Extending Use Cases To Support Activity Design In Pervasive Mobile Games

Luis Valente Bruno Feijó

VisionLab/Dept. of Informatics/PUC-Rio
Rio de Janeiro, Brazil

Abstract

Computer games are creative projects that require the input of professionals with very diverse backgrounds, including game designers, artists, and software developers. Game development frequently is a complex process due to different expectations of the involved stakeholders. With pervasive games, this situation becomes more chaotic as there are not specific processes devoted to the design and development of this type of game. In this paper, we propose a template-based language to design activities in pervasive mobile games in the conceptual design phase, helping to fill a gap between the preproduction and production stages of this type of game. We define a template for activity specification based on an extension of traditional use case templates. This extension helps in fulfilling a set of general goals that the activity modeling should address. We also present examples of using the proposed modeling approach in a real game.

Keywords:

pervasive mobile games, activity design, game development

Authors' contact:
{lvalente, bfeijo}@inf.puc-rio.br

1. Introduction

The process of developing a digital game (either a pervasive game or traditional game) has two broad stages: preproduction and production. The preproduction stage carries out most of the "creative" tasks related to a game. In this stage, game designers and artists are the main players. The game designers work on defining the storyline and non-player character behavior, among other tasks. The artists create assets such as graphical art, animations, music, and audio effects, among others. The production stage transforms the initial game vision (created in the preproduction stage) into software.

Game development processes usually are very complex due to the nature of the involved stakeholders, which have different expectations and worldviews. For example, a game designer might not understand the technical limitations of artificial intelligence implementation when designing the behavior of non-player characters. A software engineer might interfere in the initial game vision because he thinks some features are infeasible to implement (*e.g.,* technical constraints, tight deadlines). In this regard, a game development process is different from traditional software development processes because traditional software (*i.e.*, productivity software) does not have a preproduction stage.

Another important difference between digital games and traditional software refers to the main goal that these applications need to fulfill. Traditional software usually concerns productivity, while digital games offer entertainment. The focus on providing entertainment raises issues that are not present in traditional software. For example, digital games need to address abstract requirements such as "fun", "flow", and "enjoyment", among others. There are not many academic works that address these requirements, known as "emotional requirements" [Callele *et al.* 2008; Furtado *et al.* 2010].

Callele and co-authors [2005] summarize the aforementioned issues by stating that many problems in game development processes (including project failure) arise because the transition between the preproduction and production stages is not carried out properly. For example, major sources of problems in game development relate to ambitious scope and feature creep [Petrillo *et al.* 2009; Kanode and Haddad 2009]. Petrillo and co-authors [2009] also cite other problems, such as the cutting of features during the development phase, delays, and project schedules that are too optimistic. As a result, some researches [Callele *et al.* 2005; Callele *et al.* 2011] argue that in a game development process, it is necessary to create a common language that the involved stakeholders use to communicate more effectively.

When considering pervasive games, the situation becomes worse and more complex as pervasive games are a recent form of entertainment and there is no consensus about what they are. The literature on pervasive games mixes up preproduction and production issues. Part of the literature handles preproduction issues (*e.g.,* game studies, game design) and another part is concerned with production issues (*e.g.*, ubiquitous and pervasive computing) [Valente *et al.* 2013]. Also, to the best of our knowledge, specific methodologies to address pervasive game development do not exist.

Based on our experience on developing mobile and pervasive games, in this paper we propose a template-based language to design activities in pervasive mobile games in the conceptual design phase, helping to fill a gap between the preproduction and production development stages of pervasive games. In a nutshell, a pervasive mobile game is a context-aware game that uses mobile devices, being a subset of pervasive games.

We define a template for activity specification based on extensions of traditional use case templates. An activity captures the behavior of players when interacting with other players, non-players, the game scene, and the game itself through context-aware mobile devices. These extensions aim at helping to fulfill a set of general goals that we devised for activity modeling in pervasive games, which we present in Section 4. We present two examples of using the proposed modeling approach in a pervasive mobile game that we developed.

This paper is organized as follows. Section 2 presents the definition of what we consider as pervasive mobile games in this paper. Section 3 presents related works. Section 4 presents our proposal for activity modeling in pervasive mobile games. Section 5 presents two examples of using activity modeling in a real game. Finally, Section 6 presents the conclusions of this paper.

2. What Are Pervasive Mobile Games?

In the literature, there is a lot of confusion over definitions and formalisms related to pervasive games in general [Valente *et al.* 2013]. In this paper, we consider "pervasive mobile games" as games that are played in the physical world, where players use *context-aware* mobile devices to enable the interaction between the environment (physical world) and the virtual world, which creates a mixed-reality. Pervasive mobile games are a subset of the possible pervasive games.

We understand that pervasive mobile games must be "context-aware" in order to support mixed-reality. Dey [2001] defines context as *"any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. A system is context-aware if *"it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"* [Dey 2001].

We use the term "mobile device" for a device that has the following two characteristics: 1) Portability – a battery-powered device that users are able to carry around; and 2) Networking capabilities that are not limited to a specific physical place. Examples include mobile phones and tablets that use networking infrastructure provided by a mobile operator.

A "context-aware mobile device" is a "mobile device" equipped with sensors that enable it to sense context information (especially, environmental properties). A pervasive mobile game may require networking, but this is not mandatory.

By using context-aware mobile devices as elements of a mixed-reality, pervasive mobile games are able to provide game activities that happen out of a game device. A consequence is that these games may require or foster player movement in local spaces (not necessarily requiring network resources on the move).

We do not consider "portable" or "mobile" games as pervasive mobile games. A "portable game" does not require networking, while a "mobile game" requires it. Although both types of games are played with mobile devices, they are not context-aware.

Currently, we consider smartphones and tablets as the main platform for pervasive mobile games because: 1) These devices are easily accessible to the general public; 2) These devices are equipped with embedded sensors that makes it possible to implement (some) context-awareness required by pervasive mobile games; and 3) These devices makes it possible to have mobility in pervasive mobile games (*e.g.,* gameplay in physical places using networking or not). In this paper, we use the term "mobile device" as an encompassing term to "smartphones and tablets".

3. Related Work

We were not able to find works related directly to specifying activities in pervasive games. Concerning traditional games, the literature provides some works that explore designing interactions using use case diagrams, such as [Tang and Hanneghan 2008; Taylor *et al.* 2006; Bethke 2003; Siang and Rao 2004].

Tang and Hanneghan [2008] propose a domain specific modeling language to design serious games. This language has a component named as "scenario", which concerns *"the construction of a situation which consists of characters, objects, objectives, scripted events and problems to be solved through game-playing"* [Tang and Hanneghan 2008], which is similar to the scenario concept that we propose (Section 4.5). They propose extending use case diagrams with decision trees to model scenarios. The language by Tang and Hanneghan [2008] aims at defining higher-level abstractions to design games, excluding low level details of software modeling.

Taylor and co-authors [2006] propose a model named "computer game-flow design", which aims at modeling the flow of player actions through a game level. The computer game-flow design model also incorporates use case diagrams extended with decision trees

elements and flow directionality identification. Similarly to us, Taylor and co-authors [2006] also share the concern of creating a tool that artists, designers and software engineers can use to communicate.

Bethke [2003] explores UML use case diagrams to describe core interactions in a game, in the context of a traditional game design process. Bethke does not propose extensions to address specific game issues.

Siang and Rao [2004] describe gameplay as interactions between players and the game. They define "player to object" and "object to object" interactions. Based on this idea, they use UML use case diagrams and UML class diagrams to describe the interactions in two existing games, through reverse engineering.

Our activity modeling approach differs from the mentioned related works in several ways. Firstly, it provides a text-based tool to design activities, which may be simpler to use. Secondly, we consider possible impacts due to uncertainties related to sensors and networking, which may lead to non-functional requirements that may hinder game experience if they are not addressed properly (Section 4.4).

4. Activity Modeling

In our modeling approach, an *activity* captures the behavior of players when interacting in the mixed-reality with other players, non-players, the game scene, and the game itself. Each player uses a *context-aware mobile device* as the main interaction interface with the game (for input and output).

The game scene is part of the mixed-reality that the game creates, being a *local space* (a physical space where the game happens) that may be augmented with *smart objects*. The game may use *smart objects* to provide information to players, and as secondary input sources. These smart objects are deployed in the physical environment and are connected to a central entity (*e.g.,* a server) that runs the game. For example, these devices may be: 1) plain output devices (*e.g.,* public displays, audio speakers, light sources); 2) mechanical devices equipped with actuators; and 3) Autonomous computing devices equipped with sensors and/or actuators (*e.g.,* wireless sensor devices [Mottola *et al.* 2006]).

The literature provides some examples of using these kinds of objects in games and interactive installations. The ALICE project [Bartneck *et al.* 2008] creates mixed-reality environments using public displays and mechanical devices equipped with actuators. In *Manhattan Story Mashup* [Tuulos *et al.* 2007]), the game uses a public display to convey information to players. The *Fun-in-Numbers* project [Chatzigiannakis *et al.* 2011] presents several games that provide interactive installations using physical objects to communicate with players. In particular, the *Magnetize Words* project [FinN 2014] provides a demo that uses light sources as output devices. Montola and co-authors [2006] presents a game using wireless sensor devices.

These are the general goals that activity modeling aims at addressing (items in *italics* represent concepts that we will detail in later sections):

G1. Specify the *interactions* involving players, non-players, and the game scene elements (*local space*, *smart objects*), using context-aware mobile devices;

G2. Specify adequate responses (*acknowledgments*) to inform the player about what is happening in the interactions, especially when the interaction is *implicit*. Also, the *interaction granularity* should be compatible with the sensors used for input;

G3. Specify *game logic actions* that change the game state. When the game state changes, the game must inform the player about the new state through *acknowledgments*;

G4. Specify *game logic events* that the game needs to inform players (through *acknowledgments*);

G5. Specify how the game delivers *acknowledgments* through players (*e.g.,* the *actuators* or *smart objects* that the game uses to provide information);

G6. Specify how the game handles *uncertainties* related to technology (*e.g.,* especially sensors and networks);

As a tool for the conceptual design stage, we consider that activity modeling should have the level of abstraction equivalent to traditional use cases [Cockburn 2000]. This means that we are more concerned about intent, and not implementation details. We believe that this approach yields enough balance when considering the diversity of the involved stakeholders (*e.g.,* game designers, artists, and software engineers). Next subsections present the main activity modeling concepts (Sections 4.1 to 4.5), along with the activity specification template (Section 4.6).

4.1. Main Interacting Entities

Actors are the main entities that interact with the game. Among actors are *players*, *non-player characters*, and the *game* itself. Non-player characters may be represented virtual characters and humans. Human non-player characters may have direct participation in the game (*e.g.,* as actors representing roles) or indirect participation (*e.g.,* as sources of game content) [Valente *et al.* 2013].

We distinguish between primary actors and secondary actors. The primary actors are the actors that start the activity, while secondary actors are other actors

involved or affected by the primary actor actions (typically, other players).

4.2. Player Interactions

Player interactions can have *implicit* or *explicit* styles. *Explicit interactions* occurs through direct (conscious) player commands, meaning that he has conscious intention to start something to achieve a goal. An *implicit interaction* means that the interaction occurs inadvertently from the player point of view. In other words, the player is not aware that he is interacting with something when this interaction happens. Schmidt [2000] defines an implicit interaction as *"an action, performed by the user that is not primarily aimed to interact with a computerized system but which such a system understands as input"*.

An example of explicit interaction would be pushing buttons on a joystick. An example of implicit interaction would be a player walking into a room, and suddenly a door starts to open.

A motivation for having implicit interactions is when designers want to deliver activities that might cause ambiguity, surprise, unexpectedness, or disruption. This approach could also be used to draw attention to some aspect in the activity. For example, this is the idea that Rogers and Muller [2006] use in their framework to design sensor-based interactions for promoting reflections and exploration in play.

Interactions have a *granularity*, which is either *discrete* or *continuous*. A *discrete granularity* means that the player is able to use a finite set of options to perform the interaction. For example, pressing buttons, reading a RFID tag, and positioning a device in four possible directions. A *continuous granularity* means that the interaction has an unconstrained set of options on how to perform it. An example would be walking in a place to find WiFi access points, or using a mouse in traditional GUI applications. The interaction granularity must be compatible with the involved sensors for the purposes of the activity; otherwise the game experience may be affected negatively. An example of incompatibility would be using keyboard arrow keys to draw a picture on a screen.

We understand that the game needs to provide adequate responses (*acknowledgments*) in interactions, especially when interactions are sensor-based and have *implicit* style. Otherwise, the game experience might be disturbed (especially concerning implicit interactions).

4.3. Acknowledgments

We define an *acknowledgment* (or "ack") as a response that the game sends to players to inform them about an important occurrence. The acknowledgments have content (*e.g.,* the information to be delivered) and they are expressed through some media. The game delivers acknowledgments to players through output channels in mobile devices (*e.g.,* display, audio, vibration) and smart objects. Here are some examples:

- The player touched the mobile device screen and the game plays a sound to notify the player that it recognized and registered the input;
- The game registered an implicit input received from the player device. For example, the game started a Bluetooth search (without explicit player command) and later found some devices. The game then notifies the player about this occurrence through audio;
- The game updated its internal state and notifies the player about the new status.

We stress the importance of acknowledgments to keep the players informed about what is happening in the game. When dealing with sensor-based interactions, the game might be initiating several interactions on behalf of the player (especially regarding implicit interactions). This means that the players may not be in control at all times. When not handled properly, this lack of control may degrade the user experience, especially when people are not expecting an interaction to occur [Rogers and Muller 2006].

4.4. Technological Uncertainties

Technologies (especially sensors and networking) have inherent limitations regarding precision, accuracy, availability, and other properties. This includes noticeable boundaries, breaks, or gaps among technology components (*i.e.,* seams). As a consequence, the game needs to handle these issues to keep the user experience smooth. There are five general strategies to handle technology limitation issues related to sensors and networking [Benford *et al.* 2006; Bell *et al.* 2006]:

- Remove: designing activities so that limitations never appear in the game. This includes using improved technologies (which is not always possible) or designing activities that fit into the technology limitations;
- Hide: anticipating issues and "correcting" them before the player has a chance to face it. Contrary to the remove strategy, in this case the limitations appear in the game, but are "corrected" before the player notices them;
- Manage: includes having fall-backs to use when the primary mode of operation fails. In other words, the game adapts itself to the circumstances by having several modes of operation;

- Reveal: consists of presenting the limitations to users and letting them decide how to act. For example, mobile phones display the operator signal strength in the user interface and the users may decide to go to places with better signal to make calls with better quality;

- Exploit: means acknowledging the existence of issues and integrating them into the game as a feature.

Handling uncertainties is critically important to keep the player experience smooth. Ignoring the role of uncertainties in a pervasive game probably will disrupt the game experience. As a result, our modeling approach includes uncertainty handling policies as part of activity specification. The policies that we refer to are one of the five general strategies that we presented in this section. For more details on uncertainty handling and specific examples, the reader should refer to [Valente and Feijó 2013].

4.5. Scenarios

A scenario consists of a sequence of steps that describe the primary actor achieving or failing to achieve the main goals of an activity. Similar to use cases, the activity specification has a main (or default) scenario where the primary actor successfully achieves the activity goals. The activity specification has alternative scenarios to represent situations where the primary actor fails to achieve the activity goals. Scenarios need to address the general goals that we described at the beginning of Section 4.

The steps in scenarios describe *actions*. An *action* is something that affects, changes or acknowledges the game state. Actions can be *player interactions* (Section 4.2), *acknowledgments* (Section 4.3), or a *game logic action*. A game logic action refers to actions that evaluate game inputs, apply game rules, or run other processes related to game logic.

Game logic events are events that result from game logic processing. These events represent occurrences that affect the current game state. Hence, the game needs to inform the players about these occurrences (through *acknowledgments*). *Interaction events* are events (related to game logic) resulting from player interactions. For example, a player might issue a command to move in a virtual environment and hits a wall. The occurrence of "hitting a wall" is the interaction event in this example.

4.6. Activity Specification Template

The activity template is based on traditional use case templates, as the ones provided by Cockburn [2000]. We devised some extensions to represent the concepts that we defined in previous subsections. Figure 1 illustrates the complete template for activity specification, with all fields.

Primary Actors	Actors that start the activity
Secondary Actors	Other actors affected in the activity
Level	"player level", "subfunction", "mobile device events"*
Inputs*	Input sources related to mobile devices (*e.g.*, sensors, screen)
Outputs*	Output channels in mobile devices that the game uses (*e.g.*, display, vibration, audio)
Smart objects*	The smart objects used in the activity
Interaction granularity*	"discrete"*, "continuous"*, "mixed"*
Control*	"explicit"*, "implicit"*, "mixed"*

Overview
Brief overview if desired.

Main scenario
The default flow, with numbered steps.

Alternative scenarios
The specification of alternative scenarios, which can be successful or not.

Operation parameters*
Specification of these requirements if necessary.

Uncertainty handling policy*
List of policies and how they should be applied.

Miscellaneous
Miscellaneous and open issues.

Figure 1: The complete activity template. Fields marked with an asterisk represent extensions that we propose in this paper

The *primary actor** is the player that starts the activity. The *secondary actors** can be other players involved in the activity. The game being designed is always a secondary actor. As a result, we do not list it all the time in the activity specification.

All activities have a *level**, which denotes the level of detail that the activity specification presents. The level can be: "player level", "subfunction", and "mobile device event". The "player level" concerns a high level activity that describes the flow of actions involving the primary and secondary actors. The subfunction level represents reusable flow specifications to use in "player level" activities. The "mobile device event" level corresponds to activities related to events specific to mobile devices that might interrupt the activity, as "low battery situations" and "incoming calls".

The *inputs* and *outputs* section refer to sensors, actuators and output devices present in the mobile devices that players carry.

The *smart objects* section lists the smart objects that the activity uses, if any.

The *interaction granularity* section summarizes the granularity of all interactions in the activity. A *discrete*

granularity means that all interactions have discrete granularity. *Continuous* granularity means all interactions have continuous granularity. Finally, *mixed granularity* means that there is a mixture of discrete and continuous interactions in the activity.

The *control* section summarizes the type of control that the primary actor has in the activity. *Explicit* control means that all interactions are explicit. *Implicit* control means all interactions are implicit. *Mixed* control means that there is a mixture of implicit and explicit interactions in the activity.

The *main scenario** corresponds to the typical, "happy-path" flow for the activity. This corresponds to the situation where the primary actor successfully achieves the activity goals. The *alternative scenarios** present other ways to achieve the activity goal, or situations where the activity fails. Alternative flows may end the whole activity, or just the flow of actions that it has started.

The *operation parameters* section defines a set of parameters related to sensors and actuators that the game must consider. For example, an activity using location might require a maximum error of 10 meters in the reported location.

The *uncertainty handling policy* section specifies which general strategy (remove, hide, manage, reveal, and exploit in Section 4.4) that the game uses to handle uncertainties, and how the game applies it.

The *miscellaneous* section refers to special requirements, remarks, or behaviors that do not fit elsewhere.

5. Activity Modeling Examples

This section presents two examples of using our modeling approach in a real game. While presenting the examples, we introduce some extensions to traditional use case notation [Cockburn 2000] that we propose to represent activity modeling concepts.

The examples come from *Pervasive Word Search*, which is a single-player pervasive mobile game developed by the first author. The main goal in *Pervasive Word Search* is to find the letters of a word that the game draws. The player must explore the environment surrounding him to find the letters. While exploring the physical world, the player may interact with some game zones – the dark, open, and wireless zones. The "dark zone" is a place with "low" ambient light. An "open zone" corresponds to an outdoor area. A wireless zone corresponds to a place with a certain number of WiFi access points and Bluetooth devices. Interacting with these zones is an important part in this game.

In Figure 2, a player plays *Pervasive Word Search* with the word "REYNOLD". The player points the device to a tennis shoe to capture a "gray" color – thus getting the letters "g", "r", "a", and "y", and eliminating "r" and "y" of the target word. The game identifies a finite set of colors: red, yellow, orange, green, purple, blue, pink, black, white, and gray. The player goes to wireless zones to get letters that do not exist in the basic color names (like "f", considering names in English). The player has a finite time to find all letters. Interacting with some game zones changes the game clock behavior. For example, if a wireless zone has lots of Bluetooth devices, the game clock runs slower. If the player is inside an open zone, the game clock runs faster.

Figure 2: A player captures a "gray" color in *Pervasive Word Search*.

The player has a finite time to find all the letters. The player is able to get letters by capturing colors with the device camera and interacting with the wireless and dark zones. The captured colors have a limited life span. When this time span runs out, the color "dies" – it becomes unavailable and the player loses all associated letters. In this case, the player has to capture the color again. When the player enters a "dark zone", he earns "white" and "gray" automatically, and those colors remain live as long as the player stays in a dark zone.

In *Pervasive Word Search*, there are four activities of level "player level": *Capture color*, *Interact with wireless zone*, *Interact with dark zone*, and *Interact with open zone*. The first activity has *explicit* control, while the last three activities have *implicit* control. While there is an active player session, these three activities happen concurrently. The game has one activity of level "subfunction", which is *End play session*. The next subsections presents these activities: "*Capture color"* and "*Interact with wireless zone"*.

5.1. The "Capture Color" Activity

In this activity, the player wanders around and points the device to an object to capture its color. The game evaluates the color (informing the player about the results through *acks*) and updates the target word. If all letters have been found, the game ends the play session. Figure 3 illustrates the activity specification.

The scenarios should be specified using a series of numbered action steps, as in traditional use cases. In

this example, the main scenario has two main steps. Each step has an associated acknowledgment. We propose the following bold notations to represent acknowledgments:

ack (T): [information] – short for "the game triggers *acknowledgment* about *information* through medium *T*".

ack (obj, T): [information] – short for "the game triggers *acknowledgment* about *information* through *smart object obj*, using medium *T*".

```
Primary Actors            Player
Level                     player level
Inputs                    camera, screen
Outputs                   display
Interaction granularity   discrete
Control                   explicit

Main scenario
  1. Player positions camera to focus the target
     object and captures its color
     • ack (display): captured color

  2. Game identifies the color and uses the color
     name to update the target word state
     • ack (display): current target word state

Alternative scenarios
2a. All letters of target word have been captured:
  1. Perform End play session
     • ack (display): announces player victory

*a. Main game clock runs out:
  1. Perform End play session
     • ack (display): announces player defeat
  2. EOA

*a. A color dies:
  1. Update target word state
     • ack (display): info about lost color
     • ack (display): current target word state

Uncertainty handling policy
  • Hide
     ▫ The colors are represented with a limited
       set of options. Similar "colors" then are
       grouped with the same name (e.g., all
       variations of "red" are considered as
       "red")
```

Figure 3: *Capture color* specification

There are two *input sources* in this activity: the device camera and the device screen. The game uses the device screen to provide *acknowledgments* about the interactions.

This activity has only one *interaction*, which starts at step 1. The *granularity* of this interaction is *discrete*, as there is only one way to capture a color – by pointing the device camera to an object and touching the device screen to instruct the game to capture a color. Consequently, in this activity the player has *explicit control*. The game issues an acknowledgment at step 1 (through the device display) to inform the player that it recognized a color.

Step 2 represents a *game logic action* that affects the game state. As a result, at this step there is an *acknowledgment* to inform the player about the new state.

Acknowledgments may occur concurrently or may occur in a specific sequence. For concurrent acknowledgments, we propose using bullet lists to represent this concept (as in alternative scenario "a color dies" in Figure 3). For acknowledgments that should occur in a specific sequence, we propose using a numbered list to reflect the desired sequence.

The alternative scenarios may replace a specific step, or may replace the entire main scenario. For example, the alternative scenario "*2a. All letters of target word have been captured:*" is an alternative to step 2 in the main scenario. The alternative scenarios usually have a trigger condition that starts the scenario, which is represented by the condition followed by a comma. The alternative scenario 2a happens when all the letters have been found, which represents the end of the playing session. The alternative scenario 2a also demonstrates how to reference another activity ("End play session").

When an alternative scenario may replace any step in the main scenario, we use the notation `*a` to represent this situation. In this activity, there are examples of this situation: when the main game clock runs out, and when a color disappears ("dies"). These two conditions are *game logic events*. The alternative scenario represented by "*main game clock runs out*" illustrates an example of using the **EOA** notation. This notation indicates "end of activity", which means that the whole activity should be terminated at that point. We proposed this notation to make it more clear when activities should be terminated in alternative flows.

Finally, this activity applies the strategy *hide* to handle uncertainties related to using colors and camera. The game hides the imprecisions and ambiguities related to color capturing and representation by using a finite set of colors and grouping "similar" colors into the same group.

5.2. The "Interact With Wireless Zone" Activity

In this activity, the player wanders around in the physical world while the game searches for Bluetooth and WiFi devices in the background (an implicit interaction). When the search for Bluetooth and WiFi devices is over, the game uses the letters of device names (defined here as "wireless letter set") to complete the target word. The activity starts automatically when a play session starts. The activity ends if the target word is completed (the player wins), or if the main clock runs out (the player loses).

This example demonstrates some actions that occur in parallel (player wandering around and wireless device search). To represent this situation, we defined the notation `[parallel: a, b]`, where `a` and `b` specify the parallel flow steps. Figure 4 illustrates the activity

specification, which uses this notation. In the conceptual design stage, we do not distinguish whether the actions specified with this notation will be implemented using either concurrent or parallel programming.

```
Primary Actors            Player
Level                     player level
Inputs                    bluetooth, wifi
Outputs                   display, vibration
Interaction granularity   continuous
Control                   implicit

Main scenario
  [parallel: 1, 2-4]

  1. Player wanders around in the physical
     environment

  2. Game finds Bluetooth and WiFi devices, updates
     current wireless set
     • ack (display): entered wireless zone
     • ack (vibration): entered wireless zone
     • ack (display): current wireless set status
       (lost letters and letters added to the set)

  3. Game updates target word with letters of the
     current wireless set
     • ack (display): target word status

  4. Game adjust time speed to "default"
     • ack (display): time speed status

Alternative scenarios
2a. Number of Bluetooth and WiFi devices is zero:
  1. Notifications
     • ack (display): left wireless zone
     • ack (display): all wireless letters are
       lost

  2. Game updates target word state
     • ack (display): target word status

  3. Game restarts activity

3a. All letters of target word have been captured:
  1. Perform End play session
     • ack (display): announces player victory
  2. EOA

4a. Number of Bluetooth devices is greater or equal
than 6:
  1. Game adjusts time speed to "slow"
     • ack (display): time speed status

*a. Main game clock runs out:
  1. Perform End play session
     • ack (display): announces player defeat
  2. EOA

*a. A color dies:
  1. Update target word state
     • ack (display): info about lost color
     • ack (display): current target word status

Uncertainty handling policy
  • Hide
     ▫ Announce 'enter zone' and 'leave zone'
       events only – do not inform query status
       (started, finished, in progress, identify
       devices found and lost, etc.).
     ▫ Do not provide area maps with devices
```

Figure 4: *Interact with wireless zone* specification

The main scenario corresponds to the situation where the player wanders around (step 1) and finds a wireless zone (step 2), which may affect the target word state (step 3) but does not complete it. Also, the number of devices found is not enough to slow down the game clock (step 4).

As the process of interacting with the wireless zone is unpredictable, this activity has *continuous granularity*.

Step 2 describes an *interaction event* ("Game finds Bluetooth and WiFi devices") with three associated *acknowledgments*. These acknowledgments do not have to follow a specific sequence, so they are indicated as unnumbered lists. If they were to be delivered in a specific order, they would be specified using numbered lists to reflect the sequence.

Steps 3 and 4 correspond to *game logic actions* that affect the game state, and for this reason these steps have associated *acknowledgments*.

The *alternative scenarios* 2a and 4a represent other outcomes related to the search for wireless devices:

- If there are no wireless devices (Bluetooth and WiFi) around the player (2a), the player leaves the wireless zone and this may affect the target word state (2a, 2). As in this case it does not make sense to go back to the main scenario, the activity is restarted (2a, 3);

- If the number of Bluetooth devices around the player is above a certain amount (six, in the example), the game slows down the game clock. We designed this feature to foster the player to move to areas that might benefit him.

The player wins if the letters of the wireless set help him to complete the target word. Alternative scenario 3a represents this situation. When the player wins, the activity ends (hence, the **EOA** notation).

The *game logic events* "*Main game clock runs out*" and "*A color dies*" are the same ones that appeared in the *Capture color* activity. In this game, these specific events may happen in all activities.

The *Uncertainty handling policy* is *hide*. This activity applies this strategy by deliberately presenting information about the wireless zone in imprecise and ambiguous ways – it does not display a zone map and it does not inform the player about lower-level events related to queries about wireless devices. Instead, the game just informs if the player has entered or left the wireless zone. Bluetooth and WiFi queries are slow operations. In our experiments, a Bluetooth query could take up to 10s to complete. It is also possible that some queries miss some devices or return false positives. Hence, this activity does not require using Bluetooth and WiFi data in real-time fashion. This strategy also applies to other game zones in the game (the dark and open zones), controlled through other activities.

6. Conclusions

Game development is a multidisciplinary process because it involves stakeholders with different backgrounds and worldviews. As digital games take form as software, game developers have tried to apply traditional software development process to create games, only to fail in many circumstances because these traditional software development processes do not consider key features that games have. A major reason is that traditional (non-game) software often focus on productivity, while games focus on providing entertainment, which brings several issues into discussion (*e.g.,* how to handle abstract requirements such as "fun", "enjoyment", "immersion", and "flow").

Another key issue is that games, being creative products, have a preproduction phase (where designers and artists work) that does not exist in traditional software. As a result, many game projects fail due to inadequate transition between the preproduction and production stages. The integration of these teams that speak "different languages" is an important (and complex) issue in game development [Callele *et al.* 2005; Callele *et al.* 2011; Alves *et al.* 2007].

When considering pervasive games, this situation becomes more complex as this area is very recent and is no consensus on what pervasive games are (early examples of pervasive games date back from 2001). The available literature on pervasive games often mixes up preproduction issues and production ones and, as far as we know, there are no methodologies or processes for pervasive game development that integrate preproduction and production issues,

Considering the scenario described previously, in this paper we reported our attempt at creating a language to apply in the transition between preproduction and production stages in pervasive mobile game development. This language extends traditional use case templates [Cockburn 2000]. We believe that this text-based tool provides an adequate abstraction level, when considering the involved stakeholders in the conceptual design stage (*e.g.,* game designers, artists, and software engineers)

The activity modeling approach that we presented in this paper considers important aspects related to the nature of pervasive mobile games – interactions with sensors, mobile devices, and handling uncertainties related technologies that support this kind of game.

Uncertainty handling is crucially important to keep the integrity of the game experience, as it may be affected negatively if the role of uncertainties is ignored in a game. In this regard, we consider uncertainty handling in activities from the very start. Applying uncertainty handling policies may lead to non-functional requirements. Non-functional requirements place restrictions on how the desired functionality can be implemented, and not considering non-functional requirements in the beginning of a project may lead to lots of costly changes [Chung and Leite 2009].

To demonstrate the applicability of this language, we presented two activities of real game. We opted to choose one activity with explicit control and another activity with implicit control to demonstrate how to apply the concepts of our language.

With this paper, we started to touch upon a subject that still has a long road ahead. We envisage as future works in this area:

- Create more games with this language to test how it works in different examples and to uncover possible issues;
- Explore consistency rules to help in checking if an activity specification meets the general goals that we outlined at the beginning of Section 4;
- Conduct a feasibility study to understand if creating a visual version of this language would provide value for the development process;
- Evaluate the applicability of this language by requesting feedback of game designers and software engineers.

Acknowledgments

The authors thank CNPq and FINEP for the financial support to this work.

References

ALVES, C., RAMALHO, G. AND DAMASCENO, A., 2007. Challenges in Requirements Engineering for Mobile Games Development: The Meantime Case Study. In *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*. Requirements Engineering Conference, 2007. RE '07. 15th IEEE International. pp. 275 –280.

BARTNECK, C., HU, J., SALEM, B., CRISTESCU, R. AND RAUTERBERG, M., 2008. Applying Virtual and Augmented Reality in Cultural Computing. *IJVR*, 7[2], pp.11–18.

BELL, M. ET AL., 2006. Interweaving mobile games with everyday life. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp.417–426.

BENFORD, S., CRABTREE, A., FLINTHAM, M., DROZD, A., ANASTASI, R., PAXTON, M., TANDAVANITJ, N., ADAMS, M. AND ROW-FARR, J., 2006. Can you see me now? *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13, pp.100–133.

BETHKE, E., 2003. *Game Development and Production*, Plano, Tex: Wordware Publishing, Inc.

Callele, D., Neufeld, E. and Schneider, K., 2011. A report on select research opportunities in requirements engineering for videogame development. In *2011 Fourth International Workshop on Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games (MERE)*. 2011 Fourth International Workshop on Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games (MERE). pp. 26 –33.

Callele, D., Neufeld, E. and Schneider, K., 2008. Emotional Requirements. *IEEE Software*, 25[1], pp.43 –45.

Callele, D., Neufeld, E. and Schneider, K., 2005. Requirements engineering and the creative process in the video game industry. In *13th IEEE International Conference on Requirements Engineering, 2005. Proceedings*. 13th IEEE International Conference on Requirements Engineering, 2005. Proceedings. pp. 240 – 250.

Chatzigiannakis, I., Mylonas, G., Kokkinos, P., Akribopoulos, O., Logaras, M. and Mavrommati, I., 2011. Implementing multiplayer pervasive installations based on mobile sensing devices: Field experience and user evaluation from a public showcase. *Journal of Systems and Software*, 84[11], pp.1989–2004.

Chung, L. and Leite, J.C.S. do P., 2009. On Non-Functional Requirements in Software Engineering. In A. T. Borgida, V. K. Chaudhri, P. Giorgini, & E. S. Yu, eds. *Conceptual Modeling: Foundations and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 363–379. Available at: http://www.springerlink.com/content/l8p285p273461j60/ [Accessed June 25, 2011].

Cockburn, A., 2000. *Writing Effective Use Cases* 1st ed., Addison-Wesley Professional.

Dey, A.K., 2001. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5[1], pp.4–7.

FinN, 2014. Magnetize Words. Available at: http://funinnumbers.eu/index.php?option=com_content&view=article&id=15 [Accessed July 10, 2014].

Furtado, A.W.B., Santos, A.L.M. and Ramalho, G.L., 2010. Streamlining Domain Analysis for Digital Games Product Lines. In J. Bosch & J. Lee, eds. *Software Product Lines: Going Beyond*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 316–330. Available at: http://link.springer.com/chapter/10.1007/978-3-642-15579-6_22 [Accessed February 16, 2013].

Kanode, C.M. and Haddad, H.M., 2009. Software Engineering Challenges in Game Development. In *Sixth International Conference on Information Technology: New Generations, 2009. ITNG '09*. Sixth International Conference on Information Technology: New Generations, 2009. ITNG '09. pp. 260 –265.

Mottola, L., Murphy, A.L. and Picco, G.P., 2006. Pervasive games in a mote-enabled virtual world using tuple space middleware. In *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. NetGames '06. New York, NY, USA: ACM. Available at: http://doi.acm.org/10.1145/1230040.1230098 [Accessed September 23, 2013].

Petrillo, F., Pimenta, M., Trindade, F. and Dietrich, C., 2009. What Went Wrong? A Survey of Problems in Game Development. *ACM Computers in Entertainment*, 7[1], pp.13:1–13:22.

Rogers, Y. and Muller, H., 2006. A framework for designing sensor-based interactions to promote exploration and reflection in play. *International Journal of Human-Computer Studies*, 64[1], pp.1–14.

Schmidt, A., 2000. Implicit human computer interaction through context. *Personal Technologies*, 4[2-3], pp.191–199.

Siang, A.C. and Rao, G.S.V.R.K., 2004. Designing Interactivity in Computer Games: a UML Approach. *International Journal of Intelligent Games & Simulation*, 3[2]. Available at: http://www.scit.wlv.ac.uk/OJS_IJIGS/index.php/IJIGS/article/view/92 [Accessed June 22, 2014].

Tang, S. and Hanneghan, M., 2008. Towards a Domain Specific Modelling Language for Serious Game Design. In *6th International Game Design and Technology Workshop, Liverpool, UK*.

Taylor, M.J., Gresty, D. and Baskett, M., 2006. Computer Game-flow Design. *Comput. Entertain.*, 4[1]. Available at: http://doi.acm.org/10.1145/1111293.1111300 [Accessed June 21, 2014].

Tuulos, V., Scheible, J. and Nyholm, H., 2007. Combining Web, Mobile Phones and Public Displays in Large-Scale: Manhattan Story Mashup. In A. LaMarca, M. Langheinrich, & K. Truong, eds. *Pervasive Computing*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 37–54. Available at: http://dx.doi.org/10.1007/978-3-540-72037-9_3.

Valente, L. and Feijó, B., 2013. *A survey on pervasive mobile games*, Rio de Janeiro: Departamento de Informática, PUC-Rio.

Valente, L., Feijó, B. and Leite, J.C.S.P. , 2013. Features and checklists to assist in pervasive mobile game development. In *Proceedings of SBGames 2013*. 2013 Brazilian Symposium on Games and Digital Entertainment (SBGAMES). São Paulo: Sociedade Brasileira de Computação, pp. 90–99.