A Template-Based MHP Authoring Tool

Hsin-Ta Chiao¹, Kuo-Shu Hsu¹, Yu-Kai Chen¹, Shyan-Ming Yuan²

¹ Digital Video and Optical Communications Technologies Division
Information and Communications Research Laboratories
Industrial Technology Research Institute
Chutung, Hsinchu, Taiwan
{JosephChiao, kshsu, ykchen}@itri.org.tw

² Dept. of Computer Science, National Chiao Tung University, Taiwan
smyuan@cs.nctu.edu.tw

Abstract

In this paper, we propose a template-based MHP (Multimedia Home Platform) authoring tool. The temporal and spatial behavior of an MHP application can be authored and stored in an XML-based instance description file. The MHP authoring tool generates the target MHP Java source codes by utilizing the "real programs generators" for the modules used inside the MHP application being authored. The real program generator for a module is generated by a "meta program generator" according to the XML template description for the module. The Java source codes of an instance of the module are generated by the real program generator according to the XML description for the instance. The proposed design on code generation can simplify the way to extend the MHP authoring tool to support new features, such as DRM or PVR. In addition, the MHP applications generated by the authoring tool are also reconfigurable. They can adapt to both the MHP set-top box resource change and the change on transmission bandwidth of a DVB object carousel.

Keywords: IDTV, generative programming, aspect-oriented programming, XML, Java.

1. Introduction

Multimedia Home Platform (MHP) [1] is a European standard for the middleware on IDTV (interactive digital Television) set-top boxes. Its subset, Globally Executable MHP (GEM) [2] is the common core of the major IDTV middleware standards around the world, such as in Japan, and in US. Hence, in the near future, the MHP-based or GEM-based IDTV set-top boxes may be widely spreading around the world after the waves of digital-analog switch-off.

The core of the MHP 1.0.x standards is a Java-based middleware that provides a set of standard Java APIs. Since Java is easy to use and of good expressive power, MHP is very powerful and flexible for creating various kinds of IDTV applications and services. However, the barrier of Java programming hinders the people without programming skills to create MHP-based IDTV applications. MHP authoring tools are the RAD (Rapid Application Development) tools designed for such people to create MHP-based applications without direct programming by using Java.

XML (Extensible Markup Language) is popular for describing an IDTV program because it can be understand directly by the human. Some XML-based IDTV standards are available, such as the MPEG-4 XMT (Extensible MPEG-4 Textual Format) [3] and the W3C SMIL (Synchronized Multimedia Integration Language) [4]. The IDTV contents authored by using the above-mentioned XML-based standards are easy to be maintained and easy to be exchanged among different IDTV authoring tools.

For similar reason, some MHP authoring tools choose XML as the proprietary representation to describe the behavior of the MHP applications generated by them. In theory, an MHP application may parse an XML description at the run-time on a MHP set-top box for controlling its run-time behavior. However, parsing an IDTV XML description directly on the MHP set-top box may induce the following problems: First, the memory footprint of an MHP set-top box may be widely spreading around the world after the waves of digital-analog switch-off.
low cost, so that they are usually of limited system resources such as memories. Besides, for terrestrial and satellite DTV broadcasters, since the spectrums for broadcasting are the critical resources, the bandwidth for delivering an MHP application is usually small inside a DTV MPEG-2 multiplex. For example, the Digita, which is a terrestrial DTV network operator in Finland, allocates 4 Mbps bandwidth for the object carousels to transmit MHP applications in its 24-Mbps DVB-T multiplex. Each MHP application can only use about 0.35 ~ 1 Mbps among the 4 Mbps bandwidth. Hence, the Cardinal system suggests that the size of an MHP application shall be less than 500 KB [5]. However, a full-featured XML parser (XML schema validation, SAX, DOM) requires 500 KB RAM and ROM [6]. An embedded XML parser [6] with a reduced set of XML functions such as eliminating the DOM and XML validation support can reduce its memory consumption; however, there is still another problem waiting to be solved.

The second problem for parsing an IDTV XML description directly on an MHP set-top box is the performance problem. Parsing an XML document may be slow because textual parsing and schema validation are time-consuming. Good evidence can be found in the case of XML-based protocol: The XML-RPC is an XML-based RPC (Remote Procedure Call) protocol. It is 10 times slower than the hand-written protocol that directly using the java.net packages [7]. If an XML IDTV description is large and complex, its long parsing time will slow down the start-up time of an MHP application. The start-up time of an MHP application is already slow since its transmission is based on an object carousel. It broadcasts the application files in a round-robin fashion, by using a relatively narrow bandwidth.

In summary, if XML is chosen for describing the behavior of an MHP application, program generation is required by the MHP authoring tool to avoid the problem of parsing XML documents on the MHP set-top box. Hence, in this paper, we propose a code-generation-based MHP authoring tool. It employs the notion of “meta program generator, and its code generation processes are constructed by using the XML-based templates. In comparison to the traditional code generation approaches, this design can ease the maintenance and evolution of an MHP authoring tool, such as adding new UI components, adding new features, or modifying an aspect.

The rest of this paper is organized as follows: In Section 2, we describe the difficulty of evolving a code-generation-based MHP authoring tool, and introduce the notion of the meta program generator. In Section 3, several related works are discussed and compared with the proposed template-based MHP authoring tool. In section 4, we present the metadata for describing the behavior of an MHP application. In section 5, the system architecture and the main system operation flows are shown. Finally, section 6 concludes this paper.

2. The meta program generator

Program-generation-based MHP authoring tools are difficult to be maintained due to the following two reasons: First, MHP authoring tools are requested to be able to evolve because MHP itself is an evolving standard. New functions will continue to be added into the MHP specification, such as the MHP-PDR (Personal Digital Recorder) extension [8], and the DVB-CPCM (Content Protection & Copy Management) standard [9] (not finalized yet). Existent MHP authoring tools have to be enhanced to accommodate the new PVR (Personal Video Recorder) and the DRM functions.

The second maintenance difficulty comes with an aspect change. Since the MHP set-top boxes are usually resource-constrained and of different capabilities, an MHP application often has to be fine-tuned for adjusting its resource usage. The reconfiguration happens either before the application deployment, or at the initialization time on an MHP set-top box. To adjust the resource usage of an MHP application, such as section filters, memories, often requires the negotiation among the modules inside an MHP application. In other word, to change the usage of a resources mentioned above involves an aspect change inside the MHP application. An aspect [10] is a metrics or a feature that involves almost the whole parts of a program. For enhancing an MHP application to be reconfigurable to a new aspect, multiple modules inside an MHP authoring tool may have to be modified. Furthermore, the aspect change problem is often out of the scope where OOP (object-oriented programming) can help. The OOP can only solve the problems where the rule of inheritance and polymorphism can follow.

To solve the above mentioned problem, we propose a template-based MHP authoring tool that use a “meta program generator” to create the “real program generators.” A target MHP application (in the form of MHP Java codes) is comprised by the modules generated by the real program generators under the coordination of the authoring tool. For each module to be generated, the meta program generator produces a
real program generator according to the template description of the module. The template description is an XML document that follows the schema definition of the XML template description language (XML TDL). The XML TDL uses the XPATH [11] standard for XML tree navigation (XPATH is an expression language for selecting nodes in an XML tree). In addition, the XML TDL also supports template inheritance inside a template description.

A real program generator generates the Java source code for a module according to the XML instance description of the module. The module’s instance description specifies the parameters for instantiating a module. For example, an instance description of the TextButton actor (which is a GUI module that represent a text button widget) specifies the parameters of the text button widget, such as the string to be present on the button, the placement, the size, and the color of the button, and the related key events. The XML schema definition of an XML instance description for a module is also generated by the meta program generator, according to the template description of the module. Furthermore, for reducing the memory consumption of an MHP application, the code and read-only data are shared among the module instances generated by the same real program generator.

The advantage of the template-based MHP authoring tool is easier to adapt the change for accommodating a new feature. For adding the support of a new aspect, we can usually restrict the modification only on the following two parts of the MHP authoring tool: the meta program generator, and the template description of the real program generator for the root-level program module. Since the root-level program module takes care the parameters for the aspects across the whole program, its template description has to be modified to accommodate the parameters of a new aspect.

3. Related Works

Cardinal Studio Professional 4 [5][12] is an MHP authoring tool. It provides a simplified application structure that is simpler than the scene-shot-layer-actor model of the template-based MHP authoring tool proposed in this paper. A Cardinal Studio application is comprised of several acts, which are different scenes of the application. An act may have one or several layers that consist of some visible or invisible UI components. The UI components follow the standard JavaBean-based architecture. Since the JavaBean is a well-known standard and related supporting tools are mature, it is easy to create a new UI component for the Cardinal Studio. Code generation in the Cardinal Studio is only for generating the glue codes between JavaBean components. The Cardinal Studio also supports templates for editing the presentation of components and acts, but not the templates for code generation. Besides, since the Cardinal Studio does not enforce using XML for describing the state or properties of a JavaBean component, an MHP application does not have to embed an XML parser at run-time. In brief, the Cardinal Studio does not employ the notion of the “meta code generator” proposed in this paper. It uses a code post-optimizer to reduce the memory footprint of an MHP application generated by it.

ICCT (Interactive Content Creating Tool) [13] is an MPEG-4 BIFS authoring tool. The MPGE-4 BIFS is a scene-based representation for interactive multimedia. It is of good expressive power, but too low-level for direct content authoring. This is quite similar to the case of MHP: powerful but difficult for content authoring by the people without programming skills. The ICCT provides a scene designer for a user to create BIFS scenes, or to set the layout and properties of the objects inside the scenes. In addition, it also provides scripts to customize the behavior of scenes and objects. The ICCT’s internal architecture clearly separates the process for scene authoring by the scene designer, and the process for generating of the target BIFS descriptions. The template-based MHP authoring tool follows the similar system architecture that clearly separates the scene authoring and the code generation of the target MHP Java source codes. However, for code generation, the ICCT does not employ the notion of “meta code generator” in its code generation process.

The European IST Project SAVANT [14] defines an XML metadata model for describing the concept of scalable interactive TV contents, which can be delivered to various kinds of terminal devices. The XML metadata model is derived from the XML metadata of TV-Anytime [15]. Two extra tables are introduced – the SegmentInformationTable and the AdditionalContentTypeTable. The European IST Project ENTHRONE [16] also defines an XML metadata model for achieving the vision of “Universal Multimedia Access.” The MPEG-21 multimedia framework is adopted as the envelop of the XML metadata, which encapsulates the description for contents, right protection, network QoS, terminal, users, etc. The metadata of TV-Anytime is used for the purpose of content description inside the MPEG-21 framework. Since XML provides the name space
concept to facilitating the integration of the elements defined by various XML schemas, it is easy to merge various kinds of existent XML metadata to create a new XML metadata format for a new system. The above two related works are good evidences of merging the existing XML metadata standards to fulfill the particular requirement of a multimedia system. However, to extend an XML metadata model is easy, but to modify a real system to reflect the metadata change is difficult. The notions of “meta program generator” proposed is this paper is for reducing the cost of modifying an MHP authoring tool to adapt the change on its XML metadata model.

Figure 1: The application structure of the template-based MHP authoring tool

An MLP-Based (Meta-Level Programming) DSL (Domain-Specific Language) [17] was proposed for the configuration of the GUI of an MHP application. It is based on the concept of DSL product line that contains a series of DSLs sharing the same infrastructure. Only minor differences are between the DSLs. The common infrastructure of the DSLs is a template language that is tailorable to adapt the differences between the DSLs inside a product line. The template language for the MHP GUI DSL is Frag [18], which is an interpreter-based meta-level programming language [19]. The language constructs of Frag can customize the behavior of the GUI of an MHP application through the runtime structural reflection and control flow reflection [20]. However, we consider that the interpreter-based, meta-level programming may introduce quite more run-time overheads so that it may not feasible on an MHP set-top box with limited system resources. The meta program generator approach is a compile-time solution that can avoid the above-mentioned problem.

Figure 2: The UML class diagram for the metadata hierarchies of the template-based MHP authoring tool

4. Metadata

Figure 1 shows the application structure of the template-based MHP authoring tool. A shot is the unit of screenplay; it occupies the whole TV screen and can control the layout of the TV screen. A scene is comprised of one or several shots that are closed-related for a special purpose. An MHP application is a program that should contain at least one scene. Since the MHP standard adopts the HAVi display model [21], the TV output signal of an MHP set-top box is composed by the composition three layers: the background layer, the video layer, and the graphics layer. Hence, a shot may contain several layers and each layer may contain several actors (For concise presentation, Figure 1 omits the notion of layers). An actor is the unit of user interface. It can be the multimedia resources such as audio and video streams, audio and video clips, and images. It can also be the GUI components such as labels, text buttons, and graphic buttons. Figure 1 has two examples of shots (scene 1, shot 1 and scene 2, shot 2) that are composed by various kinds of actors. For an MHP application, the template-based MHP authoring tool generates an XML instance description file to describe the behaviors and structures of the MHP application. The XML instance description file contains the instance descriptions of all module instances inside the MHP application, including the instance descriptions for actors, shots, scenes and the root-level program. The real program generators generate the Java source codes of the modules according to the instance descriptions in the file.
templates of the template inheritance to enable all the child inside an MHP application, we can take advantage of feature in some situations. For example, to extend the inheritance can simplify the difficulty of adding a new Object above five kinds of templates can be specified in the Object of templates have a common parent template – the hierarchy represent the relationships between the hierarchy for the root-level program, scenes, shots, layers and actors. The above five kinds of templates have a common parent template – the Object template. The common properties among the above five kinds of templates can be specified in the Object template. The capability of template inheritance can simplify the difficulty of adding a new feature in some situations. For example, to extend the template-base MHP authoring tool to support DRM inside an MHP application, we can take advantage of the template inheritance to enable all the child templates of the Object template to associate with a RightExpression template. The instance description of the RightExpression template can contains the right expressions for DRM.

The TV-AnyTimeProgramInfo template is for the PVR support. The instance description of the TV-AnyTimeProgramInfo template basically follows the structure of the phase 1 standard for the TV-Anytime XML metadata [15]. The template-based MHP authoring tool automatically generates the instance description of the TV-AnyTimeProgramInfo template when authoring an MHP application. Since the TV-Anytime phase 1 metadata standard can only describe the attributes of audio and video contents, we extend the XML schema definition for describing the program information (tva: ProgramInformationType) to accommodate interactive content such as MHP application.

The Java source codes of an MHP application are generated in a hierarchical fashion. The UI implementation layer first parses the XML instance description file, and identifies the real program generators for the module instances used in the UI. Then, as shown in Figure 4, it invokes the real program generators for the root-level program, scenes, shots, and actors and dispatches the content of the XML description file to the corresponding real-program generator according to the identities of templates. After the code generation for the Java source codes of the MHP application is finished, the UI implementation layer invokes the JDK javac compile to compiler the Java source codes into Java bytecodes. Finally, the Java bytecodes and the imported mono-media resources are combined into the storage format of an MHP application.

5. System Architecture

Figure 3 shows the system architecture of the template-based MHP authoring tool. It can be separated into two layers: the graphic user interface layer (GUI layer) and the UI implementation layer. It also has a template repository that contains the XML schema definitions of the instance descriptions, the XML TDL templates, and the real program generators. The GUI layer and the UI implementation layer can consult the template repository to retrieve the XML documents and the real program generators when necessary.

The Java source codes of an MHP application are generated in a hierarchical fashion. The UI implementation layer first parses the XML instance description file, and identifies the real program generators for the module instances used in the UI. Then, as shown in Figure 4, it invokes the real program generators for the root-level program, scenes, shots, and actors and dispatches the content of the XML description file to the corresponding real-program generator according to the identities of templates. After the code generation for the Java source codes of the MHP application is finished, the UI implementation layer invokes the JDK javac compile to compiler the Java source codes into Java bytecodes. Finally, the Java bytecodes and the imported mono-media resources are combined into the storage format of an MHP application.

6. Conclusion

The program generator for an MHP (Multimedia Home Platform) authoring tool is difficult to maintain due to the following two reasons: First, since MHP set-top boxes are environments with limited system resources, and the transmission bandwidth of an MHP
application is always narrow, an MHP application should be reconfigurable to adapt the changes on the set-top box system resource and the transmission bandwidth. Second, the MHP itself is an evolving standard so that an MHP authoring tool has to be modified to accommodate new functions, such as PVR or DRM supports. In this paper, we proposed a template-based MHP authoring tool that can be considered as an application of aspect-oriented programming on XML-to-Java program generation. This design can help an MHP authoring tool to overcome the two maintenance problems mentioned above. In addition, unlike traditional interpreter-based, meta-level programming systems, the proposed design mainly relies on compile time mechanisms to deal with the aspect-related issues. This design is more suitable for the MHP applications that should be executed on a resource-limited MHP set-top box.

References


