The Complete Book*, pages 213–222. Prentice Hall Press, Upper Saddle River, NJ, USA, 2 edition.

Gedik, B., Andrade, H., Wu, K.-L., Yu, P. S., and Doo, M. (2008). SPADE: The System S Declarative Stream Processing Engine. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1123–1134, New York, NY, USA. ACM.

Geisler, S. (2013). Data Stream Management Systems. In Kolaitis, P. G., Lenzerini, M., and Schweikardt, N., editors, *Data Exchange, Integration, and Streams*, volume 5 of *Dagstuhl Follow-Ups*, pages 275–304. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Geisler, S., Chen, Y., Quix, C., and Gehlen, G. (2010). Accuracy Assessment for Traffic Information Derived from Floating Phone Data. In *Proceedings of the 17th World Congress on Intelligent Transportation Systems and Services (ITS)*, Busan.

Geisler, S. and Quix, C. (2012). Evaluation of Real-time Traffic Applications based on Data Stream Mining. In Cervone, G., Lin, J., and Waters, N., editors, *Data Mining for Geoinformatics: Methods and Applications*, number 83-103. Springer.

Geisler, S., Quix, C., Schiffer, S., and Jarke, M. (2012). An Evaluation Framework for Traffic Information Systems Based on Data Streams. *Transportation Research Part C*, 23:29–55.

Geisler, S., Quix, C., Weber, S., and Jarke, M. (2016). Ontology-Based Data Quality Management for Data Streams. *ACM Journal of Data and Information Quality*, 7(4):1–34.

Geisler, S., Weber, S., and Quix, C. (2011). An Ontology-based Data Quality Framework for Data Stream Applications. In *Proceedings of the 16th International Conference on Information Quality (ICIQ)*, pages 145–159, Adelaide, Australia.

Gemulla, R., Lehner, W., and Haas, P. J. (2006). A Dip in the Reservoir: Maintaining Sample Synopses of Evolving Datasets. In (Dayal et al., 2006), pages 595–606.

Ghanem, T., Aref, W., and Elmagarmid, A. (2006). Exploiting predicate-window semantics over data streams. *ACM SIGMOD Record*, 35(1):3–8.

Gipps, P. G. (1986). A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414.

Glavic, B., Sheykh Esmaili, K., Fischer, P. M., and Tatbul, N. (2013). Ariadne: Managing Fine-grained Provenance on Data Streams. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, pages 39–50. ACM.

Global Observatory for eHealth (2011). mHealth - New Horizons for Health through Mobile Technologies. Technical report, World Health Organization (WHO).

Godfrey, A. B. (1999). Total Quality Management. In (Juran and Godfrey, 1999a), chapter 14, pages 1–35.

Goh, C., Dauwels, J., Mitrovic, N., Asif, M., Oran, A., and Jaillet, P. (2012). Online map-matching based on hidden markov model for real-time traffic sensing applications. In *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 776–781. IEEE.

Golab, L. and Özsu, M. T. (2010). *Data Stream Management.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Greenfeld, J. S. (2002). Matching GPS Observations to Locations on a Digital Map. In *Proceedings of the Transportation Research Board 81st Annual Meeting.*

Gundlegard, D. and Karlsson, J. (2009). Handover location accuracy for travel time estimation in gsm and umts. *Intelligent Transport Systems, IET*, 3(1):87–94.

Gurevich, Y., Leinders, D., and Van den Bussche, J. (2007). A theory of stream queries. In *Proceedings of the 11th International Conference on Database Programming Languages*, pages 153–168. Springer-Verlag.

Habtie, A. B., Abraham, A., and Midekso, D. (2015). Cellular Network Based Real-Time Urban Road Traffic State Estimation Framework Using Neural Network Model Estimation. In *Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 38–44. IEEE.

Haghighi, P. D., Zaslavsky, A., Krishnaswamy, S., Gaber, M. M., and Loke, S. (2009). Context-aware adaptive data stream mining. *Intelligent Data Analysis*, 13(3):423–434.

Halevy, A. Y., Ives, Z. G., and Doan, A., editors (2003). *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, San Diego, California, USA. ACM.

Hall, D. L. and Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Han, J. and Kamber, M. (2006). Classification and Predicition. In *Data mining: concepts and techniques*, pages 285–382. Morgan Kaufmann.

Härri, J., Filali, F., Bonnet, C., and Fiore, M. (2006). VanetMobiSim: Generating Realistic Mobility Patterns for VANETs. In *Proceedings of the 3rd International Workshop on Vehicular Ad-hoc Networks*, pages 96–97. ACM.

Hashemi, S., Yang, Y., Mirzamomen, Z., and Kangavari, M. (2009). Adapted One-versus-All Decision Trees for Data Stream Classification. *IEEE Transactions On Knowledge and Data Engineering*, 21(5):624–637.

Hassani, M. and Seidl, T. (2012). Resource-aware distributed clustering of drifting sensor data streams. In *Proceedings of the 4th International Conference on Networked Digital Technologies (NDT)*, pages 592–607. Springer.

Hauser, J. R. and Clausing, D. (1988). The House of Quality. *HARVARD BUSINESS REVIEW*.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In *Proceedings of the 21st IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1322–1328. IEEE.

He, H., Chen, S., Li, K., and Xu, X. (2011). Incremental learning from stream data. *IEEE Transactions on Neural Networks*, 22(12):1901–1914.

Heinze, T., Aniello, L., Querzoni, L., and Jerzak, Z. (2014). Cloud-based data stream processing. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pages 238–245. ACM.

Hellendoorn, H. and Baudrexl, R. (1995). Fuzzy neural traffic control and forecasting. In *Proceedings of the International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium.*, volume 4, pages 2187–2194. IEEE.

Hidas, P. (2002). Modelling lane changing and merging in microscopic traffic simulation. *Transportation Research Part C: Emerging Technologies*, 10(5):351–371.

Hoens, T. R., Polikar, R., and Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101.

Hoffmann, K.-P. (2011). Biosignale erfassen und verarbeiten. Springer.

Hofstee, P. and Nowka, K. J. (2013). The Big Deal about Big Data - A Perspective from IBM Research. Presentation at IEEE NAS Conference, Xi'An China.

Hogan, W. R. and Wagner, M. M. (1997). Accuracy of data in computer-based patient records. *Journal of the American Medical Informatics Association*, 4(5):342–355.

Horovitz, O., Krishnaswamy, S., and Gaber, M. M. (2007). A fuzzy approach for interpretation of ubiquitous data stream clustering and its application to road safety. *Intelligent Data Analysis*, 11:89–108.

Hove, S. E. and Anda, B. (2005). Experiences from conducting semi-structured interviews in empirical software engineering research. In *Proceedings of the 11th IEEE International Symposium on Software Metrics*. IEEE.

Hoyer, R. (2003). Hinweise zur Datenvervollständigung und Datenaufbereitung in verkehrstechnischen Anwendungen. Technical report, Forschungsgesellschaft für Strassen- und Verkehrswesen. (in German).

Huber, W. (2001a). Fahrzeuggenerierte Daten zur Gewinnung von Verkehrsinformationen. Technical report, BMW AG.

Huber, W. (2001b). *Vehicle-generated Data for Aquiring Traffic Information.* PhD thesis, TU München.

Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 97–106. ACM.

Hunter, T., Abbeel, P., and Bayen, A. (2014). The path inference filter: model-based low-latency map matching of probe vehicle data. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):507–529.

ISO 14819-1 (2013). Intelligent transport systems – Traffic and travel information messages via traffic message coding – Part 1: Coding protocol for Radio Data System – Traffic Message Channel (RDS-TMC) using ALERT-C.

ISO 14819-2 (2013). Intelligent Transport Systems – Traffic and Travel Information Messages via Traffic Message Coding – Part 2: Event and Information Codes for Radio Data System – Traffic Message Channel (RDS-TMC) using ALERT-C.

Izadpanah, P. and Hellinga, B. (2007). Wide-area wireless traffic conditions monitoring: reality or wishful thinking. In *Proceedings of the 2007 Annual Conference of the Canadian Institute of Transportation Engineers (CITE)*, pages 1–18.

Jain, N., Mishra, S., Srinivasan, A., Gehrke, J., Widom, J., Balakrishnan, H., Çetintemel, U., Cherniack, M., Tibbetts, R., and Zdonik, S. B. (2008). Towards a Streaming SQL Standard. *Proceedings of the VLDB Endowment (PVLDB)*, 1(2):1379–1390.

Janecek, A., Valerio, D., Hummel, K. A., Ricciato, F., and Hlavacs, H. (2015). The Cellular Network as a Sensor: From Mobile Phone Data to Real-Time Road Traffic Monitoring. *Intelligent Transportation Systems, IEEE Transactions on*, 16(5):2551–2572.

Jarke, M., Jeusfeld, M. A., Quix, C., and Vassiliadis, P. (1999). Architecture and Quality in Data Warehouses: An Extended Repository Approach. *Information Systems*, 24(3):229–253.

Jeske, T. (2013). Floating Car Data from Smartphones: What Google and Waze Know about You and how Hackers can Control Traffic. In *Proceedings of the BlackHat Europe 2013*.

Jiang, F. and Leung, C. K.-S. (2013). Stream mining of frequent patterns from delayed batches of uncertain data. In *Proceedings of the 15th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, pages 209–221. Springer.

Jongiran, E. (2013). Evaluating Concept Drift in Traffic Applications based on Data Streams. Master's thesis, RWTH Aachen University.

Judah, S. and Friedman, T. (2014). Magic Quadrant for Data Quality Tools. Technical report, Gartner.

Juran, J. M. (1993). *Der neue Juran: Qualität von Anfang an.* verlag moderne industrie. In German.

Juran, J. M. (1999). How to think about Quality. In (Juran and Godfrey, 1999a), chapter 2, pages 1–18.

Juran, J. M. and Godfrey, A. B., editors (1999a). *Juran's Quality Handbook.* McGraw-Hill.

Juran, J. M. and Godfrey, A. B. (1999b). The Quality Control Process. In (Juran and Godfrey, 1999a), chapter 4, pages 1–29.

Kargupta, H., Sarkar, K., and Gilligan, M. (2010). MineFleet®: an overview of a widely adopted distributed vehicle performance data mining system. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 37–46. ACM.

Kemper, A. and Eickler, A. (2009). *Datenbanksysteme - Eine Einführung.* Oldenbourg.

Kensche, D., Quix, C., Chatti, M. A., and Jarke, M. (2007). *GeRoMe*: A Generic Role Based Metamodel for Model Management. *Journal on Data Semantics*, VIII:82–117.

Kerner, B. S. (2004). *The Physics of Traffic.* Springer.

Kerner, B. S., Klenov, S. L., and Wolf, D. E. (2002). Cellular automata approach to three-phase traffic theory. *Journal of Physics A: Mathematical and General*, 35(47):9971.

Kesting, A., Treiber, M., and Helbing, D. (2007). General lane-changing model mobil for car-following models. *Transportation Research Record: Journal of the Transportation Research Board*, (1999):86–94.

Khaleghi, B., Khamis, A., Karray, F. O., and Razavi, S. N. (2011). Multisensor Data Fusion: A Review of the State-of-the-art. *Information Fusion*, 14(1):28–44.

Khan, A. (2007). Intelligent Infrastructure-based Queue-end Warning System for Avoiding Rear Impacts. *IET Intelligent Transport Systems*, 1:138–143.

Khan, W. Z., Xiang, Y., Aalsalem, M. Y., and Arshad, Q. (2013). Mobile Phone Sensing Systems: A Survey. *Communications Surveys & Tutorials, IEEE*, 15(1):402–427.

Kim, S., Beckmann, L., Pistor, M., Cousin, L., Walter, M., and Leonhardt, S. (2009). A versatile body sensor network for health care applications. In *Proceedings of the 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 175–180. IEEE.

Kirschfink, H. (2000). *Informationsmanagement im Verkehr*, chapter Architektur einer modernen Telematik-Zentrale, pages 201–212. Phyisca-Verlag.

Klein, A. and Lehner, W. (2009). Representing Data Quality in Sensor Data Streaming Environments. *ACM Journal of Data and Information Quality*, 1(2):1–28.

Ko, J., Lu, C., Srivastava, M. B., Stankovic, J. A., Terzis, A., and Welsh, M. (2010). Wireless sensor networks for healthcare. *Proceedings of the IEEE*, 98(11):1947–1960.

Koch, S. (2011). Total Quality Management. In *Einführung in das Management von Geschäftsprozessen*, pages 185–220. Springer. In German.

Kolozali, S., Bermudez-Edo, M., Puschmann, D., Ganz, F., and Barnaghi, P. (2014). A Knowledge-based Approach for Real-time IoT Data Stream Annotation and Processing. In *Proceedings of the 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, pages 215–222. IEEE.

Koziolek, H. (2008). Goal, question, metric. In *Dependability metrics*, pages 39–42. Springer.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138.

Krämer, J. and Seeger, B. (2009). Semantics and Implementation of Continous Sliding Window Queries over Data Streams. *ACM Trans. on Database Systems*, 34:1–49.

Kranen, P. (2011). *Anytime Algorithms for Stream Data Mining*. PhD thesis, RWTH Aachen University.

Kranen, P., Müller, E., Assent, I., Krieger, R., and Seidl, T. (2010). Incremental learning of medical data for multi-step patient health classification. In C., P. and C., B., editors, *Database Technology for Life Sciences and Medicine*, pages 321–344. World Scientific Publishing Co Pte Ltd.

Kreps, J. (2014). Questioning the Lambda Architecture. O'Reilly Data Tools. https://www.oreilly.com/ideas/questioning-the-lambda-architecture. Date accessed: 28.10.2016.

Kuhne, R. D. (2011). Greenshields' legacy: Highway traffic. *Transportation Research E-Circular*, (E-C149):3–10.

Kuka, C., Bolles, A., Funk, A., Eilers, S., Schweigert, S., Gerwinn, S., and Nicklas, D. (2013). SaLsA Streams: Dynamic Context Models for Autonomous Transport Vehicles Based on Multi-sensor Fusion. In *Proceedings of the 14th IEEE International Conference on Mobile Data Management*, volume 1, pages 263–266. IEEE.

Kuka, C. and Nicklas, D. (2014a). Enriching Sensor Data Processing with Quality Semantics. In *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 437–442. IEEE.

Kuka, C. and Nicklas, D. (2014b). Quality Matters: Supporting Quality-aware Pervasive Applications by Probabilistic Data Stream Management. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pages 1–12. ACM.

Kumar, S., Nilsen, W., Pavel, M., and Srivastava, M. (2013). Mobile Health: Revolutionizing Healthcare Through Transdisciplinary Research. *Computer*, 46(1):28–35.

Lakshmanan, G. T., Li, Y., and Strom, R. (2008). Placement strategies for internet-scale data stream systems. *Internet Computing, IEEE*, 12(6):50–60.

Langer, A. M. (2007). *Analysis and design of information systems*. Springer Science & Business Media.

Laskov, P., Gehl, C., Krüger, S., and Müller, K. (2006). Incremental support vector learning: Analysis, implementation and applications. *The Journal of Machine Learning Research*, 7:1909–1936.

Law, Y., Wang, H., and Zaniolo, C. (2004). Query languages and data models for database sequences and data streams. In (Nascimento et al., 2004), pages 492–503.

Law, Y.-N., Wang, H., and Zaniolo, C. (2011). Relational Languages and Data Models for Continuous Queries on Sequences and Data Streams. *ACM Transactions on Embedded Computing Systems*, 36(3):1–31.

Lee, V. E., Liu, L., and Jin, R. (2015). Decision Trees: Theory and Algorithms. In (Aggarwal, 2015c), chapter 4, pages 87–120.

Lee, Y., Strong, D., Kahn, B., and Wang, R. (2002). AIMQ: A Methodology for Information Quality Assessment. *Information & management*, 40(2):133–146.

Leonhardt, A. (2012). Verkehrssimulation in der Verkehrsplanung. Presentation at Universität der Bundeswehr München, https://www.unibw.de/bauv7/strassenverkehrstechnik/aktuelles/ptv-group-simulation-axel-leonhardt-unibw-2012-12.pdf.

Levin, R., Kravi, E., and Kanza, Y. (2012). Concurrent and Robust Topological Map Matching. In (Cruz et al., 2012).

Li, J., Maier, D., Tufte, K., Papadimos, V., and Tucker, P. (2005). Semantics and evaluation techniques for window aggregates in data streams. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 311–322. ACM.

Liaw, S.-T., Rahimi, A., Ray, P., Taggart, J., Dennis, S., de Lusignan, S., Jalaludin, B., Yeo, A., and Talaei-Khoei, A. (2013). Towards an Ontology for Data Quality in Integrated Chronic Disease Management: A Realist Review of the Literature. *International Journal of Medical Informatics*, 82(1):10–24.

Lindeberg, M., Goebel, V., and Plagemann, T. (2010). Adaptive Sized Windows To Improve Real-Time Health Monitoring - A Case Study on Heart Attack Predicition. In *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 459–468.

Liu, K., Li, Y., He, F., Xu, J., and Ding, Z. (2012). Effective Map-matching on the Most Simplified Road Network. In (Cruz et al., 2012).

Liu, L. and Özsu, M. T., editors (2009). *Encyclopedia of Database Systems*. Springer.

Liu, L., Pu, C., and Tang, W. (1999). Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE Trans. on Knowl. and Data Eng.*, 11(4):610–628.

Liu, Y., Choudhary, A., Zhou, J., and Khokhar, A. (2006). A Scalable Distributed Stream Mining System for Highway Traffic Data. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Proceedings of 10th the European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, volume 4213 of *Lecture Notes in Computer Science*, pages 309–321. Springer.

Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., and Huang, Y. (2009). Map-matching for Low-sampling-rate GPS Trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pages 352–361. ACM.

Maciejewski, T. and Stefanowski, J. (2011). Local Neighbourhood Extension of SMOTE for Mining Imbalanced Data. In *Proceedings of the 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 104–111. IEEE.

Mahmood, M. A., Seah, W. K., and Welch, I. (2015). Reliability in Wireless Sensor Networks: A Survey and Challenges Ahead. *Computer Networks*, 79:166–187.

Maier, D., Li, J., Tucker, P., Tufte, K., and Papadimos, V. (2005). Semantics of Data Streams and Operators. In *ICDT 2005*, number 3363 in LNCS, pages 37–52. Springer.

Margiotta, R. (2002). State of the practice for traffic data quality. In *Proceedings of the 2002 Traffic Data Quality Workshop*.

Mariscal, G., Marbán, Ó., and Fernández, C. (2010). A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*, 25(02):137–166.

Marr, B. (2016). The Most Practical Big Data Use Cases Of 2016. Forbes Tech. http://www.forbes.com/sites/bernardmarr/2016/08/25/the-most-practical-big-data-use-cases-of-2016/#3c39d1da7533. Date accessed: 31.10.2016.

Martin, J. and Finkelstein, C. (1988). *Information engineering*. Savant.

Marz, N. and Warren, J. (2015). *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co.

Maurer, P. (2007). Aspekte der Fahrbahngriffigkeit und ihr Einfluss auf erreichbare PKW-Bremsverzögerungen. *SCHRIFTENREIHE STRASSENFORSCHUNG*, (564). In German.

Mawilmada, P. K., Smith, S. E., and Sahama, T. (2012). Investigation of decision making issues in the use of current clinical information systems. *Studies in health technology and informatics*, 178:136–143.

Mazur, F., Weber, D., Chrobok, R., Hafstein, S. F., Pottmeier, A., and Schreckenberg, M. (2005). Basics of the Online Traffic Information System AUTOBAHN.NRW. Technical report, Universität Duisburg-Essen.

McGowen, P. and Sanderson, M. (2011). Accuracy of Pneumatic Road Tube Counters. Technical report, Institute of Transportation Engineers, Anchorage, USA.

McGregor, C. (2015). A framework for online health analytics for advanced prognostics and health management of astronauts. In *Proceedings of the 2015 IEEE Aerospace Conference*, pages 1–7. IEEE.

Mechael, P. N. and Sloninsky, D. (2008). Towards the Development of an mHealth Strategy: A Literature Review. Technical report, World Health Organization.

Mensebach, W. (2004). *Strassenverkehrsplanung Strassenverkehrstechnik*. Werner Verlag.

Misra, A., Blount, M., Kementsietsidis, A., Sow, D., and Wang, M. (2008). Advances and challenges for scalable provenance in stream processing systems. In *Provenance and Annotation of Data and Processes*, pages 253–265. Springer.

Mitre, G. (2012). Real-time Map Matching in Car-To-X Applications. Master's thesis, RWTH Aachen University.

Mitunevičius, V., Nagurnas, S., Unarski, J., and Wach, W. (2009). Research of car braking in winter conditions. In *6th International Scientific Conference TRANS-BALTICA 2009*, pages 156–161. Vilniaus Gedimino Technikos Universitetas.

Moen, R. and Norman, C. (2006). Evolution of the PDCA Cycle. Technical report, Process Improvement Detroit.

Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. (1990). Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems (TOIS)*, 8(4):325–362.

Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):2221–2229.

Nagel, K., Wolf, D. E., Wagner, P., and Simon, P. (1998). Two-lane Traffic Rules for Cellular Automata: A Systematic Approach. *Physical Review E*, 58(2):1425.

Nascimento, M. A., Özsu, M. T., Kossmann, D., Miller, R. J., Blakeley, J. A., and Schiefer, K. B., editors (2004). *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, Toronto, Canada. Morgan Kaufmann.

Necasky, M. (2007). XSEM: a conceptual model for XML. In *Proceedings of the Fourth Asia-Pacific Conference on Comceptual Modelling*, pages 37–48. Australian Computer Society, Inc.

Newell, G. F. (1961). Nonlinear effects in the dynamics of car following. *Operations research*, 9(2):209–229.

Ochieng, W., Quddus, M., and Noland, R. (2003). Map-matching in complex urban road networks. *Revista Brasileira de Cartografia*, 2(55):1–14.

Oza, N. (2005). Online bagging and boosting. In *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345. IEEE.

Oza, N. C. and Russell, S. (2001). Online Bagging and Boosting. In Jaakkola, T. and Richardson, T., editors, *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics*, pages 105–112, Key West, Florida. USA. Morgan Kaufmann.

Pankratov, S. and Znamenskaya, T. (2014). Mobile Health (mHealth): A Conceptual View. *Universal Journal of Public Health*, 2(2):35–49.

Paradis, L. and Han, Q. (2007). A survey of fault management in wireless sensor networks. *Journal of Network and Systems Management*, 15(2):171–190.

Patroumpas, K. and Sellis, T. K. (2006). Window Specification over Data Streams. In *Proceedings of Current Trends in Database Technology - EDBT 2006 Workshops*, pages 445–464.

Pattara-atikom, W. and Peachavanish, R. (2007). Estimating Road Traffic Congestion from Cell Dwell Time using Neural Network. In *Proceedings of the 7th International Conference on ITS Telecommunications*, pages 1–6.

Pattara-atikom, W., Peachavanish, R., and Luckana, R. (2007). Estimating Road Traffic Congestion using Cell Dwell Time with Simple Threshold and Fuzzy Logic Techniques. In *Proceedings of the 2007 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 956–961.

Pelossof, R., Jones, M., Vovsha, I., and Rudin, C. (2009). Online coordinate boosting. In *Proceedings of the 12th International Conference on Computer Vision (ICCV), Workshops*, pages 1354–1361. IEEE.

Peng, F. and Chawathe., S. S. (2003). XSQ: A Streaming XPath Engine. Technical report, Computer Science Department, University of Maryland.

Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer Publishing Company, Incorporated, 1st edition.

Poon, C. C., Lo, B. P., Yuce, M. R., Alomainy, A., and Hao, Y. (2015). Body sensor networks: in the era of big data and beyond. *IEEE reviews in biomedical engineering*, 8:4–16.

Prati, R. C., Batista, G. E., and Silva, D. F. (2015). Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, 45(1):247–270.

Preece, A., Missier, P., Embury, S., Jin, B., and Greenwood, M. (2008). An Ontology-based Approach to Handling Information Quality in E-Science. *Concurrency and computation: practice and experience*, 20(3):253–264.

Quddus, M., Ochieng, W., Zhao, L., and Noland, R. (2003). A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7:157–167.

Quddus, M. and Washington, S. (2015). Shortest Path and Vehicle Trajectory Aided Map-matching for Low Frequency GPS Data. *Transportation Research Part C: Emerging Technologies*, 55:328–339.

Quddus, M. A., Ochieng, W. Y., and Noland, R. B. (2007). Current Map-matching Algorithms for Transport Applications: State-of-the-art and Future Research Directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328.

Quix, C., Barnickel, J., Geisler, S., Hassani, M., Kim, S., Li, X., Lorenz, A., Quadflieg, T., Gries, T., Jarke, M., Leonhardt, S., Meyer, U., and Seidl, T. (2013). HealthNet: A System for Mobile and Wearable Health Information Management. In *Proceedings of the 3rd International Workshop on Information Management in Mobile Applications (IMMoA)*, pages 36–43.

Raman, V., Deshpande, A., and Hellerstein, J. M. (2003). Using State Modules for Adaptive Query Processing. In Dayal, U., Ramamritham, K., and Vijayaraman, T. M., editors, *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 353–366, Bangalore, India. IEEE Computer Society.

Ranjan, R. (2014). Streaming big data processing in datacenter clouds. *IEEE Cloud Computing*, 1(1):78–83.

Redman, T. C. (1996). *Data Quality for the Information Age.* Artech House, Boston.

Redman, T. C. (1999). Second-Generation Data Quality Systems. In (Juran and Godfrey, 1999a), chapter 34, pages 1–14.

Redman, T. C. (2004). Data: An unfolding quality disaster. *DM REVIEW*, 14(8):21–23.

Redman, T. C. (2013). Data Quality Management Past, Present, and Future: Towards a Management System for Data. pages 15–40. Springer.

research2guidance (2014). mHealth App Developer Economics 2014 - Fourth Annual Study on mHealth App Publishing. Technical report, research2guidance.

Ridge, E. (2015). *Guerrilla Analytics: Techniques for Managing Data and Analytics Teams.* Morgan Kaufmann.

Roussopoulos, N. and Karagiannis, D. (2009). Conceptual modeling: past, present and the continuum of the future. In *Conceptual Modeling: Foundations and Applications*, pages 139–152. Springer.

Runeson, P. and Höst, M. (2009). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14(2):131–164.

Rutkowski, L., Jaworski, M., Pietruczuk, L., and Duda, P. (2014a). Decision trees for mining data streams based on the gaussian approximation. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):108–119.

Rutkowski, L., Jaworski, M., Pietruczuk, L., and Duda, P. (2014b). The CART Decision Tree for Mining Data Streams. *Information Sciences*, 266:1–15.

Sathe, S., Papaioannou, T. G., Jeung, H., and Aberer, K. (2013). A Survey of Model-based Sensor Data Acquisition and Management. In (Aggarwal, 2013).

Sauter, M. (2011). *Grundkurs Mobile Kommunikationssysteme - UMTS, HSDPA und LTE, GSM, GPRS und Wireless LAN*. Springer.

Scheer, A.-W. (1992). Architecture of Integrated Information Systems: Foundations of Enterprise Modelling.

Scheer, A.-W. (2013). *ARIS - vom Geschäftsprozess zum Anwendungssystem*. Springer-Verlag.

Scheer, A.-W. and Nüttgens, M. (2000). ARIS architecture and reference models for business process management. Springer.

Schmidt, S. (2006). *Quality-of-Service-Aware Data Stream Processing*. PhD thesis, Technische Universität Dresden.

Schmidt, S., Berthold, H., and Lehner, W. (2004). Qstream: Deterministic querying of data streams. In (Nascimento et al., 2004), pages 1365–1368.

Schmitt, L., Falck, T., Wartena, F., and Simons, D. (2007). Novel ISO/IEEE 11073 Standards for Personal Telehealth Systems Interoperability. In *Proceedings of the Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP)*, pages 146–148. IEEE.

Schnabel, W. and Lohse, D. (1997). *Grundlagen der Strassenverkehrstechnik und der Verkehrsplanung*. Verlag für Bauwesen.

Schneider, S., Hirzel, M., Gedik, B., and Wu, K.-L. (2012). Auto-parallelizing stateful distributed streaming applications. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 53–64. ACM.

Schreckenberg, M., Neubert, L., and Wahle, J. (2001). Simulation of traffic in large road networks. *Future Generation Computer Systems*, 17(5):649–657.

Schroeder, R., Linderman, K., Liedtke, C., and Choo, A. (2008). Six Sigma: Definition and Underlying Theory. *Journal of Operations Management*, 26(4):536–554.

Schützle, R. (2009). Quality Management in ROSATTE. In *Proceedings of the 16th World Congress on Intelligent Transportation Systems and Services (ITS)*.

Schützle, R., Frank, J., Delorme, C., Petit, F., Svensk, P.-O., Wikström, L., Isaksson, P., Boterbergh, B., Keith, H., and Haspel, U. (2010). ROSATTE - D5.3 - Report on validation of data quality management concept and experiences from test sites. Technical report, University of Stuttgart.

Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5(4):13–22.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., de Carvalho, A. C., and Gama, J. (2013). Data Stream Clustering: A Survey. *ACM Computing Surveys (CSUR)*, 46(1):13.

Singh, D., Ibrahim, A., Yohanna, T., and Singh, J. (2007). An overview of the applications of multisets. *Novi Sad Journal of Mathematics*, 37(3):73–92.

Smith, B., Zhang, H., Fontaine, M., and Green, M. (2004). Wireless Location Technology-based Traffic Monitoring: Critical Assessment and Evaluation of an Early-generation System. *Journal of Transportation Engineering*, 130:576.

Soleimankhani, F. (2010). Evaluation Framework for Data Stream Mining Algorithms for Traffic State Estimation in a Data Fusion Architecture. Master's thesis, RWTH Aachen University.

Sow, D., Turaga, D. S., and Schmidt, M. (2013). *Mining of Sensor Data in Healthcare: A Survey*, chapter 14, pages 459–504. In (Aggarwal, 2013).

Srivastava, U. and Widom, J. (2004). Flexible Time Management in Data Stream Systems. In Deutsch, A., editor, *Proc. 23rd ACM Symposium on Principles of Database Systems (PODS)*, pages 263–274, Paris, France. ACM.

Steenbruggen, J., Borzacchiello, M., Nijkamp, P., and Scholten, H. (2011a). Mobile phone data from GSM networks for traffic parameter and urban spatial pattern assessment: a review of applications and opportunities. *GeoJournal*, 76(3):1–21.

Steenbruggen, J., Borzacchiello, M., Nijkamp, P., and Scholten, H. (2011b). The Use of GSM Data for Transport Safety Management: An Exploratory Review. *Research Memorandum*, 2011:1–32.

Steinauer, B., Brake, M., Baier, M. M., and Kathmann, T. (2006). *Integration mobil erfasster Verkehrsdaten (FCD) in die Steuerungsverfahren der kollektiven Verkehrsbeeinflussung*. Bundesministerium für Verkehr, Bau und Stadtentwicklung, Abteilung Straßenbau, Straßenverkehr.

Stoa, S., Lindeberg, M., and Goebel, V. (2008). Online analysis of myocardial ischemia from medical sensor data streams with esper. In *Proceedings of the 1st International Symposium on Applied Sciences on Biomedical and Communication Technologies*, pages 1–5. IEEE.

Stonebraker, M., Çetintemel, U., and Zdonik, S. B. (2005). The 8 requirements of real-time stream processing. *SIGMOD Record*, 34(4):42–47.

Strong, D., Lee, Y., and Wang, R. (1997). Data quality in context. *Communications of the ACM*, 40(5):103–110.

Sullivan, M. (1996). Tribeca: A Stream Database Manager for Network Traffic Analysis. In *Proceedings of the 22nd International Conference on Very Large DataBases (VLDB)*, volume 96, page 594.

Sun, J., Faloutsos, C., Papadimitriou, S., and Yu, P. (2007). GraphScope: Parameter-free Mining of Large Time-evolving Graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDDM)*, pages 687–696. ACM.

Sussman, J., Pearce, V., Hicks, B., Carter, M., Lappin, J., Casey, R., Orban, J., McGurrin, M., and DeBlasio, A. (2000). What have we Learned About Intelligent Transportation Systems? Technical report, U.S. Department of Transportation.

Syropoulos, A. (2000). Mathematics of Multisets. In *Proceedings of the Workshop on Multiset Processing 2000*, pages 286–295.

Tang, Y., Zhu, A. D., and Xiao, X. (2012). An efficient algorithm for mapping vehicle trajectories onto road networks. In (Cruz et al., 2012), pages 601–604.

Tarnoff, P. (2002). Getting to the INFOstructure. In *Proceedings of the 2002 Transport Research Board Roadway INFOstructure Conference.*

Tayal, M. (2005). Location Services in the GSM and UMTS Networks. In *Proceedings of the 2005 IEEE International Conference on Personal Wireless Communications (ICPWC).*

Teorey, T. J. and Fry, J. P. (1982). *Design of Database Structures.* Prentice-Hall Incorporated.

Terry, D., Goldberg, D., Nichols, D., and Oki, B. (1992). Continuous Queries over Append-Only Databases. In *Proceedings of the 1992 ACM SIGMOD International Conference on Modeling of Data.*

Tettamanti, T., Demeter, H., and Varga, I. (2012). Route Choice Estimation Based on Cellular Signaling Data. *Acta Polytechnica Hungarica*, 9(4):207–220.

Thakkar, H., Mozafari, B., and Zaniolo, C. (2008). Designing an inductive data stream management system: the stream mill experience. In *Proceedings of the 2nd International Workshop on Scalable Stream Processing Systems*, pages 79–88. ACM.

Thiru, K., Hassey, A., and Sullivan, F. (2003). Systematic review of scope and quality of electronic patient record data in primary care. *BMJ*, 326(7398):1070.

Tomek, I. (1976). Two modifications to CNN. *IEEE Transactions On Systems, Man, and Cybernetics.*

TomTom (2010). TomTom HD: How TomTom's HD Traffic and IQ Routes Data Provides the Very Best Routing. `http://www.tomtom.com`.

Tran, T., Peng, L., Li, B., Diao, Y., and Liu, A. (2010). PODS: a new model and processing algorithms for uncertain data streams. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 159–170. ACM.

Treiber, M., Hennecke, A., and Helbing, D. (2000). Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805.

Treiber, M. and Kesting, A. (2010). *Verkehrsdynamik und -simulation.* Springer.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Informe técnico: Departament of Computer Science, Trinity College, Dublin, IE. https://www.cs.tcd.ie/publications/techreports/reports.

Tucker, P., Maier, D., Sheard, T., and Fegaras, L. (2003). Exploiting punctuation semantics in continuous data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):555–568.

Turner, S. (2002). Defining and Measuring Traffic Data Quality. In *Proceedings of the 2002 Traffic Data Quality Workshop.*

Valerio, D., D'Alconzo, A., Ricciato, F., and Wiedermann, W. (2009). Exploiting cellular networks for road traffic estimation: a survey and a research roadmap. In *Proceedings of the IEEE 69th Vehicular Technology Conference (VTC)*, pages 1–5. IEEE.

Van Hinsbergen, C. P., Schreiter, T., Zuurbier, F. S., Van Lint, J., and Van Zuylen, H. J. (2012). Localized extended kalman filter for scalable real-time traffic state estimation. *Intelligent Transportation Systems, IEEE Transactions on*, 13(1):385–394.

Varshney, U. (2007). Pervasive healthcare and wireless health monitoring. *Mobile Networks and Applications*, 12(2-3):113–127.

Velaga, N., Quddus, M., and Bristow, A. (2009). Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies*, 17(6):672–683.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57.

Wang, R. (1998). A product perspective on total data quality management. *Communications of the ACM*, 41(2):58–65.

Wang, R. Y., Kon, H. B., and Madnick, S. E. (1993). Data quality requirements analysis and modeling. In *Proceedings of the 9th International Conference on Data Engineering (ICDE)*, pages 670–677. IEEE.

Wang, R. Y. and Strong, D. (1996). Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 12(4):5–33.

Wang, Y. and Papageorgiou, M. (2005). Real-time Freeway Traffic State Estimation based on Extended Kalman Filter: A General Approach. *Transportation Research Part B: Methodological*, 39(2):141–167.

Wang, Y., Papageorgiou, M., and Messmer, A. (2007). Real-time Freeway Traffic State Estimation based on Extended Kalman Filter: A Case Study. *Transportation Science*, 41(2):167–181.

Weber, S. (2011). Ontology-based Data Quality Framework for Data Stream Applications. Master's thesis, RWTH Aachen University.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101.

Wiedemann, R. (1974). Simulation des Strassenverkehrsflusses. *Schriftenreihe des Instiuts für Verkehrswesen der Universität Karlsruhe*, 8.

Wirth, R. and Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39. Citeseer.

Wojnarski, M., Góra, P., Szczuka, M. S., Nguyen, H. S., Swietlicka, J., and Zeinalipour-Yazti, D. (2010). IEEE ICDM 2010 Contest: TomTom Traffic Prediction for Intelligent GPS Navigation. In Fan, W., Hsu, W., Webb, G. I., Liu, B., Zhang, C., Gunopulos, D., and Wu, X., editors, *10th IEEE Intl. Conf. on Data Mining – Workshop Proceedings (ICDMW)*, pages 1372–1376, Sydney, Australia. IEEE Computer Society.

Wu, P., Xue, H., and Hu, X. (2015). Particle filter based traffic data assimilation with sensor informed proposal distribution. In *Proceedings of the 48th Annual Simulation Symposium*, pages 173–180. Society for Computer Simulation International.

Wunnava, S., Yen, K., Babij, T., Zavaleta, R., Romero, R., and Archilla, C. (2007). Travel time estimation using cell phones (ttecp) for highways and roadways. Technical report, Florida Department of Transportation.

Yang, G.-Z., editor (2014). *Body Sensor Networks*. Springer.

Yang, G.-Z., Andreu-Perez, J., Hu, X., and Thiemjarus, S. (2014a). Multi-sensor Fusion. In (Yang, 2014), chapter 8, pages 301–350.

Yang, G.-Z., Aziz, O., Kwasnicki, R., Merrifield, R., Darzi, A., and Lo, B. (2014b). Introduction. In (Yang, 2014), chapter 1, pages 1–54.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, 26(3):276–292.

Zaharia, M., Das, T., Li, H., Shenker, S., and Stoica, I. (2012). Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. In *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*, pages 1–6.

Zander, E. and Eimer, G. (2010). Läufer muss im Ziel reanimiert werden. *Aachener Nachrichten*. In German.

Zaniolo, C. (2012). Logical Foundations of Continuous Query Languages for Data Streams. In *Datalog in Academia and Industry*, pages 177–189. Springer.

Zdonik, S., Sibley, P., Rasin, A., Sweetser, V., Montgomery, P., Turner, J., Wicks, J., Zgolinski, A., Snyder, D., Humphrey, M., and Williamson, C. (2004). Streaming for Dummies. https://core.ac.uk/display/24305267. Date accessed: 28.10.2016.

Zhang, L., Stoffel, A., Behrisch, M., Mittelstadt, S., Schreck, T., Pompl, R., Weber, S., Last, H., and Keim, D. (2012). Visual Analytics for the Big Data Era - A Comparative Review of State-of-the-art Commercial Systems. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 173–182. IEEE.

Zhao, L., Ochieng, W. Y., Quddus, M. A., and Noland, R. B. (2003). An Extended Kalman Filter Algorithm for Integrating GPS and Low Cost Dead Reckoning System Data for Vehicle Performance and Emissions Monitoring. *The Journal of Navigation*, 56(2):257–275.