Comparing and Reconciling Usability-Centered and Use Case-Driven Requirements Engineering Processes

A. Seffah, R. Djouab and H. Antunes
Department of Computer Science, Concordia University
1455 de Maisonneuve Boulevard W. Montreal, Quebec, Canada H3G 1M8
Telephone: (514) 848-3024 — Mail: seffah@cs.concordia.ca

Abstract

During the two last decades, the human-computer interaction community has developed a large variety of techniques and tools for gathering, specifying and validating usability requirements including user characteristics, tasks, work environment as well as usability goals such as effectiveness, efficiency and user satisfaction. Unfortunately, even if their importance are accepted by software developers, they are not yet cost-effectively integrated into software engineering methodologies. This paper presents the rationale for our ACUDUC approach by identifying the different issues for enhancing the use case-driven software requirements approach with RESPECT, one of the most advanced frameworks for user-centered requirements. Beyond this specific example (use cases and RESPECT), our investigations aim to reconcile user-centered and use case-driven requirements engineering and to cross-pollinate software engineering and usability engineering.

1. Introduction

Our work investigates how to integrate concerns for usability into the software development lifecycle. Many definitions of usability and frameworks for its engineering exist, making sometimes usability a confusing concept [2, 12, and 4]. According to ISO 9241 and ISO 9126 standards, two different definitions for usability are proposed.

The first one advocates a process-oriented approach to usability, whereby usable interactive systems are achieved as the result of a human-centered design process. Usability as a high-level quality objective is defined in the ISO 9241-11 standard as: "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". ISO 9241-11 support the following activities related to the user-centered design process:

- Specification of usability goal and metrics as well as evaluation against these requirements. (See ISO 9241-11 and ISO/IEC 14598-1)
- Definition of activities necessary in the development lifecycle for achieving quality in use. The standard provides a framework for applying human-centered design and evaluation techniques, and is intended to supplement existing lifecycle models.

In the second one, a product-oriented approach, usability is seen as one relatively independent contribution to software quality. Usability is defined in this way in ISO/IEC 9126 as: "A set of attributes of an interactive system that bear on the effort needed for use and on the individual assessment of such use by a stated or implied set of users". This definition can be used to specify details of the look and feel as well as the behavior of the user interface.

Our proposed ACUDUC (Approach Centered on Usability and Driven by Use Cases) is process-oriented framework that aims to unite:

- The use case-driven requirements process defined in the object-oriented software engineering methodology [8] and recently reengineered as part as of the Unified software development Process (UP) [7], and
- The user-centered requirements process defined in the RESPECT framework (REquirements SPECification in Telematics [11]. The RESPECT process is a concrete
implementation of the ISO 13407 standard on human-centered design process for interactive systems.

2. Background and Related Work

The following investigations show that the philosophy of the use case-driven software development approach [7] is highly compatible with the user-centered requirements techniques of usability engineering. Most of them suggest specific, yet powerful, enhancements to the use case-driven software development approach, particularly in the user requirements and usability specification chapters.

Artim [1] emphasizes the role of task analysis by providing a user-centric view of a suite of applications, and then emphasizes use cases by providing each application with a method of exploring user-system interaction and describing system behavior. Jarke [9] points out that scenarios are used in software engineering as intermediate design artifacts in an expanded goal-driven change process. They provide a task-oriented design decomposition that can be used from many perspectives, including usability trade-off, iterative development and manageable software design object models. Mayhew [12] describes the overall usability engineering lifecycle and highlights some challenges that should be addressed for its effective integration in the object-oriented software engineering approach proposed by Jacobson [8]. Constantine [4] suggests that use case specifiers first prepare lightweight use case model descriptions (essential use cases) that do not contain any implicit user interface decisions. Later on, the user interface designer can use these essential use cases as input to create the user interface without being bound by any implicit decisions. Nunes [13] proposes to annotate use cases using non-functional requirements at the level of abstraction at which they should be considered. Rosson [14] proposes combining the development of tasks and object-oriented models, which are viewed as a refinement of rapid prototyping and an extension of scenario-based analysis. Krutchen [10] introduces the concept of a use case storyboard as a logical and conceptual description of how a use case is provided by the user interface, including the interaction required between the actor(s) (user) and the system.

3. Comparing User-Centered with Use Case-Driven requirements Processes

3.1. Capturing Functional Requirements in UP Process

The goals of the requirements workflow, as defined in the UP, are to describe what the system should do in terms of functionalities, and allow the developers and the customer to agree on this description. This UP workflow offers a systematic and intuitive way for gathering the functional requirements, with a particular focus on the value added to each individual or external system. The main activities described in the use case-driven requirements workflow, part of the unified development process, are as follows [7]:

- Develop the vision document, which identifies the high-level user or customer view of the system to be built. In the vision document, initial requirements are expressed as key features the system must have in order to solve the more critical issues.
- Understand the needs of stakeholders, and future users.
- Structure the use case model
- Detail the use case model
- Model and prototype the user interface
- Prioritize the use cases for implementation

This process captures also the non-functional requirements to a certain extent. This includes user characteristics that cannot be associated with any particular use case. The authors of the Rational UP process suggest that the non-functional requirements presented in the IEEE 610.12.1990 standard can be described in this document [7]. This is one of the major weaknesses of use case-driven requirements that user-centered requirements approaches can substantially improve.

This process is supported by several techniques for gathering requirements such as interviews, questionnaires, brainstorming, use case workshops, storyboarding, role-playing and reviewing existing requirements. The use case diagram is the most important requirements artifact. It is used as starting point for building an object-oriented model. Other requirements artifacts related to usability and the interface aspects are:
Additional requirements, which are primarily non-functional requirements
Use case storyboarding, which is a logical and conceptual description of how use cases are provided by the user interface, including the required interaction between the actor(s) and the system. Storyboards represent a high-level understanding of the user interface, and are much faster to develop than the user interface itself. Use case storyboards can thus be used to create and evaluate several versions of the user interface before it is prototyped, designed and implemented [10]
- User interface prototypes

### 3.2. Capturing Context of Use in RESPECT Process

The RESPECT process is based on the iterative human-centered design process for interactive system outlined in the ISO-13407 standard [11]. RESPECT collects and generates user requirements under a series of semi-structured text forms. It distinguishes three iterative phases: user context and early design, prototype and user tests, and user requirements documentation. Each of these phases brings to light the following activities:
- Understanding the user and organizational needs for a system, and planning the user-centered design process
- Understanding the user context, also known as the context of use
- Specifying user and organizational requirements
- Developing design concepts or operational prototypes
- Testing whether the prototype meets the user and organizational requirements

By the end of the third phase, the RESPECT process produces twelve text-based forms. These forms detail the general system characteristics, organizational structure, task scenario and interaction steps required to carry out key end-user tasks, technical environment, system functions and features, user interface design, user support, physical, social and organizational environments, standards and style guides to apply, usability testing plan, as well as implementation plan. Table 1 describes the scenario and interaction steps form required to carry out key end-user tasks.

<table>
<thead>
<tr>
<th>User</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User inserts card for identification purposes.</td>
<td>1. System reads card, regardless of how inserted, and displays message prompt.</td>
</tr>
<tr>
<td>2. User enters PIN or presses finger on fingerprint pad.</td>
<td>2. If recognized, system displays options.</td>
</tr>
<tr>
<td>4. User selects or enters amount up to the maximum.</td>
<td>4. System responds that the cash is ready and displays menu of other services.</td>
</tr>
</tbody>
</table>

Table 1. Task Scenarios and Interaction Steps

To facilitate the elicitation and validation of user requirements, several usability techniques are suggested. Among them, we list brainstorming, interviews, surveys, observation, focus groups, group discussion, task analysis and allocation, storyboarding, paper, video and rapid prototyping, scenario building, walkthroughs and wizard of Oz prototyping.

### 3.3. Complementarities and Foundation of ACUDUC

While use case artifacts generally describe functional requirements including details needed by the software developers, RESPECT enhances functional requirements descriptions by adding information that improves the users’ understanding of the future system. Basically, this additional information concerns the context of use and takes into account usability problems that may be encountered by end-users when performing certain tasks.

Table 2 summarizes the complementarities that we identified and the ways that ACUDUC reconciles them.
RESPECT captures a complete description of the context of use and usability goals/metrics, including user characteristics, task analysis, as well as the physical, technical and organizational environments in which the system will be used.

Although in theory use cases have the potential to gather the non-functional requirements that are a simplified description of the context of use, in practice, use cases have been used for gathering the system functionalities and features including technical capabilities and constraints.

1- ACUDUC defines context of use and functional requirements as two views of the same shared requirements picture.

2- The functional view on this picture is a set of artifacts describing the functional requirements. The usability view is a set of artifacts describing the context of use and the usability goals/metrics in which the functionalities will be used.

The context of use is described using a non-formal notation that is easy to understand by users and stakeholders alike. However, these forms are a cause for inconsistency and ambiguity when used by software developers.

3- ACUDUC supports Artim's [1] opinion of "one model, but many views and notations." We strongly share his belief that different notations for the same concept may foster communication between all the persons involved in the requirements process.

4- In ACUDUC, the artifacts related to the context of use should be described using text-based forms or any other information notation. Functional requirements artifacts are manipulated using the use case model.

5- ACUDUC should mediate the communication between the individuals involved in the requirements process: users, stakeholders, and software and usability engineers. It should maintain the correspondence between the two views of requirements.

The context of use is used by usability specialists as an important input for usability testing. The functional requirement artifacts are used by software developers as a starting point for design.

6- ACUDUC includes activities for reviewing and validating the integrity and consistency of all requirements artifacts from both the usability and developers' perspectives. It then generates portfolios called usability and implementation plans.

Table 2. The Foundations of ACUDUC

<table>
<thead>
<tr>
<th>User-Centered Requirements Process</th>
<th>Use Case-Driven Requirements Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPECT captures a complete description of the context of use and usability goals/metrics, including user characteristics, task analysis, as well as the physical, technical and organizational environments in which the system will be used.</td>
<td>Although in theory use cases have the potential to gather the non-functional requirements that are a simplified description of the context of use, in practice, use cases have been used for gathering the system functionalities and features including technical capabilities and constraints.</td>
</tr>
</tbody>
</table>

### 4. Keys Activities of the ACUDUC Framework

The ACUDUC approach distinguishes four steps for integrating context of use and usability techniques in the use case-driven requirements lifecycle:

1- Summarizing the system from the user’s perspective.
2- Gathering and specifying the context of use
3- Specifying and/or generating functional requirements including UI widgets and use cases
4- Reviewing and validating an integrated picture of requirements

These steps were defined and validated through various industrial projects. All the projects are related to Internet-based interactive systems. They were conducted at the Computer Research Institute of Montreal (CRIM) with industry partners. The requirement specification step always involved a group comprised of:

- Usability engineers – those who specify the context of use and conduct usability requirement reviews and testing sessions.
- Software developers – those who detail the specification of the system’s functionalities and develop the use case model and the user interface prototype.
- Users – those who use the system. They may be the direct users (generally called end-users) who use the system to complete their tasks, or indirect users who use it for other
purposes, such as system administrators, installers and demonstrators. Even if the cause-effect relationship between usability problems and indirect users’ jobs can be easily demonstrated, indirect users’ requirements are most often neglected in both software and usability engineering approaches. For example, a system whose installation is not easy to understand will be incorrectly installed in terms of user preference and specific need. This will be source of many usability problems.

Stakeholders – those who are affected by the system or can influence its development, such as marketing staff and purchasers. Their input is mainly used as constraints or additional requirements. For example, marketing staff may like to add to the system a specific function that another company is planning to implement. In many projects, we observed that such functions are more a source of distraction and ambiguities for users than added values. Consequently, functions such as these are classified in specific category in the requirements.

Each of these contributors is characterized by its role and responsibilities. Roles are expressed in terms of the activities the contributor performs, and each contributor is associated with a set of activities. The responsibilities of each contributor are usually expressed in relation to certain artifacts that the contributor creates, modifies or controls [Figure 3].

Figure 3. Relationships between Requirement Artifacts, Contributors and Activities in ACUDUC

4.1 Summarizing the System from the User's Perspective

It is important for usability-to-software engineering collaboration and for consistency and coherence of ease of use and functional requirement artifacts to gain a high-level understanding of the system from the end users perspective. Therefore, ACUDUC starts when a representative set of users and/or stakeholders are invited to summarize the system from the future user’s perspective. They are mainly asked to answer different questions that we organized in a system summary form [Figure 4]. Maguire [11] and Constantine [4] suggested similar questions. Users and stakeholders, the main contributors during this step, are invited to give brief answers to these questions. All completed forms are then analyzed and compiled in a unique system summary form by usability engineers. This compiled form is approved by software developers, stakeholders and users. It is used as a roadmap during the requirement process and represents a general consensus on the system.

<table>
<thead>
<tr>
<th>Question</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the purpose of the system?</td>
<td>ISO 9000-based quality system over an Intranet</td>
</tr>
<tr>
<td>Why is this system necessary?</td>
<td>Supporting the development of the company outside the</td>
</tr>
</tbody>
</table>
Table 4. An Example of the System Summary Form

<table>
<thead>
<tr>
<th>Who will use the system?</th>
<th>country (new clients, remote offices.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>What will the users accomplish with the system?</td>
<td>Access to quality procedures and associated forms. Learn the quality system and the ISO 9000 standard</td>
</tr>
<tr>
<td>Where will the system be used?</td>
<td>Standalone workstations and personal digital assistants</td>
</tr>
<tr>
<td>How will users learn to use the system?</td>
<td>Introductory course and online assistant</td>
</tr>
<tr>
<td>How will the system be installed?</td>
<td>By a Webmaster for the server version, and by employees on their PDA (download from the server)</td>
</tr>
<tr>
<td>How will the system be maintained?</td>
<td>By a Webmaster and a quality control manager</td>
</tr>
</tbody>
</table>

Table 5. Summary of Context of Use Portfolio

<table>
<thead>
<tr>
<th>Form</th>
<th>Main Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>User characteristics</td>
<td>Knowledge, skill, experience, training, physical attributes, habits and motor-sensory capabilities</td>
</tr>
<tr>
<td>Characteristics of tasks that users will perform</td>
<td>Goals, frequency and duration of tasks</td>
</tr>
<tr>
<td>Work environment in which the system will be used.</td>
<td>Organizational attributes including structure of the operational teams, individual staff members' level of autonomy, business process. Technical platform and constraints including hardware, operating systems, network, and software on which the new system depends. Physical factors including potential health, safety and security issues if applicable</td>
</tr>
</tbody>
</table>

4.2. Gathering and Validating the Context of Use Portfolio

The context of use portfolio describes all the aspects that have an important impact on the system usability including (Table 5):

- User characteristics of each group of direct and indirect users.
- Task description form including goals, frequency and duration. Task flowchart [5] may be used instead of this form.

- Organizational, technical and physical environments.

The context of use portfolio includes also a description of usability goals such as effectiveness, efficiency and satisfaction, quantitative/qualitative metrics such as required user performance and satisfaction, as well as guidelines and usability patterns to which the user interface should conform.

4.3. Specifying Functional Requirements

The functional requirements portfolio include the following artifacts use cases, system functionalities, characteristics and constraints as well as UI prototypes.

Our investigations have shown that it is possible to generate some functional requirements artifacts from the context of use artifacts. This is true for the use case diagram and UI widgets. If this generation can be automated, it will certainly bridge the gap between context of use artifacts and functional requirements. At the same time, it will improve the communication between software and usability developers.

To illustrate this fundamental result, let us consider a substantial real-world example for which we have attempted to write a use case diagram that would reproduce the task descriptions described by usability engineers. The task analysis was conducted by usability engineers with hopes of computerizing the handling of “psychological/social” requests at a community health center in Quebec, Canada. The task load of social workers at a health center involves a considerable amount of
communication with others (the requestor, the patient and fellow healthcare professionals). Table 6 describes the Handling a Request task. Use cases and specifications of tasks both describe the task in different ways. Beside tasks and use cases, which have nearly the same meaning in both approaches there, we listed the following ways that a task analysis model can complement a use case model:

- User characteristics form can be used to provide further information about actors
- Technical, physical and organizational environments forms can be used to describe the context of use related to a use case diagram

<table>
<thead>
<tr>
<th>Table 6. Task Analysis and Use Cases Diagram</th>
</tr>
</thead>
</table>

### CLSC Task 1: Handling a Request

**CLSC Task 1.1: Receive Request /*by telephone or in person*/**

- **CLSC Task 1.1.1: Get Details of Request**
  - **CLSC Task 1.1.1.1: Situate the Request**
  - **CLSC Task 1.1.1.3: Evaluate the Request’s Urgency**
  - **CLSC Task 1.1.1.5: Identify Special Elements of the Request**

- **CLSC Task 1.1.4: Respond to an Immediate Request**

**CLSC Task 1.2: Evaluate Urgency of Request /*this task is usually carried out by a single individual and can extend over several sessions*/**

**CLSC Task 1.2.4: Identify Solutions**

---

5. Conclusion

In this paper, we presented the rationale behind the ACUDUC approach and our understanding of how to integrate usability in the use-case driven software development lifecycle. With respect to experimentation, two specific processes constitute the focus of our interests: use case-driven and the user-centered requirements engineering processes. Further to the ACUDUC framework we defined, we identified the following principles which we consider as fundamental enhancements to the functional requirement lifecycle.

First, the requirements of an interactive system must be defined on two levels, but not independently of one another. The first level is concerned with the specification of the context of use, and the second focuses on functional
requirements. Different specification notations may be used for the two levels, but they should exploit an integrated representation of all the requirements artifacts. In our case, we elected to use text-based forms from RESPECT and the graphical representation of use cases as defined in Unified Method Language [7].

Secondly, the list of artifacts describing the context of use ensures a good usability specification. Better still, this list can assist with generating functional requirements, at least to a limited extent. This result is fundamental because it can minimize requirements artifacts inconsistency and improve communication between software and usability engineers.

Thirdly, the classification of the contributors that we established clarifies roles and responsibilities of each contributor. Furthermore artifacts such as the system summary form are a simple and effective way to maintain a quasi-permanent consensus between people that do not speak the same language and prefer to use different notations.

Our work to date has addressed two of the three major issues for reconciling use case and usability driven approaches namely, what artifacts should be developed, and what activities should be performed to develop the artifacts. We have yet to address the third issue of how to facilitate human-to-human collaboration amongst the users, stakeholders, usability engineers, and software engineers during the process.

6. References


