

How Artifacts Support and Impede Requirements Communication

Olga Liskin^(✉)

Leibniz University Hannover, Hannover, Germany
olga.liskin@inf.uni-hannover.de

Abstract. [Context & motivation] Requirements artifacts, like specifications, diagrams, or user stories, are often used to support various activities related to requirements. How well an artifact can support a specific activity depends on the artifact's nature. For example, a plain text document can be adequate to provide contextual information, but is not well suited in terms of documenting changes. [Questions / problem] We wanted to understand how practitioners in various roles use requirements artifacts, how they manage to work with multiple artifacts at a time, and whether they use current practices for linking related artifacts. [Principal ideas / results] We have conducted an interview study with 21 practitioners from 6 companies. The interviews indicate that often a variety of artifact types is needed to successfully conduct a project. At the same time, using multiple artifacts causes problems like manual translation effort and inconsistencies. Mapping mechanisms that explicitly relate different artifacts are needed. However, existing methods are often not used. We investigate why these methods challenge developers in practice. [Contribution] We show challenges and chances of requirements artifacts. Our findings are grounded on true experiences from the industry. These experiences can support software developers in planning and improving their processes with regard to better requirements communication and researchers in making mapping methods more applicable in industry.

Keywords: Requirements artifacts · Requirements communication · User stories

1 Introduction

When Cockburn described the temperature of different communication channels [1], the hottest communication channel was not talking face-to-face, but *talking face-to-face at a whiteboard*. The reason is that writing down things helps clarify them. This is only one of many important powers of requirements artifacts. Moreover, they can help documenting information for later look-up, enable splitting requirements into explicit individual items for efficient management, and much more.

However, not all artifacts types are equally suited for all activities in software and requirements engineering. Artifacts like specifications, user stories, or GUI mockups foreground certain aspects of the set of requirements and hide others. This influences, for example, which information gets concretized or how well relations come into

view. Moreover, artifacts are used by many different persons, with various roles and different requests based on their individual work throughout the project. More and more companies strive to employ an iterative approach in the day-to-day development, requiring the appropriate artifacts. At the same time, teams following an agile approach realize with an increasing frequency that user stories and a backlog are not always enough. Especially in larger projects, having additional artifacts to integrate overall information, allow early general decisions, or meet regulatory needs, pans out.

Often, there is not one perfect kind of artifact that will serve the needs of all participants so that the project needs to deploy a whole variety of different artifacts. This, in turn, carries the risk of inconsistencies or inefficiencies emerging from the dependencies between multiple artifacts. Successful integration of requirements artifacts is an important matter in requirements engineering. In order to advance in this field, more research is needed to understand the challenges and chances of requirements representations. With the presented study, we contribute to improving this understanding.

2 Related Work

Several empirical studies have been conducted to study *requirements communication*. Bjarnason et al. [2], [3] and Abelein and Paech [4] have conducted interview studies on requirements communication in practice. They report on communication gaps in requirements engineering in general and on gaps in user-developer communication, respectively. Marczak et al. [5] conducted a field study where they regarded the communication network of developers working on related requirements. Knauss et al. [6] have developed a systematic scheme of requirements clarification patterns and report on a case study in which they investigated the patterns occurring in practice. Our paper extends this work on investigating requirements communication in practice by analyzing the facet of communication aided by artifacts.

Research on *requirements artifacts* addresses how they can be used to support software engineering activities and communication. Kumar and Wallace [7] describe communication patterns – including artifact facilitated discussion – and their outcome. Fernandez and Penzenstadler [8] research artifact based RE methods in contrast to activity based ones. They have designed and evaluated various artifact based RE models and then combined them into a domain-independent approach (AMDIRE). Gross and Doerr [9] analyze how artifacts and their contents should be constituted in order to support the needs of different roles in software engineering. Sharp et al. [10] use the distributed cognition approach to investigate the role of physical artifacts on communication within agile teams. Gallardo-Valencia et al. [11] explore whether agile requirements artifacts are sufficient for development and show that adding use cases can be beneficial.

The *mapping of requirements artifacts* has been repeatedly discussed in literature. Patton [12] describes techniques for (implicitly) mapping different story artifacts to each other. Imaz and Benyon [13] present an approach for enhancing relations between user stories and use cases. Antonino et al. [14] suggest a method for lightweight linking of requirements and development artifacts, which includes the

mapping between user stories and individual requirements. Further research focuses on mapping requirements to more abstract items that are related to them. Abelein and Paech [15] describe the mapping of requirements to decisions. Rashid et al. [16] analyze how early aspects can be brought into requirements engineering and the according artifacts. Gotel et al. [17] describe how, in general, visualization of requirements and their connections could be used to improve software development. Creighton et al. [18] use sequences of video clips to visualize requirements in a user understandable way and then map these to more formal specification elements such as use case models and sequence diagrams. We investigate which of the available methods are actually applied in current practice and how they are working out. Our analysis contributes new knowledge to the field of mapping requirements artifacts by pointing out experiences and problematic areas.

The field of *tracing* provides many techniques to map requirements artifacts to subsequent project artifacts, like design artifacts, code, and tests. Boullion et al. [19] present scenarios in which requirements traceability is relevant in practice. Ben Charada et al. [20] analyze code changes and then employ tracing tools to automatically identify outdated requirements. Research on improving tracing has many facets. For example, Anderson and Sherba [21] enhance automated management of traceability links by using open hypermedia techniques. Huffman-Hayes et al. [22] use information retrieval techniques to improve requirements tracing. Tracing mainly focuses on links between requirements artifacts and subsequent development artifacts, like architectural components, code, and tests. In contrast to that, we focus on enhancing links among requirements artifacts of the same or different kinds.

3 Study Design

Our objective is to study the usage of different artifacts in practice. We did this in two steps. First, we examined artifacts themselves and their support of development tasks. Then, we looked at the work with multiple types of artifacts at a time.

In the first phase of this study, our goal was to get an overview of how requirements artifacts are used in practice. We wanted to understand the values and impediments of different artifact types and the consequences of working with multiple artifacts at a time. The first two research questions guided this phase:

RQ1: What are the values and impediments practitioners see in different requirements artifacts? Throughout a project, different roles come into contact with requirements and perform different activities based on these. The requirements' representations can be more or less suited to support these activities. In conjunction with this research question, we create an overview of relevant activities and show which artifacts can or cannot support these.

RQ2: Which problems do practitioners face when using multiple different requirements artifacts within a project? Our study shows that oftentimes multiple different artifacts are used in order to support different activities. Often, artifacts have overlapping content, which can lead to inconsistencies. We want to find out which problems practitioners have actually experienced and consider relevant.

When working with multiple artifacts, many problems could be diminished if related parts within artifacts were explicitly mapped to each other. In the second phase, we focused on whether mapping methods are used in practice and what reasons prevent developers from implementing mapping methods. While still seeking insights and validation for the first two questions, we added the following two research questions:

RQ3: Which methods are used in industry to link multiple different requirements artifacts? Linking from one artifact to another one – for example simply by referring to the other artifact’s ID – can help identify related content. This can be used to avoid inconsistencies when documenting changes. A more sophisticated method is to use clickable links that bring up additional content from a related artifact right away. Moreover, two artifacts could directly operate on the same content – serving as two views to the same content. It is not well known which of these methods are actually known or even used in industry. We want to close this gap with this research question.

RQ4: What challenges arise when linking multiple requirements artifacts? Often, only simple methods for artifact mapping are used. At the same time, developers find it challenging to work with multiple requirements artifacts. We want to find out what prevents practitioners from using more sophisticated requirements mapping methods. We want to know whether it is the creation of links that confronts them with problems or whether they see too little value in using links afterwards.

Table 1. Interview participants

Company Type	Company	Size	ID	Role
IT Service Provider	C1	500 - 1000	I1	Project Manager
	C2	1000 - 1500	I2	PO & Project Manager
In-House IT	C3	100 - 500	I3	Developer
			I4	Project Manager
			I5	Customer Rep.
			I6	Application Owner
			I7	Project Manager
			I8	Customer Rep.
			I9	Architect
			I10	Customer Rep.
	C4	1000 - 1500	I11	Customer Rep.
			I12	Project Manager
			I13	Developer
I14			Process Engineer	
			I15	Team Leader
			I16	Developer
			I17	Developer
			I18	Developer
Standard Software Producer	C5	<100	I19	Team Leader
	C6	<100	I20	Team Leader
			I21	Team Leader

Data Gathering and Analysis: We interviewed 21 practitioners from 6 companies. Table 1 shows an overview of the companies, projects, and roles of the participants. The company type influences the relation to the customer and therewith also requirements communication. Therefore, we interviewed persons from different company contexts. To ensure coverage of a wide range of requirements related activities, we interviewed people in different roles. We used semi-structured interviews, which mostly lasted about 75 minutes. We recorded and transcribed the interviews, and then coded and categorized the statements.

4 Results

4.1 Classification of Requirements Artifacts

The variety of requirements artifacts used in software projects is very high. Our interviewees mentioned mainly three types of artifacts: containers, individual elements, and solution models. In the course of the interviews, we found further characteristics of the artifacts that influenced their handling. We subdivide our artifact categories accordingly to accommodate these differences in the further analyses. Figure 1 depicts the categories of artifacts we found and the concrete artifacts that the practitioners reported to use.

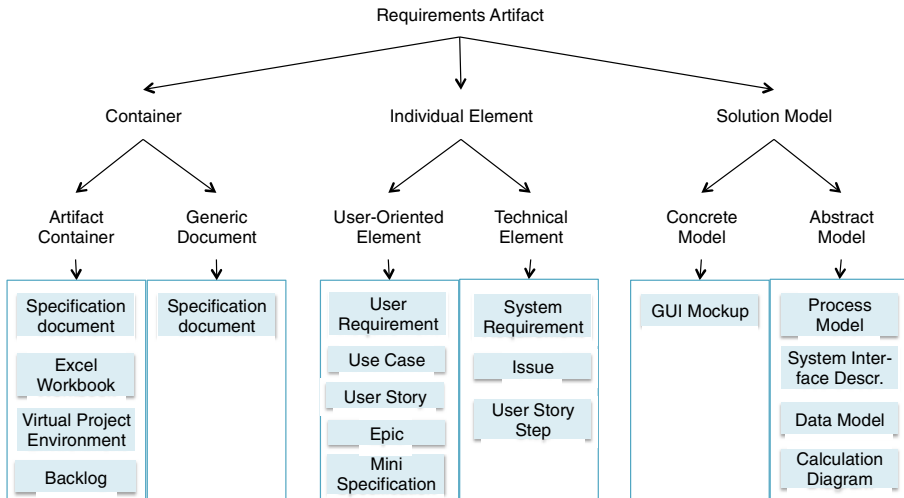


Fig. 1. Classification of used artifacts

Containers are characterized by their value to hold everything together in one place. We found that it makes a difference whether a container consists only of other artifacts (individual elements and solution models) or enables to include generic content. Generic elements make it easier to enter any important information quickly but at the same time carry the risk of the information being unsuitable for later tasks.

Virtual project environments and backlogs are artifact containers, while specification documents and excel workbooks can come in both forms. Generally, text documents are very generic and allow information to be entered in a variety of ways. Blocks of plain text can include one or several requirements at a time and mix them with goals, background information, or relevant policies. At the other end of the spectrum, documents can have a very strong formal structure, only consisting of elements that have a defined type and ID. Excel workbooks also provide very generic elements and could potentially be used to enter generic information. However, we have only seen it being used as an artifact container with each content element being a defined artifact like a user story, GUI mockup, or process model.

For *individual elements*, we found the most important aspect to be whether they are user oriented or not. It determines how well the users can contribute to the creation and assessment of that artifact. An element is considered user-oriented if it is clear to the user what will change with the completion of that element.

Elements referred to as *user stories* occurred in both forms, as user-oriented and technical elements. While user stories describe what users do as part of their job, technically oriented stories often describe smaller steps on the way towards a user story. A user can see, test, and comment them, but not understand them by herself (without translation by developers) because they do not clearly relate to the actual business tasks in the user's daily work. To distinguish such technical stories, we refer to them as *user story steps*.

Mini specifications are smaller specifications that describe just a part of a system instead of the whole system. They are used to gradually elaborate requirements details in iterative development or enhancement projects. The interviewees described mini specifications as easy to handle. Challenges emerge because a system possesses many such specifications, which are in addition interrelated. Therefore we classified them as elements instead of containers.

Solution models illustrate aspects of the future solution in the form of formal or graphical models. Concrete models directly relate to concrete situations a system's users experience. In contrast, abstract models show more universal generalizations of different concrete manifestations. For example, process models often show general abstract workflows spanning multiple roles and phases, while the step-by-step description of one specific partial path within this workflow is a concrete instantiation for a specific person. Our interviewees stated that they had experienced some user representatives to struggle with thinking in an abstract way. This influenced the models' effectiveness for aiding communication.

4.2 Values and Impediments of Different Requirements Artifacts (RQ1)


The main values of a requirements artifact lie in its ability to support a software engineering activity. We asked our participants which activities related to requirements they perform in their role throughout a project and which artifacts, if any, they use for them. Then, we inquired whether they had experienced an artifact to be supportive or troublesome for that activity.


Table 2 shows the activities that had come up in the interviews. In each cell, it displays how many interviewees mentioned that an artifact was supportive or troublesome for that activity. To better understand what led to the interviewees' assessments we also asked them about the relevant properties that made an artifact supportive or hindering. These properties are collected in Table 3.

One of the most strongly discussed activities was the *clarification of requirements details*. Many of the interviewees were struggling to find the right form of artifact for communication with the users. Although there are many templates and diagrams that are suggested for this activity, the interviewees reported that their customers could not work with most of them. It was considered helpful to use very concrete model artifacts for this communication. Users and user representatives were found to be best able to assess and contribute to requirements representations in the form of GUI-mockups or very concrete usage scenarios. Abstract models like flow charts of a process were considered helpful for some customers but dangerous for others. Customers who did not fully understand the notion or the abstract contents of such a model (and due to time pressure did not have time to address such problems), were

Table 2. Activities performed by interviewees. Numbers indicate number of interviewees who mentioned that artifact was suited for activity or could lead to problems, respectively.

ID	Activity	Artifact Container	Generic Document	User-Oriented Element	Technical Element	Abstract Model	Concrete Model	Verbal Comm., Email
A1	Understand & document overarching aspects	1	7 1	2 4	2 5		2	2
A2	General project planning		5 1				1	
A3	Collect requirements	2	6 1	7	1 1	1 2	4	1
A4	Reveal contradictions, inconsistencies	1	1 1	3 1		2	2	
A5	Prioritize Requirements		1 1	1	3 2			4
A6	Plan & control iteration		1 3	4 2	9 1	1		1 1
A7	Clarify requirements	1	1 2	7 2	2 2	4 6	7	2
A8	clarify requirements beyond analysis phase	1	1		1			5
A9	Manage requirements		3 4	4 4	4 2	1 1	1	4 6
A10	Review & report release completion		2 1		1	1	2	6 1
A11	Formal contract issues		4	1	1			1

 # Int. mentioned that artifact was suited for activity

 # Int. mentioned potential problems with artifact

sometimes found to check it too superficially and to accept a presented model too quickly. This resulted in wrong assumptions and a false sense of security for the developers.

Interestingly, in most cases where user stories were employed, they were not well understandable for the users. Often, they were sliced in a way that they were not relating to the user’s job tasks, but were too much on a system level. The team had drifted off to *user story steps* instead of user stories and lost some potential on involving users into the development and, most importantly, prioritization. This was partly compensated by communication, however at the price of additional translation effort (for explanation of user story steps).

Often, details also need to be *clarified beyond the analysis phase*, i.e. during implementation or testing. In these situations, very quick forms of communication and documentation, like emails or phone calls, were considered well suited. However this made later activities like *reconstructing a requirement’s history* much harder or even impossible. Similarly, some interviewees stated that this activity was hard when working with a specification document. Besides the difficulties of finding a specific requirement within a document, relevant information like a requirement’s creator or updater, the release it was shipped with, or the rationale behind it often simply were not recorded per requirement when using a specification document.

Table 3. Positive and negative properties of artifacts referred to by interviewees

Artifact	Positive Properties	# Int.	Negative Properties	# Int.
Container	collects everything in one central place	6		
	acceptable	4		
Artifact Container			new tool that needs to be learned	5
Generic Document	universal (everybody understands office documents)	1	difficult to search	6
	permits many ways to write down things	2	contents are too vague, generic	4
			contents are too detailed	4
Individual Element	divides big items into small manageable pieces	9	relations to overarching elements unclear	5
	good granularity for clarifying details	4	relations to other elements unclear	3
	easy to attach attributes (like author, release)	4	difficult to understand structure for persons who do not work with the elements regularly	3
User-Oriented Element	understandable for users	3	too coarse for development	2
Technical Element	good granularity for checking whether all necessary information is available	3	not understandable for users	6
			exact scope not clear	2
Solution Model	little room for interpretation	4	limited expressive power	1
	good for finding information quickly	4		
Concrete Model	best understood by customers	8		
Abstract Model			not understood by all customers	4

Planning and controlling implementation was reported to be well supported by elementary artifacts. Dividing the specification into elementary artifacts allows to attach additional information per artifact. Further, through the divide and conquer principle, it makes each element more tangible and manageable. This mostly aligns itself with what is known from literature on agile methods.

However, we also found activities that were considered problematic when only working with elements like user stories. The reported problems mostly relate to situations, which require more of an overall view on requirements. First, some interviewees mentioned that it was important to *understand and document the vision* of a project. They reported that sometimes, a requirement itself looked fine – for example, it was clear, self-contained, and had acceptance tests attached to it. However it was not making sense on a more general level because it was not solving the users' actual needs. To identify such situations, developers needed to understand the context of a story, like related stories or goals, which they not always were able to establish from their requirements artifacts.

A second important activity was to *develop tests that go beyond the scope of a single element* like a user story. For example, developers need to write automated tests that cover the collaboration of multiple stories. Similarly, acceptance testing for a release required additional information about general user goals. When just working with what was written on elementary requirements, they missed some connecting test cases. A third mentioned aspect was the *inclusion of strategic goals*. Strategic goals were mainly found to influence prioritization of work or introduce new requirements.

It was important to document such overarching information so that it does not get overlooked in the later project phases. However, it was considered difficult to document them just with elementary artifacts like user stories or use cases. Here, specification documents were considered helpful because of the high freedom they gave the author to note information. Sometimes, also slides with concrete or abstract models were kept as a reference for the overall vision and goals.

4.3 Problems Practitioners Face When Using Multiple Different Requirements Artifacts Within a Project (RQ2)

As the results in RQ1 indicate, many participants work with multiple different artifacts in order to be able to better support different activities. Another mentioned benefit was that displaying the same ideas in two different ways allowed the participants to better check whether they had interpreted requirements correctly. However, using multiple artifacts comes at the price of additional effort for creating, maintaining, translating, and preventing artifacts from inconsistencies. Table 4 shows the problems that were mentioned by the interviewees, as well as their reinforcers and effects. # Int. depicts the number of interviewees who mentioned an item. Some participants mentioned multiple items within a topic. Therefore, # Int. in the lines *reinforcers, problems and effects* displays the total number of interviewees who had talked about the according topic.

Table 4. Benefits and problems observed by interviewees when working with multiple artifacts

		# Int.
Reinforcers		5
R1	Some contents overlap, others are disjoint	2
R2	Non-trivial relations between (parts of) artifacts	5
Problems		15
Pr1	(R1 ->) Duplication of effort for creating multiple artifacts	3
Pr2	(R2 ->) Uncertainty about completeness of translation	6
Pr3	(R1 ->) Changes must be documented in multiple places	4
Pr4	(R1, R2 ->) Inconsistencies	3
Pr5	(R1, R2 ->) More difficult to find relevant information in multiple places	4
Effects		5
E1	Higher costs for performing tasks or preventing problems	2
E2	Higher costs when problems occur (mistakes, misunderstandings)	1
E3	Decreased trust in up-to-dateness of artifacts	2

We found that more than 70% of our interviewees had experienced problems when working with a variety of artifacts. Besides the extra effort for documenting and finding information in multiple places (Pr1, Pr3, Pr5), they had struggled with inconsistencies (Pr4). Further, they reported that it was difficult to check whether all (relevant) elements from one artifact type had been recorded in the other one (Pr2). Problems led to extra effort for preventing them (E1), but could not always be mitigated. If an inconsistency or other information was overlooked, misunderstandings or wrong assumptions occurred, which in turn potentially led to higher costs through mistakes (E2). Another reported effect was that people very quickly lost trust in a document (E3) – and hence stopped using it – if they repeatedly had found the contents to be not up-to-date or inconsistent.

In practice, two circumstances make working with multiple requirements artifacts particularly challenging. The artifacts do not just describe disjoint information, but various aspects of the same requirements (R1). Therefore, some – but also not all – information is contained in multiple artifacts. Further, artifacts are not always in a simple hierarchical one-to-many relationship, like when dividing a story into tasks (R2). For example, a process model and a set of user stories can have complex relations with the process model depicting the interaction of multiple stories, while at the same time, a subset of activities illustrating one user story.

4.4 Methods Used in Industry to Link Multiple Different Artifacts (RQ3)

We found different ways to map requirements artifacts that are used in industry. We have classified them into four kinds of mapping. Table 5 shows the kinds of mappings we found.

Table 5. Categories of mapping methods used in practice

ID	Mapping Method	# Int.
M1	Manual textual reference	6
M2	Attachment	10
M3	Link	3
M4	Generated artifact	1

The simplest way was to manually reference a related element by mentioning its ID in a textual way. This technique was mostly used to refer to parts of a specification document. For example, a change request contained the specification's chapter with the original requirements. Similarly, a developer who had translated a specification into User Stories, added the chapter with the original requirements to the stories. She reported that the specification's structure was changing, however, which rendered some of the references obsolete. In another project, the participants simply textually referred from User Stories to an overall GUI mockup and an overall flow chart.

Another common technique we found was to attach an element to another element. When working with the container element, the attachment can be directly accessed which makes it easy to obtain detailed information. However, the attached element only exists within the container element and cannot be accessed otherwise. Therefore, this technique is beneficial for hierarchical structures. Often, tools can create this kind of mapping automatically during the attachment process.

Linking two elements that exist by themselves was also used in the discussed projects, but only in very few cases. The direct link between the two elements allows to directly open one element from a referencing element. In one reported case, developers were managing their work iteratively, but based on technical tasks. They added a special type of artifact to represent user goals and linked the goal with all technical tasks that were necessary to fulfill it. This allowed the customers to see how the project was progressing on their goals, while the developers were still able to structure their work based on dependencies between development tasks.

Another mentioned technique is generating or constructing one artifact from multiple other artifacts. This technique is mostly used to create specification documents from requirements elements and models. It avoids the duplication of content by keeping the content in one place and just displaying it in another place. In order to use this technique, special attention to the arrangement of the elements must be paid.

4.5 Challenges of Linking Multiple Requirements Artifacts (RQ4)

We found indications for both, the effort for the creation of artifact links being perceived as too high as well as a too low perceived value. Table 6 presents challenges that interviewees mentioned to encumber or even prevent them from linking artifacts.

Table 6. Challenges of linking multiple requirements artifacts

ID	Challenge	# Int.
C1	Time pressure	5
C2	Interruption of other tasks	4
C3	Requires clear guidelines	1
C4	Difficult if requirements are not isolated from each other	2
C5	Manual links can become obsolete	2

When asked why they had not established an explicit link between particular artifacts, the interviewees' most common answer was that they had no time. We tried to find out more precisely, what it was that drove them not to want to spend time with such a task. One mentioned problem was that often the persons worked with related artifacts when they were in the middle of a different task. They were working on code or other artifacts when they had searched for additional information in artifacts. In this situation, they did not want to interrupt that task to create artifact links.

One interviewee, who had worked with links before, mentioned that clear guidelines are needed in order to establish a good linking structure. For example, it must be specified that each story has to be linked to a user goal. Thinking through such guidelines is an additional barrier that prevents practitioners from using linking structures.

Further, linking was considered challenging when the parts to link were not isolated. For example, if requirements are just contained in a block of plain text, or if a model element cannot be addressed isolated from the whole model, it is more challenging and imprecise to denote related elements. Whether it is easy to link parts of a model to other artifacts, mainly depends on the tooling used for creating the models. This is especially a problem in enhancement projects, where the developers have to build upon existing documentation.

Another mentioned demotivating factor was the high chance of breaking links on changes. One team had tried to maintain a set of links from user stories to detail chapters in a specification document. However, the chapters changed from time to time – rendering the links useless – so that the team ultimately gave up.

5 Discussion

Handling multiple requirements artifacts is challenging. Our results indicate that only one kind of requirements artifacts often is not enough for a software project. In most projects, multiple artifacts are used to support different requirements communication activities. However, our results also indicate that requirements artifacts often are not well integrated. Relations and dependencies between artifacts are not visible. If developers or customers do not keep them in mind or spend extra time to search artifacts, they miss important information.

Requirements communication with customers is not supported well. We see a strong need for more work on supporting customers or business analysts in communicating requirements. Customers are forced to create or accept artifacts in formats suited for

developers. They cannot understand most of these languages and time pressure does not leave them the time anymore to learn them. Instead, they should communicate requirements in a form that is tangible for them and developers should be able to integrate these forms into later work items. User stories answer the purpose of making it easier for customers to communicate requirements and even participate in guiding development. However, often they are not used for this purpose. We have seen user stories being used as a means for developers to split work items and make work more manageable, having to be translated permanently for customer communication. In order to support the needs of both, developers and users, *stories at different levels of granularity are needed*. As suggested by interviewees, it makes sense to work with business user stories and technical user stories in combination.

Mapping requirements artifacts has a high potential. We have seen many problems that could be mitigated if requirements artifacts were used. A lot of effort could be saved for manually checking items for consistency, or proving that all items have been translated. In addition, several requirements engineering activities could also be improved. For example, developers could be warned about dependencies before they implement a story card or when user oriented elements are changed. If abstract models could be linked with concrete models or requirements elements, the requirements engineer could also use complex abstract models for customer communication. Parts of the abstract model could be directly translated to concrete models, providing an understandable view on the details. In the interviews, we got the impression that many of the potential benefits of mapping seem vague to most practitioners.

Good lightweight requirements communication is working well. Many interviewees reported that they were solving many tasks through direct communication. As Table 2 illustrates, many of the discussed activities were aided by direct communication. Many interviewees stated that they had intensified verbal communication between different roles – mostly through weekly or biweekly meetings – only in the last few years. They reported to have experienced many improvements since the introduction of such meetings. This is a good advancement. Lacking communication between the customer and development sites has been a problem for several years. However, we also saw new problems come up in the interviews when the reliance on verbal communication was too high. Interviewees reported that sometimes, the only way to detect a dependency or misunderstanding was when one particular person - who often was the only one having the necessary knowledge - brought it up in the according planning meeting. This strategy has worked out in many cases but is quite incidental. The described situations raise questions, like whether more means are needed to improve knowledge distribution in teams, and how such communication-reliant approaches can be scaled.

6 Threats to Validity

This section discusses threats to validity based on Runeson and Höst [23].

A threat to *construct validity* arises because the information provided by the participants is interpreted by the researcher to form categories. This categorization is not unique. The interview character of the study implied that not all questions were posed explicitly. By using interviews (in contrast to surveys, for instance) we were able to counteract misunderstandings with the participants.

Various aspects could influence the *internal validity* of our results. The types of a project and its customers greatly influence requirements communication and therefore the success of requirements engineering activities and utilized artifacts. To mitigate this threat, we interviewed practitioners from different types of companies and different projects. The participants were self-selected, i.e. they knew in advance that the interview would cover requirements artifacts and had agreed to participate in the study. We cannot rule out the possibility that they had a higher interest in requirements engineering practices and the usage of artifacts than the majority of software engineers. The methods they apply in requirements engineering, the artifacts they use and their perceptions of the benefits of those artifacts could be influenced by their general interest in RE.

The number of participants could influence *external validity*. We have interviewed only 21 practitioners, so it is likely that we have not covered all situations in requirements communication. Also, the participants are all from German companies. However, since we have spoken to people within different company settings, different projects, and different roles, the variety of covered perspectives is very high. In addition, we have reached a state in which answers were repetitive to insights from preceding interviews and further interviews lead to a diminishing number of results for our research questions (similar to theoretical saturation in Grounded Theory [24]).

Reliability is affected by the number of participants, which is too low to claim statistical significance, and the fact that the interviews and their analysis were conducted by one person. A different researcher could convey the questions and also interpret the answers differently.

We have used a qualitative research approach, which reflects subjective opinions and experiences of the participants. These cannot be generalized. Despite this and the above limitations, we believe that our results have a value for researchers and also for practitioners. They provide insights into the practice, increase the understanding of the employment of requirements artifacts, and indicate possible challenges.

7 Conclusions

We have interviewed 21 practitioners about their handling of requirements artifacts and report on their experiences, named challenges, and advances in using mapping techniques. We have found that various artifacts are needed. Developers require detailed items of fine granularity but also need to keep an eye on overarching aspects like the product vision and goals. Customers need very concrete artifacts to express their expectations. Project managers need a way to see the connections to the total amount of upcoming work.

The employment of multiple different artifacts imposes challenges like scattering of information, incomplete translations, or inconsistencies between artifacts. For these reasons, methods for mapping and linking requirements artifacts should be common proficiencies in requirements engineering. However, we have seen it rarely employed in practice. Most participants stated that they found explicit mapping or linking of artifacts too costly in their project context. Indeed, mapping is not necessary in all situations. However, if the methods and tools could be improved to better facilitate artifact mapping, this would assist in many software projects.

With our results, practitioners can get an increased understanding of an artifact's utility for different activities, get an overview on mapping techniques and understand what might prevent project members from using them. Researchers gain insights into the handling of requirements artifacts in practice and into challenges that need to be solved as well as investigated further.

In the future, we would like to work on improving facilitation of requirements mapping by building on the insights from this study. Further, we have seen that artifact mapping is not crucial in all kinds of projects. It would be interesting to investigate which project aspects constitute a need for mapping. Another interesting aspect is how to determine – especially in early project phases – which linking techniques and also which artifacts will be helpful in the project

Acknowledgments. I would like to thank all interview participants for their time, all the valuable insights, and the exceedingly interesting conversations.

References

1. Cockburn, A.: *Agile Software Development*. Addison Wesley (2002)
2. Bjarnason, E., Wnuk, K., Regnell, B.: Requirements are slipping through the gaps - a case study on causes & effects of communication gaps in large-scale software development. In: *Requirements Engineering Conference (RE)* (2011)
3. Bjarnason, E., Wnuk, K., Regnell, B.: Are you biting off more than you can chew? a case study on causes and effects of over-scoping in large-scale software engineering. *Information and Software Technology* **54**(10), 1107–1124 (2012)
4. Abelein, U., Paech, B.: State of practice of user-developer communication in large-scale IT projects. In: Salinesi, C., van de Weerd, I. (eds.) *REFSQ 2014*. LNCS, vol. 8396, pp. 95–111. Springer, Heidelberg (2014)
5. Marczak, S., Damian, D., Stege, U., Schroter, A.: Information brokers in requirement-dependency social networks. In: *Requirements Engineering Conference (RE)* (2008)
6. Knauss, E., Damian, D., Cleland-Huang, J., Helms, R.: Patterns of continuous requirements clarification. *Requirements Engineering Journal* (2014)
7. Kumar, S., Wallace, C.: A tale of two projects: a pattern based comparison of communication strategies in student software development. In: *Frontiers in Education Conference*. IEEE (2013)
8. Fernandez, D.M., Penzenstadler, B.: Artefact-based requirements engineering: the AMDiRE approach. *Requirements Engineering Journal* (2014)
9. Gross, A., Doerr, J.: What you need is what you get!: the vision of view-based requirements specifications. In: *Requirements Engineering Conference (RE)* (2012)

10. Sharp, H., Robinson, H., Petre, M.: The role of physical artefacts in agile software development: Two complementary perspectives. *Interacting with Computers* **21**(12), 108–116 (2009)
11. Gallardo-Valencia, R.E., Olivera, V., Sim, S.E.: Are use cases beneficial for developers using agile requirements?. In: Fifth International Workshop on Comparative Evaluation in Requirements Engineering (CERE) (2007)
12. Patton, J.: *User Story Mapping*. O'Reilly Media (2014)
13. Imaz, M., Benyon, D.: How stories capture interaction. In: *INTERACT 1999*, pp. 321–328. IOS Press (1999)
14. Antonino, P.O., Keuler, T., Germann, N., Cronauer, B.: A non-invasive approach to trace architecture design, requirements specification and agile artifacts. In: *23rd Australian Software Engineering Conference (ASWEC)*, pp. 220–229 (2014)
15. Abelein, U., Paech, B.: A proposal for enhancing user-developer communication in large IT projects. In: *5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 1–3 (2012)
16. Rashid, A., Sawyer, P., Moreira, A., Araujo, J.: Early aspects: a model for aspect-oriented requirements engineering. In: *Requirements Engineering Conference (RE)* (2002)
17. Gotel, O.C.Z., Marchese, F.T., Morris, S.J.: On requirements visualization. In: *2nd International Workshop on Requirements Engineering Visualization (REV)* (2007)
18. Creighton, O., Ott, M., Bruegge, B.: Software cinema – video-based requirements engineering. In: *Requirements Engineering Conference (RE)* (2006)
19. Bouillon, E., Mäder, P., Philippow, I.: A survey on usage scenarios for requirements traceability in practice. In: Doerr, J., Opdahl, A.L. (eds.) *REFSQ 2013*. LNCS, vol. 7830, pp. 158–173. Springer, Heidelberg (2013)
20. Ben Charrada, E., Koziolok, A., Glinz, M.: Identifying outdated requirements based on source code changes. In: *Requirements Engineering Conference (RE)* (2012)
21. Anderson, K.M., Sherba, S.A.: Using open hypermedia to support information integration. In: Reich, S., Tzagarakis, M.M., De Bra, P.M.E. (eds.) *AH-WS 2001, SC 2001, and OHS 2001*. LNCS, vol. 2266, pp. 8–16. Springer, Heidelberg (2002)
22. Hayes, J.H., Dekhtyar, A., Osborne, J.: Improving requirements tracing via information retrieval. In: *Requirements Engineering Conference (RE)* (2003)
23. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* **14**(2), 131–164 (2009)
24. Glaser, B.G., Strauss, A.L.: *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Observations (Chicago, Ill.). Aldine de Gruyter (1967)