Integrating Human Factors in Information Systems Development: User Centred and Agile Development Approaches

Leonor Teixeira[1,3(✉)], Vasco Saavedra[1], Beatriz Sousa Santos[2,3], and Carlos Ferreira[1,3]

[1] Department of Economics, Management and Industrial Engineering, University of Aveiro, Aveiro, Portugal
{lteixeira, vsaavedra, carlosf}@ua.pt
[2] Department of Electronics, Telecommunications and Informatics, University of Aveiro, Aveiro, Portugal
bss@ua.pt
[3] Institute of Electronics and Informatics Engineering of Aveiro (IEETA), Aveiro, Portugal

Abstract. This paper presents an overview and discussion based on the literature review of recent research of some practices that incorporate human factors, emphasizing the user-centred design (UCD) and agile software development (ASD) approaches. Additionally, this article presents an experience of the development of a web-based application that aims to manage the clinical information in haemophilia care, which benefited from these practices, making use of some methods to support the collaboration and communication between designers, users, and developers. The results of our experience show that the hybrid approach, that combines the principles of UCD with values of ASD can help to integrate human factors into the software development process in a highly complex environment, characterized by missing information, shifting goals and a great deal of uncertainty, such as the healthcare field.

Keywords: Human factors · Information system development · User-Centred design · Agile software development · Interactive software

1 Introduction

The dynamics that currently characterize the software market also require the same dynamic and some flexibility when managing the software requirements. On the other hand, the traditional models of software engineering (SE) and management practices for the development of this type of projects limit the flexibility for the adaptation of those requirements, often compromising the quality of the final product.

In SE, it has been intuitively accepted that user involvement during the Systems Development Life Cycle (SDLC), leads to a better management software requirements, and consequently can lead to system success [1]. User involvement in SDLC facilitates the understanding of their work environment and can improve the quality, accuracy and completeness of the requirements.

V.G. Duffy (Ed.): DHM 2016, LNCS 9745, pp. 345–356, 2016.
DOI: 10.1007/978-3-319-40247-5_35

Users typically have important tacit knowledge about the system domain and context of the system usage [1] that can be difficult to be articulated with traditional techniques from SE. For this reason, the development process of interactive software (IS), albeit with major contributions coming from the SE, recently has integrated methods from other knowledge areas to cover the social and human aspects associated with the interaction component.

On the other hand, aspects such accelerate of time to market, increase the final product quality, align the information and communication technologies (ICT) with business strategies, and promote flexibility [2, 3] are increasingly important values in the software development context. In order to consider this values, the software development industry has been adopting agile methods, because they are more flexible and can promote benefits such productivity gains and business alignment [2].

The user-centred design (UCD) and agile software development (ADS) emerge as appropriate counterproposals to traditional development methodologies, changing the values of project management, and centring the focus on people.

This paper presents an overview of these two approaches based on a literature review of recent research. Additionally it describes an experience of developing a Health Information System that benefited from these practices.

The remainder of this paper is organized as follows. Section 2 shows some concepts related to human factors in the software development process with focus on UCD and ASD approaches. Section 3 describes an experience of developing a Health Information System that benefited from a hybrid approach, combining traditional methods with UCD and ASD practices. Finally, Sect. 4 presents some conclusions.

2 Background: Human Factor in Software Development Process

Actually, the human factors play a very important role in the software development (SD) with a major impact on the process performance and product success. In this area of research, the growing importance of human factors are proved by the existence of specific tracks devoted to this topic in conferences related with SE, as well as the specific issues in some important journals in the field, such as Information and Software Technology [4].

SD has been characterized as a set of activities comprising a set of tasks grouped in system analysis, system design, coding, and testing. Moreover, the SD has been considered a socio-technical endeavour, and particularly in the case of SE, the effectively communication with users and team members is increasingly important.

Actually, in the development of interactive software (IS), it is important to consider, not only the functional and technical specifications, but also all the aspects related to the user interface and the interaction process. From the user's point of view, the system is usually used and evaluated as a whole, and the separation between technical/functional components and the user interface is not possible. Conceptually

these components can be designed using concepts from different knowledge areas. While the former is defined from the user's specification, and is generally addressed by SE, the interface component, which deals with issues related to the interaction between the user and the IS, is associated with Human-Computer Interaction (HCI) and/or Usability Engineering (UE).

In order to facilitate the development process of IS, several methods and techniques have been proposed to provide solutions for the effective involvement of users and that take into account the human factors. Joint Application Development (JAD) [5], Agile Software Development (ASD) methods [6, 7], Lean Software Development (LSD) [8, 9], Effective Technical and Human Interaction with Computer based Systems (ETHICS) [1] are some examples of those techniques. Other contributions attempt to accommodate methods from SE and HCI areas in the same process. For example, Harmelen [10] suggests the Object-Oriented and Human-Computer Interaction (OO&HCI) method, which integrates object-oriented (OO) modelling techniques, and HCI concepts for developing IS. Other prominent proposals in this area are the Design for User-Centred Innovation (DUCI) presented by Zaina and Álvaro [11], which attempt to integrate HCI into the traditional OO development, combining the merits of both approaches in order to design software that is both usable and useful. The proposal by Mayhew [12], implemented through the Framework 'Usability Engineering Life-cycle', also resulted from an attempt to redesign the process of software development involving methods and activities of the UE.

In fact, the terms 'user-centred' and 'customer-focus' are the most used in the human factors' community.

Nowadays, the set of most well-known techniques used by software development industry to include human factors has been characterized by two major approaches: on the one hand the techniques that follow the principles of agile software development (ASD), which aims to achieve increased velocity and flexibility during the development process; on the other hand the techniques which put the goals and users' needs at the centre of software development, in order to deliver software with appropriate usability, known as user-centred design (UCD).

According Brhel *et al*. [13], while the agile methods focus on the question of "how useful software can be developed, with customer value being understood as primarily driven by providing an appropriate functional scope"; the UCD ensures that the "goals and needs of the system's end-users is the focus of the product's development" [13].

Given that the UCD approach focuses on the *user* and produces *usable* software and ASD focuses on the *customer* and produces *useful* software, hybrid developments can contribute to the production of usable and useful software, trying to combine the merits of both approaches.

2.1 About User-Centred Development Approach

In recent years, the methodologies for the development of IS have placed great emphasis on iterative and incremental development practices, with evaluation processes throughout the entire development cycle. On the other hand, important advances

in the SE area have emerged from adaptations of traditional development methods, based on iterative and incremental models supported by the principles of user-centred design (UCD).

The literature reports that the UCD methodologies, with a central focus on continuous evaluation, in iterative processes of a formative evaluation, have been the most sought for the development of IS. UCD is an approach to interactive system development that focuses specifically on making usable systems [14]. UCD is defined by Vredenburg *et al*. [15], as "an approach to designing ease of use into the total user experience with products and systems, involving two fundamental elements: multidisciplinary teamwork and a set of specialized methods of acquiring user input and converting it into design".

The development according to principles of UCD arises from the attempt to merge the best practices from SE and HCI and/or UE, and is defined as a philosophy that puts the user into the centre of the development process. In this approach, apart from the user, the tasks and the environment or usage-context emerge as important requirements, having as a main objective the creation of systems after a solid knowledge about the characteristics of users and the tasks they perform. Thus, the result of a good design is reflected in a usable system.

Considering the standard on human-centred design by ISO 13407 [14], there are five critical processes that should be performed in order to incorporate usability into the software development process: (i) plan the human-centred design process; (ii) understand and specify the context of use; (iii) specify the user and organizational requirements; (iv) produce designs and prototypes; and (v) perform user-based assessment.

There are several works which attempt to incorporate usability into the software development process, following a methodological approach based on the principles of UCD. However, these approaches are not governed by formal methods, but by a set of techniques and principles that put the user at the centre of development [16].

The International Usability Standard, ISO 13407 [14], specifies the principles and activities that underlie UCD:

- The process is iterative;
- Users are involved throughout design and development;
- The design addresses the whole user experience;
- The design is based on explicit understanding of users, their tasks and environment;
- The design is driven and refined by user-centred formative evaluation.

A good example is the Framework presented by Kushniruk [17] that depicts a structure relating the main techniques and assessment methods with the respective stages of the IS development cycle, combining a set of techniques from HCI and/or UE, with the traditional methods of SE (see Fig. 1).

Typically, as shown in the Framework of Fig. 1, the UCD approach is a philosophy that uses a set of techniques and methods already known in other knowledge areas, aiming to produce usable systems that meet the needs of those who use them.

Techniques coming from the Social and Cognitive Sciences, such as questionnaires, interviews, documentation analysis, and ethnographic techniques (direct observation) are the most used for the knowledge of the problem and system analysis.

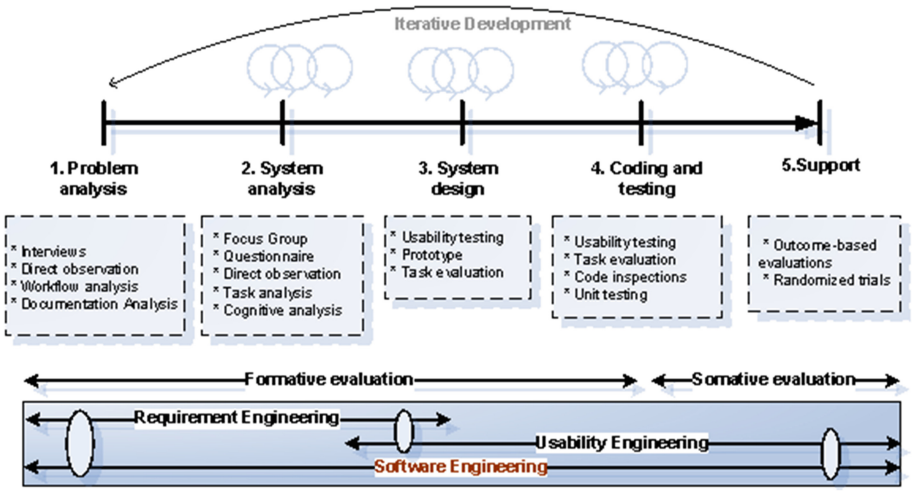

Fig. 1. Framework of IS development considering techniques and methods from different knowledge areas (adapted from [17])

For a better understanding of the mental model of the users, there are other techniques coming from HCI and/or UE, including task analysis and cognitive analysis, useful for validating the requirements previously found and for stabilizing the most volatile requirements. Prototypes accompanied by usability testing and task evaluation represent also excellent tools for requirements validation and evaluation of the solution acceptance by the users that can be used in system design, as well as in the remaining phases of the SDLC.

2.2 About Agile Software Development Approach

The term Agile Software Development (ASD) was created in 2001, by a group of people involved in defining new software development methods around a common name '*Agile Manifesto*' [18].

One of the main beliefs of the promoters of the '*Agile Manifesto*' is that the traditional methods of SE, strongly based on a strict specification and supported by formal contracts, cannot cope with the rapid change and uncertainty that the processes of a dynamic and competitive economy require.

The '*Agile Manifesto*' was developed in response to the emphasis placed by main stream software development research on planning, control, and efficiency [19], providing "a set of practices that allow for quick adaptations matching the needs of the software development" [20]. The ASD implements evolutionary and flexible software processes that answer to changes in customer requirements based on cooperation and communication, rather than on bureaucracy with models and written documents, to communicate the requirements and validate the solution [19].

According to Jyothi and Rao [3] the two most important characteristics of the agile approaches are "handling unstable and volatile requirements throughout the development life cycles and delivering products in short time frames and under budget constraints".

Agile methods attempt to valorise a development process with quick responses in real time through the involvement of people in close collaboration with customers, getting their feedback through functional software in a highly flexible structure to change. Most agile methods tries to minimize the risk of failure by developing in short periods, called iterations. Each iteration works as the development of a small project, implementing only certain features.

Agile methods are also characterized by a real time and face-to-face communication, thus eliminating excess paperwork and documentation that predominates in traditional methods, also described as being 'heavy'.

Agile methods are in their essence based on values and principles defined on the '*Agile Manifesto*' and composed by agile practices [18]. Taking into account the '*Agile Manifesto*', the values of agile methods are:

- Individuals and interactions rather than processes and tools;
- Functional software, rather than comprehensive documentation;
- Collaboration with the customer rather than contract negotiation;
- Response to changes, rather than follow a predefined planning.

With regard to the agile practices, Table 1 reports the 16 most used, taking into consideration the survey study in Version One State of Agile 2014 Research [21].

Table 1. The agile practices most adopted (%) according a survey study in [21]

Position	Practices	%	Position	Practices	%
1	Daily stand-up	80 %	9	Iteration reviews	53 %
2	Short iterations	79 %	10	Task board	53 %
3	Prioritized backlogs	79 %	11	Continuous integration	50 %
4	Iteration planning	71 %	12	Dedicated product owner	48 %
5	Retrospectives	69 %	13	Single team	46 %
6	Release planning	65 %	14	Coding standards	43 %
7	Unit testing	65 %	15	Open work area	38 %
8	Team-based estimation	56 %	16	Refactoring	36 %

According to Campanelli and Parreiras [2], the different agile practices can be grouped into (i) management practices, (ii) software process practices, and (iii) software development practices. While management practices are principles such as: on-site customer, daily stand-up meetings, release planning and open work area; the software process practices include simple design, coding standards and collective code ownership, and the software development practices correspond to, for example, pair programming and unit testing.

However, the various existing practices all share the same principles, following an iterative development process, based on strong communication between the development team members, applying minimal effort in documentation or in the creation of intermediate artefacts. Delivering working software in short time periods, with high quality and under budget and handling unstable requirements, are the main distinctive characteristics of agile methods when compared with traditional ones [2].

There are several agile methods, some of them hybrids, benefiting the best from several other methods, with slight differences in the practices they apply. eXtreme Programming (XP) [22], Scrum [23], Crystal Methods [24], Adaptive Software Development (AdapSD) [25], Feature-Driven Development (FDD) [26], Lean Software Development (LSD) [9], and Dynamic Systems Development Methodology (DSDM) [2] represent the most popular agile methods.

3 Experimental Study Using User-Centred Design and Agile Development Approaches

Although the literature often refers to the UCD and ASD approaches as methodologies, in fact they are philosophies that define a set of practices based on certain principles, using for this purpose a set of techniques and methods coming from other knowledge areas.

As these approaches have different focuses at different stages of system development, they can become useful as complementary approaches in some projects.

While the UCD approach requires more investment in the early stages of the lifecycle, reducing the risk of unexpected changes in requirements, the ASD can provide a great contribution in coding phases, reducing service costs and all associated bureaucracy.

Given the complementary nature of these approaches, there are already several projects that experienced hybrid approaches, trying to combine the merits of UCD and ASD approaches in order to design software that is both *usable* and *useful* [13, 27–32]. The experimental study briefly presented in this paper is one such example which adopts a hybrid approach, taking advantage of both the UCD and the ASD, and having been applied in the development of an application to manage the clinical information in the hemophilia care.

3.1 Overview of the Project and Brief Characterization

The project at issue aimed to develop a technological application to manage the clinical information in haemophilia care, as well as to support the process of registry and submission of the data generated in the home-treatments by patients [33]. In order to manage the data, this application integrates three actors: (i) Patient who has access to a restrict online area allowing the registry of all data generated from home treatments; (ii) Physician responsible for the management of all patient's clinical data; and (iii) Nurse responsible for managing the stocks of the drugs used, as well as the registry of the hospital-treatments (see more details of application in [34–36].

The problem emerged in the scope of a highly complex environment, characterized by missing information, shifting goals and a great deal of uncertainty. Given the type of the problem and the peculiarities of the project, the development of the technology involved a strong interaction with the domain experts in the early stages of the SDLC, once the users were the main holders of the tacit knowledge needed to define the system requirements. Actually, decisions in healthcare are complex processes strongly based on tacit knowledge, which contributes for a difficult process of requirement elicitation using traditional methodologies. For this reason, it was decided to incorporate the principles of the UCD approach in the early stages of the project, more specifically in the requirement engineering phase [37].

On the other hand, and since the developers team was physically displaced, there was the need to mediate the process between users and the team of developers through a figure of the analyst. The analyst was responsible for understanding the problem, collecting the requirements and validation with the users, ensuring also a proper communication of the requirements with development team.

3.2 Framework with the Overview of the Development Approach

The approach used in the development of the present technological solution was inspired by a hybrid approach, combining UCD and ASD techniques (see Fig. 2).

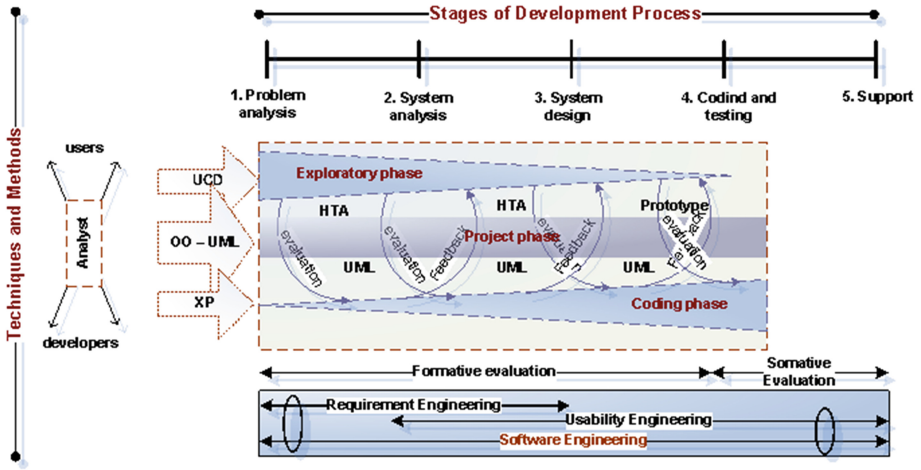

Fig. 2. Framework of development proposed based on experimental study

As shown in the framework presented in Fig. 2, the development approach adopted was based on an iterative and incremental model comprising three main phases: (i) exploratory phase, (ii) project phase and (iii) coding phase.

While the exploratory phase focused its work in the early stages of the development process, the coding phase was more prominent in the final stages, although it had begun at the same time as the requirement analysis.

The exploratory phase used some contextual techniques for understanding the problem, focusing on the document analysis, direct observation, and informal interviews with the domain experts.

As the knowledge of the domain problem was increasing, and the first requirements were collected, the first version of the conceptual model was created, using the UML notation (Use-Case diagram and Class diagram). The choice of this notation was due to UML being Object-Oriented (OO) and therefore suitable to the evolutionary characteristic of the project (incremental development). In order to validate the models with users, techniques derived from HCI were selected, specifically techniques for task analysis - hierarchical task analysis (HTA) [38], and prototypes [39].

The assessed requirements were implemented (coding phase) using the principles of the eXtreme Programming (XP) approach. As the programming team was geographically displaced from the users responsible for defining the requirements, the analyst had an important role in the process of mediation, using techniques for proper communication with each of the stakeholders.

3.3 Some Remarks About Proposed Framework

In the project development cycle (see the framework of Fig. 2 from left to right), it can be seen that the uncertainty in terms of requirements decreases, the importance of the activities of each phase changes, reversing the effort and the work required, particularly in the exploratory phase and coding phase. The project phase is the communication link between the front-end component of the project, where are the users and customers, and the back-end component, where are the developers. As such, and given the importance of the work on this level and its impact on the final result, the framework recommends a double approach to the work at this stage. First, one OO analysis using UML to build the conceptual solution in an iterative way so it can be understood by the programmers working on the back-end component of the project. On the other hand, in order to interface with users and customers on the validation component, verification and gathering new requirements, models less abstract and more easily interpretable by people without computer knowledge are recommended, such as HTA models and prototypes.

The UML has a great potential for documenting evolving projects and for communicating with developers; however, is not an easy language to interpret and, as such, is not ideal for interfacing with users without a computer background. For this reason, it is concluded that the interface with users in order to validate and complete the data previously obtained, should be made based on simple models coming from HCI and/or UE areas.

In this experiment, the HTA model and prototypes revealed having a great potential for communication with users, integrating the proposal as an essential component in the Requirements Engineering process. Regarding the type of prototype, the vertical one could be the best solution, being in line with evolutionary prototype that characterizes the XP approach.

This approach combining techniques coming from HCI with ASD, not only allowed more easily capture and validate the requirements with the end user, as also provided an excellent basis for obtaining new requirements, particularly for the most difficult requirements to capture (emerging requirements).

4 Conclusion

The present work discussed the development process of usable and useful software, focusing on human factors, based on some recent literature. The literature points to the existence of several methods that attempt to incorporate human factors in the development process, highlighting two categories: (i) the approaches that are governed by the principles of UCD; and (ii) the approaches that follow the values of ASD.

Although the UCD and the ASD have different approaches, they do have some similarities. Both philosophies are iterative, i.e., they progress in small steps providing opportunities for validation and refinement the results during the development process, and both are human-centred approaches, despite the UCD being focused in the user and the ASD focused in the customer.

Given their complementarity, there are already several proposals reported in the literature that attempt to combine the principles of the two approaches, taking advantage of the best each has to offer. However, the literature shows a clear need for more empirical and/or experimental studies regarding UCD and Agile Methods.

This paper presented an experimental study relating to the development of an Information System in healthcare, which used a hybrid approach, following the principles of UCD and ASD, aiming to reach a useful and usable final product.

The results of our experience show that a hybrid approach, that combines the principles of UCD with values of ASD can help to integrate human factors into the software development process in a highly complex environment, characterized by missing information, shifting goals and a great deal of uncertainty, such as the healthcare field.

Finally, it should be noted that despite the successful experience with this approach, this proposal has some limitations, having to be adjusted according to the type of project. It should be emphasized the high degree of demand in terms of availability from the analyst and the motivation and willingness of users who will participate in the process.

Acknowledgments. This work is funded by National Funds through FCT - Foundation for Science and Technology, in the context of the project PEst- OE/EEI/UI0127/2014.

References

1. Bano, M., Zowghi, D.: A systematic review on the relationship between user involvement and system success. Inf. Softw. Technol. **58**, 148–169 (2015)
2. Campanelli, A.S., Parreiras, F.S.: Agile methods tailoring – A systematic literature review. J. Syst. Softw. **110**, 85–100 (2015)

3. Jyothi, V.E., Rao, K.N.: Effective implementation of agile practices ingenious and organized theoretical framework. Int. J. Adv. Comput. Sci. Appl. **2**, 41–48 (2011)
4. Amrit, C., Daneva, M., Damian, D.: Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue. Inf. Softw. Technol. **56**, 1537–1542 (2014)
5. Duggan, E.W., Thachenkary, C.S.: Integrating nominal group technique and joint application development for improved systems requirements determination. Inf. Manag. **41**, 399–411 (2004)
6. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Human Behav. **51**, 915–929 (2014)
7. Losada, B., Urretavizcaya, M., Fernández-Castro, I.: A guide to agile development of interactive software with a "user objectives"-driven methodology. Sci. Comput. Program. **78**, 2268–2281 (2013)
8. Ebert, C., Abrahamsson, P., Oza, N.: Lean software development. IEEE Softw. **29**, 22–25 (2012)
9. Poppendieck, M., Cusumano, M.A.: Lean software development: A tutorial. IEEE Softw. **29**, 26–32 (2012)
10. van Harmelen, M.: Interactive system design using OO&HCI methods. In: Object Modelling and User Interface Design: Designing Interactive Systems, pp. 365–427. Addison Wesley (2001)
11. Zaina, L.A.M., Alvaro, A.: A design methodology for user-centered innovation in the software development area. J. Syst. Softw. **110**, 155–177 (2015)
12. Mayhew, D.J.: The usability engineering lifecycle. Morgan Kaufman, San Francisco (1999)
13. Brhel, M., Meth, H., Maedche, A., Werder, K.: Exploring principles of user-centered agile software development: A literature review. Inf. Softw. Technol. **61**, 163–181 (2015)
14. ISO: ISO 13407 - Human-centred design processes for interactive systems. Ergonomics (1999)
15. Vredenberg, K., Isensee, S., Righi, C.: User-Centered Design: An Integrated Approach with Cdrom. Prentice Hall PTR, Upper Saddle River (2001)
16. Norman, D.A., Draper, S.W.: User Centered System Design: New Perspectives on Human-Computer Interaction. L. Erlbaum Associates Inc., Hillsdale (1986)
17. Kushniruk, A.: Evaluation in the design of health information systems: application of approaches emerging from usability engineering. Comput. Biol. Med. **32**, 141–149 (2002)
18. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Agile Manifesto. http://www.agilemanifesto.org
19. Hansson, C., Dittrich, Y., Gustafsson, B., Zarnak, S.: How agile are industrial software development practices? J. Syst. Softw. **79**, 1295–1311 (2006)
20. Papadopoulos, G.: Moving from traditional to agile software development methodologies also on large, distributed projects. Procedia - Soc. Behav. Sci. **175**, 455–463 (2015)
21. VersionOne: 9th Annual State of Agile Survey (2015)
22. Beck, K., Andres, C.: Extreme Programming Explained. Addison Wesley, Pearson Education, Reading, Upper Saddle River (2005)
23. Schwaber, K., Beedle, A.: Agile Software Development with SCRUM. Prentice- Hall, Upper Saddle River (2002)
24. Cockburn, A.: Crystal Clear: A Human-Powered Software Development Methodology for Small Teams. Addison-Wesley, Reading (2001)

25. Highsmith, J.: Agile Software Development Ecosystems. Addison-Wesley Longman Publishing Co., Boston (2002)
26. Coad, P., Palmer, S.: Feature-Driven Development. Prentice Hall, Englewood Cliffs (2002)
27. Sohaib, O., Khan, K.: Integrating usability engineering and agile software development: A literature review. In: 2010 International Conference on Computer Design and Applications (ICCDA), pp. V2-32–V2-38 (2010)
28. da Silva, T.S., Martin, A., Maurer, F., Silveira, M.: User-centered design and agile methods: a systematic review. In: 2011 Agile Conference (AGILE), pp. 77–86 (2011)
29. Fox, D., Sillito, J., Maurer, F.: Agile methods and user-centered design: how these two methodologies are being successfully integrated in industry. In: Agile 2008 Conference, pp. 63–72 (2008)
30. Blomkvist, S.: Towards a model for bridging agile development and user-centered-design. In: Seffah, A., Gulliksen, J., Desmarais, M.C. (eds.) Human-Centered Software Engineering — Integrating Usability in the Software Development Lifecycle. Human-Computer Interaction Series, pp. 219–244. Springer, Heidelberg (2006)
31. Chamberlain, S., Sharp, H., Maiden, N.A.M.: Towards a framework for integrating agile development and user-centred design. In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 143–153. Springer, Heidelberg (2006)
32. Salah, D., Paige, R.F., Cairns, P.: A systematic literature review for agile development processes and user centred design integration. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 5:1–5:10. ACM, New York (2014)
33. Teixeira, L., Ferreira, C., Santos, B.S., Saavedra, V.: Web-enabled registry of inherited bleeding disorders in Portugal: conditions and perception of the patients. Haemophilia **18**, 56–62 (2012)
34. Teixeira, L., Saavedra, V., Simões, J.P.: Dashboard to support the decision-making within a chronic disease: a framework for automatic generation of alerts and KPIs. In: Magdalena-Benedito, R., Soria-Olivas, E., Martínez, J.G., Gómez-Sanchis, J., Serrano-López, A.J. (eds.) Medical Applications of Intelligent Data Analysis, pp. 160–171. IGI Global, Hershey (2012)
35. Teixeira, L., Saavedra, V., Ferreira, C., Sousa Santos, B.: Improving the management of chronic diseases using web-based technologies: an application in hemophilia care. In: Proceedings of the Conference on IEEE Engineering in Medicine and Biology Society, vol. 106, pp. 2184–2187 (2010)
36. Teixeira, L., Ferreira, C., Santos, B.S., Martins, N.: Modeling a web-based information system for managing clinical information in hemophilia care. In: International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 2610–2613 (2006)
37. Teixeira, L., Ferreira, C., Santos, B.S.: User-centered requirements engineering in health information systems: a study in the hemophilia field. Comput. Methods Programs Biomed. **106**, 160–174 (2012)
38. Teixeira, L., Ferreira, C., Santos, B.S.: Using task analysis to improve the requirements elicitation in health information system. In: 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS 2007, pp. 3669–3672 (2007)
39. Teixeira, L., Saavedra, V., Ferreira, C., Simões, J., Sousa Santos, B.: Requirements engineering using mockups and prototyping tools: developing a healthcare web-application. In: Yamamoto, S. (ed.) HCI 2014, Part I. LNCS, vol. 8521, pp. 652–663. Springer, Heidelberg (2014)