Increasing the Quality of Use Case Definition Through a Design Thinking Collaborative Method and an Alternative Hybrid Documentation Style

Alexandra Matz(✉) and Panagiotis Germanakos

User Experience, Products & Innovation ICD, SAP SE,
Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
{alexandra.matz,panagiotis.germanakos}@sap.com

Abstract. Use cases are a critical milestone in the User Centered Design process referring to a list of action steps that define an interaction between two entities sharing a common goal. They express a structural representation of a usage scenario aiming to generate highly usable prototypes and user interfaces of a system or application. However, today, use cases are often not sufficiently created and documented, or they are not in the expected quality. The lack of a step-by-step collaborative approach towards the composition of more inclusive and readable use cases create unnecessary iterations increasing the cost, time, and resources utilization in an organization. Hence, in this paper we propose a Design Thinking collaborative method based on an alternative hybrid use case documentation style that enhances active participation and learning across team members. We emphasize on the methodology and tool and we present the benefits as those extracted from real-life business scenarios.

Keywords: User centered design · User experience · Collaborative learning · Use cases · Design thinking · User interfaces · User research

1 Introduction

Use cases are considered one of the most important tools of user research and user experience design since they encapsulate a crucial activity step in the User Centered Design (UCD) process, which guides the main philosophy and strategy of most companies today during the development of their business applications and systems. Use cases, in more practical terms, show the aim and the subsequent objectives of a system and the assigned user roles (also called actors), by expressing a list of steps and interactions among them towards a common goal. They could be written in a textual form or represented with the use of flow charts, sequence charts, Petri nets, or programming languages [1, 2]. In order for use cases to be effective and consequently convey the expected outcomes during the design and development of software solutions, they should be readable and understandable by all project stakeholders, sponsors and the end-users. Usually, their development is taking place after the creation of storyboards and before the actual design of the mock-ups and/or prototypes. It is the

P. Zaphiris and A. Ioannou (Eds.): LCT 2016, LNCS 9753, pp. 48–59, 2016.
DOI: 10.1007/978-3-319-39483-1_5

step that more abstract and fuzzy descriptions of the activities and tasks are transformed into tangible interactions between the involved entities (user roles and system). The more precise and inclusive a use case is the higher the probability to develop more qualitative user interfaces and system designs that will increase the user experience of the end-users.

However, one of the biggest problems nowadays in large organizations is that often project teams, that usually consist of different roles with diverse backgrounds and perspectives, miss to define use cases correctly (if not omit them) given the tight time delivery schedules and the lack of: (a) a consistent collaborative methodology that could put the various suggested use case styles into practice, enabling continuous learning and generating cumulative knowledge of real-life situations, and (b) a use case documentation tool that could ensure the same level of understanding and acceptability by all. Empirical research has shown that working and learning in highly synergetic collaborative environments [3] increase engagement, active participation, creativity, responsibility and awareness across project team members leading to deeper learning and more sophisticated and innovative solutions to real-world problems.

Thus, in this paper we outline a use case framework namely *Usee*, emphasizing on the methodology for creating use cases applying Design Thinking (DT – [4]) and facilitating collaboration, empathy, and integrative thinking and learning among project team members in large organizations during the software development process; and an alternative definition of a hybrid (diagrammatic and textual) use cases documentation style that bridges possible knowledge gaps and enhances understanding of their added value and use.

2 Related Work and Challenges in Large Organizations

The construction and documentation of use cases as an approach for identifying and capturing more inclusively the functional and behavioral requirements for the development of software systems is not new. It is also referred to as use case driven development (that could also be suitably aligned with the agile development methodology [5]), and has been gradually widely adopted, with the necessary modifications and alignments, to the business and process models of many large organizations. Jacobson et al. [6] was the first back in 1986 that created a number of techniques (textual, structural and visual modeling) for specifying and analyzing use cases, in an attempt to more comprehensively capture software requirements of large-scaled systems. Since then, many researchers, mostly from the area of software and systems engineering, approached adequately the topic suggesting different methods for developing and documenting use cases. Even though minor or significant variations in their viewpoints can be identified with respect to the style of presentation or the formulation of the content, most of them agree that, broadly speaking, there is not a standard way to create a use case, as there are no universally defined components or structures that could satisfy all the needs through a unique representation of the various parts it consists of (apart from some distinct items used across the various recommendations such as the actors, purpose or goals, preconditions and interactions). Yet, the length, complexity and the detail that a use case could be described in is guided predominantly from the case or the

situation under investigation, the specific characteristics or its surrounding contextual factors. Fowler and Scott presented in [7] various use cases, class diagrams and interaction models using the UML language distinguishing between the title, main success scenario and extensions (an outcome condition of the various interactions derived from the main success scenario) as the body of a use case. Cockburn, in one of his highly used textbooks [2], maintained a more flexible open approach (one column of text) in the writing of use cases and separated those that need to be described at higher level (casual) from those that need to be detailed more extensively (fully dressed). For Cockburn sometimes even a more simple structure of a use case composed only from the primary actor, scope, level and the story (in a narrative format describing the situation) might be sufficient for the needs of a project. He suggested a number of symbols to graphically indicate the subsequent levels of a use case ranging from the very high summary, summary, user goal (preferred level aka "goal level" or "sea level"), to sub-function and the too low level. A variation of this style could be considered the one-column table or the two-column table, which, even though extensively used today in the business sector, someone could argue that the within lines might hinder the flow of the actual writing [2]. In general, the purely textual layouts have the disadvantage that they maintain a serial or column approach with no clear (visual) distinction of the actors or the flow of information. This could be proved overwhelming especially in big and complex scenarios (though a useful rule of thumb is that a use case should not be in total more than nine action steps). Other use case styles are the Rational Unified Process (RUP – [8]), IF-Statement style, Occam style, and the Diagram style as nicely outlined by Cockburn in his book [2]. Finally, Constantine and Lockwood introduced the essential use case (aka business use case), where they used a structured narrative (conversation format), for capturing the user interface requirements and the purpose or the intentions that influence an interaction [9]. Therefore, there are various ways for creating use cases like in a diagrammatic format (using e.g. the UML language), in textual format (e.g. in a table), or using index cards (as introduced by Beck and Cunningham [10] in [11]).

However, most of these approaches refer to specific roles, as software engineers or developers, who have more technical skills, knowledge and training and embrace a particular way of thinking driven by their expertise. Nowadays, large-scaled software development project teams consist of heterogeneous roles (such as product owners and managers, architects, developers, user researchers, interaction and visual designers, etc.), tackling the real-life problems with innovative solutions usually found in the boundaries of their expertise. Hence, in order to achieve a successful and effective development of use cases, we must first face an existing challenge, especially in large organizations, of how to develop a method that will be able to bridge the knowledge gaps, experiences, educational backgrounds, and business roles of the various stakeholders. For such a method to be successful, it should share attributes, notations and semantics that will enable an active participation and learning through their continuous interaction, while participants are located in the same physical space or collaborate through virtual environments. Main concern is to maintain a common level of understanding and breadth and depth of analysis to the benefit of all. This way teams will be able to take advantage of the diversified capabilities and expertise of its members towards their common goal that is the creation of seamless, functional and highly usable user interfaces and systems. More specifically, we have currently identified, to the best

of our knowledge, a number of problems, among others, that hinder the smooth development and understanding of use cases, and which can be broadly distinguished as the:

- *Lack of a consistent collaborative methodology for creating use cases*. Even though there are noteworthy guidelines in the literature, as briefly discussed above, describing the various parts and components of use cases, there is not a consistent methodology showing how these could be applied by a project's team members in a consistent and collaborative manner. Inevitably, this creates different perspectives, lack of common understanding and acceptability of their potential benefit. Furthermore, ad-hoc or different approaches are utilized across teams resulting to not having a highly synergetic development process and a homogeneous outcome that could be cross-validated. Consequently, in many cases the same use cases (often belonging to different user roles) are re-defined driven by the different scope and viewpoints of various project teams, and leading to redundancies and/or incomparable results.

  In addition, the existence of such a collaborative methodology could increase the monitoring and support during the development process ensuring the expected qualitative definition of use cases. In particular, the lack of a subsequent step-by-step approach towards turning/interpreting more abstract statements (i.e. found in the previous process step, namely "Storyboards") to more deterministic/specific ones that serve as the basis of interaction design and functionality create unnecessary repetitions increasing the cost and time consumption. For example, many times we observed team members to use statements as, a user 'thinks', 'considers', 'evaluates', etc., instead of 'reports', 'chooses', 'deletes', etc., that could move the interaction process forward and could clearly indicate "who has the ball" each time in an interaction.
- *Lack of a hybrid (diagrammatic and textual) documentation style*. Currently, apart from the more "technical" UML based graphical styles, most project team members use pure textual or tabular style formats for the creation/documentation of their use cases. Inevitably, this creates confusion to those that follow the UCD approach, since it presupposes a totally different way of thinking than the one they used to have until this point in the process. The transition from a graphical high-level description method (as is Storyboards) to a pure text-based specific statements write-up creates inconsistencies and mental gaps to the participants and difficulties of adopting their way of thinking to this style. Also, this tabular format does not facilitate ease of use interaction and quick cross check validation among use cases since they are composed of pure text that masks the relevant information overwhelming the team members during the analysis (or consolidation) phase. For example, it is not easy to compare their length or their level of detail. Furthermore, there is no distinction between the user role and the system (they both belong under one column), and also there is no provision for marking possible repetitions among two activities or interaction points between two entities, that could be measurable and give added-value to one interaction step (i.e. the relationship of the two entities at a particular stage of the process). Finally, there are no interaction lines indicating the flow of information as traditionally most of data modelling tools use for the representation of information.

3 The *Usee* Collaborative Framework

In response to the above challenges and concerns, we outline in this section a use cases framework namely *Usee*, that aims to provide the basis for an end-to-end process of developing, validating and sharing qualitative use cases among projects' stakeholders. It provides components and tools that facilitate an effective collaboration, proactive support, knowledge sharing and learnability to the various transdisciplinary teams that participate in large-scaled projects and have the same objective; to increase the usability and user experience of user interfaces, applications and systems to the benefit of the end-users. The framework comprises of three main entities:

(a) *The User Interface*. It enables users to: (i) manage use cases (create new, review and modify existing), (ii) manage personas, (iii) maintain rules and messages which will facilitate the syntactical and semantical validations of use case data, and (iv) research, analyze and compare existing use cases through statistical methods and tools for their level of similarity, complexity, semantic quality, etc.
(b) *The Cloud Application Server*. It consists of four plus one components: (i) the use case maintenance, where the creation (providing two alternative ways: step-by-step and/or Graphical (WYSIWYG)), maintenance and storage of use cases data are handled; (ii) the use case administration, where the semantical and syntactical algorithms as well as the validation rules and messages are handled; (iii) the use cases and personas analysis and research, where various statistical models and comparison algorithms are executed, (iv) the persona maintenance, where the viewing and creation of personas' data are facilitated; and also (v) the cloud system administration, accommodating system and user administration.
(c) *The Cloud Usee Collaboration Network*. It is composed of three elements: (i) and (ii) refer to the use case and persona collaboration and discussion forum, where project teams can publish/share their use cases and personas and discuss related questions and issues, and (iii) the open source use case rule framework, where use case rules and algorithms can be shared, extended and modified by the network community, and which can be downloaded to the *Usee* application server.

Furthermore, the framework it will provide the possibility to upload use cases and personas to the SAP User Experience Communities (https://experience.sap.com and https://www.experiencesplash.com), in order to run usability validation tests with customers and end-users. Finally, there will be the provision of inviting customers for these usability validation sessions through the SAP Customer Engagement (CEI) initiative.

A fundamental prerequisite for the generation of qualitative use cases using the *Usee* framework is their comprehensive composition in a highly synergetic collaborative manner and their inclusive documentation in a style that will be understandable and purposeful for all the participating members of a project team. Accordingly, for the purpose of this paper, at first, we suggest a Design Thinking-based methodological approach liable to increase teams' learnability and efficiency towards the development of their use cases while at the same time reducing unnecessary time consuming and costly iterations. At a second level, we describe the main parts, structure and attributes of

a new re-designed alternative hybrid (diagrammatic and textual) form for documenting use cases (as this will be extracted by the *Usee*), that increases customization, readability, comparability and enhances flexibility of use.

3.1 A Use Case Collaborative Methodology Using an Alternative Hybrid Style

As stated above the suggested collaborative methodology for the composition of use cases is based on the Design Thinking approach which is widely used in large organizations. DT originates as far back as the late 1980s and early 1990s when the need for a different, creative as well as innovative resolution of challenges emerged in the fields of engineering and architecture [12]. In DT, the approaches and problem solving methods of designers are merged with the viewpoints and practices of technology and business. As a result, it has evolved into a discipline that helps to strategically design and create products that satisfy the needs of users while at the same time opening up new business opportunities for companies [13]. DT advocates an open mindset and constant learning by observing and thoroughly understanding a problem space. Using the different skills and backgrounds of a multi-functional Design Thinking team, this approach helps to uncover new ideas through sharing of insights and by building on the ideas of others. During the entire process, keeping the open mindset remains important. If an idea or a method does not suffice, teams are prepared to not stick to one path, but re-iterate and even throw away ideas and start-over. To fail early in the process, and to fail often is encouraged as it allows constant learning through iterating on ideas as well as prototypes and helps to create innovative and at the same time solid and validated solutions.

Hence, this collaboration method apart from increasing the engagement, intrinsic motivation, empathy, learnability, and bridging diverse perspectives and backgrounds as those dictated by the various business roles, adheres to a number of teamwork dynamics as is: the facilitation of active and meaningful participation towards achieving the shared goals defined by the case at hand; the team-oriented approach to creativity, problem-solving and decision making; recognition of variable communication patterns; optimization of interpersonal trust and information processing; experiential skills development to effectively engage the main challenges teams usually face, like conflict resolution; etc. [14]. The biggest asset though derived from this constructive collaboration is the cumulative knowledge for a specific business situation that can be reproducible and retrofitted back to the business case at hand or to other teams dealing with similar cases saving effort, possible redundancies and resources. This is achieved through the creation of open ended opportunities (e.g. extensive engagement in open ended tasks) for the participants who have to construct their own learning objects and flexible knowledge through active experimentation, observation, reflection and conceptualization of their experience (two well-known approaches that support this method are Kolb's Experiential Learning Model [15] and the Problem-Based Learning [16]).

However, a necessary condition for a use case to be successful while teams are working in a DT collaborative mode is to perform continuous iterations and validation based on the situation-specific objectives and the main success scenario that they have

decided upon at the very beginning of the process. The material and information that should be available at this stage combine the analysis and outcomes from the end-user research, conducted earlier through interviews, observations, field studies, etc., during the requirements collection phase. Typically, these deliverables include the personas, user story maps, activity flows, task analysis, storyboards, etc., that will determine the content and will guide the information flow and action steps of a use case. Once those are sufficiently documented then the team can proceed to the creation of the first use case which will in turn sketch the grounds, in the next step, for the interaction and visual design of the user interface, before the actual system integration and development starts (see Fig. 1).

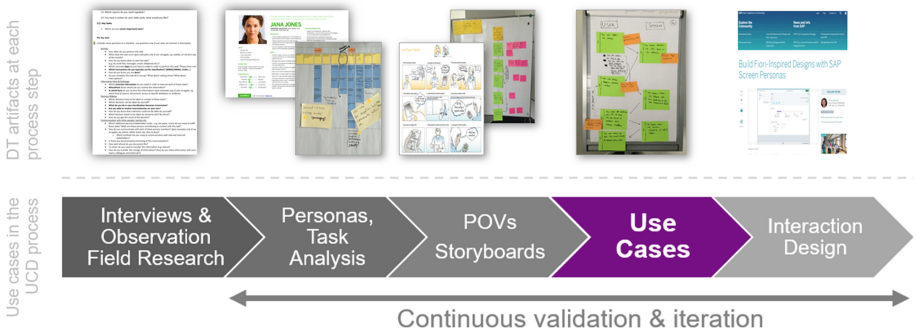

Fig. 1 The use cases in the UCD process and related DT artifacts

We hereafter describe the suggested components of the alternative hybrid style that teams could employ for creating their use cases (see Fig. 2b). For a better understanding of its realization in a real business situation we give next to each attribute an example based on a hypothetical scenario related to the 'Resolve Leave Request Conflicts' case. Structure-wise the proposed layout is composed of two main parts:

(a) The more *static part* which contains the following data:
   i. **Use Case ID** – used for cross reference and linking with the corresponding persona (*e.g. UC0001*);
   ii. **Use Case Name** – short use case name (*e.g. Resolve Leave Requests Conflicts*);
   iii. **Primary Role** – the user role who uses the system to fulfill a specific goal (*e.g. Manager – Linda*);
   iv. **Secondary Role** (optional) – the user role who receives information from the system, but is not the primary user (*e.g. Employee*);
   v. **Use Case Goal** (or Point of View (POV) – it describes the user's goal; what does (s)he wants to achieve; User + Need + Insight (*e.g. Linda needs a way to review the leave requests of her team in order to ensure no absence conflicts*);
   vi. **Background** – includes short description in free format about the scenario and any necessary assumptions (*e.g. Linda has a large team and often she*

receives many leave requests. Due to the nature of the work many times she comes across with leave request conflicts. Therefore, prior to the approval of a leave request she needs to see whether any leaves overlap exist);

vii. **Pre-conditions** – it describes the prerequisites which should be true before the use case can start (*e.g. Access to the system and authorization to view and approve her team's requests*); and

viii. **Trigger** – it identifies the action or event(s) that gets the use case started (*e.g. Linda received a leave request from an employee and opened the notification*).

(b) The more *dynamic part* which contains entity boxes (representing the user and the system), interaction lines, alternative paths, failure cases, and repetition notations. More specifically:

i. An **entity box** is broken down into three sections: (a) the first (left) section indicates the **number of the activity** (*e.g. 3*), (b) the second (middle) section describes the **action steps** that needs to be undertaken by the user to achieve a goal using the main path (*e.g. The system provides details about the conflicting leaves and graphically indicates the overlapping days*), and (c) the third (right) section describes the **user interaction data** and data sources (internal or external) needed to complete the action steps (*e.g. Employee names, leave dates, date timeline, conflict indicator, etc.*);

ii. The interaction lines describe the **interaction points** (numbered as Ia (Interaction a), Ib, Ic, etc.), expressing the relationships among the user and the system. They are the interception points among two actions and show how the interactions are situated in contexts of use. Each interaction point shows the reaction of the system on a required action from the user and vice-versa. Interaction points can be used for various reasons such as further analysis of signifying the path and the length of a use case, comparison among the same or similar use cases belonging to different applications and/or roles, point of reference on complex and iterative (i.e. loops) use cases, isolation and in depth analysis of a particular relationship (for example teams can use statistical models and methods to apply a more detailed quantitative and/or qualitative analysis with respect to i.e. effectiveness, efficiency, of an interaction among two actors), etc.;

iii. The **alternative paths** are ways in which the main success case can succeed, and they are placed under the entity box they are referring to, denoted by the number of activity, followed by the number of the alternative path, followed by a small description (in our case, *e.g. 3a. The system proposes how to resolve this conflict, i.e. it considers the employee's tasks, remaining days, etc. and suggests an optimized leave period*);

iv. The **failure cases** describe possible ways in which the main success case can fail and they are denoted by the letter *F*, followed by the number of activity, followed by the number of the failure case, followed by a small description (*e.g. F3a. The suggested days are outside the period the Employee wants to go on vacation*); and

v. Lastly, a dotted line connecting two or more entity boxes describes possible **repetition(s)** of specific action steps (*e.g. Action steps 2 to 4 might need to be iterated*).

At the very end of the suggested style there is another more static part, called **Annotations**, where teams can take note of clarifications, references, information that are noteworthy, need to be re-visited or to be considered in the future pertinent to the specific use case or activity steps (e.g. in action step 3 Linda has to decide whether she will accept the suggestion by the system or she will try manually to find a solution to the problem.)

3.2 Use Cases Documentation, Generation and Storage

The *Usee* application, implemented in both mobile and desktop technological environments, will provide a unique tool to compose and document use cases (see Fig. 2a) generated in the hybrid use cases style described above. The documentation will follow a fully controlled cloud-based creation process providing the necessary step-by-step guidance, support and storage. More specifically, this application will add value to project team members in various ways, such as: (a) initially, while users entering the data of a use case (already worked out in the DT collaborative mode) through the smart user interface, dedicated validation mechanisms will check for similarities e.g. of roles entered or goals. In case of potential similarity, the user will be prompted and enabled to review the existing use cases and the related information. This way we will increase transparency across development units and will avoid producing duplicate or similar use cases for a similar or the same role or user activity; (b) it will increase learnability by providing immediate feedback through suggestions and validations of the inserted text based on rules which will ensure the quality of the content, e.g. instead of phrases like, a user 'thinks' or 'considers', the application will propose 'reports' or 'chooses'; and (c) it will inform users in real-time about the strength and the quality of their use case, based on the interaction steps and flow, etc. Currently, the *Usee* application is in the development phase. However, apart from the DT collaborative methodology, a

Fig. 2 (a) The *Usee* wireframes and (b) the generated alternative hybrid use cases style

visually enriched Microsoft Excel-based template consisting of the components detailed earlier is available and used by project teams with encouraging results and feedback as we discuss below.

4 Benefits Derived from Usability Tests in Real-Life Scenarios

During the last two years the proposed use cases methodology and style have been extensively validated in numerous workshops internally in SAP with product development teams and externally with various co-innovation customers, accommodating positive feedback and acceptability. More specifically, the meta-analysis of the observations, interviews and focus groups yielded a number of benefits for the teams and the quality of their products' design and development. The added value of the suggested approach can be summarized as follows:

(a) *Enhanced flexibility and adaptability*. It is based upon the well-established DT method, where most project teams are familiar with and trained. A team's members need to continue to work together in a flexible and iterative manner, following the described use cases guidelines for their development. In this context, they can (re)define entities, shuffle the action steps and the interaction flow until they reach to an acceptable qualitative version. Furthermore, they can all actively participate and brainstorm, embracing the same understanding irrespective of their different knowledge backgrounds.
(b) *Speed up processing*. It maintains the same level of approach to the benefit of a project's team members. This means, there is no need to switch their working context i.e. jumping from a more visual thinking (e.g. storyboards), to pure tabular and textual format.
(c) *Ease of use*. It is more concentrated and presents a balanced visual and textual representation of data, showing clearly the flow of all the pertinent information with respect to one activity (as most data flow diagrams), leveraging the cognitive overload of team members.
(d) *Increased clarity*. There is a clear visual distinction between the user and the system, and their respective activities.
(e) *Ease of customization*. It provides an easy of use and quick comparison among two or more use cases, allowing quick adjustments irrespective of the various contexts of use. It also enables the rapid association with the elements of a persona (where a user role is detailed, i.e. tasks, needs, requirements), and of transparent identification of overlaps (i.e. same use case different roles or applications, or same role/use case but different applications).
(f) *New functionality*. Except of the repetition dotted line which indicates a possible loop among two activities, the suggested interaction points are revealed from the interaction/activity of two entities. A team's members can further compare or analyze (i.e. by isolating and applying an in depth analysis of a particular/key relationship of two entities) or can use them as point of reference on complex and iterative use cases.

5 Conclusions

Nowadays, use cases are increasingly used by organizations to document their business processes, to detect behavioral system requirements and identify how it reacts in different conditions. A use case primarily shows the interaction between a user and a system when (s)he tries to accomplish a goal given a specific business scenario. Even though use cases refer to a critical step of the UCD process ensuring the usability and acceptability of user interfaces and interaction designs of a system, quite often teams do not compose them sufficiently or even miss to address them. This happens on hand due to the tight schedule delivery constraints or limitation of resources and on the other hand due to the lack of a guided collaborative approach that will increase engagement, active involvement, learnability, or simply will motivate them to pursue them. In addition, it seems that the divergent business roles, backgrounds, experiences and skills of individuals that usually participate in the UCD product development process in large organizations cannot fully take advantage of the existing use cases documentation styles and tools since they fail to facilitate the same level of understanding; creating many times confusion, disorientation or overwhelming the team members.

In light of the abovementioned challenges, in this paper we have proposed a DT collaborative method for creating use cases as well as a new alternative hybrid use case style for their documentation in the context of the *Usee* framework. The main aim is to facilitate the inclusive and qualitative generation of comparable use cases through continuous learning and active participation of all stakeholders that share the same goals. From the usability tests and validations we conducted so far, in various business scenarios and with different size and kinds of business roles and teams, we could conclude that there is a positive tendency, acceptability, and satisfaction towards the utilization of the suggested method and tool. This is really encouraging for the future of this work since it could increase the business value of companies by facilitating a more sound communication among the various stakeholders and enhance clarity and understanding of their business cases. Future work includes the release of the *Usee* application as a final product and the further validation of the proposed solution in different business settings and more complex business scenarios in an attempt to further enhance and optimize its use.

Acknowledgements. We would like to thank our colleagues Mandana Samii and Timo Bess for the visual enhancements of the proposed use case definition style, our managers Matthias Berger and Joerg Roesbach for their support and allocation of resources as well as all the product organizations, teams (especially the Suite Engineering UX team), customers and individuals in SAP, who have participated in the usability tests and provided their constructive comments and valuable suggestions.

References

1. Wiegers, K.: Software Requirements, 2nd edn. Microsoft Press, Redmond (2003). ISBN 10:0735618798
2. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, Boston (2001). ISBN 0-201-70225-8

3. Barkley, E.F., Cross, K.P., Major, C.H.: Collaborative Learning Techniques, A Handbook for College Faculty. Wiley, Hoboken (2014)
4. Plattner, H., Meinel, C., Leifer, L. (eds.): Design Thinking: Understand – Improve – Apply. Springer Science & Business Media, Berlin (2011). ISBN 10:364226638X
5. Cohn, M.: Succeeding with Agile: Software Development Using Scrum. Addison-Wesley Professional, Boston (2009). ISBN 10:0321579364
6. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G.: Object-Oriented Software Engineering – A Use Case Driven Approach, 1st edn. Addison-Wesley Professional, Boston (1992). ISBN 978-0-201-54435-0
7. Fowler, M., Scott, K.: UML Distilled – A Brief Guide to the Standard Object Modeling Language, 2nd edn. Addison Wesley, Boston (1999). ISBN 0-201-65783-X
8. Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley Professional, Boston (2004). ISBN 0-321-19770-4
9. Constantine, L.L., Lockwood, A.D.L.: Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. ACM Press/Addison-Wesley, New York (1999). ISBN 0-201-92478-1
10. Beck, K., Cunningham, W.: A laboratory for teaching object oriented thinking. ACM Sigplan Not. **24**(10), 1–6 (1989)
11. Ambler, S.W.: The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge University Press, Cambridge (2004)
12. Rowe, G.P.: Design Thinking. The MIT Press, Cambridge (1987). ISBN 978-0-262-68067-7
13. Brown, T.: Harvard Business Review, pp. 84–92, June 2008
14. Levi, D.: Group Dynamics for Teams. Sage Publications, Thousand Oaks (2013). ISBN 10:1412999537
15. Kolb, D.: Experiential Learning as the Science of Learning and Development. Prentice Hall, Englewood Cliffs (1984)
16. Hmelo-Silver, C.E.: Problem-based learning: what and how do students learn? Educ. Psychol. Rev. **16**(3), 235–266 (2004)