A Service-Oriented Platform for iTV applications deployment

Didier DONSEZ, Stéphane CHOMAT, Kiev GAMA, Walter RUDAMETKIN, Lionel TOUSEAU
Université Grenoble 1, BP 53, F38041 Grenoble Cedex 9, France
{Firstname.Lastname}@imag.fr

***Abstract*—Interactive television is a new field for applications developers. However, the applications deployment has to take into account broadcast network constraints. This demonstration shows the design of service-oriented platform for the deployment of modular and dynamic iTV applications. Our proposition named OSGiTV is validated by a prototype of iTV platform using the OSGi platform and DVB MHP standards.**

Keywords-component; iTV, Service Orientation, Deployment OSGi

I. INTRODUCTION

Interactive television (iTV) is a new and promising field for the application developers. As TV sets are ubiquitous and are used to reach a large number of consumers, iTV sets will probably be one of the main entry points for online services. The new generation iTV digital terminals are now able to visualize xHTML documents and run flash presentations or Java applications. Applications and documents are pushed by TV operators to millions of terminals over broadcast networks (satellite, cable, terrestrial, 3G).

Digital Terrestrial Television (DTTV) offers many channels and free services which change the iTV economical model mainly based on paid subscription. One of the main effects is the introduction to the market of a large variety of terminals (ranging from lower-end less expensive terminals to the video game systems) purchased by the user which "will manage" them. This new diversity introduces new challenges to operators and developers of interactive television applications which must take into account the large variety of terminals available with heterogeneous hardware capabilities and a wide range of software configurations. However, current interactive television middleware requires that an operator deploys homogeneous terminals among its subscribers. Those middlewares are generally not aware about the evolution of deployed applications. Moreover, iTV applications design remains monolithic, which does not facilitate the portability of applications among several terminal runtimes and hardwares.

We think that the iTV applications as well as iTV middlewares, could be more modular in order to enable incremental deployment and update without needing application reboot. In this demonstration, we propose a platform that relies on the top of OSGi for a very large scale deployment and the automatic execution of iTV middleware and applications components while respecting the main standards of this field (MPEG2-TS, DVD-MHP, JavaTV). This demonstration shows our iTV platform including an OSGi-based terminal, an operator server broadcasting software installations and updates, applications data and metadata and smartcards storing user profiles and triggering applications launching on the terminal.

The rest of this article is organized as follows: section 2 presents our service-oriented architecture (SOA) for iTV applications and middleware. Section 3 describes the demonstration and section 4 briefly describes the platform and its main components.

II. SOA FOR ITV APPLICATIONS AND MIDDLEWARE

Current interactive television middleware only allows the development of applications with monolithic structure set at development time. The application is a self-contained graph of classes and resources provisioned in a single artifact (i.e. a Jar file). Application deployment consists of putting the artifact in a carousel-based file system and adding an entry in the Application Information Table (AIT) as defined in DVB-MHP.

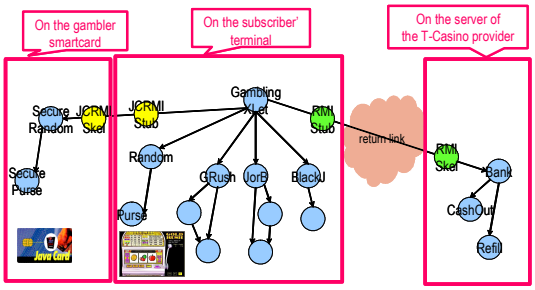

Fig 1 : Example of a SOCA iTV application

We propose to develop modular iTV application according the OSGi programmation paradigm. So, the application is designed as a dynamic graph of OSGi services implemented by classes and resources provisioning in one or more OSGi bundles. Services can be added, updated or removed from the application structure at runtime as long as the consistency rules of the application are respected. Moreover, services can be developed using component models such as ServiceBinder, Declarative Services or iPOJO in order to harden the application code. The figure 1 illustrates the dynamic evolution of our demonstration application architecture.

III. THE DEMONSTRATION APPLICATION

This iTV application allows the user to gamble money at a casino (slot machine, poker, black jack...) using his TV set. The T-Casino service provider gives to gamblers smartcards containing a secure betting engine. The card refill for bettting

(Refill interface) and the cash out (CashOut interface) can be done occasionally with the T-Casino server if the return channel is available. The remote method calls use a RMI stub. According to the same principle, the methods calls of the T-Casino card use a JavaCard-RMI stub. A pseudo betting engine allows the use of this service in demonstration mode. In our dynamic model, the application is designed as a service graph. The T-Casino application is consistent since the root service TCasinoXlet can use at least one betting engine service (Random interface) and at least one game engine service (GameEngine interface). While running, the T-Casino card insertion causes the deployment and the activation of the card agent (for example a JavaCard-RMI stub) which publishs a new service: the application has to take into account from now this new service to perform bets with the card.

The iTV middleware components are traditionally set until the next firmware update which compel to do a terminal reboot. We believe that the iTV middleware can take advantage for the OSGi dynamic model to dynamically deploy its components (libraries of the running environment, peripheral driver) in the same way as the applications components.

IV. The OSGiTV Runtime Platform

The runtime platform, named OSGiTV, is decomposed in 3 subplatforms: the operator broadcasting server, the iTV terminal and the subscriber smartcard. The next subsections describes them as well as the data pushed by the operators.

A. *Information Tables*

OSGiTV uses DVB-MHP tables for managing applications, bundles and drivers. The applications are described by the standard AIT table. Each entry contains an application name and an action to perform at the reception time. These actions can be the addition (STORE), the removal (UNSTORE) or automatic activation of the application. The standard information is completed by two proprietary tables. The Bundle Information Table (BIT) lists information describing the bundles deployed by the operator. Each entry contains a list of imported and exported packages and the list of the required and provided services. The Driver Information Table (DIT) lists information relating to peripherals drivers. Each entry contains information required by the Device Access Manager (DAM) according to the OSGi specifications. This table is used by the terminal driver selector. All that information is extracted from the bundles manifest and for the component metadata (ServiceBinder) by an utility tool running on the server.

B. *iTV operator platform*

The role of iTV operator subplatform is essential for managing the broadcast of the tables, applications and middleware artifacts (ie OSGi bundles). They are broadcasted using a Carousel-based file system (CarouselBroadaster in fig. 2) that regulary send artifacts and table chunks in multicast (IP class D address) datagrams.

C. *iTV terminal platform*

The terminal platform runs over a standard OSGi framework with preinstalled bundles. The figure 2 depicts the main technical services. The carousel-based file system receiver (CarouselReceiver) receives and brings together the broadcasted files chunks. These files are then stored in the terminal cache which manages the local file system in flash memory or on an extra hard drive.

On reception, the information tables are notified by the TableLocators (one by table type) which generate an event object for each entry of a table. These objects then go to an EventDispatcher which notifies the subscribers of this event type. The applications manager, XletManager, subscribes to AI type of events and updates its information base. The XletManager starts an application when it receives an AI object with an autostart action. For this, it delegates the application install to the ServiceOnDemand installation service by seeking a javax.tv.xlet.Xlet service having the XletName property equals to the identifier in the AI. The installation manager, called ServiceOnDemand, allows to install and to start recursively the bundle containing the Xlet and its package dependencies. For this, it uses its information base which is fed by the BI events (Bundle Information) notified by the EventDispatcher. The driver manager respects the OSGi specification “Device Access Manager” for the installation of the terminal’s device drivers. However according to the broadcast network constraints, our platform implements a DriverLocator using of the information contained in the received BITs.

The application startup can be initiated by the operator, by the user using a program guide (i.e. the EPGXlet) or by the insertion of a smart card. In those 3 cases, the XletManager takes care from startup to the reception of an AI object marked as AUTOSTART.

D. *iTV subscriber smartcard platform*

The subscriber smartcard is a JavaCard card applet storing the subscriber’s profile and the AIT for his prefered iTV applications. The JavaCard can also host card applets securing sensible data and processing for the iTV application as in the case of the T-Casino’ SecureGamblingCardlet.

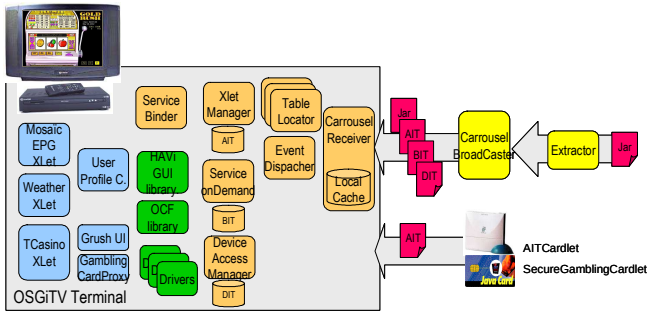

Fig 2 : Architecture of the OSGiTV deployment subplatforms.