Managing risks and opportunities in cyber-physical systems with software architecture assessments

Tuovinen, Antti-Pekka

de Gruyter

2021

Tuovinen , A-P , Christophe , F , Kettunen , P , Mikkonen , T & Männistö , T 2021 , Managing risks and opportunities in cyber-physical systems with software architecture assessments . in K Engemann & R O'Connor (eds) , Project risk management : Managing software development risk . vol. 2 , Developments in managing and exploiting risk , de Gruyter , Berlin , pp. 3-24 . https://doi.org/10.1515/9783110652321-002

http://hdl.handle.net/10138/341401
https://doi.org/10.1515/9783110652321-002

publishedVersion

Managing Risks and Opportunities in Cyber-Physical Systems With Software Architecture Assessments

Antti-Pekka Tuovinen[*], François Christophe[†], Petri Kettunen[*], Tommi Mikkonen[*], and Tomi Männistö[*]

[*]University of Helsinki, Helsinki, Finland

[†]Häme University of Applied Sciences, Hämeenlinna, Finland

Abstract

With the advances in digitalization, the balance between software-related risks and opportunities is becoming a key decision, but without a thorough insight into the possibilities and liabilities of software, this is a difficult step to take. Hence, companies more commonly follow an approach where they have a linear model for product evolution, and try to avoid large-scale changes in the system as a whole. The software architecture of a cyber-physical system (CPS) is one of the main factors that determine the sustainability of the system from the point of view of development, maintenance, and evolution. However, a software architecture is not inherently good or bad, it is just more or less fit for purpose, and software architecture assessment is an effective way to establish its fitness. In this chapter, we share our experiences on using a series of software architecture assessment workshops as a mechanism to identify risks and opportunities of an existing CPS software product line and to help in planning the renewal of the software system accordingly, taking into account the evolutionary line of new features as well as potential future disruptive technologies. The assessments took place at a company that provides industrial automation solutions and that takes the usual risk-oriented view to software engineering. The factors they would like to study include feature creep, sensitivity for control points, and scaling the current product line to meet changing customer demand.

Keywords: Risk management, opportunity management, planned staged investment, software architecture assessment, cyber-physical systems.

1 Introduction

When a new generation of cyber-physical system (CPS) emerges, it is often unclear which are risks and which are opportunities within the scope of the new generation. With the advances in digitalization, the balance between software risks and opportunities is becoming a key decision, but without a thorough insight into the possibilities and liabilities of software in the system, this is a difficult step to take. Hence, companies more commonly follow an approach where they have a linear model for product evolution, and try to avoid large-scale changes in the system as a whole. Such issues have been encountered in various contexts, including in particular mobile devices, but few practical approaches have been proposed. One of those that has been used in industry is planned staged investments [1], which divides the life cycle of the product into steps of investment and harvesting. During the former, an investment is made in the system under development by introducing new features and capabilities, and by improving quality. During harvesting, software is maintained at minimum cost, and no large investments in new features or improved quality are made.

The systems architecture of a CPS sets a framework for its key qualities and structures. The software architecture is one of the main factors that determine the sustainability of the system from the point of view of development, maintenance, and evolution. However, a software architecture is not inherently good or bad; it is just more or less fit for purpose. In order to assess the fitness of a software architecture for its particular context and requirements, the architecture can be assessed using established, mature methods. A software architecture assessment (a.k.a software architecture evaluation) can also have specific goals – identifying risks when planning changes, or, considering the feasibility of further investment in a system vs. its replacement are common reasons for conducting a software architecture assessment, for example.

In this chapter, we share our experiences on using a series of software architecture assessment workshops as a mechanism to identify risks and opportunities of an existing CPS software product line and to help in planning the renewal of the software system accordingly, taking into account the evolutionary line of new features as well as potential future disruptive technologies. In terms of planned staged investments, the goal is to identify opportunities to be gained during the next planned investment period, as well as to manage risks during the ongoing maintenance period.

The rest of the chapter is structured as follows. In Section 2, we introduce the case company's CPS domain, software product lines, and software architecture assessments. In Section 3, we discuss the role of architecture assessments as a risk management tool in the context of software product lines. In Section 4, we present our case study, executed together with a company operating in the domain of cyber-physical systems in industrial automation. In Section 5, we provide an extended discussion on our findings. Finally, in Section 6, we draw our conclusions.

2 Background

The background of this work consists of three different dimensions, Cyber-Physical Systems, Software Product Lines, and architecture assessments. In the following, we introduce briefly each of them in separate subsections.

2.1 Cyber-Physical Systems

Cyber-Physical Systems (CPS) are systems that simultaneously act in the physical and digital space, comprising both physical and computational processes and involving people [2]. Typical examples of CPSs include drones, various robots, and autonomous vehicles and larger, complex systems such as Smart Grids. Since a major part in their development includes the design of physical, mechanical and electrical elements, the development has been executed under their terms and engineering disciplines and software has traditionally played only a minor role inside each device and system component independently. The situation is now changing rapidly, and software is becoming a major factor in innovation in CPSs [3] [4]. Modern CPSs are increasingly interconnected and utilize multiple sources of data [5]. Such capabilities are inherently software-based.

In the advent of the fourth industrial revolution, the Industrial Internet, software is becoming more and more entangled in physical machines, each of them playing a role in achieving a system level goal [6]. Such a goal is accomplished by machines forming a cyber-physical system [7]: a network of machines executing software in a distributed and asynchronous way [8]. The impact of cyber-physical systems on industrial services in manufacturing is considerable [9], turning companies that have been designing machinery to software companies.

Proficient design of modern, complex CPSs requires advanced competencies due to their heterogeneous nature, physical world concurrent processes, and timeliness requirements [10] [5]. Notably, there are considerable research problems concerning for example multidisciplinary integrated system architecture modeling. With the increasingly central role of software in most CPSs, developing such new system architecture designs requires extensive software architecture competencies.

As with any software, the architecture plays a key role in ensuring the continuous operation of any CPS. In particular, industrial systems need to be operated continuously regardless out-of-order issues of components of this system. Due to high reliability requirements, software architecture plays a decisive role in all phases of the life cycle of a CPS with the software development phase having the most impact on the entire CPS sustainability [11]. To meet these goals at system level, software architecture of industrial CPSs needs to answer requirements of system orchestration, machine availability, predictive maintenance, and failure assessment. As for practical guidelines for meeting quality, interoperability and compatibility needs, the Industrial Internet Consortium has developed a reference architecture for designing software components for CPSs [12], highlighting the growing importance of software and software architecture in the CPS domain.

New CPS technologies offer significant opportunities, but they also pose considerable risks. The key source of development opportunities is the possibility to build new “smartness” and intelligence into the integrated and interconnected systems in totally new ways. For example, modern electricity network Smart Grids are large-scale CPSs with advanced control functions and automated metering services [13]. However, they have also introduced new software-related risks such as cyber-security issues. Both recognizing such new opportunities and managing the risks call for advanced software architectural capabilities.

2.2 Software Product Lines

A software product line (SPL) [14] is a collection of methods, techniques, tools, software components, and other assets that are used to create a collection of related products, sometimes referred to as a product family. The technical components that form the fundamental part of the product line are commonly referred to as core assets. These core assets are then reused in different products, and if necessary, they can be complemented with product-specific software components. While building on flexibility characteristics of software, SPLs can be applied in the design of CPSs [15]. Examples include cars, TVs, mobile phones, and many other mass-manufactured systems in which software plays a key role [16] [17].

A key element of any SPL is product-line architecture (PLA), which defines how the core assets and product specific components are organized to create products. In addition to the usual things included in software architectures, PLA also includes information about creating different variants. Building on PLA, a common way to partition an SPL is to organize core assets as a platform that can be extended, specialized, or tailored for product-specific use, at various levels [18]. Hence, two roles are needed: platform engineering and application engineering, both with different responsibilities. Platform engineering creates reusable components – the platform – that eventually make up the platform, which require assumptions on how future products will be built using them. In contrast, application engineering creates actual products, which requires a stable platform.

As platform and application engineering are run in parallel, but in the end share the same business goals, they need a common management function to steer the development. Examples of management decisions include resourcing of different flavors of engineering, schedules, and customer care. However, as the short-term technical goals of platform and application engineering are different, it is often difficult to balance between the different needs. Moreover, overlooking either type of engineering can lead to severe problems in the long run – focusing only on platform engineering leads to failing to deliver products in a timely fashion, and focusing only on products leads to increasing technical debt in core assets.

Planned Staged Investments [1] is a technique for managing and rebuilding SPLs in a sustainable way, based on technical and market needs. The overall aim is to manage more effectively SPLs when conflicting requirements simultaneously emerge from needs to redesign and reuse the software.

The key idea of Planned Staged Investments is to differentiate between two different operational modes – investment and harvesting – to coordinate the competing, parallel needs of redesign and reuse. These alternating modes can be characterized as follows:

- Investment: During investment, engineering effort is put into improving reusable asset creation. Development focuses on designing and improving product line’s core assets. In fact, they might even partly integrate product development. As an example, so-called lead products, commonly used in SPLs, are typically representatives of the first generation products built on a new generation of core assets forming the platform.

- Harvesting: During harvesting, benefits are gained from the investment in the form of simplified and faster product creation. The focus is placed on product development, and investments to core assets are minimized to only those that are critical for stability and robustness, thus reducing the need for product line engineering.

To summarize, the investment mode is a step change that requires careful planning, requirements, and technical surveys on technically feasible solutions. In contrast, harvesting mode supports iterative, rapid, and agile product creation. There are also pitfalls associated with the approach [1]. The most obvious one is that a prolonged harvesting stage will always lead to decreased productivity and lower quality while product-specific needs become increasingly difficult to meet owing to accumulating technical debt. Then, the management may consider that an investment needed is actually an indication of poor engineering rather than as a logical consequence of the overly extended harvesting period. Therefore, the harvesting period must be long enough to be profitable, and the investment phase must be extensive enough to renew the system. From the technical point of view, it is often difficult to developers to accept that the software made during harvesting contains numerous issues and problems that could be eliminated with some attention from designers.

2.3 Software Architecture Assessment

Assessing software architectures is a practical necessity for ensuring that the designed architecture meets its functional and quality requirements [19]. Over the past twenty years or so, several methods for evaluating and assessing software architectures have been developed (see e.g., [20] [21] [22] [23] [24] [25] [26]). Providing a comprehensive overview of the various methods falls beyond the scope of this chapter. However, in the following we introduce the salient properties of the prominent approaches that we have used, together with some first-hand experiences.

Two fundamentally different approaches to software architecture assessment exists: those based on experts asking questions and reviewing architectural artefacts (e.g., ATAM [27] [20]) and those based on measurements.

When performing reviews, the assessment team first collects information regarding the expectations of the stakeholders of the system. In scenario-based review approaches, the concerns and questions are posed as concrete scenarios involving a particular situation and stimuli that the system must respond to in a satisfactory manner. The scenarios exhibit important quality concerns of stakeholders, and they are evaluated together with the team responsible for the architecture. Evaluating a scenario means determining, with technical experts, whether or not the system will be able to produce a satisfactory response and identifying those aspects of the design that either support or inhibit reaching a favorable outcome. Scenarios can be predefined and reused in many different assessments virtually unmodified because they often address common situations related to, for example, security and maintainability.

As an example of another kind of review, the DCAR method [24] focuses on identifying architectural design decisions (meaning both a technical solution for a design issue and the actual resolution to use it), their rationale, and the relationships between the decisions. The decisions are then ordered by importance. In the evaluation part, the participants (typically the architect, the product owner, domain experts, and evaluation facilitators) discuss the forces affecting the most important decisions and their consequences (i.e., pros and cons) and vote whether each decision is good or needs to be reconsidered.

We have found it valuable to combine both the DCAR and ATAM approaches into a workshop style of architecture assessment [28]. The DCAR part of the workshop focuses on recovering the key aspects of the design and its history, while the scenario part can explore also future aspirations, opportunities, and risks and their impact on the architecture. This is the approach we have followed also in the case reported here.

Using measurements for assessing software architectures contrast expert reviews (like ATAM and DCAR above) in the way that the goal is not to raise questions about the system but to produce answers to questions as hard numbers. However, whereas not yet implemented designs can often be reviewed, measurements need a concrete object to measure: a simulation, prototype, or an at least partially implemented system in a test environment. For example, [29] presents an approach to evaluate architectural options by using reinforcement learning to find an optimal balance of incurred costs and benefits of alternative architectural choices run in a simulated system. It is important to recognize that such measurements are specific to a particular system and its quantified requirements. Indeed, there are no generally applicable, universal measures for the "goodness" of an architecture: an architecture is only more or less fit for its purpose as defined by the quality requirements of the system. Furthermore, relying too much on numbers (e.g. static metrics computed from code) can have a detrimental effect on quality – you will certainly get what you measure but that may not be what you actually need [30]. However, reviews and measurements can be used together as the RATE architecture assessment approach demonstrates [31] [30].

Scenario-based methods typically require effort and input from several stakeholders. A thorough assessment typically requires two or three full day meetings over a few weeks of calendar time and the participation of several key persons, adding up to tens (even hundreds) of person hours [31]. There is also a learning curve [32] [23]. Unsurprisingly, scenario-based methods are often perceived as heavy by the practitioners [33] [34]. On the other hand, assessment results are valuable and usually well received [31] [32] [35] [34], although they can be hard to quantify for management for decision-making [31]. As an example of usefulness of the results, in [30] the authors state that 75% of the over 50 assessments they performed led to concrete actions. Scenarios are a powerful tool not only for assessing the adequacy of the system under evaluation but also for making the technical people aware of the needs of the business and for making the business people aware of the opportunities provided and the challenges and risks posed by technology [31] [32].

There is evidence that scenario-based assessment and its derivatives are the most well-known methods in industry [33]. For recent reports on industrial experiences on architecture assessment and assessment methods (see e.g., [34] [28] [36] [24] [31] [23] [32]).

3 Software Architecture Assessment as a Risk Management Instrument

In the world today, there is a vast number of software-intensive CPSs that have different missions, size, technological basis, and dependencies. There are systems that are in their inception and there are very mature systems that have been around for decades. Architecture assessment can be performed not only in the early phase of the life-cycle of a CPS but also in some significant turning point in its life. Therefore, assessments can have very different goals and each system has its unique characteristics [26, p. 125]. Still, there are common problems that many systems have to cope with. Typical assessment goals and questions include (paraphrased from [31]):

- How suitable is the architecture as a basis for future products?
- Which framework or technology fits the needs best?
- How can performance, maintainability, or other important qualities be improved?
- How can the system be modularized to meet new productization and other business goals?
- What is the overall quality of the system and should it still be maintained or scrapped and redeveloped?
- How well does the designed architecture meet the key requirements?

- How can the system be modernized to meet new requirements and use modern technologies?

Identifying risks and is an integral part of ATAM [20] [37] and a major motivation for software architecture assessment in general. However, an explicit link to risk management processes is usually missing from the descriptions of the assessment methods. We try to bridge this gap here by projecting the risk related assessment activities onto the risk management process defined in the international standard ISO/IEC 16085-2006 [38] that is compatible with the system and software life-cycle process standards ISO/IEC 15288 and ISO/IEC 12207.

When evaluating scenarios in ATAM, one outcome is to identify architectural risks. In this context, labeling an architectural design decision as a ’risk’ means that the decision affects negatively an important quality attribute embodied in some scenario and hence poses a risk that the resulting system will not meet stakeholders’ requirements. The formulation of a scenario should state the required response in as concrete terms as possible, which makes the *risk criteria* [38] explicit. So, identified risks are mainly about failing to reach the desired level of some capability. ATAM does not mandate how to document the risks in terms of *risk exposure* [38], for instance. However, because the evaluated scenarios reflect be the most important stakeholder requirements, the risk of not satisfying them should be taken seriously.

ATAM recommends collecting risks that have a common (or closely related) root cause in ’Risk Themes’ for easier linking to business goals and for reporting to decision makers. Themes correspond to *risk categories* [38], although themes are often fine grained focusing on technical aspects. The purpose of collecting risks and risk themes is to facilitate planning of mitigating actions thereof. However, proposing *risk treatments* [38] is out of scope of ATAM and software architecture assessment in general. So, in terms of the risk management process defined in [38], the role of architecture assessment is mainly as a task in the *performing risk analysis* activity – focusing on architectural design decisions and their consequences.

In [39], a retrospective analysis of 18 ATAM assessments was done to find patterns in the risk themes identified. A characterization of 99 themes into 15 categories was developed. The categories range from architecture (run-time & development-time qualities) to processes and organization, which demonstrates the wide range of issues that can come up in assessments where business goals act as a starting point for deriving assessment criteria (i.e. the scenarios). The main findings of the study were that twice as many risk themes stem from ”omission” rather than ”commission”. That is, they concern design decisions not done, missing or misunderstood requirements, or other overlooked issues rather than the consequences of the architectural decisions already made. Interestingly, the study did not find any correlation between the risk themes identified and the requirements or the domain of the assessed system. That is, the type of a system does not seem to predict what kind of risks will come up. As a practical recommendation, the authors suggest that assessors should be acutely aware of risks stemming from the organizational context and the process of architecting rather than the kind of system under development while being on a constant lookout for important things missed.

Managing the risk related to changes in software is a major reason for doing architecture assessments according to [30]. They see two distinct ways in which architecture assessment can proved input for mitigating the risks related to software change requests: (1) by evaluating how the system and its architecture can accommodate a set of anticipated changes (that are more or less likely to happen), and (2) by determining the potential impact of concrete change requests currently at hand. The engineering branch of a development organization typically initiates these kind of assessments. As examples of external initiators of assessment they give a potential customer who wants to gauge risks prior to investment, or an existing customer who wants to sort out known problems [31] [30]. In a case study focusing on the adaptability of a CPS based on its software architecture [40], the authors evaluate alternative architecture designs using four criteria concerning well-known aspects of design and implementation that affect how well the system can adapt to changing needs and execution context.

Because of the wide range of modern new CPSs, their potential risks stem from a variety of sources. Not only the cyber parts but also the electronic, hardware and mechanical parts in conjunction to the humans involved must be taken into account. In addition, interconnected systems add to the complexity. System risk factors like safety and security are crosscutting. Consequently, engineering high-confidence CPSs requires advanced multidisciplinary competencies and co-development [5].

From the discussion above, it is clear that risk analysis in architecture assessment is typically focused on identifying things that could go wrong in the architecture and its development leading to a systemic failure. The findings are distilled and reported in terms that are understandable for business owners and managers so that the findings can be fed into the risk management process (e.g. into *the project risk profile* [38]) so that the managers can decide about the *risk action requests* [38] for treatments to mitigate or remove risks. Naturally, immediate corrective actions can be agreed on during assessment if managerial decisions are not required.

However, the literature is lacking examples of viewing risks as opportunities – of being proactive and recognizing options instead of just reacting to changes forced on by external developments. Some assessment methods do explicitly mention recognizing opportunities for architectural improvements as a motivation for assessments.

On the other hand, in addition to technical findings, other positive effects of assessments have been recognized. Because an architectural assessment usually means a deep discussion about product goals and technical possibilities, it not only helps to create a common frame of reference for the business and technology sides but it also provides a rare chance to share experiences, knowledge, and the rationale behind architectural choices [32] [26, p. 6] in the organization. Also, it gives the opportunity to educate business owners about the potential of technology and the existing software assets. These 'soft effects' may in practice be even more valuable than the hard technical results [24]. Therefore, we wanted to explore in our case study how to bring in the other side of risk analysis, recognizing opportunities, as an additional perspective to architectural assessment.

4 Case Study

In this section, we present an improvement case study we conducted at a company that provides industrial automation solutions. The goal of the case study was to help the case company revisit its CPS software product line in preparation for the foreseen increasing role of the software in the future. A series of software architecture assessment workshops were used as the concrete mechanism to facilitate discussions and to identify risks and opportunities related to the software.

Giving a detailed account of the product line and the technical findings is not the purpose of this work and, consequently, we describe the product and the findings in general terms. Our focus is on describing the assessment process, its conduct, and the value of the outcomes.

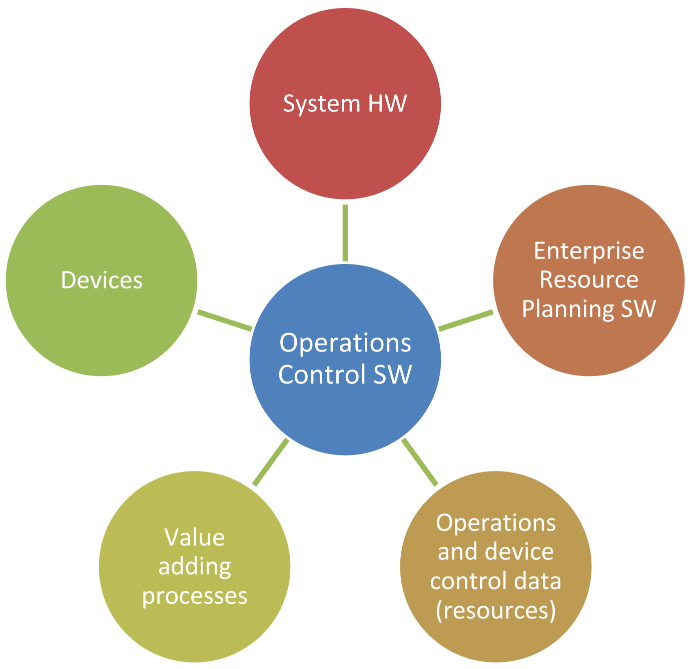
System HW
Devices
Enterprise Resource Planning SW
Operations Control SW
Value adding processes
Operations and device control data (resources)

Figure 1: Functional scope of Operations Control Software

4.1 Case Company and Case Product

The company provides industrial automation solutions for controlling various devices and follows the usual risk-oriented view to software they are engineering. The factors they would like to study reflect to potential risks like feature creep, sensitivity for control points, and scaling the current product line to meet changing customer demand. At the same time, various opportunities have been identified, including new business openings, widening the scope of the product line, reducing overheads and shortening lead-times in developing customer specific variants of the software, and reducing the need for bespoke device interfaces by promoting and embracing new standards in the field.

We call the case product under study Operations Control System (OCS). Figure 1 visualizes the functional scope of OCS. The system controls various industrial Devices using the Operations and Device Control Data provided by human operators. Individual Devices are typically combined into conglomerates that together perform an industrial process with the help of additional hardware (System HW). The OCS exchanges also information with Enterprise Resource Planning systems and other Value Adding Processes. OCS has gone through significant architectural and technological changes over its life cycle. The installed base of the system (base version and variants) is in the thousands.

From the company’s perspective, the motivations for conducting a review of the software architecture of OCS comprised of the following questions:

1. Are the architecture and the technological choices on as sound a basis as we think?

2. Do outside experts see any risks or weaknesses?

3. How long can we keep on adding features to a single platform to serve growing customer needs and what would be the options?

4. How far does the performance of the system scale up in terms of the amount of operational data and the number of devices controlled?

4.2 Data Collection

Table 1 lists the participants of the series of workshops. Eleven people (the Informants I1–I11) participated from the case company and five from University of Helsinki (the Researchers R1–R5). As the table shows, the informants were very experienced and had core competences bearing on the product. The researchers had significant academic and industrial experience.

Table 1: Study participants.

Id	Role	Experience and expertise
Case Company Employees		
I1	Dept. manager	20 y, responsible for the study at the company
I2	Chief architect	15 y, responsible for architecture of the target system Dev.
I3	Dev. manager	20 y, responsible for the development of the target system
I4	Product Owner	10 y, product manager for digital services
I5	CDO	20 y, senior executive, chief digital officer of the company
I6	Sales agent	10 y, customer interface, sales support
I7	Engineer	15 y, developer, user experience
I8	Engineer	10 y, developer, robotics
I9	Engineer	10 y, developer
I10	Engineer	15 y, architect, user interface
I11	Programmer	1 y, doing Master's thesis on the topic
Researchers (University of Helsinki)		
R1	Professor	10 y industrial experience, 15 y academic research in industrial collaboration
R2	Professor	3 y industrial experience, 25 y academic research in industrial collaboration
R3	Senior researcher	15 y industrial experience, 10 y academic research in industrial collaboration
R4	Researcher & lecturer	12 y industrial experience, 10 y academic research
R5	Post Doc. Researcher	3 y industrial experience, 10 y academic research

Table 2 lists in chronological order the face-to-face meetings held at the company premises during the study that were the primary way of collecting the data for the study. The actual conduct of the assessment process will be explained in required detail below. The table gives the duration of each event, lists the participants using the IDs given in Table 1, and explains the main outcomes or purpose of each event.

As we can see from Table 2 there was strong presence from the company in each event, which shows a high level of commitment. It is in fact remarkable that the key persons found time in their busy schedules for this work; it is a common experience that 'daily workload wins over architecture evaluation' [26, p. 119].

Although the original planned timetable was not met, all the parties showed flexibility and resilience in seeing the work through. The meetings had clear goals and although the discussions did sometimes take a meandering course, they resulted in a wealth of high quality data. This is also reflected by the actual results obtained in the end and in the expressed interest of the company to continue the co-operation with the researchers in this area.

Table 2: Data collection events (at the company premises).

Event	Duration [h]	Participants	Focus of Outcomes
Workshop 1, Nov 2018	4	I1, I2, I3, I4, I5, I6, I7 R2, R3, R4, R5	Kick off and introductions, overview of the CPSs of the company and the short-term and longer-term business needs
Workshop 2, Feb 2019	3,5	I1, I2, I3, I4, I11 R1, R2, R3, R4, R5	Architecture assessment tutorial, setting goals for the assessment
Workshop 3, Feb 2019	5	I1, I2, I4, I7, I8, I11 R2, R3, R4, R5	Architecture presentation of the OCS core system, formulating the initial list of design decisions and scenarios
Workshop 4, May 2019	5	I1, I2, I3, I4, I7, I8, I9, I10, I11 R1, R2, R3, R4, R5	Review of the documented design decisions
Workshop 5, June 2019	5	I1, I2, I4, I7, I8, I11 R2, R3, R4, R5	Prioritization of scenarios and evaluation of the most important ones

4.3 Chronological Description of Activities and Events

The activities of the study were centered on the main events recorded in Table 2 over approximately seven months of calendar time. The first meeting, Workshop 1, introduced the product and the company's current and projected future business needs. In this meeting, the general objectives for the study and the forms of co-operation were agreed. After the meeting, the research team formulated the first plan with an overall timetable.

Workshop 2 consisted of a tutorial about software architecture and software architecture assessment given by the research team. Video and other materials were provided for self-study at the company. In addition, the goals and the scope (focusing on the core functional parts of OCS device control and data management) were set in the meeting. After the meeting, a more detailed plan for the next workshop was produced. The idea was to follow the DASE approach of lean assessment explained in [28]. Following the approach, the research team prepared a preliminary list of design decisions and scenario sketches. The company representatives were asked to come up with their own suggestions for scenarios based on the researchers' list, which they did.

Workshop 3 began with a presentation by the OCS architect (Informant 2) about the design of the system under study and about the most recent changes it had gone through, as well as the reasoning behind. During the presentation, the researchers asked questions and collated lists of important aspects of the design in order to reconstruct a list of design decisions (decisions had not been systematically recorded before). In addition, possible scenarios were sketched during the first part of the workshop. The original plan was to select the most important decisions and to document them for analysis and voting (”OK” – ”OK with some issues” – ”Not OK”) during the first part of the workshop and then, during the second part of the day, form a list of scenarios and evaluate them. However, this turned out to be an unrealistic plan. The architecture presentation, the questions, and the discussions on the aspects of the design and their rationale took almost all of the time. There was no time left for documenting decisions, but some time was used to go through a few scenarios prepared by the researchers in advance. However, a good picture of the architecture and the design issues was acquired. At the end of the day, it was clear that two full day meetings would be needed in order to analyze the design decisions and evaluate scenarios properly.

After Workshop 3, the researchers formulated a top list of architectural design decisions and sent them over to the company for commenting and documenting using the appropriate DCAR template[1]. The company was again asked to prepare scenarios. Over the next few months, the company representatives went through the initial list of decisions selecting the most important ones from the list and adding some decisions they thought were relevant. This resulted in eleven carefully documented decisions. They also worked on scenarios but they found that rather difficult. There was also a lack of time for the work.

The goal of Workshop 4 was to evaluate the design decisions documented by the company representatives. Thanks to the thorough preparations of the company people, the evaluation went smoothly and all documented decisions were analyzed and voted on. Only the informants with the relevant technical knowledge from the responsible development team were allowed to vote. Several issues were noted down. In addition, a brief look at the few scenarios prepared so far was taken. It was clear that effort and help from the researchers’ side was needed to move this task forward. The document including the decisions and the voting results (marked using a ’traffic-light’ coloring scheme for OK–some issues–not OK) as well as any comments was sent at the end of the workshop to the company representative.

In the final phase of the assessment, the researchers prepared fifteen scenarios divided into four themes. The themes addressed (1) the current strategic goals of the company, (2) potential technological and business developments that could present opportunities or pose risks, (3) threats, and (4) software development topics. The scenarios were partially documented using an ATAM-style scenario template, and a separate spreadsheet was prepared that listed the names and other characteristic attributes of the scenarios. The characteristics include the usual risk-related factors of probability and potential impact to business, the estimated time frame for the realization of the scenario, the difficulty or effort of realizing the scenario (where applicable), and whether or not the scenario includes opportunities or risks (or both). The company representatives ran their own scenario gathering sessions and added scenarios and filled in some of the known attributes of the scenarios in the sheet. This resulted with the final list of 22 scenarios in the four themes with a relatively even distribution.

Workshop 5 was started by first reviewing the list of scenarios and then selecting those that the participants considered the most important. This resulted in six scenarios with at least one from each theme. Next, the scenarios were evaluated by discussing how the architecture would either support or not achieving a

[1] `http://www.dcar-evaluation.com/?page_id=4`

favorable outcome. However, not all scenarios were actually stated in a way that would have allowed determining a definitive response. Some of the scenarios represented such a visionary state that they were very much outside the scope of the actual system under study. These could not be handled in a meaningful way and they were left for future when there would be an actual design to reflect on. Overall, six scenarios were evaluated thoroughly. Based on the session, three actions points were recorded for the company for immediate execution. The actions concerned the current version of the system.

In addition to the face-to-face information sharing in the workshops (Table 2), the case company provided during the study period supplementary documentation and the presented materials to the researchers. These were especially valuable given that the researchers were not experts of the industry domain of the case company.

4.4 Results

The major findings resulting from the reviews of the design decision and the scenarios of the assessed core part of the software system are listed in Table 3. The findings are categorized by the expected time frame for required actions from the company's side, ranging from Immediate (do now) to Long (in a few years), and by a uniting topic, or, risk theme, as they are called in ATAM. Each entry also shortly describes what kind of risk or opportunity is involved. Because the details of the findings are not important for this exposition, we describe the issues in general terms. We have included an indicator (*D* for Decision-based review and *S* for Scenario-based review) for the phase of review where the issue was discovered and recorded; some issues came up in both reviews.

Some of the scenarios turned out to be difficult to prioritize in the assessment. For example, although the participants from the platform team (responsible for developing the OCS core software) acknowledged the customer need for a cloud-based system solution, they also saw this approach risky for the time-critical functions of OCS. Consequently, there is a potential trade-off between important system qualities, and the company wanted to discuss the impact of different options confidentially with their customers.

At a general level, the cooperation between the company and researchers was mutually beneficial. In particular, the iterative nature of the approach that we followed was essential because it provided time for both parties to understand the details, practicalities and limitations of the other party. In other words, the researchers learned a lot about products and product development at the case company, and the company representatives had several lessons about software architecting and architecture assessments. For example, in the beginning, scenarios as a concept were not so well first understood by the company representatives and thinking of them spurred vivid discussion, but capturing them in text was easily left for later. However, during the course of the workshops, the company representatives were quick to pick up the idea of using a concrete example to demonstrate a technical detail in their design, to the extent that scenarios might become a permanent means to justify technical decisions in the case company. Overall, working with design decisions was easier for them than working with the scenarios.

When asked for feedback afterwards, the Department manager (Informant I) stated that they found the results useful for the company. The findings will help in developing the current platform further and when doing the groundwork for a new architecture. They also valued the systematic way of evaluating architecture, and they appreciated the outside view that the researchers brought to the process. In this way, they found their time well spent.

Table 3: Identified risks and opportunities.

Time frame	**Topic**	**Risk or Opportunity**
Immediate (do)	Technical debt	A testing application suffers from feature bloat and is difficult to maintain, which is a minor but non-trivial risk for project work (D[†])
Short (start)	Performance	Hard numbers about certain areas of performance are missing which is a potential risk for some projects (S[‡])
	Security	OCS uses standard network security mechanisms that depend on the facilities provided by the customer. Because industrial processes are moving towards on-line computing, a thorough analysis of ensuing security risks should be conducted in order to device appropriate countermeasures. (D, S)
Medium (year)	Sharing of factory resources	Sharing of common operational resources between devices controlled by separate OCS instances is a potentially important feature that requires some system and software architecture work. This is both an opportunity and a risk because the demand for such solutions is increasing. (S)
Long (few years)	Technology dependency	A dependency on a technology was identified as a potential future risk that needs to be addressed if the situation changes. (D, S)
	Cloud-based system	There is an inevitable technology trend towards cloud-based services and data sharing in the industry domain, which was categorized potentially as an opportunity to be addressed in the future, but also as a risk if left unaddressed for too long. (S)

[†] D = Decision-based review [‡] S = Scenario-based review

5 Discussion

As already mentioned, our experiences are based on continuous, iterative cooperation with the case company. In the following, we provide an extended discussion to some of the key elements of our approach, and how they are reflected in our experiences with the case company.

5.1 Role of Planned Staged Investments

Since architecture assessments utilize scenarios as a mechanism for identifying risks and potential problems, they at the same time are also an effective mechanism for identifying opportunities, or options that can be easily incorporated in the existing design. Furthermore, increasing risk awareness associated with the present design also enables considerations regarding actions to be taken to mitigate the risks as well as to improve the design in a rational, planned fashion rather than having to resort to hacks at the last possible moment on a per-customer basis.

To summarize, an evaluation of the present architecture with regard to risks it contains also enables thinking of potential directions for the future versions, thus unveiling potential opportunities. Furthermore, a timeline can be created to highlight the schedule for mitigating risks and grasping the opportunities. Based on the timeline, it is then possible to allocate different features to releases, using Planned Staged Investments as the strategy for the allocation.

Based on the experiences with the case company, it is clear that the most urgent issues will be directly included in the different products, with the present version of the platform as the baseline. Moreover, some of the features might even be patches to already existing systems, in particular when considering security-related risks in case of connected systems. In contrast, some road mapped value adding functionalities of the future may require a new platform so that they can be scheduled for release to the whole product line.

5.2 Lessons Learned: Walking the Line between Risk and Opportunity

Balancing between risks and opportunities turned out to be surprisingly difficult. The tendency was to always consider risks first, and opportunities only later. To some extent, this can be explained by the fact that the platform team is used to getting requirements from the product teams, and there is limited experience in being able to put in ideas for future features to product teams proactively. An exception to this observation is actions to renew the software technically from within, meaning that their newer counterparts could replace older, partly deprecated subsystems. We believe that this is a somewhat natural situation when considering platform teams operating in the CPS domain. The responsibility for implementing and maintaining the software platform weighs more on the teams than visioning new products.

To improve the situation, assessments would require even deeper participation by people from the customer interface who would be closer to the needs and everyday life of the customers. Moreover, they would be at a better position to consider the opportunities and their importance from the business perspective.

Overall, we experienced and discovered several notable learnings and findings in our industrial case study presented in Section 4:

- Planning and performing software architecture assessments in systematic ways require significant resources – particularly time – both from the assessors and from the software development organization.
- In case of large systems such as the OCS, the scope and focus of the assessments should be planned and prioritized according to realistic budgets.
- Because CPS software is by nature deeply coupled and intertwined with the other elements of the system and its operating environment, it is imperative to have sufficient high-level comprehension of the entire CPS in order to be able understand the role and dependencies of the software (e.g., hardware connections) in the whole system (c.f., Fig. 1).
- Even when conducting just software architecture assessment, the key business drivers and particular company targets should be known at a general level.

- That helps rationalizing the design choices in the context. Consequently, the software assessors should have access to such information in advance and preferably also the business and product stakeholders participating in the actual assessment process as we did in our company case.
- In practical industrial settings the architectural knowledge may be partially tacit and the documentation incomplete. This is understandable in particular in cases of large systems with very long life times (even tens of years).

The assessors should be ready to work on such knowledge constraints. It is then also important to be able to discuss directly with the senior software designers who can recollect the key information at the time of the architectural decision-makings possibly done many years ago.

5.3 Threats to Validity

The validity of as study is basically about the knowledge claims that can be made based on the results [41]. As our intent was to gain experiences on the usage of particular architecture evaluation methods, one particular issue in terms of validity is that of the role of the evaluation approach itself in the results achieved. The separation of the approach used from the experience of the facilitators in the actions taken is fundamentally hard. In this study, the researchers, who acted as the facilitators, have a rather high level of experience in industrial software engineering and in software architecture research and practice in particular. This is something one may need to take into account if aiming to apply (generalize) the results in other cases. On the other hand, the approach to use was defined in advance and clearly documented while doing, so the guiding decisions made by the facilitators were not based on intuition or experience alone.

In terms of construct validity, even the central concepts of the study area are not uniformly defined and much of the domain terminology was not initially familiar to the researchers, and therefore, a risk for misunderstandings is real. However, as the collaboration with the company representatives and researchers was very tight, a form of member checking [42] was continuously used, as the understanding of the researchers was reflected back to the company participants and special attention was paid on trying to ensure we were talking about the same thing. Furthermore, the lack of domain understanding potentially leading to misunderstanding by the researchers was, at least partially, alleviated by the emphasis on the need of the case company participants understanding the overall process and taking the responsibility of the domain issues.

6 Conclusion

We have reported here the practical experiences we gained in using architecture assessments as a basis for identifying risks and opportunities in the domain of CPSs. The findings of our architecture assessments are two-fold. On the one hand, the case company found it easy to discuss scenarios that are close to its event horizon and build on business requirements from existing customers. These are hardly the key opportunities for future business, but rather contain potential risks. On the other hand, getting to a level where business benefits of extended digitalization and more elaborate software features will start to emerge requires in-depth connection with the case company and long-term commitment to elaborate the opportunities thoroughly. Additional discussions including the company top-management setting the business strategy and positioning of the particular product offering would be grounding.

We can conclude that architecture assessment is an effective way of uncovering risks that bear on architectures' capability to support business. This is especially true when examining an established system; the assessment will help to determine and affirm the limitations and the scope of the current design. However, addressing opportunities is not so straightforward. Although, these can be recognized and discussed, they may not fit the current scope of the system and thus be difficult to analyze further – unless there already is a clear requirement for such features from business owners. A possible way forward would be to develop alternatives for a future architecture and assess them against the opportunistic scenarios to pave the way for creating a transition path from the current system to the new one.

Acknowledgments

The authors wish to express their sincere thanks to the case company. This research was funded by DIMECC[1] and the Finnish public funding agency Business Finland.

References

[1] J. Savolainen, N. Niu, T. Mikkonen and T. Fogdal, "Long-term product line sustainability with planned staged investments," *IEEE software,* vol. 30, p. 63–69, 2013.

[2] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008.

[3] J. Lee, H.-A. Kao and S. Yang, "Service innovation and smart analytics for industry 4.0 and big data environment," *Procedia Cirp,* vol. 16, p. 3–8, 2014.

[4] M. Mikusz, "Towards an understanding of cyber-physical systems as industrial software-product-service systems," *Procedia CIRP,* vol. 16, p. 385–389, 2014.

[5] H. A. Müller, "The Rise of Intelligent Cyber-Physical Systems," *Computer,* vol. 50, pp. 7-9, 12 2017.

[6] A. Gilchrist, Industry 4.0: The Industrial Internet of Things, 2016.

[7] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir and T. Eschert, "Industrial Internet of Things and Cyber Manufacturing Systems," 2016.

[8] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn and K. Ueda, "Cyber-physical systems in manufacturing," *Cirp Annals,* vol. 65, p. 621–641, 2016.

[9] M. M. Herterich, F. Uebernickel and W. Brenner, "The impact of cyber-physical systems on industrial services in manufacturing," *Procedia Cirp,* vol. 30, p. 323–328, 2015.

[10] S. K. Khaitan and J. D. McCalley, "Design Techniques and Applications of Cyberphysical Systems: A Survey," *IEEE Systems Journal,* vol. 9, pp. 350-365, 6 2015.

[11] M. Törngren and U. Sellgren, "Complexity Challenges in Development of Cyber-Physical Systems," in *Principles of Modeling: Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*, M. Lohstroh, P. Derler and M. Sirjani, Eds., Cham, Springer International Publishing, 2018, p. 478–503.

[1] `https://www.dimecc.com/`

[12] Industrial Internet Consortium, "Industrial Internet Reference Architecture," *Technical Report,* 2015.

[13] X. Yu and Y. Xue, "Smart Grids: A Cyber–Physical Systems Perspective," *Proceedings of the IEEE,* vol. 104, pp. 1058-1070, 5 2016.

[14] F. J. Van der Linden, K. Schmid and E. Rommes, Software product lines in action: the best industrial practice in product line engineering, Springer Science & Business Media, 2007.

[15] E. Niemelä and T. Ihme, "Product line software engineering of embedded systems," *ACM SIGSOFT Software Engineering Notes,* vol. 26, p. 118–125, 2001.

[16] A. Sangiovanni-Vincentelli and G. Martin, "Platform-based design and software design methodology for embedded systems," *IEEE Design & Test of Computers,* vol. 18, p. 23–33, 2001.

[17] P. Liggesmeyer and M. Trapp, "Trends in embedded software engineering," *IEEE software,* vol. 26, p. 19–25, 2009.

[18] T. Myllymäki, K. Koskimies and T. Mikkonen, "Structuring product-lines: A layered architectural style," in *International Conference on Object-Oriented Information Systems*, 2002.

[19] J. Bosch and P. Molin, "Software architecture design: evaluation and transformation," in *Proceedings ECBS'99. IEEE Conference and Workshop on Engineering of Computer-Based Systems*, 1999.

[20] R. Kazman, M. Klein and P. Clements, "ATAM: Method for Architecture Evaluation," 2000.

[21] P. Bengtsson, N. Lassing, J. Bosch and H. van Vliet, "Architecture-level modifiability analysis (ALMA)," *Journal of Systems and Software,* vol. 69, p. 129–147, 2004.

[22] T. Kettu, E. Kruse, M. Larsson and G. Mustapic, "Using Architecture Analysis to Evolve Complex Industrial Systems," in *Architecting Dependable Systems V*, vol. 5135, R. de Lemos, F. Di Giandomenico, C. Gacek, H. Muccini and M. Vieira, Eds., Springer, 2008, p. 326–341.

[23] E. Woods, "Industrial Architectural Assessment Using TARA," *Journal of Systems and Software,* vol. 85, p. 2034–2047, 2012.

[24] U. van Heesch, V.-P. Eloranta, P. Avgeriou, K. Koskimies and N. Harrison, "Decision-Centric Architecture Reviews," *Software, IEEE,* vol. 31, pp. 69-76, 1 2014.

[25] M. Raatikainen, J. Savolainen and T. Männistö, "Architecture Management and Evaluation in Mature Products: Experiences from a Lightweight Approach," in *Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures*, New York, NY, USA, 2014.

[26] J. Knodel and M. Naab, Pragmatic Evaluation of Software Architectures, Springer, 2016.

[27] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson and J. Carriere, "The Architecture Tradeoff Analysis Method," in *Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems, 1998. ICECCS '98*, 1998.

[28] A.-P. Tuovinen, S. Mäkinen, M. Leppänen, O. Sievi-Korte, S. Lahtinen and T. Männistö, "Unwasted DASE: Lean Architecture Evaluation," in *18th International Conference on Product-Focused Software Process Improvement (PROFES 2017)*, Cham, 2017.

[29] D. Sobhy, L. Minku, R. Bahsoon, T. Chen and R. Kazman, "Run-time evaluation of architectures: A case study of diversification in IoT," *Journal of Systems and Software,* vol. 159, 2020.

[30] J. Knodel and M. Naab, "Mitigating the Risk of software change in practice: Retrospective

on more than 50 architecture evaluations in industry (Keynote paper)," in *2014 Software Evolution Week – IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 2014.

[31] J. Knodel and M. Naab, "Software Architecture Evaluation in Practice: Retrospective on More Than 50 Architecture Evaluations in Industry," in *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on*, 2014.

[32] V. Reijonen, J. Koskinen and I. Haikala, "Experiences from Scenario-Based Architecture Evaluations with ATAM," in *Software Architecture, 4th European Conference on (ECSA 2010)*, vol. 6285, M. Ali Babar and I. Gorton, Eds., Springer, 2010, p. 214–229.

[33] A. Banijamali, P. Heisig, J. Kristan, P. Kuvaja and M. Oivo, "Software Architecture Design of Cloud Platforms in Automotive Domain: An Online Survey," in *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, Kaohsiung, Taiwan, 2019.

[34] P. Cruz, H. Astudillo, R. Hilliard and M. Collado, "Assessing Migration of a 20-Year-Old System to a Micro-Service Platform Using ATAM," in *IEEE International Conference on Software Architecture Companion (ICSA-C)*, Hamburg, 2019.

[35] S. Ferber, P. Heidl and P. Lutz, "Reviewing Product Line Architectures: Experience Report of ATAM in an Automotive Context," in *Software Product-Family Engineering*, vol. 2290, F. van der Linden, Ed., Springer, 2002, p. 364–382.

[36] S. Bellomo, I. Gorton and R. Kazman, "Toward Agile Architecture: Insights from 15 Years of ATAM Data," *IEEE Software,* vol. 32, p. 38–45, 2015.

[37] P. Clements, R. Kazman and M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley, 2002.

[38] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), *International Standard ISO/IEC 16085:2006 Systems and software engineering — Life cycle processes — Risk management,* ISO/IEC, 2006.

[39] L. Bass, R. Nord, W. Wood and D. Zubrow, "Risk Themes Discovered through Architecture Evaluations," in *2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*, 2007.

[40] M. Mayrhofer, C. Mayr-Dorn, A. Zoitl, O. Guiza, G. Weichhart and A. Egyed, "Assessing adaptability of software architectures for cyber physical production systems," in *ECSA 2019, Lecture Notes in Computer Science*, vol. 11681, T. Bures, D. L. and I. P, Eds., Springer Nature Switzerland AG, 2019, pp. 143-158.

[41] W. R. Shadish, C. D. Thomas and C. D. Thomas, Experimental and quasi-experimental designs for generalized causal inference, Houghton Mifflin Company, 2002.

[42] J. W. Creswell, Research Design: Qualitative, Quantitative and Mixed Methods Approaches, Sage Publications, 2009.