A Model for CBR Systems that Adapt to Rapidly Changing Context

Björn Decker and Markus Nick

Fraunhofer Institute for Experimental Software Engineering (IESE)
67663 Kaiserslautern, Germany
{bjoern.decker, markus.nick}@iese.fraunhofer.de

Abstract. Challenges in context-sensitive applications are not limited to reasoning. Rapid changes in the environment, which are reflected by the context information, pose a particular challenge. During the run of a single CBR cycle, a lazy adaptation can be used to deal with this challenge of rapidly changing context. In an ambient intelligence system, there is also the need to modularize the knowledge according to the overall system. We present a model for CBR in such a setting. The model consists of an extended/modified CBR process, a knowledge model, and an architecture pattern for embedding a CBR module into an ambient intelligence system. Our focus is on the context-aware adaptation of the actions to be executed when a certain situation is recognized. The technical feasibility of our model has been evaluated with an application in the area of ambient assisted living for elderly people.

1 Introduction

The application of case-based reasoning to context-aware systems offers a promising new application field. Most of the systems reported in the literature rely on explicit user interaction to derive applicable cases [1] [2] [3]. By using context information, explicit user interaction is reduced or does not take place. This is also the case for so-called ambient intelligence systems, which use sensors to automatically obtain context information from the environment, reason using this context information, and act using devices in the ambient intelligence system. We are researching on ambient intelligence systems in the domain of assisted living for elderly people[1]. Several studies show that the percentage of elderly people will grow in the upcoming years. Elderly people can stay in their homes longer through intelligent assistance, which results in cost savings for the society as well as benefits to the elderly people by offering them a more self determined life.

Summarized, context-aware systems capture the environment of a user. Some properties of this environment will change during the execution of actions and,

[1] Part of this work has been funded by the Bundesministerium für Bildung und Forschung (BMBF) and the Ministerium für Wissenschaft, Weiterbildung, Forschung und Kultur of Rheinland-Pfalz in the context of the BelAmI project (www.belami-project.org).

therefore, force the context-aware system to adapt its actions accordingly. The underlying knowledge model of the context-aware system, which is the basis for this adaptation, is known during application time and remains stable. Since ambient intelligence systems are context-aware system with a high distribution, one need distribution strategy for this knowledge model. The actual context information based on this model will be influenced by those changes in the environment at runtime. Thus, a CBR system inside such a context-aware system has to take into account that information relevant for selecting, adapting, and evaluating/applying cases might change within one single run of a CBR cycle. Furthermore, since this change can happen within a short period of time, part of the information relevant for a CBR system/module can only be acquired at the execution time of actions proposed by the CBR system/module.

In the remainder of this paper – which is influenced by our research background – we investigate context-aware systems in the area of ambient intelligence. The paper itself is structured as follows: First, we present and discuss definitions for the terms context, context-aware systems and related terms as used in this paper. Second, we present the major problems identified by us and our solution idea for building CBR systems that adapt to rapidly changing context. Then, we present the CBR system model consisting of CBR process, architecture pattern, and knowledge representation. Afterwards, we present our application in the area of assisted living for elderly people, which validated the technical feasibility of our model. This is followed by an overview of related work. A conclusion and outlook to future work is the last section of this paper.

2 Definition of Related Terms

Dey and Abowd [4] define context as *“any information that can be used to characterize the situation of an entity. An entity is a person, place, or an object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”* Furthermore, they identify several aspects of the context of entities inside context-aware systems that need to be taken into account:

- Identity: What is the identity of the entity? To which degree can it be distinguished from other entities, i.e., does it have a unique identifier or is the class of a certain entity identified only? This identification is of particular importance for deriving further context characteristics.
- Location: What is the location of an entity? This information is used for determining information about the entity itself (e.g. whether a person moves or not) or to derive relations between entities (e.g., that a person is close to a certain device).
- Activity: What activities can be performed by a certain entity? Which one is currently being performed? This information is used for presenting further applicable activities in the context of an entity.
- Time: What is the time when a certain situation occurred? To which point in time is a certain activity applicable? (E.g., the user should not be disturbed between midnight and six o’clock in the morning). This information can be used for

determining whether certain activities are performed in the correct sequence, or for scheduling activities.

With this notion of context, context information is bound to the entities inside the context-aware system. Therefore, context is – at least logically - distributed across the entities within a context-aware system. References among entities can be used for obtaining further context information from other entities. The information about the context bound to one object is called the primary context of an entity, while context information derived from referenced objects is called secondary context. Furthermore, this notion of context allows defining the retrieval vector of context-aware applications: The retrieval vector can be created by analyzing the entities relevant for the application and by applying the context aspects mentioned before.

The term context-aware systems is currently not precisely defined and is used as synonym for ubiquitous computing and Ambient Intelligence. Therefore, we give the definition of context-aware computing first. The terms ubiquitous computing and Ambient Intelligence are then compared to the explanation of context-aware systems.

Concerning context-aware computing, [5] defines context-awareness as *"a term from computer science, that is used for devices that have information about the circumstances under which they operate and can react accordingly. Context-aware devices may also try to make assumptions about the user's current situation*". In their detailed survey on context and context-aware systems, [6] define context-aware systems as: *"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"*. Although context-aware systems can be seen as a specialization of ubiquitous systems (e.g., [7]) , the aspect of adapting a certain context to the user's need is the key concept in the term context-awareness.

This definition does not make any assumption on how the information about the context is actually acquired. However, to get information about the context, a context-aware system needs to derive information from multiple sources, i.e., it needs to be ubiquitous. This leads to the term Ubiquitous Computing, which *"integrates computation into the environment, rather than having computers which are distinct objects"* [8]. According to [9], it emerged from the field of distributed and mobile computing, using context to determine the best possible adaptation to a user's need.

A term closely related to ubiquitous computing is Ambient Intelligence (AmI). Ambient Intelligence is not clearly defined by ISTAG [10], the organization that originally coined this term. However, referring to Wikipedia, it is defined as follows: *"The concept of ambient intelligence or AmI is a vision where humans are surrounded by computing and networking technology unobtrusively embedded in their surroundings"* [11]. Using the term intelligence, Ambient Intelligence defines a general quality that should be reached by obtaining context information and applying adaptations in distributed systems: A sensible collaboration of the involved nodes for the benefit of the user.

Since our research is done in the area of ambient intelligence, we do need to take into account that a CBR module/node needs to be able to integrate itself into a distributed AmI environment.

3 Problems and Idea

We are using CBR to build a monitoring and assistance component of an ambient system. The nature of our scenario leads to rapid changes in the context – e.g., a monitored person might change the room during the application of a multi-step treatment, which requires that reminders now get activated in the room to which the person moved.

We have identified the following three major problems to be addressed when using CBR to build context-aware ambient intelligence systems:

1. *Instability of context:* Whenever context changes, the impact of such changes needs to be taken into account for reasoning. Thus, developers of CBR systems need to consider whether this change has an impact on the operation of their CBR system. Some part of the context information relevant for adapting actions is only known right before the execution of an action. Therefore, the "lifecycle" of context information needs to be taken into account, i.e., how quickly some parts of the context usually change.
2. *Raw sensor information and uncertainty:* Context information in AmI systems comes from sensors. These sensors might not be able to provide clean information, i.e., their data might need to be pre-processed to be usable by a CBR module. Furthermore, some part of the context information might have a degree of uncertainty, e.g., due to tolerances or drifts of the sensors.
3. *Distribution of the system:* In order to enable dynamic integration of unknown nodes into ambient intelligence systems at runtime, the knowledge should be stored with the node that it is related to.

Our idea is to use two modularization principles for separating concerns in the CBR system:

Principle 1: The reasoning is modularized by the stability of the context. Here we distinguish only between stable context and rapidly changing context. *Stable context* is likely to be stable for one run of the CBR cycle. *Rapidly changing context* is likely to change between the creation of a plan (i.e., some actions) and its execution over time. This leads to two reasoning modules (**Figure 1**): First, the stable context selects abstract actions. These *abstract actions* describe actions in a manner that is independent from the rapidly changing context. Second, right before its execution, each abstract action is adapted to the rapidly changing context, which results in the *concrete actions* that are then executed in/on the real world.[2]

[2] If the stable context changes, this is subject to re-planning. Since this re-planning is subject to "traditional" planning methods in AI, we do not address this in the remainder of this paper.

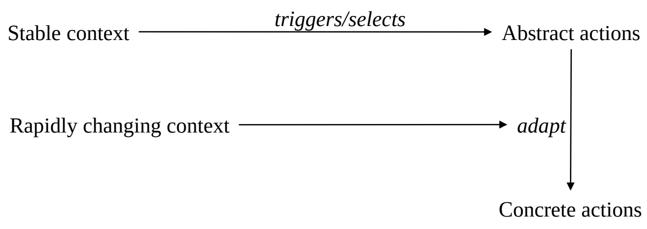

Figure 1: Modularization of reasoning according to context stability

Principle 2: The knowledge itself is modularized to reflect the distribution of the system. This means that the knowledge for ‘adapt to rapidly changing context’ is put into the respective adaptation module.

Both principles impact the CBR process. Principle 2 also impacts the system architecture and knowledge representation. These are presented in the following sections.

4 A Context-Aware CBR Process

We use the 4Re CBR cycle of [3] as a theoretical basis for our context-aware CBR process. This 4Re CBR cycle has the underlying assumption that the values used for selecting a certain case remain stable for the whole CBR cycle. In classical CBR systems, the information used for retrieval and adaptation is stable for the whole cycle (e.g., because it is entered by a user).

To solve the instability-of-context problem, a rather naïve implementation would be to integrate the volatile context information into the retrieval vector of the case. However, this volatility might cause frequent checks throughout the cycle to find out if the context remains stable. With the variety of sensors and actuators, this could also lead to a large number of cases, because cases might contain sensor- and actor-specific information. Furthermore, since each case would cover a very specific situation, the effort for aggregation and maintenance would be high.

To overcome such deficiencies, we modified and extended the 4Re CBR cycle as shown in **Figure 2** and described in the following.

- *Retrieve:* During retrieval, the most similar cases are selected based on the retrieval vector. However, if context data volatile during execution time is taken into account, this data needs to remain stable until a case is selected (and applied). Otherwise, it needs to be checked in every phase whether or not the information used for retrieving cases has changed. If this information has changed, the impact on the retrieval of cases has to be determined and is has to be decided whether the currently selected case is adapted accordingly or whether the whole CBR cycle is aborted. Thus, only stable context information should be used during retrieval.
- *Reuse*: During reuse, the case is adapted to fit to the current situation. However, if context data volatile at runtime influences behavior, adaptation needs to be performed at the execution time, i.e., right before the application of a certain case. Therefore, parts of the adaptation might be postponed until the revise phase.

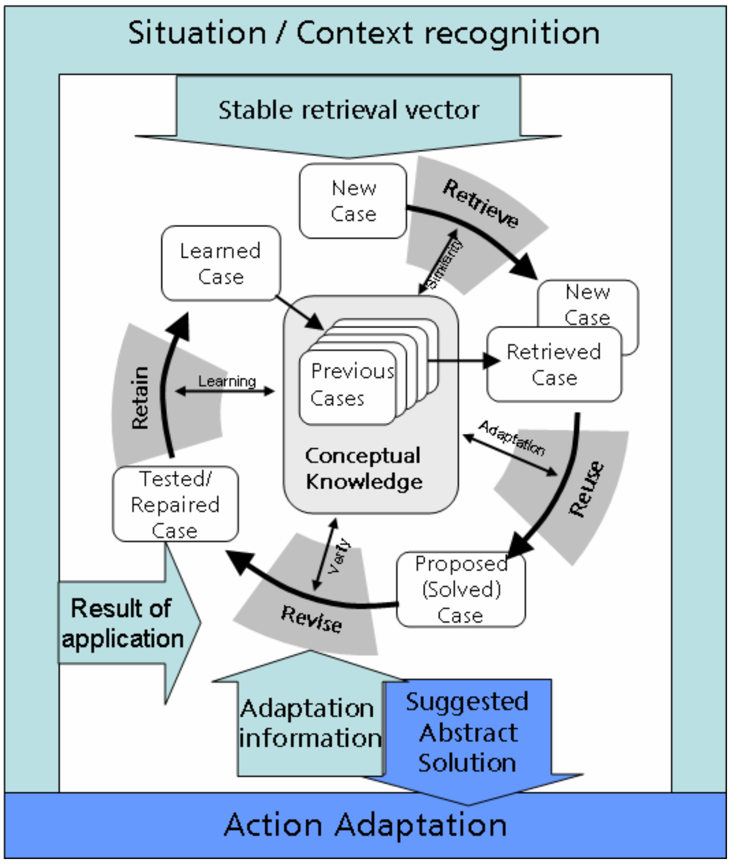

Figure 2: CBR Cycle, according to [3]

- *Revise*: In an AmI system, the case outcome (i.e., success or not, etc.) is determined using sensor information. Due to the uncertainty/noise issue of sensor information, the case outcome is also determined with some degree of uncertainty.
- *Retain*: During retain, the case is stored in the case base, with an indicator of whether it was successful or not. For context-aware systems, it needs to be taken into account to which degree the adaptation performed during the reuse and revise phase has an impact on the success of cases. Furthermore, it needs to be checked if the information used for retrieval has been changed and if this had an impact on the success as well, because then the case outcome has to be related to the new context or even to a description of the change of the context over time.

This modification of the 4Re cycle defines a lazy strategy for coping with rapidly changing context: By delaying the adaptation of cases to the latest point in time possible, the likelihood of changes and thus adaptation cost is reduced. However, this lazy adaptation might not be suitable when the adaptation costs are low.

To conclude, there are two implications that have a major influence on an architectural design:

- The information used for retrieval should be as stable as possible to avoid replanning at least for the retrieval and reuse steps. Therefore, actions are represented on an abstract level, which is adapted by a separate component right before the execution of the action.
- The CBR application has to rely on information from other components to derive the success of a case. Furthermore, it has to take into account the adaptation of actions performed by other components.

5 Service-oriented Architecture for Context-Aware CBR Systems

Based on these considerations and the above two principles, we have developed the architecture depicted in **Figure 3**. CBR-based Action Selection is responsible for selecting actions and for determining their success. Information about the current context is provided by the Situation/Context recognition component. The adaptation of actions is performed by the component Action Adaptation.

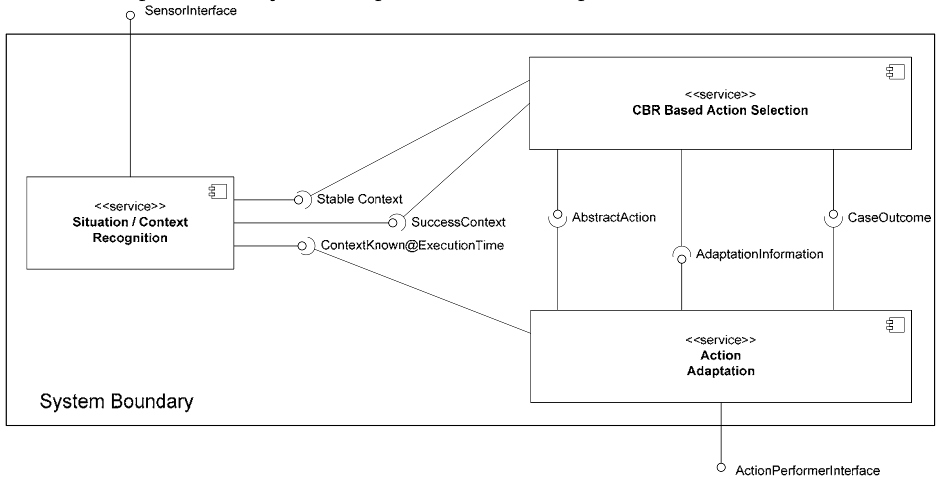

Figure 3: Service-oriented Architecture for Context-Aware CBR

These components are connected via the interfaces that deal with the information exchange needed for the steps of the CBR cycle:

- *StableContext* is used in the Retrieval and Reuse step. It contains the information relevant for selecting cases, which is assumed to remain stable during one cycle.
- *AbstractAction* is an interface for the component Action Adaptation to receive the action to be adapted.
- *ContextKnown@ExecutionTime* provides the necessary information for adaptation right before the execution of an action.

- *AdaptationInformation* is an interface to provide information on how the action was adapted. This information is needed in the Revise step to evaluate the impact of the adaptation on the success of an action.
- *CaseOutcome* provides information on whether the abstract action was performed successfully to allow the adaptation service.
- *SuccessContext* is used in the Revise step to evaluate whether the case outcome was successful or not.

The interfaces SensorInterface and ActionPerformerInterface provide the access to the sensors and actuators of the context-aware system.

6 Knowledge Representation

In this section, we describe the impact on the knowledge representation. We describe this using Richter's four CBR knowledge containers, i.e., vocabulary, similarity model, adaptation knowledge, and cases [12]:

- The *vocabulary* (attributes, predicates etc.) comprises the domain model.
  The main influence on this knowledge container imposed by context-aware application is the structure of the context information itself. The work of [9] gives an overview of how context information can be represented, while the different aspects of the context information defined by [6] provide a general structure for this container. In addition to the context information, an overview of the applicable action is needed.
- The *similarity measures* are used for comparing cases with queries.
  In context-aware applications, the similarity value determines how well a case fits to a certain context / situation. The main influence on the similarity measures in context-aware applications is how the user preferences are taken into account. We identified several possibilities of how those user preferences can be taken into account. The first one is to specify a *weighting* of attributes based on the user's preferences. This weighting is based on the evaluation of which values are better for a certain attribute than other values. This approach has the advantage that it can be easily integrated into CBR systems.
  The second possibility for taking user preferences into account is to influence the *similarity on an attribute* level. For example, a user with good visual capabilities might specify that visual information is almost equal to acoustic information, while a visually impaired user – which is common in our assisted living - will state a lower similarity.
  Finally, the user might specify preferences for certain situations (e.g. when he is asleep). This can be implemented in the similarity functions in two ways: First, filters can be used for selecting cases only appropriate for a certain situation. However, for each new situation to be differentiated, new preferences need to be stated. Second, the situation of the user might be integrated into the similarity function by stating the similarity of the situation itself. This allows specifying a default situation.

- The *adaptation knowledge* accommodates past solutions to current problems.
  As pointed out above, the adaptation knowledge is separated into two parts:
  1. The adaptation of the case and its actions on the abstract level.
  2. Mapping abstract actions to concrete actions at action execution time using context information.

  Both adaptations can address user preferences on the respective levels.
- The *case base* for the main CBR cycle is kept free from sensor- and actor-specific information. Thus, the cases are at one abstraction level only, which is expected to result in a smaller case base, which is easier to maintain.

We have presented two ways of representing the knowledge required for adapting to user preferences: (1) by means of similarity measures; and (2) by means of adaptation. Richter stated that, in general, knowledge can be moved between the containers. Here, there is an exception to Richter's rule: For adapting to user preferences on the level of concrete actions, principle 2 and the architecture pattern strongly recommend that the respective knowledge is stored in the 2nd adaptation knowledge container together with the action mapping knowledge. The only exception is if this cannot be resolved by selecting from different concrete actions for an abstract action. This would mean different abstract actions and, therefore, different cases.

7 Application in Ambient Assisted Living

After the theoretical considerations implied by adding context-awareness to CBR, we will present the application example that is currently being developed for a demonstration laboratory for assisted living in the BelAmI project. Assisted living is about supporting the day-to-day lives of elderly or handicapped persons by technical means. This laboratory performs demonstrator-based research in the field of engineering ambient intelligence applications.

In the laboratory, there are various sensors that receive information about the current context of the assisted person (e.g. mobility, drinking behavior, etc.). Furthermore, it contains several devices for interacting with the assisted person and other users (e.g., caregivers and healthcare personnel).

One scenario supported by the assisted living demonstrator is to schedule and monitor treatments for the assisted person. The software component responsible for scheduling and monitoring is called MonA (Monitoring and Assistance). Within MonA, the CBR Framework INTERESTS (Intelligent Retrieval and Storage System) [13] is used for supporting the 4Re Cycle. Examples of these treatments are reminders to drink or to take a certain pill. Based on the activities of the person (e.g., no drinking activity for the last 2 hours), it issues a reminder to the person to drink. Furthermore, it tracks whether the assisted person actually drank something, i.e., whether the treatment was successful or not.

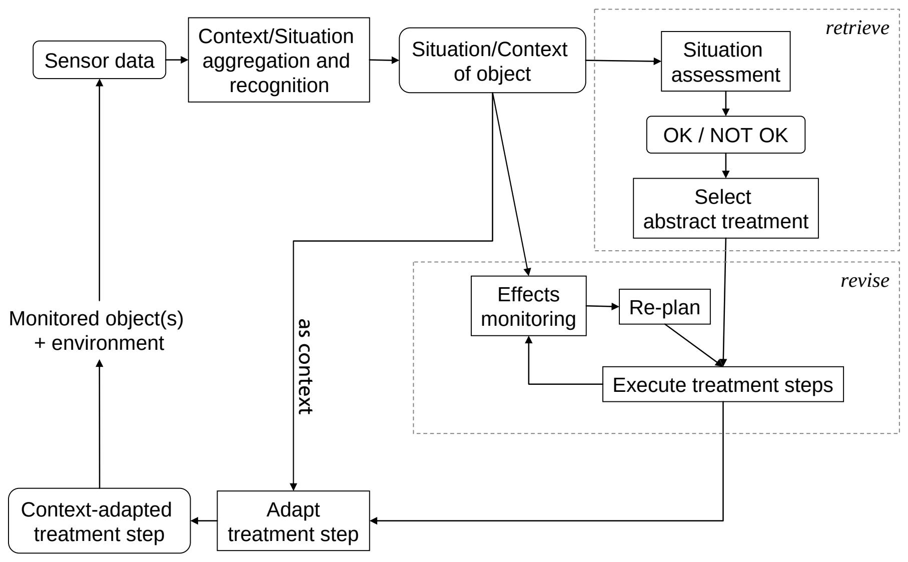
retrieve
revise
Sensor data
Context/Situation aggregation and recognition
Situation/Context of object
Situation assessment
OK / NOT OK
Select abstract treatment
Effects monitoring
Re-plan
Execute treatment steps
Monitored object(s) + environment
as context
Context-adapted treatment step
Adapt treatment step

Figure 4: Flow inside MonA. The refinements of the 4Re CBR cycle steps are marked. The actions from the generic model are "treatment steps". The mapping from the architecture pattern to the three services is straightforward.

MonA aggregates the sensor data to a situation description. The situation is then assessed as being OK or NOT OK. For situations that are NOT OK, MonA selects and abstracts a treatment description. The selected abstract treatment is then added to the treatment plan, which contains all running treatments. Furthermore, it is checked whether the selected treatment is consistent with recent and running treatments. If not, the treatment is delayed or cancelled. In accordance with its timing, the steps of the selected treatment are executed. Each treatment is adapted to the current context right before its execution in order to assure that the instantiated treatment step follows possible changes of the object or changes in its environment between retrieval and treatment application (e.g., a reminder has to be sounded in another room because the person moved from the room after the selection of the treatment). Furthermore, the effects of the treatments are monitored to check whether the expectations are met. Re-planning is done if deviations make it necessary. These operations are implemented in MonA as follows:

- The *aggregation* of the sensor data to a situation description uses a rule-based reasoning mechanism.
- Currently, the *situation assessment* is very simple: A situation is considered as being not ok if there is a respective treatment case in the case base. For more sophisticated tasks, we plan to use different AI methods for situation assessment and treatment to be selected in order to safely assess the situation before selecting the best matching treatment.
- The *consistency check* for the treatment will be implemented using a constraint resolver and/or a failure memory (as for Hammond's CHEF system [14]). The evaluation of the adequacy of these approaches is ongoing work.

- The *effects monitoring* uses the experience feedback loop method of [Nick, 2006].
- The *adapt treatment step* is subject to different services that interact with MonA.

A service that implements the "adapt treatment step" is the reminder service. It takes the abstract treatment "reminder" and selects the best way to remind a certain person. This decision takes into account the (dis-)abilities and the current situation of the assisted person. Therefore, it ensures the adaptation of the abstract activity "remind person" to a reminder issued via a concrete device.

The current version of the reminder service contains a simple set of if-then rules for activity adaptation. However, a research direction we are currently investigating is to also perform this adaptation of the reminders based on CBR.

Using CBR for reminder adaptation has several advantages: First, the current if-then rules can be easily transferred into cases. The condition mentioned in the if-part will become the retrieval vector, while the actions specified in the then part become the actions performed. This allows creating an initial set of cases easily. Second, the abstract treatment in which the reminder was issued can be stored within the case via a simple reference.

In this scenario, the following interfaces between the two services are needed:

- Reuse Steps MonA to Revise Step Reminder Service: MonA needs to report the kind of reminder that should be issued (e.g., emergency, gentle reminder) to allow the reminder service to determine an adequate way of reminding. The reminder service then reports which kind of adaptation it performed by providing a reference to the case that was selected in the reminder service. This information is needed for MonA to evaluate the success of the treatment, which might depend on the adaptation performed by the reminder service.
- Retain Step Mona to Retain Step Reminder Service: When the user does not confirm that he was informed by a message, the reminder service needs to know from other sources whether the reminder was successful or not. MonA reports whether the treatment – whose ID was handed over in the first interface – was successful or not. Based on the assumption that a reminder was successful when the underlying treatment was performed successfully as well, the reminder service can deduce which kinds of reminders are more successful than others. The reminder service stores the number of successful applications for each case.

8 Related Work

The overview of related work is divided into three sections: First, we give an overview of the related work in the area of context-aware systems for elderly people. The second part provides an overview of Open Source infrastructures for context-aware systems that can be used by CBR systems. Finally, we present an overview of standards to specify context. For a general overview of the application of artificial intelligence systems applied in the area of elderly persons, refer to [15].

Examples of context-aware systems for elderly people are:

- House_n project [16] uses wearable biometric sensors and cameras to detect symptoms of congestive heart failures (CHF) and suggest treatments (e.g., exercises)
- LiveNet [17] also uses werable biometric sensors to stream bio-signals to remote caregivers.
- Intel has developed CareNet [18], an ambient display to show the relationships between elderly persons, family and other caregivers.
- The inHaus (intelligent House) [19] provides a testbed for building automatization, providing service functions that are also useful for elderly persons.

In addition, we performed a survey about open source frameworks for context-aware systems. These systems are currently being investigated further in order to find out to which degree they can support the future development of our applications:

- The context toolkit [20] described in [21]. The toolkit itself is implemented in Java, but has connectors to .Net and Flash.
- ContextFabric [22], described in [23], provides a store for context data, a specification language for context, and protection mechanisms for privacy needs. It is written in Java.
- OpenCMA by [24] is a project under development that has not released any files yet.

9 Conclusion and Outlook

We have presented a model for CBR systems that adapt to context changing within a single run of a CBR cycle. Our model extends/modifies the existing 4Re CBR cycle of [3] and the CBR knowledge container model of [12] with a separate module for adapting abstract actions to concrete actions and the respective adaptation knowledge. The technical feasibility of the model has been demonstrated with modules of an ambient intelligence system in the area of ambient assisted living for elderly people.

Our future research will focus on two directions: First, we will investigate architectural patterns for distributing the CBR cycle in service-oriented architectures. The basis for this investigation are the experiences gained in the distribution of services in our application example. It is of particular interest to us to define concrete interfaces for the exchange of information needed during the CBR cycle and to examine under which circumstances a push or pull strategy for information about context changes is better.

On the application level, we will extend the functionality of MonA and the reminder service. For MonA, our plan is to perform constraint-based scheduling of treatments. For the reminder service, we will further investigate the CBR-based selection of reminding actions.

References

1. Bartsch-Spörl, B., M. Lenz, and A. Hübner, *Case-Based Reasoning-Survey and Future Directions.* XPS-99: Knowledge-Based Systems-Survey and Future Directions. Proc. of the 5th German Biennial Conference on Knowledge-Based Systems, Springer Verlag, 1999: p. 67-89.
2. Kolodner, J., *Case-based reasoning*. 1993, San Mateo: Morgan Kaufmann Publishers, Inc.
3. Aamodt, A. and E. Plaza, *Case-Based Reasoning: Foundational Issues.* Methodological Variations, and System Approaches, AICOM, 1994. **7**(1): p. 39-59.
4. Dey, A.K. and G.D. Abowd, *Towards a Better Understanding of Context and Context-Awareness.* 1999.
5. Wikipedia, *Context Awareness*. 2006.
6. Dey, A. and G. Abowd, *Towards a Better Understanding of Context and Context-Awareness,* GVU Technical Report GIT-GVU-99-22. College of Computing, Georgia Institute of Technology,(1999).
7. Baldauf, M. and S. Dustdar, *A survey on context-aware systems.* International Journal of Ad Hoc and Ubiquitous Computing, 2004.
8. Wikipedia, *Ubiquitious Computing*. 2006.
9. Strang, T. and C. Linnhoff-Popien, *A Context Modeling Survey.* Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp 2004), 2004.
10. IST_Advisory_Group, *Ambient Intelligence: from vision to reality*.
11. Wikipedia, *Ambient Intelligence*. 2006.
12. Richter, M.M. *The knowledge contained in similarity measures*. in *First International Conference on Case.Based Reasoning, ICCBR'95,*. 1995. Sesimbra, Portugal.
13. Nick, M., *Experience Maintenance through Closed-Loop Feedback*. PhD Theses in Experimental Software Engineering. Vol. 16. 2005, Stuttgart: Fraunhofer IRB Verlag.
14. Hammond, K., *Case-based planning - viewing planning as a memory task.* 1989, San Diego, CA, USA: Academic Press Professional, Inc.
15. Pollack, M.E., *Intelligent Technology for an Aging Population: The Use of AI to Assist Elders with Cognitive Impairment.* AIII, 2005.
16. Intille, S., *Designing a home of the future.* Pervasive Computing, IEEE, 2002. **1**(2): p. 76-82.
17. Sung, M. and A. Pentland, *LiveNet: Health and Lifestyle Networking through Distributed Mobile Devices.* Proceedings WAMES, 2004.
18. Consolvo, S., P. Roessler, and B. Shelton, *The CareNet Display: Lessons Learned from an In Home Evaluation of an Ambient Display.* Proceedings of the 6th Int'l Conference on Ubiquitous Computing: UbiComp, 2004. **4**.
19. Anonymous, *InHouse Homepage*. 2006.
20. Dey, *Context Toolkit Homepage*. 2006.
21. Dey, A., G. Abowd, and D. Salber, *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications.* Human-Computer Interaction, 2001. **16**(2, 3 & 4): p. 97-166.

22. Hong, J., *Context Fabric Homepage*.
23. Hong, J., *The context fabric: an infrastructure for context-aware computing.* Conference on Human Factors in Computing Systems, 2002: p. 554-555.
24. Forslund, D., *OpenCMA Homepage*. 2006.