Step Towards m-Health in Pediatrics

Rui Oliveira\textsuperscript{a}, Simão Frutuoso\textsuperscript{b}, José Machado\textsuperscript{a}, Manuel Santos\textsuperscript{c}, Filipe Portela\textsuperscript{c} and António Abelha\textsuperscript{a,*}

\textsuperscript{a}Universidade do Minho, CCTC, Braga, Portugal  
\textsuperscript{b}Centro Hospitalar do Porto, Oporto, Portugal  
\textsuperscript{c}Universidade do Minho, ALGORITMI, Guimarães, Portugal

Abstract

Together with a pediatrician, it was identified the necessity of a mobile tool to support clinical practice and decision. To develop it, there were used several elicitation and prioritization techniques. It was also done a study of the mobile development technologies. In order to develop quality software it is necessary to grasp the users and project sponsor’s needs. Prioritizing requirements is also important, since most projects have more requirements than those that are possible to develop. The choice of the right development framework is essential, and it depends on the results of the previews steps and on the available technologies.

© 2013 The Authors Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
Selection and/or peer-review under responsibility of SCIKA – Association for Promotion and Dissemination of Scientific Knowledge

Keywords: elicitation; prioritization; mobile health; mobile frameworks; cross-platforms.

* Corresponding author. Tel.: +351-253604430; fax: +351-253604471.  
E-mail address: abelha@di.uminho.pt
1. Introduction

The development of the human body in early ages is done through very different phases. That’s why the World Health Organization created the growth curves of size, weight, body mass index, head circumference and blood pressure. It’s difficult for pediatricians to know every “normal” value for every age.

Together with a pediatrician working in the Santo António Hospital, in the city of Porto, it was identified the necessity of a mobile tool to support clinical practice and decision. This tool is going to aid pediatricians to offer patients the best practice in the most common situations.

In order to develop the pediatrics mobile application, it was used the requirements elicitation and prioritization processes. Both processes are related with a wider subject called Requirements Engineering. Requirements engineering on the project level is the process by which the requirements for a software project are gathered, documented and managed throughout the software development lifecycle. Software requirements are the critical determinants of software quality, given empirical studies showing that errors in requirements are the most numerous in the software lifecycle and also the most expensive and time-consuming to correct [1].

A survey conducted with 350 USA organizations revealed that about half of the managers interviewed identified poor requirements as a major source of problems, along with other factors such as low user involvement and unclear objectives. In one healthcare related case study, it was observed that there was a huge gap between the daily operations of a hospital and the software developer’s domain knowledge of these operations, though every year healthcare organizations spend large amounts of money and resources on IT systems. The authors of the study argue that this gap is due to insufficient requirements gathering and misunderstanding requirements due to the lack of domain knowledge.

In order to deliver high quality software systems on time and on budget it is essential to have properly structured and controlled requirements specifications that are understandable, comprehensive and consistent. It is important to have a good understanding of stakeholder goals and ensure their involvement in the requirements engineering process [1].

In the development of the pediatrics mobile application, there are going to be addressed two disciplines related to the requirements engineering: requirements elicitation and requirements prioritization. The requirements gathering used a set of elicitation techniques that are presented in detail in section 2. To know what requirements should be implemented first, it was used a requirements prioritization technique, presented in section 3. Finally, it is also important to select the best available framework for the project. That is only possible by the analysis of the current technologies when the gathering and prioritization of the technologies are already done, and it is presented in section 4.

2. Requirements Elicitation

Requirements elicitation is the process of seeking, uncovering, acquiring, and elaborating requirements for computer based systems. It is a complex process involving many activities with a variety of available techniques, approaches, and tools for performing them [2].

The elicitation of requirements begins in an initial stage of a project and is considered to be a critical part of the development process of a software system. These requirements may be found in a variety of sources, such as: stakeholders, documentation and other existing systems. Eliciting the right requirements is considered a vital but difficult part of software development projects.

In reality requirements elicitation is an activity that relies heavily on the communication skills of requirements engineers and the commitment and cooperation of the system stakeholders. One of the major
problems that were identified in the software development project teams is the communication barriers and agreement about the requirements.

There are several elicitation techniques available to use. The choice of a elicitation technique(s) for a specific project depends on a variety of factors including time and cost, the availability of resources, the safety criticality of the system, and any legal or regulatory constraints.

A substantial part of elicitation is dedicated to uncovering, extracting, and surfacing the wants of the potential stakeholders. It is a process, in which a set of requirements is gathered. The number of requirements will probably be bigger than initially expected. In an initial stage it is better to gather a lot of requirements than just a few. After the elicitation of the requirements there are other techniques to filter the requirements, such as: requirements prioritization.

The requirements elicitation process has several typical activities, such as: Understanding the application domain, identifying the sources of requirements, analyzing the stakeholders and selecting the Techniques to use and eliciting the requirements from stakeholders and other sources.

To understand the Application Domain it is important to investigate and examine in detail the situation in which the system will reside. It is necessary to explore the current environment, so it is possible to identify any existing constraints [2].

Requirements may be spread across many sources and exist in a variety of formats. In all software development projects a number of possible sources for requirements may be identified. Stakeholders represent the most obvious source of requirements for the system. Other sources of information may be: subject matter experts, existing systems and existing documentation.

It is also necessary to analyze and find out the range of stakeholders of the project. Stakeholders are people who have an interest in the system or are affected in some way by the development and implementation of the system. The customer, and more specifically the project sponsor, is usually the most apparent stakeholder of the system. In some cases however the actual users of the system may be the most important.

It is generally accepted that an individual requirements elicitation technique or approach cannot possibly be suitable for all projects. The choice of techniques to be employed is dependent on the specific context of the project and is often a critical factor in the success of the elicitation process. Clearly requirements elicitation is best performed using a variety of techniques. In the majority of projects several methods are employed during and at different stages in the software development life cycle, often in cooperation where complementary [2].

After the identification of the requirements sources and specific stakeholders, the actual elicitation of the core requirements then begins using the selected elicitation techniques, approaches, and tools. During this activity it is important to investigate in detail the needs and wants of the stakeholders, especially the users, as well as to determine the future processes the system will perform with respect to the business operations.

The requirements elicitation process is always limited by several aspects, such as: specific characteristics of the project, organization and environment, the budget or the schedule are some of the variables that directly affect the requirements elicitation. In reality its completion is often determined by time and cost constraints rather than achieving the required level of requirements quality and completeness. Typically the process begins with an informal and incomplete high-level mission statement for the project. This may be represented by a set of fundamental goals, functions, and constraints for the target system, or as an explanation of the problems to be solved. The result of this process forms the basis of further investigation and refinement of requirements in a typically iterative and incremental manner. At the end of the requirements elicitation, there normally is a set of detailed requirements in natural language text and simple diagrammatic representations with additional information including descriptions of the sources, priorities and rationales.
Frequently requirements engineers are responsible for documenting the requirements elicited. This role is particularly important as it represents the production of results from the elicitation process, and forms the foundation for the subsequent project phases. Evaluation of the elicitation process and the work performed by the analyst is based on these resultant artifacts, which in some cases may form the basis of contractual agreements.

The Volere is a requirements specification template that represents the most basic type of tool used by analysts to support the process of requirements elicitation [2]. A requirements specification template makes the requirements writing easier to do and more convenient. Volere is a template that sets out a complete description of the product's functionality and capabilities. It is a distillation of literally hundreds of requirements specifications and is currently used by thousands of organizations all over the world [3].

2.1. Requirements elicitation techniques

Following it is presented a set of techniques used for requirements elicitation nowadays. These techniques are only a small portion of all the available techniques; however these techniques are some of the most commonly used.

2.1.1. Interviewing

When interviewing, the requirements engineering team puts questions to stakeholders about the system that they currently use and the system to be developed. Requirements are derived from the answers to these questions. Most requirements engineering processes use this technique, which can take two forms: formal and informal. The formal ones are closed interviews, where the stakeholder is asked a pre-defined set of questions. In the other hand, informal interviews are made of a set of open questions. The requirements engineering team explores a range of issues with system stakeholders and hence develop a better understanding of their needs [4].

In practice, interviews with stakeholders are normally a mixture of both of these. You may have to obtain the answer to certain questions but these usually lead on to other issues that are discussed in a less structured way. Completely open-ended discussions rarely work well.

Interviews are good for getting an overall understanding of what stakeholders do, how they might interact with the new system, and the difficulties that they face with current systems. However, interviews are not so helpful in understanding the requirements from the application domain. Mainly, the difficulty in eliciting domain knowledge with interviews is due to the fact that application specialists use terminology and jargon that are specific to a domain and therefore difficult to grasp; and that stakeholders may not be able to easily explain a specific part of the domain knowledge or even mention it at all.

Interviewing on its own is liable to miss essential information and so it should be used in conjunction with other requirements elicitation techniques [4].

2.1.2. Domain analysis

Domain analysis consists in the examination of the existing and related documentation and applications. It is a very useful way of gathering early requirements as well as understanding and capturing domain knowledge, and identifying reusable concepts and components. These types of investigations are particularly important when the project involves the replacement or enhancement of an existing legacy system [2].

Some examples of documentation for domain analysis are design documents, instruction manuals, hardcopy forms and files used in the business process; while the application studies often include looking at both upstream and downstream systems, as well as competitive or like solutions.
Analogies and abstractions of existing problem domains can be used as baselines to acquire specific and detailed information, identify and describe possible solution systems, and assist in creating a common understanding between the analyst and stakeholders.

This elicitation technique and its results are often used in conjunction with, and as the input to other elicitation techniques [2].

2.1.3. Apprenticing

Apprenticing involves the analyst actually learning and performing the current tasks under the instruction and supervision of an experienced user. In this technique the analyst is taught the operations and business processes by observing, asking questions, and physically doing, rather than being informed of them. Similar to Role Playing but more involved, apprenticing is very useful when the analyst is inexperienced with the domain, and when the users have difficulty in explaining their actions [2].

2.1.4. Questionnaires

Questionnaires are mainly used during the early stages of requirements elicitation and may consist of open and/or closed questions. To be effective, the terms, concepts, and boundaries of the domain must be well established and understood by the participants and questionnaire designer. Questions must be focused to avoid gathering large amounts of redundant and irrelevant information. They provide an efficient way to collect information from multiple stakeholders quickly, however they are limited in the depth of knowledge requirements engineers are able to elicit. In the same way they provide no mechanism for the participants to request clarification or correct misunderstandings [2].

3. Requirements Prioritization

Requirements Prioritization is a discipline that concerns the prioritization of requirements of a given project, with the aim of developing the functionality that is most desired by the customers, as well as least risky, least costly, and so forth. Prioritization is a crucial step to make good decisions regarding product planning for single and multiple releases. The quality of a software product is often determined by the ability to satisfy the needs of the customers and users. Hence, eliciting and specifying the correct requirements and planning suitable releases with the right functionality is a major step towards the success of a project. If the wrong requirements are implemented and users resist using the product, it does not matter how solid the product is or how thoroughly it has been tested [5].

In this section, it is introduced a set of important concepts when performing prioritizations, such as: prioritization techniques, several aspects to consider in the prioritizations, among other issues.

The result of prioritizations suggests which requirements should be implemented. Hence, the prioritization techniques could be a valuable help to get an understanding of what is important and what is not for a given project.

Decision-making becomes a complex task to perform when the number of different aspects to consider starts growing. A decision that is easy to do when only considering one aspect gets tougher if a few more aspects are also to be considered. For example, the functionality that is most important for the customers might not be as important when other aspects (e.g. price) are also considered.

Most software projects have more candidate requirements than can be released within the time and cost constraints. Prioritization helps to identify the most valuable requirements from this set by distinguishing the critical few from the trivial many. Some activities requirements prioritization provides support are:

- In the stakeholder’s decision on the core requirements for the system;
In the selection of only a subset of the requirements and still produce a system that will satisfy the customer(s);
In establishing relative importance of each requirement to provide the greatest value at the lowest cost. The resulting prioritization might be used as a guide throughout the project [5].

3.1. Aspect of Prioritization

When prioritizing requirements there are many aspects that may be taken into account. An aspect is a property or attribute of a project and its requirements that can be used to prioritize requirements, such as: penalty, cost and risk. When prioritizing requirements based on a single aspect, it is easy to decide which one is most desirable. If there are other aspects to be considered, such as cost, customers can change their mind and high priority requirements may turn out to be less important. Often, the aspects interact and changes in one aspect could result in an impact on another aspect. Several aspects can be prioritized, and the decision of which aspects to consider depends on each specific project [5]. Below it is presented a list of the aspects that were considered to be important in the pediatrics mobile application.

3.1.1. Importance

When considering importance, the stakeholders are expected to prioritize the most important requirements for the system. However, the term importance may mean different things for different stakeholders. Importance could, for example, be urgency of implementation, importance of a requirement for the product architecture, strategic importance for the company, etc. Hence, it is necessary to specify the meaning of the word importance, or the different kinds of importance that should be considered [5].

3.1.2. Cost

The implementation cost is usually estimated by the developing organization. The cost is usually measured by: complexity of the requirement, the ability to reuse existing code, the amount of testing and documentation needed, etc. Cost is often expressed in spent hours by the staff (effort) since the main cost in software development is often related to the necessary time [5].

3.1.3. Risk

Risk is part of every project, and may be divided in internal (technical and market risks) and external risks (e.g. regulations, suppliers), both taken in risk management account. Their probability and impact must be considered when evaluating the project tasks. Risk management is also useful for identifying possible problems of desirable requirements for a system that could arise in the future. Such risks could for example include performance risks, process risks, schedule risks etc. Based on the estimated risk likelihood and risk impact for each requirement, it is possible to calculate the risk level of a project [5].

3.2. Using multiple aspects

There should be considered multiple aspects when prioritizing requirements. It is possible to combine different aspects in many different ways, and the better choice depends on the specific situation.

In the Cost-Value approach there are considered the value (importance) and cost aspects resulting a prioritized requirements that give most value for the money. The Planning Game (PG) from eXtreme Programming (XP) uses a similar approach when importance, effort (cost), and risks are prioritized. In Wiegers’ approach, the relative value (importance) is divided by the relative cost and the relative risk in order
to determine the requirements that have the most favorable balance of value, cost, and risk. This approach further allows different weights for different aspects in order to favor the most important aspect (in the specific situation) [5].

3.3. Prioritization Techniques

The prioritization techniques have the objective of establishing a relative order between different requirements. There are several techniques available for this purpose, some more powerful than others.

Some of the techniques assign a priority value to each of the requirements, while other techniques assign a priority values to a group of requirements [5]. Below it is presented the prioritization technique used for the development of the pediatrics mobile application.

3.3.1. 100-Dollar Test

The 100-dollar test is a very straightforward prioritization technique where the stakeholders are given 100 imaginary units (money, hours, etc.) to distribute between the requirements. The result of the prioritization is presented on a ratio scale.

In spite of being a straightforward technique, there were identified a few problems that it is necessary to be aware of. If there are too many requirements it was found that the 100 imaginary units may be scarce to prioritize all the requirements. In these situations, it is suggested to give the stakeholder a larger amount of units (e.g. 100,000 units). Another problem that may happen is that at the end of the prioritization the total sum is not equal to 100 units, as a consequence of a miscalculation. This can be prevented by using a tool that keeps count of how many points have been used. The prioritization process should be done only once, since stakeholders may change their evaluation if they don’t get one of theirs favorite requirements as a top priority. The same can happen if a stakeholder decides to spend all his credits in his favorite requirements, knowing that some other necessary requirements will get credits anyway. A possible solution may be limiting the amount of units that could be spent on a single requirement, although it may prevent the stakeholder from genuinely prioritize the requirements [5].

4. Analysis of current technologies

When looking into the smartphones market, it is not difficult to spot an operating systems fragmentation issue. Making an analogy with the computers market it becomes clearer. After 20 years of Personal Computers existence only two operating systems accounted for the overwhelming majority: Microsoft Windows and Macintosh’s Mac OS. On the other hand, after nearly 20 years of mobile devices existence, there are too many different operating systems available with a considerable market share [6]. When an organization or a group of programmers decide to develop an application for a specific operating system (e.g. Android), their final product will inevitably reach a limited amount of the existing market smartphones.

There are several smartphone operating systems available on the market, such as: Android, iOS, Windows Phone, BlackBerry, Symbian OS, etc.

There is also no sign of operating systems reduction over time. Recent news show that several worldwide recognized technology organizations have been working in releasing their own smartphone operating systems, in an attempt of establishing themselves in this market. Some examples are: Ubuntu for phones [7], Firefox OS [8] and Tizen [9].
In order to reach most smartphone users, there are two possible approaches: the first is to develop a native application for each of the dominant operating systems; the second is to use a cross-platform development framework in order to develop the application once, and deploy it to several operating systems.

In the pediatrics application project, it is intended to release a piece of software that may run on the majority of the smartphones on the market. As a result it is going to be used a cross-platform development framework in order to shorten the development time.

4.1. Cross-platform development frameworks

In the past few years, many cross-platform frameworks have emerged. There has been an explosion of activity in this area as mobile devices become faster and more widely adopted, and particularly with a fast-growing market for applications [10]. Cross-platform development frameworks are intended to write an application once, and deploy it to several operating systems. This task may sound simple in theory, but is in fact highly complex, technically, due to a range of factors such as the highly fragmented mobile technology landscape, rapidly evolving standards, limitations imposed by the mobile devices themselves (screen size, input methods, display capabilities, etc.) and also constraints of the mobile network such as high latency and low bandwidth.

As each vendor implements its own application development stack, achieving cross-platform and cross-device consistency is a non-trivial task. Fortunately as the web becomes ubiquitous and its technologies evolve, with more and more mobile browsers implementing new standards like HTML 5, CSS 3 and JavaScript, web applications are rapidly becoming an attractive and cost-efficient way of developing mobile applications that can rival native apps in terms of rich user experience and access to advanced capabilities like storage and geo-location [11].

The frameworks fall into two categories: those that let you create a native mobile application using cross-platform APIs, and HTML/CSS/JavaScript frameworks that let you build cross-platform interfaces that run in a web browser. It is common practice to combine these to create cross-platform native applications. Some examples of native cross-platform frameworks are Rhodes, PhoneGap and Titanium; while some HTML/CSS/JavaScript frameworks are Sencha, jqTouch, and iWebKit [10].

When performing a deeper analysis of each of the frameworks available, there are several limitations that indicate cross-platform frameworks still have several important improvements to make. Which technology to use in a given project is a decision that only should be made after knowing the purpose of the application, what functionalities the final system is going to have and the development team. Some technologies are open source like PhoneGap, while others are property frameworks, like Corona. There are some frameworks that are more likely to be adopted by web programmers, since they provide JavaScript APIs and allow the use of HTML and CSS (e.g. PhoneGap and Sencha Touch). The Operating Systems that it is intended to deploy the final application may also limit the choice of the framework (e.g. with Corona it is not possible to deploy to Windows Phone) [10]. If the programmers already have experience with programming languages like C/C++ or Ruby, then frameworks like MoSync and Rhodes may be a better choice since it becomes possible to take advantage of those programming languages. Another limitation may be the ability of the framework to take advantage of device capabilities. For example: The PhoneGap core APIs do not incorporate any Bluetooth capabilities, in spite of being available on the web several plugins developed by third parties [11].

Taking a look specifically at the pediatrics mobile application, it is known that there are going to be, for example, several calculations involving dates. Some frameworks have limitations when it is necessary to work with calendars, such as: PhoneGap and Titanium. PhoneGap core APIs don’t incorporate calendar functionalities in spite of already being available third party plugins for Android and iOS, while the Titanium
framework only provides calendar capabilities for iPhone and iPad devices [12] [13]. As it is intended to deploy the pediatrics application for at least Android, Apple iOS and Blackberry both of the above frameworks should not be used for its development.

5. Conclusion

In order to deliver a quality software product, it is necessary to understand and prioritize its requirements. Several studies concluded that a lot of the problems found in the development of software project are related with poor requirements. The processes of gathering and prioritizing the requirements were already studied by the requirements engineering discipline, and are known as: requirements elicitation and requirements prioritization. Requirements elicitation is the process of seeking, uncovering, acquiring, and elaborating requirements. Its process is composed by several activities, such as: understanding the application domain, identifying the sources of requirements, analyzing the stakeholders, selecting the techniques to use and eliciting the requirements from stakeholders and other sources. It is considered to be a complex but critical part of the development process of a software system [1]. The prioritization process is concerned with the prioritization of the previously gathered requirements given a set of aspects that characterize a specific project (e.g. importance and costs). The quality of a software product is frequently associated with its capacity to satisfy the needs of its customers and users. As most software projects have more gathered requirements than those that can be developed within the project time and cost constraints, requirements prioritization becomes imperative [5].

The smartphones operating systems fragmentation becomes an issue for those who pretend to release a product for the majority of the devices available on the market. Unlike what happened with Personal Computers, Smartphones have a large number of operating systems and there is no sign of its reduction. One possible solution is the adoption of a cross-platform development framework in order to develop an application once and deploy it to several operating systems. However, the choice of the framework must be done carefully. These frameworks have their limitations, which if not identified on time may compromise the whole project.

References