Defining Requirements Strategies in Agile: A Design Science Research Study

Defining Requirements Strategies in Agile: A Design Science Research Study

Amna Pir Muhammad[1], Eric Knauss[1], Odzaya Batsaikhan[1], Nassiba El Haskouri[1], Yi-Chun Lin[1], and Alessia Knauss[2]

[1] Dept. of Computer Science and Eng., Chalmers | University of Gothenburg, Gothenburg, Sweden
[2] Zenseact AB, Gothenburg, Sweden

Abstract. Research shows that many of the challenges currently encountered with agile development are related to requirements engineering. Based on design science research, this paper investigates critical challenges that arise in agile development from an undefined requirements strategy. We explore potential ways to address these challenges and synthesize the key building blocks of requirements strategies. Our design science research rests on a multiple case study with three industrial cases in the domains of communication technology, security services, and automotive. We relied on a total of 20 interviews, two workshops, participant observation in two cases, and document analysis in each of the cases to understand concrete challenges and workflows. In each case, we define a requirements strategy in collaboration with process managers and experienced engineers. From this experience, we extract guidelines for defining requirements strategies in agile development.

Keywords: requirements strategy, design science research, requirements engineering, large-scale agile development

1 Introduction

Agile development methodologies aim to shorten the time to market and incorporate maximum changes during the sprint to meet customer needs [21] and have been adapted at small-scale as well as large-scale organizations [18]. With its' focus on interactions and working software over rigid processes and extensive documentation, traditional well established Requirements Engineering (RE) processes have been neglected. Research shows that many of the challenges currently encountered with agile development are related to requirements engineering [14] for example, misunderstanding customer needs, missing high-level requirements, and difficulty to achieve having just enough documentation.

In this study, we identify specific RE-related challenges and related solution strategies in agile development. Based on this knowledge, we derive necessary building blocks as different viewpoints that should be considered when thinking strategically about RE in agile development. In this, we are inspired by test

strategies, which guide testing activities to achieve the quality assurance objectives [8] and which mandate that each project must have its test plan document that clearly states the scope, approach, resources, and schedule for testing activities [20]. We argue that defining a so called *requirements strategy* similar to a test strategy for RE can be critical for successful agile development.

In this paper, we aim to establish the concept of *requirements strategy for agile development* by investigating the following research questions based on iterative design science research in three industrial case studies.

RQ1 Which challenges arise from an undefined requirements strategy?
RQ2 How do companies aim to address these challenges?
RQ3 Which potential building blocks should be considered for defining a requirements strategy?

Since we particularly target agile development, we aimed to investigate requirements challenges independent from process phases or specific documents. Instead, we took the lens of *shared understanding* [7] to investigate different RE activities (i.e., elicitation, interpretation, negotiation, documentation, general issues). According to Fricker and Glinz, an investigation of *shared understanding* may primarily target how such shared understanding is *enabled* in an organization, how it is *built*, and how it is *assessed* for relevance [7].

Therefore, our contribution are guidelines on how requirements strategies should be described for agile development. Through building three complementary perspectives, we see that the requirement strategy guidelines capture relevant information and provide a useful overview. We suggest that a strategy defines the structure of requirements to create a shared language, define the organizational responsibilities and ownership of requirements knowledge, and then map both structure and responsibilities to the agile workflow.

In the next section, we provide the related work for our study. In Section 3 we elaborate on our design science research method before revealing our findings in Section 4 in order to answer our research questions. Then, in Section 5, we present our artifact - guidelines on how to define a requirements strategy for RE in agile development. Finally, we discuss and conclude our paper in Section 6.

2 Related Work

Literature shows that many companies adopt agile methods [13, 17] due to its numerous benefits, for example, flexibility in product scope which improves the success rate of products [3], in contrast to traditional development methods [27]. Furthermore, agile methods incorporate maximum change and frequent product delivery [13], encourage changes with low costs, and provide high quality products in short iterations [21]. Due to its success, agile methodologies are become widely popular and adopted by both small and large companies [18]. The term *large-scale agile* refers to agile development, which includes large teams and large multi-team projects [4]. Dikert et al. define large-scale agile development as agile development that includes six or more teams [3].

However, despite the success of agile methods, large-scale companies also still face several challenges. Dikert et al. (2016) [3] conducted a systematic literature review of empirical studies. The authors identified several challenges and success factors for adopting agile on a large scale. The most mentioned challenges are change resistance, integrating non-development functions, difficulty to implement agile methods (misunderstanding, lack of guidance), requirement engineering challenges (e.g., high-level requirements management largely missing in agile methods, the gap between long and short term planning). Based on a literature review, Dumitriu et al. (2019) [5] identified 12 challenges of applying large-scale agile methods at the organization level. The most cited challenge is the coordination of several agile teams. Kasauli et el. (2021) [14] identified 24 challenges through multiple case studies across seven large-scale companies. Some of the identified challenges are building long lasting customer value, managing experimental requirements, and documentation to complete tests and stories. The authors conclude that strong RE approaches are needed to overcome many identified challenges.

When it comes to RE in agile development, challenges that have been identified include lack of documentation, project budget, time estimation, and shared understanding of customer values [1,6,12,15,24] First attempts have been made to tackle some of the challenges of RE in agile development, e.g., Inayat et al. and Paetsch et al [12,22]. suggest combining traditional RE with agile methods and encounter challenges like how much documentation is just enough documentation [11] to have a shared understanding of customer values.

Considering that there are many challenges related to RE that can be solved through RE approaches, this paper proposes to use the concept of a requirements strategy as a method to define requirements engineering practices to tackle challenges related to requirements engineering in agile development.

3 Design Science Research Method

Our research aims to design suitable ways of defining requirements strategies for organizations with agile software development. Such requirements strategies should be suitable for addressing real-world needs, incorporating state-of-the-art knowledge, and ideally being empirically evaluated in practice. Thus, we decided that design science research [10,28,29] is a good fit.

Design Science Research. Our research questions are targeted towards design science research, with RQ1 focusing on the problem domain, RQ2 investigating potential solutions, and RQ3 targeting towards deriving the artifact. Our artifact are guidelines on how to define a requirements strategy in agile development. Refining on well-known challenges with RE in agile development, we needed to gain in-depth insights into those challenges related to a lack of a clear requirements strategy throughout the agile development organization (RQ1). Throughout our cases, we discuss those challenges with respect to potential mitigation strategies (RQ2) for those challenges. Finally, we systematically synthesize the building blocks of requirements strategies (RQ3) from solution strategies.

Table 1. Research questions in relation to cases and research cycles

	Case 1	Case 2			Case 3
	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5
Identify challenges (RQ1)	■■■■■■	■■■■■■	■■■■□□	■■□□□□	■■■□□□
Identify solutions (RQ2)	■■■■□□	■■□□□□	■■■■■■	■■■■□□	■■■□□□
Building blocks (RQ3)	■■□□□□	□□□□□□	■□□□□□	■■□□□□	■■■■■■
Data source	11 Interviews, document analysis	9 Interviews, participant observation, document analysis, 1 workshop			Participant observation, document analysis, 1 workshop

Inspired by the regulative cycle [29], the artifact (guidelines for defining a requirements strategy based on good practices from our cases) has iteratively evolved, allowing to refine the knowledge with respect to each research question. Table 1 provides an overview of our research method. As can be seen, we relied on three case studies over which we distribute a total of five research cycles. The cycles differ in how much focus is given to each of our three research questions:

Case 1 - *Exploring the problem through the lens of requirements engineering and shared understanding:* Case 1, an information and communication technology company, focuses on a strategy to achieve a shared understanding about customer value throughout the development organization. Our research aims were two-fold: understand the real world problem and conceptualize a design artifact that may address this problem. Within a Master's thesis [2], we developed an appropriate lens that combined both the concept of shared understanding (as expressed by Glinz and Fricker through enabling, building, and assessing shared understanding [7]) and commonly used RE activities (such as elicitation, interpretation, negotiation, and documentation). We then relied on 11 interviews to understand customer value and its common understanding, information sharing, tools and channels for sharing, and tools and methods for documenting. Since our first cycle focuses on the exploration of the problem we locally relied on the case study research method for our research with respect to Case 1 [2]. As Table 1 shows, we complemented the interviews with document analysis to produce an overview of challenges and related solution strategies.

Case 2 - *Refining the requirements strategy artifact iteratively:* We then followed up in *Case 2*, a company producing security smart alarms and services. In this case, the focus was on a more general requirements strategy that covers both stakeholder and system requirements. Again, through a Master's thesis [8], we investigated concrete requirements challenges of an agile team, defined a requirements strategy along the lines of the result from Case 1, and investigated in depth to what extent it could help with the challenges in practice. At this point, we further focused on investigating whether there are reusable building blocks for a requirements strategy.

Case 3 - *Applying and evaluating the artifact:* Finally, we brought our experience from the previous two cases into *Case 3*, an automotive supplier, focusing on complex safety critical and software intense systems. Here, we focus less on challenges and solution strategies, in particular, since the case company already had compiled a good overview. Instead, our focus is to refine the artifact (guidelines for requirements strategies) by discussing, applying, and improving our understanding of the building blocks of a requirements strategy. At the time of the research, the first author of this paper did an internship with this automotive supplier and helped to make the requirements strategy explicit. The company did already identify some of the challenges and making a first step towards implementing their solution strategies. Thus, she was able to investigate the phenomenon as a participant observer, contrasting it with documents and ways of working in the practical context, allowing us to fine-tune our guidelines for requirements strategies to provide an overview of challenges and solution strategies for continuous process improvement.

Data collection. We relied on a mix of different methods for data collection, including interviews, participant observation, document analyses, and workshops.

Interviews - We relied on interviews in Case 1 and Case 2, in particular, to understand the problem (RQ1) in each specific case. 20 interviews were conducted using interview guides (details in [2, 8]), relying on a mix of closed and open-ended questions. Interviews were recorded, transcribed, and then coded. In both cases, we recruited interviewees through purposeful sampling [23]. We relied on convenience sampling so that interviewees were both available and knowledgeable. We employed diversity sampling to capture the views of multiple relevant roles and stakeholders and similarity sampling to ensure that we received multiple views for each relevant perspective for triangulation.

Participant Observation - The fourth author was very familiar with Case 2, in which she had worked for several years before starting her Master thesis [8]. Her work included defining a testing strategy, which provided intimate knowledge about the agile ways of working in the case company, which were also helpful for understanding the requirements-related challenges, defining a requirements strategy, and conducting the evaluation in Case 2. The first and last authors did work with RE and the continuous improvement of requirements processes of Case 3. Through this work, we were able to verify that previously identified challenges in Case 3, as well as initiatives to address them, were of similar nature and matched well with our recent work on requirements strategies. Both co-authors relied on our requirements strategy work to support the ongoing initiatives on requirements processes and on integrating RE practices into agile ways of working. This allowed us to evaluate the suitability of our requirements strategy concept. Knowledge from these activities was collected through field notes, presentations given at the case company, and discussions with other co-authors.

Document Analysis - In all three cases, a subset of the authors studied the documents related to the flow of requirements (Case 1: Author 3 and 5, Case 2: Author 4, Case 3: Author 1 and 6). Since all three cases embraced agile ways of working, we considered that not all relevant information might be found in formal

documents. However, we ensured that documentation did match or at least not contradict our data. We found relevant documentation of requirements, e.g., as user stories, in all three cases to match and support our other data sources. Document analysis also allowed us to better understand the implied requirements strategy and processes.

Workshops - We relied on two workshops in cases 2 and 3 to evaluate the proposed requirements strategies and, by that, also our requirements strategy guidelines. In case 2, a workshop was conducted to present the challenges identified, the proposed solution candidates, and different versions of the specific requirements strategy of each case. In case 3, a workshop was used to understand the requirements strategy that was used to address certain challenges. Expert participants were sampled similarly as for interviews. They were asked to bring up additional challenges that we may have missed, give feedback on the criticality of the challenges that we had found, provide their opinion about the solution candidates, and evaluate the structure, presentation, and concrete advice on the requirements strategies. Depending on the circumstances in each case, we recorded and transcribed, took live notes for all participants to see, or shared our notes after the workshop for validation.

Data analysis. In order to analyze interview transcripts, field notes from participatory observation and document analysis, and workshop notes/transcripts, we relied on typical coding approaches for qualitative research [26]. This allowed us to report on challenges that relate to a missing or undefined requirements strategy. For example, the following quote from Case 1 contributed to identifying the challenge *d) lack of communication with customers*: *"The thing that sometimes does not work as it should, is communication with some of the customer units. It heavily depends on the competence of the customer unit people."*

In each case, we had access to an industry champion from the respective company, who helped to suggest practical solutions. For example, the following quote suggested a solution for challenge above as *c) ability to initiate on demand meetings with customer representatives: "The right people to nail down a requirement should be put together in the meeting to have a requirement handshake."* In addition, the second author was involved as an academic supervisor in all three cases, providing pointers toward relevant published knowledge. We regularly presented and discussed our findings at the case companies, focusing on strong tracing between challenges, solution candidates, and the proposed requirements strategies. Together with iterative refinements, this allowed us to analyze the data in depth.

Threats to Validity. *Internal validity* aims to reveal factors that affect the relationship between variables, factors investigated, and results. A key threat to internal validity of this study is the risk of misinterpretations, particularly during the interviews and observations. *Construct validity* defines the extent to which the investigated measures describe what the researchers analyze and what is studied according to research questions [25]. We mitigated threats to internal and construct validity through interacting closely with industry partners in study

design and interpretation of results. We also worked iteratively and triangulated across our iterations and cycles as well as different data sources. *External validity* relates to identifying to what extent our findings can be generalized [19]. We identified common challenges in all three case companies. Thus, we expect that in particular the structure and perspectives of our requirements strategy guidelines can be transferred to other contexts. *Reliability* reflects to what extent other researchers can produce the same results repeating the same study methodology. In a qualitative study, it is always hard to achieve *reliability* since one cannot argue based on statistical significance. We mitigate this threat by elaborating our research method in detail to support other researchers in replicating our research and in recovering from any possible differences in results.

4 Findings

4.1 RQ1: Which challenges arise from an undefined requirements strategy?

The left column of Table 2 depicts RE challenges identified based on our three cases that are encountered without a clear requirements strategy existing for agile development.The challenges are categorized in RE practices and related to Glinz and Fricker's [7] practices of shared understanding, grouped in three categories, i.e., enable, build, and assess. Enable practices describe what is needed to form and establish a common foundation of knowledge. Building practices aim to provide the structured knowledge that can be communicated within the team or company through explicit artifacts or by constructing a body of implicit knowledge for shared understanding. Assessing methods determine how all team members have a shared understanding of a topic or artifact. Some methods can be used for both building and assessing practices. Indices indicate in which of the cases a challenge was relevant.

a) Teams struggle to integrate RE in their agile work efficiently[1,2,3] - Agile development enables organizations to respond to change. If there is a change in code and tests, the requirements should usually be updated. Or if requirements change, then the code and tests need to be adjusted accordingly. Teams struggle with this since requirements tools do not integrate well with agile software development work and do not support parallel changes from several teams. Thus, it is hard to integrate RE work into the agile work effectively.

b) No formal event to align on customer value[1] - There were no formal events to create awareness of customer value in Case 1. Even when the customer unit took the initiative and organized some events, there were only a few participants. Such events must be better integrated in the organization and workflow.

c) Insufficient customer feedback[1,2] - In Case 1 and 2, developers lack customer feedback, which is crucial for agile workflows.This can be due to a lack of formal events, or due to scale and distance to customers. It impacts the ability of an organization to assess whether shared understanding has been reached. Customer feedback should be integrated into the workflow across organizational levels and take into account the specific needs of product owners and developers.

d) Lack of communication with customer[1] - Customer-facing units have a key role and are on the boundary between development teams and customers. We encounter difficulties with communication in both directions: between customer-facing teams and development teams and between customer-facing teams and the customers. These challenges are mainly due to a lack of systematic guidance on how such communication should take place, thus depending completely on the individual skills of those involved. Companies would have to find a way to ensure good and transparent communication, for example by having product owners moderating direct meetings between developers and customers.

e) Who owns customer value[1] - Requirements enter the development organization mainly through the hierarchy of product owners (PO) in Case 1. However, a significant amount of requirements originate from other sources, e.g., development teams or system managers, and in those cases, it is less clear who is able to define or who owns the customer value.

f) Inconsistent elicitation[2] - POs or application specialists collect requirements when needed and apply techniques such as interviews. There is, however, no systematic strategy to elicitation integrated into the workflow.

g) Lack of feedback on elicitation[2] - Without a systematic validation of elicitation results, misunderstandings will only surface late in the agile workflow, e.g. during acceptance testing and result in additional costs and effort.

h) Unclear why requirement is needed[2] - Due to scale, distance to customers, or because a customer value description is not available for developers (see Challenge o), application specialists and POs may lack information on why specific low-level requirements are needed. This can result in a gap between what product owners want and how the development teams interpret their requirements.

i) Wrong assumptions about customer value[1] - Interviewees highlighted that one of the significant challenges is that people assume customer value based on their tacit knowledge, leading to the development of faulty assumptions.

j) Unclear and volatile customer needs[2] - Requirements change, for example when the customer changes their mind or did not have a detailed opinion in the beginning. When assessing the interpretation of requirements, this can cause friction, since the team tries to "hit a moving target".

k) Decentralized knowledge building[3] - Different teams develop requirements, architecture, and also processes at the same time. This decentralized way of working is needed to yield the benefits of agile work at scale, but requires some infrastructure to enable knowledge sharing and alignment. Otherwise, conflicting decisions will be made throughout the organization.

l) Focus on technical details[1,2] - Often customer value is not explicitly described; instead, customer needs and technical solutions are more explicit. When we asked participants in Case 1 and 2 to describe the customer value of specific requirements, they explained the technical solutions rather than customer values. This finding is consistent with documentation, where often technical details are described instead of linking to a business reason for motivating the requirement.

m) Requirements open for comments[3] - In agile development, everyone who has access to the system can create issues related to requirements in the require-

Table 2. Overview of Challenges in Relation to the Solution Strategies. Indices ([1],[2],[3]) show in which case study a challenge or strategy was encountered.

	Shared Understanding			
RE	*Enable*	*Build*	*Assess*	**Solution Strategy**
General issues	a) Teams struggle to integrate RE in their agile work efficiently[1,2,3]	b) No formal event to align on customer value[1]	c) Insufficient customer feedback[1,2]	a) Tools that allow developers to take ownership of req.[1,2,3] b) Regular meetings with customer representat.[1,2]
Elicitation	d) Lack of communication with customer[1] e) Who owns customer value[1]	f) Inconsistent elicitation[2]	g) Lack of feedback on elicitation[2]	c) Ability to initiate on demand meetings with customer representatives[1,2]
Interpretation	h) Unclear why requirement is needed[2]	i) Wrong assumptions about customer value[1]	j) Unclear and volatile customer needs[2]	d) Fast feedback cycles[1,2]
Negotiation	k) Decentralized knowledge building[3]	l) Focus on technical details[1,2] m) Req. open for comments[3]	n) No time for stakeholder involvement[2]	e) Req. template includes customer value & goals[1,2] f) Define team responsibilities for different parts of req. and review updates regularly[2,3]
Documentation	o) Customer value description lost between systems[1] p) Lack of knowledge about writing requirements[1,2,3] q) No dedicated time for requirements[1,2,3]	r) Too much/not enough document.[1,2] s) Trace the requirements to all levels, (test, and code)[3]	t) Inconsistency b/c of requirements change[3]	g) Rationale must always be provided[1] h) Just enough documentation[1,2] i) Plan time for requirements updates[3] j) Educate and train the development teams[2,3] k) Tools need to be setup to support traceability [3]

ments management tool. While it is positive to include as many stakeholders as possible in discussions, without a defined process that respects the development lifecycle, this can result in an unstructured discussion and very late changes.

n) No time for stakeholder involvement[2] - Getting stakeholders' feedback after interpreting the elicited requirements is challenging since stakeholders do not have time for several meetings.

o) Customer value description lost between systems[1] - At the scale of Case 1, it is not unusual to use several different tools to manage requirements at various abstraction layers. Customer-facing units use one tool, in which they define stakeholder requirements and customer value. Development teams interact with different tools, and it is the task of the POs to refine and decompose the

stakeholder requirements from tool 1 into work items for the agile teams in tool 2. At this step, documentation about customer value is often not transferred and thus not available to the developers.

p) Lack of knowledge about writing requirements[1,2,3] - Throughout our cases, we found that those who are responsible for documenting requirements often do not have the right training. In addition, we frequently saw a lack of structure and no requirements information model. Thus, teams mix stakeholder and system requirements and are challenged with writing high-quality user stories, system requirements, and in particular quality requirements. In particular, the quality requirements might not get documented at all and teams will work on them without making them visible on the sprint dashboard.

q) No dedicated time for requirements[1,2,3] - Since agile methods focus on reducing time to market, spending time on writing formal requirements is not considered. Instead, agile teams rely on verbal requirements. Dedicated time to work on requirements should be integrated in the agile workflow, e.g. each sprint.

r) Too much/not enough documentation[1,2] - Because agile focuses on less documentation, some essential information could be missing (e.g., such as the "why" part of the requirement). Thus, in agile development, determining the right amount or sweet spot of documentation is challenging.

s) Trace the requirements to all levels, (test, and code)[3] - Due to ISO26262 and ASPICE compliance, the automotive company needs to guarantee full traceability between all requirements levels, (tests, and code). This places a big challenge on the entire company, since most teams work on something related to requirements, tests, or code and those artifacts evolve in parallel.

t) Inconsistency because of requirements change[3] - Agile methods embrace change and, consequently, teams will make changes on requirements during their work. However, it is challenging to handle sudden change requests and opinions from different team members, especially at scale. The consequence can be that teams inconsistently change related requirements, or that the scope is increased without central control. The problem is known, yet there is a lack of guidance on how to handle this in large-scale agile development to avoid expensive rework.

4.2 RQ2: How do companies aim to address these challenges?

The last column of Table 2 summarizes the answers to RQ2 on solution strategies associated with the challenges with each phase of RE in respective rows, derived from interviews, literature, or workshops and confirmed by experts in each case.

a) Tools that allow developers to take ownership of requirements[1,2,3] - In order to allow developers to take ownership of requirements, we need to find requirements tooling that integrates into the mindset and the development environment of developers to provide an efficient way of manipulating requirements. For instance, developers work closer to the code, so the requirements tool that supports commit/git is highly encouraged.

b) Regular meetings with customer representatives[1,2] - The customer-facing unit should arrange regular meetings with customers. These meetings should be well integrated in the agile workflow and mandatory for team members.

c) Ability to initiate on demand meetings with customer representatives[1,2] - There should be a setup to initiate meetings with customers whenever developers need feedback. Since access to customer representatives is a sparse and valuable resource, a strategy for such meetings should be well aligned with the organizational structure and the agile workflow.

d) Fast feedback cycles[1,2] - All teams use direct communication with stakeholders and fast feedback cycles as a baseline to get the correct interpretation. Customer insight is abstract knowledge and could be hard to write down. There is a need to arrange events where people can meet, interact, and share customer values and feedback.

e) A requirements template that includes customer value and goals[1,2] - To avoid challenges related to a lack of awareness of customer value, there should be specific fields or tracelinks that show how each requirement adds customer value. It is important to check their usage regularly.

f) Define team responsibilities for different parts of requirements and review updates/comments regularly[2,3] - In order to yield benefits from agile workflows, RE must be integrated into the agile workflow. This means that agile teams need to take responsibility of maintaining requirements and to monitor changes of requirements that are potentially related. This allows to manage requirements updates in parallel and at scale. However, responsibilities have to be carefully delegated and clearly assigned.

g) Rationale must always be provided[1] - The rationale for the requirement should mandatorily be provided by the role/person writing the requirement. Moreover, it should effectively be passed on from tool to tool.

h) Just enough documentation[1,2] - Balancing sufficient communication and documentation is crucial in agile development. We should not spend too much time documenting; however, it should have all the necessary information. Developers need clear guidelines to achieve this balance.

i) Plan time for requirements updates[3] - Teams should plan (update, change, review) the requirements in time to align with the updated scope. Such a plan should consider that updating requirements in the scope of one team may imply also requirements updates in other scopes.

j) Educate and train the development teams[2,3] - If development teams should take more responsibility of requirements, they need to be trained in RE as well as in the specifics of the overall requirements processes in their organization. A clear requirements strategy can be a good starting point to plan such training.

k) Tools need to be setup to support traceability[3] - Requirements are usually represented in different forms (e.g., textual requirements, user stories) and on different levels (e.g., system level and software level). Teams could get requirements at higher level and then derive the lower level requirements (e.g., software/technical requirements). Tracing requirements could be hard in a large complex system. Tools are needed and they should be aligned with a requirements strategy for agile workflows, i.e. allow parallel work for many teams.

4.3 RQ3: Which potential building blocks should be considered for defining a requirements strategy?

This section systematically develops the building blocks of a requirements strategy from our findings in all three cases.

In Case 1, the company was challenged to establish a shared understanding. Proposed solution strategies for specific challenges in Case 1 can be categorized as ***structural***, ***organizational***, or related to the ***work and feature flow***. For example, for the challenge *l) focus on technical details)*, a related solution strategy is *e) requirements template includes customer value and goals.* This strategy explains that, to avoid the lack of awareness about customer value, there should be specific fields related to customer value in the requirements templates. This solution shows that there is a need for improvement at the ***structural level***. In contrast, *b) no formal event to align on customer value* is a challenge related to stakeholders' roles and responsibilities that needs to be well integrated into the ***organization***. The last column in Table 2 provides a solution strategy related to this challenge as *b) regular meetings with customer representative*, which relates not only to the ***organizational perspective***, but also to the ***work and feature flow***.

In Case 2, we found the same perspectives (***structural, organizational***, as well as ***work and feature flow***) in in solution strategies for their specific challenges. As in Case 1, the solution strategy to introduce *e) requirements templates that include customer value and goals* is a ***structural*** example. In contrast, the challenge *g) lack of feedback on elicitation* can lead to misunderstandings late in an agile workflow. The solution strategy is to establish the *c) ability to initiate on-demand meetings with customer representatives.* Providing access to a sparse and valuable resources such as a customer representative relates to the ***organizational*** perspective. Another related solutions strategy, *d) fast feedback cycles*, for the challenge *j) unclear and volatile customer needs* falls into the ***work and feature flow*** perspective, by arranging events where people can meet, interact, and share customer values and feedback.

After looking deep into the concrete solution strategies in Case 1 and Case 2 (see Table 2), we found that many of these strategies were already successfully implemented in Case 3. However, the company still faced some RE challenges in agile development, allowing us to check whether the same building blocks are also applicable in Case 3. For example, the challenge *s) trace the requirements to all levels* can be addressed with the ***structural*** solution strategy *k) tools to set up traceability.* Similarly, the challenge *k) decentralized knowledge building* can be addressed by the ***organizational*** solution strategy *define team responsibilities for different parts of requirements and review updates/comments regularly.* Finally, an example of a ***work and feature flow*** related solutions strategy is to *i) plan time for requirements updates* in agile sprints to counter the challenge of having *q) no dedicated time for requirements.*

In summary, in order to address specific challenges related to enabling, building, and assessing shared understanding of requirements in agile development, specific solution strategies fall into three distinct categories: ***structure***, ***organi-***

zation, as well as ***work and feature flow***. Thus, a requirements strategy that bundles solution strategies for a concrete case should cover all three perspectives.

5 Artifact: Guidelines for Defining a Requirements Strategy

Our artifact is a set of guidelines for defining a *Requirements Strategy* as a means to define RE activities in agile development. As a design science research study, we built this artifact in parallel to answering our research questions iteratively. In particular, RQ3 provides empirical validation of the building blocks. At the time of research, the term "requirements strategy" has not been widely used. This is in contrast to, for example, "test strategy", which has quite widely been accepted to describe how testing practices can be integrated in development workflows, such as in agile ways of working. In our work, we refer to "requirements strategy" as a general strategy for including RE practices in agile methods.

Definition: Requirements Strategy. A requirements strategy is an outline that describes the requirements engineering approach in systems or software development cycles. The purpose of a requirements strategy is to support decision makers with a rational deduction from organizational, high-level objectives to actual requirements engineering activities to meet those objectives and to build a shared understanding about the problem space and requirements.

The creation and documentation of a requirements strategy should be done in a systematic way to ensure that all objectives are fully covered and understood by all stakeholders. It should also frequently be reviewed, challenged, and updated as the organization, the ways of working, and the product evolve over time. Furthermore, a requirements strategy should also aim at aligning different requirements stakeholders in terms of terminology, requirements types and abstraction levels, roles and responsibilities, traceability, planning of resources, etc.

Table 3. Building Blocks of a Requirements Strategy

	Support for shared understanding of requirements		
Perspective	**Common language**	**Knowledge flow**	**Examples**
Structural	Define reqts. levels	Define structural decomp.	Stakeholder, System, Component Requirements
	Define reqts. types	Define traceability demands	Requirements and Traceability Information Model
	Define templates		User stories include customer value and goal
Organizational	Define ownership of reqts. types	Define roles and responsibilities	Training plan per type/role; Team responsibility sheet
Work and feature flow	Define lifecycle of types	Map structure to workflow	Elicitation strategy, definition of done
		Map organization to workflow	Stakeholder map, requirements review strategy

Therefore, our contribution is a model of how requirements strategies should be described for agile development. Through providing three complementary perspectives, the proposed guidelines help to capture relevant information and provide an useful overview. Our guidelines are summarized in Table 3, including reoccurring examples and good practices abstracted from the three case studies. We propose that a requirements strategy should include the following building blocks: a structural perspective, an organizational perspective, and a work and feature flow perspective. Across these perspectives, a requirement strategy aims to support a shared understanding of requirements, in particular with respect to establishing a *common language* (i.e., enabling perspective in Table 2) and with respect to facilitating the exchange and *flow of knowledge* (i.e., building and assessing perspective in Table 2).

We suggest to start with a structural view to create a common language. A good starting point can be the artifacts in the development lifecycle model, for example the requirements information model in the Scaled-Agile Framework SAFe [16], or to define templates for user stories including customer value. Based on these initial definitions, refinements can be provided based on experience, e.g., after sprint reflections.

As a second step, we propose to make the organizational perspective explicit. Define the roles and responsibilities with respect to the definitions in the structural view. This can, for example, be done with a one-pager that describes the responsibilities of a team. Also, state who owns which part of requirements (e.g., requirements on certain subsystems) to determine specific training needs.

Finally, the work and feature flow perspective needs to be defined. A good starting point can be a lifecycle model for each critical type, which is then mapped to the intended workflow. In agile development, this can partially be provided by defining done criteria. In particular, it needs to be defined when and by whom certain information must be provided. If requirements elicitation efforts are anticipated, guidance should be given on obtaining the information from stakeholders. The workflow should be related to the roles and responsibilities as well as ownership. A stakeholder map can provide valuable information: who owns an artifact, who should be kept informed, and who needs to review it. An explicit review strategy can be very valuable, affecting not only the requirements quality but also keeping reviewers informed about recent changes.

6 Discussion and Conclusion

In this design science research study, we identified challenges related to agile requirements engineering in three case companies. Based on these three case studies, we identified solution strategies for resolving the identified challenges and derived building blocks as substantial parts of a requirements strategy. For each case we investigated a concrete requirements strategy. The individual requirements strategies have been well received by experts in each case company. Specifically, we recognize the need to enable, build, and assess shared understanding of requirements in agile development. As our experience grew, we noticed

reoccurring building blocks on what should be part of such a requirements strategy. For our design science research, we choose therefore *guidelines for creating requirements strategies* as our artifact, which we develop in parallel to investigating our knowledge questions Our results suggest that a requirements strategy should describe how requirements are structured, how work is organized, and how RE is integrated in the agile work and feature flow.

Buidling on previously published challenges and solution proposals for RE in agile development (e.g. [1, 14]), our contribution is to enable organizations to define a holistic approach to RE that integrates with their agile development. Since our guidelines shall be applicable in agile development, they do not primarily relate to explicit documentation or a dedicated requirements phase within a development lifecycle, as for example custom in waterfall processes. Instead, we rely on the theory of shared understanding to embrace RE as a knowledge management problem and give suggestions on how organizations can approach it in their agile development.

Ideally, such a strategy should be documented concisely and made available to all stakeholders. Our requirements strategy can be interpreted as an instance of situational method engineering [9] where we focus on the context of agile system development and requirements methods in particular. By this, we aim to make it easier for practitioners to integrate RE in their agile workflows. This supports its evolution through the reflection opportunities built into agile methods. We hope that our requirements strategy guidelines facilitate future research on how to manage knowledge related to requirements in agile development.

References

1. Alsaqaf, W., Daneva, M., Wieringa, R.: Quality requirements challenges in the context of large-scale distributed agile: An empirical study. Information and software technology **110**, 39–55 (2019)
2. Batsaikhan, O., Lin, Y.C.: Building a Shared Understanding of Customer Value in a Large-Scale Agile Organization: A Case Study. Master's thesis, Dept. of Computer Science and Engineering, Chalmers | University of Gothenburg (2018), https://hdl.handle.net/20.500.12380/304465
3. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software **119**, 87–108 (2016)
4. Dingsøyr, T., Moe, N.B.: Towards principles of large-scale agile development. In: International Conference on Agile Software Development. pp. 1–8. Springer (2014)
5. Dumitriu, F., Meșniță, G., Radu, L.D.: Challenges and solutions of applying large-scale agile at organizational level. Informatica Economica **23**(3), 61–71 (2019)
6. Elghariani, K., Kama, N.: Review on agile requirements engineering challenges. In: Int. Conf. on Computer and Information Sciences. pp. 507–512 (2016)
7. Glinz, M., Fricker, S.A.: On shared understanding in software engineering: an essay. Computer Science-Research and Development **30**(3), 363–376 (2015)
8. Haskouri, N.E.: Requirement Strategy in Large-Scale Agile Development: A Design Science Research. Master's thesis, Dept. of Computer Science and Engineering, Chalmers | University of Gothenburg (2021), https://gupea.ub.gu.se/bitstream/2077/69096/1/gupea_2077_69096_1.pdf

9. Henderson-Sellers, B., Ralyté, J.: Situational method engineering: state-of-the-art review. Journal of Universal Computer Science (2010)
10. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly **28**(1), 75–105 (2004)
11. Hoda, R., Noble, J., Marshall, S.: How much is just enough? some documentation patterns on agile projects. In: Proceedings of the 15th European Conference on Pattern Languages of Programs. pp. 1–13 (2010)
12. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Computers in human behavior **51**, 915–929 (2015)
13. Jorgensen, M.: Relationships between project size, agile practices, and successful software development: results and analysis. IEEE Software **36**(2), 39–43 (2019)
14. Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., de Oliveira Neto, F.G.: Requirements engineering challenges and practices in large-scale agile system development. Journal of Systems and Software **172**, 110851 (2021)
15. Kasauli, R., Liebel, G., Knauss, E., Gopakumar, S., Kanagwa, B.: Requirements engineering challenges in large-scale agile system development. In: International Requirements Engineering Conference (RE). pp. 352–361 (2017)
16. Knaster, R., Leffingwell, D.: SAFe 4.0 distilled: applying the Scaled Agile Framework for lean software and systems engineering (2017)
17. Lagerberg, L., Skude, T., Emanuelsson, P., Sandahl, K., Ståhl, D.: 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. pp. 348–356 (2013)
18. Larman, C.: Practices for scaling lean & Agile development: large, multisite, and offshore product development with large-scale scrum. Pearson Education (2010)
19. Maxwell, J.: Understanding and validity in qualitative research. Harvard educational review **62**(3), 279–301 (1992)
20. Méndez, E.M., Pérez, M.A., Mendoza, L.E.: Improving software test strategy with a method to specify test cases (mstc). In: ICEIS (1). pp. 159–164 (2008)
21. Meyer, B.: The ugly, the hype and the good: an assessment of the agile approach. In: Agile! pp. 149–154. Springer (2014)
22. Paetsch, F., Eberlein, A., Maurer, F.: Requirements Engineering and Agile Software Development. In: International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 308–313. IEEE (2003)
23. Palinkas, L.A., Horwitz, S.M., Green, C.A., Wisdom, J.P., Duan, N., Hoagwood, K.: Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. Administration and Policy in Mental Health and Mental Health Services Research **42**(5), 533–544 (2015)
24. Ramesh, B., Cao, L., Baskerville, R.: Agile requirements engineering practices and challenges: an empirical study. Information Systems Journal **20**(5), 449–480 (2010)
25. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering **14**(2), 131–164 (2009)
26. Saldaña, J.: The coding manual for qualitative researchers. Sage, 3rd edn. (2015)
27. Serrador, P., Pinto, J.K.: Does Agile work? A quantitative analysis of agile project success. International Journal of Project Management **33**(5), 1040–1051 (2015)
28. Vaishnavi, V., Kuechler, W.: Design Science Research Methods and Patterns: Innovating Information and Communication Technology. Taylor & Francis (2007)
29. Wieringa, R.J.: Design science as nested problem solving. In: 4th Intl. Conf. on Design Science Research in Inf. Sys. and Techn. pp. 1–12. Philadelphia (2009)