Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project

Rodrigo O. Spínola, Felipe C.R. Pinto, and Guilherme H. Travassos

PESC-COPPE/UFRJ
Cx. Postal 68.511, CEP 21945-970, Rio de Janeiro, RJ, Brasil
{ros,felipecrp,ght}@cos.ufrj.br

Abstract. The development of ubiquitous software project demands the use of specific software technologies to deal with the inherent complexity of this type of project. Despite the advances in the software engineering field, the building of ubiquitous software still represents a grand challenge. For instance, secondary and primary studies indicated the existence of 13 ubiquity characteristics that can influence ubiquitous software projects. Therefore, in this paper we describe these ubiquity characteristics organized into a body of knowledge regarding ubiquitous computing and used to characterize ubiquitous software projects. Besides, an on-going research concerned with supporting ubiquity requirements definition and verification (checklist based inspection) activities is also introduced.

Keywords: Ubiquitous Computing, Requirements Engineering, Software Quality, Experimental Software Engineering.

1 Introduction

The increasing complexity and exposure to new risks can prevent traditional software technologies to keep their effectiveness when used to develop ubiquitous software projects. This is due to the different software characteristics involved in the engineering of such projects that must be considered for assuring the delivering of quality products [4, 11, 13].

Into this software engineering context, some development challenges such as quality, time and budget constraints can be made explicit by answering questions such as: (1) What (new) software technologies are necessary to deal with the software ubiquity characteristics?; (2) What are the risks associated with ubiquitous software projects?; (3) What quality characteristics software engineers should have in mind when accomplishing ubiquitous software projects? (4) How to support the requirements definition and quality assurance activities in ubiquitous software projects?

Additionally, answering these questions can represent big challenges because:

- Ubiquitous computing (ubicomp) represents a multidisciplinary and new research intensive knowledge area [11]. Consequently, we can observe a constant evolution of ubicomp concepts in software related areas such as computer networks, signal processing, optimization, and artificial intelligence among others;

- A small number of papers evidencing the use of software engineering principles to support the development of ubiquitous software projects can be currently found in the technical literature, making hard identifying the ubiquity characteristics influence in software projects.

These challenges combined with our experience on dealing with software projects involving requirements regarding ubiquity motivated us to try to understand what could be the ubiquity characteristics influence in the software development life cycle. The difficult on dealing with this new software category requirements have driven us to think about how to enlarge the usual body of knowledge regarding the development of conventional software projects to also embrace ubicomp applications.

Sakamura [13] states that the creation of ubiquitous software applications is hard and can involve several ubiquity characteristics. It was also observed by Spínola et al. [14], which identified and evaluated 10 relevant characteristics concerned with ubiquitous software, such as service omnipresence, invisibility, context sensitivity, adaptable behavior, experience capture, service discovery, function composition, spontaneous interoperability, heterogeneity of devices and fault tolerance by undertaking a systematic literature review. In complement, information regarding functional and restrictive factors concerned with the main issues when developing ubiquitous software projects have been described. At this point, a functional factor is concerned with the facts or situations related to functional requirements. A restrictive factor is concerned with the facts or situations related to non- functional requirements.

This information can be useful when a software engineer is looking for software technologies to use in the software projects. For instance, some software requirements technologies can be used to deal with one or other ubiquity characteristic [26, 27, 28, 21, 20, 19, 10, 8, 2]. However, their use in ubiquitous software projects can be limited due to the lack of knowledge on how to apply them when ubiquity characteristics are combined into the project.

Aiming at providing support for software developers in characterizing and developing ubiquitous software projects, this paper describes an on-going research towards the creation of a framework to support software technologies concerned with the definition and quality assurance of ubiquity requirements. The first target is represented by organizing knowledge regarding ubiquitous software projects through secondary and primary studies and making it available to the practitioners. It is intended to support development activities concerned with ubiquity requirements specification and validation (checklist based inspection). So, a set o facilities is going to be available supporting:

- To choose relevant ubiquity characteristics for the software project (level of relevance can depend on the application domain or project's requirements);
- To identify the ubiquity requirements (functional) through a list of functional factors regarding each selected ubiquity characteristic;
- To define the ubiquity requirements, by guiding the software engineer to properly detail all requirements accordingly the selected ubiquity characteristic; and,
- To assure the ubiquity requirements quality, by providing checklists that can make the software engineer able to inspect whether all expected ubiquity features were appropriately captured by the requirements specification.

This paper is organized in eight sections, including this Introduction. In the following section, the definition of ubicomp and its characteristics are discussed. In sequence, the results of a detailed analysis for each ubiquitous characteristic considering its functional and restrictive factors are presented. Then, we present an approach to characterize applications considering their ubiquity adherence level. We also present some results obtained with the use of the characterization approach. In sequence, the evaluation of the concepts involving ubiquitous computing previous discussed is presented. After that, some concepts about requirements engineering and related works are analyzed. In the following section, the proposed framework is explained. Finally, we summarize the main contributions of this paper and future perspectives of this research.

2 Ubiquitous Computing Characteristics

Weiser introduced the area of ubiquitous computing and put forth a vision of people and environments augmented with computational resources providing information and services when and where desired [17]. Weiser's vision also described a proliferation of devices at varying scales, ranging in size from hand-held "inch-scale" personal devices to "yard-scale" shared devices. That is, the computer is integrated into the environment in such a way that its use becomes non intrusive. This definition set the origin of the term Ubiquitous Computing and, although it is important for presenting a new computing paradigm, we believe it is not currently complete at all. This positive lack of completeness reflects its importance and innovation at that time, and how fast technology has evolved.

Therefore, an investigation towards an updated definition of ubiquitous computing seems to be necessary. So, this section intends to present the reached results of a systematic review whose goals were the field understanding and describing an up-to-date ubiquitous computing definition that could support our research: (Q0) What is ubiquitous computing?; (Q1) How ubiquitous computing is currently being presented?; (Q2) What characteristics do define applications for ubiquitous computing?

To accomplish this systematic review, it was elaborated and accomplished a research protocol. The items below define the main characteristics of this protocol:

- Keywords: ubiquitous computing, pervasive computing, ubiquitous application, ubiquitous system, ubiquitous software, pervasive application, pervasive system, pervasive software, feature, requirement, characteristic, definition, characterization, and concept.
- Paper sources: IEEE Portal, ACM Digital Library, INSPEC, and EI COMPENDEX. These digital libraries have been chosen by convenience because they were fully available to the researchers.
- Example of a search string (Q0 only) for ACM Digital Library:
  +"ubiquitous computing" abstract:concept abstract:definition abstract:characteristc
  +"pervasive computing" abstract:concept abstract:definition abstract:characteristic
- Inclusion and exclusion criteria: These criteria define statements that must be true for the paper be included in the selected papers set. They must be available on the internet, be written in English, provide a ubiquitous definition (Q0 only), report

current applications regarding ubiquitous computing concepts (Q1 only), report software application (applications concerned with supporting software are not considered) and present characteristics associated with ubiquitous systems (Q2 only).
- Preliminary studies selection process: each returned publication must have its abstract and introduction analyzed by two researchers and, based on the inclusion and exclusion criteria, they select it or not to a more thorough analysis.

To summarize, following the research protocol 41 papers among 751 were selected to extract information. These papers supported the identification of an updated ubiquitous computing definition besides a set of concepts characterizing ubiquitous software projects, as described below.

Ubiquitous computing is present when computational services or facilities become available to the people in such a way that computer is no longer visible nor needed to be used as an essential tool to access these services or facilities. The services or facilities can materialize themselves at any time or place, transparently, through the use of common daily devices. To make it happens it is desired that systems composing this domain take into consideration the following characteristics (an illustrative scenario aiming at providing our interpretation is also provided):

- **Service Omnipresence (SO):** It allows users to move around with the sensation of carrying computing services with them;
  - Scenario: An employee is taking part in a business meeting but needs to leave it. However, it also needs monitoring the meeting progress to report its results for the manager. When the employee leaves the meeting's room, the distance conference managing software can be activated on the smartphone. Thus, the employee will have access to the meeting everywhere when it is moving around.
- **Invisibility (IN):** The ability of being present in daily use objects, weakening, from user's point of view, the sensation of explicit use of a computer and enhancing the perception that objects or devices can provide services or some kind of "intelligence". Thus, it demands natural interfaces to facilitate a richer variety of communications capabilities among humans and computer systems. The goal of these natural interfaces is to support common forms of human expressions and leverage more of our implicit actions in the world [23];
  - Scenario: a personal health care system that must be constantly monitoring some health variables without patients' intervention.
- **Context Sensitivity (CS):** Ubiquitous systems should have mechanisms to collect information from the environment where it is being used;
  - Scenario: a system to control a refrigerator should be constantly monitoring the temperature to keep the device in the ideal state for products conservation.
- **Adaptable Behavior (AB):** The ability of dynamically to adapt itself accordingly the offered environment services, respecting its limitations;
  - Scenario: By identifying the bandwidth reduction to the point of harming the audio and video transmission, the video conference management software should reduce the audio and video quality allowing the normal communication flow between the conference participants.

- **Experience Capture (EC):** Ubiquitous systems should have mechanisms to support the capturing and registering of experiences for later use;
  - Scenario: A software for ambient intelligence can identify common user behaviors, for example: when arrives at home, the user turns on the room light, heats water for coffee, turns on the tub and sets the water temperature to 28 ° C. The software can manage these activities as soon as it identifies the user arrives at home without repetitive user's commands.
- **Service Discovery (SD):** This characteristic states that ubicomp systems should have mechanisms to support pro-active discovery of services, which should be according to the environment where it is being used in order to find new services or information to achieve some desired target;
  - Scenario: A smartphone software can identify services provided by a supermarket to support the purchase of products by the customer, such as a map with promotions and products location.
- **Function Composition (FC):** To be able to creating a service required by the user based on available basic services;
  - Scenario: A user needs to have a spreadsheet view and generate a PDF file with the view outcome and these services are not available on the workstation. The software can identify the necessary services and makes them available for use.
- **Spontaneous Interoperability (SI):** The ability to change partners during its operation and according to its movement;
  - Scenario: A person is moving and the software, running on a PDA, is playing a task-intensive processing. During the moving, the software can interact with other devices in the environment for temporary allocation of processing.
- **Heterogeneity of Devices (HD):** It provides application mobility among heterogeneous devices. That is, the application could migrate among devices and adjust itself to each one of them;
  - Scenario: A software for stock market monitoring allows the access to all its functionalities through a workstation at the office. However, in other organizational unit only a PDA is available. In this situation, the software should migrate part of its features to be accessed by a PDA in a way that a user could continue to access the necessary information.
- **Fault Tolerance (FT):** The ability to self adapt when facing environment's faults (for example, on-line/off-line availability).
  - Scenario: ubiquitous systems are liable to a large number of events that can cause system failure, such as, sensors with hardware problem, network failure, among others.

At this point, it is important to notice that the ubicomp definition captures the conditions where we can access computerized resources in a ubiquitous way. Besides, ubiquitous systems have a well-defined scope, and this scope is influenced by the ubiquity characteristics set present in the application. We believe it happens because ubiquity can be considered a system property and it can be also partially achieved. Thus, a system can implement completely or partially the functionalities associated with the ubiquity characteristics.

3 Functional and Restrictive Factors Related to Ubicomp Characteristics

The ubiquity characteristics previously mentioned are still described in high abstraction level, making hard to understand how they could influence the software functionalities or restrict the software design possibilities. Therefore, it is important to make explicit functional and restrictive information for each one of the ubiquity characteristics. For this, a complementary systematic literature review was undertaken. Its goal was to answer the question: what are the functional and restrictive factors associated with each one of the ubiquitous software characteristics?

The research protocol previously mentioned (section 2) was reused and evolved to support this study. The items below define the main variations:

- Keywords: ubiquitous computing, pervasive computing, functional requirement, functionality, feature, characteristic, non-functional requirement, quality requirement, invisibility, context sensitivity, adaptable behavior or task dynamism, capture of experiences, service discovery, spontaneous interoperability, device heterogeneity and fault tolerance.
- Paper sources: IEEE Portal and ACM Digital Library. These digital libraries have been chosen for the sake of simplicity (reduction in the number of search strings) and full availability to the researchers.
- Example of a search string for the IEEE Portal: (('pervasive computing' <or> 'ubiquitous computing') <in> metadata) <and> ((('functional requirement' <or> functionality <or> feature <or> characteristic) <or> 'non-functional requirement' <or> 'quality requirement')) <in>metadata) <and> ('computer everywhere')
- Inclusion and exclusion criteria: the papers must be available on the internet, the papers must be written in English and the papers must provide information regarding functional and/or restrictive factors associated with each ubiquitous characteristic.
- Preliminary studies selection process: each returned publication must have its abstract and introduction analyzed by one researcher and, based on the inclusion and exclusion criteria, to be selected or not to a more thorough analysis. Conflicts will count with a second researcher to help on the inclusion decision.

To summarize, 59 papers among 599 were selected to extract information following the research protocol. Using the acquired data, it was possible to identify 168 factors (123 functional and 45 restrictive) (the complete set of functional and restrictive factors can be found at http://www.cos.ufrj.br/~ros/ubforms.html). Moreover, it was also possible to group the factors accordingly their meaning and conceptual linkage, associating each factor to just one group of factors. For example, for the "Context Sensitivity" characteristic, the factors "Contextualize obtained information" and "Store information" can be grouped into the "Context Information Management" factor group.

The grouping made easier the analysis process due to the great number of factors found by the systematic literature review. Table 1 summarizes quantitatively the reached results. The first column shows the ubiquity characteristics. The second and third columns show how many studies were found regarding each characteristic, in absolute and percentage values respectively considering the set of selected papers for analysis. The fourth and fifth columns show how many functional and restrictive factors were found for each ubiquity characteristic, respectively. At the last column, it is presented the percentage distribution of factors per characteristic.

From Table 1 it is possible to observe that, for all ubiquity characteristics but service discovery and fault tolerance, the focus is concentrated in the functional factors. This observation is based on the fact that the number of identified functional factors is greater than the number of restrictive factors. Besides, it can represent an indication that more researches on ubicomp have been made with the purpose of investigating how ubiquitous software projects can be defined in terms of functionalities. However more investigation is necessary to understand this behavior.

Table 1. Ubiquity Characteristics and correspondingly amount of functional/restrictive factors

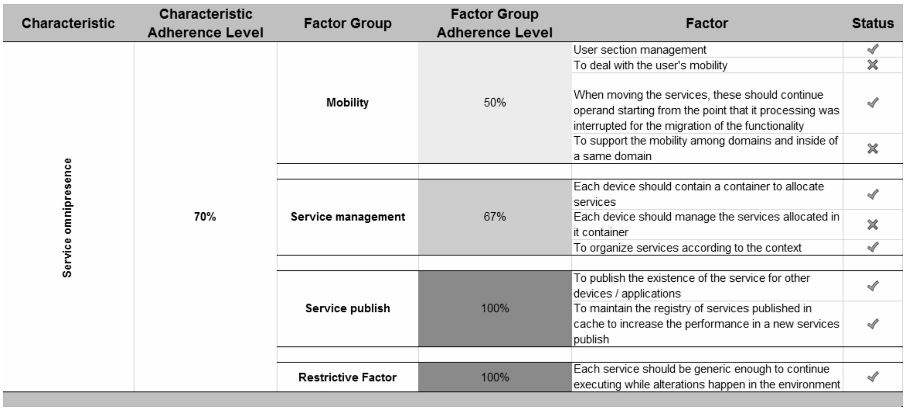
Ubiquity Characteristic	Presence	% of 59	Functional	Restrictive	% of 168
Service Omnipresence (SO)	28	47,5	9	1	6,0
Invisibility (IN)	26	44,0	8	2	6,0
Context Sensitivity (CS)	56	94,9	22	8	17,9
Adaptable Behavior (AB)	52	88,1	24	8	19,0
Experience Capture (EC)	11	18,6	7	0	4,2
Service Discovery (SD)	28	47,5	13	13	15,5
Function Composition (FC)	19	32,2	18	5	13,7
Spontaneous Interoperability (SI)	21	35,6	10	2	7,1
Heterogeneity of Devices (HD)	18	30,5	9	3	7,1
Fault Tolerance (FT)	11	18,6	3	3	3,6
Total of factors			**123**	**45**	

4 Characterizing Ubiquitous Software Projects

Ubiquitous software projects can exhibit different levels of adherence to the ubiquity characteristics and their respective factors (these different levels of adherence can be a consequence of the application domain and project's requirements, for instance) [14]. It seems that the comprehension about how these ubiquity characteristics are usually explored in software projects can be important to support the proposal of new software engineering technologies regarding the development of ubiquitous software projects.

Therefore, taking into account the concepts previously described, we have designed a checklist and proposed an approach to support the characterization of software projects accordingly their ubiquity adherence level. Basically, this characterization approach is composed by three steps: (1) Identifying the presence of the functional and restrictive factors of each ubiquity characteristic considering the ubiquitous software project requirements set; (2) Assessing the adherence level of each ubiquity characteristic for the software project based on the presence/absence of each correspondent functional and restrictive factor; (3) Representing the ubiquity adherence level for the system through using the values obtained in the step 2 to generate a graph.

To support the steps 2 and 3, it has been built a spreadsheet-based form to calculate the adherence level for each ubiquity characteristic. Fig. 1 shows a fragment of the proposed checklist. Basically, as the user fills in the Status column, the Factor Group Adherence Level and Characteristic Adherence Level columns can be calculated for each group of factors and the ubiquitous computing characteristic, respectively. In the final step, the evaluated percentage values of the Ubiquity Characteristics Adherence Level column are used to draw a graph representing the software project ubiquity adherence level. For instance, Fig. 2 (left graph) represents the obtained graph when applying this checklist to the ubiquitous application presented in [16]. We can notice that a real ubiquitous software project can differ from the captured ubiquitous scenario (sections 2 and 3) observing the left and right graph presented on Fig. 2.



Characteristic	Characteristic Adherence Level	Factor Group	Factor Group Adherence Level	Factor	Status
Service omnipresence	70%	Mobility	50%	User section management	✓
				To deal with the user's mobility	✗
				When moving the services, these should continue operand starting from the point that it processing was interrupted for the migration of the functionality	✓
				To support the mobility among domains and inside of a same domain	✗
		Service management	67%	Each device should contain a container to allocate services	✓
				Each device should manage the services allocated in it container	✗
				To organize services according to the context	✓
		Service publish	100%	To publish the existence of the service for other devices / applications	✓
				To maintain the registry of services published in cache to increase the performance in a new services publish	✓
		Restrictive Factor	100%	Each service should be generic enough to continue executing while alterations happen in the environment	✓

Fig. 1. A checklist fragment to characterize ubiquitous software projects

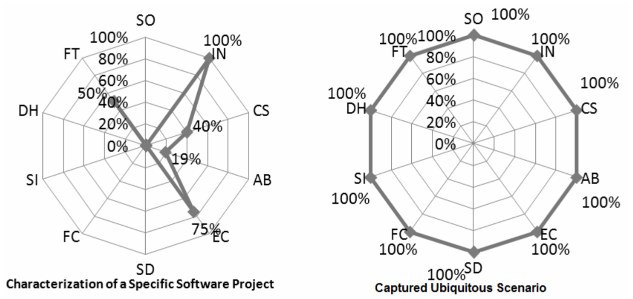


Fig. 2. Example of ubiquity characteristics and their adherence levels

This checklist has also been used to characterize 12 different ubiquitous software projects [1, 3, 5, 6, 7, 9, 12, 15, 16, 18, 29, 30]. A detailed description of the characterization process and its results can be found in [22]. An interesting behavior could be observed: if the number of factors identified for each ubiquity characteristic increases or decreases, the same happens with the number of factors implemented in the software projects. This behavior can indicate: a) there is a natural gap between the state-of-the-art and state-of-the-practice; b) ubiquitous software projects can capture those ubicomp characteristics differently. However, an exception to this behavior was observed regarding the Function Composition characteristic. None of the 12 ubiquitous software projects reported to deal with any of the 23 factors regarding the FC characteristic. This behavior was not expected because this ubiquity characteristic has been considered required in about 32.2% of the analyzed papers from the second systematic review (section 3). A possible explanation could be the difficulty to deal with the inherent complexity regarding the composition of functions in software. As stated before, more investigation is necessary to also understand this behavior.

An additional observed behavior is regarding the focus on some specific ubiquity characteristics. It seems that ubiquitous software projects pay more attention to the invisibility, context sensitive and adaptable behavior characteristics. The other ones seem to appear as isolated initiatives, even considering the analyzed projects represent examples in the years' range 2000-2007, where some technological evolution took place. We did not found any feasible explanation for this behavior. However, some questions could be raised:

- Does this behavior represent a natural gap between the state-of-the-art and state-of-the-practice that deserves further investigation?
- Is there a need to evaluate the set of identified ubiquity characteristics and their functional and restrictive factors?

Some feasible answers to these two questions could be, respectively:

- The set of identified ubiquity characteristics and their functional and restrictive factors make sense, and the distance between the state-of-the-art and state-of-the-practice is natural and relates to the technology evolution;
- There are some adjustments that must be applied to the set of identified ubiquity characteristics and their correspondingly functional and restrictive factors.

One could consider the answers can make sense. However, further investigation is necessary, what leads us to consider the ubiquity characteristics and their associated factors evaluation directly in the field. Therefore, we considered to survey ubicomp researchers which is described in the next section.

5 Evaluating Ubicomp Concepts through a Survey

The goal of this study was to **analyze** the previously described ubiquity characteristics, their factors, and group of factors extracted from the technical literature **with the purpose of** characterizing **with respect to** their applicability and scope **into the context of** ubiquitous software projects **from the point of view of** software engineering researchers working with the research and development of ubiquitous software projects.

This study was planned to survey ubicomp researchers considering the following questions regarding the previously described set of ubiquity characteristics and functional and restrictive factors:

- Is there any additional ubiquity characteristic that is not present in the initial set that should be included?
- Is there any ubiquity characteristic present in the initial set that should be excluded?
- Is there any additional ubiquity characteristic factor group or factor that is not present in the initial set that should be included?

- Is there any ubiquity characteristic factor group or factor present in the initial set that should be excluded?
- Are the ubiquity characteristics and their associated factors and factor groups applicable to characterize ubiquitous software projects?

This survey has already been planned and executed, in a first moment, considering the Brazilian researcher's population. The choice of the subjects was based on the CNPq's (National Council for Scientific and Technological Development) Research Groups Search Directory considering those ones which list ubicomp as one of their research interests. The subjects were contacted by e-mail (at total 60 ubicomp researchers have been invited), and the questionnaire was also sent by e-mail. The questionnaire was organized to be filled in three steps:

1) Characterization of the subjects' background and skills. In this step the subjects were asked about his/her personal data (name, email), academic degree, experience level on software project development (in years), and the number of executed software projects per ubicomp characteristic;
2) Identification of the ubiquity characteristics that should be included/excluded/kept in the initial set. The subject can confirm which ubiquity characteristics are important to characterize ubiquitous software projects, input additional characteristics that he/she considers important, or exclude some characteristics of the initial set;
3) Identification of the ubiquity characteristic factor groups and factors that should be included/excluded/kept in the initial set. For each factor group and factor, the subject can confirm their importance, input additional factor groups or factors not included in the initial set that he/she considers important, or exclude some of them.

At the end, 10 subjects (about 17% of the invited researchers) answered the questionnaire (8 of them PhDs). Table 2 shows the researchers' skill level for each ubiquity characteristic where: (1) **H**igh: researches and has taken part of more than two software projects considering the ubiquity characteristic; (2) **M**edium: researches and has taken part of one or two software projects considering the ubiquity characteristic. (3) **L**ow: just researches about the ubiquity characteristic; (4) **N**one: does not research neither has taken part of a software project with the ubiquity characteristic.

For the analysis stage, each subject had a different weight according to his/her background and skill level. Researchers with higher experience/skill level had greater weight. After the weights definition, the answers from all subjects were evaluated for each evaluated ubicomp characteristic, factor group, and factor. It is important to notice that, except the fault tolerance characteristic, all others have been evaluated by at least one researcher with high skill level.

The results allowed us to evolve the initial set of ubiquity characteristics, factors and their factor groups. Basically, the changes were: (1) Inclusion of three additional ubiquity characteristics: scalability, quality of service, and privacy and trust; (2) Reorganization of the ubiquity characteristics considering the two perspectives: functional and restrictive; (3) Exclusion of three functional factors.

Table 2. Researchers' skill level

ID	Academic Degree	SO	IN	CS	AB	EC	SD	FC	SI	DH	FT
R01	M.Sc.	H	M	H	M	H	M	M	L	N	L
R02	M.Sc.	H	H	H	H	M	M	M	M	M	M
R03	Ph.D.	L	L	M	M	L	M	M	M	N	L
R04	Ph.D.	H	L	L	L	L	L	L	L	L	L
R05	Ph.D.	H	M	H	H	L	M	M	L	H	N
R06	Ph.D.	H	M	H	M	L	L	L	L	H	L
R07	Ph.D.	M	M	H	H	H	M	L	M	M	L
R08	Ph.D.	M	M	L	H	H	H	M	M	N	N
R09	Ph.D.	L	L	H	H	H	N	M	M	M	M
R10	Ph.D.	H	L	L	H	M	L	H	H	N	N

The initial ubiquity characteristics set organization before and after survey execution is shown in Fig. 3. Before survey execution, 10 ubiquity characteristics were identified by the systematic literature reviews (sections 2 and 3). The survey execution allowed us to observe that those 10 characteristics can also be structured considering the two different perspectives: functional and restrictive. This new categorization seems to make sense because there are characteristics that are clearly related with non-functional software aspects and they can bring some constraints on how the functional characteristics are implemented. Moreover, 3 new restrictive ubiquity characteristics were identified: scalability, quality of service, and privacy and trust. Additionally, the fault tolerance characteristic was included into the ubicomp restrictive characteristics group.

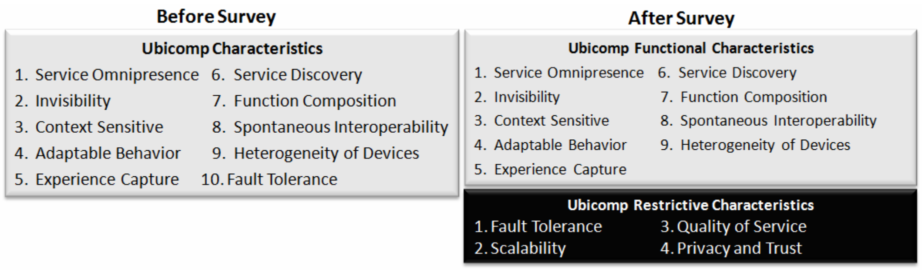


Fig. 3. Ubicomp characteristics definition before and after survey execution

These findings allowed us to evolve the set of knowledge regarding ubiquitous software projects and their characteristics. At this point in time, it was able to define a body of knowledge regarding ubicomp, to provide a conceptual framework to guide new researchers and practitioners in the ubiquitous software projects development, and, to evaluate the proposed checklist. Considering the obtained qualitative data, researchers suggested the importance of supporting software engineering activities regarding ubiquitous software projects, mainly those related with requirements specification and project planning.

6 Ubicomp and Requirements Engineering

The results obtained in our research regarding software development in general increased our interest in the challenges related with ubicomp requirements engineering. We are particularly interested on software technologies to support activities regarding requirements definition and quality assurance considering the ubicomp scenario. Well specified requirements can be considered a success factor to deliver software products with the expected quality and following the project budget [25]. It should not be different when dealing with ubiquitous software projects.

For this, we accomplished an ad-hoc literature review influenced by the systematic review principles. The review goal was to identify the existence of approaches supporting requirements definition and verification activities regarding the ubiquity characteristics (section 5). The following sources of information were analyzed: IEEE and ACM Digital Library, Ubicomp Proceedings and International Requirements Engineering Conference Proceedings. Additionally, one paper [2] analyzing the bibliography reference on the selected papers has been identified.

This review resulted in the selection of nine papers [2, 8, 10, 19, 20, 21, 26, 27, 28] to extract information. All of them presented requirements definition techniques or examples of requirements definition. Among them, the approach proposed by Chiu et al. [27] presents a set of steps to support activities regarding requirements elicitation and specification that could be used in a more general way. However, in the paper it is limited to the context sensitivity characteristic.

The other proposals have a more limited scope. Two of them just show how the requirements of a ubiquitous system are defined [21, 28]. In both cases, the requirements are only listed and textually detailed in the paper without any elaborated description about the technique supporting their definition and verification. Finally, it is important to notice that it was not found any quality assurance technique regarding ubiquity requirements for ubiquitous software projects.

7 A Framework to Support Definition and Quality Assurance Activities Regarding Ubiquity Requirements in Software Projects

Hereafter, we are going to present a framework proposal to support activities regarding the definition of ubiquity requirements and their verification and validation. It represents an on-going research aiming at the establishment of a software technology to deal with all nine ubicomp functional characteristics (section 5). We recognize the importance of non-functional requirements for software development. However, the decision regarding initially research support for functional requirements about ubiquitous software projects is due to the fact that ubicomp researches currently are more focused on the functional requirements of software [14].

The proposed framework is composed by a set of facilities associated with ubiquity requirements definition, verification, and validation activities. Fig. 4 shows the supported facilities (white boxes), their relationship with requirements activities, and their respective consumed/produced artifacts (gray boxes). It also shows that framework facilities are based on the ubicomp body of knowledge previously discussed and acquired through secondary and primary studies (systematic reviews and survey).

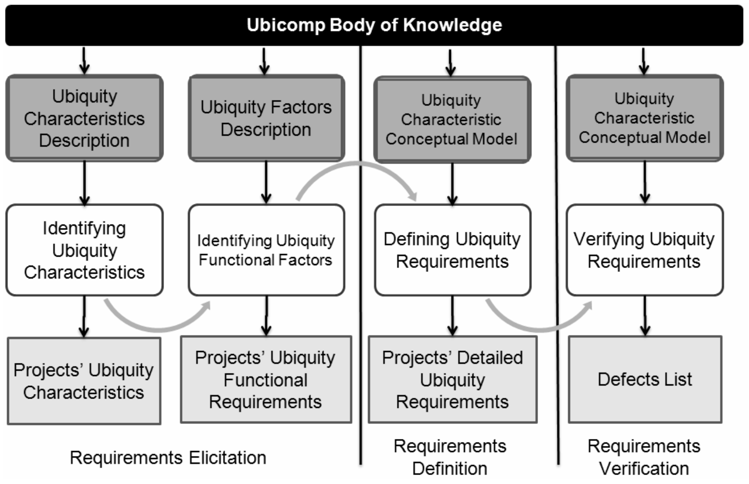

Ubicomp Body of Knowledge
Ubiquity Characteristics Description
Ubiquity Factors Description
Ubiquity Characteristic Conceptual Model
Ubiquity Characteristic Conceptual Model
Identifying Ubiquity Characteristics
Identifying Ubiquity Functional Factors
Defining Ubiquity Requirements
Verifying Ubiquity Requirements
Projects' Ubiquity Characteristics
Projects' Ubiquity Functional Requirements
Projects' Detailed Ubiquity Requirements
Defects List
Requirements Elicitation
Requirements Definition
Requirements Verification

Fig. 4. Overview of the framework

In the next three subsections, we will present how the framework intends to provide a complementary set of facilities that is not given by the conventional software engineering techniques to properly deliver ubiquitous software products in the context of requirements activities.

7.1 Requirements Elicitation

Requirements elicitation is the practice of obtaining the requirements of a system from users, customers and other stakeholders [25]. In the context of this work, we are concerned with ubiquity requirements for software projects. In order to support this activity, the proposed framework has two facilities:

Identifying ubiquity characteristics. This facility intends to help the requirements analyst to identify the ubiquity characteristics that should be considered in the software project. Therefore, for each ubiquity characteristic, a set of questions was defined to help identifying which one of them should be included in the software. The definition of these questions was intended to be the system user because these questions are part of a checklist that guides the user interview activity. As the requirements analyst gets the answers from users, s/he registers in the checklist if the ubiquity characteristic is desired or not.

Identifying ubiquity functional factors. Once the software ubiquity characteristics have been defined, its functional requirements can be identified.

Therefore, a functional factors list associated to the defined ubicomp functional characteristics (section 5) was used to create a complementary checklist. Again, the checklist was defined in a way that it could be used for requirements gathering activities. Thus, based on the identifying ubicomp characteristics step, the checklist is dynamically organized with specific questions considering the selected ubiquity characteristics. For each corresponding functional factor (section 3), the requirements analyst identifies if it is or not necessary.

In the final step, for each selected factor, the requirements analyst is requested to define the correspondent functional requirement. It is important to notice that a ubiquity functional factor is not necessarily a functional requirement. However, it can motivate the definition of a specific functional requirement for the software project.

7.2 Requirements Definition

The next step regards supporting the requirements analyst to detail each identified functional requirement. For this, firstly, the analyst can group the functional requirements (because the relationship between the functional requirements and requirements specification[1] is M:N). Next, a set of information to be defined by the analyst is presented for each requirement that will be detailed.

This information can be different for each ubiquity characteristic and its associated factors. For each ubiquity characteristic, a conceptual model based on their respective functional factors was defined in a way that the models could capture their most relevant concepts and relationships. Based on this information, it was possible to define what must be captured to define requirements in accordance with the ubicomp characteristics.

7.3 Requirements Verification

This step is provided to assure the specified requirements quality. It is supported by an additional checklist that guides the reviewer through the reading of the generated requirements document and helps him to identify possible defects.

For the checklist construction, nine conceptual models (relating to the nine ubicomp functional characteristics and their corresponding factors) were elaborated to capture the most relevant concepts and relationships. These models, besides providing a better way to understand the ubiquity characteristics, allow the definition of what needs to be specified in the requirements to completely cover the chosen ubiquity characteristic.

For instance, from the conceptual model for context sensitivity ubicomp characteristic it is possible to exemplify questions that will compose the verification checklist, such as: (1) Do all information have a data source attached to it?; (2) Are the available devices associated with the context in which they should work?; (3) Is the context information of each available data source described?; (4) Are all information classified in one of the four perspectives: physical, infra-structure, system or user information?

At the end of this step, a list of defects can be created and used to improve the requirements specification quality.

8 Conclusions and Further Works

According to Weiser (1991), computers should be embedded into the environment in such way that their use becomes natural and transparent. This prospective scenario represents new research challenges in many areas like computer network, signal processing, optimization, and artificial intelligence. Particularly, from the point of view of software engineering, these challenges can effectively be observed on the development of different software technologies, such as methodologies, software processes, testing approaches, and quality assurance techniques.

[1] Requirements and use case descriptions.

In this paper we intended to present a framework proposal to support the definition and quality assurance activities regarding ubiquity requirements on ubiquitous software projects. We believe it represents an important step towards the development of increased quality ubiquitous software projects.

The context sensitivity characteristic was chosen as the first one to be included into this framework. This decision was based on the fact that this characteristic seems to be the more investigated by the research community [14].

Moreover, to facilitate the development of this framework, it was important to execute a comprehensive and systematic literature review. It results allowed us to obtain a set of ubiquity definitions and characteristics reflecting the concepts of ubicomp used currently by the scientific community. Thus, this paper also presents contributions as: (1) an updated definition for ubiquitous computing and ubicomp systems; (2) a set of ubiquity characteristics to achieve ubicomp on software projects; (3) the identification of functional and restrictive factors for each ubiquity characteristic; (4) a checklist to characterize ubiquitous software projects using the ubiquity characteristics as a way to evaluate its ubiquity adherence level, and (5) identifying which ubiquity characteristics have been currently considered on ubicomp software projects. All these results were reached using the scientific method represented through systematic literature reviews and knowledge evaluation using surveys.

Finally, it is important to reinforce that the creation of ubiquitous software applications is a hard task [13]. Based on that, as the identification of ubiquity characteristics and factors can provide better understanding of ubicomp, this research can represent an important step to deal with this kind of software and also creating subsidies to support other project development phases, including planning and design.

This work is still in progress. The next steps include: (1) Replicating the survey considering a broader audience, which will allow to observe the feasibility of the initial results; (2) Evolving the definition of the framework to support activities of definition and quality assurance regarding ubiquity requirements specification for ubiquitous software projects; (3) Creating an initial version of an infrastructure to support the framework activities mentioned above; (4) Experimentally evaluate the infrastructure and improving it according to the experimental study results.

Acknowledgments. The authors would like to thank CAPES, CNPq, FAPERJ for the financial support to this work and Dr. Karin Breitman for her valuable comments and motivation. This work has been developed as part of the Experimental Software Engineering and Science in Large Scale CNPq Project (475459/2007-5) and FAPERJ research activities.

References

1. Ali, J.A., Won-Sik, Y., Jai-Hoon, K., We-Duke, C.: U-kitchen: application scenario. In: Proc. of the Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems 2004, May 11-12, 2004, pp. 169–171 (2004)
2. Cheng, B.H.C., Berry, D.M., Zhang, J.: The four levels of requirements engineering for and in dynamic adaptive systems. In: 11th Int. Work. on Requirements Engineering Foundations for Software Quality. Co-located with CAiSE 2005, Porto, Portugal (June 2005)

3. Bossen, C., Jorgensen, J.B.: Context-descriptive prototypes and their application to medicine administration. In: Proc. of the Conference on Designing interactive systems: processes, practices, methods, and techniques 2004, pp. 297–306 (2004)
4. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.-C.: Ambient Intelligence: From Vision to Reality. IST Advisory Group Draft Rep., Eur. (2003)
5. Hatala, M., Wakkary, R., Kalantari, L.: Rules and Ontologies in Support of Real-time Ubiquitous Application. Journal of Web Semantics, 5–22 (2005)
6. Joel, S., Arnott, J.L., Hine, N.A., Ingvarsson, H., Rentoul, R., Schofield, S.: A framework for analyzing interactivity in a remote access field exploration system. SMC(3), 2669–2674 (2005)
7. Kientz, J.A., Boring, S., Abowd, G.D., Hayes, G.R.: Abaris: Evaluating Automated Capture Applied to Structured Autism Interventions. In: Proc. of the 7th Int. Conference on Ubiquitous Computing, Tokyo, Japan, September 11-14 (2005)
8. Jorgensen, J.B., Bossen, C.: Requirements Engineering for a Pervasive Health Case System. In: 11t IEEE Int. Requirements Engineering Conference 2003, pp. 55–64 (2003)
9. Lee, S.H., Chung, T.C.: System Architecture for Context-Aware Home Application. In: Proceedings of the Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, May 11-12, 2004, pp. 149–153 (2004)
10. Goldsby, H., Cheng, B.H.C.: Goal-Oriented Modeling of Requirements Engineering for Dynamically Adaptive Systems. In: 14t IEEE Int. Requirements Engineering Conf., September 11-15, 2006, pp. 345–346 (2006)
11. Niemela, E., Latvakoski, J.: Survey of requirements and solutions for ubiquitous software. In: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, pp. 71–78 (2004)
12. O'Neill, E., Kindberg, T., Schieck, A.F., gen, J.T., Penn, A., Fraser, D.S.: Instrumenting the city: developing methods for observing and understanding the digital cityscape. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 315–332. Springer, Heidelberg (2006)
13. Sakamura, K.: Challenges in the Age of Ubiquitous Computing: A Case Study of T-Engine, An Open Development Platform for Embedded Systems. In: Proceedings of the 28th International Conference on Software Engineering, pp. 713–720 (2006)
14. Spínola, R.O., Silva, J.L.M., Travassos, G.H.: Checklist to Characterize Ubiquitous Software Projects. In: XXI Simpósio Brasileiro de Engenharia de Software, João Pessoa. Anais do XXI Simpósio Brasileiro de Engenharia de Software. Porto Alegre: Sociedade Brasileira de Computação, 2007. vol. 1, pp. 39–55 (2007)
15. Tahti, M., Rauto, V., Arhippainen, L.: Utilizing context-awareness in office-type working life. In: Proc. of the 3rd Int. Conf. on Mobile and Ubiquitous Multimedia 2004, College Park, Maryland, pp. 79–84 (2004)
16. Vainio, A.M., Valtonen, M., Vanhala, J.: Learning and adaptive fuzzy control system for smart home. In: Proc. of the AmI.d 2006, September 20-22, pp. 28–47 (2006)
17. Weiser, M.: The Computer for the 21st Century, pp. 94–104. Scientific American (1991)
18. Zhou, P., Nadeem, T., Kang, P., Borcea, C., Iftode, L.: EZCab: A Cab Booking Application Using Short-Range Wireless Communication. In: Third IEEE International Conference on Pervasive Computing and Communications PerCom 2005, 8-12 March 2005, pp. 27–38 (2005)
19. Hong, D., Chiu, D.K.W., Shen, V.Y.: Requirements Elicitation for the Design of Context-aware Applications in a Ubiquitous Environment. In: Proceedings of the 7th international conference on Electronic Commerce, Xi'an, China, pp. 590–596 (2005)

20. Xiang, J., Liu, L., Qiao, W., Yang, J.: SREM: A Service Requirements Elicitation Mechanism based on Ontology. In: 31st Annual International Computer Software and Applications Conference, 2007. COMPSAC 2007, 24-27 July 2007, vol. 1, pp. 196–203 (2007)
21. Cherif, A.R., Hina, M.D., Tadj, C., Levy, N.: Analysis of a New Ubiquitous Multimodal Multimedia Computing System. In: Proceedings of the 9th IEEE International Symposium on Multimedia, pp. 161–168 (2007)
22. Spínola, R.O., Silva, J.L.M., Travassos, G.H.: Characterizing Ubicomp Software Projects through a Checklist. In: WPUC 2007 - I Workshop on Pervasive and Ubiquitous Computing, 2007, Gramado; Proceedings of WPUC 2007, vol. 1, pp. 1–6. Sociedade Brasileira de Computação, Porto Alegre (2007)
23. Abowd, G.D., Mynatt, E.D.: Charting Past, Present and Future Research in Ubiquitous Computing. ACM Transactions on Computer-Human Interaction (TOCHI) 7(1), 29–58 (2000); Special issue on human-computer interaction in the new millennium, Part 1
24. Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H.: Systematic Review in Software Engineering. Technical Report ES 679/05. COPPE/UFRJ (2005)
25. Pfleeger, S.: Software Engineering: Theory and Practice, 2nd edn. Prentice Hall, Englewood Cliffs (2007)
26. Bo, C., Xiang-Wu, M., Jun-Liang, C.: An Adaptive User Requirements Elicitation Framework. In: Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), vol. 2, pp. 501–502 (2007)
27. Chiu, D., Hong, D., Cheung, S.C., Kafeza, E.: Towards Ubiquitous Government Services through Adaptations with Context and Views in a Three-Tier Architecture. In: Proc. of the 40th Hawaii Int. Conf. on System Sciences, January 2007, p. 94 (2007)
28. Cheng, J., Goto, Y., Koide, M., Nagahama, K., Someya, M., Utsumi, Y., Sshionoiri, A.: ENQUETE-BAISE: A General-Purpose E-Questionnaire Server for Ubiquitous Questionnaire. In: IEEE Asia-Pacific Services Computing Conference, 11-14 December, pp. 187–195 (2007)
29. Kindberg, T., Barton, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frig, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., Spasojevic, M.: People, places, things: Web presence for the real world. In: Third IEEE Workshop on Mobile Computing Systems and Applications, pp. 19–28 (2000)
30. Nawyn, J., Intille, S., Larson, K.: Embedding Behavior Modification Strategies into Consumer Electronic Devices: A Case Study. In: Proc. of the 8th Int. Conference on Ubiquitous Computing (2006)