A Case Study into the Development Process of Cyber Physical Systems

Wesam Shanaa, Steven Spier, Bastian Tenbergen

Department of Computer Science
State University of New York at Oswego, United States
wshanaa@oswego.edu, sspier@oswego.edu, bastian.tenbergen@oswego.edu

Abstract. [**Context and motivation**] For the past several years, Cyber Physical Systems (CPS) have emerged as a new system type like embedded systems or information systems. CPS are highly context-dependent, observe the world through sensors, act upon it through actuators, and communicate with one another through powerful networks. It has been widely argued that these properties pose new challenges for the development process. **[Question/problem]** Yet, how these CPS properties impact the development process has thus far been subject to conjecture. An investigation of a development process from a cyber physical perspective has thus far not been undertaken. **[Principal ideas/results]** In this paper, we conduct initial steps into such an investigation. We present a case study involving the example of a software simulator of an airborne traffic collision avoidance system. **[Contribution]** The goal of the case study is to investigate which of the challenges from the literature impact the development process of CPS the most.

Keywords: Cyber Physical System, Model-Based Requirements, Simulation, Traffic Collision Avoidance, Safety-Critical Embedded System, Case Study

1 Introduction

Cyber Physical Systems (CPS) blur the line between traditional system types. On the one hand, CPS observe the environment through sensors and act upon it like Embedded Systems [1]. On the other hand, they communicate through powerful network interfaces like Service-Based Systems and interact with human users like Information Systems [2]. The individual systems are not necessarily new and may be systems that are already in everyday application [3]. What makes these systems cyber physical is that they collaboratively achieve goals any individual system cannot achieve [4].

Consider the home health network example in [2]. Sensor devices (e.g., heart rate sensor, blood glucose meter, etc.) are distributed in a home of a patient to observe vital signs. If one of the sensors detects a medically concerning reading, the CPS network can automatically contact emergency services and submit vital signs to the dispatched emergency crews. In this example, the medical sensors and their networking ability build on existing technologies, yet the functionality of monitoring, collecting, and transmitting the relevant medical information is a cyber physical functionality.

In this example, the individual CPS that make up the CPS network each have a specific role as part of the home health network. However, as outlined in [4], CPS may also form dynamic networks at runtime. In [4], this is demonstrated by means of a smart automotive adaptive cruise control systems that work together to resolve traffic jams. These systems may be by different vendors, older or newer revisions, and offer additional functionality that can be integrated to resolve the traffic jam. Moreover, cars may drive away and leave a CPS network to form new networks with other cars. Potentially, the traffic jam may also include vehicles that are not equipped with a smart adaptive cruise control. In summary, the composition of the CPS network can change at runtime and in unpredictable ways.

These dynamic and heterogeneous properties of CPS networks must be accounted for during the development of any individual system that becomes part of a CPS network. This means that the development process of traditional systems, such as adaptive cruise control systems, or blood glucose meters, must take on a cyber physical perspective. In the past several years, existing approaches have been applied or extended and novel approaches have been proposed to aid a cyber physical development process. Such approaches encompass many areas of software engineering (e.g., security engineering, [5, 6], requirements engineering [7], or safety engineering [8]) and many application domains (e.g., automotive systems [9], Industry 4.0 [10], or energy systems [11]). These approaches have in common that they assume that challenges for the software development process of CPS exist, but a systematic investigation of these challenges has thus far not been undertaken.

In this paper, we take initial steps to investigate the challenges faced during software development of Cyber Physical Systems. We present a case study of the development of a simulator of an airborne Traffic Collision Avoidance System (TCAS). This paper is structured as follows: The following Section 2 reviews the related work on challenges in the development of CPS and outlines CPS properties that are subject of investigation in the case study. Section 3 gives a brief overview over the TCAS systems and motivates why it is a suitable example of a CPS. Section 4 discusses the case study design and the research questions that are based on the challenges from Section 2. Section 5 discusses the results of the case study with regard to the research questions from Section 4 before Section 6 concludes the paper.

2 Related Work: Properties and Challenges for Development

Several authors have defined overlapping properties of CPS. Some authors place emphasis on the connected nature of CPS or focus on the cloud-like aspect of their networks (like Systems of Systems, see e.g., [9, 12]). Other authors focus on the interaction between CPS and human users (see, e.g., [13, 14, 16]) or the control of real-world processes (like embedded systems, see e.g., [10, 11, 15]). In this paper, the focus of investigation lies on the properties of CPS networks. According to [4, 12], these properties can be categorized into orthogonal dimensions: static vs. dynamic and heterogeneous vs. homogeneous CPS networks, as summarized in Table 1.

Table 1. Properties of CPS Networks based on [4, 12].

	Static CPS Networks	**Dynamic CPS Networks**
Homogeneous	The CPS network is composed of a fixed number of individual CPS and each CPS has a known feature set. In this case, the CPS network is akin to Multi-Agent Systems [17].	The CPS network forms new connections at runtime with nodes posessing a known feature set. Dynamic allocation of nodes to a CPS network is akin to artificial hive intelligence systems [19].
Heterogeneous	The CPS network is composed of a fixed number of individual CPS, but the individual CPS devices have different feature sets. In this case, the CPS network is akin to Systems of Systems [18].	The CPS network forms new connections at runtime with nodes posessing an unknown feature set. Dynamic allocation of nodes to a CPS network is akin to Service-Oriented Architectures [20].

In Table 1, the term *"homogeneous"* implies that CPS have the same externally observable features, but does not imply that all CPS are the same type. Instead, individual CPS can be developed by a different vendor, etc. All four categories have in common that the CPS within the network collaboratively achieve a common goal. In contrast to Multi-Agent Systems [17] and Service-Oriented Architectures [20], these objectives must not necessarily be known at development time. This implies that in contrast to artificial hive intelligence systems [19] and Systems of Systems [18], where the role of each node is predetermined, the role of a CPS can only be assumed during development and may change at runtime. Changes in runtime pose novel challenges to the software development of these systems.

In the past several years, a plethora of papers have been published discussing these challenges (e.g., [2-4, 9, 11, 14, 16, 21, 23, 24]). The following Table 2 lists some of these challenges. For the sake of providing an overview, several similar challenges reported in different papers were condensed into one. Challenges that do not pertain to the properties outlined in Table 1 have been omitted (e.g., social challenges [2], application-domain specific challenges [11], or hardware challenges [22]).

Table 2. Challenges for CPS Networks Proposed in the Related Work.

ID	**Challenge**	**Description**	**Source**
1	Uncertainty due to Open Contexts	CPS operate in the real-world, with an indeterminate set of entities and context properties influencing their behavior, which are monitored or influenced by the CPS.	[2],[3],[4]
2	Behavioral Adaptability and Predictability of Behavior Adaptation	Heterogeneous CPS networks may be composed of CPS of different vendors, newer and older revisions, or CPS with limited compatibility. The CPS network must adapt to changes in the context as well as it's own composition and predictably adapt their behavior.	[2],[4],[9], [11],[21],[24]
3	Sensing Human Intent, Information Exchange, Human-in-Loop Control	CPS must be aware of human intentions, provide adequate information depending on the current operational situation, and adapt to changes in both.	[13],[14], [16],[21]
4	Requirements Elicitation	Existing requirements elicitation approaches may not scale up to elicit the requirements for CPS networks comprising a very large number of nodes.	[16],[23]
5	Requirements Modeling and System Modeling	Approaches to graphically or formally model artifacts of individual CPS and CPS networks must account for changes in network composition, uncertainty due to open contexts, and CPS collaboration.	[2],[4], [11],[24]

6	Deployment and Maintenance	Deploying large CPS networks is burdened by the number of nodes in the network. Even for autonomous CPS, network maintenance requires physical access, which does not scale up with large CPS networks.	[21],[23]
7	Validation/Verification of Behavior	Runtime adaptation of CPS network topology and of CPS network behavior must be validated and verified with regard to the operational purpose of each CPS, the common goal of the CPS network, and human intentions.	[3],[4],[14], [16],[21],[24]
8	Failure Robustness, Security, and Safety	Not only must each CPS be robust to failures and not pose harm to other CPS or human users, but the cooperative functionality of the network must be reliable and safe also. Moreover, the CPS network must be secure from unauthorized manipulation.	[2],[3],[4], [9],[16],[21]

As can be seen from Table 2, there are two classes of challenges. Challenge 1, 2, and 3 pertain to the dynamic nature of CPS networks during operation that must be accounted for during software development. The remaining Challenges 4 through 8 pertain to the development disciplines that must be tailored to account for CPS adaptation as well as for large scale networks.

3 Case Example: Traffic Collision Avoidance System (TCAS)

The purpose of this paper is to investigate the development process of an individual CPS, which becomes part of a CPS network during operation. In this section, we describe the case example, which was the artifact to be developed in the case study, and we discuss the rationale for selecting the TCAS as the case example.

3.1 TCAS Abstract Functionality

The case example is a Traffic Collision Avoidance System (TCAS). TCAS is an avionics system designed to prevent mid-air collisions between aircraft in flight. To do so, TCAS detects other aircraft (i.e., *"traffic"*) within a certain distance in front of the own aircraft, alerts the pilots through *"traffic advisories"*, and produces *"resolution advisories."* Resolution advisories are recommended changes in the vertical speed (i.e. climb or descend rate) in order to increase the separation altitude between the own aircraft and the threat aircraft such that a collision can be avoided. Fig. 1 shows TCAS' simplified functionality based on [25, 26].

As can be seen, a TCAS monitors the airspace around the aircraft for traffic equipped with a corresponding active transponder ("*Mode-S Interrogation"* in Fig. 1). This allows establishing two-way communication between the own aircraft and traffic nearby to coordinate collision avoidance. If some other aircraft in the vicinity is not equipped with an active transponder (e.g., many recreational aircraft), or no transponder at all (e.g., ultralight or hostile aircraft), the TCAS uses the own aircraft's transponder and onboard radar to detect uncooperative traffic ("*Mode-C-Only All Call"*). The function *"target surveillance"* determines the flight trajectories for all traffic detected through radar contacts and transponder replies and discriminates traffic into "intruders," i.e. traffic that might pose a collision threat, and other traffic.

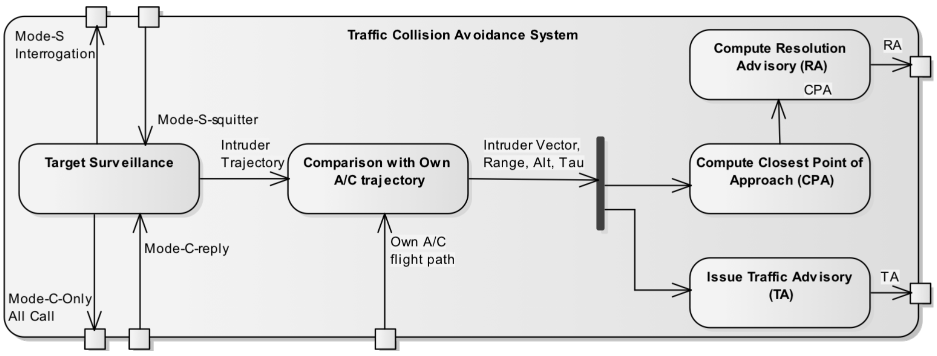
Traffic Collision Avoidance System
Mode-S Interrogation
Mode-S-squitter
Target Surveillance
Intruder Trajectory
Comparison with Own A/C trajectory
Intruder Vector, Range, Alt, Tau
Compute Resolution Advisory (RA)
RA
CPA
Compute Closest Point of Approach (CPA)
Issue Traffic Advisory (TA)
TA
Mode-C-reply
Own A/C flight path
Mode-C-Only All Call

Fig. 1. Principle Functionality of TCAS based on [25, 26].

For each intruder, and under consideration of the own aircraft's flight path, the function "Comparison with Own A/C trajectory" computes the relative approach vector to the intruder, the distance to the intruder ("Range" in Fig. 1), the intruder's relative altitude ("Alt") and the time until the intruder is intercepted ("Tau").

Based on this information, a traffic advisory ("TA") is issued. Concurrently, the TCAS computes the closest point of approach ("CPA"), i.e. the point in three-dimensional space, where the own aircraft is closest to the intruder. If it is determined that the CPA is smaller than a minimal safe separation altitude, the intruder is considered a collision "threat" and a resolution advisory ("RA") is produced through auditory alerts and cockpit instruments.

3.2 Rationale for Selecting TCAS as the Case Example

In Section 2, we have listed challenges previous authors have assumed to burden the development of static and dynamic, homogeneous and heterogeneous CPS networks and in Section 3.1, we have introduced the case example. We have selected the implementation of a TCAS simulator because, depending on the specific interactions between the TCAS of two or more aircraft, the TCAS can participate in all four CPS network types from Section 2:

- **Homogeneous vs. Heterogeneous:** A CPS network involving multiple TCAS instances is *homogeneous*, if the TCAS instances are cooperative in the sense outlined in Section 3.1, i.e. using Mode-S transponder functionality actively work together to avoid collisions. A CPS network involving multiple TCAS instances is *heterogeneous*, if at least one intruder aircraft is uncooperative, e.g., when the intruder aircraft is not equipped with TCAS functionality or a hostile aircraft.
- **Static vs. Dynamic:** A CPS network involving multiple TCAS instances is *static*, if there is a known set of aircraft, either cooperative or uncooperative, in the vicinity of the own aircraft and their altitudes must be coordinated such that no aircrafts collide. In reality, this is an atypical case. A CPS network involving multiple TCAS instances is *dynamic*, if aircraft spontaneously enter each other's air space

during flight, resulting in the need to increase separation altitude. This is the typical use case of TCAS.

It must be noted, that even when one intruder aircraft is uncooperative in the sense outlined in Section 3.1, other TCAS in the CPS network can still react appropriately such that the common goal of avoiding a collision is achieved collaboratively (unless an intruder is actively attempting a collision). It follows that the TCAS example can partake in the different CPS networks outlined in Section 2. TCAS is therefore a suitable surrogate for investigations into the development process of CPS participating in such networks.

4 Case Study Design

The purpose of the case study is to investigate the development process of a TCAS, which will form CPS networks with other TCAS during operation. In this section, we describe the design of the case study following the checklist and guidelines from [28]. The case example system and background theory on challenges in CPS development have already been discussed in Sections 2 and 3, respectively. In the following subsections, we discuss the case study context, research questions, and data analysis.

4.1 Study Context: Course, Participants, and Procedure

The case study took place as part of a software engineering seminar course at the authors' affiliation, instructed in fall 2016 by the third author. The course is geared towards advanced students with considerable knowledge in development. The course objective is for students to apply their theoretical knowledge about software development in a significant development project over the course of one semester.

Participants. There was a total of eight students enrolled in the course. Three of the students participated in the TCAS case study, while the remaining students worked on other course projects. All three participants were enrolled in an undergraduate computer science or software engineering program. As for the course prerequisite, participants had demonstrable prior experience in software development, yet no experience with CPS development. Age and gender were assumed to not impact development ability and hence were not recorded.

Requirements Phase. The course was structured over 15 weeks with three class meetings every week. Approximately half of the semester (seven out of 15 weeks) was dedicated to requirements engineering and system design. In this phase, the first class meeting each week was dedicated to lecturing on model-based requirements artifacts. Specifically, lectures included context diagrams, KAOS goal models, and UML sequence diagrams for scenario modeling (see [29]) as well as technical architecture models (see [30]), which students were instructed to use for the case study. The second weekly class meeting was dedicated to modeling examples using the technique discussed in the previous meeting. The purpose was to give students more exposure to the practical application of the respective modeling technique. The final weekly meeting was dedicated to independent work of participants on their respective tasks using the modeling techniques. Participants were free to meet individually, as a team, or with the instructor and set their own objectives for their weekly modeling task. The requirements phase consisted of four milestones. Each milestone objective was to create an initial version of a requirements model and revise the models from the previous milestone to ensure consistency and completeness. Milestones were presented weekly for review, critique, as well as instructor and peer feedback.

Implementation Phase. After the requirements phase completed, the case example implementation began. Implementation followed the agile SCRUM methodology, as given the limited timeframe of a semester, agile implementation approaches allow for quick progress in producing executable artifacts. At the onset of this phase, students were introduced to agile development and asked to derive a backlog from their requirements artifacts, which were developed a more rigid approach (i.e. [29], see above). Each week was considered a sprint, were at least one item from the backlog ought to be implemented, and the requirements artifacts were to be updated, if needed. Class meetings during the implementation phase began with a daily SCRUM meeting, where students presented their progress, current issues, and the next issue on their agenda. Class meetings were dedicated to presentation of progress, discussion of implementation strategy and technology, and assisting other case example teams.

The implementation phase concluded at the end of the semester with a graded final presentation. Participants' course grades were established by means of a departmental criteria list, which is solely based on the students' individual performance as a team member and their application of knowledge from previous courses. Participants' course grades were hence independent of project success or case study outcome. Approval from the authors' university's ethical review board was sought before students agreed to participate in the case study by providing informed consent and usage rights for their artifacts and code.

4.2 Objectives and Research Questions

The key goal is to gain initial insights into the specific challenges that arise when developing a system from a cyber physical perspective, as outlined in Section 1. To do so, the following research questions were defined:

- **RQ1: How did students execute the development process in the context of the course instruction?** Approaching system development from a cyber physical perspective was not a main focus of the curriculum in the students' degree program and may have influenced the results.
- **RQ2: Which challenges from the literature are the most relevant?** As was illustrated in Section 2, in the literature, a large amount of hypothetical challenges for the development process of CPS have been suggested. These challenges pertain to the dynamic allocation of CPS to homogeneous or heterogeneous networks. Yet, which of these hypothetical challenges impact development is to be explored.
- **RQ3: Which development discipline is impacted the most by these challenges?** Each of the challenges discussed in Section 2 may affect some development discipline more than others. For example, challenges in requirements modeling (Chal-

lenge 5 in Table 2) may impact the requirements engineering phase of development, but also implementation. For approaches to be able to address these challenges, the impact of these challenges must be known.

4.3 Case Selection and Unit of Analysis

The TCAS case example from Section 3.1 is intended to be implemented as a plugin for the X-Plane flight simulator [27]. X-Plane is a recreational flight simulation game as well as a flight instruction platform for future pilots. It features a realistic flight model based on empirical data and real-time computations. Due to an available API, X-Plane is extendable through plugins, which allow execution of arbitrary C++ code. The specific unit of analysis for the case study is the development process of a TCAS plugin for X-Plane. The research questions from Section 4.2 are investigated within the context of the requirements and implementation phases (see Section 4.1).

4.4 Data Collection and Analysis

There were three types of artifacts that were collected and analyzed: requirements artifacts milestones from the requirements phase, implemented executable aritfacts (e.g., code, test cases, configurations), and reports from the daily SCRUM meetings.

Requirements Artifact Milestones. The four milestones during the requirements phase collected through an online campus system every second week. Since a milestone consisted of revising previous versions of artifacts based on feedback and knowledge discovery, milestones were incremental. Moreover, participants were asked at the end of the implementation phase to produce a fifth milestone of their requirements artifacts to account for the changes to the artifacts necessitated during the implementation phase. During data analysis, milestones were compared to one another with regard to revisions made due to knowledge discovery, i.e. revisions to one artifact that became necessary due to progress made while working on another artifact. This means that changes due to instructor feedback were not recorded. In particular, changes due to the cyber physical properties of TCAS were recorded and categorized according to each research question (see Section 4.1).

Implementation Code. All source-code-based artifacts, which included code, third-party libraries, image artwork for TCAS instruments, configuration files, etc., were collected incrementally throughout the implementation phase using a remote repository. Participants were asked to commit their changes to these artifacts at least once at the end of each sprint. During data analysis, changes to code and code quality were not taken into account. However, progress during implementation was tracked with regard to the specific aspect that was worked on during a sprint. In particular, we took note on whether or not a particular implementation aspect pertained to traditional system properties or cyber physical properties of the TCAS. For example, when issues arose regarding networking, interface rendering, or platform-plugin interaction, this was considered an implementation issue pertaining to either traditional system properties or pertaining to the simulation platform. On the other hand, when an issue arose pertaining to establishing a connection between the own aircraft and other aircraft and subsequently discriminating other aircraft into threats, intruders, or other nearby traffic, this was considered an issue pertaining to CPS.

Participant Reports. During the daily SCRUM meetings, participants reported their progress in the previous sprint, and reported on current issues they were working on. Albeit these reports are subjective and qualitative, they became the richest source of insight into the development process, as they gave an "uncensored" account on the specific challenges students encountered during development. We use the term "uncensored" to address the notion that participants were eager to remind us of the level of difficulty and challenges they experienced with the TCAS case example system. Similarly, when a challenge was resolved, participants keenly reported the specific strategy which lead to success. For the purpose of the case study, we took particular note of challenges pertaining to the interaction of two or more TCAS systems during operation, as these reflect the cyber physical nature of TCAS.

4.5 Threats to Validity

Albeit great care was taken to design and conduct the case study, threats to validity remain. In the following, we discuss these threats, their mitigation, and open threats.

Internal Validity: The case study must be thoroughly designed to answer the research questions. We have applied the best practices outlined in [28] to design the case study. We derived research questions based on the literature discussed in Section 2 and describe the design in detail in Section 4. We provide an account of the raw artifacts and development process that we base our results on in Section 4.5. Nevertheless, the context of the study may have impacted results, as with any case study.

External Validity: The case study must be representative and allow for generalizability of results. In Section 3, we have explained the case example system in sufficient detail to allow replication in larger contexts. Moreover, we have outlined in Section 3.2, why the case example is a suitable surrogate for a CPS. Nevertheless, the case study did not develop a "real" TCAS, but was concerned with developing a simulator, leaving open questions as to the level of realism of the simulated plugin. Albeit the development process aimed at developing realistic TCAS functionality based on official regulatory documents, this threat partially remains.

Construct Validity: Measurements must be appropriate to measure results. As with many case studies, the study at hand basis its findings on qualitative observations, participant reports, and artifacts created by participants. These information sources are inherently subjective. To mitigate this threat, we resort to a mere objective report on the development process in Sections 5.1 and 5.3. Moreover, to increase confidence in our work, we make available all case study materials to researchers and enthusiasts (see footnote in Section 4.5).

Conclusion Validity: Conclusions must be free of bias. One potential source of bias is the fact that one of the authors was the instructor of the course in which the case study took place. This may have influenced case study success and participant performance. Of course, it is the third author's key interest to see his students succeed. Yet, by university policy, course grades do not depend on project success. Moreover, findings are objective reports regarding the development process, not student performance. Nevertheless, individual student performance may have impacted case study results. Lastly, the process of summarizing CPS challenges in Section 2 may have influenced conclusions regarding the impact of the cyber physical nature onto development. Albeit we defined a clear scope in Section 2 to investigate the challenges, more investigations into the development process of CPS are necessary to confirm our results.

5 Results of the Case Study

In the following subsections, we report on the findings of the case study with regard to the research questions from Section 4.1.

5.1 RQ1: How did students execute the development process in the context of the course instruction?

The case study began in the first week of the semester with participants researching the basic functionality of TCAS by studying FAA regulatory documents (e.g., [25, 26]) and other online resources[1]. Moreover, participants familiarized themselves with plugin development for X-Plane. The first milestone in the requirements phase entailed the construction of a rough context model, representing the interplay of the TCAS with the simulated components. The context model is shown in Fig. 2.

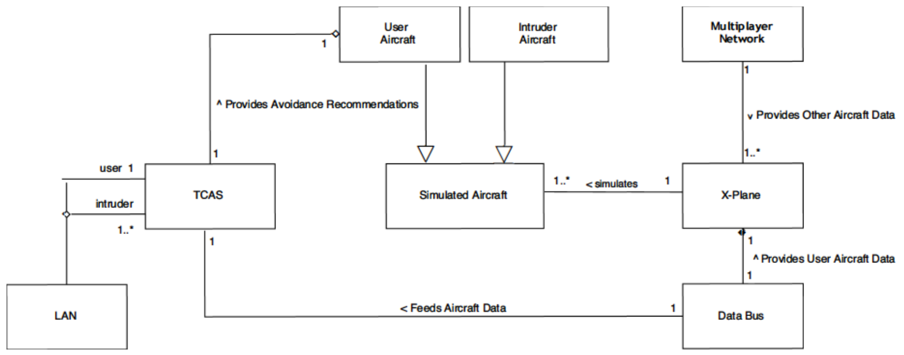

Fig. 2. TCAS Context Model depicted using UML class diagram notation.

As can be seen, the X-Plane platform simulates all aircraft. These can be either the user's own aircraft, or potential intruders. Aircraft Data for non-user aircraft are collected through a X-Plane, through the Multiplayer Network connection. The Multiplayer Network connection represents aircraft in the simulated world. This component

[1] An example of such a resource can be found on the StackExchange post by user "DeltaLima" (available at https://goo.gl/fnXOhI, accessed January 3, 2017). Albeit StackExchange is neither a reliable academic nor an official technical reference, it provided sufficient technical details for the participants to move forward with the case study.

was already existing and not part of the case study. Aircraft Data is exchanged using X-Plane's internal Data Bus, which is a collection of variables that store values pertaining to the current simulation. The TCAS is designed as a plugin for X-Plane, which makes use of the Data Bus and queries Aircraft Data such as, vertical speed, airspeed, or latitudinal and longitudinal positions. From the perspective of the user's own aircraft, it cannot be known whether or not the intruder is cooperative or not. Therefore, only the user aircraft is composed of a TCAS in Fig. 2. A Local Area Network (LAN) connection is used by the TCAS to simulate transponder and radar sweep (i.e. Mode-S and Mode-C calls in Fig. 1), in order to find the subset of simulated aircraft close to the user's own aircraft within the simulated world. This subset constitutes nearby traffic (see Section 3.1).

Based on the context model and TCAS' principle functionality outlined in regulatory documents, a goal model was developed. An excerpt is shown in Fig. 3.

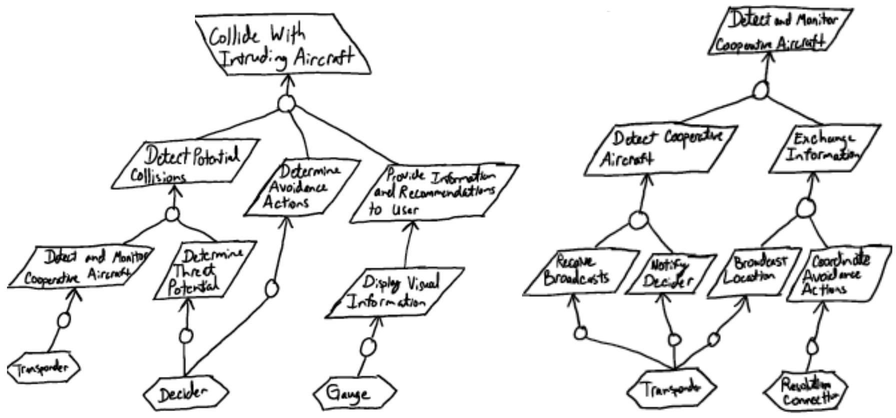
Collide With Intruding Aircraft
Detect Potential Collisions
Determine Avoidance Actions
Provide Information and Recommendations to User
Detect and Monitor Cooperative Aircraft
Determine Threat Potential
Display Visual Information
Transponder
Decider
Gauge
Detect and Monitor Cooperative Aircraft
Detect Cooperative Aircraft
Exchange Information
Receive Broadcasts
Notify Decider
Broadcast Location
Coordinate Avoidance Actions
Transponder
Resolution Connection

Fig. 3. TCAS Goal Model using KAOS notation (excerpt).

Participants began with the anti-goal "Collide With Intruding Aircraft," which is to be avoided. Following the KAOS tutorial from [31], participants refined this goal into one goal model depicting the entire, abstract functionality of TCAS (left diagram in Fig. 3). Once participants felt the goal model to be sufficiently detailed, they assigned KAOS "agents" to the goals, which were intended to represent components of the TCAS plugin (e.g., "Transponder," "Decider," "Gauge," in the left diagram in Fig. 3). Thinking about agent implementations, participants felt that some lower-level goals were still too abstract to implement. Hence, participants opted to refine the goal model further, by taking the lowest-level goals and refining them into own diagrams. For example, the right side in Fig. 3 shows the refinement diagram of the goal "Detect and Monitor Cooperative Aircraft" in the left diagram in Fig. 3. This process iteratively continued until each goal was assigned to an agent (e.g., "Transponder and "Resolution Connection" in the right diagram in Fig. 3) and subjectively, goals were sufficiently detailed.

Once goal models were complete, scenario models were constructed. Fig. 4 shows two example scenarios.

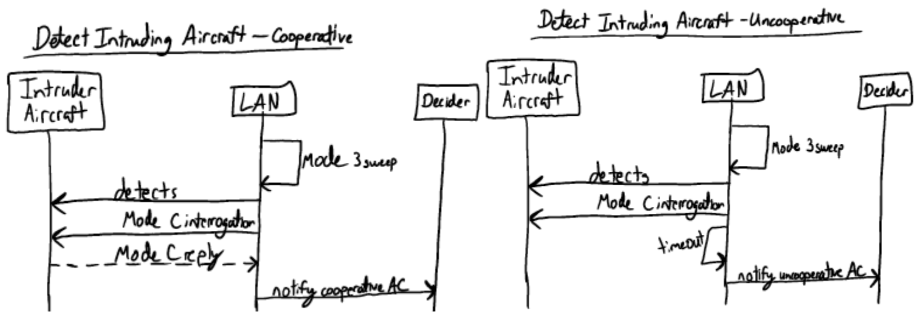

Fig. 4. Initial TCAS Interaction Scenarios with Cooperative and Uncooperative Intruders.

As shown in Fig. 4, scenarios were created depicting the conceptual interaction between TCAS components. Following the approach outlined in [29], at least one scenario was created for each goal, showing its exemplary fulfillment. For example, the scenario on the left in Fig. 4 shows the exemplary interaction between components to fulfill the goal "Detect Cooperative Aircraft" from Fig. 3 (right goal diagram), which also is intended to fulfill the goals "Receive Broadcast" and "Notify Decider." Doing so resulted in the identification of missing goals (e.g., a corresponding goal for handling uncooperative intruders, see right diagram in Fig. 4), which triggered iterative revisions of goal models and scenario models.

In total, participants produced more than 20 goal, scenario, and context diagrams, which were iterated and refined weekly and more than 5,000 lines of code. Due to space limitations, these artifacts cannot be fully discussed here. Therefore, we are making these artifacts available[2] to interested researchers and enthusiasts.

5.2 RQ2: Which challenges from the literature are the most relevant?

The case study was predominantly impacted by challenges pertaining to the interaction between the user's own aircraft and intruder aircraft. In particular, a considerable challenge was to differentiate simulated aircraft close to the user's aircraft from those that are far away, e.g., on another continent, but within the simulated world. This challenge arose due to the specific architecture of the simulated aircraft: The Multiplayer Network connection does not discriminate between close and far away aircraft, making it necessary for the TCAS simulator to make this differentiation before regular TCAS functionality can be carried out. Real-life TCAS identify nearby aircraft through the transponder and radar, making this challenge unique to the case study. Moreover, once nearby traffic was identified, only cooperative intruders could be reliably identified (i.e. when they actively send a Mode-S reply, see Fig. 1). For uncooperative intruders, it can only be heuristically established through Mode-C replies and the absence of a Mode-S reply.

[2] https://goo.gl/kjvyw

This introduced uncertainty (Challenge 1 in Table 2) considering the specific makeup of the CPS network: in previous literature, it was assumed that CPS are designed to either be part of a homogeneous or heterogeneous network. Yet, as we learned through the case study, the network makeup in this sense cannot be reliably established and may change at runtime. Especially the fact that the CPS network makeup changes during runtime requires the TCAS to react appropriately, necessitating predictable behavioral adaptation (Challenge 2). For example, when the own aircraft is to avoid a non-cooperative intruder, a decision must be made to climb or descend to increase separation altitude. In the case study, a Decider component was created, which issues "climb" or "descend" RAs depending (based on heuristics from [25, 26]). However, the question came up, whether or not the reason why a "climb" or "descend" RA was issued needed to be communicated to the pilots (Challenge 3). Similarly, if the pilots do not follow the resolution advisories (e.g., climb, even though the RA indicates to descend), issues arose during development of how to make the intruder aircraft aware such that a collision can be avoided.

5.3 RQ3: Which development discipline is impacted the most by these challenges?

The development process largely resembled that of an embedded system. However, uncertainty and behavioral adaptation (Challenges 1 and 2) mainly impacted requirements engineering phases of development (Challenges 4 and 5) as well as the quality assurance (Challenges 7 and 8), while the implementation phase was nearly equivalent to previous projects from the embedded systems domain. Specifically, uncertainty about the specific type of the CPS network constituents (i.e. homogeneous or heterogeneous CPS) required frequent revisions of the model-based requirements artifacts (Challenge 5). For example, frequent changes to scenario models necessitated by different interactions motivated changes in the goals. It is interesting to note that changes in requirements models were preceded by frequent consultations of the context model (i.e. Fig. 2). While the context model was only rarely adapted, the information documented therein lead to fruitful discussions about the interaction between the user's own aircraft and a possible intruder, which guided the implementation.

The interaction of several cyber physical TCAS impacted deployment (Challenge 6), verification (Challenge 7), and functional robustness (Challenge 8) also. After unit tests were complete, verification included testing the TCAS functionality using two computers, each running an X-Plane instance, to which the compiled TCAS plugin was deployed. Initially, participants did not account for testing TCAS functionality using different versions (e.g., the latest build together with a TCAS that is a few revisions old). Instead, when a new version was compiled, the new revision was deployed to both test computers. During later development stages, when frequent bug fixes and redeployment took place, this process was streamlined to deploying merely onto one computer. This introduced a kind of CPS heterogeneousness into the CPS network that was discussed in [4]: the CPS network consists of nodes with the same functionality, but different versions. This type of heterogeneousness was unanticipated throughout the case study and in many cases resulted in failure of TCAS' functionality. Specifically, intruders were not recognized reliably or RAs were computed incorrectly. The result was impaired functional robustness (Challenge 8), which burdened quality assurance in late stages of development.

6 Discussion and Outlook

In this paper, we presented a case study into the development process of homogeneous and heterogeneous, dynamic Cyber Physical System networks. The case example system of an airborne Traffic Collision Avoidance System was chosen to investigate which challenges previously suggested in the literature impact the software development process of a CPS. Case study results show that the most significant challenges pertain to runtime uncertainty of CPS network configurations, predictable behavioral adaptation, and human-in-the-loop control. The most significantly impacted development disciplines are requirements engineering, quality assurance, and deployment.

The case study at hand provides merely an initial, exemplary insight into the development process of a system from a cyber physical perspective. It is not as a definitive investigation into the challenges underlying CPS development in general. Additional empirical investigations are needed to investigate the challenges of CPS development in more detail and with more robust evidence. This is subject of future work.

We encourage repetition and extension of our research. Therefore, we make available the current implementation of the TCAS simulator along with its development artifacts (see Section 5). The current status of the TCAS simulator handles arbitrary TAs and RAs for the closest threat.

Acknowledgments. The authors thank Laminar Research for making available their X-Plane simulator platform, their industry partners for their consultation regarding TCAS, and our participants. The project is sponsored in part by the Research Fund for Chemistry and Math/Science.

References

1. Wolf, W. (2009) Cyber-physical Systems. IEEE Comp. 42(3).
2. Broy, M.; Cengarle, M.; Geisberger, E. (2012) Cyber-Physical Systems: Imminent Challenges. Monterey Workshop.
3. Lee, E. (2008) Cyber Physical Systems: Design Challenges. 11th IEEE Symp. Obj. Oriented Real-Time Distr. Comp.
4. Daun, M.; Salmon, A.; Tenbergen, B.; Weyer, T. (2015) Today's Challenges and Potential Solutions for the Engineering of Collaborative Embedded Systems. 2nd Intl. EITEC WS.
5. Pasqualetti, F.; Dörfler, F.; Bullo F. (2013) Attack Detection and Identification in Cyber-Physical Systems. IEEE Trans. on Automatic Control 58(11).
6. Fawzi, H.; Tabuada, P.; Diggavi, S. (2014) Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks. IEEE Trans. on Automatic Control 59(6).
7. Daun, M.; Brings, J.; Weyer, T.; Tenbergen, B. (2016) Fostering Concurrent Engineering of Cyber-Physical Systems – A Proposal for a Context Framework. 3rd Intl. EITEC WS.

8. Battram, P.; Kaiser, B.; Weber, R. (2015) A Modular Safety Assurance Method considering Multi-Aspect Contracts during Cyber Physical System Design. 1st Intl. RESACS WS.
9. Wan, J.; Zhang, D.; Zhao, S.; Yang, L.; Lioret, J. (2014) Context-aware vehicular Cyber-Physical Systems Support: Architecture, Challenges, and Solutions. IEEE Comm. 52(8).
10. Lee, J.; Bagheri, B.; Kao, H.-A. (2015) A Cyber-Physical Systems Architecture for Industry 4.0-based Manufacturing Systems. Manufacturing Letters 3.
11. Palensky, P.; Widl, E.; Elsheikh, A. (2013) Simulating Cyber-Physical Energy Systems: Challenges, Tools, and Methods. IEEE Trans. on Systems, Man, and Cybernetics 44(3).
12. Broy, M. (2013) Engineering Cyber-Physical Systems: Challenges and Foundations. 3rd Intl. Conf. on Complex Systems Design & Management.
13. Munir, S.; Stankovic, J.; Liang, C.-J.; Lin, S. (2013) Cyber Physical System Challenges for Human-in-Loop Control. 8th Intl. WS on Feedback Computing.
14. Schirner, G.; Erdogmus, D.; Chowdhury, K. (2013) The Future of Human-in-the-Loop Cyber-Physical Systems. IEEE Comp. 46(1).
15. Wang, Z.; Song, H.; Watkins, D.; Ong, K.; Xue, P.; Yang, Q.; Shi, X. (2015) Cyber-Physical Systems for Water Sustainability: Challenges and Opportunities. IEEE Comm. 53(5).
16. Giaimo, F.; Yin, H.; Berger, C.; Crnkovic, I. (2016) Continuous Experimentation on Cyber-Physical Systems: Challenges and Opportunities. Scientific WS of 2016 XP Conf. Series.
17. Woolridge, M. (2009) Introduction to Multi-Agent Systems, 2nd Ed., John Wiley & Sons.
18. Luzeaux, D.; Ruault, J.-R. (Eds.) (2013) Systems of Systems, John Wiley & Sons.
19. Bonabeau, E.; Dorigo, M.; Theraulaz, G. (1999) Swarm Intelligence – From Natural to Artificial Systems, Oxford University Press.
20. Erl, T. (2004) Service-Oriented Architecture A Field Guide to Integrating XML and Web Services, Prentice Hall.
21. Leitao, P.; Colombo, A.; Karnouskos, S. (2016) Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. Comp. in Industry 81.
22. Bogdan, P. (2015) A cyber-physical systems approach to personalized medicine: challenges and opportunities for noc-based multicore platforms. Design, Automation & Test in Europe Conf. & Exhibition.
23. Broy, M.; Schmidt, A. (2014) Challenges in Engineering Cyber-Physical Systems. IEEE Comp. 47(2).
24. Mosterman, P.; Zander, J. (2016) Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. J Software & Systems Modeling 15(1).
25. US FAA (2011) Introduction to TCAS II, Version 7.1. Available at http://goo.gl/EPCYzI, accessed January 3, 2017.
26. US FAA (2009) Advanced Avionics Handbook. Available at https://goo.gl/S2HQ5B, accessed January 3 2017.
27. Laminar Research (2016) X-Plane, Version 10. Available at http://www.x-plane.com, accessed January 3, 2017.
28. Runeson, P.; Höst, M. (2009) Guidelines for Conducting and Reporting Case Study Reseearch in Software Engineering. Empir Software Eng 14.
29. Daun, M.; Tenbergen, B.; Weyer, T. (2012) Requirements Viewpoint. In: Model-Based Development of Embedded Systems, Springer.
30. Weber, R.; Reinkemeyer, P.; Henkler, S.; Stierand, I. (2012) Technical Viewpoint. In: Model-Based Development of Embedded Systems, Springer.
31. Respect-IT (2007) KAOS Tutorial, Version 1.0. Available at https://goo.gl/Q4pi8D, accessed January 3, 2017.