Multi-Channel Gaming Platform

João Ribeiro[1]

Instituto Superior Técnico, Av. Rovisco Pais 1049-001 Lisboa, Portugal

Abstract Multi-platform content distribution has been highly debated during the last few years. Some models have been proposed in order to support the idea of create once, publish anywhere. This thesis assesses the possibility of extending an existent open-source web-based gaming platform in a way to enable its games to be more ubiquitous to its users.

A reengineering process was made to this platform, making possible multi-channel interaction while still keeping backward compatibility. To keep up with all the possible constrains in visualization, brought by the devices capable of interacting with the platform (PC, PDA, mobile phones…), there were introduced changes in the current web interface and also demonstrated the ability to include new interfaces, working in parallel, using an SMS channel interface as an example.

By the end of this work, it was possible to understand the compromises made during the development of the solution and also assess about the advantages of a platform of this kind.

1 Introduction

Online, web-based games are gaining more and more popularity each day. This thesis assesses the possibility of extending an existent open-source web-based gaming platform, to enable multi-channel interaction.

The platform used is called DimensioneX[1], with some extra features added by a previous project[1]. It is written in Java and uses two HTTP servlets running under any Java Web Container.

2 State of the art

Given the subject of this thesis, not much information was found about similar platforms. This lack of information forced the separation of the research[2] in two components: a study about multi-channel content distribution systems [3][4] and another study about online multiplayer game engines.

The analysis of multi-channel content distribution was important to understand which were the solutions available for content adaptation. The projects analysed proposed methods of adaptation from both the client end [5] and the server end [3][4]. With the wide range of newly mobile devices capable of network interaction, it was also critical to identify the major capabilities of these devices and its limitations [6].

Although the game engine (to adapt during this thesis) had already been chosen, the study of other online multiplayer game engines helped understand the inherent characteristics of a platform of this kind, specially the way client interaction would be[7]. In the end of this phase, it was decided how the

[1] DimensioneX, Multiplayer Game Engine: http://www.dimensionex.net

content to be distributed (a game) would be modelled: the current representation of a client state in a game (his current information plus his available options).

After the conclusion of both studies, it was then defined the architecture used by this multi-channel game engine. The idea was to return, after each client interaction, his updated state in a game. This representation would then be adapted according to the interaction channel used.

3 Analysis

The first phase of this work was to analyse the current structure and workflow of the platform. Being the main goal to try to separate the presentation from the game engine, this analysis consisted mainly in identifying the components of both layers and to understand its functioning.

The presentation layer consisted of the generation of HTML code. Its study focused mainly on the identification of information used from the domain (game engine) objects and also the mechanisms supported to change the interface layout.

The game engine layer aggregated mostly all the internal features of the platform, such as: the management of games (and clusters), the activation of the periodic event (onTick) on each game, persistence support in each game and multi-user interaction (concurrency).

By the end of this phase it became possible to specify the requirements needed to the re-design of DimensioneX. The development of the new platform consisted mostly in enabling multi-channel interaction, however some internal problems had also to be resolved.

3.1 Requirements

These were the specified requirements to achieve in the end of development, separated by target:

Game Engine

1. Multi-Channel Interaction
   a. Game Engine Encapsulation
      i. Definition of the services available by the game engine (commands, arguments types and return types).
      ii. Creation of the domain objects views used by the previous definitions.
      iii. High and Low quality sources to the IMAGE object (to be used according to the channel bandwidth).
   b. Maintaining maximum compatibility with current platform
      i. Creation of a game management authority.
      ii. Persistent support.
      iii. Concurrency support.

Presentation

1. Generic Web Interface
   a. Selective information display (according to the level of information supported by the client device).
   b. Easy layout configuration.
   c. Keeping same functionalities of the current platform interface.

2. SMS Interface
    a. User management functions (register, login, logout, …).
    b. Player's state information.
    c. Game commands invocation.

4 New Architecture

The new DimensioneX architecture was based on the design pattern Model-View-Controller. The game engine would behave as the model, being only accessed through a set of services defined in a service layer (acting as the controller). Several interfaces (the views) would then take care of the presentation of the game engine to its clients by making the mapping between client interactions and service invocations.

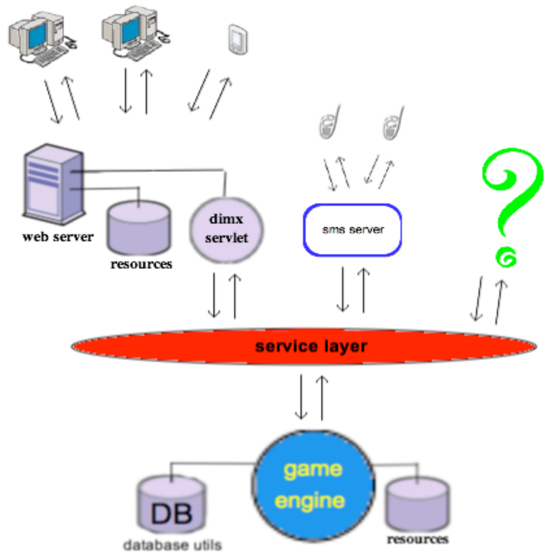

Figure 1 – New DimensioneX architecture.

This new architecture makes the game engine completely independent of the game interface and at the same time enables the possibility to receive multiple service invocations from an unlimited number of sources.

The development of this project was then divided in three sections: the service layer, the game engine and the interfaces.

4.1 The Service Layer

The service layer was implemented using JAX-WS[2]. This way, the communication between interfaces and game engine would be made using only web standards (HTTP, SOAP and WSDL)[3]. In this development phase, the views of the game engine objects were defined along with all the available services specification (name, arguments types and return types).

4.2 The Game Engine

The game engine development focused mainly in the organization of the code, domain objects were

[2] Java Api for XML Web Services, https://jax-ws.dev.java.net
[3] Standards defined by World Wide Web Consortium, http://www.w3c.org

kept practically the same (except for the IMAGE object alteration).
A service hierarchy, containing all the operations supported by the game engine, was created. This not only made it easier to include new functionalities, but also helped maintaining access controls and concurrency.
A game manager was also created. Its responsibility was to serve as the only gateway to the platform available games: access a game; add/remove a game; move objects between games in the same cluster; manage each game periodic event (onTick).
Some internal problems detected during the platform analysis were also resolved. These were scattered through several areas, such as: persistence, security, concurrency and performance.

5 Interfaces

After the redesign and encapsulation of the game engine (also enabling multi-channel interaction), it was now time to start the development of the platform interfaces. This work consisted in the adaptation of the actual Web Interface and afterwards in defining the creation process of new interfaces (with a given example).

5.1 Web Interface

Initially, it was made an effort to adapt the current Web Interface to support the new platform design. However, given the complexity of adapting the interface to the new design, and also because of the lack of compatibility to mobile devices, this approach was replaced by the creation of a new Web interface.
This new interface would have the same capabilities of the former one but with some new features: the support for a wider number of devices (PCs, PDAs, browser capable mobile phones…) and the support for extensive layout configuration.
To make this new web interface available for a wider number of devices, several features were also included to it. The main objective was to display the state information to a client depending of its screen capabilities.

Added features

- Small screen detection: through the use of some Javascript code, a system was created that would show high quality or low quality images depending of the client's device screen.
- Fixed sized boxes with scroll: Used to simulate the former messages view.
- Minimize/Expand buttons: These buttons were included in each interface listing (like people, items, inventory and status) as a way to hide unnecessary information to the client.
- Command grouping by state: The display of available commands now depends of the state of the client. For example, if the client currently doesn't have any object selected, then no commands that would receive one or two arguments are displayed.

As it was given more importance to the quality of the markup code generated, XHTML was adopted over HTML to keep a more strict definition. This code, formerly generated through printings to the servlet response buffer, was now generated using templates[4] of the possible return documents. This enabled the possibility for any web designer (with none java knowledge) to re-organize all presentation.

5.2 New Interfaces

As stated before, it is now possible to include multiple interfaces interacting with one only game engine. It is also worthy of note that the inclusion and removal of interfaces is totally independent of the game engine execution.
The building process of new interfaces can be describe in 4 steps:

New Interface Creation Process

1. Choose interaction channel with client.
2. Choose game engine services to be used.
3. Define the level of detail of the information, returned by each chosen service, to display.
4. Code the mapping between client interactions and the invocation of the services.

During this process, only presentation issues are put to the interface creator. Given the capabilities of the channel used, it is the creator responsibility to choose which services are used and how the information is presented to the client.

Example: SMS Interface

To give an example of multi-channel interaction, a cell phone simulator was created to simulate the use of SMS technology to interact with the game platform.
Given its limitations, the services that would be used were only the ones related with game accounts (like register, login / logout user) and user status in a game. Additionally it would also be possible to invoke game commands, but with very limited feedback.
The information returned in each of these services to the client is very reduced and obviously only text. The commands available at each instant, for example, would have to been already known by the client.

6 Results

The development of the game engine was the more time consuming phase during this project. Although the implementation of the technology (JAX-WS) was almost straight forward, several attempts had to be made to get the game engine running properly. This happened because of the lack of information about the internal functioning of the DimensioneX, which forced me to reverse engineer the extensive and unstructured code of the former application.
During the development of the web interface, some effort was put to always validate the generated markup code. This way, all incorrect browser presentation of the interface could be then blamed on its implementation and never on the generated code, making it much more clean and portable.
The use of a template structure also added the possibility for any user to re-organize the layout of all

[4] Freemarker, http://freemarker.sourceforge.net

the interface without any Java knowledge (just some Freemarker concepts and XHTML+CSS). It should be also noted that the new interface dropped HTML frame support mainly because of the excessive number of requests generated by this approach being used a single frame document response in each client interaction.

The new interface (SMS) served as a good proof of concept of the new capabilities this platform offers. It correctly showed how we can extend the platform to new channels and at the same time making aware of the need for content adaptation.

7 Further Work

At this point there should be two lines of development in this platform.

The first way should be to develop and Integrated Development Environment (IDE) to support the creation of games. At this moment, the only support that can be achieved is syntax highlighting, which clearly is insufficient. The use of a graphical application to aid the creation of the game model would greatly reduce the development time of a game, further examples could be the detection of syntax errors (during coding of events) or even direct integration with the Web Container to facilitate the deploy and/or the debug of games.

The second way would require further studies about the creation of multi-channel games. It could be interesting to investigate new modes of interaction with a game and also new kind of games that take advantage of this feature. (One example could be [8]).

8 Conclusions

The development of this thesis showed the problems that may arise in the creation of a multi-channel application. Not only there can be connection issues (related to the way components communicate) but also content adaptation is needed. I believe that the use of standards (for both communication and representation) is always the best option when there is the need of acceptance by the industry. This way, a much wider range of technologies could interact with the created application, thus enriching it.

Although this game platform already has an extensive user manual[9], its internal operation is still not well documented. The new DimensioneX architecture incorporates a better-structured and easily maintained platform becoming now possible to easily extend the platform in both levels (game engine or interface), giving a more pleasant invitation to contributors, as this is an open-source application.

9 References

[1] G. Carreiro e P. Marques: Trabalho Final de Curso - Jogo Distribuído e Multicanal de Gestão de Equipas de Futebol. Instituto Superior Técnico (October 2006).

[2] João Ribeiro: Introdução á Investigação - Plataforma Para Jogos Multi-Canal. Instituto Superior Técnico (January 2007).

[3] M. Adorni, F. Arcelli, L. Baresi, C. Bantini, A. Bianchi, D. Bianchini, A. Caforio, C. Cappiello, V. De Antonellis, C. Franza, G. Giunta, A. Limonta, P. Losi, A. Maurino, M. Melideo, S. Modafferi, C. Pandolfo, B. Pernici, P. Plebani, C. Raibulet, F. Tisato, E. Valle, A. Zilli: MAIS Reference Model. MAIS Project Report 1.3.2, (November 2003).

[4] S. Ceri, F. Daniel, M. Matera: Extending WebML for Modeling Multi-Channel Context-Aware Web Applications. IEEE International Conference on Web Services (2004).

[5] S. Ceri, P. Dolog, M. Matera, W. Nejdl: Model-Driven Design of Web Applications with Client-Side Adaptation. In Proc. Of the 4th International Conference on Web Engineering (2004).

[6] E. Bertini, M. Billi, L. Burzagli, T. Catarci, F. Gabbanini, P. Graziani, S. Kimani, E. Palchetti, G. Santucci: Usability and Accessibility in Mobile Computing Computing. MAIS Project Report 7.3.3 (May 2004).

[7] M. Brambilla, S. Ceri, M. Passamani, A. Riccion: Managing Asynchronous Web Services Interactions. IEEE International Conference on Web Services (2004).

[8] S. Boll, J. Krosche, C. Wegener: Paper Chase Revisited - a Real World Game Meets Hypermedia, Proc. Of the 14th ACM conference on Hypertext and hypermedia (2003).

[9] C. Leoni: DimensioneX version 6.3.3 Developer's Reference (August 2007) http://www.dimensionex.net/en/docs/default.htm .