A Method for Requirements Capture and Specification Based on Disciplined Use Cases and Screen Mockups

Gianna Reggio, Maurizio Leotta(✉), and Filippo Ricca

DIBRIS, Università di Genova, Genoa, Italy
{gianna.reggio,maurizio.leotta,filippo.ricca}@unige.it

Abstract. We present a novel method for capturing and writing requirements specifications that enriches disciplined use cases with screen mockups. Disciplined use cases are characterized by a quite stringent template, which allows to impose a large number of constraints helping to prevent common mistakes and to increase the quality of the specifications. Disciplined use cases are expressed using natural language, but the strong structuring allows to reach a good level of precision without having to introduce new (and more complex) notations. Screen mockups associated with the steps of the scenarios present the corresponding GUIs as seen by the human actors before/after the step executions, improving the comprehension of the requirements, and allowing also to precisely present non-functional requirements of the user interface. The method has been successfully applied in an industrial project and the effectiveness of the screen mockups has been validated by means of controlled experiments.

1 Introduction

A number of methods, techniques and approaches have been proposed in literature for representing software requirements. Among them, use cases are a widely used technique to specify the purpose of a software system, and to produce its description in terms of interactions between actors and the subject system [5]. However, even presented by means of use cases, the requirements could be difficult to comprehend, incomplete, inconsistent, contradictory and may cause/provoke defects in the software system under development.

Screen mockups (also known as user interface sketches, user interface prototypes or wireframes) are used for prototyping the user interface of a subject system [7,9]. Mockups can be used in conjunction with use cases, associating them with the steps of the scenarios, to improve the comprehension of functional requirements and to achieve a shared understanding on them. Simultaneously, they allow to represent the non-functional requirements concerning the user interface [6]. However, *enriching the use cases with the screen mockups arises the problem of guaranteeing that the mockups are coherent with the textual part of the use cases*, and that they truly provide information on how to structure the GUIs supporting the functionalities presented by the use case.

P. Abrahamsson et al. (Eds.): PROFES 2015, LNCS 9459, pp. 105–113, 2015.
DOI: 10.1007/978-3-319-26844-6_8

As a matter of fact, the consistency between use cases and screen mockups cannot be guaranteed if the former are poorly structured and have a low level of *precision*. Requirements specifications based on use cases may have very different levels of precision from scarcely structured scenarios made by lists of freely formed natural language sentences, to use cases presented following quite detailed and structured templates, for example Cockburn [5], till to methods where the use cases are represented by means UML models [2] or even by formal specifications [4]. We believe that disciplined natural language specifications, i.e. where the text must follow very detailed and stringent patterns [8,10] and a glossary of terms is added to reduce ambiguity, are a good compromise. For this reason, we have conceived a method for capturing and describing requirements specifications based on *disciplined use cases* and screen mockups, taking the Cockburn guidelines as starting point. *Disciplined use cases* are: (1) characterized by a high level of precision without having to introduce additional notations and the consequent effort required to learn and to use them, (2) suitable to be enriched in a consistent way with the screen mockups, and (3) able to help the requirements analyst to detect errors, incompleteness, bad smells (e.g. unused elements), and bad quality factors (e.g. too many extensions and too many steps in a scenario) in the requirements specification (thanks to many well-formedness constraints).

Even if Screen Mockups are quite common in many industries and several proposals are emerging to integrate/use them in conjunction with use cases (or more in general with requirements) [15,17], our method is the only one that allows to obtain consistent screen mockups that are fully integrated in the development process, and thus that are not just a bunch of drawings added to the use cases. To the best of our knowledge, this is the main novelty aspect of our work.

This paper extends our previous work [11], where we described the template of disciplined use cases and how to integrate screen mockups, with: (i) the description of a comprehensive method for capturing and writing requirements specifications (see Sect. 2), (ii) some details on the empirical assessment of the method (see Sect. 3), and (iii) the complete list of well-formedness constraints for the precise use cases integrated with screen mockups (see Appendix A).

2 Requirements Capture and Specification Using Disciplined Use Cases and Screen Mockups

The starting point of our method (see Fig. 1) is what we call a "free use case specification", i.e. a use case specification based on whatever template, in general allowing a lot of freedom to the specification writers. In case not yet available, the free specification may be easily produced by stakeholders or domain experts with or without the assistance of the analyst.

Once the free specification has reached a stable form, the analyst may start the task of making the use case disciplined and adding the screen mockups (i.e., as described in [11], and verifying that the well-formedness constraints shown in Appendix A hold). Obviously, the result of such activity is the detection of

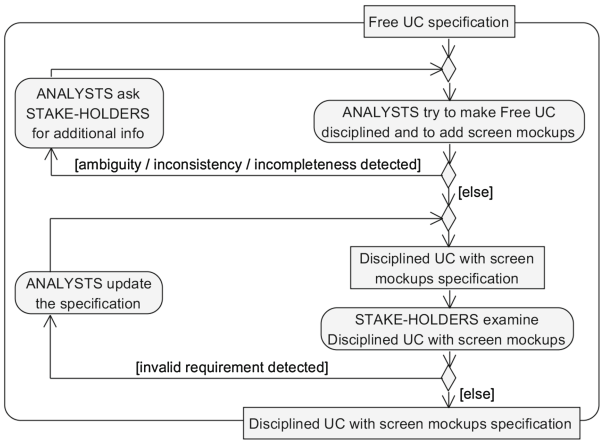

Fig. 1. Requirements capture and specification using disciplined use cases and screen mockups

inconsistencies (i.e. two different points of the specification express two contrasting statements about something), *ambiguities* (i.e. the specification uses words without stating their precise meaning relying on some common, but not always shared, understanding), and *incompleteness* (i.e. it is not possible to understand how the system should work in some specific cases). In these cases the analyst should ask the stakeholders and/or the domain experts additional information to be able to obtain a disciplined use case specification equipped with screen mockups; notice that also adding the screen mockups may generate many questions about the system under specification.

Once the analyst has terminated her/his work, the resulting disciplined specification augmented with screen mockups may be given to the stakeholders to get the final approving. They have no problem in reading and understanding it, since it is essentially structured natural language text; moreover the presence of the screen mockups offers a kind of paper prototyping, allowing them to validate also the user interface. Any change request may be easily processed by the analyst because the strong structuring of our use case specifications offers a good support to trace the influence on the whole specification of a change. This characteristic of our specifications will be valuable also on the case of future evolution of the requirements on the system.

3 Empirical Assessment

In this section we describe (1) how our method has been fine-tuned during several editions of a software engineering course at the University of Genoa [1], and (2) how we evaluated its applicability in the industry through a case study.

Moreover, we evaluated by means of a series of controlled experiments: (1) the effectiveness of screen mockups in improving the comprehension of functional requirements [12,14], and (2) the effort required to build the screen mockups [13,16].

3.1 Students Projects

The proposed method has been used during several student projects in several editions of the software engineering courses in the last decade at the University of Genoa, where two of the authors of the present paper were teaching. Each year, students had to realize a Java desktop application whose requirements, produced by ourselves, were given as a use case based specification. First, students had to model a design by means of UML, and then, they had to implement it in Java [1]. Initially, no screen mockups were used, and even if standard requirements on the GUIs were provided (i.e. usability requirements), the use cases often resulted difficult to understand and ambiguous for what concerns the user interaction. For example, a use case step for EasyCooking — a software helping to write recipes and diets — requiring "to list foods" was interpreted by all the students as "to list the complete nutritional details of foods" while the intention of the requirements authors was simply to list the names of such foods. We discovered that, after the introduction of screen mockups the number of misunderstanding about the required user interactions decreased, and moreover, we witnessed that the effort of producing the screen mockups had as a "side effect" that various ambiguities/incompleteness even mistakes were discovered during the writing of the requirements specification. Some examples of the disciplined requirements specification with screen mockups for the students projects may be found at http://sepl.dibris.unige.it/2015-UseCasesMockups.php.

3.2 Industrial Case Study

Our method has been applied with success during a joint project [3] involving the University of Genoa, Italy, and two local companies, having the goal of developing the EC2M system. Such system consists in an improved ECM making use of ontologies to better classify, retrieve and share documentation among different branches of the companies. The functionalities offered by EC2M can be classified as: interactive and non-interactive: the first allow the user(s) to interact with EC2M using GUIs (e.g. logging in and inserting/retrieving documents) while the latter focus on the interactions between software systems using specific protocols (e.g. exchanging information or documentation using SOAP and REST). Since our method has been devised for describing the requirements specification of interactive software systems, in this project we applied our method only for the portion of requirements describing the interaction between the EC2M system and the user(s).

Creating the Free UC Specification. As described in Sect. 2 our method takes a free UC specification as input. Using the information gained in the course of the first two meetings with our industrial partners, we developed a preliminary version of the UC specification, composed by 12 use cases and three primary actors. The free UC specification reflects the requirements as informally expressed by the industrial partners (i.e. the two local companies) during the meetings. For this reasons, later, we discovered that there were several problems, for instance: (1) the meaning of the terms reported in the use cases was not always agreed by all the partners (the glossary is missing in this phase), (2) the granularity of the actions described in the use cases steps was not uniform (some too abstract and others too detailed), and (3) only a few extensions to the main success scenarios were reported.

Making the UC Specification Precise. Starting from the free UC specification, we first developed a "precise UC specification" (i.e. complying with the Well-formedness constraints listed in Table 1) asking, when needed, the industrial partners some clarifications. In this way, we greatly improved the quality of the requirements specification. e.g. by adding the Glossary, leveling out the granularity of the steps (e.g. subdividing a single step in more steps), redefining some actors, and considering new scenarios. Once the "precise UC specification" was settled, we added to it the screen mockups verifying that they comply to the well-formedness constraints (see Table 2). We chose to associate the mockups with the most relevant steps whose subject was EC2M. In this phase, the Pencil tool has been used; it proved to have the capability to quickly create realistic screen mockups.

Verifying Stakeholders Requirements. Finally, we organized a meeting where the "precise UC with screen mockup specification" has been shown to the industrial partners. That occasion has been very useful for identifying some misunderstandings between our understanding of the EC2M system and what the industrial partners really desired. Moreover, the screen mockups have allowed to perform a sort of prototype verification helping the industrial partners to detect problems difficult to find inspecting only textual use cases. After the meeting we fixed the identified problems. The final specification was composed by 15 use cases with the relative descriptions, three primary actors, two secondary actors, 11 glossary entries, and 10 screen mockups.

During the project's meetings, the industrial partners found the screen mockups (and the glossary) very effective to improve the comprehensibility of the use cases and useful to find the ambiguities in the requirements specification. As an example, the EC2M system allows to semantically classify the documents by means of an ontology and at the same time the ontology allows to search documents using semantic tags. Since an ontology is a quite complex object, several ways can be chosen to graphically represent and use it. They are strongly different for what concerns, for instance, the usability perspective. The creation of screen mockups for the steps related to the usage of the ontology allowed to: (1) improve the understanding of

the operations involving the ontology for non-experts, and (2) precisely define the kind of interactions required to the system users in order to complete the steps mentioned above. The professionals involved in the creation of the functional requirements specification for EC2M were satisfied of the method. One of them at the end of the project reported the following phrase: "discipline and mockups are two essential ingredients to improve the quality and comprehension of requirements!".

4 Conclusion and Future Work

In this work we have proposed a novel method for capturing and specify requirements of a software system. The novelty is the integration and synergy between screen mockups and use cases. In fact, the proposed method produces disciplined use cases and coherent screen mockups that are fully integrated in the development process. The positive aspects of the requirement specifications produced with our method are: – easier to comprehend (thanks to the screen mockups), – less prone to inconsistencies (thanks to the glossary), – less prone to incompleteness (thanks to the strong constrains on the form of the scenarios), – and in general of "good quality" since checking the many well-formedness constraints associated with our template results in a deep and strongly structured inspection.

It is important to note that differently from specifying use cases using formal languages or modelling notations our specifications may be read and understood also by non-experts. On the other side, the production of this kind of requirements specifications requires an extra effort compared to producing use case specifications adhering to a very loose template, and obviously the knowledge of our quite precise template.

The method has been successfully applied in an industrial project and used for many years in software engineering student's projects at the University of Genoa.

As future work, we intend to implement a supporting tool able to: (1) help in the creation and visualization of the requirements specifications using our format, and (2) automatically check the compliance of the requirements specifications to the various well-formedness constraints.

A Precise Use Cases with Screen Mockups Specification Well-Formedness Constraints

See [11] for the definition of the form of the requirement specification based on disciplined use cases augmented with screen mockups.

Table 1. Well-formedness constraints for requirements specification

- A summary use case cannot be included in either a user-goal or a subfunction use case
- A user-goal use case cannot be included in a subfunction use case
- A summary use case must have the goal and the stakeholder parts
- If a use case C includes C1 in the use case diagram, then at least a line corresponding to "include C1" must appear in the description of C, and vice versa (i.e. every inclusion in the use case descriptions must appear in the use case diagram)
- If a use case C extends C1 in the use case diagram, then at least a line corresponding to an extension point for C must appear in the description of C1, and vice versa (i.e. every extension point in the use case descriptions must correspond to an extension relationship in the use case diagram)
- The actors listed in the use case descriptions should be in accord with those appearing in the use case diagram and vice versa
- Each listed actor of a use case must appear at least in one step of its scenarios
- The subject of a step of a scenario different from the system, must appear among the use case actors
- If a use case has no actors, then it must have a trigger*
- If a step has a condition *cond* different from true, then there should be some extensions starting from the same step with conditions $cond_1, \ldots, cond_n$ s.t. the logical disjunction of $cond, cond_1, \ldots, cond_n$ is true
- Each complete scenario must include at least a step where the subject is the system
- All the initial steps of a set of extensions starting from the same point must have the same subject
- Each data listed in the glossary must appear at least in one step of a use case
- Each system attribute listed in the glossary must at least
  - appear in the effect part of a step of a use case
  - appear either in the condition or in the interaction part of a step of a use case

* *(e.g. a use case describing a periodic activity made by the system each hour)*

Table 2. Well-formedness constraints for screen mockups

Actor as Subject and Initial Mockup (i.e., at the beginning of the step)

- Let $S_0, S_1, \ldots, S_n$ $(n \geq 0)$ be some steps having an actor as subject s.t. $S_1, \ldots, S_n$ are the first steps of extensions starting from S_0, and let M be the initial mockup of $S_0, S_1, \ldots, S_n$
  - If the interaction part of S_i $(0 \leq i \leq n)$ refers to some communication from the actor to system, then some means to represent it must appear in M (e.g. when S_0="Client confirms", S_1="Client refuses", and two buttons "Confirm" and "Refuse" appear in M)
  - If M contains some means for realizing some communication from the actor to system, then there should be S_i $(0 \leq i \leq n)$ referring to such interaction

Actor as Subject and Final Mockup (i.e., at the end of the step)

- Let $S_1, \ldots, S_k$ $(k \geq 1)$ be some steps having an actor as subject, and let M be the final mockup of $S_1, \ldots, S_k$ ($S_1, \ldots, S_k$ must be steps having the same interaction part appearing in different scenarios of even different use cases)
  - If the interaction part of S_1 (that it is coincident with those of $S_2, \ldots, S_k$) refers to some communication from the actor to system, then M should show how it is going to be realized (e.g. the step S_1 has the form "User confirms the deletion", and in M there is a button "Confirm Deletion"; notice that this step may represent the confirmation of different kinds of deletions)
  - If the interaction part of S_1 (that it is coincident with those of $S_2, \ldots, S_k$) includes a reference to some specific information (flowing from the actor to system), then such information must appear in some way in M (e.g. "Actor inserts the password" and "password" appears in M)
  - If M shows how some communication is going to be realized (from the actor to system), then the interaction part of step S_1 (that it is coincident with those of $S_2, \ldots, S_k$) should refer to it

System as Subject and Mockup

- Let $S_1, \ldots, S_m$ $(m \geq 1)$ be some steps having the system as subject, and let M be the final mockup of $S_1, \ldots, S_m$ ($S_1, \ldots, S_m$ must be steps having the same interaction part appearing in different scenarios of even different use cases)
  - If some information appears in M, then it should be derived by the interaction parts of the previous steps or by the system attributes (e.g. in M appears "You are logged as John Doe", and the name of the current logged user "John Doe" is recoverable by the system attributes or it was provided by the user in some previous step)
  - If the interaction part of S_1 (that it is coincident with those of $S_2, \ldots, S_m$) refers to some communication from system to actor, then some means to represent such communication must appear in M (e.g. "System confirms the required deletion" and either a pop-up or a message box containing a sentence equivalent to "deletion confirmed" appears in M)

References

1. Astesiano, E., Cerioli, M., Reggio, G., Ricca, F.: A phased highly-interactive approach to teaching UML-based software development. In: Proceedings of Educators Symposium at MoDELS 2007, pp. 9–18. University of Goteborg (2007)
2. Astesiano, E., Reggio, G.: Knowledge structuring and representation in requirement specification. In: Proceedings of 14th International Conference on Software Engineering and Knowledge Engineering, SEKE 2002, pp. 143–150. ACM (2002)
3. Briola, D., Amicone, A., Laudisa, D.: Ontologies in industrial enterprise content management systems: the EC2M project. In: Proceedings of 5th International Conference on Advanced Cognitive Technologies and Applications, pp. 153–160. IARIA (2013)
4. Choppy, C., Reggio, G.: Improving use case based requirements using formally grounded specifications. In: Wermelinger, M., Margaria-Steffen, T. (eds.) FASE 2004. LNCS, vol. 2984, pp. 244–260. Springer, Heidelberg (2004)
5. Cockburn, A.: Writing Effective Use Cases. Addison Wesley, New York (2000)
6. Ferreira, J., Noble, J., Biddle, R.: Agile development iterations and UI design. In: Proceedings of Agile Conference, AGILE 2007, pp. 50–58 (2007)
7. Hartson, H.R., Smith, E.C.: Rapid prototyping in human-computer interface development. Interact. Comput. **3**(1), 51–91 (1991)
8. Leotta, M., Reggio, G., Ricca, F., Astesiano, E.: Towards a lightweight model driven method for developing SOA systems using existing assets. In: Proceedings of 14th International Symposium on Web Systems Evolution, WSE 2012, pp. 51–60. IEEE (2012)
9. O'Docherty, M.: Object-Oriented Analysis and Design: Understanding System Development with UML 2.0. Wiley, Hoboken (2005)
10. Reggio, G., Leotta, M., Ricca, F., Astesiano, E.: Business process modelling: five styles and a method to choose the most suitable one. In: Proceedings of 2nd International Workshop on Experiences and Empirical Studies in Software Modelling, EESSMod 2012, pp. 8:1–8:6. ACM (2012)
11. Reggio, G., Ricca, F., Leotta, M.: Improving the quality and the comprehension of requirements: disciplined use cases and mockups. In: Proceedings of 40th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2014, pp. 262–266. IEEE (2014)
12. Ricca, F., Scanniello, G., Torchiano, M., Reggio, G., Astesiano, E.: On the effectiveness of screen mockups in requirements engineering: results from an internal replication. In: Proceedings of 4th International Symposium on Empirical Software Engineering and Measurement, ESEM 2010, pp. 17:1–17:10. ACM (2010)
13. Ricca, F., Scanniello, G., Torchiano, M., Reggio, G., Astesiano, E.: On the effort of augmenting use cases with screen mockups: results from a preliminary empirical study. In: Proceedings of 4th International Symposium on Empirical Software Engineering and Measurement, ESEM 2010, pp. 40:1–40:4. ACM (2010)
14. Ricca, F., Scanniello, G., Torchiano, M., Reggio, G., Astesiano, E.: Assessing the effect of screen mockups on the comprehension of functional requirements. ACM Trans. Softw. Eng. Methodol. **24**(1), 1:1–1:38 (2014)
15. Rivero, J.M., Grigera, J., Rossi, G., Luna, E.R., Montero, F., Gaedke, M.: Mockup-driven development: providing agile support for model-driven web engineering. Inf. Softw. Technol. **56**(6), 670–687 (2014)

16. Scanniello, G., Ricca, F., Torchiano, M., Gravino, C., Reggio, G.: Estimating the effort to develop screen mockups. In: Proceedings of 39th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2013, pp. 341–348 (2013)
17. Zhang, J., Chang, C., Chung, J.Y.: Mockup-driven fast-prototyping methodology for web requirements engineering. In: Proceedings of 27th International Computer Software and Applications Conference, COMPSAC 2003, pp. 263–268. IEEE (2003)