The Journal of Systems and Software 109 (2015) 62–77

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Assessing requirements engineering and software test alignment—Five case studies

Michael Unterkalmsteiner*, Tony Gorschek, Robert Feldt, Eriks Klotins

Department of Software Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

ARTICLE INFO

Article history:
Received 22 December 2014
Revised 26 May 2015
Accepted 8 July 2015
Available online 17 July 2015

Keywords:
Requirements engineering
Software testing
Coordination

ABSTRACT

The development of large, software-intensive systems is a complex undertaking that we generally tackle by a divide and conquer strategy. Companies thereby face the challenge of coordinating individual aspects of software development, in particular between requirements engineering (RE) and software testing (ST). A lack of REST alignment can not only lead to wasted effort but also to defective software. However, before a company can improve the mechanisms of coordination they need to be understood first. With REST-bench we aim at providing an assessment tool that illustrates the coordination in software development projects and identify concrete improvement opportunities. We have developed REST-bench on the sound fundamentals of a taxonomy on REST alignment methods and validated the method in five case studies. Following the principles of technical action research, we collaborated with five companies, applying REST-bench and iteratively improving the method based on the lessons we learned. We applied REST-bench both in Agile and plan-driven environments, in projects lasting from weeks to years, and staffed as large as 1000 employees. The improvement opportunities we identified and the feedback we received indicate that the assessment was effective and efficient. Furthermore, participants confirmed that their understanding on the coordination between RE and ST improved.

1. Introduction

Requirements Engineering (RE) is the discipline of eliciting, analyzing, specifying, validating and managing needs and constraints on a software product [13,53]. Software Testing (ST) is the verification that a software product provides expected behaviors, as expressed in requirements [6,13]. As such, RE and ST[1] are intrinsically related and leveraging on this relationship would be beneficial for both disciplines [30]. Tassey [69] projected the cost of inadequate testing in the US to 60 billion dollars per year. Furthermore, a survey by Garousi and Varma [26] found that defects introduced in the requirements were among the most expensive to repair (besides stress/performance problems). Bjarnason et al. [8] identified sixteen REST alignment challenges, spanning from requirements and test quality to requirements abstraction levels [29] and traceability.

First steps to a better understanding of the REST alignment phenomenon were undertaken by studying and classifying alignment practices [70]. The main contribution of this classification is the definition of an epistemic base [47] that can be used to explain how and why REST alignment practices work. In this paper we use this base in order to provide a practical method, REST-bench, to assess REST alignment. A prerequisite for any improvement is the characterization of the current condition of the phenomenon under study. Based on this agreed state and the definition of goals, changes can be designed and implemented. Postmortems [7] are one possibility to elicit best practices but also issues in the execution of projects, feeding the results into an organizational knowledge repository [36]. Even though guidelines for executing postmortems exist [18,20], postmortem reviews are seldom held, some suggest for lack of time [27,37], even though their benefits are well reported [74]. REST-bench follows, like postmortems, the principle of the Experience Factory [3], where improvements are based on data collection and analysis of experience from past projects.

We designed REST-bench to be lightweight, in terms of resource use (30–50 person-hours per application), by focusing the assessment effort to the specific issue of requirements engineering and test coordination which is of major interest to organizations developing software intensive systems [8]. REST-bench is interview- and workshop-based, providing structured guidelines to collect and analyze data originating from project participants. In essence, REST-bench illustrates the impact of project artifacts on the coordination between RE and ST, and provides a set of analytical tools (the

* Corresponding author. Tel.: +46 455 385815.
E-mail addresses: mun@bth.se (M. Unterkalmsteiner), tgo@bth.se (T. Gorschek), rfd@bth.se (R. Feldt), ekx@bth.se (E. Klotins).

[1] We abbreviate requirements engineering and software testing as "RE and ST" or "REST" in the remainder of this paper.

http://dx.doi.org/10.1016/j.jss.2015.07.018
0164-1212/

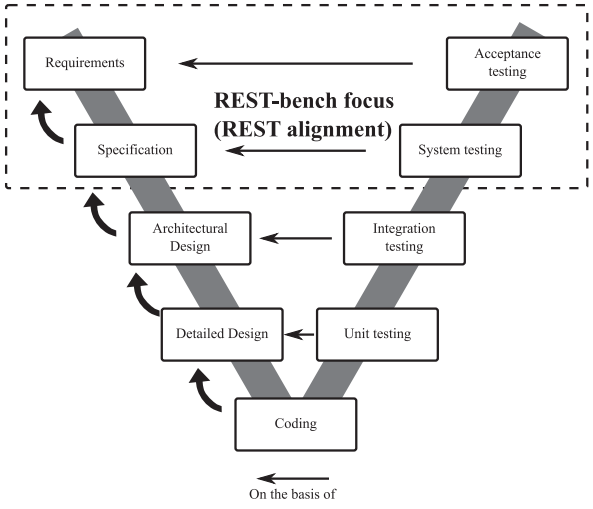


Fig. 1. V-Model of software development.

artifact map, seeding questions) that drive the analysis and elaboration of improvement suggestions.

In this paper, we present REST-bench in an example-driven manner, describing data elicitation, data preparation and the collaborative analysis. We illustrate the application of REST-bench in five companies that exhibit diverse characteristics. In all five cases we could identify relevant improvement opportunities. The participants of the assessment judged REST-bench as an efficient and effective mean to assess the coordination between RE and ST.

The remainder of this paper is structured as follows. We discuss background and related work in Section 2. Section 3 illustrates the research method we followed to validate and improve REST-bench. We introduce the assessment method in Section 4, together with a running example that shows the application of REST-bench and with the improvements we implemented, the collected data, the analysis of the results and the identified improvement potential. In Section 5 we show the results of the remaining four case studies. We answer our initially stated research questions in Section 6 and conclude the paper in Section 7.

2. Background and related work

2.1. REST alignment

The development of software-intensive systems[2] is a collaborative effort of experts, each contributing a part to the solution [60]. Expertise and capabilities are distributed on different roles, rendering the coordination between people in software projects and development phases essential for success [41]. From a process perspective, software development consists of transitions from system concept, requirements specification, analysis and design, implementation, and test and maintenance [44]. This abstraction holds for both plan-driven process models (e.g. spiral [11] and evolutionary [50], and the unified process model [42], as well as Agile models, although to a lesser extent as activities may be blended, eliminating transitions altogether (e.g. in eXtreme Programming [5]).

Fig. 1 illustrates the *V*-Model of software development, which originates from system engineering [14,23] and was adopted in software engineering [56]. This model illustrates vertical transitions between software development activities (left hand side) and the corresponding testing activities (right hand side) that ought to ensure the quality of the resulting work products. The importance of enabling these vertical transitions and align the intentions and activities across is demonstrated by an abundance of research (e.g. between requirements abstraction levels [28], requirements and software architecture/design [2,31,40], software architecture/design and implementation [1,22,49], and software architecture/design and testing [48,61]).

[2] A software-intensive system is "any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole" [35].

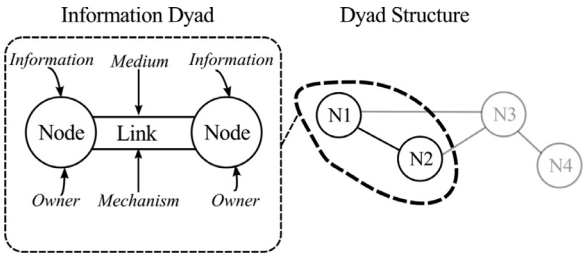


Fig. 2. Information dyad and dyad structure.

While research has produced an ample amount of software technologies, models and frameworks to coordinate people, to ease the transition between software development phases and to align the intentions and activities therein, literature is sparse on methods that focus on improving the coordination between requirements engineering and software testing [8]. In earlier work [70] we defined REST alignment as the *adjustment of RE and ST efforts for coordinated functioning and optimized product development*. The key in this definition is the intuition that RE and ST efforts need to be adjusted together in order to avoid sub-optimization of either one of the two aspects. This adjustment can be achieved by various means, spanning from process-centered activities that foster the collaboration between RE and ST roles (e.g. by forming cross-functional teams [45]), over techniques that use requirements as a driver for testing activities (e.g. by formulating testable contracts [46] or model-based testing [71]), to methods or processes that establish and/or maintain requirements to test traceability links [57]. Fundamental for any form of coordination is the exchange of information [73]. Therefore, in order to characterize the means of REST alignment, we use information as a central entity, as described next.

2.2. The REST taxonomy

To characterize means aimed at achieving REST alignment, we developed a taxonomy that uses the information dyad as building block to describe alignment methods [70]. Fig. 2 (left) illustrates the components of an information dyad: two nodes, characterized by the information they contain and their owner, are connected through a link. Creating the REST taxonomy, we identified different linking media and mechanisms. A medium can be of several different types: structured artifacts (e.g. documents, email, diagrams, database records); unstructured artifacts (e.g. audio and video); tools that act as means to share, transfer or transform information (e.g. modeling tools, language analysis tools); processes (one or more activities, performed repeatedly); or the organization of work environment (co-location, role/responsibility rotation). Furthermore, we identified four types of linking mechanisms: implicit connection (information is connected by volatile and implicit links that are not formalized); connection (information pertaining in each node is connected, establishing a logical link between the two nodes); bridge (information pertaining to each node is connected and augmented in order to achieve fitness of purpose in both nodes); and transformation (information, packaged for one node in the alignment dyad, is re-packaged in order to satisfy the needs of the other node).

Information dyads form a structure, Fig. 2 (right), which in turn may exhibit certain properties, listed in Table 1. We have used these properties to compare REST alignment methods reported in

Table 1
Dyad structure properties.

P1	Number of nodes – links between nodes need to be maintained over time. Hence, the total number of nodes allows one to reason on complexity and effort to maintain REST alignment.
P2	Branches – a branch exists, if a node acts as a source or sink for more than one node. Branches may reduce local complexity, require however also means to synchronize and merge information.
P3	Intermediate nodes – characterized by information that belongs to the design/analysis or implementation software development phases/activities.
P4	RE and ST node proportion – assuming that a node is associated with a certain cost (e.g. establishing/maintaining the information therein and links in between), it is of interest to know the node distribution among the RE and ST software development phases/activities.
P5	Links – the linking mechanism between software development phases/activities determines how changes of information are propagated.
P6	Scope – allows one to reason upon the interface between RE and ST and other software development phases/activities.

literature [70]. In this paper, one objective is to study to what extent these properties support the identification of improvement opportunities for REST alignment. Concretely, we used the dyad structure properties to generate seeding questions that support the analysis of the collected data in the REST-bench method. The steps in this process and the used seeding questions are illustrated in Section 4, while the question whether all dyad structure properties were useful is answered in Section 6.

2.3. Related work

The challenges of coordinating development teams operating at different sites have been described by Herbsleb and Grinter [32]. They point out the boundaries of explicit coordination mechanisms such as plans, interface specifications and process descriptions, but also the lack of informal coordination opportunities in geographically distributed sites, leading to misunderstandings and increase in cycle time for fixing issues. Following this line of thought, Herbsleb and Mockus [33] developed an empirical theory of coordination (ETC) for software engineering, based on decision and constraint networks, and later applied to identify coordination requirements among software developers that can be used to improve the design of collaboration tools [15]. Along a similar line, Ko et al. [39] observed in a field study what information developers seek in their day-to-day work, which sources they use and what the reasons are for not being able to gather the needed information, calling for innovation in tools, processes and notations. REST-bench shares with these studies the insight that satisfying information needs, timely and as exhaustive as possible, is key to successful software development in teams.

In order to represent information flow in requirements engineering activities, [62] developed a notation that can be used to model both formal and informal communication. A benefit of the flow notation is that it can be used to describe the officially required and the actually executed process, showing differences and instances where improvements can be implemented [66]. Furthermore, FLOW mapping has been used to plan and manage communication in distributed teams, requires however a considerable amount of manual work to keep the maps up-to-date with continuously changing communication patterns [65]. REST-bench shares with FLOW the idea to represent information and connections in a diagram that can be discussed in collaboration with practitioners to identify improvement opportunities. On the other hand, REST-bench provides a concrete assessment process, differentiates between data collected from the different roles, and provides heuristics that can be used to analyze the collected data and to generate analysis points that may lead to improvement suggestions.

Project postmortems have been successfully applied in the past ([20] provides an overview and examples) to identify improvements, following the Experience Factory principle [3]. However, Glass [27] has observed that they are seldom conducted due to the fast paced nature of software projects where teams are split up and reassigned to new tasks. Therefore, collecting information timely after the conclusion of a project seems more likely to identify improvement potential. In order to increase the data accuracy, Bjarnason et al. [9] propose project timelines that are prepared in advance. Depending on the particular analysis goals, certain aspects of the collected data are visualized in a timeline, allowing the postmortem participants to recall the illustrated events. However, since the method can be used for any generic improvement goal, it does not provide prompting questions or aids that could facilitate the analysis of the collected data. While REST-bench relies also on the collection of data from a specific past project, eliciting practitioners' experience on how information is used and created, it provides also seeding questions (Table 4) that support the analysis or the collected data. The focus on a particular goal (coordination of RE and ST), the structured data collection, and the analysis guided by a set of predefined questions and an artifact map, sets REST-bench apart from traditional project postmortems, as for example described by Dingsøyr [20].

3. Research method

Our overall research approach is oriented towards design research [34] which provides a concrete framework for implementing the dynamic validation phase in the technology transfer model proposed by Gorschek et al. [29]. Our research method is best described as technical action research [75, Chap. 19], i.e. we aim at improving and validating the fitness of purpose of an artifact by applying it in a real-world environment [76]. In particular, we want to answer the following research questions:

- RQ1: To what extent are the dyad structures from the REST taxonomy useful to elicit improvement opportunities?
- RQ2: To what extent is REST-bench useful in Agile and plan-driven environments?
- RQ3: To what extent is REST-bench usable?

In our previous work, we characterized REST alignment methods by means of information dyads [70]. Furthermore, we piloted the idea of using dyad structures as a mean to drive REST alignment assessment. With RQ1 we aim to validate this idea by applying the REST-bench method in a series of case studies. When we planned the validation, one of our concerns was the methods' reliance on documentation as a proxy to determine REST alignment. Therefore, we questioned whether we can apply REST-bench at all in an Agile environment. We address this concern in RQ2, where a plan-driven environment means that the development teams work in a traditional, document driven manner [54]. In order to be adopted by practitioners, a method should also be usable. We analyze the usability of REST-bench from the perspective of the analyst, i.e. the researcher who applied the method, as well as through the practitioners' feedback who participated in the study.

3.1. Project characteristics

We applied REST-bench in five companies located in Sweden: Ericsson and Telenor in Karlskrona, ST Ericsson in Lund, CompuGroup Medical and Volvo Cars in Gothenburg. All companies were approached based on personal contacts, providing them an executive summary of the goals, and expected cost and benefits of REST-bench. We did not preclude any particular company or project type in the selection. In the remainder of this paper we anonymized project characteristics, collected data and results by referring to Company A, B, C, D, and E. Table 2 illustrates the characteristics of the projects where

Table 2
Project characteristics of the participating companies.

Company	A	B	C	D	E
Project duration	12 months	2 months/release	36 months	48 months	7 months
Staff	150	20	9	1000	20
Software development approach	Agile (in transition from plan-driven)	Agile	Agile (embedded in plan-driven)	Plan-driven	Plan-driven
Requirements #	350	5–20	300	2000	50
Test-case #	700	Not stated	300	500	24
Assessment date	2012/06	2013/03	2013/03	2013/04	2013/12

we applied REST-bench. Note that Company A was, at the time of the assessment, in a transition from a plan-driven to an Agile software development approach. The team in Company C was working in an Agile manner while still being embedded in a plan-driven process. We selected the particular projects based on the criteria defined by REST-bench, described in Section 4.1.

3.2. Data collection and analysis

We collected data from two sources: (1) the process of applying REST-bench, e.g. effort spent and identified useful seeding questions; (2) a dedicated questionnaire, distributed to the participants after the assessment. Table 5 shows the 10 questions. The first four questions are open ended, addressing the overall experience of using REST-bench. The remaining questions required an answer on a 5 point Likert scale (strongly agree, agree, neutral, disagree, strongly disagree), while explanatory feedback was still possible. Out of the 13 participants in total, 10 returned the questionnaire.

In order to answer RQ1, we collected the seeding questions (see Table 4) used in the five collaborative workshops and analyzed their association with the dyad structure properties shown in Table 1. Furthermore, we use answers from the questionnaire to provide evidence for or against the relevance and usefulness of REST-bench.

Data to answer RQ2 stems from the project characteristics of the participating companies (Table 2), allowing us to stratify the results from the questionnaire into plan-driven/Agile sets. Furthermore, the assessment results from plan-driven and Agile environments provide indications on the usefulness of the approach.

In order to answer RQ3, we use data from the questionnaire, in particular questions regarding efficiency and effectiveness of the approach, and collect data on effort spent for the different steps in REST-bench.

3.3. Limitations

A major threat to validity in any technical action research is the involvement of the researcher in the application of the to be validated artifact [77]. The question whether the validation results depend solely on the ability of the researcher to use the artifact cannot be answered conclusively. In this paper, the first author performed all steps in REST-bench. An alternative would have been to train practitioners and to let them apply REST-bench on their own, observing the application and collecting data of spent effort, usefulness and usability. However, this would have led to a considerably higher assessment cost per company. Nevertheless, the validation did not solely depend on the researcher since practitioners where involved in the REST-bench application process, influencing thereby considerably the results.

We applied convenience sampling [58] to identify the case companies, i.e. we approached engineers and managers we had collaborated in the past. We prepared information material on REST-bench (purpose, expected effort, outcome and deliverables), providing the contact persons in the companies a basis on which they can decide whether or not to engage (all approached companies did). Note however that the selection of projects and interviewees followed the requirements of Step 1 of the REST-bench method (see Section 4.1). We have no indications that the contact persons at the companies influenced, other than by supporting the identification of interviewees, the outcome of the assessment with REST-bench or the questionnaire results.

We applied REST-bench in projects with a diverse set of characteristics (see Table 2). There is no feature or constraint in REST-bench that could conceivably exclude certain classes of projects or companies, even though very small teams that work co-located and do not require coordination with other teams might not benefit from an assessment with REST-bench. Note that REST-bench does *not* assess coordination through oral and informal communication which are still central for the successful execution of a software project (see answers to Q10 shown in Table 5). The post-assessment questionnaire was returned by 10 (out of 13) participants. While this is a high relative response rate (77%), the evidence provided by the questionnaire, in absolute terms, is still weak.

4. REST-bench

REST-bench follows the macro procedure of lightweight software process assessment [55], however it has a focused improvement goal (coordination between RE and ST) and utilizes elicitation and analysis practices tailored to this goal. This focus leads to a low investment cost, typically between 30 and 50 person-hours (p-h), depending on how many people are involved in the assessment (typically 3 to 5). We have a project view to make the assessment concrete, and avoid people give us the best or worst cases picked together from any part at any time in the company. This project focused assessment has been used before with success [68].

The aim of REST-bench is to assess the state of REST alignment [70], eventually initiating changes to improve that alignment. A critical enabler for any kind of change in an organization is support by senior management [21,52,78]. Therefore, we regard a champion in the organization that can provide resources and support the dissemination of the assessment results an essential pre-requisite before embarking in a REST-bench assessment.

We performed a REST-bench assessment at five case companies, and used the experiences to refine REST-bench. The first four cases (A–D) are illustrated in chronological order in Section 5, where we show the results of the assessment and the adaptions we made to the method. In this section we use the fifth case (E) as a running example to illustrate the individual steps of REST-bench. In all five case illustrations, the first author of this paper acted as moderator and analyst during the assessment.

REST-bench is interview-driven and begins therefore with the selection of interviewees (Step 1 in Fig. 3). After data elicitation (Step 2), the analyst creates an artifact map (Step 3). In Step 4, the analyst meets with the interviewees to collaboratively identify improvement opportunities, using the artifact map as input. The results of this assessment workshop are collected in a report (Step 5). In Sections 4.1–4.5 we exemplify the steps shown in Fig. 3, providing effort estimation and a compilation of best practices for the application of REST-bench.

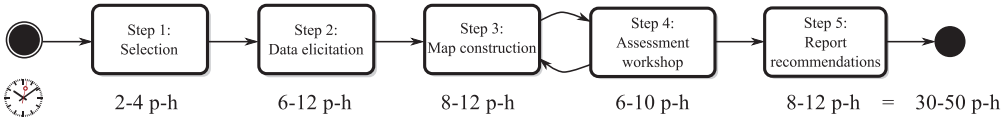


Fig. 3. The steps in the REST-bench method and budgeted effort in person-hours (p-h).

4.1. Step 1 – Selection

REST-bench has an interview-driven data elicitation process. Therefore, the selection of the particular interviewees is pivotal for the assessment, not unlike most process assessment methods such as e.g. CMMI SCAMPI [64] or SPICE [59]. A local champion in the organization, supporting the initiative, can be a great accelerator in identifying interview candidates. In order to maintain REST-bench's lightweight and focused nature, the roles of the 2–4 interviewees should pertain to the requirements domain, e.g. requirements engineers, business analysts, product managers, and the testing domain, e.g. test and quality assurance engineers. We recommend that the selection process considers the following characteristics:

1. Work experience in the company and in the particular role. A candidate that knows "how things work" and has progressed in the same company to a senior position is preferable over a qualified, but new hire.
2. The candidate from the RE and ST domain must have collaborated on the same project, preferably in more than one instance.
3. The project has recently been closed or is in a late stage.

These characteristics allow us to elicit data that reflect how engineers actually work as opposed to how they ought to work according to a process prescribed by a company policy. Choosing a project on which all interviewees collaborated allows us to conduct episodic interviews which facilitates the collection of "everyday knowledge about objects or processes" [4, p. 85]. In episodic interviews, we try use actual events that are connected to actions, experiences and consequences, to elicit relevant information from project participants. A project that is in its early stage can not set the context as this everyday knowledge (episodes) has still to be acquired. We detail data elicitation in Section 4.2.

In some organizations, the notion of a "project" is not clear cut. For example, one case company applies SCRUM [63] with three week sprints to implement customer requirements, releasing a new version of the product every two sprints. In this situation we chose the last sprint as "project" scope for the interviews. The relevant decision here is to agree on a specific context such that all interviewees can relate to it.

When selecting the interviewees it is important to keep the main goal of the assessment in mind: identify improvement opportunities for the coordination between requirements engineering and testing aspects of software development. Optimal candidates are employees in a senior position, being however involved in the day-to-day practice in their respective area. This leads usually to very fruitful analysis during the collaborative issue identification (Section 4.4).

Effort and best practices

The budgeted effort for this step is 2–4 person-hours, where the majority of the expense is on the organization, identifying the interview candidates. The analyst can support this by providing detailed selection criteria for projects and interviewees, described in this section. We recommend to schedule all meetings in advance (interviews, workshop) such that the overall assessment procedure is not extended to a long time period. Interviews should be held on one day, increasing the efficiency of data collection. The assessment workshop should be scheduled between 4 and 10 business days after the interviews, allowing time to analyze the collected data and preventing a loss of context by waiting too long after data collection.

Table 3
Example artifacts.

Powerpoint presentations, spreadsheets, text documents, specification / use case / user story stored in requirements management tool, acceptance / integration, system, unit test case stored in test management tool, UML / entity-relationship diagrams, emails, meeting notes, yellow sticky notes

Example - Step 1

Company E outsources the implementation and verification of parts of their product to an external supplier. In this scenario, the coordination between RE (Company E) and ST (supplier) is particularly challenging since company, country and time-zone borders are crossed. The selection of interviewee candidates was supported by the company's responsible for process improvement. We organized a seminar for employees of Company E, presenting the goals and example applications of REST-bench. This pro-activeness raised interest in the method and helped us to identify the project and interviewees for the assessment. We chose an Analyst Lead and a Business Process Expert/Acceptance Tester for the RE perspective, both working for Company E. For the ST perspective, we chose an Acceptance Test Manager working at Company E and the System Test Manager working for the supplier. The chosen project had a duration of 7 months involving a total staff of 20 employees (both Company E and supplier).

4.2. Step 2 – Data elicitation

We argued in our earlier work on requirements engineering and software testing alignment methods [70] that the means of connecting and using information is central to any software development effort. In order to operationalize this concept in a data collection procedure, we elicit which artifacts requirements and test engineers use and create in their daily work. We chose artifacts as a tangible proxy for information which can be retrieved during an interview. Furthermore, every software organization produces some set of artifacts, independently how agile or lean the organization is. An artifact can be any kind of digital or analog document produced by an employee. In our interview guide we exemplify what we mean by artifacts (see Table 3 for a list of examples).

Agreeing on the concept of artifacts is key to the episodic interview technique as it sets the scope of what data is elicited. Therefore, Phase 1[3] requires the interviewee to list all artifacts he/she has used or created in his/her daily work. The interviewee should follow the timeline of the project under investigation, recalling his/her involvement during the different project phases that required use or creation of any artifact. In Phase 2 of the interview we elicit the following data on each of the artifacts:

- Purpose: content of the artifact and reason for its creation.
- Creator: role of the person who creates the artifact, in which phase of the project.
- User: role of the person using the artifact, in which phase of the project.
- Modifier: role of the person changing the artifact, in which phase of the project.

[3] In Phase 0, we ask a set of introductory questions to elicit context information, such as name of the project, project duration, staff size, applied software development method, number of system requirements and test cases.

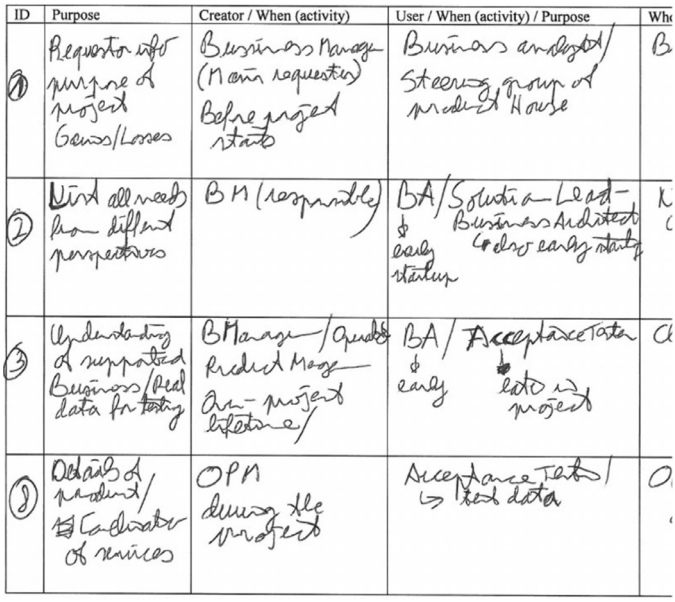


Fig. 4. Artifact elicitation example.

- Link or Mapping: a link is a uni-directional connection from one artifact to another; a mapping is a bi-directional connection between information contained in two artifacts.
- Use: a reference to any other artifact that is used as input to create this artifact.

We collect this data by compiling a simple template (see Fig. 4). It is common that the list elicited in Phase 1 is incomplete and is extended while detailing the artifact information in Phase 2 of the interview.

In order to reduce mutual influence during data collection, we interview requirements and test engineers separately. We also recommend to audio record the interviews (provided the interviewees give consent). With the episodic interview technique, we don't elicit data on artifacts in a vacuum, but enrich the information by relating it to the studied project and what has been experienced in the use and creation of artifacts. The analyst can then, while constructing the artifact map (see Section 4.3), develop a better understanding of the coordination between RE and ST, and prepare targeted analysis points for the assessment workshop.

Effort and best practices

The budgeted effort for this step is 6–12 person-hours. We allocate 1.5 hours per interview, which translates into 6 person-hours for one analyst and two interviewees as we interview RE and ST separately. Adding two interviewees doubles the effort to 12 person-hours.

Interviewees should have access to the project documentation during the data elicitation. This allows them not only to exemplify the artifacts they mention, but also to provide more accurate information on how artifacts are linked and related to each other.

Example - Step 2

The three interviews at Company E were performed on two days. The ST representative from the supplier company was interviewed by phone, however not audio recorded upon request by the interviewee. Fig. 4 shows an excerpt of the simple elicitation form used to record data. We experienced that notes are sufficient to record facts about artifacts, eliciting them in a structured fashion. However, episodic information on their use or misuse is complex and requires audio recordings that can be analyzed offline. When interviewees refer to events in the project, the analyst should focus on understanding these occurrences, in order to pose follow-up questions, rather than spending his attention on recording them manually.

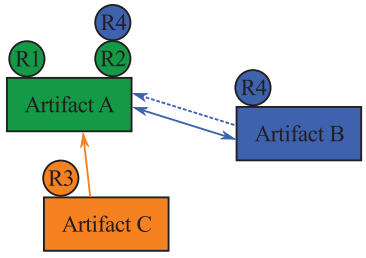


Fig. 5. Artifact map - illustration of components.

4.3. Step 3 – Map construction

An artifact map is an easy to understand, graphical representation of the data collected during the interviews. The visualization serves mainly three purposes:

1. Illustrating commonalities and differences of the RE and ST perspective on use and creation of artifacts.
2. Generating input for the collaborative issue identification.
3. Communicating and displaying the artifact topology to employees not involved in the assessment.

Fig. 5 illustrates the components of an artifact map. A rectangular box represents an artifact, a sphere with an identifier indicates a creator (on the top-left of the box) or an user (on the top-right of the box) of an artifact. "Linked-to" relationships are illustrated by solid lines with a single arrow, meaning that one artifact is linked to another. "Mapped-to" relationships are illustrated with double arrows, meaning that the information in the artifacts is mapped bi-directionally. "Used-to-create" relationships are illustrated by dashed lines with a single arrow, meaning that information in one artifact is used to create another artifact. The color coding indicates whether the data stems from RE (orange), ST (blue) or from both roles (green). The map in Fig. 5 reads as follows: *RE and ST agree that Artifact A is created by R1 and used by R2, even though ST adds R4 as an user. RE states that Artifact C is created by R3 and is linked to Artifact A. ST states that Artifact B is created by R4, using Artifact A is input, and information in Artifact B is mapped to information in Artifact A.*

This example illustrates a rather common situation where ST states that a role (R4) uses information from one artifact (A) to create another artifact (B), whereas RE is not aware of this information need. This leads to an analysis point that needs to be addressed in the collaborative assessment workshop. One outcome could be that missing to mention Artifact B and R4 was an oversight by RE. Then the map is simply updated. Another potential outcome is that RE is indeed unaware of the information need by R4. This misalignment could then be analyzed in more depth during the assessment workshop, identifying potential causes but also consequences. For example, if Artifact A is seldom updated, R4 might use outdated information to create Artifact B. The solution could be to maintain Artifact A during the project. Another likely solution: R4 should use Artifact C which would otherwise have no use (no user was defined for it during the elicitation).

Table 4 lists questions that the analyst can use to identify analysis points for the assessment workshop. The questions in set *P0* focus on finding causes of disagreement between RE and ST, which could be a misunderstanding in data elicitation, but also a genuine divergence that potentially causes misalignment. The questions in sets *P1, P2, P5* and *P6* are mapped to the information dyad structure properties identified in our previous work [70]. Note that these questions are meant as an inspiration for the analyst and do not apply to every

Table 4
Seeding questions to prepare the REST-bench assessment workshop.

P0 - What is the source of disagreement between RE and ST on...
1. ... the existence of an artifact?
2. ... the creator/user of an artifact?
3. ... who changes when an artifact?
4. ... "linked-to"/"mapped-to" relationships between artifacts?
5. ... linking mechanism between artifacts?
6. ... "used-to-create" relationships between artifacts?
P1 - Number of artifacts
6. Is there an information need that was not fulfilled by the used artifacts?
7. If a new artifact is added, how would that impact other artifacts in terms of maintaining information consistent?
8. Given that artifact A doesn't have any user OR is only used by role R, could the information in artifact A be merged into artifact B?
P2 - Artifact relationships
9. How is the information in artifact A kept consistent with the information in artifact B, in the case C changes (C has two "linked-to" OR "used-to-create" relationships to sibling artifacts A and B)?
10. If inconsistencies between artifacts A and B arise, how does that impact the users of those artifacts and their work?
11. What is the purpose of artifacts that are not related to any other artifact?
12. Do the creators of artifact A deliver timely, i.e. can the information actually be accessed in ST when needed?
P5 - Artifact and role changes
13. Does inconsistency of information among artifacts affect the work in: RE, ST, the interface between both?
14. In case requirements change, by whom/how/when are these changes propagated to linked artifacts?
15. How does staff turnover affect the quality of requirements and derivative artifacts?
P6 - Artifact and role scope
16. Would involvement of RE/ST in creating artifact A improve the alignment?
17. How is consistency between input from non-RE/ST artifacts and RE/ST artifacts maintained over time?

artifact map. While applying the REST-bench method, we improved the formulation and extended the set of questions.

An artifact map encodes only a subset of the information elicited during the interviews. It is the task of the analyst to use the remaining data together with the collected episodic information to create analysis points for the collaborative assessment workshop. For example, the data elicitation may reveal that an artifact is regularly updated during a project. However, there is no mechanism to propagate changes to artifacts or roles that use this modified information, leading potentially to misalignment. Episodic information, such as an event where an ST had to redesign a test suite due to outdated requirements specifications can be supporting evidence for a pattern observed in the artifact map.

Effort and best practices

The budgeted effort for this step is 8–12 person-hours. Some effort (40%) should be spent to represent the artifact map graphically. Representing the artifact map in a clear, easy to understand graph is crucial for the assessment workshop since the participants do not have much time to familiarize with complicated notation. In the cases studies, we used an all-purpose diagramming tool [79], however have since then developed a dedicated tool[4] that supports the analyst in creating the artifact map. The remainder of the effort should be spent on preparing a list of questions for the assessment workshop, using the prepared artifact map and the audio recordings of the interviews. Note that the audio recordings are not meant to be transcribed, but serve as source for details that were not captured yet in the map and to verify the created map. The list of questions should be separated into a clarifying part, addressing potential misunderstandings from the interviews, and an analysis part, seeded by the questions from Table 4.

[4] A prototype is available at http://lmsteiner.com/restbench.

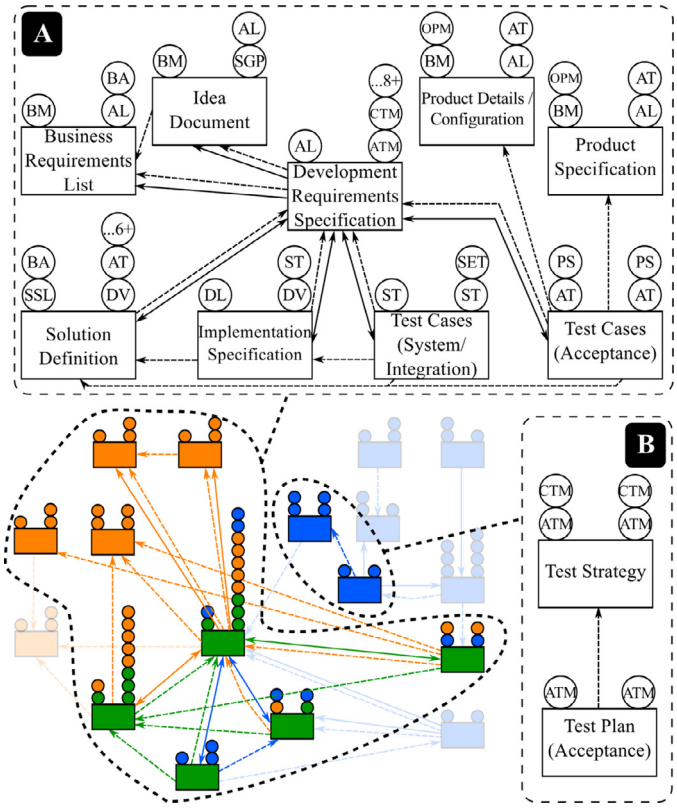


Fig. 6. Artifact map of Case E.

Example - Step 3

Fig. 6 shows the artifact map elicited at Company E. Note that for reasons of presentation clarity, we omit details of the complete artifact map and show only the elements important for the analysis, discussed during the assessment workshop, in panels A and B (further discussed in Section 4.4). The color coding (orange for RE, blue for ST and green for artifacts mentioned by both perspectives) illustrates the degree of agreement on the creation and use of artifacts. The commonly identified artifacts represent thereby the interface between RE and ST. Inconsistencies emerged from the data collection (and partially visible in the map) were picked up as entry points for analysis during the assessment workshop, for example:

- Even though both RE and ST agree on Solution Definition and Development Requirements Specification (see Fig. 6 and panel A therein), RE states that there is an explicit mapping between individual requirements and solution descriptions while ST claims there is no linking at all.
- The Test Strategy artifact was mentioned by the ST at Company E, but not by the ST at the supplier.
- A disagreement between RE of Company E and ST of the supplier emerged on the change frequency and time of the Development Requirements Specification.

These divergences between the RE and ST perspective on the artifact map are a result of the seeding questions P0 in Table 4 and are complemented with further questions originating from this seeding set, as shown in the next step.

4.4. Step 4 – Assessment workshop

The REST-bench assessment workshop is led by the analyst, introducing to all interviewees the process of the collaborative issue identification:

1. Presentation and explanation of the artifact map
2. Error identification by interviewees
3. Clarification questions by analyst
4. Collaborative map analysis
5. Summary and wrap up

The analyst brings two printouts of the artifact map to the workshop, one for himself to note any corrections and one for the participants, explaining the notation and content of the artifact map. He should also point out immediately that the information in the map stems from the interviews, setting the context to the particular project that was selected for the data elicitation. The interviewees should spend 5–10 min with the map and try to identify any unclear artifacts, roles or incorrect relationships between artifacts. The analyst clarifies any uncertainties that emerged in the map construction[5]. The main task on which most of the allocated time should be spend on is to answer the questions prepared by the analyst, based on Table 4.

As for the initial data elicitation, we recommend to audio record the assessment workshop, while the analyst should take notes of the major identified issues. An issue might be related to the process (or not followed process), structure, content or distribution of artifacts, coordination between roles using artifacts, documentation infrastructure or lack thereof. It is important that all participants agree on an issue and the implications of it. The analyst should elicit evidence of these implications, e.g. in form of events during the project under discussion, from the workshop participants.

Effort and best practices

The budgeted effort for this step is 6–10 person-hours. A realistic time-frame for the REST-bench assessment workshop execution is 2 h.

Even though the analyst initiates and drives the assessment workshop with prepared analysis points and questions, improvement opportunities are most likely to be acted upon when they emerge from the company employees. Therefore, the main goal of the analyst is to provoke an exchange between RE and ST, supported by the artifact map as a mean of communication, avoiding however conflicts and steer the discussion into a constructive direction. During the assessment workshop it is also important to maintain the context of the studied project, even though extrapolation to other projects is possible in order to understand the impact of an identified improvement opportunity.

Example - Step 4

Looking at Panel A in Fig. 6, the analyst observed that the Development Requirements Specification (i.e. the requirements therein) is mapped to the central artifacts in acceptance testing, system and integration testing, and development (Implementation Specification and Solution Definition). This mapping allows the company to assess the coverage of requirements both in terms of testing and implementation, enabling the monitoring of both development and testing progress. The analyst argued that a mapping between individual business requirements, e.g. from the Business Requirements List or the Idea Document, and the Development Requirements Specification would be important in order to know which business requirement corresponds to a particular development requirement. Indeed, RE confirmed during the workshop that it is not always clear to them which business requirements are already in production.

RE pointed out that such a mapping is difficult to achieve, since the Business Requirements List is not a standardized document and does also contain information not relevant for the requirements analysis. RE stated that the naming of use cases (part of the Development Requirements Specification) should describe the corresponding business requirement, creating an implicit mapping. According to RE, this is however not trivial and requires a lot of thought, as the use cases must illustrate the business process such that the use cases can be used "as-is" documentation for other projects. The analyst concluded that different roles have different requirements on the use cases and their names, making them not necessarily ideal for determining coverage of business requirements. Therefore, it might be better to pursue an explicit mapping, requiring however a formalization of the business requirements.

RE stated that it is sometimes difficult to get the Development Requirements Specification reviewed by the customer (internally, this is a Business Manager who requests a set of features). It is the customers' responsibility to look into the Development Requirements Specification and understand what they have agreed upon and verify the coverage of the business requirements. RE stated that even though it worked well in this project, it is still challenging in general to get customers to read the Development Requirements Specification and confirm that the business requirements are covered. The analyst concluded that an explicit mapping between business and development requirements would render the coverage analysis more effective.

Looking at Panel A in Fig. 6, the analyst observed a second issue: on the acceptance level, test cases are created by using information from the Product Specification, Product Details/Configuration, Development Requirements Specification and the Solution Definition. Business requirements, represented in the Business Requirements List and the Idea Document are not used at all. Given the lack of mapping between business requirements and solution requirements, resulting in a potentially unknown business requirements coverage, the analyst concluded that using business requirements as input for creating acceptance test cases would be a good idea.

This issue was controversially discussed during the assessment workshop. On one hand, ST stated that it should not be required to use the business requirements directly as input, since all business requirements selected for the project should be represented in the Development Requirements Specification and the use cases therein. On the other hand, RE and the analyst argued that the Development Requirements Specification and the Solution Definition in particular are documents of the solution space, tending to describe the designed solution rather than the problem. The purpose of acceptance tests is to verify that the problem has been addressed by the solution whereas the purpose of system/integration tests is to verify that the solution implements the requirements correctly [13]. The analyst concluded that a potential consequence of broadening the purpose of acceptance tests is that verification is duplicated, e.g. covering aspects that have been already covered in system tests. Indeed, RE confirmed that this can happen whereas ST stated that this is not the case in general.

Less controversial was the observation made by the analyst on Panel B in Fig. 6 that there is no RE role involved in defining the Test Strategy or the Test Plan. Both RE and ST agreed that input from the RE perspective would be beneficial. A review of these artifacts by RE would validate that the tested functionality is the actually required functionality.

4.5. Step 5 – Report recommendations

The purpose of this step is to summarize the findings from the workshop and to communicate them to a wider audience. For that reason, the report should be concise while still providing enough context in order to be useful for employees that did not participate in the assessment or in the studied project.

Effort and best practices

The budgeted effort for this step is 8–12 person-hours. We recommend to provide a short introduction of the assessments purpose and

[5] This is why map construction and collaborative issue identification in Fig. 3 are shown as iterative steps.

scope. Then, the artifact map, based on the data collected in the interviews, should be presented, highlighting inconsistencies and analysis points relevant for the assessment workshop. The workshop summary should answer these analysis points, summarize the identified improvement opportunities, pointing to evidence in the updated artifact map, and eventually conclude with a set of recommendations. The report should be send to the study participants for review before it is communicated to a wider audience, allowing for corrections by the participants.

Example - Step 5

In summary, based on the observations made during the collaborative workshop (see the example in Step 4), three areas for improvement were identified:

- Business requirements gap: one of the Business Managers tasks is the review of the Development Requirements Specification in order to verify coverage of business requirements. The effectiveness of this review depends on the skill/capability of the customer in understanding the Development Requirements Specification. One improvement would be to strengthen these skills by training. On the other hand, creating an explicit mapping from Development Requirements Specification to business requirements would remove the need for review. However, such a solution would require a formalization of the Business Requirements List.
- Acceptance and System/Integration Test alignment: there are indications that these test-levels have at least a certain overlap in what they actually verify. As long as this overlap is intentionally created, the additional effort can be controlled, otherwise this is an area where test efficiency can be improved.
- Involvement of the RE perspective, e.g. involving RE in Test Strategy and Test Plan reviews would help in validating that the development requirements are tested correctly.

5. Case studies

In this section we present the four remaining cases where we applied REST-bench, chronologically such that improvements in the method follow logically from the described cases. Each subsection introduces first the case company, summarizes the assessment results and discusses the impact the lessons learned had on REST-bench.

5.1. Company A

We selected a system manager with 12 years experience in his current role as RE representative. For the ST representative we chose a verification engineer with 14 years experience. The project in which the two engineers collaborated completed in autumn 2011, while we performed the assessment in June 2012. The project staff consisted of 150 engineers, split up into seven teams. The system requirements consisted of approximately 350 user stories, whereas the system test cases amounted to 700, of which 550 were automated.

5.1.1. Assessment results

Fig. 7 shows the artifact map used during the REST-bench assessment workshop. For clarity, we highlight only those details of the map (Panel A and B) relevant for the identified improvement opportunities.

Panel A in Fig. 7 illustrates that the Software Verification Team (SVT) uses four artifacts as input to create the Test Cases. The Requirements Documentation is the main source, complemented by the other artifacts. During the assessment workshop the analyst questioned whether inconsistencies in these documents, caused for instance by requirements changes during the design or implementation, affect ST. According to RE, changes in user stories (part of the Requirements Documentation) are propagated to the SVT, which is

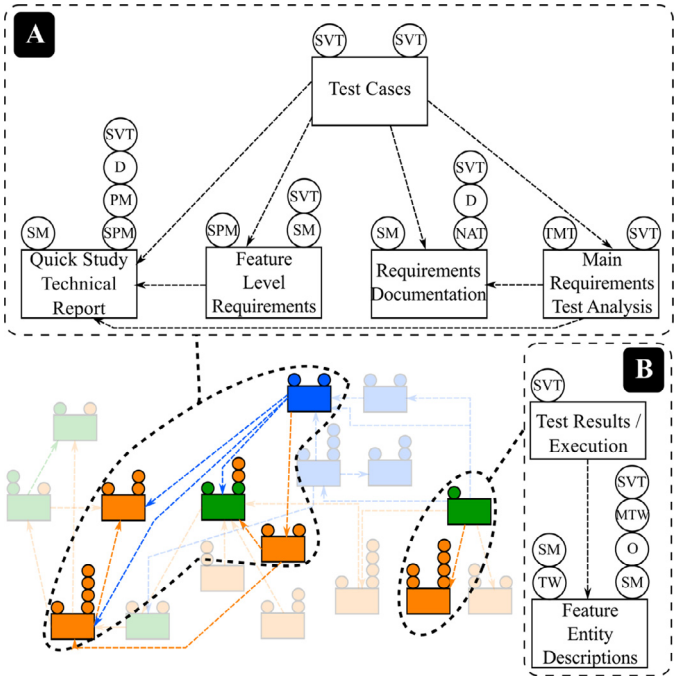


Fig. 7. Artifact map of Case A.

invited to the presentations where changes are discussed. In the context of this particular project, RE stated that parts of the solution where redesigned late in the project, when the development already had started. This was confirmed by ST, asserting that there were inconsistencies between Test Cases and Requirements Documentation due to a too early test analysis.

The Requirements Documentation artifact was elicited from both interviewees and represents therefore an important interface between RE and ST. According to ST, there is however no linking between Requirements Documentation and Test Cases at the Integration Test level. This is due do the difficulty to keep these links up-to-date; early attempts with spreadsheets failed and importing the required information into the test management tool is labor intensive. According to RE, this lack of linking may lead to lower test coverage of the requirements. ST stated that the lack of traceability is to some extent compensated by the Technical Manager for Test (TMT) whose responsibility it is to define the test scope. The analyst concluded that the TMT acts thereby as a link between RE and ST. However, testers need to pull information from TMT, rendering the spread of knowledge on requirements changes a matter of individual initiative, that is, person dependent.

Panel B in Fig. 7 shows that Feature Entity Descriptions, describing the system functionality on a compound level, are used by the SVT to interpret the Test Results and Execution. However, RE stated that Feature Entity Descriptions are written late in the project, by external consultants, describing how the system is actually implemented. RE stated that this is a local sub-optimization that saves effort in the RE department, affects however the work of ST. The analyst concluded that, to render Feature Entity Descriptions useful to ST, they should be created and maintained during the project as the implementation stabilizes. To summarize, the following improvement opportunities were identified:

- Test Cases are created by using different sources of information which are potentially inconsistent.
- Requirements changes, tracked by a dedicated management role (TMT), should be propagated to the responsible test engineers.
- Feature Entity Descriptions should be written earlier such that they are of more use to ST.

5.1.2. Impact on REST-bench

We intended to collect and analyze both artifacts (tangible information) and events when information is exchanged informally, e.g. in ad-hoc meetings, email and on-line conversations, or phone calls. However, eliciting each individual instance of informal communication seemed unrealistic for the targeted effort budget (6–12 person-hours) for data elicitation. The positive results of this first assessment, i.e. the identification of issues, supported our intuition that we already collect enough data that can be efficiently analyzed in the assessment workshop and lead to concrete improvement suggestions.

Regarding the relationships between artifacts, we aimed at producing rich analysis points, identifying *what* information is used by *whom* to create *which* artifact. Therefore we focused to elicit information regarding "used-to-create" relationships. However, using an artifact to create another does not necessarily mean that the two artifacts are linked, rendering the REST taxonomy heuristics [70] more difficult to apply. The workshop in Company A revealed that links between artifacts, or lack thereof (in that case, Requirements Documentation and Test Cases at the Integration Test level), would also be useful to analyze. Therefore we decided to also elicit "linked-to" relationships between artifacts and present them as solid lines in the map. Note that in the following two assessments, at Company B and C, we focused on eliciting only "linked-to" relationships (partially to maintain the elicitation time budget, partially to keep the artifact map easy to interpret). However, we realized that "used-to-create" relationships between artifacts are useful for the analysis (see Section 5.3.2), and introduced them again in the assessments of Company D and E.

Finally, we relied exclusively on note taking during the interviews. Even though this sped up the artifact map creation process, much time during the assessment workshop was spent on correcting the map. Hence we decided to audio record future interviews and workshops, given that participants gave consent.

5.2. Company B

In Company B we could not identify a single project, since Scrum teams (one business analysis and two development/test teams) work continuously on a product, releasing a new version every two months. We chose therefore the last release-cycle as time-frame for the assessment and selected a requirements engineer and a test lead as interviewees. The total staff consisted of 20 engineers while the assessment took place in March 2013.

5.2.1. Assessment results

RE stated that regulatory requirements enforce that the development process provides traceability, illustrating that safety risks are considered in the requirements, and that the developed services or products comply to these requirements (verification). However, according to RE, maintaining traceability over time is complicated by storing information in different formats. While all artifacts are stored in a content management system providing revision control, creating and maintaining links between artifacts is not supported by the system. ST stated that this complicates impact analysis (even though traces between artifacts exist, they need to be found manually as the content management system does not handle traces), e.g. when a requirement changes and the corresponding Test Items and Test Templates need to be updated or rerun. The analyst concluded that this "infrastructure gap" may lead to an increased effort in impact analysis and to potential inconsistencies in test artifacts. Another related issue is the redundancy of manual traceability links. For example, looking at Panel A in Fig. 8, both Product Backlog Item and Use Case refer to a particular risk. RE stated that these links are maintained manually, causing additional effort to check consistency. ST stated that inconsistencies in these manual links may be caught (early) during the review of the use case or (late) when the sprint is approved. RE and ST agreed

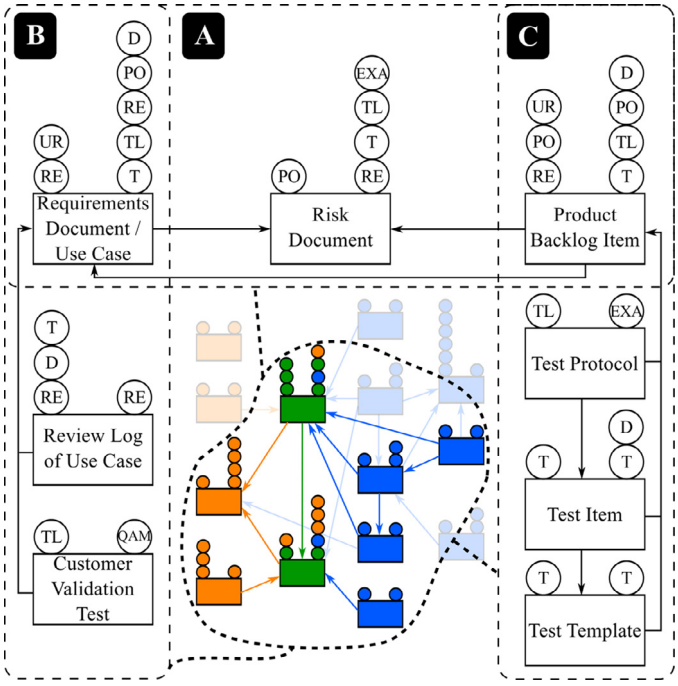


Fig. 8. Artifact map of Case B.

that it would be sufficient to link the Risk Document to Product Backlog Items.

While traceability is generally established with unique references, ST stated that Customer Validation Tests are linked implicitly to Use Cases by naming conventions (Panel B in Fig. 8). These naming conventions, created by the Test Lead, are however not enforced or documented. The analyst concluded that staff turnover may therefore easily break this link. Furthermore, as soon as the complexity of the validation tests rises, a more sophisticated tracing mechanism may be required in order to keep track whether and how the Use Case is covered by the executed scenarios in the Customer Validation Tests.

Looking at Panel B in Fig. 8, the analyst observed that ST roles are involved in reviewing documents created by RE. There is however no such involvement of RE roles in reviewing documents created by ST (Panel C). RE stated that they occasionally, when a complex function needs to be verified, provide feedback to ST on particular test cases. Such ad-hoc reviews do however not verify the test scope. The analyst concluded that this could be achieved by having a review of the Test Template and the test-cases therein (early, before the tests are actually executed), or of the Test Protocol (late, when the test results are available). To summarize, the following improvement opportunities were identified:

- Manual maintenance of traceability links leads to increased effort in impact analysis. Redundant, manually maintained, traceability links may lead to inconsistencies requiring rework.
- Customer Validation Tests are not explicitly mapped to Use Cases.
- RE is little involved in reviewing document created by ST.

5.3. Company C

We selected a requirements engineer/software architect with 3 years experience in his current role as RE representative. For the ST representative we chose a test lead with 1.5 years experience. The assessment took place in March 2013, approximately at halftime of the expected three year project duration. The selected project had a staffing of 9 engineers, operating in a Scrum team. The overall development process in Company C is plan-driven, posing challenges to the project that iterates in two week sprints. The project was responsible for approximately 300 system requirements.

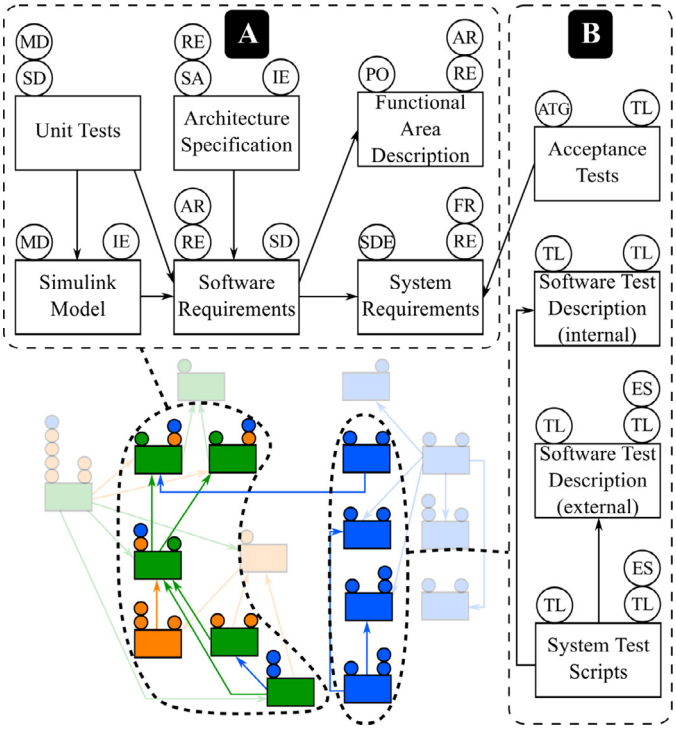


Fig. 9. Artifact map of Case C.

5.3.1. Assessment results

An aspect that is not visible in the artifact map is the project lifecycle and how it influences the artifacts that are created and used. This project was estimated to last three years in total. This investigation took place approximately 1.5 years into the project, at a time when the high-level requirements (System requirements and Functional Area Descriptions, see Panel A in Fig. 9) were stabilized and soon to be frozen. Due to the volatility of the requirements in the initiation phase of the project, the development team focused on establishing the feasibility of the product, adapting continuously on changing requirements. This led to postponing the formalization of documentation and links, since this process was time consuming and there was a resistance to start it before there was an indication that the system was actually working. Therefore, even if Panel A in Fig. 9 suggests that traceability from high-level requirements down to unit-tests is given, this was only a goal that has not yet been achieved at this stage of the project. In particular, the links from the Software Requirements to System Requirements and Functional Area Descriptions were in the process of being re-engineered and implemented in the requirements specification tool (SpecTool).

Looking at Panel B in Fig. 9, the analyst observed that there is a gap between the integration tests (Software Test Description artifact) and the requirements they are intended to verify (System Requirements and Functional Area Description). Software Requirements had, according to RE, a 1-to-1 mapping to unit-tests. On the other hand, System Test Descriptions, which specify how and what was tested in integration, were not linked to any requirements documentation. Both RE and ST agreed that linking the System Test Descriptions to System Requirements and Functional Area Descriptions would provide benefits:

- ability to perform requirements coverage analysis
- change impact analysis, leading to reduced test-time
- allow reviews in which both RE and ST participate and validate the effectiveness of integration tests

In this project, according to RE, the software team started to integrate System Requirements and Functional Area Descriptions in SpecTool, allowing them to link their information to Software Requirements. Although this was an improvement (as manual linking to word documents was not necessary anymore), linking the requirements to the integration tests was still a challenge, since test artifacts were stored in a separated infrastructure, requiring a manual linking and maintenance process. According to RE, the link implementation in SpecTool by itself may turn out to be problematic. Even minor changes in parent documents, such as correction of spelling mistakes or improving descriptions, would trigger a change event, leading to child documents to be flagged as outdated. There was no possibility in SpecTool to qualify the level of a change. That may be a risk, leading to a lower quality of documentation. To summarize, the following improvement opportunities were identified:

- Time consuming re-engineering of Software Requirements and linking to other artifacts.
- Lack of traceability between integration tests and requirements they are intended to verify.

5.3.2. Impact on REST-bench

As we assessed Company B and C in parallel, we summarize here the lessons learned from both REST-bench applications. As a major change, we reintroduced the elicitation of "used-to-create" relationships between artifacts, adding how information from one artifact is used to create another artifact. This differs from the "linked-to" semantics and is useful as it provides a dynamic aspect to artifact relationships that is lost when considering only links between artifacts. For example, consistency between artifacts is much more difficult to achieve and maintain when there exist "used-to-create" but no "linked-to" relationships between artifacts.

Nevertheless, the assessment in Company C illustrated also one of the weaknesses of a static artifact map. At different points in time, the particular use of an artifact may change in a project. This dynamism is not visible in an artifact map, but can be captured during data collection and then considered while discussing the map at the assessment workshop.

5.4. Company D

We selected a requirements engineer and a system designer, with 10 and 5 years experience respectively, as RE representatives. We chose a test lead with 5 years experience as representative for the ST perspective. The selected project had a four year duration, involving in peak times about 1000 hard-and software engineers. During the assessment on April 2013, the project was in the closing phase. The project handled approximately 2000 system requirements and 500 functional test cases.

5.4.1. Assessment results

RE stated that Implementation Proposals [25] were typically written as "change this existing functionality like that", in collaboration by RE and ST roles, and were used by system designers to architect the software and by ST to develop the test plan (see Panel A in Fig. 10). Function Descriptions, on the other hand, were written when the software has been implemented, providing a complete description of a function, and were used by ST to write test specifications. ST reported that the information in the Implementation Proposals and the respective Function Descriptions may be inconsistent, leading to situations where they attempted to test functionality that is not supported by the delivered software. Inconsistencies stemmed from the fact that Function Descriptions were created late in the project forcing ST to use the Implementation Proposal to write test-specifications, and that no role has been assigned the responsibility to check and maintain consistency between Implementation Proposals and Function Descriptions.

Due to the amount of Implementation Proposals and Functional Descriptions per project, maintaining consistency among them is

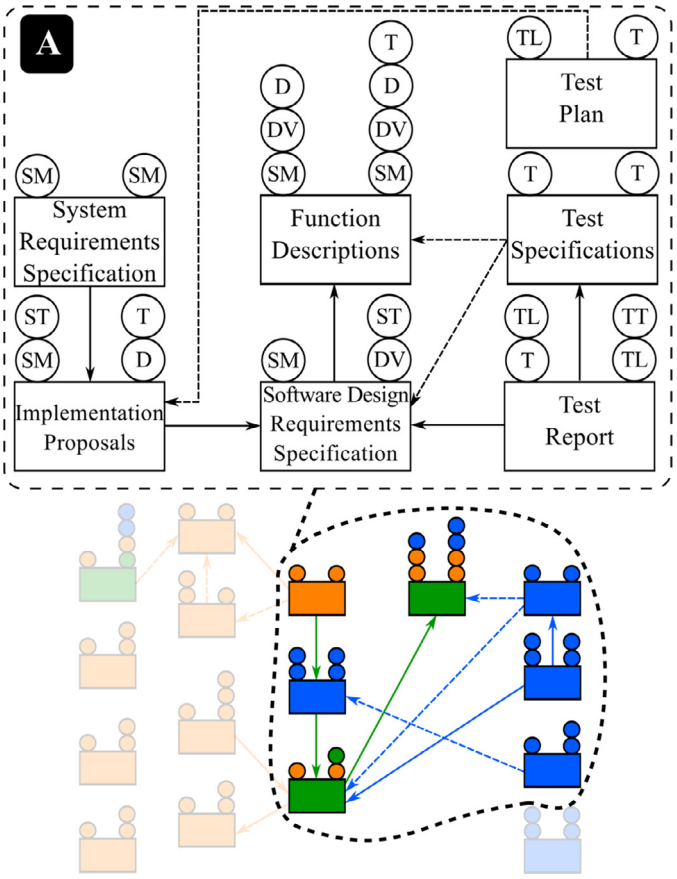


Fig. 10. Artifact map of Case D.

a challenge and may not even be possible with reasonable effort. One possible improvement, mentioned by RE, would be to implement a mechanism that alerts the reader of potential inconsistencies. Furthermore, the responsibility (who, why) and the procedure (what, when) of updating/creating Function Descriptions should be defined. The Implementation Proposal was written by both RE and ST, whereby ST was responsible for the test specific parts. RE stated that the distance in time in which different parts in the Implementation Proposals were written has been very long (up to one year): while RE was working on an Implementation Proposal, ST might be heavily involved in testing, leading to the situation that ST was actually not collaborating with RE in the specification. Then, when ST was ready to specify the testing part, the document may have been out-of-date as it was not updated by RE during the project. RE proposed an improvement to the process, that is, to ensure the commitment of both RE and ST to collaborate at the same time on the Implementation Proposal specification.

ST explained that in their work process, they mapped the requirements (from the Software Design Requirements Specification) to the test cases they execute. In the Test Report they documented test results and listed which requirements were covered by the executed test cases (see Panel A in Fig. 10). There was however, according to RE, no feedback loop from ST to RE, leading to the inability to improve low quality requirements when they were marked as untestable in the Test Report. Furthermore, test-coverage could not be efficiently measured. Both RE and ST agreed that a requirement to test-case mapping would have the following benefits:

- Regression tests could be efficiently defined (assuming that Software Design Requirements Specification is traced to requirements at higher abstraction levels).
- Change requests involving the addition or removal of requirements would be visible on the ST side, i.e. it would be transparent whether new test cases are needed, or old ones can be removed.
- Statements of verification could be generated without manual effort.

Looking at the artifact map in Fig. 10, the analyst observed that ST was, except in the definition of Implementation Proposals where the benefit is immediate, not involved in RE activities. RE stated that previous efforts to include ST had failed, mostly due to scheduling issues. With the initiative to improve the quality of the Software Design Requirements Specification, new opportunities for ST involvement could arise. For example, ST could provide useful input on the requirements quality by looking at their Test Reports. Not testable requirements identified in these reports can either be improved or even removed from the database. A second opportunity for ST involvement arises as soon as the requirements to test case mapping is implemented. New requirements need to be reviewed and mapped to either existing test cases or the decision needs to be taken to create new test cases. ST could be assigned this responsibility. Then, with the requirements to test case mapping in place, ST can provide feedback, during and after the testing phase, to RE on the quality of requirements, preventing the decay of requirements quality. To summarize, the following improvement opportunities were identified:

- Maintaining consistency between Improvement Proposals and Function Descriptions in order to support ST in testing the correct functionality.
- Schedule collaboration between RE and ST such that work on common artifacts leads to coordination rather than misunderstandings.
- Map test cases to requirements such that faults in requirements identified by ST can be fed back to RE.

5.4.2. Impact on REST-bench

In Company D, we analyzed the lack of mapping between individual requirements and test cases. Even though requirements are linked in the Test Report to individual test cases, RE could not benefit from that information as no bi-directional links existed. This observation, together with the analysis of Company C (see Section 5.3.1), where we did not differentiate between uni-directional and bi-directional relationships, led us to the introduction of the “mapped-to” relationship. The semantics of the “mapped-to” relationship allows us to describe links between artifacts that can be followed from either side (corresponding to backward and forward traceability).

6. Discussion

Looking at the assessments performed with REST-bench, one commonality encountered in all five cases is the identification of improvement possibilities in the linking mechanisms between nodes of information. This means either to establish a link in the first place, or improve an existing link such that it is more resilient, accurate or reliable over time. In some cases, a solution would be to introduce explicit traces between artifacts, either by adopting a viable trace model [57] or by automatically recovering trace links (see Borg et al. [12] for an overview of techniques). In other cases, the involvement of test engineers in requirements engineering tasks would be beneficial, as observed by Damian and Chisan [19], Uusitalo et al. [72] and Kukkanen et al. [43]. Generally, the assessed companies were well aware that information either from RE or ST is not used to its full potential for decision support. Consequentially, the study participants valued REST-bench’s ability to identify gaps in the coordination between RE and ST. Similar observations, focused on gaps between RE and downstream development affecting test coverage and scoping were made by Bjarnason et al. [10] in the context of large-scale

software development. However, since the assessments did not include solution development, monitoring and evaluation of the implemented changes, we do not have resilient results that would allow us to provide general recommendations on how to improve REST alignment. In the remainder of this section we answer the research questions stated in Section 3.

6.1. RQ1: To what extent are the dyad structures from the REST taxonomy useful to elicit improvement opportunities?

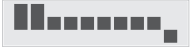
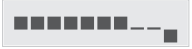
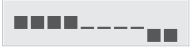
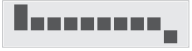
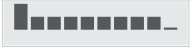
Table 4 illustrates the mapping between the seeding questions we used to prepare the assessment workshops and the dyad structure properties, summarized in Table 1, we identified in our earlier work [70]. We established the mapping in Table 4 in order to understand which dyad structure properties are actually useful in generating analytical questions regarding the elicited artifact maps. While properties *P1, P2, P5* and *P6* were useful, we could not yet derive questions by using properties *P3* and *P4* (see Table 4 for a description of these properties). However, this does not preclude the possibility that in future assessments those properties might generate useful seeding questions.

Looking at the overall focus of REST-bench, it seems straightforward why no questions regarding *P3*, intermediate nodes, were generated: the data elicitation is geared towards artifacts that are used and created by requirements engineers and software testers (see Section 4.2). Q4 in our post assessment questionnaire addressed this specific aspect, i.e. whether other roles (e.g. software architects, developers) should be included in the assessment. Nine out of ten participants suggested including other roles (six argued for software developers, one for a configuration manager, two for no specific role). While we decided not to include more roles, in order to keep the overall assessment effort low, one could argue that a developer participating in the assessment workshop alone would be beneficial as a user of requirements and creator of the system under test, thereby contributing to the identification of improvement potential.

The second dyad structure property for which we could not generate analytical questions is *P4* - RE and ST node proportion. This property expresses the relative effort spent in creating and maintained RE and ST artifacts respectively. However, we did not find any evidence that this proportion could indicate an issue in REST alignment or serve as an analytical lever to identify improvement opportunities.

To gauge the overall relevance and usefulness of REST-bench, we asked questions Q1–Q3 (see Table 5) to participants. Regarding the relevance of REST alignment (Q3), eight out of ten participants reported that they had in the past discussions on how to better coordinate requirements engineering and testing activities. As one participant put it, "*we have had some discussions in my team how to get the correct information when we need it from RE. But we have not discussed that with the RE organization yet.*" All ten participants stated that they would use REST-bench as a complement to project post-mortems (Q2), as "*it gives a complete picture of the artifacts, the relations between them and a good material for use in discussions about potential process improvements*", and "*is a very good and easy way to ensure that all involved use and acknowledge each others documents and their purpose.*" This is further corroborated by the results on Q5 and Q6, where we asked the participants whether REST-bench is efficient and effective (see Table 5). One participant stated that "*the method gives more truth on what is actually used. We already have processes for a lot of things but these are somewhat theoretical and not adapted per project.*" On the other hand, another participant stated that "*it is not really guaranteed that we take action just because we have identified the issues.*" With respect to Q1, whether participants learned something new from the artifact map or the assessment workshop, 8 out of 10 participants indicated that they identified aspects in the way of working they were not aware of. This is further supported by the results to Q7 in Table 5, even though a participant stated that "*I believe I already had a pretty good overview of the situation. Probably because we are a relatively small team where everybody works with a broad range of tasks*", and some other stated that "*it confirmed my thoughts.*" Another aspect we investigated is whether REST-bench increases the

Table 5
Post assessment questionnaire.

Open ended questions		
Q1	Did the elicitation (interview session), the artifact map or the collaborative workshop reveal something new, you did not know before, w.r.t. activities, responsibilities or artifacts in: requirements engineering, software testing, the interplay between both?	
Q2	Given the resources you have invested, would you consider to use this method as a complement to project post-mortems to improve the coordination between RE and ST? Please motivate, why yes/no/maybe.	
Q3	Did you discuss the coordination between RE and ST in your organization already before this assessment? If yes, which specific issue(s) have been discussed and why?	
Q4	In general, to elicit more relevant and precise information, would you suggest to include another role (e.g. Software Architect, Developer) in the assessment? Please motivate, why yes/no/maybe.	
5 point Likert scale with possibility to provide explanatory feedback		Results[3]
Q5	The assessment (including elicitation and workshop) is an *efficient*[1] mean to identify issues in relation to the coordination between Requirement Engineering and Software Test.	
Q6	The assessment (including elicitation and workshop) is an *effective*[2] mean to identify issues in relation to the coordination between Requirement Engineering and Software Test.	
Q7	The artifact map and the discussion in the workshop increased my understanding of the coordination between RE and ST in the studied project.	
Q8	The artifact map and the discussion in the workshop increased my awareness for potential waste (e.g. unused documentation).	
Q9	The artifact map and the discussion in the workshop increased my awareness for potential gaps in the coordination between RE and ST.	
Q10	Interactions (scheduled meetings, but also casual encounters over coffee, in the corridor or at the office door) are equal or even more important as written documentation for the successful execution of the project.	

[1] Efficiency in general describes the extent to which time or effort is well used for the intended task or purpose.
[2] Effectiveness is the capability of producing a desired result. When something is deemed effective, it means it has an intended or expected outcome.
[3] The bars represent the 10 responses from the study participants on a Likert scale (strongly agree, agree, neutral, disagree, strongly disagree).

awareness of potential waste (Q8) or gaps (Q9) in the coordination between RE and ST. The results (see Table 5) suggest that the potential is more geared towards identifying gaps than waste. REST-bench can be used to identify waste candidates [38]. However, to actually remove waste all project stakeholders are necessary. This corroborates the result from Q4 discussed earlier, including developers could improve the potential of REST-bench to eliminate waste.

6.2. RQ2: To what extent is REST-bench in Agile and plan-driven environments useful?

One of our concerns when we planned the validation of REST-bench was the methods' focus on using documentation as a proxy to determine REST alignment. Indeed, in Case A described in Section 5.1, we planned to complement the data collection with recording instances of informal communication, however rejected the idea due to the involved effort and inaccuracy to do that manually. Therefore, we questioned whether we can apply REST-bench in Agile environments at all, and collect relevant data and produce useful results. Agile approaches are notorious for promoting as little documentation as possible [16,17,24,51].

In order to understand whether there is a difference in usefulness of REST-bench, we analyzed the post assessment questionnaire by stratifying the answers into plan-driven and Agile groups, using the project characteristics illustrated in Table 2. There were five participants in each group. The largest divergence can be observed in Q8 (REST-bench increases the awareness for potential waste). The Agile group tends to disagree ( ), while the plan-driven group tends to agree ( ). On the other hand, both the Agile ( ) as well the plan-driven ( ) group agree that REST-bench increases the awareness of gaps in the coordination between RE and ST (Q9). Since REST-bench assesses coordination through the creation/use of documentation, this result from the Agile respondents might seem surprising. However, this conforms to the observations made in a survey among Agile practitioners where the majority reported that documentation is important or very important [67]. Both groups show similar tendencies to the remaining questions:

- Q5 - efficiency of REST-bench: Agile ( ) and plan-driven ( )
- Q6 - effectiveness of REST-bench: Agile ( ) and plan-driven ( )
- Q7 - increased understanding: Agile ( ) and plan-driven ( )
- Q10 - importance of informal interactions: Agile ( ) and plan-driven ( )

These results suggest that REST-bench is useful both in Agile and plan-driven environments. This is further corroborated by the actual case assessment results, where for both Agile and plan-driven projects relevant improvement opportunities were identified.

6.3. To what extent is REST-bench usable?

We answer this question from the perspective of the researcher who applied the method. REST-bench has not been transferred to an industry partner so that ease of use from the perspective of a practitioner can not be validated yet. However, the results from the questionnaire indicate that REST-bench is, from a participants' perspective, efficient and effective (see Q5 and Q6 in Table 5).

We look at three aspects of usability, breaking it down to each step in REST-bench: required up-front investment, the effort to perform, and the support REST-bench provides in each step. With up-front investment we mean the one-time cost for the analyst to learn a particular step. In Table 6, we provide the relative cost of each step. In total, we estimate the up-front cost to learn the REST-bench method to 8–12 h.

Table 6
Usability assessment of REST-bench.

Step	Cost (%)	Effort (p/h)	Support within REST-bench
Selection	5	2–4	Heuristics on participant profile
Data elicitation	15	6–12	Defined procedure / templates
Map construction	40	8–12	Seeding questions / mapping tool prototype
Collaborative issue identification	30	6–10	Defined procedure
Report recommendations	10	8–12	Report structure

The required estimated effort is shown in the third column in Table 6, and further motivated in the respective sections where each step is explained (Sections 4.1–4.5). We regard these estimates as upper limits (with the suggested number of participants), if the assessment is performed in regular intervals. A common detractor for conducting postmortems is lack of time [27,37]. Therefore, we designed REST-bench to be efficient, while still providing value to the organization conducting the assessment. This efficiency is achieved by providing guidelines and heuristics within REST-bench (see fourth column in Table 6). Furthermore, each of the steps is supported by a series of examples presented in this paper.

7. Conclusions

The paper presents a method to identify improvement opportunities in the coordination between requirements engineering (RE) and software testing (ST). The method, REST-bench, is based on the premise that information, the way it is created, used and linked, is key for understanding how requirements and test engineers coordinate and how their collaboration can be improved. We used the information dyad and the emerged dyad structure properties from our earlier work [70] to drive an assessment process that is designed to be lightweight in terms of resource use (30–50 person-hours per assessment).

We validated REST-bench by applying it in five companies. In all assessments we could identify issues that were taken up by the involved companies to trigger improvements. With respect to RQ1, whether the dyad structure properties from the REST taxonomy were useful, we found that all but two properties generated seeding questions that can be used to identify issues and elicit improvement opportunities. Furthermore, the feedback gathered from the assessment participants supports our conclusion that REST-bench is an effective method to identify gaps, and to a lesser degree waste, in the artifacts that support RE and ST coordination. With respect to RQ2, applying REST-bench in Agile and plan-driven contexts, we conclude that the proposed method is useful in both while it is more likely to identify waste in plan-driven contexts. Furthermore, projects with very small teams requiring little coordination will likely not benefit from a REST-bench assessment. Answering RQ3, we found that REST-bench is usable from the perspective of the analyst who conducted the assessments.

To corroborate these results we plan to further validate the usefulness and usability of REST-bench by training practitioners in the autonomous use of the method, supported by efficient tools for data collection and artifact mapping.

Acknowledgments

We would like to thank the participating companies for their collaboration. The research was funded by EASE Industrial Excellence Center for Embedded Applications Software Engineering (http://ease.cs.lth.se).

References

Aldrich, J., Chambers, C., Notkin, D., 2002. ArchJava: connecting software architecture to implementation. In: Proceedings of 24th International Conference on Software Engineering (ICSE). IEEE, Orlando, USA, pp. 187–197.

Amyot, D., Mussbacher, G., 2001. Bridging the requirements/design gap in dynamic systems with use case maps (UCMs). In: Proceedings of 23rd International Conference on Software Engineering (ICSE). IEEE, Toronto, Canada, pp. 743–744.

Basili, V.R., Caldiera, G., 1995. Improve software quality by reusing knowledge and experience. Sloan Manage. Rev. 37 (1), 55–64.

Bauer, M.W., Gaskell, G., 2000. Qualitative Researching with Text, Image and Sound: A Practical Handbook for Social Research. SAGE.

Beck, K., 1999. Embracing change with extreme programming. Computer 32 (10), 70–77.

Bertolino, A., 2007. Software testing research: achievements, challenges, dreams. In: Proceedings Workshop on the Future of Software Engineering (FOSE). IEEE, Minneapolis, USA, pp. 85–103.

Birk, A., Dingsøyr, T., Stålhane, T., 2002. Postmortem: never leave a project without it. IEEE Softw. 19 (3), 43–45.

Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B., Sabaliauskaite, G., Loconsole, A., Gorschek, T., Feldt, R., 2014. Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empir. Softw. Eng. 19 (6), 1809–1855.

Bjarnason, E., Svensson, R.B., Regnell, B., 2012. Evidence-based timelines for project retrospectives - a method for assessing requirements engineering in context. In: Proceedings 2nd International Workshop on Empirical Requirements Engineering (EmpiRE). IEEE, Chicago, USA, pp. 17–24.

Bjarnason, E., Wnuk, K., Regnell, B., 2011. Requirements are slipping through the gaps - a case study on causes & effects of communication gaps in large-scale software development. In: Proceedings of the 19th Requirements Engineering Conference (RE). IEEE, Trento, Italy, pp. 37–46.

Boehm, B.W., 1988. A spiral model of software development and enhancement. Computer 21 (5), 61–72.

Borg, M., Runeson, P., Ardö, A., 2014. Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. Empirical Software Engineering 19 (6), 1565–1616.

Bourque, P., Fairley, R.E. (Eds.), 2014. Guide to the Software Engineering Body of Knowledge, 3rd. IEEE.

Bröhl, A.-P., Dröschel, W., 1995. Das V- Modell. Der Standard in der Softwareentwicklung mit Praxisleitfaden. Oldenbourg Verlag.

Cataldo, M., Wagstrom, P.A., Herbsleb, J.D., Carley, K.M., 2006. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In: Proceedings 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW). ACM, Banff, Canada, pp. 353–362.

Chau, T., Maurer, F., Melnik, G., 2003. Knowledge sharing: agile methods vs. tayloristic methods. In: Proceedings of the 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE, Linz, Austria, pp. 302–307.

Cohen, D., Lindvall, M., Costa, P., 2004. An introduction to agile methods. In: Advances in Computers, vol. 62. Elsevier, pp. 1–66.

Collier, B., DeMarco, T., Fearey, P., 1996. A defined process for project post mortem review. IEEE Softw. 13 (4), 65–72.

Damian, D., Chisan, J., 2006. An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. Trans. Softw. Eng. 32 (7), 433–453.

Dingsøyr, T., 2005. Postmortem reviews: purpose and approaches in software engineering. Inf. Softw. Technol. 47 (5), 293–303.

Dybå, T., 2005. An empirical investigation of the key factors for success in software process improvement. Trans. Softw. Eng. 31 (5), 410–424.

Elrad, T., Aldawud, O., Bader, A., 2002. Aspect-oriented modeling: bridging the gap between implementation and design. In: Proceedings of the 1st Conference on Generative Programming and Component Engineering (GPCE). Springer, Pittsburgh, USA, pp. 189–201.

Forsberg, K., Mooz, H., 1991. The relationship of system engineering to the project cycle. In: Proceedings of the National Council for System Engineering (NCOSE) Conference. Center for Systems Management, Chattanooga, USA, pp. 57–65.

Fowler, M., Highsmith, J., 2001. The agile manifesto. Softw. Dev. 9 (8), 28–35.

Fricker, S., Gorschek, T., Byman, C., Schmidle, A., 2010. Handshaking with implementation proposals: negotiating requirements understanding. IEEE Softw. 27 (2), 72–80.

Garousi, V., Varma, T., 2010. A replicated survey of software testing practices in the Canadian province of Alberta: what has changed from 2004 to 2009? J. Syst. Softw. 83 (11), 2251–2262.

Glass, R.L., 2002. Project retrospectives, and why they never happen. IEEE Softw. 19 (5), 111–112.

Gorschek, T., Wohlin, C., 2006. Requirements abstraction model. Requir. Eng. 11 (1), 79–101.

Gorschek, T., Wohlin, C., Carre, P., Larsson, S., 2006. A model for technology transfer in practice. IEEE Softw. 23 (6), 88–95.

Graham, D., 2002. Requirements and testing: seven missing-link myths. IEEE Softw. 19 (5), 15–17.

Hall, J.G., Jackson, M., Laney, R.C., Nuseibeh, B., Rapanotti, L., 2002. Relating software requirements and architectures using problem frames. In: Proceedings of 10th International Conference on Requirements Engineering (RE). IEEE, Essen, Germany, pp. 137–144.

Herbsleb, J.D., Grinter, R.E., 1999. Splitting the organization and integrating the code: Conway's law revisited. In: Proceedings of 21st International Conference on Software Engineering (ICSE). ACM, Los Angeles, USA, pp. 85–95.

Herbsleb, J.D., Mockus, A., 2003. Formulation and preliminary test of an empirical theory of coordination in software engineering. In: Proceedings of 9th European Software Engineering Conference Held Jointly with 11th International Symposium on Foundations of Software Engineering (ESEC/FSE). ACM, Helsinki, Finland, pp. 137–138.

Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design science in information systems research. MIS Q. 28 (1), 75–105.

ISO/IEC, 2007. ISO/IEC standard for systems and software engineering - recommended practice for architectural description of software-intensive systems.

Ivarsson, M., Gorschek, T., 2012. Tool support for disseminating and improving development practices. Softw. Qual. J. 20 (1), 173–199.

Keegan, A., Turner, J.R., 2001. quantity versus quality in project-based learning practices. Manage. Learn. 32 (1), 77–98.

Khurum, M., Petersen, K., Gorschek, T., 2014. Extending value stream mapping through waste definition beyond customer perspective. J. Softw.: Evol. Process 26 (12), 1074–1105.

Ko, A.J., DeLine, R., Venolia, G., 2007. Information needs in collocated software development teams. In: Proceedings of 29th International Conference on Software Engineering (ICSE). IEEE, Minneapolis, USA, pp. 344–353.

Kop, C., Mayr, H.C., 1998. Conceptual predesign bridging the gap between requirements and conceptual design. In: Proceedings 3rd International Conference on Requirements Engineering (RE). IEEE, Colorado Springs, USA, pp. 90–98.

Kraut, R.E., Streeter, L.A., 1995. Coordination in software development. Commun. ACM 38 (3), 69–81.

Kruchten, P., 2000. The Rational Unified Process: An Introduction, 2nd Addison-Wesley Longman Publishing, Boston, USA.

Kukkanen, J., Väkeväinen, K., Kauppinen, M., Uusitalo, E., 2009. Applying a systematic approach to link requirements and testing: a case study. In: Proceedings 6th Asia-Pacific Software Engineering Conference (APSEC). IEEE, Penang, Malaysia, pp. 482–488.

Laplante, P.A., 2007. What every engineer should know about software engineering, 1st CRC Press.

Marczak, S., Damian, D., 2011. How interaction between roles shapes the communication structure in requirements-driven collaboration. In: Proceedings 19th International Conference on Requirements Engineering (RE). IEEE, Trento, Italy, pp. 47–56.

Melnik, G., Maurer, F., Chiasson, M., 2006. Executable acceptance tests for communicating business requirements: customer perspective. In: Proceedings Agile 2006 Conference. IEEE, Minneapolis, USA, pp. 12–46.

Mokyr, J., 2005. Long-term economic growth and the history of technology. In: Handbook of Economic Growth, vol. 1, Part B. Elsevier, pp. 1113–1180.

Muccini, H., Bertolino, A., Inverardi, P., 2004. Using software architecture for code testing. Trans. Softw. Eng. 30 (3), 160–171.

Murphy, G.C., Notkin, D., Sullivan, K.J., 2001. Software reflexion models: bridging the gap between design and implementation. Trans. Softw. Eng. 27 (4), 364–380.

Naumann, J.D., Jenkins, A.M., 1982. Prototyping: the new paradigm for systems development. MIS Q. 6 (3), 29–44.

Nerur, S., Mahapatra, R., Mangalaraj, G., 2005. Challenges of migrating to agile methodologies. Commun. ACM 48 (5), 72–78.

Niazi, M., Wilson, D., Zowghi, D., 2006. Critical success factors for software process improvement implementation: an empirical study. Softw. Process: Improv. Pract. 11 (2), 193–211.

Nuseibeh, B., Easterbrook, S., 2000. requirements engineering: a roadmap. In: Proceedings of 22nd International Conference on Engineering, Future of Software Engineering Track (ICSE). ACM, Limerick, Ireland, pp. 35–46.

Petersen, K., Wohlin, C., 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices: an industrial case study. Empirical Softw. Eng. 15 (6), 654–693.

Pettersson, F., Ivarsson, M., Gorschek, T., Öhman, P., 2008. A practitioner's guide to light weight software process assessment and improvement planning. J. Syst. Softw. 81 (6), 972–995.

Pfleeger, S.L., Atlee, J.M., 2009. Software Engineering: Theory and Practice, 4th Prentice Hall.

Ramesh, B., Jarke, M., 2001. Toward reference models for requirements traceability. Trans. Softw. Eng. 27 (1), 58–93.

Robson, C., 2002. Real World Research: A Resource for Social Scientists and Practitioner-researchers, 2nd John Wiley & Sons.

Rout, T.P., El Emam, K., Fusani, M., Goldenson, D., Jung, H.-W., 2007. SPICE in retrospect: developing a standard for process assessment. J. Syst. Softw. 80 (9), 1483–1493.

Rus, I., Lindvall, M., 2002. Knowledge management in software engineering. IEEE Softw. 19 (3), 26–38.

Samuel, P., Mall, R., Kanth, P., 2007. Automatic test case generation from UML communication diagrams. Inf. Softw. Technol. 49 (2), 158–171.

Schneider, K., Stapel, K., Knauss, E., 2008. Beyond documents: visualizing informal communication. In: Proceedings of 3rd International Workshop on Requirements Engineering Visualization. IEEE, Barcelona, Spain, pp. 31–40.

Schwaber, K., 1997. SCRUM development process. In: Business Object Design and Implementation. Springer London, pp. 117–134.

SEI, 2006. Appraisal Requirements for CMMI, Version 1.2 (ARC, V1.2). Technical Report CMU/SEI-2006-TR-011. Software Engineering Institute, Carnegie Mellon. Pittsburgh, USA. http://www.sei.cmu.edu/library/abstracts/reports/06tr011.cfm

Stapel, K., Knauss, E., Schneider, K., Zazworka, N., 2011. FLOW mapping: planning and managing communication in distributed teams. In: Proceedings of the 6th International Conference on Global Software Engineering (ICGSE). IEEE, Helsinki, Finland, pp. 190–199.
Stapel, K., Schneider, K., Lübke, D., Flohr, T., 2007. Improving an industrial reference process by information flow analysis: a case study. In: Proceedings 8th International Conference on Product-Focused Software Process Improvement (PROFES). Springer, Riga, Latvia, pp. 147–159.
Stettina, C.J., Heijstek, W., 2011. Necessary and neglected?: an empirical study of internal documentation in agile software development teams. In: Proceedings 29th International Conference on Design of Communication (SIGDOC). ACM, Pisa, Italy, pp. 159–166.
Svahnberg, M., Gorschek, T., Nguyen, T.T.L., Nguyen, M., 2015. Uni-REPM: a framework for requirements engineering process assessment. Requir. Eng. 20 (1), 91–118.
Tassey, G., 2002. The Economic Impacts of Inadequate Infrastructure for Software Testing. Technical Report 7007.011. National Institute of Standards and Technology, Gaithersburg, USA.
Unterkalmsteiner, M., Feldt, R., Gorschek, T., 2014. A taxonomy for requirementsengineering and software test alignment. Trans. Softw. Eng. Methodol. 23 (2), 1–38.
Utting, M., Pretschner, A., Legeard, B., 2012. A taxonomy of model-based testing approaches. Softw. Test., Verification Reliability 22 (5), 297–312.
Uusitalo, E.J., Komssi, M., Kauppinen, M., Davis, A.M., 2008. Linking requirements and testing in practice. In: Proceedings 16th International Conference on Requirements Engineering (RE). IEEE, Catalunya, Spain, pp. 265–270.
van de Ven, A.H., Delbecq, A.L., Koenig, R., 1976. Determinants of coordination modes within organizations. Am. Sociol. Rev. 41 (2), 322–338.
Verner, J.M., Evanco, W.M., 2005. In-house software development: what project management practices lead to success? IEEE Softw. 22 (1), 86–93.
Wieringa, R., 2014. Design Science Methodology for Information Systems and Software Engineering, 1st Springer, Berlin, Germany.
Wieringa, R., 2014. Empirical research methods for technology validation: scaling up to practice. J. Syst. Softw. 95, 19–31.
Wieringa, R., Moralii, A., 2012. Technical action research as a validation method in information systems design science. In: Proceedings of the 7th International Conference on Design Science Research in Information Systems and Technology (DESRIST). Springer, Las Vegas, USA, pp. 220–238.
Wohlin, C., Aurum, A., Angelis, L., Phillips, L., Dittrich, Y., Gorschek, T., Grahn, H., Henningsson, K., Kagstrom, S., Low, G., Rovegard, P., Tomaszewski, P., van Toorn, C., Winter, J., 2012. The success factors powering industry-academia collaboration. IEEE Softw. 29 (2), 67–73.
yWorks, 2014. yEd - GraphEditor. http://www.yworks.com/en/products_yed_about.html, (accessed 27.07.15).

Michael Unterkalmsteiner received a BSc degree in applied computer science from the Free University of Bolzano/Bozen (FUB) in 2007 and a MSc degree in software engineering at the Blekinge Institute of Technology (BTH) in 2009. He is currently working toward a PhD degree at BTH where he is with the Software Engineering Research Lab. His research interests include software repository mining, software measurement and testing, process improvement, and requirements engineering. His current research focuses on the coordination of requirements engineering and verification and validation processes. For more information or contact: www.lmsteiner.com

Dr. Tony Gorschek is a Professor of Software Engineering at Blekinge Institute of Technology (Sweden) and part time at Chalmers University. He has over ten years industrial experience as a CTO, senior executive consultant and engineer, but also as chief architect and product manager. In addition he has built up five start-ups in fields ranging from logistics to internet based services. Currently he manages his own consultancy company, works as a CTO, and serves on several boards in companies developing cutting edge technology and products. His research interests include requirements engineering, technology and product management, process assessment and improvement, quality assurance, and practical innovation. He is a member of IEEE and ACM. For more information and publications or contact: www.gorschek.com

Dr. Robert Feldt is a professor of software engineering at Chalmers University of Technology, Sweden and at Blekinge Institute of Technology, Sweden. He has also worked as an IT and software consultant for more than 18 years. His research interests include human-centered software engineering, software testing and verification and validation, automated software engineering, requirements engineering and user experience. Most of the research is of empirical nature and conducted in close collaboration with industry partners. He received a Ph.D. (Techn. Dr.) in computer engineering from Chalmers University of Technology in 2002.

Eriks Klotins is a PhD student in software engineering at the Blekinge Institute of Technology (BTH). His research interests include software-intensive product development practices in start-up companies. Eriks Klotins received an MSc in software engineering from BTH. Contact him at ekx@bth.se.