9
Conclusions

This thesis aimed at understanding what pervasive games are and proposing a methodology for conceptual design of activities in these kinds of games. This chapter presents an overall discussion for this research work, with concluding remarks. For further discussion on the analyzed games, pervasive game features (and their checklists), and prototypes developed in this research work, we gently ask the reader to refer to the appendices.

9.1
Immaturity in the area

The field of pervasive games is still immature and there are neither standards nor consensus about the central issues. This applies both to concepts (as the definition of pervasive games) and methods for development of the software part of pervasive games.

The first problem we had was to understand the concept of pervasive games. The research field of pervasive games is young, as the first games labeled as "pervasive games" date back from 2001.

We have tried to search for the ultimate definition for *pervasive games*, but in the end the conclusion is that this is hardly achievable. The research work on this topic has revealed that several different parties had been involved with exploring pervasive games – as computer scientists, designers, game practitioners, for example. All of them tried to define pervasive games according to their specific backgrounds. The end result is that there are a lot of viewpoints on the subject, some of them conflicting. This variety brings a lot of confusion to this area, and sometimes makes it a bit difficult for newcomers to grasp the essence of it. However, the debate generated from this variety of perspectives is very good to enrich the discussion on this subject, making it possible to explore all alternatives. It is important to remember that the research field of (digital) games is multidisciplinary. We hope that our survey is helpful for the reader to get acquainted with the field of pervasive games, by discussing the main concerns of common approaches for defining pervasive games.

Although this research work does not try to come up with yet another definition for pervasive games, it defined a boundary criteria (Chapter 4) to identify games as pervasive games (more specifically, *pervasive mobile games*):

1. Games using mobile devices (phones, PDAs, tablets):	mandatory
2. Games that are context-aware:	mandatory
3. Games that access remote data on the move:	optional
4. Multi-player players:	optional

Contrary to cultural viewpoints found in the literature (Montola *et al.* 2009; Montola *et al.* 2006; McGonigal 2006; Davies 2007), our viewpoint is inherently *technological*, as it emphasizes technological aspects (context-awareness, mobile devices, networking) and presents artifacts related to Software Engineering processes (the domain specific language). However, contrary to other technological viewpoints we do not see pervasive games "just as an application of pervasive/ubiquitous computing", but as another type of application that may apply those concepts (along with other from other fields). In this sense, it is similar to the notion of "computer-augmented games" (Section 3.2.2). The notion of "computer-augmented games" is helpful to emphasize that a pervasive game *is not just* a software application, but something *that is realized through* a software application.

This research work has placed emphasis on the *mobile* component of pervasive games by labeling those games as *pervasive mobile games*. We believe mobile devices (especially smartphones) can be the main facilitators for realizing this kind of game.

9.2 What makes pervasive games unique?

Motivated by the confusion in this field, this research has formulated the question – *what makes pervasive games unique?* Guided by this question, we have started to investigate pervasive games thoroughly to discover central characteristics that make those games very different from traditional digital games. In this regard, the boundary criteria were helpful to select which games should be analyzed – otherwise, there would be too many possibilities.

Also, due to distinct characteristics of pervasive games, this research work considers that pervasive mobile games are “first class citizens” and not “an exercise in technology”.

With that frame in mind, this research work produced knowledge about the domain of pervasive mobile games, with the following levels:

- Boundary criteria for approaching the problem domain – the pervasive mobile games;
- A list of sixteen *pervasive game features*, and descriptions of each item;
- A set of *verification questions* for each pervasive game feature;
- A list of perspectives to group pervasive game features;
- A categorization scheme for the concept of *pervasiveness.*

The initial pervasive game feature list can be useful to identify pervasive mobile games, or to inspire new ideas for new pervasive mobile game projects. For the software part of pervasive mobile games, the features could be used to discover software requirements (functional and non-functional).

However, this initial feature list and perspectives need refinement, specially considering its usage as helping discovering software requirements. It is important also to discover how the features and perspectives favor or hinder others, as well as other interdependencies. This research work has provided an initial step towards this direction, by starting exploring this information in Section 5.2 and Appendix B. In particular, Appendix B provides a more thorough discussion on the initial pervasive feature list and the corresponding checklists. The section about future works (Section 9.5) provides more directions on this issue.

This research work has also provided an initial analysis on how the pervasive features are related (the *pervasive game feature perspectives*) in form of a set of *perspectives* – ways to observe a concept (the pervasive mobile game) through different lenses, where each one concerns with a different aspect of the concept. Six perspectives have been identified: spatiality, sociality, availability, accessibility, sensor capability, and immersion.

9.3 The prototypes

We wanted to explore pervasive mobile games through experimenting with prototypes. In this regard, this research has produced seven innovative prototypes: *Location-based Quiz Game*, *Pervasive Word Search*, and *The Audio Flashlight* series (5 games). *Pervasive Word Search* has been used as an example of using our methodology throughout this research work. Appendix C presents details about the other prototypes.

The prototypes have explored a number of issues, including context-awareness, mobile networking, multi-player gaming, and novel ways of interaction in games (with sensors). In particular, we have been exploring interacting with sensors in mobile phones since this alternative became possible. In 2007, when Nokia first unveiled an API for programming the accelerometer in their N95 smartphone, we developed *Accelerinvaders* (Valente 2007) which was the first game on the Nokia N95 smartphone (and possibly on any mobile phone platform) to use the accelerometer sensor for game input. The findings of this experiment had been applied to *The Audio Flashlight 1* (and the sequels), which has been the first non-visual game for mobile phones, also using gesture-based input. This specific research has produced two publications, (Valente *et al.* 2008; Valente *et al.* 2009).

Developing the prototypes has shed light on the importance of the *uncertainty handling policy (UHP)*, discussed in Section 5.1.10 and Appendix B.10. The practical experience with sensor uncertainty and literature research has motivated including the UHP up front in the requirements for pervasive mobile games, reflected in the methodology this thesis proposed.

Prototype development also formalized an open source project for using Bluetooth networking in Qt applications for Nokia Symbian devices – the QBluetooth project (Valente and Ftylitakis 2011), whose original author is Nikolaos Ftylitakis.

9.4 Supporting pervasive mobile game development

As a way to support the development of new pervasive mobile game projects, this research work has defined a domain specific language (DSL) for specifying activities in those games, using mobile devices, sensors and actuators as the main interface elements. This DSL has two levels. The first level represents an ontological level, where it defines the main concepts for specifying activities, as well as consistency rules for these concepts. The second level represents an operational level, with scenarios as the main concept. This operational level defines languages for specifying scenarios in text and graphics formats.

The proposed methodology has concentrated on providing features and elements as they relate to the player experience and central characteristics of pervasive mobile games. Defining a common vocabulary is important to improve the quality of communication in the area.

This research work was also concerned with defining a methodology that would not be too much intrusive for game developers. Game developers are notoriously away from very heavy-weight or detailed processes for software development. In this regard, this research work has defined a DSL for specifying activities on the conceptual design level, not including implementation details. The visual elements for the DSL uses a known standard, being based on UML Activity Diagrams – an alternative for specifying activities visually that is not as heavy-weight as other available options (as UML Sequence Diagrams, for example).

We have applied this methodology for specifying activities in some of the prototypes. The light-weight nature of this methodology helped in catching a glimpse of the “big picture” in those games. As this methodology does not concentrate on low-level (implementation) details, it has served as a “guide” for the implementation – a way to keep focused on the intent of activities and not getting lost in the source code. We also have been able to find error in the implementation and correct the textual specification with the diagram artifacts of the DSL.

9.5
Future works

This section discusses some possible future works derived from this research.

9.5.1.1
Pervasive features and software requirements

Regarding the software part of pervasive mobile games, a future work would be producing a more detailed and specific set of non-functional requirements for pervasive mobile games, derived from the pervasive game features and perspectives. This also includes alternatives for the operationalization of the requirements – design alternatives for meeting a given requirement. An alternative for this is to use the NFR framework (Chung *et al.* 1999; Chung and Leite 2009). The NFR framework is an approach to gather requirements of some domain along with their interdependencies and operationalizations.

Discovering functional and non-functional requirements for pervasive mobile games may foster the development of middleware to address specific requirements of pervasive mobile games, making the development of those games easier.

Another future work related to this topic could be refining the methodology to make it "more formal", in regard to specifying the software requirements.

In this sense, Heitmeyer and co-authors (1998) provide the example of the SCR (Software Cost Reduction) tools. Their method shares similar concerns with the one this research work has presented: applications that use sensors and actuators. The SCR provides a tabular notation for specifying requirements and a set of tools to check consistency automatically. The method is based on a mathematical foundation proposed by Parnas and Madey (1995), which aim at representing software documentation (including requirements) as sets of mathematical relations.

However, it is important to notice that tools as SCR have been developed for highly-specific domains (software applications for military airplanes), which may have stricter requirements (especially to safety and real-time concerns). Those kinds of domain are very different from the domain of pervasive games.

Pervasive mobile games may not require such degree of sophistication (and specialization), and as discussed earlier, a high degree of formalism may steer away game developers, as software development is only a component of a much bigger process in game development.

9.5.1.2
Domain Engineering for pervasive mobile games

Domain engineering is a process defined by Arango (1988), which concerns producing a reusable infrastructure to solve problems in a domain. As Arango (1988) defined it, the domain engineering process consists of three stages: domain analysis, domain design (infrastructure specification), and domain implementation (infrastructure implementation). The purpose of each stage has been defined originally as this:

1. Domain analysis: the identification and acquisition of reusable information in a problem domain to be reused in software specification and construction

2. Infrastructure specification (domain design): the modularization and organization of reusable information by a reuse plan; for example, libraries of sub-programs, object bases, libraries of transforms, database schemas

3. Infrastructure implementation (domain implementation): the design and encoding of the pieces resulting from the specification process using particular representations required by technology of reusers; for example, procedural programming languages, object-oriented languages, …

In this regard, this research work has performed a domain engineering process for pervasive mobile games, by defining a domain boundary, organizing domain knowledge, and producing a DSL for specifying activities in pervasive mobile games.

However, this research work produced a DSL in the specification level. For example, currently it is not possible to use a computer to process the language, nor to have consistency rules automatically applied, nor to generate source code from it.

In this sense, a future work could be implementing the DSL, which would be another part of a Domain Engineering process. For example, implementing a DSL could involve developing a language parser to use in a “smart editor”, which would assist developers in several tasks. An example could be performing automatic consistency checks, and preventing developers from using invalid constructs when specifying activities.

9.5.1.3
Validating the methodology

The methodology this research work has produced is a first step towards a more formal treatment of developing software for pervasive mobile games. A future work in this direction would be having more researchers to apply this approach with more prototypes as an alternative to validate the concepts this work has proposed.

9.5.1.4
Experimenting with external sensors and actuators

This research has produced prototypes that use sensors and actuators that are embedded in mobile phones. Although the proposed approach *includes* external (*i.e.* not embedded) sensors and actuators, this alternative has not been explored in this work, in practice. A future work would be exploring this alternative and observing its effects, especially regarding how it works with the pervasive features. This alternative is useful to realize *event games* (Section 5.1.7). An option for this task is to use the Arduino platform (2011). The developers of Arduino describe it as "*an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board*". This platform can use several sensors and actuators, as well as networking, making it suitable for exploring pervasive mobile games that interact with (external) resources in the physical world.