DEVELOPING AN ARCHITECTURE OF GAMES FOR A BLUETOOTH-BASED UBIQUITOUS ENVIRONMENT

José Miguel Rubio, Francisco Reyes and Jorge Inostroza
Pontificia Universidad Católica de Valparaíso, Av. Brasil 2241, Valparaíso, Chile

Keywords: Mobile technology, PDA, Smartphones, Cell phones, Videogame, Communication, Architecture, Ubiquitous environment, Bluetooth.

Abstract: Mobile technology is in an increasingly competitive market, and mobile devices, of various ranges and technologies such as PDAs, smartphones, cell phones, among others are their representatives, offering a wealth of services and resources to use.
As it grows, the development of these devices and their processing and storage capabilities also increases the production of larger applications that exploit these capabilities. This is where the mobile gaming market appears. However due these devices are in a variety of ranges, operating systems and implementations, it creates a problem in game development area because of the lack of uniformity and consistency in the communication between devices.
This article proposes an architecture model game under a ubiquitous environment for mobile devices, primarily focused on the communication of these devices using the Bluetooth protocol with the aim of standardizing applications for different platforms and devices providing support to developers of video games.

1 INTRODUCTION

Mobile technology is advancing very fast and with it the market for mobile devices. A big number of applications and operating systems for these devices have been developed over this last period, more and more powerful applications that use these resources have evolved in parallel excels the video game market, which in the beginning marked a fashion with game consoles, computers and laptops, but at this period has also migrated to mobile devices exploiting their resources, regardless of whether they are classified as limited. Mobile games have evolved and equally so have the requirements of demanding users in this field, which is why players and are unsatisfied when they interacting with their devices, now they are looking to interact with other players and measured with them in playing scenarios through a connectivity medium. This presents a great challenge for game developers because of the huge range of devices on the market. Consequently the development of video games for multiple mobile platforms has been a constant dilemma.

Because of the problems mentioned in this article defines an architectural model of video games that will solve this dilemma, integrating the essential requirements of game development, and standardize the components and technologies for communication between mobile devices that are developed for platform multiplayer.

2 UBIQUITOUS ARCHITECTURE OF GAMES

This section aims to introduce the definition of a gaming architecture under a ubiquitous environment (Weiser 2001), where mobile devices to interact in games multiplayer platform game. Before defining an architecture as such is necessary to define its functionality and features that this architecture must meet. The following describes the main features and technical specifications set by this architectural model.

2.1 Communication

Communication is the focus of this article and turn the main body of the proposed architecture model.

Miguel Rubio J., Reyes F. and Inostroza J.
DEVELOPING AN ARCHITECTURE OF GAMES FOR A BLUETOOTH-BASED UBIQUITOUS ENVIRONMENT.
DOI: 10.5220/0002808801910194
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

This feature is the configuration and deployment of technologies of communication between different mobile devices to interact in a ring game. Arrange and specify the protocols by which devices should connect and share common services.

2.2 Processing

Another feature is that the architecture will provide a module dedicated to processing information wich is transmitted by the devices.
Represents the core of the architecture and is responsible for the overall configuration of each one of the components that interact in the architecture.
Its goal is to generate direct communication with the videogame in development code and the modules of the architecture. That is the request shall be sent from the game parameters to the modules of the architecture while sending the response to the logic of the game.
Within its submodules processing module will incorporate features that will be present in each of the modules and be responsible for transferring data between modules and the videogame code.

2.3 Persistence

Another feature of the architecture is the persistent storage of the information for the items of the videogame. In this case the persistence can be established in two ways: internally or externally. Internal if the information will remain in the internal memory or ROM device, or external if the user plans to store the information on an external server to the device and can connect to a database and hosted on a computer or use web services to store information on the games.
The information will this functionality is directly related to the logic of the games, ie record of scores, statistics time items, the possibility, animations, settings, characters, audio, video, etc.

2.4 Logic

An important feature also is that which relates to the encoding logic games, this means the management and control of major characteristics that have the games and in turn differ from each other.
An important point in this module is the creation of different Artifical Inteligence(AI) techniques that provide direct support to game developers and the grouping of modules focused on the different categories that have the games. (Games of strategy, shooting, role, etc).

2.5 Security

Finally, the architecture will have a functionality responsible for the security of the transfer of information for the items in game, very similar as in most applications security vulnerabilities are always present, is why this module should responsible for solving these problems through security algorithms for different types of games that want to develop.
The module will include features for creating session for the items in the game, asking the user ID and password as previously registered when creating the game, this data should have encryption systems acceptable to establish safe playing games.

These features listed in most cases will depend on each other, which means it will generate some dependency between modules, which is why development should be highly modular.
After defining the main problems in game development, and exposed the features that can fill them, this architecture is defined as a set of packages containers for code libraries, grouped by features with the aim of providing support in developing uniform videogames.
The figure below sets out the package module architecture, where processing package figure as the central entity responsible for communicating with other modules and receive requests from the game code to return them to these modules.

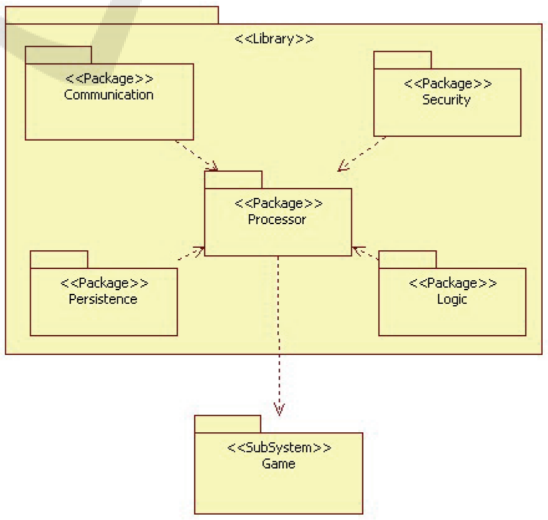

Figure 1: Architecture Package Diagram.

3 STATIC VIEW

One goal of this architecture is to give the possibility to extend functionality of existing components and extend its capabilities after the creation of new ones, this is why modules are designed with specific functionalities and black boxes to make this happen.
Each module will have a class called **Manager** to process the information and transmit it to the processing module which will also feature a central manager, so this way the processor will play a rol enabling communication between modules .
For the creation of new components within the architecture of each module will be called **Driver** classes that extend from the entities manager, and will keep the intention of implementing specific functions associated with each module.
The drivers should have implemented the extended methods to work properly, because the drivers are loaded by the processor and used to make calls to abstract methods of the manager.

3.1 Bluetooth Driver

After extending CommunicationManager, it implements the functionality of the bluetooth technology through a class called **BluetoothDriver**. This implementation addresses all the methods offered by the abstract class **CommunicationManager**.
BluetoothDriver functions are based on client-server architecture. Bluetooth protocols implemented for data transfer are **RFCOMM** and **L2CAP** (Brieva 2004). To have better control over the game session, was created interface **GetClientListener**, the class that implements this interface will be who controls events arrivals of new players to games, and will achieve a higher order process when new players to the meeting.

4 DINAMIC VIEW

4.1 Coordination in the Searching for Devices and Services

The implementation of the Bluetooth standard works on the concept of **mutual exclusion** when seeking services or devices, providing blocking methods, this in order to provide better control over the work of searching and have the operation bounded, resulting in a margin of control, a result at the end of the blocking method and a clear model to program in a clear way and standard.

To achieve this in BluetoothDriver methods: searchDevices() and searchServices(), their implementation uses a ***synchronized*** segment together with an object called a ***lock*** (of type *Object),* which at the time of entering into the search is set to blocked state, and thus do not block the main program. When the search finishes releasing the object *(notify)* allowing to follow the normal course and return to the main program to search results. This represents a process similar to Facade (Gamma et al. 1995) of the operation on Java native Bluetooth, wrapping to provide more reliability at the time of use.

5 USED TECHNOLOGIES

5.1 J2ME

The proposed architecture design using the **Java 2 Micro Edition** (Li et al. 2005), language for the development of the mentioned modules. This technology comes with certain limitations because the devices have limitations in its capabilities of processing and persistence. J2ME virtual machine also called **KVM** (Kilobyte Virtual Machine), works with a limited 32-bit . There are also two important points to consider at the moment of developing in J2ME: the **profiles** and **configutations** (Sun 2005).

6 CONCLUSIONS

This article has presented an architectural model for games focused on ubiquitous mobile devices. Been mentioned the main problems associated with the development of videogames for the wide range of mobile devices on the market.
It also raised the most suitable communication technology for this scenario, where Bluetooth has been chosen to establish connectivity between the devices during play sessions, for reasons of cost and portability in devices.
After developed both implementations of communication protocols: L2CAP and RFCOMM, it is possible to make comparisons related with performance, where clearly the speed of

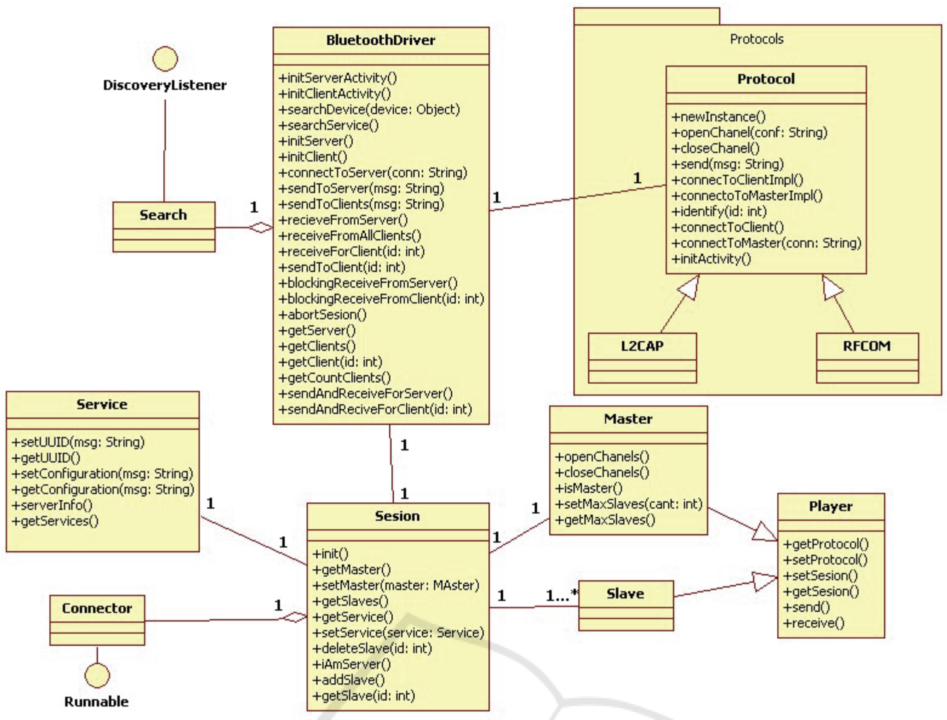

Figure 2: Bluetooth Driver.

transmission in L2CAP protocol is higher because the data is sent at byte level.

As future work is planned to implement functionality in Logic modules, Persistence and Security to thus further enhance the architectural model proposed here.

To address this will continue with a highly modular programming to allow extensibility of the architecture, building powerful and homogeneous features, and thus allow the development of multiplayer games on several platforms regardless of the range of the device for which develops.

REFERENCES

Brieba, A.G., 2004. *JSR-82: Bluetooth from Java*.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Professional Computing Series.

Li, S., Knudsen, J., 2005. *Beginning J2ME: From Novice to Professional*, Third Edition, Apress Editorial.

Sun, 2005. *Mobile Information Device Profile (MIDP)*, JSR 37, JSR 118 Overview.

Weiser, M., 1991. *The Computer for the Twenty-First Century*, Scientific American.