Boundary Objects in Agile Practices: Continuous Management of Systems Engineering Artifacts in the Automotive Domain

Rebekka Wohlrab
Chalmers | University of Gothenburg
Systemite AB
Gothenburg, Sweden
wohlrab@chalmers.se

Patrizio Pelliccione
Eric Knauss
Chalmers | University of Gothenburg
Gothenburg, Sweden
{patrizio.pelliccione,eric.knauss}
@cse.gu.se

Mats Larsson
Systemite AB
Gothenburg, Sweden
mats.larsson@systemite.se

ABSTRACT

Automotive companies increasingly include proven agile methods in their plan-driven system development. The adoption of agile methods impacts not only the way individuals collaborate, but also the management of artifacts like requirements, test cases, safety documentation, and models. While practitioners aim to reduce unnecessary documentation, there is a lack of guidance for automotive companies with respect to what artifacts are needed and how to manage them. To close this knowledge gap and create practical guidelines, we conducted a design-science study together with 53 practitioners from six automotive companies. Using interviews, surveys, and focus groups, we analyzed categories of artifacts and practical challenges to create applicable guidelines to collaboratively manage artifacts in agile automotive contexts. Our findings indicate that different practices are required to manage artifacts that are shared among different teams within the company (boundary objects) and those that are relevant within a specific team (locally relevant artifacts).

CCS CONCEPTS

• **Software and its engineering → Agile software development**; **Documentation**; *Collaboration in software development*;

KEYWORDS

Agile systems engineering, empirical software engineering, large-scale agile, documentation

ACM Reference Format:
Rebekka Wohlrab, Patrizio Pelliccione, Eric Knauss, and Mats Larsson. 2018. Boundary Objects in Agile Practices: Continuous Management of Systems Engineering Artifacts in the Automotive Domain. In *ICSSP '18: International Conference on the Software and Systems Process 2018 (ICSSP '18), May 26–27, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3202710.3203155

ICSSP '18, May 26–27, 2018, Gothenburg, Sweden

1 INTRODUCTION

The agile paradigm has been in the focus of research for many years and is becoming more widely adopted in industry [7]. Based on benefits and success in software development, the agile paradigm has also been applied to systems engineering and product development contexts in several domains [13]. As it has become apparent that companies in competitive markets need to adopt cost-effective practices and flexibly react to change, the automotive industry also aims to improve agility and flexibility on system level [8]. However, besides the growing system complexity, the importance of safety, and a plethora of variants, also knowledge management has been reported a challenge in automotive software engineering [4, 8].

This knowledge is often represented in artifacts like requirements, test cases, and architecture models. We refer to these artifacts as *systems engineering artifacts* in this paper. When adopting agile methods and aiming to become more efficient, practitioners intend to reduce unnecessary documentation and improve knowledge management [14, 25].

However, finding the 'right' amount of documentation in agile projects is challenging [30]. To support practitioners in large-scale agile contexts, Dingsøyr et al. advocated more research on inter-team coordination and knowledge sharing and "case studies to provide industry practitioners with research-based advice" [6]. There is a lack of guidance for automotive companies with respect to what documentation is needed in agile automotive contexts and how to manage this documentation in large-scale agile organizations [16].

In this paper, we seek to create guidelines for the management of systems engineering artifacts in agile automotive contexts based on an understanding of challenges and practices to manage artifacts. In this context, we are interested in understanding what typical relevant artifacts are and what they are used for. Concretely, we aim to get this understanding by answering the following research questions:

RQ1: What are practices to manage artifacts in agile automotive systems engineering?

RQ2: What practical challenges exist with managing systems engineering artifacts in agile automotive contexts?

Based on these findings regarding practices and challenges, we developed guidelines for practitioners to manage systems engineering artifacts in several iterations.

In our study, we relied on the concept of "boundary objects" [33], i.e., artifacts used across boundaries between several groups in an

organization. This was done to focus the interviews on artifacts relevant for several groups and identify what guidelines are applicable to manage such artifacts.

We conducted a design-science study [12] with six automotive companies that are aiming to become more agile and reduce unnecessary documentation. In this design science study, we iteratively created and evaluated guidelines for the continuous management of systems engineering artifacts. Besides a review of related work, we conducted a case study to explore artifacts, practices, and challenges. We developed and evaluated guidelines in several iterations, using focus groups, interviews, and a questionnaire.

The participating companies were four automotive OEMs, an automotive supplier, and a supplier of an information management tool used in the automotive domain. The participants were 53 experts with various backgrounds and 41 anonymous questionnaire respondents.

Contributions: Our contributions are (i) an analysis of artifacts and practices, (ii) a catalog of challenges, and (iii) guidelines to manage systems engineering artifacts.

Our findings indicate that artifacts that are only relevant within one team should be created only when they are actually needed. In contrast, boundary objects should be created upfront with a lightweight approach to allow for early impact analysis and decision making. Our guidelines address several practical challenges, e.g., degradation of artifacts and collaboration across different locations. We expect the guidelines to be applicable to automotive or even general systems engineering, but future studies are needed to examine their generalizability.

The remainder of this paper is structured as follows: We present related work in Section 2. Participating companies are described in Section 3 and our research method is presented in Section 4. Section 5 presents our findings. Each of its subsections ends with a discussion. Table 3 shows an overview of our findings which are discussed with conclusions and future work in Section 6.

2 RELATED WORK

Several studies have analyzed the role of communication in agile practices, concluding that while intra-team communication improved when transitioning to agile, coordination between teams is challenging [26]. Hummel et al. motivated the need for different communication modes in distributed agile teams, especially if knowledge needs to be retained for maintenance reasons [14]. Other findings indicate that agile methods can facilitate knowledge sharing in a team, but documentation cannot be replaced completely [18].

Bass examined artifact inventories in large-scale agile development, and concluded that "there are no agile ceremonies specifically designed to create and refine any of these artifacts", which are especially needed to coordinate cooperating teams [1]. Our paper works towards closing this gap by designing guidelines to manage artifacts, also across team borders.

Rüping [30] listed typical artifacts required in agile contexts, but stated that it is challenging to elicit documentation requirements and identify the right amount of documentation. The need for more "research on methods and tools that facilitate agile documentation" has been proclaimed [35].

Heldal et al. [11] presented a classification of models: Descriptive artifacts "describe a subject that already exists", either implicitly in the modeler's mind (an idea of the future architecture) or explicitly (e.g., representing an already running system). Prescriptive artifacts are used to create a subject that does not exist yet, e.g., models used for code or artifact generation. We consider this classification in our study, and broaden the scope to other artifacts.

The underlying concepts of this study have been influenced by coordination theory, concerned with the management of dependencies between activities [21]. This is often supported by shared resources and various artifacts. The focus on genre repertoires helps understand communication in various (often distributed) communities and how it changes over time [24]. In our study, we discuss the concept of boundary objects [33] to focus discussions on artifacts that can be used to collaborate between different teams. This concept was initially coined by Star and Griesemer [33]. There exist several case studies related to boundary objects, e.g., related to architectural descriptions as boundary objects in agile systems engineering and requirements for their creation [25]. Blomkvist et al. [2] focused on how boundary objects can be leveraged in distributed agile user-centered design. They concluded that in distributed teams, boundary objects can facilitate communication.

In this paper, we focus on the automotive domain which comes with several challenges. These challenges are connected to knowledge management [4], traceability and scattered information [28], variability, and compliance with several standards. Diebold et al. [5] concluded that it is possible to combine agile methods with Automotive SPICE compliance, but that there still exist open challenges and a need for guidelines, e.g., with respect to architecture documentation. Our study aims to address these challenges and develop practical guidelines to support practitioners with the management of systems engineering artifacts.

3 INVOLVED COMPANIES AND PARTICIPANTS

This section presents the companies and participants involved in our study.

Company A, which participated in our exploratory case study, is an automotive manufacturer with more than 15,000 employees. The study came into place because we were asked for support with the continuous management of artifacts. The top-level organization follows the V-model process. Whereas some software development teams moved to agile development three years ago, other departments started the transition about a year ago. In one department, the SAFe framework is followed [19]. In another department, an adapted form of Scrum [31] has been used for about three years. It is a large-scale software development context with several dozen cross-functional teams collaborating across at least five locations.

Companies B, C, and E are automotive OEMs and Company F is an automotive supplier. We included different companies to validate our findings and get diverse perspectives on the continuous management of artifacts. Whereas in Company B, the transition to agile practices has been finalized several years ago, the other companies face very similar situations as Company A: Parts of the organizations still work in a plan-based way while especially

Table 1: Participants with companies, roles, and experience

No.	Company	Role	Experience
1	A	Logical architect	> 20 years
2	A	Chief architect	> 17 years
3	A	Product owner	> 15 years
4	A	Requirements manager	19 years
5	A	Process manager	> 25 years
6	A	Team leader (functional dev.)	17 years
7	A	Product owner	15 years
8	A	Scrum master	> 2 years
9	B	Software architect	9 years
10	C	Systems architect	~ 20 years
11	D	Chief Technical Officer	> 30 years
12–28	E & F	Testers	1–30 years
29–53	E & F	Software developers/engineers	1–30 years

inside of development teams, agile practices are common. The most commonly followed agile method in these companies is Scrum.

Company D is a supplier of an information management tool used in the automotive domain, supporting customers in determining what artifacts to capture in the tool and customizing the tool accordingly. The interviewee was selected to validate the findings because of his experience and insights that went beyond the scope of a single automotive company.

Table 1 gives an overview of our 53 participants in interviews and focus groups with their companies, roles, and the years of experience with systems engineering. Additionally, our study involved 41 anonymous questionnaire respondents.

4 RESEARCH METHOD

We followed a design-science approach [12]. Figure 1 shows our methods and our contributions to the knowledge base of agile information management. Our contributions are practices to manage artifacts, challenges, as well as guidelines to manage systems engineering artifacts in agile automotive contexts. Our guidelines provide principles and advice to create, maintain, and use artifacts.

The following sections present our research method in further detail. We discuss threats to validity in Section 4.4.

4.1 Understand Environment

We explored the environment from a research perspective by reviewing and discussing related work (see Section 2). We conducted

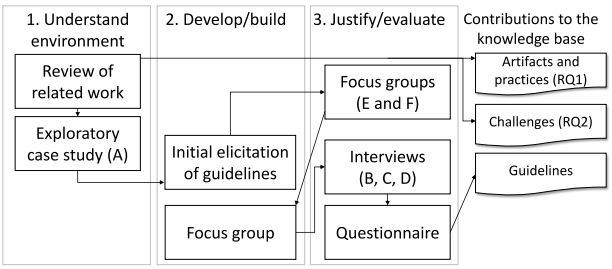

Figure 1: Overview of our design-science research approach

an informal literature review to inform our study, searching systematically for literature on the management of documentation/artifacts in large-scale agile software engineering. As we were unable to find guidelines for practitioners concerning the management of artifacts, we decided as a first step to extend the body of knowledge with an exploratory case study, collecting participants' experiences and discerning guidelines based on emerging themes in the qualitative data. We focused on Company A to get an in-depth understanding of artifacts, practices, and challenges. Over a period of 14 weeks, we conducted semi-structured interviews with participants 1–8, who are involved in agile initiatives or chosen because of their in-depth knowledge of agile methods and systems engineering artifacts. We used an interview guide[1] with open-ended and specific questions. The length of the interviews was between 45 and 80 minutes, with an average of 60 minutes.

We recorded and transcribed the interviews and followed Creswell's analysis approach for qualitative data [3]. We created an analysis guide with a detailed explanation of our analysis method[2]. We systematically *coded* the data using an *editing approach* [29], which involved the creation of new codes and constant revision, merging, and splitting of codes. We then discussed relations between codes and arrived at themes to report on.

4.2 Develop/Build

In this phase, we elicited and developed guidelines. The first ideas for potential guidelines were elicited based on our case study findings and our literature review. We then organized a focus group [34] of about two hours to discuss initial findings. The participants included two representatives from industry and two senior researchers with several years of experience with agile methods and documentation. We first presented practices, challenges, and initial ideas for guidelines. We then asked the participants to discuss the guidelines' comprehensibility, consistency, and completeness. The guidelines were refined and interdependencies and relations to challenges were discussed in several iterations. Another discussion point were the implications and relevance for researchers and practitioners. As an outcome of the focus group, the findings were grouped and structured as in Table 3.

4.3 Justify/Evaluate

This phase was concerned with the validation and refinement of our findings. Two focus groups of about 1.5 hours were organized with Companies E and F. Complementing the selection of interviewees in the case study, we set up one focus group with 17 testing experts and one with 25 software developers and engineers. First, we presented the identified artifacts, practices, and challenges. We then discussed the guidelines to understand differences between companies and their transitions to agile.

Additionally, we conducted interviews of 45 minutes with participants 9–11. We started with a short presentation. We asked the interviewees whether they agreed with the findings, whether any information was missing, and what conclusions or implications they thought of.

[1] https://www.dropbox.com/s/uprp1a4e373bdt2/interview_guide_syseng_artifacts.pdf
[2] https://www.dropbox.com/s/t2gj5w4w38qx0u5/analysis_guide_syseng_artifacts.pdf

Finally, we prepared an anonymous questionnaire to gather quantitative data. The questionnaire[3] included Likert scale questions [20] in order to measure to what degree practitioners agreed with the findings. The questionnaire included an explanatory video of about 5 minutes about main findings, terminology, and concepts. We sent the questionnaire to all participants and additional contacts from the automotive domain. We received 41 answers, filtered out the ones that only answered demographic questions on the first page, and used 31 responses for analysis.

4.4 Threats to Validity

We consider six validity threats in qualitative research, as presented by Maxwell [22]:

Descriptive Validity: We used transcripts and notes to analyze our participants' responses. The transcripts did not allow us to analyze the tone of voices, which might influence the meaning. To mitigate this, we captured timestamps that allowed us to revisit the recordings in case of unclear statements. Moreover, we contacted all participants to validate our findings using a questionnaire.

Interpretative Validity: To improve interpretative validity, we tried to establish a common terminology, use the interviewees' wording, and begin focus group sessions with presentations to explain the context of the study. We used paraphrasing during the interviews to resolve unclarities. However, we had to translate some interviews from Swedish to English which might have influenced our interpretation.

Theoretical Validity: To mitigate bias and preconceptions developed during the study, we critically discussed our findings with other researchers. We also thoroughly evaluated our findings using the methods described in Section 4.3.

Researcher Bias: The researchers' background, values, and preconceptions influence the conclusions of any qualitative study. A thorough evaluation of our findings helped to mitigate this threat. We included a diverse sample of participants in order to get different perspectives on the topic.

Reactivity: It is difficult to avoid the researchers' influence on participants and the setting of the study [22]. Many influencing factors exist, e.g., the way questions are phrased and intoned. We tried to mitigate this threat by trying to formulate the questions as unbiased as possible. The interview guide was reviewed to rephrase ambiguous questions.

Generalizability: We conducted our study together with more than 50 participants from six companies. As our design-science approach consists of several phases, the participants were involved in different parts of the data collection. Validating our guidelines in several steps with various automotive companies helps to mitigate threats to generalizability. We expect our findings to be transferable to other industries, especially in large-scale agile contexts. However, dedicated studies are needed for this purpose, as will be discussed in Section 6.

5 FINDINGS

In our study, we endeavor to understand practices and challenges of the continuous management of systems engineering artifacts and facilitate the management by designing guidelines. The data was initially collected in the exploratory case study with Company A, and confirmed by participants of other companies. In this section, we present artifacts and practices (Section 5.1), challenges (Section 5.2), and designed guidelines for practitioners (Section 5.3). Each subsection ends with a discussion of the findings with related work. Table 3 provides an overview of all findings with their implications and discussed related work.

5.1 Artifacts and Practices (RQ1)

This section answers RQ1: What are practices to manage artifacts in agile automotive systems engineering?

We found that the scope and role of artifacts have a vast impact on how they are created, used, and managed. In this study, we discuss artifacts from the perspective of a team in an organization and its collaboration with other teams. Table 2 gives an overview of artifacts, current practices, the main effort of managing them, their relevance and role, and the involved organizational areas. It should be noted that in the interviews, we did not discuss concrete instances of artifacts, but rather talked about general concepts and types. However, in some cases, interviewees also mentioned concrete documents or instances.

5.1.1 Architecture Models and Descriptions. The architecture was considered especially important by our interviewees because it gives stakeholders an understanding of the big picture of the system, and facilitates communication:

> *"If someone wants to get the full picture, then it's impossible to not have some kind of understanding how things are working, how things are coupled and connected. [...] And that's why the architecture is so important [...], to communicate about the product."* (Logical architect)

Some departments use a systems engineering tool to store the architecture in a model, whereas text descriptions are common in other departments. Company A struggles with capturing a model of the high-level architecture and managing it as a living document. It is partly due to distributed departments that store information in heterogeneous ways. The consequence of not having an architecture model is architecture degradation, which is difficult to counteract.

5.1.2 High-level Requirements. Requirements were mentioned by all interviewees. Parts of Companies A and C still work in plan-driven practices with requirements specifications that are handed over using formal processes. Requirements specifications are partly replaced by user stories.

A chief architect stated that *system requirements* on a high level are needed, specifying the functionality and interaction of features. High-level functions need to be documented as a common reference for subsequent steps in the development and testing processes. A requirements manager stressed the effort of finding the "right level of detail" and communicating this information to involved stakeholders.

5.1.3 Variability Information. Automotive companies develop products with many variants and it is essential that this complexity can be handled throughout the systems' lifecycles. A logical architect stated that variants are the artifacts he most commonly looks for in the tool. Variability information is currently not specified in

[3] https://www.dropbox.com/s/3t17e341r8mj0ch/questionnaire_syseng_artifacts.pdf

Table 2: Classification of artifacts with current practices, management effort, relevance, and affected organizational areas

Scope	Artifact	Current practices	Main effort for stakeholders	Relevance	Area
Boundary objects	Architecture Models & Descriptions	Practices differ in departments. Not consistently managed as a living document.	Handle scattered information. Counteract architecture degradation.	"Big picture" of the system's structure and design decisions.	Architecture, development, maintenance
	High-level Requirements	Used in many parts of the organization. To some extent replaced by user stories.	Find appropriate level of detail so that information is useful for stakeholders.	High-level functionality and purpose, common reference of product.	Product management, development
	Variability Information	Spread throughout several tools and systems.	Manage complexity and deal with scattered information.	Central for all phases of the lifecycle.	All areas
Locally relevant artifacts	Documentation	Varying, depending on processes. Mostly text documents.	Create documents on an appropriate level of detail, deal with changes, motivate developers.	Compliance with legislations and standards.	Function design and development
	Low-level Requirements & Tests	Less detailed descriptions, more test cases than during plan-driven development.	Keep requirements up-to-date. Find the right level of detail to avoid overspecification.	Low-level quality assurance. Comm. with suppliers.	Lower-level development and test
	Prescriptive Models	Used in teams to generate code and signal databases.	Ensure that people are aware of (otherwise implicit) rationales and intentions of models.	Code generation on a low level, ensured consistency.	Lower-level design and development

a central location but instead spread throughout several tools and systems. This complicates the collection of relevant information. However, in one of the departments' visions, it is stated that the main features and variants should be documented consistently "as a reference for development, production, and maintenance."

5.1.4 Documentation. Certain documentation is required by customers and authorities. This stays relevant when moving to agile. A chief architect mentioned documentation for customers to describe how functions can be configured. Apart from this, all interviewees mentioned documentation for authorities or compliance purposes. Documentation practices vary between organizational units, but are mostly based on textual descriptions written by developers. The effort consists mostly in finding the right level of detail, dealing with change, and motivating developers to invest time in it.

5.1.5 Low-level Requirements and Tests. Traditionally, the logical design is described with a high level of detail in a systems engineering tool. A logical architect elaborated on an exemplary requirement and described that "if you put a semicolon in the end, you could almost generate code from it." These artifacts are needed for quality assurance on a low level. However, the detailed specification requires developers to keep the models and the code up to date and in sync. Detailed requirements specifications are especially necessary when suppliers are involved, as mentioned by a logical architect. The interviewee adds that when used in in-house development, some information is "so close to the software code level that [he is] questioning" its relevance. A chief architect stated that many developers have an aversion to documenting low-level requirements and keeping them in sync with the code.

A chief architect mentioned that whereas requirements were neglected after the change to agile, the importance of test cases increased. More test cases exist than when plan-driven practices were followed and to some extent, "test cases themselves start to be the requirements." The chief architect stressed that this only works because there has not been any major development of completely new functionality.

Note, that in our model, low-level requirements sent to suppliers also belong to locally relevant artifacts as they are important for a smaller team in a company. Looking at it from an inter-organizational perspective, they can also be seen as boundary objects between the OEM and suppliers.

5.1.6 Prescriptive Models. Two kinds of prescriptive models play a role in Company A: Simulink models and models used to generate signal databases for communication networks. A chief architect described that using Simulink models for code generation motivates developers. Simulink models are automatically consistent with the final code, as opposed to low-level requirements mentioned above. A logical architect described that because it is used in this way, "it must be updated. Otherwise, it will not work."

However, there are some issues with using only prescriptive models. A Scrum master pointed out that "you capture how it's done but not why." Also, a team leader for functional development stressed that prescriptive models should be complemented with high-level descriptions:

> *"The model gives you the solution. This is how to do it. And then you don't know if that is the actual requirement. So there needs to be, I think, some written text on a higher level [...], saying what the intention is."* (Team leader)

This shows the importance of setting models into a context and pointing out their intentions and rationales.

5.1.7 Summary and discussion of findings. When analyzing the scope of using artifacts, we see that architecture models and descriptions, high-level requirements, and variability information are relevant for several teams and areas. As they are central to integrate work of different teams, these artifacts can be considered boundary objects. As mentioned before, we consider boundary objects from

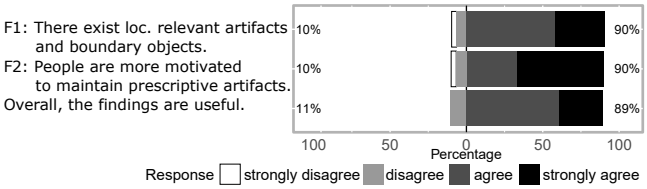

Figure 2: Questionnaire responses regarding F1 and F2

the perspective of a team and the coordination with other teams. We also found a number of artifacts relevant for a smaller group of people: Documentation, low-level requirements and tests, and prescriptive models. We arrive at the first research finding:

F1: *Artifacts are either important to support system-level thinking in an organization or play a role inside a team.*

The effort of managing locally relevant artifacts is less connected to finding information scattered across an organization, but more to the intentions and maintenance of actual information. Related work confirms especially the importance of architecture models and descriptions that can play the role of boundary objects [25]. The importance of architecture models in the automotive domain (and above all when moving to agile in the large) and the need to explicitly document them has been stressed by related work [27, 28]. We found that engineers are more motivated to keep prescriptive artifacts updated as they are directly related to other artifacts:

F2: *Stakeholders' motivation to manage prescriptive models is higher than dealing with descriptive artifacts.*

Selic mentioned that indeed, modeling languages that produce executable models are a promising approach for agile documentation [32]. Developers using prescriptive architecture models in the automotive domain keep them up to date as there is a direct impact on the implementation [9]. However, descriptive artifacts are important, e.g., to provide a "big picture" and support decisions, to capture variability, and as documentation for customers and authorities.

Figure 2 shows the questionnaire responses regarding F1, F2, and the usefulness of our findings. A majority agrees or strongly agrees with the findings and considers them useful.

5.2 Challenges With Managing Artifacts (RQ2)

In this section, we answer RQ2: What practical challenges exist with managing systems engineering artifacts in agile automotive contexts?

Ch1: Diversity vs alignment. The first challenge is concerned with the trade-off between autonomy for cross-functional teams and the required alignment of their work:

> *"You want that the teams work similarly when it comes to questions about the whole system. But at the same time, we want that the teams are as autonomous as possible so that they work at their own pace."* (Product owner)

This issue does not only have an impact on the organization, but also on artifacts like the architecture. In order to address this challenge, one department in Company A plans to create a forum for experts to exchange system-level ideas.

Ch2: Degradation of artifacts. Architecture degradation is a prevalent issue after introducing agile practices. A chief architect explained this challenge as follows:

> *"We thought the teams would try to influence the complete product. But [they] have been focused inwards, on their deliverables. When they looked outside then to see where their functionality fits. There is no drive from the organization itself to change the architecture."* (Chief architect)

Several interviewees reported that ownership and responsibilities to maintain artifacts is not naturally assumed by stakeholders. According to a Scrum master, especially architecture models and requirements face degradation.

Ch3: Mix of plan-driven and agile methods. A product owner mentioned that a big problem is that the top-level organization does not have "a full understanding of software", due to different professional backgrounds. The interviewee mentioned the importance of mechanics and the synchronization with their development cycles. These artifacts cannot be easily integrated every day but require appropriate representations, e.g., in models. Practitioners need to find agile ways of working within one team or department and synchronization mechanisms with external organizational units.

Ch4: Deciding on important artifacts. One needs to constantly identify what artifacts are important and this is difficult in practice, as stressed by a team leader for functional development and a requirements manager:

> *"A lot of things you do in an iteration is only needed to communicate between members of a team. So you have to think carefully about what is most important to maintain the software and correct it."* (Requirements manager)

A system architect from Company C faces similar problems in his organization: Besides identifying what artifacts are relevant, also the level of granularity is difficult to set. Currently, architectural models are on a too detailed level which should be reduced in the future.

Ch5: High staff turnover. The quality of systems engineering artifacts depends a lot on the individuals in charge. A product owner stated that this is a problem when, for example, the architectural design should be documented:

> *"We have had a high turnover of consultants who write this. So that functions that we created were changed because the new ones did not understand them."* (Product owner)

The staff turnover is one of the motivators to reconsider information management and capture knowledge explicitly.

Ch6: Different locations and backgrounds. A team leader for functional development saw geographic distribution as the biggest challenge and pointed out that following an agile method "requires that you can talk just like this." Different time zones complicate the scheduling of meetings.

Other issues are related to professional backgrounds: Business stakeholders "speak a different language" than engineers. A process manager stated that barriers between groups have to be overcome to establish an agile culture and "fight the control culture." This interviewee underlined the necessity of actively investing in keeping the culture.

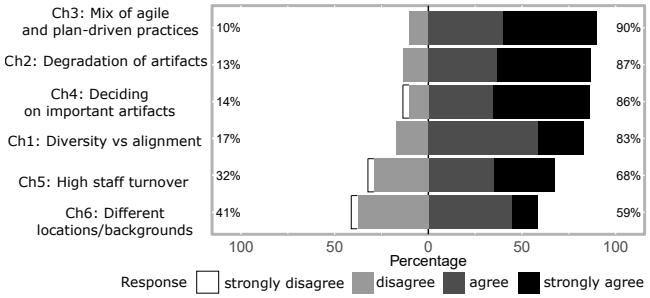

Figure 3: Questionnaire responses regarding the challenges

5.2.1 Summary and discussion of findings. We found several challenges of managing artifacts: Aligning diverse teams, degradation of artifacts, mixing plan-driven and agile methods, identifying important artifacts, staff turnover, and different locations and backgrounds. Figure 3 shows the responses to the questionnaire regarding the identified challenges. Many of the challenges were confirmed, not only by questionnaire respondents but also by interviewees and focus group participants. For instance, a tester from Company E stated that they "have seen virtually all challenges in [their company]."

We confirm several of the challenges reported by Petersen and Wohlin [26], e.g., the management overhead to align the work of different teams. While they report that the move to agile reduced the need for documentation, they do not refer to any challenges with artifacts. The challenges of deciding on important artifacts and handling artifact degradation was mentioned by other related studies [10, 18, 30]. Moreover, related work confirms that different departments typically establish agile practices at different points in time, and a mix of methods is used [17].

5.3 Guidelines to Manage Artifacts

In this section, we present the developed guidelines to support the management of systems engineering artifacts in agile automotive contexts. In Figure 4, our guidelines are depicted in boxes. Arrows show transitions between guidelines and loops show that they should be executed frequently. We describe guidelines grouped by

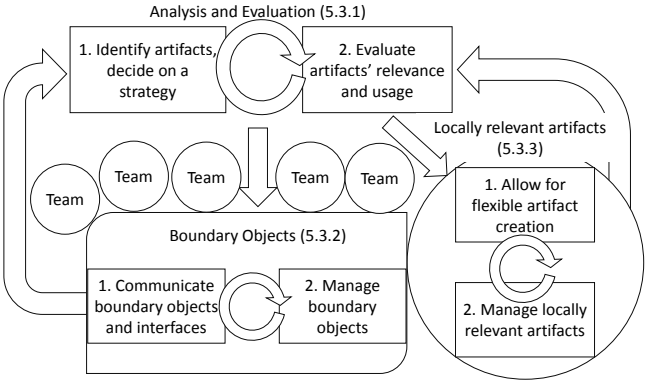

Figure 4: Guidelines to manage artifacts

three areas, i.e., analysis and evaluation (Section 5.3.1), boundary objects (Section 5.3.2), and locally relevant artifacts (Section 5.3.3).

5.3.1 Analysis and Evaluation. An agile organization needs to identify relevant artifacts and periodically evaluate them:

Identifying boundaries and deciding on a strategy should be the first step when defining an information management strategy, as a requirements manager and a process manager mentioned. Structuring the collected information in a table like Table 2 helps to categorize artifacts and discuss them. Our interviewees stressed that this collection of data has to cover the whole organization and the product's lifecycle.

We phrase the following guideline:

(G1) *Involve stakeholders from different areas to identify what artifacts are boundary objects and locally relevant artifacts. Find out how and by whom they are used, who should be responsible at what point in time, and how traceability can be established. A structure like Table 2 can support this task.*

The following aspects should be part of the strategy:

Relevance and intentions of capturing artifacts. To avoid that information is not read, the intentions of capturing information have to be known when documents are produced. A requirements manager explained that it is a central task to define boundary objects and make sure that they exist for a reason. An agreement should be found as to what level of detail is needed to avoid "over-specification." In Table 2, we capture this aspect in the "Relevance" column.

Affected areas and stakeholders of artifacts. A requirements manager and a product owner stressed that responsibilities and ownership should be clarified. A chief architect gave several examples of teams that varied the level of internal documentation over time, for instance, due to varying complexity of the developed functionality and the team members' backgrounds. In Table 2, we show affected organizational areas in the "Area" column.

Strategies for traceability management. Apart from the consumers and producers, traceability between artifacts should be part of the strategy. Six interviewees mentioned the importance of traceability. For instance, a logical architect stressed the importance of traceability for change impact analysis. A Scrum master and a chief architect stated that traceability is especially needed to demonstrate compliance with standards.

Evaluating artifacts' relevance and usage is necessary at periodic intervals to identify whether artifacts are still needed or should be discarded. A Scrum master mentioned that features to automatically analyze the usage of information might help with these activities in the future.

(G2) *Make sure that you evaluate artifacts' relevance and usage at frequent intervals. Depending on how the system evolves throughout its lifecycle, artifacts might become more relevant or should be discarded. Boundary objects could be converted into locally relevant artifacts and vice versa.*

5.3.2 Management of Boundary Objects. Boundary objects impact several groups and should thus be handled with care. We found

that practitioners should (1) communicate boundary objects and interfaces, and (2) manage boundary objects.

Communicating boundary objects and interfaces is supported by the currently used systems engineering tool where artifacts are stored and linked to top-level objects. One example of a top-level object is the high-level architecture, including components and their interfaces. A requirements manager stressed that one needs to "define why things are connected. Otherwise, you start to add [information] but you actually break an intention."

For some boundary objects, the communication of decisions can be done in two levels of communities or groups. A product owner explained that in a top-level community, decisions concerning boundary objects could be taken, and then propagated to lower level groups "to spread the knowledge." In this context, it is recommendable to store artifacts in a tool that is easy to access, to mitigate issues of scattered information. According to the CTO from Company D, different views of the data can help teams see their parts in the "big picture" and satisfy different information needs.

We phrase the following guideline:

(G3) *For each boundary object within your organization, establish a group of representatives from affected teams to discuss issues and later propagate that information. Store information in an accessible tool.*

In some cases, we saw that requirements specifications play the role of boundary objects to collaborate with suppliers. In these cases, the communication is more formal than in the case of intra-organizational boundary objects.

Managing boundary objects and refining them with time is important. A requirements manager stated that one needs to decide on relevant boundary objects upfront but also come back to them later on. Especially for the architecture, it is important to capture the right information at the right point in time, as we express in Guideline 4:

(G4) *Find a lightweight and flexible approach to defining high-level artifacts upfront. Exploit this information to make impact analysis and changes, to avoid suboptimal decisions. With time, the artifacts should be refined and become more mature.*

A Scrum master mentioned that "keeping it up to date is the tricky part" and that regular reviews are necessary to identify relevant artifacts and see "which parts are less needed." However, artifacts should only be updated if the outdatedness becomes a problem and their relevance should be continuously assessed. Some can be deleted completely whereas others should be archived.

5.3.3 Management of Locally Relevant Artifacts. Locally relevant artifacts are managed within a well-connected team. We found that practitioners should (1) allow for flexible ways of artifact creation, and (2) manage locally relevant artifacts.

Allowing for flexible artifact creation is relevant as the order of artifact creation might change and become more flexible when transitioning to agile practices. A product owner stated that it is not required to have formal requirements specification on a detailed level in the beginning. At a later point in time, requirements are needed for maintenance, compliance with safety regulations, etc. Artifacts are typically not directly needed at the beginning of the development but need to be captured and versioned at a later point:

> *"[There is] the concept phase, where it is very fluent and agile. Then settling it down into the formal versioning. And then sometime it [...] lies around, but is still needed."* (Scrum master)

Whereas some documentation "lies around", as this interviewee puts it, it is necessary to update other artifacts and keep them consistent. Concretely, we found that documentation or low-level requirements and tests are not required in the concept phase, but need to be in place at a later point in time for maintenance and aftermarket.

(G5) *Produce locally relevant artifacts, especially those for documentation, as late as possible and only when they are actually needed. If possible this documentation should be automatically generated from other artifacts and code.*

Managing locally relevant artifacts in continuous ways is a need expressed by several interviewees. It is also important to establish and maintain traceability to high-level artifacts. This point was seen as essential by a software architect from Company B who is involved in a project improving traceability. A logical architect stated that both the traceability and the artifacts themselves need to be maintained:

> *"As many people rely on this information, especially to reason about changes, it is important that it is maintained. Today, it is up to each engineer to take care of this task."* (Logical architect)

We found that especially for Simulink models and other prescriptive models there is a high intrinsic motivation to keep artifacts updated. Stakeholders see a use in creating and maintaining this information.

(G6) *Aim to make locally relevant artifacts reusable (as with prescriptive models) and convey their relevance and use. Establish traceability to boundary objects.*

5.3.4 Summary and discussion of findings. We presented general guidelines regarding the analysis and evaluation of artifacts and dedicated guidelines for the management of boundary objects and locally relevant artifacts. Figure 5 shows the responses to the questionnaire regarding the guidelines. A majority of the respondents consider the guidelines at least moderately valuable. In the focus group sessions in Companies E and F, several participants stressed the usefulness of the distinction between boundary objects

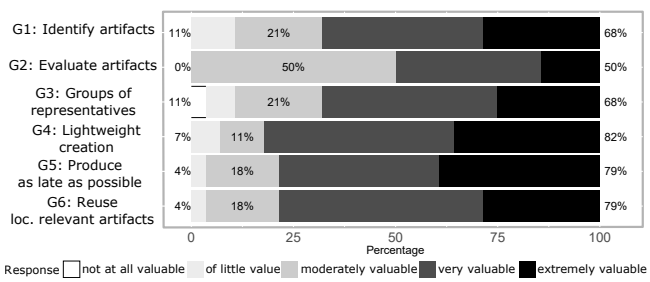

Figure 5: Questionnaire responses regarding the guidelines

Table 3: Our findings, participants, implications, relation to challenges, and discussed related work

	Finding	Part.	Implications/Relation to Challenges (RQ2) [10, 17, 18, 26, 30]
Artifacts (RQ1) [9, 25, 27, 28, 32]	F1: Artifacts are either important to support system-level thinking in an organization or are used inside a team.	1, 2, 4, 8, E&F[4]	I1: Tool solutions should allow for flexibility with the definition of important artifacts and processes to create, update, and use them. Practitioners would benefit from methods and tools supporting organization-wide collaboration.
	F2: Stakeholders' motivation to manage prescriptive models is higher than dealing with descriptive artifacts.	1, 2, 5–8	I2: Tools and methods supporting the automatic management of prescriptive models would be useful for practitioners. Especially descriptive models that do not play the role of boundary objects should be carefully assessed.
Guidelines [2, 9, 15, 23, 30]	G1: Involve stakeholders from different areas to identify what artifacts are boundary objects and locally relevant artifacts.	2, 3, 4, 6, 10	I5: Reflecting on boundary objects and locally relevant artifacts helps to identify important artifacts (Ch4). Practitioners can establish boundary objects as a "contract" between plan-driven and agile teams (Ch3) and across locations and backgrounds (Ch6).
	G2: Make sure that you evaluate artifacts' relevance and usage at frequent intervals.	4, 8	I6: Following this guideline helps to counteract challenges with the degradation of artifacts (Ch2). It can also mitigate the consequences of high staff turnover (Ch5).
	G3: For each boundary object within your organization, establish a group of representatives.	1–7, 11	I7: This guideline can constitute a mechanism to align the work of different teams while allowing for diversity (Ch1).
	G4: Find a lightweight and flexible approach to defining high-level artifacts upfront.	2, 4, 6, 8, E&F[4]	I8: Researchers should work towards lightweight methods to find the right level of upfront specification. Practitioners can follow the guideline to counteract artifact degradation from the start (Ch2).
	G5: Produce locally relevant artifacts as late as possible and only when they are actually needed. Aim to generate artifacts	3, 5, 8	I9: Producing artifacts as late as possible helps to avoid the degradation of artifacts (Ch2). This guideline is in line with F2 as the motivation to manage prescriptive models (for artifact generation) was found to be high.
	G6: Make locally relevant artifacts reusable and convey their relevance and use. Establish traceability to boundary objects.	2, 7, 8, 9	I10: Practitioners trying to align different groups in the organization (Ch1) should focus on traceability to boundary objects. Researchers can focus on increasing the reusability of artifacts to alleviate the maintenance effort.

[4] E&F means that there existed a general consensus about the finding/guideline in the focus groups.

and locally relevant artifacts. They want to work towards concrete lightweight creation approaches in the future.

G4 is in line with the more specialized related work on "just enough architecting", i.e., postponing all unessential decisions until the appropriate time [23]. G3 recommends to create groups of representatives for each boundary object, which is related to the literature on *Communities of Practice* [15].

Rüping suggested to start agile documentation early in the project and update it depending on how frequently it is read, potentially even on a day-to-day basis [30]. Apart from how frequently users access the artifacts, we found that also the role of the artifacts in the organization plays a role and should be assessed when updating them.

Finally, locally relevant artifacts are always relevant within a "team." Contrariwise, there can be different levels of boundary objects, shared between different companies (e.g., an OEM and its suppliers), shared in the entire company, or with a scope limited to a few teams.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we sought to create guidelines for the management of systems engineering artifacts in agile automotive contexts, based on an understanding of challenges and practices to manage artifacts. We used a design-science approach with more than 50 practitioners from six companies.

Table 3 shows an overview of our findings. For each of the findings, we present implications for research and practice. We also show the relation between the guidelines and the identified challenges (RQ2) in the Implications column. In the leftmost column, we show references to related work discussed in Section 5. The participants (see Table 1) who mainly support the findings are shown in the column "Part."

Our findings suggest that it is central to identify artifacts and evaluate artifacts' relevance and usage at frequent intervals. Boundary objects (used to cross boundaries between several groups) should be managed upfront with a lightweight approach and be continuously revised. Artifacts used within one team can be managed with a flexible approach and should only be created when they are actually needed.

This study allows practitioners to get insights into guidelines for artifact creation and maintenance. Our guidelines can be used as a starting point to critically reflect on the use and relevance of artifacts and create company-specific strategies. Researchers can benefit from a new perspective on continuous management of systems engineering artifacts based on empirical evidence. Our findings indicate that it might be desirable to not design all artifacts up-front, but allow for flexible artifact creation.

Future work: Methods and tools to support inter-team collaboration to manage boundary objects could help practitioners in the future. Researchers can also work towards tool support to assess artifacts' relevance and usage depending on how artifacts are accessed by stakeholders in different teams.

Future work is needed to understand boundary objects and locally relevant artifacts in other contexts and domains. The actual boundary objects and locally relevant artifacts might differ depending on the involved disciplines and relationships of involved

companies (e.g., supplier and OEM relationships). For instance, the difficulty of managing variability information might not be as grave in other industries. We believe that the presented guidelines are valuable also for other domains, but dedicated studies are needed to scrutinize their applicability.

ACKNOWLEDGMENTS

We are very grateful for the support of the participants in this study and we thank for all the clarifications provided when needed.

This work was partially supported by the Next Generation Electrical Architecture (NGEA) and NGEA step 2 projects by VINNOVA FFI (2014-05599 and 2015-04881), the Software Center Project 27 on RE for Large-Scale Agile System Development, and by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] Julian M. Bass. 2016. Artefacts and agile method tailoring in large-scale offshore software development programmes. *Information and Software Technology* 75 (2016), 1–16. https://doi.org/10.1016/j.infsof.2016.03.001

[2] Johan Kaj Blomkvist, Johan Persson, and Johan Åberg. 2015. Communication through Boundary Objects in Distributed Agile Teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, USA, 1875–1884. https://doi.org/10.1145/2702123.2702366

[3] John W. Creswell. 2008. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (3 ed.). Sage Publications Ltd., London, England.

[4] Joseph D'Ambrosio and Grant Soremekun. 2017. Systems engineering challenges and MBSE opportunities for automotive system design. In *International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2075–2080. https://doi.org/10.1109/SMC.2017.8122925

[5] Philipp Diebold, Thomas Zehler, and Dominik Richter. 2017. How do agile practices support automotive SPICE compliance?. In *Proceedings of the International Conference on Software and System Process (ICSSP'17)*. ACM, New York, NY, USA, 80–84. https://doi.org/10.1145/3084100.3084108

[6] Torgeir Dingsøyr, Nils Brede Moe, Tor Erlend Faegri, and Eva Amdahl Seim. 2018. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering* 23, 1 (Feb 2018), 490–520. https://doi.org/10.1007/s10664-017-9524-2

[7] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 9-10 (2008), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

[8] Christof Ebert and John Favaro. 2017. Automotive Software. *IEEE Software* 34, 3 (may 2017), 33–39. https://doi.org/10.1109/MS.2017.82

[9] Ulf Eliasson, Rogardt Heldal, Patrizio Pelliccione, and Jonn Lantz. 2015. Architecting in the Automotive Domain: Descriptive vs Prescriptive Architecture. In *Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA'15)*. IEEE, 115–118. https://doi.org/10.1109/WICSA.2015.18

[10] Geir K. Hanssen, Aiko Fallas Yamashita, Reidar Conradi, and Leon Moonen. 2009. Maintenance and agile development: Challenges, opportunities and future directions. In *Proceedings of the 25th IEEE International Conference on Software Maintenance (ICSM'09)*. IEEE, 487–490. https://doi.org/10.1109/ICSM.2009.5306278

[11] Rogardt Heldal, Patrizio Pelliccione, Ulf Eliasson, Jonn Lantz, Jesper Derehag, and Jon Whittle. 2016. Descriptive vs Prescriptive Models in Industry. In *Proceedings of the 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016)*. ACM, New York, NY, USA, 216–226. https://doi.org/10.1145/2976767.2976808

[12] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28 (2004), 75–105. https://doi.org/10.2307/25148625

[13] Dan X. Houston. 2014. Agility beyond software development. In *Proceedings of the International Conference on Software and System Process (ICSSP'14)*. ACM, New York, NY, USA, 65–69. https://doi.org/10.1145/2600821.2600837

[14] Markus Hummel, Christoph Rosenkranz, and Roland Holten. 2013. The role of communication in agile systems development: An analysis of the state of the art. *Business and Information Systems Engineering* 5, 5 (2013), 343–355. https://doi.org/10.1007/s12599-013-0282-4

[15] Tuomo Kähkönen. 2004. Agile Methods for Large Organizations – Building Communities of Practice. In *Proceedings of the Agile Development Conference*. IEEE, 2–10. https://doi.org/10.1109/ADEVC.2004.4

[16] Mira Kajko-Mattsson. 2008. Problems in agile trenches. In *Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM'08)*. ACM, New York, NY, USA, 111. https://doi.org/10.1145/1414004.1414025

[17] Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Kitija Trektere, Fergal McCaffery, Garousi Vahid, Michael Felderer, Oliver Linssen, Eckhart Hanser, and Christian Prause. 2018. Hybrid Software Development Approaches in Practice: A European Perspective. *IEEE Software* PP, 99 (2018). https://doi.org/10.1109/MS.2018.110161245

[18] Lina Lagerberg, Tor Skude, Par Emanuelsson, Kristian Sandahl, and Daniel Stahl. 2013. The impact of agile principles and practices on large-scale software development projects: A multiple-case study of two projects at Ericsson. In *International Symposium on Empirical Software Engineering and Measurement*. IEEE, 348–356. https://doi.org/10.1109/ESEM.2013.53

[19] Dean Leffingwell. 2007. *Scaling Software Agility: Best Practices for Large Enterprises (The Agile Software Development Series)*. Addison-Wesley Professional.

[20] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 22, 140 (1932), 5–55.

[21] Thomas W. Malone and Kevin Crowston. 1994. The Interdisciplinary Study of Coordination. *Comput. Surveys* 26, 1, Article 1 (March 1994), 33 pages. https://doi.org/10.1145/174666.174668

[22] Joseph Alex Maxwell. 2012. *Qualitative Research Design: An Interactive Approach*. SAGE Publications.

[23] Robert L. Nord and James E. Tomayko. 2006. Software architecture-centric methods and agile development. *IEEE Software* 23, 2 (March 2006), 47–53. https://doi.org/10.1109/MS.2006.54

[24] Wanda J. Orlikowski and JoAnne Yates. 1994. Genre Repertoire: The Structuring of Communicative Practices in Organizations. *Administrative Science Quarterly* 39, 4 (1994), 541–574. https://doi.org/10.2307/2393771

[25] Lars Pareto, Peter Eriksson, and Staffan Ehnebom. 2010. Architectural Descriptions as Boundary Objects. In *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (MODELS 2010)*. Springer Berlin Heidelberg, Berlin, Heidelberg, 406–419. https://doi.org/10.1007/978-3-642-16129-2_29

[26] Kai Petersen and Claes Wohlin. 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study. *Empirical Software Engineering* 15, 6 (2010), 654–693. https://doi.org/10.1007/s10664-010-9136-6

[27] Christian R. Prause and Zoya Durdik. 2012. Architectural design and documentation: Waste in agile development?. In *Proceedings of the International Conference on Software and System Process (ICSSP'12)*. IEEE, 130–134. https://doi.org/10.1109/ICSSP.2012.6225956

[28] Alexander Pretschner, Manfred Broy, Ingolf H. Krüger, and Thomas Stauner. 2007. Software engineering for automotive systems: A roadmap. In *Future of Software Engineering (FoSE'07)*. IEEE, 55–71. https://doi.org/10.1109/FOSE.2007.22

[29] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (19 Dec 2009), 131–164. https://doi.org/10.1007/s10664-008-9102-8

[30] Andreas Rüping. 2003. *Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects* (1 ed.). Wiley Publishing.

[31] Ken Schwaber and Mike Beedle. 2001. *Agile Software Development with Scrum* (1 ed.). Prentice Hall PTR.

[32] Bran Selic. 2009. Agile documentation, anyone? *IEEE Software* 26, 6 (2009), 11–12. https://doi.org/10.1109/MS.2009.167

[33] Susan Leigh Star and James R. Griesemer. 1989. Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science* 19, 3 (1989), 387–420. https://doi.org/10.1177/030631289019003001

[34] Monica Chiarini Tremblay, Alan R. Hevner, and Donald J. Berndt. 2010. The Use of Focus Groups in Design Science Research. In *Integrated Series in Information Systems*. Springer US, 121–143. https://doi.org/10.1007/978-1-4419-5653-8_10

[35] Stefan Voigt, Jörg von Garrel, Julia Müller, and Dominic Wirth. 2016. A Study of Documentation in Agile Software Projects. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '16)*. ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/2961111.2962616