Characterizing Ubicomp SoftwareProjects through a Checklist

Rodrigo O. Spinola, Jobson Massollar, Guilherme H. Travassos
COPPE / UFRJ – System Engineering and Computer Science Department
Caixa Postal 68.511 – 21.945-970 – Rio de Janeiro – RJ – Brasil
{ros,jobson,ght}@cos.ufrj.br

Abstract

Ubiquitous computing brings new research challenges in many areas. Particularly, from the point of view of software engineering, these challenges can effectively be observed on the development of new software technologies. The current software engineering body of knowledge needs to evolve to represent such new technologies and how they could influence on the planning and developing of ubicomp software projects. Thus, it is fundamental to have an instrument to support knowledge acquisition about ubicomp software projects. The process of acquiring information to reduce the risks concerned with the development of ubicomp software projects should begin with the identification of the impact of ubicomp characteristics on software projects. In this context, this paper presents a checklist-based approach to characterize ubicomp software projects regarding some ubiquitous computing characteristics and factors. The checklist has been used to characterize 12 ubicomp software projects (4 discussed in this paper) suggesting insights about the distance between the states of the art and practice regarding ubiquitous computing research.

1. Introduction

In 1991, Weiser stated that computers should be embedded into the environment in such way that their use becomes natural and transparent. Since then, many software projects have included requirements regarding the initial perception of ubiquitous computing.

This prospective scenario can represent new research challenges in many areas [2] driving us to think about the need of a complementary software engineering body of knowledge [13] [15] to support, for instance, the answers to the following questions: (1) What (new) software engineering technologies are necessary to deal with the ubiquity characteristics of software?; (2) What are the risks associated with ubicomp software projects?; (3) What quality characteristics should software engineers have in mind when accomplishing ubicomp software projects?

Considering this context, the process of acquiring information to reduce the risks concerned with the development of ubicomp software projects should begin with the identification of the impact of ubicomp characteristics on software projects. For this, it is important to identify what characteristics have been currently used on ubicomp software projects and to understand how these projects deal with them.

According to the Cambridge Dictionary: (1) Characteristic is a “typical or noticeable quality of someone or something”. Thus, we can consider characteristic as a typical or noticeable quality of ubicomp software; (2) Factor is a “fact or situation that influences the result of something”. In the context of this work, we can consider factor as a fact or situation that influences an ubicomp software characteristic. This way, we believe that an essential initiative could be to look for some way that allows the evaluation of software projects based on the properties that identify them as ubicomp. Thus, we have started some research intending to support the discovery of these characteristics and factors aiming at the description of a checklist to characterize software projects regarding ubiquitous computing.

In order to reach a solid scientific level, we decided to undertake systematic literature reviews rather than ad-hoc ones [3] [10] to characterize the state-of-the-art. We have conducted two complementary systematic reviews. The first one aimed at the identification of the current ubiquitous computing definition, identifying where it is currently being used and what are its main characteristics. The second one aimed at the identification of functional and restrictive factors associated with these ubicomp characteristics. These factors are concerned with the facts or situations related to the functional and non-functional requirements respectively.

After that, a checklist based approach (supporting the identification of which ubicomp characteristics have been currently considered on ubicomp software
projects) has been proposed using the software project characteristics and their correspondingly factors identified on the systematic literature reviews. To exemplify its use, we have applied such checklist to characterize 8 different ubicomp software projects [9] [18] [1] [4] [6] [11] [21] [7]. The results of this application have been reported in [17]. Recently, we have amended the checklist evaluation including the following 4 projects [8] [19] [14] [12], what can give us some suggestions on how to use it for a larger population of software projects, insights about the distance between the state of the art and practice regarding ubiquitous computing research, and identify what ubicomp characteristics has been more explored on current ubicomp projects. This last point is important to observe new possible research trends regarding software engineering techniques to be applied in ubicomp software projects.

Besides this introduction, this paper contains four main sections. In section 2, the definition, characteristics, and factors of the ubiquity are presented. In Section 3, we propose an approach to characterize applications considering their ubicomp adherence level. In section 4 we present some results obtained with the characterization approach. Finally, in section 5 we present the main contributions of this paper and future perspectives of this research project.

2. Ubiquitous Computing: Definition, Characteristics, and Factors

Weiser (1991) defines ubiquitous computing as being the use of computer through its availability in the surrounding environment, making it effectively invisible to the user. This definition set the origin of the term Ubiquitous Computing and, although it is important to suggest a new computing paradigm, it is not complete. This lack of completeness reflects how this proposal was innovative and technologically ambitious at that time.

In order to reach a more concrete definition of ubiquitous computing, 57 papers were selected from the first systematic review. These papers allowed us to compose a definition for ubiquitous computing, ubicomp system, and identifying ubiquity characteristics [16]. More details about the selection process of these papers can be found on [17].

Ubiquitous computing is present when computational services or facilities become available to the people in such a way that computer is no longer a visible or essential tool to access these services or facilities. To make this happen it is necessary that systems taking part in this scenario take into consideration the following project issues, called ubiquitous computing characteristics: (SO) Service omnipresence - it allows users to move around with the sensation of carrying computing services with them; (IN) Invisibility - ability of being present in objects of daily use, weakening, from user’s point of view, the sensation of explicit use of a computer and enhancing the perception that objects or devices provide services or some kind of “intelligence”; (CS) Context sensitivity - ability to collect information from the environment where it is being used; (AB) Adaptable behavior - ability to, dynamically, to adapt offered services according to the environment where it is being used, respecting its limitations; (EC) Experience capture - ability to capture and register experiences for future use; (SD) Service discovery - pro-active discovery of services according to the environment where it is being used; (FC) Function composition - ability of, based on basic services, to create a service required by the user; (SI) Spontaneous interoperability - ability to change partners during its operation and according to its movement; (HD) Heterogeneity of devices - provides application mobility among heterogeneous devices. That is, the application could migrate among devices and adjust itself to each one of them; and, (FT) Fault tolerance - ability to adapt itself when facing environment’s faults (for example, on-line/off-line availability).

We can notice that ubiquitous computing definition represents the “philosophy” of this new computing paradigm. This way, it defines the ideal conditions where we could access computational resources in a ubiquitous way. On the other hand, ubicomp systems definition has a well-defined scope and it is strongly related to different characteristics that compose ubiquitous computing. It happens because, as ubiquity can be a property of a system, it can be achieved completely or partially. This variation is related to the fact that a particular system can implement or not all the functions concerned with the ubiquitous computing characteristics.

Although it was possible to capture a clear-cut set of characteristics, they are still represented at a high abstraction level, that is, it is possible to define their meaning, but it is difficult to establish their influences on ubicomp software projects. This way, we need to go down into a set of more concrete factors associated with each characteristic. For this, the second systematic review was conducted. More details about the selection process of these papers can be found on [17].

The analysis of the 59 selected papers of the second systematic review was made up in three steps: (1) Identifying the presence of the ubicomp characteristics; (2) Identifying the factors of each characteristic, and; (3) Grouping corresponding factors in factors group. These groups were created as a consequence of the
identified factors and their relationship. For example, for the “Context Sensitivity” characteristic, the factors “Contextualize obtained information” and “Store information” can be grouped on “Context Information Management” factor group, because of their complementary relationship. This grouping made easier the analysis process due to the great number of factors found during the execution of the systematic review (it was possible to identifying 168 factors - 123 functional and 45 restrictive). As a result, it was produced a list of ubicomp characteristics and their corresponding factors, distributed according shown in Table 1.

Table 1. Ubicomp characteristics and #factors.

<table>
<thead>
<tr>
<th>Ubicomp Characteristic</th>
<th>Number of Identified Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO</td>
<td>10</td>
</tr>
<tr>
<td>IN</td>
<td>10</td>
</tr>
<tr>
<td>CS</td>
<td>30</td>
</tr>
<tr>
<td>AB</td>
<td>32</td>
</tr>
<tr>
<td>EC</td>
<td>7</td>
</tr>
<tr>
<td>SD</td>
<td>26</td>
</tr>
<tr>
<td>FC</td>
<td>23</td>
</tr>
<tr>
<td>SI</td>
<td>12</td>
</tr>
<tr>
<td>HD</td>
<td>12</td>
</tr>
<tr>
<td>FT</td>
<td>6</td>
</tr>
</tbody>
</table>

3. An Approach to Characterize Ubicomp Software Projects Based on Checklist

Based on the different perspectives presented in the papers selected by the systematic reviews (section 2), it was noticed that ubiquitous computing usually comes up in its totality when its ten characteristics can be implemented in software projects. Thus, in a first analysis and according to our perspective, to be considered totally ubiquitous, a software project should contemplate the different factors of each ubicomp characteristic. However, we can also have ubicomp software projects with different levels of adherence to the ubicomp characteristics. It can be a consequence of the application domain and project’s requirements, for instance. It is also important to say that those different levels of adherence could represent the absence of some ubicomp characteristic in the software project. This way, it is possible to have software project with different levels of ubiquity.

It is important to state that this paper does not intend to define whether a software project is more ubicomp than other. Its goal concerns only observing how the different ubiquitous computing characteristics have been captured in software projects, supporting some understanding on how they could influence the software project planning and development.

Taking into account the concepts described in section 2, it was designed a checklist to characterize software projects according to their ubicomp adherence level. This characterization comprises three steps: (1) to check the presence of each characteristic’s functional and restrictive factors; (2) to consolidate the software project adherence level of each characteristic based on the presence/absence of its functional and restrictive factors, and; (3) to generate a graph considering the ubicomp characteristics (using the values calculated in step 2) representing the software project adherence level. To support the characterization steps, we built a spreadsheet-based form to calculate the adherence level for each ubiquitous computing characteristic. Doing so, the steps 2 and 3 can be supported by a spreadsheet-based software.

Table 2 shows a fragment of the form that allows the software engineers to capture the following information:

- Ubiquitous computing characteristic: shows the ubiquity characteristics presented in section 2;
- Adherence Level: shows the percentage of adherence based on the Status column. Note that in this initial proposal each factor has the same weight. The adherence level is calculated as the average of the attended factors. The calculus is given by the expression: \( \text{Adherence Level} = (\Sigma \frac{\text{attended factors}}{\text{Number of factors}} \times 100) / \text{Number of factors} \) where:
  - Attended factors are the factors whose status value is 1 for a specific characteristic;
  - Number of factors is the total number of identified factors for a specific characteristic.
- Factor group: shows the factors groups identified in the second systematic review presented in section 2;
- Factor: shows the functional and restrictive factors identified in the second systematic review presented in section 2;
- Status: factor presence (1) or absence (0). The software engineer provides this information.

Table 2. A checklist (fragment) to characterize ubicomp software projects.

<table>
<thead>
<tr>
<th>Ubiquitous Computing Characteristic</th>
<th>Adherence Level</th>
<th>Factor Group</th>
<th>Factor</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO</td>
<td>%</td>
<td>Mobility</td>
<td>User section management</td>
<td>1 or 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To deal with the user's mobility</td>
<td>1 or 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Factor n</td>
<td>1 or 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Service management</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Factor Group n</td>
<td>...</td>
</tr>
<tr>
<td>IN</td>
<td>%</td>
<td>...</td>
<td>...</td>
<td>1 or 0</td>
</tr>
</tbody>
</table>
A complete version of the checklist can be found at http://www.cos.ufrj.br/~ros/ubforms.html.

As the user fills in the Status column, the Adherence Level column can be calculated for each ubiquitous computing characteristic. As a final step, the evaluated percentage values are used to draw a graph that represents the adherence level of the software project according to the perspective of ubiquity.

4. Applying the Checklist

In this section, the results of applying the checklist on the sample of 12 ubicomp software projects (state-of-the-practice) are discussed. In Table 3, it is summarized the results regarding the first 8 software projects presented in [17]. Table 4 describes the additional 4 software projects that amend those previous data.

Table 3. Summary of results [17].

<table>
<thead>
<tr>
<th>Ubicomp Characteristic</th>
<th>% of the Attended Factors per Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SP1</td>
</tr>
<tr>
<td>SO</td>
<td>0</td>
</tr>
<tr>
<td>IN</td>
<td>50</td>
</tr>
<tr>
<td>CS</td>
<td>23,3</td>
</tr>
<tr>
<td>AB</td>
<td>9,4</td>
</tr>
<tr>
<td>EC</td>
<td>50</td>
</tr>
<tr>
<td>SD</td>
<td>25</td>
</tr>
<tr>
<td>SI</td>
<td>16,7</td>
</tr>
<tr>
<td>HD</td>
<td>0</td>
</tr>
<tr>
<td>FT</td>
<td>0</td>
</tr>
</tbody>
</table>

The data used to reach the results of Table 3 and 4 can be found at http://www.cos.ufrj.br/~ros/summarizedData.html. From that data set it was generated the graph on Figure 1 representing the number of ubicomp attended factors of each investigated software project. It also shows the total number of identified factors for each ubicomp characteristic (dashed horizontal lines) from the technical literature.

Observing the graph on Figure 1, the additional 4 software projects did not change the previously observed behavior [17]: if the number of factors identified for each ubicomp characteristic increases or decreases (dashed horizontal lines), the same happens with the number of factors implemented in the software projects (vertical bars). The exception to this behavior is the function composition characteristic (FC). None of the 12 software projects reported to deal with any of the 23 factors regarding this characteristic. This behavior is not expected by the fact that this characteristic has been considered required in about 32.2% of the analyzed papers from the second systematic review. A possible reason could be the difficulty to deal with the inherent function composition characteristic factors complexity.

Table 4. Description of additional software projects.

Software Project 9 [8]  
This work presents an example of an automated capturing application which provides access to details of discrete trial training, a highly structured intervention therapy often used with developmentally disabled children.

Software Project 10 [19]  
This work proposes to use a context sensitive and proactive fuzzy control system for controlling the home environment.

Software Project 11 [12]  
This paper presents ViTo, a system to allow behavior modification in embedded electronic devices.

Software Project 12 [14]  
This paper approaches the design of ubiquitous computing systems in the urban environment as integral to urban design. To understand the city as a system encompassing physical and digital forms and their relationships with people’s behaviours, this paper presents the developed, applied, and refined methods of observing, recording, modelling and analysing the city, physically, digitally and social.
Figure 1. Attended factors.

An additional observed behavior is regarding the focus on some specific ubicomp characteristics. It seems that ubicomp software projects pay more attention to the invisibility, context sensitive and adaptable behavior characteristics. The other ones seem to appear as isolated initiatives yet, even considering the analyzed projects representing examples along the years 2000 to 2007, where some technological evolution took place. We did not found any feasible explanation for this behavior. However, it seems to represent a gap between state-of-the-art and state-of-the-practice that deserves further investigation.

5. Final Comments

The extent and the level of abstraction of the principles that guided the initial work of [20] allow many interpretations of what ubiquitous computing is. This way, the research aiming at the design of this checklist brings together, besides the checklist, some contributions:

- A current definition for ubiquitous computing and ubicomp systems;
- A proposal of a set of concrete characteristics to achieve ubiquitous computing;
- Identification of functional and restrictive factors for each ubicomp characteristic;
- A proposal of a checklist of ubicomp software projects using the ubicomp characteristic as a way to evaluate the adherence level, and identify which ubicomp characteristics have been currently considered on ubicomp software projects. As said before, this result can give us some directions to new trends of research regarding software engineering techniques to be applied in ubicomp software projects
- The list of characteristics and factors included in the checklist can guide the software requirements elicitation activity.

Besides that, the software projects analyzed allowed us to conclude that ubicomp software projects are too much restricted on their scope; they are still limited to research centers, and; they present solutions that take into consideration just a small number of ubicomp characteristics and its respective factors.

The results obtained with the execution of the checklist are particularly important because they allow us to have an insight of which ubicomp characteristics are currently being implemented on ubicomp applications. Certainly, the application domain determines that not all the ubicomp characteristics shall be or need to be implemented. However, the low adherence level can provide indications about the difficulties found in the attempt to deal with these characteristics’ factors. These difficulties can be directly related to the inherent complexity of each one, what lead us to some additional questions: (1) Which are the influences of each ubicomp characteristic and its factors on the software architecture and design?; (2) Which (new) software engineering methodologies are needed to deal with these characteristics/factors?; (3) How to guarantee the software quality regarding these characteristics?; (4) What are the risks associated with the implementation of each characteristic/factor?; (5) Which technologies has to be created or adapted to support these characteristics/factors? To exemplify these kinds of issues, in the context of ambient intelligence research area, some issues regarding these last two questions have been made [5]. Traditional software is often based on two main integration principles. Firstly, software is partitioned into abstraction layers with fixed, simple interfaces between layers. Secondly, systems are often vertically integrated with fixed, proprietary linkages between
components. The ambient intelligence vision challenges both these principles. The need to anticipate user behavior, to gather contextual information, to detect and adapt to all kinds of changes entails a corresponding need for extensive cross-layer interactions. This scenario sets new challenges for methodologies and technologies for service and application development [5].

The creation of ubicomp software applications is a very severe problem [15]. Based on that, the identification of ubicomp characteristics and factors seems to indicate an important step to deal with this problem since its understanding can provide subsidies to establish the software planning and development phases according to the ubicomp characteristics/factors adopted in a particular project.

Finally, it is important to enforce that the characteristics and the characterization approach presented in this paper represent just starting points to new research activities regarding ubiquitous computing in software engineering and should be improved. For this, we are currently executing a survey considering a set of research centers that consider these issues in their research.

6. Acknowledgments

The authors would like to thank CAPES and CNPq (Experimental Software Engineering Grant: 472135/2004-0) for the financial support to this work. This work has been partially developed in collaboration with HP Brazil R&D.

7. References