

Seeing the Forest and the Trees: Focusing Team Interaction on Value and Effort Drivers

Matthias Book, Simon Grapenthin, Volker Gruhn
paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen, Gerlingstr. 16, 45127 Essen, Germany
{matthias.book, simon.grapenthin, volker.gruhn}@paluno.uni-due.de

ABSTRACT

Large-scale information system development is often plagued by defects and deadline overruns that can be traced to insufficient communication within the project team, particularly between stakeholders from the business, technical and management side. Although agile process models put a strong emphasis on team communication, they provide only little support for focusing the communication on the most relevant issues. We therefore introduce the concept of so-called “interaction rooms”, where teams work with a pragmatic combination of model sketches and annotations to foster understanding of the system and its business domain, to reveal risks and uncertainties, and discuss those system aspects that are most critical for project success.

Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tools and Techniques; D.2.9 [Software Engineering]: Management

Keywords

Teams, communication, value-orientation, risks

1. INTRODUCTION

Today, software engineers have at their disposal a multitude of notations to specify virtually all aspects of an information system, and tools to support considerable parts of the development process – and yet software projects in practice often run over time or budget, do not meet quality or functionality expectations, or are even aborted before completion. In most cases, the reason for such wasted efforts can be traced to communication problems. Curtis et al. identified the “thin spread of application domain knowledge” as a major factor contributing to project troubles already almost 25 years ago [4]. While method support has matured considerably since then, the complexity of today’s information systems has multiplied as well. The increasing flexibility and interconnection of business processes introduce whole

new levels of complexity and thus an even higher need for stakeholder communication [6].

This is confirmed by our observations of industrial software development practice in a multitude of medium- and large-scale projects, mostly in the insurance, financial services and healthcare industry, over the past years: While domain experts and technical experts both strive for a joint understanding of the information system that shall be built, more or less obvious barriers and negligence soon tend to impede communication. These observations suggest that better support for team members’ communication and collaboration is required. However, most approaches to provide collaboration support revolve around development tools instead of addressing the underlying cognitive aspects [10].

Agile process models have been introduced to ensure more frequent and consistent alignment of business and technical stakeholder positions through continuous feedback cycles [7]. However, while agile process models encourage (and actually depend on) intensive communication, they typically do not provide explicit operational support for it. Instead, we observe that the different background, expertise, culture and goals of different stakeholder groups (software engineers, domain experts, managers) often lead to poor communication and thus the following effects, which we believe to be the prime contributors to project risk:

- **Poor understanding/overview** of business domain requirements and technical design rationales: Domain experts tend to be sidelined once the initial requirements analysis process has been completed, with discussion shifting to the technical aspects. This usually does not occur consciously, but implicitly due to the choice of more technical modeling notations, focus on technical design decisions, etc. However, without a consistent and continuous understanding of both business and technical aspects of the system, conflicts and errors become apparent only at late stages when they are costly to fix.
- **Negligence of value and effort drivers:** We often observe that teams focus on those aspects of a system that are well understood, since it is easy to produce voluminous specifications and functioning artifacts for them. However, this often keeps the team’s attention on the more trivial aspects of a system, while a blind eye is turned (consciously or subconsciously) to the actual value and effort drivers – i.e. those components that hold particular value for the viability and effectiveness of the system, or those that demand higher

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGSOFT’12/FSE-20, November 11–16, 2012, Cary, North Carolina, USA.
Copyright 2012 ACM 978-1-4503-1614-9/12/11 ...\$15.00.

effort due to their intrinsic complexity or special requirements. The negligence of the value and effort drivers in favor of boilerplate functionality generates an illusion of project progress that is dangerous because those system aspects that would merit most attention, and should shape the design of the rest of the system, are not addressed until project resources are more scarce, and possibly conflicting design decisions have already been established.

The above issues are related to a lack of understanding of project aspects that are not explicitly expressed in typical models of software systems, but crucial to project success. In order to prevent these issues from growing unnoticed until they pose significant risks to a project, we aim to make them more visible and tangible for all stakeholders, and thus bring them into the focus of interaction before they can develop into problems.

2. THE INTERACTION ROOM

In order to achieve this, we advocate the creation of so-called “interaction rooms” for complex software projects. An interaction room is a physical room that is outfitted to visualize and facilitate discussion of key aspects of an information system: Surrounding a conference table, the walls of the room are covered with large sketches of models describing those system aspects that are most critical for project success. Each of the four walls is dedicated to a particular modeling perspective, as described below – however not displaying it in the dreaded fine-printed wallpaper-size completeness, but focusing on those aspects that are essential for orienting the team members’ discussions, in order to keep the models clear and the approach pragmatic:

- **Process map:** One wall is dedicated to the dynamic aspects of the business domain, i.e. the relevant parts of the business processes that the system shall support. It provides a visual overview of the system’s main functional requirements, and can thus serve as a reference for understanding the system and its context, making high-level design decisions, prioritizing artifacts, etc.
- **Data map:** Another wall is dedicated to the static aspects of the business domain, i.e. the relevant parts of the entities, documents etc. that the processes work on. These models should initially just visualize the business relationships so as not to anticipate design decisions; however, these relationships may evolve to more closely reflect the technical data structures as the project progresses.

Complex software projects are typically not implemented on a clean slate – rather, a new system is usually replacing a legacy system that is integrated in an existing enterprise software landscape. While the previous two maps focus on the vision of the system being built, the next two focus on particular challenges of brown-field software development:

- **Migration map:** Replacing a legacy system requires careful planning of the transition from the old data structures to the new ones. One wall of the interaction room is therefore dedicated to visualizing the legacy data structures already in place, and mapping out their transitions to the new system’s structures.

This involves determining entities, attributes and relations that should be added, converted, removed, restructured etc. – operations that require thorough understanding of the business and technical aspects of both systems, and can be discussed here side by side.

- **Integration map:** Another wall is dedicated to visualizing the application landscape that the new system will be integrated in – this includes other in-house or external systems and services, and their interfaces. The map does not aim to be a detailed interface specification, but a guide to the communication and coordination requirements that the new system will have to satisfy.

To intuitively create and manipulate these maps, we prefer the room’s walls to be furnished with large magnetic whiteboards to draw and stick model fragments on. It might seem tempting to augment the room with digital technology, and we certainly recognize that e.g. large interactive screens and tables promise considerable opportunities for integrating and visualizing complex project information, or even supporting collaboration between distributed teams – opportunities for pragmatic augmentation therefore also are a topic of our ongoing research. For most teams, however, we believe a non-digital room that can be easily set up without large investments into technology and training will already make a considerable contribution to more effective team communication in industrial development practice.

3. COMMUNICATION CATALYSTS

Outfitting the walls of the interaction room with model fragments is just the most basic prerequisite for fostering understanding, discussion and resolution of project issues with team members who may be from different domains, but should all be “on the same page.” In addition, we combine the maps with visualization techniques that directly address the typical risk factors we described earlier:

3.1 Joint Understanding of the System

Many software projects produce copious amounts of detailed specifications, some of which may describe trivial standard aspects, while others spell out important details of a system’s static and dynamic properties. In either case, the sheer volume of information makes it difficult for team members from different backgrounds to keep an overview of the whole system, have a firm grasp of the larger dependencies between features, components and data structures, and notice the important details and pitfalls among the trivia.

The **process, data, migration and integration maps** address this challenge by deliberately remaining incomplete, and instead showing only the most critical aspects of a system. This does not preclude business and technical experts from having more detailed, more complete models on their own desks, or temporarily bringing them into the room for discussion. The interaction room’s maps, however, should not be treated as a complete specification, but as an overview that provides orientation on the project’s goal, state, and interdependencies, and thus makes the more detailed, actual specifications accessible. For this purpose, we also do not prescribe particular modeling languages for the various maps, but let teams choose a notation that all stakeholders understand (we do recommend notations that provide constructs for hierarchical abstraction though).












Value annotations	Effort annotations
 exposed to high number of users	 requires high availability
 involves financial responsibility	 subject to time constraints
 affects company image	 resource-intensive
 executed frequently	 special storage requirements
 implements company policies	 subject to legal regulations
 other value- or effort-driving factors that are not immediately obvious from the model	

Table 1: Value and effort annotations for model elements

3.2 Identifying Value and Effort Drivers

As we have said earlier, the interaction room is all about focusing the team’s attention on those aspects of a system that are most crucial for the project’s success. To identify these aspects, we took inspiration from value-based software engineering [2], i.e. focusing resources on those features that add particular value to the system. In the interaction room, we encourage this approach by letting stakeholders annotate model elements with icons that indicate process segments or artifacts that merit particular consideration. Table 1 shows an overview of the icons we use (in the form of magnetic buttons to be stuck on whiteboards).

Value annotations are typically business-oriented and denote a particularly important element of the system – for example, an activity that performs calculations involving large amounts of money, a report that is sent out to thousands of customers, or a process that will reflect strongly on the company’s image. In addition, we use **effort annotations** as warning signs for system elements that can be expected to require more work than immediately meets the eye. Examples of such annotations include particular performance or storage requirements, the need for compliance with legal regulations, etc. While these annotations seem to be more relevant for technical stakeholders at first sight, they also help to point out effort drivers that may not be immediately obvious to business or management stakeholders, and thus foster better overall understanding of design decisions and effort estimates.

As the examples suggest, the value and effort annotations can be loosely associated with non-functional requirements, but are often more fuzzy. We therefore find it more intuitive for business and technical stakeholders to simply attach these “heads up” icons to models in the course of their discussion. Once in place, the annotations continually help to focus attention away from the trivia and onto those elements that will actually make or break the system.

4. INTERACTION ROOM IN PRACTICE

To illustrate how all the elements and techniques we have described work together in practice, Fig. 1 shows a process map from the example of a new claims processing component for an insurance company. The business process is sketched as a UML activity diagram on a relatively high abstraction level here, so business and technical experts can come to a joint understanding of the key claims management steps and the component to be built.

Here, the *appraise case including history* activity has been annotated with an “affects company image” icon, indicating that the details of the case appraisal procedure pose not just

a technical question, but involve more far-reaching customer relationship implications. Similarly, the *calculate insurance benefit* activity is annotated with an “involves financial responsibility” icon. While the details of this activity seem to be well understood already (as it has a UML sub-activity indicator referring to an existing detail specification), the annotation indicates that particular care should be taken in implementing and testing this step, as calculation errors could have considerable financial consequences. While the models on the walls of an interaction room cannot accommodate sufficient detail to work out the solutions to these warnings *in situ*, the annotations ensure that they are considered by developers as the models are refined further.

Note that the process map in the example also contains additional elements that were not described in detail here for the sake of brevity, such as cloud shapes to explicitly indicate uncertainty, and artifact printouts to visualize implementation progress and “solidity” of artifacts.

We are currently in the initial stages of applying the interaction room’s concepts in practice, through projects with a large software development company, an insurance company, and a large bank. Managers and developers have been very open to the idea in each case, expressing that the room addresses a real issue that has been troubling them. In the case of a complex migration project away from a legacy system, we already observed that applying just a subset of the visualization concepts relevant to migration, combined with guidance by an interaction room “coach”, yielded significantly better understanding between stakeholders on that team. The other projects are still in phases too early to evaluate.

5. RELATED WORK

Joint Application Design [5] and explicit documentation of design rationale [3] were notable earlier approaches to make different project stakeholders more aware of what they are building, and why. The concept of “team rooms” is also proposed in agile methodologies such as Scrum [9], however, apart from planning tools such as Kanban boards and burndown charts, their walls are typically not dedicated to visualizing particular aspects of the system. Although agile methods encourage interaction between stakeholders, they provide virtually no methods for focusing it on other aspects than progress. In fact, as Petre et al. observed, there are striking differences between how agile vs. traditional teams use their room walls to visualize system aspects and project progress [8]. One aim of our research is to integrate these perspectives on a project and its environment.

Facilitating interaction is also at the core of the computer-supported collaborative work (CSCW) community. Works

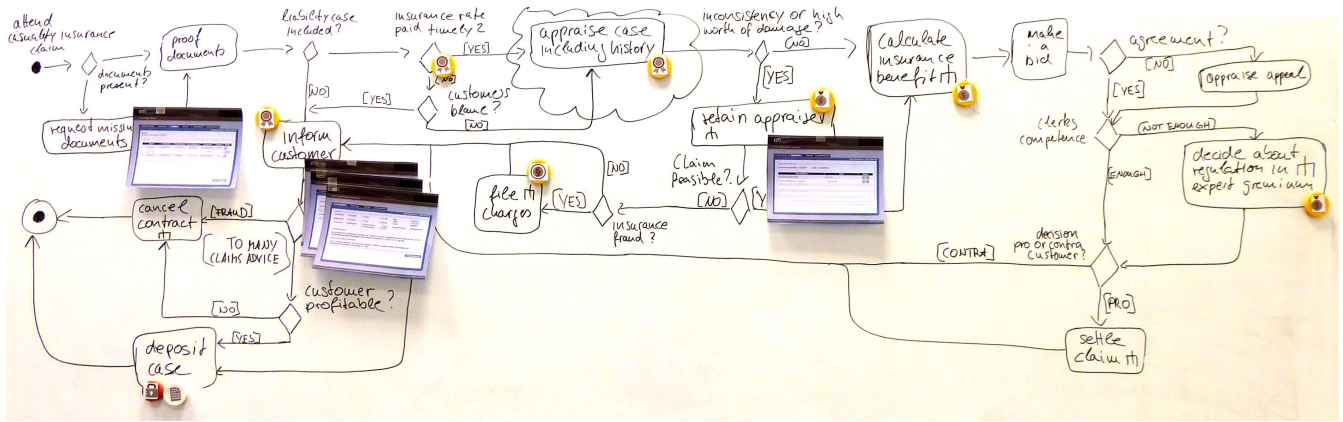


Figure 1: Process map for an insurance claims processing component

such as ReticularSpaces [1] provide a lot of enabling technology that we can build on when developing augmented interaction rooms. Most CSCW approaches however are domain-independent and thus provide only limited support for the concrete challenges of software engineering.

In the software engineering field, related works can also be found in the software visualization and program comprehension communities. However, most of these approaches tend to focus on existing software, work closer to the source code level, and are therefore more exclusively geared toward developers. In contrast, we strive to foster joint understanding of the overall system by technical and business stakeholders.

6. CONCLUSIONS

Although the walls of a typical interaction room are covered with models, the room is not intended as a replacement of existing modeling techniques or a tool for creating complete system specifications. Instead, the novel aspect of the interaction room is that it pragmatically uses sketches of system aspects to identify, discuss, understand and resolve critical issues that are often not made explicit or obvious in traditional specifications.

We believe that the main effects of such improved communication – understanding the business domain and its technical implementation better, and raising awareness of critical system aspects (in terms of value and effort) – can benefit business and technical stakeholders in traditional and agile software processes alike. Even if the underlying complexity of a system and its business domain can never be completely overcome, we expect that the use of an interaction room should help all team members to adopt a pragmatic, accessible perspective of their project that enables easier understanding of its details and better control of its risks.

Our ongoing research and evaluation will focus on how teams work with the different walls and elements of an interaction room, and which other project aspects should be visualized on its walls; how the room is integrated into different software process models, how different roles use the room, and if a dedicated facilitator is needed; if the room’s usage barrier is as low as we aspire to make it, especially for non-technical stakeholders; if there are measurable improvements in software quality and project velocity, and how these compare to the organization’s investment in setting up an interaction room for a particular team. Further re-

search will examine how digital augmentation can alleviate observed shortcomings of a physical interaction room, and if the necessary added investment is justifiable. We also believe many approaches dealing with human factors in software engineering could be anchored in an interaction room, and are looking for opportunities to examine this in practice.

7. REFERENCES

- [1] J. Bardram, S. Gueddana, S. Houben, and S. Nielsen. ReticularSpaces: activity-based computing support for physically distributed and collaborative smart spaces. In *Proc. 2012 ACM Conf. on Human Factors in Computing Systems*, CHI ’12, pages 2845–2854, New York, NY, USA, 2012. ACM.
- [2] B. Boehm and K. Sullivan. Software economics: A roadmap. In *Proc. Future of Softw. Eng., FOSE ’00*, pages 319–343. ACM, 2000.
- [3] J. Burge, J. Carroll, R. McCall, and I. Mistrík. *Rationale-Based Software Engineering*. Springer, 2008.
- [4] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Comm. ACM*, 31(11):1268–1287, Nov 1988.
- [5] E. Davidson. Joint application design (JAD) in practice. *Journal of Systems and Software*, 45(3):215–223, Mar 1999.
- [6] K. Kautz, S. Madsen, and J. Nørhbjerg. Persistent problems and practices in information systems development. *Information Systems Journal*, 17(3):217–239, Jun 2007.
- [7] C. Larman and V. Basili. Iterative and incremental developments – a brief history. *IEEE Computer*, 36(6):47–56, Jun 2003.
- [8] M. Petre, H. Sharp, and S. Freudenberg. The mystery of the writing that isn’t on the wall: Differences in public representations in traditional and agile software development. In *Proc. 5th Intl. Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE ’12. ACM, 2012.
- [9] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*. Prentice Hall, 2002.
- [10] J. Whitehead. Collaboration in software engineering: A roadmap. In *Proc. Future of Softw. Eng., FOSE ’07*, pages 214–225. IEEE Computer Society, 2007.