Interactive Television in Brazil: System Software and the Digital Divide

Luiz Fernando Gomes Soares
TeleMídia Lab - Catholic Univ. of Rio de Janeiro
lfgs@inf.puc-rio.br

Guido Lemos de Souza Filho
Federal Univ. of Paraíba
guido@lavid.ufpb.br

Abstract
This paper presents an overview of the new concepts and characteristics brought out by Ginga, the standard middleware of the Brazilian digital TV system.

1. Introduction

Different from common middlewares for terrestrial digital TV, the standard middleware of the Brazilian DTV (ISDTV-T), named Ginga, has its declarative environment based on an XML application language, named NCL (Nested Context Language), which focus on how media objects are structured and related in time and space. As a glue language, NCL does not restrict or prescribe the media-object content types, and it can even include an XHTML-based media object, as defined in other common DTV standards.

Ginga procedural (Ginga-J) and declarative (Ginga-NCL) environments also bring novel facilities, as for example new possibilities for user interactions and response exhibitions, and new support to handle unbound applications.

This paper is organized as follows. Section 2 presents the Ginga architecture, leaving for Sections 3 and 4 an overview of its Ginga-NCL and Ginga-J modules, respectively. Section 5 discusses the bridge between these two modules and Section 6 brings the final remarks.

2. Ginga Architecture

This section briefly introduces the Ginga Architecture.

Ginga applications are classified into two categories depending upon whether the initial application content processed is of a declarative or a procedural nature. These categories of applications are referred to as declarative and procedural applications, respectively. Application environments are similarly classified into two categories depending upon whether they process declarative or procedural applications, and are called Ginga-NCL and Ginga-J, respectively, as shown in Figure 1.

A Ginga application needs not to be purely declarative or procedural. In particular, declarative applications often make use of script content, which is procedural in nature. Furthermore, a declarative application may reference an embedded Java Xlet. Similarly, a procedural application may reference declarative content, such as graphic content, or it may construct and initiate the presentation of a declarative content. Therefore, either type of Ginga application may make use of facilities of both declarative and procedural application environments.

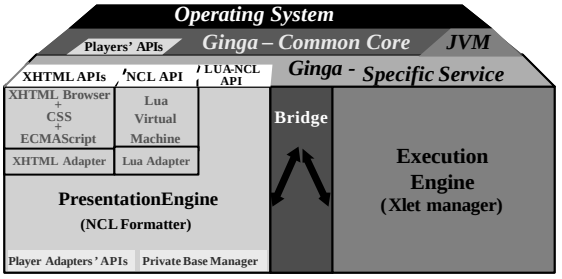

Figure 1 – Ginga architecture

Ginga-NCL is a logical subsystem of the Ginga System that processes NCL [Soares, 2006] documents. Key components of Ginga-NCL are the declarative content decoding engine (NCL formatter) and its Private Base Manager module.

The NCL Formatter is in charge of receiving an NCL document and controlling its presentation, trying to guarantee that the specified relationships among media objects are respected. The formatter deals with NCL documents that are collected inside a data structure known as private base. Ginga associates a private base with a TV channel. NCL documents in a private base can be started, paused, resumed, stopped, and can reference each other. The Private Base Manager is in charge of receiving NCL live editing (see Section 3) commands and maintaining the active NCL documents.

Other important modules of Ginga-NCL are the XHTML-based user agent, which includes a stylesheet (CSS) and ECMAScript interpreter, and the LUA engine, which is responsible for interpreting LUA scripts [Soares, 2006]. Depending on the XHTML implementation, Ginga-NCL can be compatible with other declarative standards.

Ginga-J is a logical subsystem of the Ginga that processes active Java based object content. Thus, it has as a key component the procedural content

execution engine composed by a Java Virtual Machine.

Ginga-J is GEM (Globally Executable MHP) [Souza 2006] complaint. GEM defines *Core* and *Specific* APIs (Application Program Interfaces). *Core* APIs must be supported by all GEM conformant middlewares: DVB MHP, ATSC ACAP and OCAP, ISDB ARIB and ISDTV Ginga. In order to provide support for Brazilian specific requirements and to explore opportunities created by the new convergent digital TV hardware, in terms of processing power and Home Area Network interfaces, Ginga *specific* APIs provide support for multiuser, multidevice and multinetwork interactions. Ginga *specific* APIs also provide support for *unbound* applications, which can be received, saved and, later, accessed and executed.

Ginga Common Core supports both the Ginga declarative application environment (Ginga-NCL) and the Ginga procedural application environment (Ginga-J). It is composed by common content decoders (for the decoding and presentation of common content types such as PNG, JPEG, MPEG and other formats), and procedures to obtain contents transported in MPEG-2 Transport Streams and via the return channel. The DSM-CC is adopted in Ginga for carrying live editing commands in MPEG-2 TS elementary streams. DSM-CC stream events and DSM-CC object carousel protocol are the basis for application handling in Ginga. The Ginga Common Core shall also support the conceptual display graphic model defined by the Brazilian DTV system.

3. Ginga-NCL

Unlike HTML or XHTML, NCL has a stricter separation between content and structure and it provides non-invasive control of presentation linking and layout. As such, NCL does not define any media itself. Instead, it defines the glue that holds media together in multimedia presentations.

NCL document only defines how media objects are structured and related, in time and space. As a glue language, it does not restrict or prescribe the media-object content types. In this sense, we can have image objects, video objects, audio objects, text objects, execution objects (e.g, Xlet, LUA, etc.), etc., as NCL media objects. Which are the media objects supported depends on the media players that are embedded in the NCL formatter. In the Brazilian DTV system, one of these players is the MPEG-4 decoder/player, implemented in hardware in the DTV receptor. In this way, note that the main MPEG-4 video and audio is treated like all other media objects that can be related using NCL.

Another NCL media object required in ISDTV-T is the HTML-based media object. Therefore, NCL does not substitute but embed HTML-based documents (or objects). As with other media objects, what HTML-based language will have support in an NCL formatter is an implementation choice, and, therefore, will depend on which HTML browser will act as a media player embedded in the NCL formatter.

Although an XHTML-based browser must be supported, the use of XHTML elements to define relationships (including XHTML links and ECMAScripts) should be dissuaded when authoring NCL documents. Structure-based authoring should be emphasized.

During the exhibition of media-object contents, several events are generated. Examples of events are the presentation of marked segments of a media-object content, the selection of a marked content segment, etc. Events may generate actions on other media objects, like to start, pause or stop their presentations. Hence, events must be reported by media players to the NCL formatter that, in its turn, can generate actions to be applied to these or other players. Ginga-NCL defines an adapter API to standardize the interface between the Ginga-NCL formatter and each specific player. Third part players, including XHTML-based browsers, usually need an adapter module to realize their integration.

Finally, for live editing, Ginga-NCL has also defined NCL stream events (DSM-CC events) in order to support live generated events in stream media, in particular the main program stream video. These events are a generalization of the same concept found in other standards, like for example the b-events of BML. Again, the use of XHTML elements to define relationships (stream event elements in this case) should be dissuaded in authoring NCL documents, for the same motivation: structure-based authoring should be emphasized.

3.1 NCL and SMIL

Nowadays, declarative languages focusing on synchronization are not common in DTV middlewares.

SMIL, the W3C standard for synchronization, as well as NCL define a set of modules to support temporal synchronization events, including interactivity as a special case. Both languages also provide support for content and presentation alternatives, which are very important to context (user, device, location, etc.) aware adaptations. NCL provides, in addition, support for spatial synchronization; for multiple-device simultaneous

interactions and responses; for variable handling; and also a very powerful script language, named Lua (Soares, 2006) that among its facilities can manipulate the receptor hardware and interact with the Java environment of Ginga. Only recently SMIL has announced its draft for a "set-top box profile", where some of these issues are addressed.

The ISDB ARIB B-24 considered the use of SMIL language, but abandoned the idea, because at that time: "SMIL was quite static representation scheme. It was ready for pre-programmed timing, not real-timing. It was inconvenient for live program", as they stated. Live program editing and synchronization are usually accomplished through DSM-CC stream event handling. Ginga-NCL treats stream events as real-time editing operations on NCL documents. Among other advantages the Ginga-NCL solution preserves the same document specification both in the authoring environment and in the receptor.

More recently, a joint work has begun in W3C, aiming at a middleware implementation that would allow both NCL and SMIL application support.

4. Ginga-J

Ginga get access to streams of video, audio, data, and other media assets through the Ginga Common Core. Ginga can receive input from users via conventional remote controls or any other peripheral having a keyboard, a cell phone, etc. In response to an input, Ginga may present visual information on the television set itself, on the screen of other output devices, or even as an audio output to loudspeakers. A single interaction device may have both input and output capabilities. Usually, an interaction coming from these devices redirects the output answer to themselves. A PDA is an example of such devices. Many devices may interact with Ginga at the same time. In this case, the platform must distinguish the commands sent by and responses sent to each device.

As aforementioned, Ginga-J is GEM complaint. As such, it includes the GEM *Core* APIs and defines other *Specific* ones to fulfil the Brazilian particular requirements. Ginga-J *Core* APIs are: Sun Java TV™ API 1.1 (JSR 927); Java Media Framework Specification (JMF 1.0); Connected Device Configuration (CDC) 1.1 (JSR 218); Foundation Profile (FP) 1.1 (JSR 219) and Personal Basis Profile (PBP) 1.1 (JSR 217); DAVIC API; and HAVi API.

Ginga-J *Specific* API set is composed by: JMF 2.1.1 (JSR 920), which allows working with the TV set to input and output media streams; an extension of the GEM's return channel API, which allows sending asynchronous messages; and an extension of the ISDB-ARIB B.23's Service Information API. Ginga-J Specific APIs set also includes a Multiuser API, a Device Integration API and a Ginga-NCL Bridge API. The first one provides support for multiple simultaneous user interactions with the TV set; the second supports the integration of devices like cell phone and PDAs using typical Home Area Networks technologies (Bluetooth, Wifi, PLC etc.); finally, the third one provides support for controlling the presentation of NCL documents through Java Xlets. A complete specification of Ginga-J APIs can be found in [Souza, 2006].

5. The Bridge Ginga-NCL ? Ginga-J

The two-way bridge between Ginga-NCL and Ginga-J is done:

? in one way, through NCL relationships, defined in NCL <link> elements that refers to <media> elements representing Xlet (application/x-ginga-NCLet type) codes supported by Ginga-J; and through Lua scripts (<media> elements of the application/x-ginga-NCLua type) referencing Ginga-J methods;

? in the reverse way, through Ginga-J functions that can monitor any NCL event and can also command changes in NCL elements and properties, through relationships defined in NCL <link> elements or through NCL live editing commands.

6. Final Remarks

The Brazilian Digital TV System (ISDTV-T) will start its transmissions in the end of 2007. Conceived for terrestrial ("free-to-air") TV, the system took into account all other DTV system reference models without neglecting the topographic, political and social peculiarities of the country and its people. Moreover, ISDTV-T took profit of being conceived when emerging technologies previously unfeasible were available. ISDTV is a variant of ISDB, and Ginga is the standard middleware of this new DTV system.

References

Soares, L.F.G. (2006). *Standard 06 - ISDTV-T Data Codification and Transmission Specifications for Digital Broadcasting, Volume 2 – GINGA-NCL: Environment for the execution of declarative applications*. São Paulo, SP, Brazil. ISDTV-T Forum.

Souza, G.L. (2006). *Standard 06 - ISDTV-T Data Codification and Transmission Specifications for Digital Broadcasting, Volume 4 – GINGA-J: Environment for the execution of procedural applications*. São Paulo, Brazil. ISDTV-T Forum.