POLVO - software for prototyping of low-fidelity interfaces in agile development

Júnia Gonçalves, Caroline Santos

Computing Departament
Universidade Federal dos Vales do Jequitinhonha e Mucuri
Diamantina – Minas Gerais – Brazil
{junia.goncalves, carol.qs} AT gmail DOT com

Abstract. In the process of software development, the ease with which the user can perform his/her tasks in the system - commonly called usability - is an important requirement. The prototyping of user interfaces is one of the most widely used techniques to specify this type of requirement. This paper presents the importance and need to improve and increase the agility of prototyping interfaces in agile development processes. The authors propose a software that is able to build low-fidelity prototypes, document them and support user testing, to aid the process of interface building in the Scrum methodology.

Keywords: Low-fidelity Prototype, Interaction Design, Agile Methods, Scrum

1 Introduction

Traditional models of software development, such as Waterfall, are generally focused on following the plan, instead of satisfying the customer. So when changes are needed, either requirements or technology, this results in an increase in costs proportional to the stage where the project is [6] - The more advanced the process, the greater the increase in costs. Even the unified processes of software development following the incremental-iterative[1] model are not focused on customer satisfaction. Thus, agile methods have gained importance in software development, rather than traditional processes, since the former are better suited to dynamic environments and tight deadlines, [6]. This adaptability exists because the agile methodologies are focused on continued delivery of a functioning system, the reduction of the burden of documentation on the development and frequent contact with the client, resulting in improved response to changes [5]. Among the agile, Scrum stands out with its product management practice.

[1] According to Sommerville [17], the iterative-incremental models are those in which the system requirements are identified and prioritized, followed by a series of development stages, and each of these stages results in the delivery of a subset of system functionalities.

For applications to stand out today, it is necessary that, for their development, a stance toward the user be adopted, with interfaces that are usable and easy to learn, because when they are not, more resources will have to be used in training and supporting the user, and this creates dissatisfaction with the tasks to be performed [1]. However, interaction design approaches are not clearly provided in software development, either in traditional processes, or in agile methodologies; therefore, the systems do not give support to usable form tasks [3]. Thus, it is necessary that an attitude directed to the user and use be inserted during the procedure in order to add greater value to the final product.

In the process of software interface design, prototypes are designed to facilitate communication between developers and stakeholders [12]. According to Ambler [2], the conceptual modeling in agile methodologies happens in draft form, drawn on white boards, sheets of paper or flip charts, as these sketches are enough for the explanation of what should be modeled and then produced. Besides, Nielsen [10] believes that the performance of processes focused on system usability can be improved through computer support, stressing that it is desirable to develop an application that supports the creation of mock-ups of user interface and user testing. Thus, this study proposes to develop software for a low-fidelity prototyping, agile practices aligned with the Scrum methodology, supporting documentation and usability testing.

This work consists of five sections, the first of which presents the problems worked on here, as well as their motivations. Section number 2 presents concepts and theories about the design of user interfaces, followed by section 3, which presents concepts on software engineering and agile methods. The subsequent sections, 4 and 5, are related to the evolution of concepts and theories mentioned above, basing the development methodology used and details of the methodology concerning the design of the application, together with the documentation of its development. Finally, the concluding remarks are presented.

2 Human-Computer Interaction and Interface Prototype

Once systems began to be marketed on a large scale and distributed to many users, also started the concern that users could interact with the system. It became necessary to adapt these systems to the needs and goals of users, facilitating the use and development of their work and leisure activities. In literature, there are three main approaches for developing system interfaces, usability engineering, user-centered design and usage-centered design. The first is a process directed to the ease of learning and use of systems, providing them with a friendly interface. The second is to change the focus of development, leaving the focus on technology, moving to focus on the user, in which he/she is studied, included and takes part in testing. The last process focuses on the use of the system, in which activities it should be conducted and what the system should cancel, leaving the figure of the user aside [14].

The User-Centered Design (UCD) is a philosophy that is based on the needs and interests of users, emphasizing the creation of usable and understandable products [11]. Thus, the UCD approach can be used to create various products, including software. As Donald Norman introduced the concept of User-Centered Design, was

also embedded the concept of Activity-Centered Design, in which tasks and user behavior were studied for framing systems. But later, Norman concluded that the model-driven tasks produced inadequate results [4]. Thus, Cooper, Reimann and Cronin [4] suggested that not only the profile, the activities and environment of users be investigated, but also their goals. The technology-focused organizations rarely have a proper UCD process, if ever presented [4]. Even if the Activity-Centered Design manages a system that can be modeled, this is just one increment, since it does not provide solutions to differentiate the product on the market and does not satisfy the user correctly. Thus, the Goal-Directed Design (GDD) is to be a link between the research user with design, using ethnographic techniques, interviews with stakeholders, market research, detailed models of users, scenario-based design and a set principles and patterns of interaction, meeting the needs of users and organizations.

2.1 Interface Prototype

The prototyping of interfaces can be defined as a limited representation of a design that enables users to interact and explore it, and its main idea is to create something that resembles the final product and can be tested by end users, saving resources [10]. In this work, prototypes will be understood as limited representations and non-executable user interfaces, which can be manipulated in order to validate the actual interfaces of the system under development. Prototypes can be classified into high-fidelity prototypes and low-fidelity prototypes. The first type of prototype is one that closely resembles the final product, being developed under a programming language and sometimes executable. They are used to demonstrate a real image of the system and evaluate patterns and style guides [14]. The low-fidelity prototypes are those that have little resemblance to the final product, using simplistic means for their representation, instead of metal and electronic displays. These prototypes are useful, since they tend to be simple, cheap and rapidly produced. They can be quickly modified, supporting the exploration of alternative designs. They are useful for identifying market requirements, assess multiple design concepts, deal with issues on layout of the screen [12], and, finally, are useful for exploring the possibilities of navigation [14].

For Snyder [16], paper prototyping is a technique that involves the creation of low-fidelity prototypes on paper that can be manipulated by a facilitator that simulates the behavior of the system to conduct usability tests. This technique provides benefits in terms of skill development, ease of communication between multidisciplinary teams and stakeholders, encourages creativity, does not require technical skills, which allows end users to be involved in prototyping the interface, enhancing the quality of the final product [5], though users might find it strange at first [16].

3 Agile Software Development

Software Engineering consists of a set of activities that aim to create software product - commonly called software process [17]. In the 90s, dozens of developers joined together, sharing the dissatisfaction with the prevailing process of software development - UP[2] - and wrote the so-called Agile Manifesto, initiating a series of agile methodologies. In the context of software engineering, agility reflects the capacity to accommodate the necessary changes that originate during the process of software development [13]. Thus, agile methods present a great response to the changes, since following the plan and providing comprehensive documentation are items of low priority, indicating the absence of a rigid structure, which leads to creativity, self-manageable processes and increasing returns. Besides agility, agile methodologies embody other values to their process, such as the creation of cohesive teams, communication between the implementation teams, engineers, managers and stakeholders, and the latter are considered part of the development team and add greater value to products that are potentially deliverable to the customer.

Scrum is a framework for agile development of complex products, within which it is possible to employ various processes and techniques [15], one method to manage the development of a product of any technology, including software [7]. The devices available in Scrum are the Product Backlog, the Sprint Backlog, the Burndown Release Delivery and the Sprint Burndown. The Product Backlog is a list that prioritizes all that is needed in the product. The Sprint Backlog is a list of tasks to turn the Product Backlog into a potentially shippable product. Both Burndown Release Delivery and Sprint Burndown charts measure how much effort and time are still needed. The former is directed to the Product Backlog and the second to Sprints. Moreover, in Scrum, there are time-boxes, ie, events of fixed duration, which are the Release Planning Meeting for Delivery and Sprint.

4 The Project

The purpose of the software presented here is to develop a system of low-fidelity prototyping, according to the story of the Scrum framework and tasks that require a user interface and supporting documentation and usability testing. The development was focused on construction of key features, with no effort at that time to implement non-functional requirements such as security and performance, because it is an early application in its evolutionary chain. However, the non-functional requirement known as usability has been strongly sought and tested. In order to obtain usability in the system, an approach that would integrate UCD and agile methods was chosen, but there is no formal model for this, which has led several authors to apply

[2] Unified Process is a methodology for developing iterative-incremental software, which is characterized by extensive documentation through UML diagrams [17].

effort in the integration of UCD with models of software development with agile methodologies [8, 9, 18]. To make this integration difficult, there are still UCD activities that are not effective in practice [19], making specific case studies to be developed to test the compatibility of these UCD activities with agile methods [8].

A model for such integration is suggested by Sy [18], which is the focus of the agile team on a few features of the system at a time, which means that each feature receives a more careful work, and can capture design flaws earlier, provide changes and incorporate these adjustments. In addition, there must be different cycles of design and implementation, although they may occur in parallel, in which functionality is implemented with a basic interface as its design is made. So, the next iteration, the final design is implemented. There should also be the so-called Cycle Zero, which is the phase of gathering usability requirements.

4.1 Development Methodology

Based on Sy's model [18], a similar methodology was used in this work, consisting of a mix between Goal-Directed Design process and Scrum agile methodology. The Zero Cycle is divided into three activities of requirement gathering system based on the users' goals, according to the initial stages of Goal-Directed Design. Subsequently, the Sprints take place in parallel with the design process and prototyping of the interfaces, resulting in an interface refining activity through usability testing and participatory design sessions. Finally, the final interface can be implemented. The activities of the methodology are presented below and illustrated in picture 1.

The activities consist of Research: it is the literature review of the areas of Software Engineering and Human-Computer Interface, and is presented and cited throughout the text; Modeling: it is the creation of personas and goals based on information obtained during the research; Requirement Definition: it is the analysis of the personas and goals to generate the specification of the users' scenarios; Definition of the Product Backlog: from scenarios, The Product Backlog is built on the application, together with the story definition, its importance, demonstration, initial estimates and essential UML diagrams; Definition of Sprint Backlog: The development loop occurs in activities 5, 6 and 7, starting with the activity known as Sprint Backlog Definition, when the stories are chosen and will compose the current Sprint, which leads to a potentially shippable product until the entire Product Backlog is implemented; Defining the Framework: activity that occurs parallel with Execution of Sprint, in which occurs prototyping and creating the visual identity of the application according to the stories present in the current Sprint; Sprint's implementation: it is the implementation of selected stories in the activity known as Setting the Sprint Backlog, with the development of an intermediate interface that will be replaced later; Refinement: it happens after the loop of activities 5, 6 and 7, consisting of performing usability testing, participatory design sessions with users of focus and redesign of user interfaces; Special Sprint: it consists of a Sprint in which end-user interfaces are implemented, replacing the interfaces developed in the intermediate activity of Sprint Implementation. This activity takes place parallel with the activity Development Support. Development Support: it consists in monitoring the

implementation of end-user interfaces if any development challenges are identified and lack some adjustment.

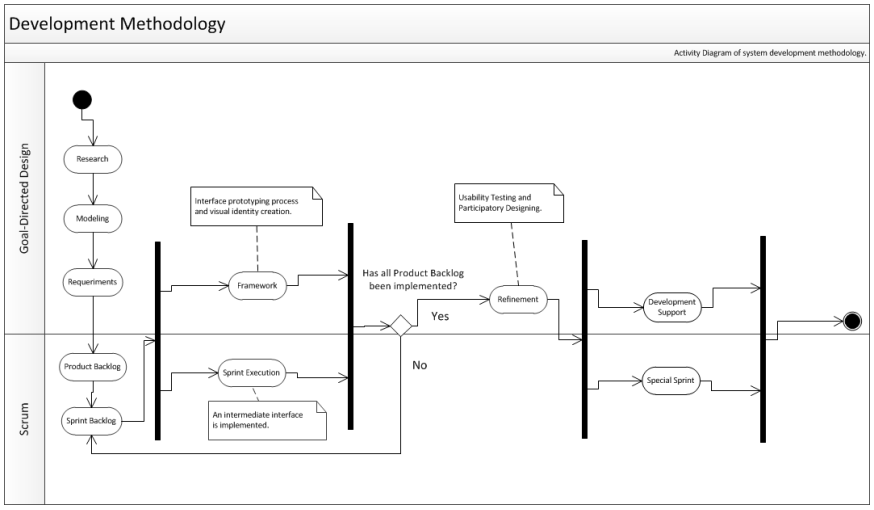

Fig. 1. Development Methodology: Mix between Goal-Directed Design and Scrum

4.2 POLVO and its agile development directed by goals

This is the breakdown of major activities conducted during the development of the system:

4.2.1 System Models

The system was modeled after the creation of six personas, models that focus on users of the system focusing on characteristics and behavior [4], representing its primary users, secondary, additional, served and negative. The primary personas are those that are directly related to the functionality of the system, while the secondary personas are satisfied with the features presented, in spite of having special needs. The additional persona is one that is not primary or secondary, whose needs are a combination of the needs of primary and secondary users, and is completely satisfied with the functionality of the system. The served persona is the one who does not use the product, but is affected by its use, while the negative persona is one whose needs are not supported by the product [4]. The personas were described according to their context, and all the personas were, on some level, set in the context of software development with Scrum. They were also described as for the activities they carry out, their attitudes, skills, motivations and goals. Moreover, they were mapped as to their experiences and knowledge on technology, Human-Computer Interface, participation in the implementation of applications and interest in the prototyping process in order to show the level and the difference between the personas.

4.2.2 Requirement Definition, Product Backlog and Sprints

Based on established personas, their scenarios were developed, which consist of how the persona uses the product in its day-to-day activities in order to achieve its goals [4], and these scenarios are the system requirements, added to the device Product Backlog as part of user stories. Thus, the stories are constructed based on user scenarios, divided into six categories: Scrum, Prototyping, Usability Testing, Security and Documentation, with the first three being the highest priorities. According to the device Product Backlog, the software must support an authentication system and different access control to "designer" and "user", and only the "user" can perform usability tests, preventing the "designer" from performing tests and generating erroneous results, but the designer can still view the navigation between the prototypes. The "User" can also create interface prototypes, but the user cannot view or modify prototypes that are not his property. Also, it should support product management, user stories, tasks and Sprints and construction of prototypes based on the tasks to be performed in parallel. Usability testing should be performed by Sprint since the prototypes are interconnected through links or actions, and must mark what prototype is the Sprint start screen. It is important to note that the prototypes are built as interactive and non-executable, that is, are navigable, but do not process data, similar to the prototypes on paper. The present application versioning allows alternative prototypes to be created and stored, providing a documentation of the construction of interfaces, just like the notes come from the documentation generated by the designer during the usability test. Finally, the user can add "designer" or "user" to his product in order to generate collaborative development processes, although the change control is left to the users.

The Sprints were defined according to the importance of the user story in the Product Backlog, followed by the duration of the stories already chosen. Thus, three sprints were defined, the first forming the core functionality of the application, construction of prototypes, Products, Stories and Tasks and Sprints, the second consisting of Versioning features, Notes and Execution of Usability Testing, and finally the last Sprint is made up of the features of Result of Usability Testing, System Authentication and Access Control and Adding Designer / User.

4.2.3 Prototyping, Usability Testing and Special Sprint

Alongside the implementation of Sprint, there was the process of low-fidelity prototyping on paper in the application. Several alternative prototypes were built until one was chosen based on a Heuristic Evaluation, out of which the most usable prototype would be built on interactive paper, to be used in usability testing with users on focus. Thus, the prototypes were built on paper according to the stories implemented in the Sprint, and the interactive prototype was finally built. In addition, the visual identity of the application was also designed, being partially embedded in the interactive paper prototype and fully incorporated into the final interface implemented.

At the end of the loop in the activities of Sprint Definition, Framework Definition and Implementation of Sprint, the activity known as Refinement of the user interface started, through sessions of usability testing conducted with a small number

of users on focus (from 1 to 5), who were invited to develop a product that consisted of a website for lyrics, in which these were published collaboratively by its users. This product was named "Lyrics", with four user stories, four tasks, two Sprints and two interface prototypes. The tasks of the usability testing focused on product, user stories, tasks and Sprints management, prototype building, versioning, annotations and adding other users. Participants were free to create / prototype interface as they wished. Usability testing consisted of an initial questionnaire of adherence to the focus group, the tasks to be performed on prototype interactive paper and a final questionnaire on the impressions of the software, as well as being invited to "think aloud". Based on the responses of the final questionnaire, participants were invited to a meeting of Participatory Design in which they built their solutions and / or ideas on interactive prototype paper, using simple tools such as pen and scissors. The solution was built and tested if there were some related problem detected during testing, whereas the ideas, concepts and opinions about the visual identity were not incorporated into the prototype. The biggest problems were the creation of user stories and their allocation to Sprint and adding designer / user.

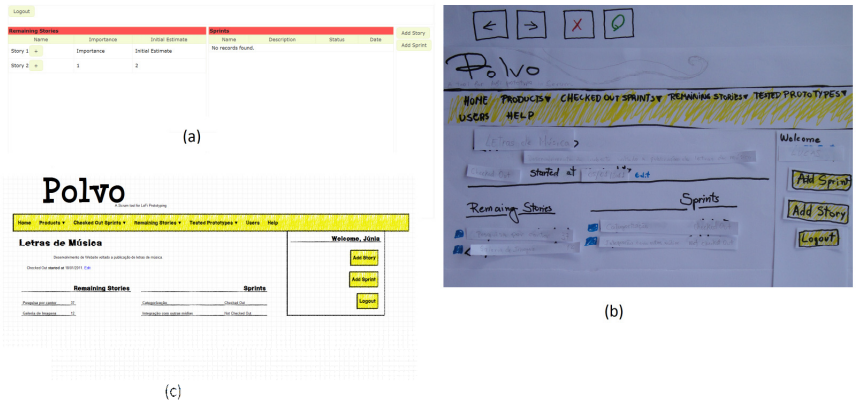

Fig. 2. Intermediate interface (a), interactive paper prototyping (b) and real final interface (c).

After the sessions of Usability Testing and Participatory Design and redesign based on these sessions, the interactive paper prototype was implemented along with its previously designed visual identity, featuring Sprint Special. During the implementation of the final interface user, development challenges were not detected, not requiring design review.

5 Conclusions

The prototyping of user interfaces is perceived in various software development methodologies, since it presents a better way to express the requirements

of the user interface, instead of diagrams and textual descriptions [17]. There is also a need for these prototyped interfaces to be tested and documented. Because of that, the software "POLVO" was proposed, which brings such features. Possibly, the use of "POLVO" will bring benefits to software developers, such as increased agility in the development of user interface prototypes, ease of usability testing application with interactive prototypes, carrying out participatory design sessions and documentation of prototypes.

Although the tool has been developed, it needs to be evolved, adding other non-functional requirements such as enhanced security, performance, reliability and availability. Moreover, its functionalities can be expanded to include all of the Scrum management added to its current capabilities. The construction of such software has emerged from studies on agile development and prototyping of interfaces. For evaluation of its features, the use of the software by a team of developers would be required. Thus, within the proposed evolution of the software "POLVO", there can also be case studies focusing on the integration of the agile process Goal-Directed Design and Scrum and adaptations to the current model used.

6 References

[1] S. Ambler. (2005-2006). *The Agile Unified Process (AUP)* [Online]. Available: http://www.ambysoft.com/unifiedprocess/agileUP.html.
[2] S. Ambler. "User Interface Design Tips, Techniques, and Principles", Ambysoft.
[3] S. Blomkvist. "User-Centred Design and Agile Development of IT Systems" M.S. thesis, Departament of Information Technology, Uppsala University, Uppsala, Sweden, 2006.
[4] A. Cooper *et al*. *About Face 3: The Essencials of Interaction Desgin*. Indianapolis: Wiley Publishing Inc., 2007.
[5] J. Gullisken *et al*. "Key Principles for User Centred Systems Design", *Behaviour and Information Tecnology*, vol. 22, no. 6, 396-409, Nov-Dec. 2003.
[6] J. Highsmith, A. Cockburn. "Agile Software Development: The Business of Innovation", *IEEE Computer*, vol 39, no. 9, 120-127, Sep. 2001.
[7] A. Koch. "Agile Software development: Evaluating Methods for your Organization". Norwood: Artech House, 2005.
[8] P. McInerney, F. Maurer. "UCD in Agile Projects: Dream Team or Odd Couple?", *Interactions*, vol. 12, no. 6, 19-23, Nov-Dec. 2005.
[9] T. Memmel *et al*. "Agile Human-Centered Software Engineering" in *People and Computer XXI HCI*, Beijing, China, 2007.
[10] J. Nielsen. *Usability Engineering*. Boston: Academic Press, 1993.
[11] D. Norman. *The Design of Everyday Things*. New York: Basic Books, 1988.
[12] J. Preece *et al*. *Interaction Design:Beyond Human-Computer Interaction*, 2th ed. Indianapolis: Wiley Publishing Inc., 2007.
[13] R. Pressman. "Software Engineering: A Practitioner's Approach", 6th ed. New York: McGraw-Hill Science/Engineering/Math, 2004.
[14] C. Rosemberg *et al*. "Elicitação de Requisitos e Design Participativo através de Protótipos de baixa Fidelidade - um Estudo de Caso" in *Congresso Tecnológico InfoBrasil*, Fortaleza, Brazil, 2008.

[15] K. Schwaber. "Scrum Guide". Seattle: Scrum Alliance, 2009.
[16] C. Snyder. "Paper Prototyping: The fast and easy way to design and redefine user interfaces". San Francisco: Elsevier, 2003.
[17] I. Sommerville. "Software Engineering", 6th ed. Addison-Weasley, 2000.
[18] D. Sy. "Adapting Usability Investigations for Agile User-Centered Design" in *Journal of Usability Studies*, Canada, 2007, 112-132.
[19] K. Vredenburg *et al.* "A Survey of User-Centered Design Practice" in Computer-Human Interaction 2002, Minnesota, USA, 2002.