Collaborative Working: Understanding Mobile Applications Requirements

Lizbeth Gallardo-López, Beatriz A. Gónzalez-Beltrán, Roberto García-Madrid
Marco Ferruzca, Irma A. Zafra-Ballinas† and José A. Reyes-Ortíz
Universidad Autónoma Metropolitana, Azcapotzalco Campus
Av. San Pablo No. 180, Col. Reynosa Tamaulipas, Del. Azcapotzalco, C.P. 02200, México, D.F.
Email: {glizbeth, bgonzalez, grma, mvfn, azb, jaro}@correo.azc.uam.mx

***Abstract*—The Rational Unified Process (RUP) as well as the related work embrace the importance of collaborative working teams during software development; but, user-interface designers and system analysts work in parallel or in sequential mode. However, this kind of relationship may not be effective, resulting on functional software but not meeting usability issues. Our proposal is that in order to understand mobile applications requirements work should be made collaboratively between analysts and user-interface designers through sharing artifacts like use-case scenarios, sketching and mock-up. In this paper, we propose a collaborative work framework to meet mobile applications requirements. Also, we show preliminary results of a case study to assess this approach. Results suggest that the collaborative team got a common understanding about system limits and functional and usability requirements.**

I. Introduction

The Rational Unified Process (RUP) is a software development process, which is divided into four phases: Inception, Elaboration, Construction and Transition [2]. In this phases, working teams loop under six sets of activities called process workflows: Business Modeling, Requirements, Analysis and Design, Implementation, Test and Deployment. This paper focuses on requirements workflow, under the Inception and Elaboration phases; the artifacts that must be generated in this workflow are: Vision document, Supplementary specification document, Use-case model and Glossary.

The RUP considers different working teams: the system analyst team, the user-interface team and the technical team. Inside each team work is made collaboratively but each team work in a sequential way [2]. Thus, the system analyst team propose a system specification in the Elaboration phase; the user-interface design team propose a user-interface model or prototype during Inception phase or in the beginning of the Elaboration phase; and the technical team propose the code of the system during the Construction phase [6]. According to Obendorf and Finck [4] "this sequential division may affect the resulting software product so that it is functional but not very usable".

We consider that RUP proposes a good process for the development of mobile applications because it clearly states phases and workflows through an incremental and iterative development. Moreover, it specifies the artifacts that are part of the project memory easing the communication between working teams. However, the sequential work between user-interface designers and system analysts hinder the mobile software development with a strong usability component. Therefore, we propose to use RUP as a software development process but promoting collaborative work between user-interface designers and system analysts to understand mobile applications requirements through sharing artifacts like use-case scenarios, sketching and mock-up.

We believe that this collaborative work allow to develop functional mobile applications that takes into account usability issues, particularly interaction and navigation. Collaboration must be guided by a framework organizing a team and planning a serie of meetings in the requirements workflow, mainly in the Inception and elaboration phases. We think that use-case scenarios foster functional requirements but that sketching and mock-ups are artifacts that foster creativity and facilitate discussion among user-interface designers and system analysts, allowing them to have a more accurate system model and to have a quick idea about the interaction and navigation of the user interface.

In the following sections, we present some related work, section II. Later, we explain our proposal, section III. Also, we show preliminary results, section IV. Finally, we describe conclusions and future work of this research, section V.

II. Related work

This section surveys previous work in software development methodologies to find how collaboration work is done. Three features will be highlighted: the methodology approach, the level of collaboration between working teams and the artifacts applied.

Song et al [6] explore a methodology of designing user interfaces of handheld devices based on the Android platform. The method was divided into three parts from the perspective of the task carried out by three major roles in different stages of development: 1) requirements analyst, 2) UI designer, and 3) software engineer. The methodology proposed is role and task oriented and each part is made in chronological order. However, it is not clear which artifacts are built in each stage.

Rahimian and Ramsin [5] propose an hybrid methodology for mobile software development as an agile risk-based approach and influenced by the Adaptive Software Development Methodology and new Product Development. This work proposes the following phases: 1) idea generation, 2) project

initiation, 3) detailed analysis with prototyping, 4) architectural design, 5) development, and 6) commercialization. This methodology is oriented towards product development. As they incorporate ideas from Adaptive Software Development, they argue that collaboration is needed for unpredictable parts of the project, but it is not specified the level of collaboration. Although they propose prototyping within detailed analysis, it is not clear which artifacts are built.

Obendorf and Finck [4] propose an integrated process combining Extreme Programming and Scenario-Based Usability Engineering. They adopted techniques from Scenario-Based Engineering [3] and Rapid Contextual Design [1]. They argue that the use of use-case scenarios can induce the beneficial effect of connecting development task to the use context, and thus indirectly even decisions concerning software architecture to the question of how to best support documented use scenarios. Contextual investigation was a method for gathering information about the use context and users roles and responsibilities. Requirements scenario was a technique for collecting requirements. Use scenario was an artifact resulting of mixing activities with sketches.Their notion of stories included paper prototyping in the re-design process. Tasks were used to differentiate between sequential development tasks, but also to divide the labor between different teams.

III. Collaborative-working:Understanding mobile applications requirements

A. Research questions and Findings

Our research question is: The collaborative work between user-interface designers and system analysts, through the use-case scenarios, sketching and mock-up, allow a common understanding of requirements in order to develop useful mobile applications?

The findings are: The RUP as well as the related work, embrace the importance of collaborative working teams during software development; however, system analyst and user-interface teams work in parallel or sequential mode. We observe that the related work support sketching and mockup as useful artifacts in the software design. However, we have not found a proposal where team members work in a closed collaboration.

We hold that the collaboration will allow teams to understand the problem, the system limits and will promote the building of a common functional and usability vision about mobile applications requirements. We believe that the collaborative work through sharing artifacts facilitate the generation of ideas; also, we think that sketching and mock-up help to define better use-case scenarios, because the feedback between system analysts and user-interface designers is provided immediately.

B. Framework

In order to establish a collaborative work between system analysts and user-interface designers, we propose a framework based in three concepts: team, working sessions and rules.

1) Team: The working team is formed by persons which roles are: i) System analyst that build the following artifacts: vision, suplementary-specification, glossary, use-case diagram and use-case scenarios; and ii) User-interface designer that build artifacts: sketching and mock-up. The user is an important role in the development process, but we do not consider him part of the team.

Each team must name a coordinator: i) a system analyst coordinator ii) An user-interface designer coordinator. Each coordinator assign resposabilities to their members; also he chairs working sessions.

2) Working sessions: The goal of working sessions is that the working team build collaboratively a solution, sharing artifacts that describe the functional requirements and usability issues, particularly interaction and navigation.

We propose three kinds of working sessions in requirements workflow. We recommend to assign an hour and a half maximum to a working session, in order to capture the best performance of team members and users. Also, we recommend defining an iterative process (loop) by a week (five days) to mobile applications.

a) Meeting: The team and users work together. Meetings serve to capture the user requirements and validate the functional requirements.

- Capture user requirements. The team employs interviews to the users.
- Validate the functional requirements. The team employs visual elements to communicate with the user, like: use-case diagram, sketching and mock-up. However, they should have at work artifacts like: use-case scenarios, vision and suplementary-specification.

In a meeting, team members take note of users comments and they sketch ideas that will be taken up in the next workshop, or that will be assigned to a team member.

b) Workshop: The working team makes analysis of interviews, notes and sketches that have been taken in a meeting, and defines a new set of tasks. Each coordinator assigns tasks that will be performed in pairs or individually. The working team starts these tasks in the workshop.

The working team explores potential solutions using artifacts constructed previously in a workshop or individually, and builds one or several proposals, which are expressed in a new version of artifacts.

c) Individual work: System analysts and user-interface designers work individually in an artifact. This task could be assigned in a workshop or in a meeting.

3) Rules: Two important rules to team members: i) The responsible to make an artifact must share it, at least one day before a meeting or a workshop, and ii) The others team members, users included, must review it before the next meeting or the next workshop.

IV. Results

This research project is ongoing. We perform a case study with the MOCA system, in order to test this framework. MOCA will be a system for cancer symptoms monitoring.

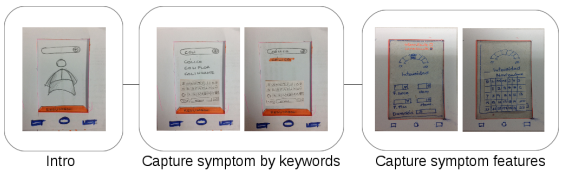

Fig. 1. MOCA system: First sketching to register symptom

It is defined by use-cases: i) register a symptom, ii) modify a symptom, iii) remove a symptom, and iv) view symptoms summary. From this case study, we answer partially our research question: The collaborative work between user-interface designers and system analysts, through the use of shared artifacts, allow a common understanding of requirements in order to develop useful mobile applications? We contrast versions of artifacts: use-case scenario and sketchig about "Register symptom".

A. Team and working sessions

1) Team: The working team was formed by three system analysts and three user-interface designers. Team attended two users: patients diagnosed with cancer. The working team defined an iterative process (loop) by a week (five days). In the inception phase, the team performed two loops and in the elaboratio phase team perform three loops.

2) Working sessions: A loop included: a workshop at Monday, individual work at Tuesday and Wednesday, workshop at Thursday, if necessary, and a meeting at Friday. The time assigned to working sessions was one hour and a half.

The table I shows the initial use-case scenario "Register symptom". Team and users found some disadvantages:

1) This use-case scenario does not describe clearly the functionality. Team asks: Does the list of symptoms include all the symptoms registered in the application?
2) This use-case scenario does not offer a functional alternative in order to search a symptom. For example, searching a symptom by typing it on the keyword or by choosing words in a dictionary (ABC).

The sketch (see figure 1) shows an idea oriented to search a symptom by keyword. The application filters the symptom through the letters introduced by the user. The team build a mock-up shows the navigation that the user must follow in order to register a symptom. Team and users observed some disadvantages of this proposal:

1) When the user chooses a symptom, he must specify the symptom features: i) intensity, ii) initial date and iii) initial hour, iv) final date and v) final hour, and vi) behavior. This is not suitable. First, the user may be feel sick at that moment. Second, the user still does not know date and hour finals, nor the symptom behavior.
2) User is forced to interact with MOCA in a window-like environment. This design does not provide opportunities to employ touch screens because several clicks are needed before register a symptom.

TABLE I
INITIAL USE-CASE

Use-case: Register symptom
Summary: This use-case describes the option to register a symptom.
Actors: Pacient
Pre-conditions: The pacient must be recorded and must be autenticated in the system.
Post-condition: A new symptom was recorded in the system.
Trigger: Registering a symptom start when the pacient chooses the option "new symptom".
Principal flow: 1) The system shows the symptom list. 2) The patient chooses one symptom. 3) The patient chooses the option "symptom features" 4) The system shows a formulary to register symptom features. 5) The patient registers: a) Initial date. b) Initial hour. c) Symptom intensity (range 1-10). d) Final date. e) Final hour. f) Behavior (constant, incremental or decremental). 6) The patient chooses the option "ok" 7) The system shows a symptoms summary. 8) The system habilitates the option "save". 9) The patient chooses the option "save". 10) The system saves symptom features. 11) The system returns to the main view.
Alternative flow: a. The patient can in any moment of their edition to cancel the record. 5.1 The patient did not register symptom features. 5.2 The patient chooses the option "ok" 5.3 The system notifies to patient "The symptom features are not be registered". The use-case continues to step 8 of main flow.

The sketch (see figure 2) shows another idea oriented to search a symptom through the human anatomy. The team build a mock-up to show the navigation that the user must follow to register a symptom. The user must choose one anatomical region, for example, the stomach. The application respond with a list of symptoms associated with-it. When the user choose a symptom, the intensity is incremental while the finger is slided to the right, and it is decremental while the finger is slided to the left. The application register the following features: i) symptom name ii) intensity iii) date and iv) hour of these moment. The team and users observed some advantages of this interface:

1) When the user chooses a symptom, he must only specify the intensity. The others symptom features will be specified as from the symptoms summary, where the features, excluding symptom name, will be modified.
2) The interface where the user must choose the symptom features is now a slided panel, which head shows a pictogram associated to each feature. Under the panel, the system shows the list of symptoms recorded. A symptom in blue color indicates that the symptom is completed. A symptom in orange color indicates that the symptom is not completed, i.e. lacking features. The final option of the panel is "delete" a symptom.

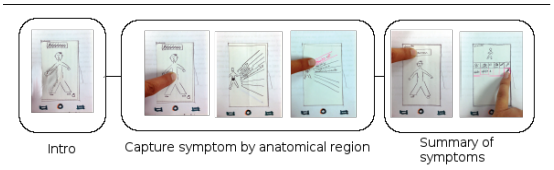

Fig. 2. MOCA system: Second sketching to register symptom

TABLE II
FINAL USE-CASE

Use-case: Register symptom
Summary: This use-case describes the option to register a symptom.
Actors: Patient
Pre-conditions: The patient must be recorded and must be authenticated in the system.
Post-condition:A new symptom was registered in the system
Trigger: Registering a symptom start when the patient chooses the option "new symptom".
Principal flow: 1) The system shows a human anatomy (frontal or behind) by patient gender. 2) The patient chooses an anatomical region. 3) The system shows the list of symptoms associated with this anatomical region. 4) The patient chooses a symptom. 5) The system asks for the "symptom intensity and description". 6) The patient chooses the intensity (range 1-10). 7) The patient writes a brief description of the symptom (optional). 8) The patient chooses the option "ok". 9) The system save the following symptom features: • Name. • Intensity. • Description. • Current date and hour. 10) The system shows a symptoms summary in a emergent window.
Alternative flow: a. The patient can cancel the register at any time. b. The patient can in any moment to choose the option "summary". The use-case View symptoms summary will be triggered. 1.1 The patient chooses the textual view. 1.2 The system shows the list of symptoms. 1.3 The system provides two search mode: • The patient searches by dictionary (ABC). • The patient searches by typing letters. 1.4 The system shows symptoms filtered by the search criteria (ABC dictionary or typing letters). The use-case continues in the step 5 of the main flow.

3) This interface employs gestures provided by a mobile device.

The use-case scenario modified after the collaborative work is showing in the table II). Team observed the following advantages:

1) This use-case scenario describes details about expected functionalities. The list of symptoms shows only the symptoms that match to the user search. The use-case scenario describes a filtering operation.
2) This use-case scenario offer three functional alternatives in order to search a symptom: i) Typewriting it, ii) dictionary (ABC), and iii) choosing from anatomical region.

Results suggest that the team got a common understanding of requirements. This is reveled by use-case scenarios I and II. The system limits were established through working sessions with users, these limits were shown by a use-case diagram. Usability issues like: interaction and navigation were established in sketches and mock-ups, which improved the use-case scenario. Functionalities not considered in the initial version, were reveled by artifacts (sketchig and mock-up) build in a collaborative work.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a framework to collaborative work between system analysts and user-interface designers to understand mobile applications requirements. We have an empirical evidence about the efficiency of collaborative work, through a case study. The preliminary results, based in contrasting first and final artifacts suggest that the team got a common understanding about functional and usability requirements and system limits.

ACKNOWLEDGMENT

The authors would like to thank the support of Universidad Autónoma Metropolitana, Azcapotzalco campus.

REFERENCES

[1] Beyer, H., Holtzblatt, K., Baker, L.: An agile customer-centered method: Rapid contextual design. In: Zannier, C., Erdogmus, H., Lindstrom, L. (eds.) Extreme Programming and Agile Methods - XP/Agile Universe 2004, Lecture Notes in Computer Science, vol. 3134, pp. 50–59. Springer Berlin Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-27777-4_6

[2] Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition). Prentice Hall PTR, Upper Saddle River, NJ, USA (2004)

[3] McGraw, K., Harbison, K.: User-centered Requirements: The Scenario-based Engineering Process. L. Erlbaum Associates Inc. (1997)

[4] Obendorf, H., Finck, M.: Scenario-based usability engineering techniques in agile development processes. In: CHI '08 Extended Abstracts on Human Factors in Computing Systems. pp. 2159–2166. CHI EA '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1358628.1358649

[5] Rahimian, V., Ramsin, R.: Designing an agile methodology for mobile software development: A hybrid method engineering approach. In: Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on. pp. 337–342. IEEE (June 2008)

[6] Song, M., Song, H., Fu, X.: Methodology of user interfaces design based on android. In: Multimedia Technology (ICMT), 2011 International Conference on. pp. 408–411. IEEE (July 2011)