An Eclipse-Based Authoring Tool For Developing Model-Based Adaptive Service Frond-Ends

Safdar Ali, Umair Javed
Human Computer Interaction Laboratory, SAP AG
Bleichstr. 8, 64283 Darmstadt, Germany
{firstname.lastname}@sap.com

ABSTRACT
This paper describes the initial results of our research within the context of Serenoa project, where we have developed the first version of an authoring tool, which has been developed as a plugin for the widely used Eclipse IDE in the industry and academia. This authoring tool supports all the CRUD (Create, Retrieve, Update, and Delete) operations for the adaptation rules and front-end UIs development for the service definitions at Abstract level. The adaptation rules and the front-end UIs are developed using the model-based languages, namely the Advanced Adaptation Logic Description Language (AAL-DL) and the Advanced Service Front-End Description Language (ASFE-DL), respectively, which have been developed in the project.

Author Keywords
Serenoa Project, Authoring Tool, Adaptive Service Frontends, Model-Driven Architectures, Eclipse-Plugin

ACM Classification Keywords
Algorithms, Design, Languages

General Terms
Algorithms, Design, Languages, Theory

INTRODUCTION AND BACKGROUND

Model-based approaches mainly aim at helping the developers to understand user needs and design solutions in an effective way. Nowadays, many industrial and academic initiatives for implementing context-aware adaptation of user interfaces rely on model-based approaches for UI design [1] and Model-Based Integrated Development Environments (MB-IDEs) [2]. The purpose of model-based design is to identify high-level models, which allow designers to specify and analyze interactive software applications from a more semantic oriented level rather than starting immediately to address the implementation level. This allows them to concentrate on more important aspects without being immediately confused by many implementation details and then to have tools which update the implementation in order to be consistent with high-level choices. Thus, by using models which capture semantically meaningful aspects, designers can more easily manage the increasing complexity of interactive applications and analyze them both during their development and when they have to be modified [3].

Serenoa Project [4] is aimed at developing a novel, open platform for enabling the creation of context-sensitive or adaptive SFEs. A context-sensitive SFE provides a user interface (UI) that exhibits some capability to be aware of the context and to react to changes of this context in a continuous way. As a result such a UI will be adapted to a person's devices, tasks, preferences, and abilities, thus improving people's satisfaction and performance compared to traditional SFEs based on manually designed UIs. Serenoa will provide domain-independent reference models and languages (ASFE-DL, AAL-DL) devoted to simplify the development of adaptive SFEs. It is developing the ASFE-DL language targeted to the specification of context-sensitive SFEs that can adapt to different platforms, modalities or users in ubiquitous computing environments. Also, it is developing AAL-DL as a high level description language for expressing advanced adaptation logic (Rules). The adaptations of SFEs are carried out according to these rules. Furthermore, Serenoa is aimed to provide the technology for the automatic generation of context-sensitive SFEs with user involvement and machine learning.

An authoring tool is a software package which developers use to create and package content deliverable to end users. It is a software that allows (usually non-programmer) users to create their own courseware, web page, or multimedia applications and the associated navigating tools. It requires less technical knowledge to master and is used exclusively for applications that present a mixture of textual, graphical and audio data. It could be developed as a plugin or it could be developed as a web based application as well.

AUTHORING TOOL AND ARCHITECTURE

In Serenoa project, we have developed an Eclipse-Plugin based authoring tool, where our objective was to develop two editors, namely the Service Editor and the Rules Editor to create the service definitions and context rules, respectively. The Service Editor can be used to create Abstract-level service descriptions, while the Rules Editor supports creating the context-sensitive transformation rules using the Design and/or Code views. Another important

objective was to allow the authoring tool to communicate with a remote repository via RESTful Web Services based interface and support all the CRUD (Create, Retrieve, Update, and Delete) operations on the service definitions and context-dependent transformation rules.

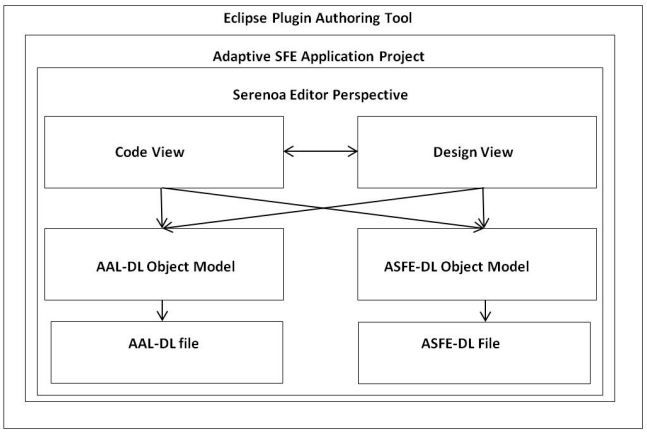

Figure 1: Authoring Tool in Eclipse Plugin Environment

Figure 1 shows the high-level implementation architecture of the authoring tool plugin, which shows how the two different views, namely Code view and the Design view have been implemented along with the plugin specific perspective. On the other hand, Figure 2 shows the screenshot of Rules editor with the Design view (the description of each part of the editor is explained in the caption of the figure). The important thing is that the Action element of the rule can include nested rules and the Condition element can be made as complex condition (nested conditions) with different logical and arithmetic operators. The Rules Editor is designed as a multi-page editor with Graphical Editor and XML Editor to facilitate the creation and editing of rules in graphical and code views, as shown in Figure 2. The Code View is displayed simply in an XML Editor as the model languages are based on XML as underlying language. The Graphical editor is developed using GEF, which internally uses MVC [5] architectural pattern and provides technology to create rich graphical editors and views for Eclipse Workbench UI. The languages schemas and example rule files provided by the project partners helped us in designing the graphical part of the rules editor.

The graphical part is designed to have a tool palette which contains elements' list for creating rules. The element list is divided into three categories as *Event*, *Condition* and *Actions* (the structure used for specifying *Rules*). The drawing area is used to drag and drop elements from the palette and displays the rules in a graphical representation (in the form of boxes). This helps the developers to easily and quickly develop the rules for Adaptive SFE applications. The editor can also display the *Outline View* in the form of tree structure and provides displaying and easy navigation feature for the Rules drawn in the drawing area. The *Property View* shows the properties associated with each element that can be set in the fields. The miniature view of the drawing area provides an overview of when editing in the XML editor. The Graphical editor also provides features such as Selection, Moving, and Resizing of elements, Actions (like Redo, Undo, Delete, and Zoom), keyboard shortcuts, context menu, drag and drop, cut and paste. Both the XML editor and the Graphical editor are synchronized to reflect changes made at any time in their respective views.

IMPLEMENTED FUNCTIONALITIES

Currently the following functionalities have been implemented and supported by the authoring tool in the form of wizards for each of the functionality.

Creation of a Serenoa Project

In order to create rules or UI definitions for the services, first of all a project must be created, which will serve as a container for holding other project related elements. In order to do this, the user creates a Serenoa project with the help of a wizard, where the user can set different properties and structure.

Loading of XML Schemas and Data Type Definitions

In order to create rules or UI definitions for the services, the relevant XML schemas and DTSs files are loaded and used by the respective wizard to create the desired depth or hierarchy of the structure of the rule.

Creation of AAL-DL Rules

Once the XML schemas and DTSs have been loaded in the project, the user can create adaptation rule, either with the help of provided wizard, or simply creating an XML file in the project. After this step, the user opens this file in AAL-DL Editor, as shown in Figure 1, for creating new rules or editing the existing ones.

Creation of ASFE-DL Service Definition Locally

Once the XML schemas and DTSs have been loaded in the project, the user can create UI definitions for the services. Currently, only the *Abstract* UI definitions are supported, but in later versions, the editor will support creating the *Concrete* and *Final* UI definitions.

Creation of ASFE-DL Service Definition Remotely

Once the Abstract UI definitions have been created locally, they can be stored by creating a new service on a remote repository via RESTful services interface. In order to do this, the user selects the service folder from the project explorer and presses *Shift + C*, which opens a dialog where the user selects the respective files and finishes the process by clicking on *Create* button.

Deletion of Rules and Service Definition

The user can delete adaptation rules or service definitions not only from the local project, but also from the remote

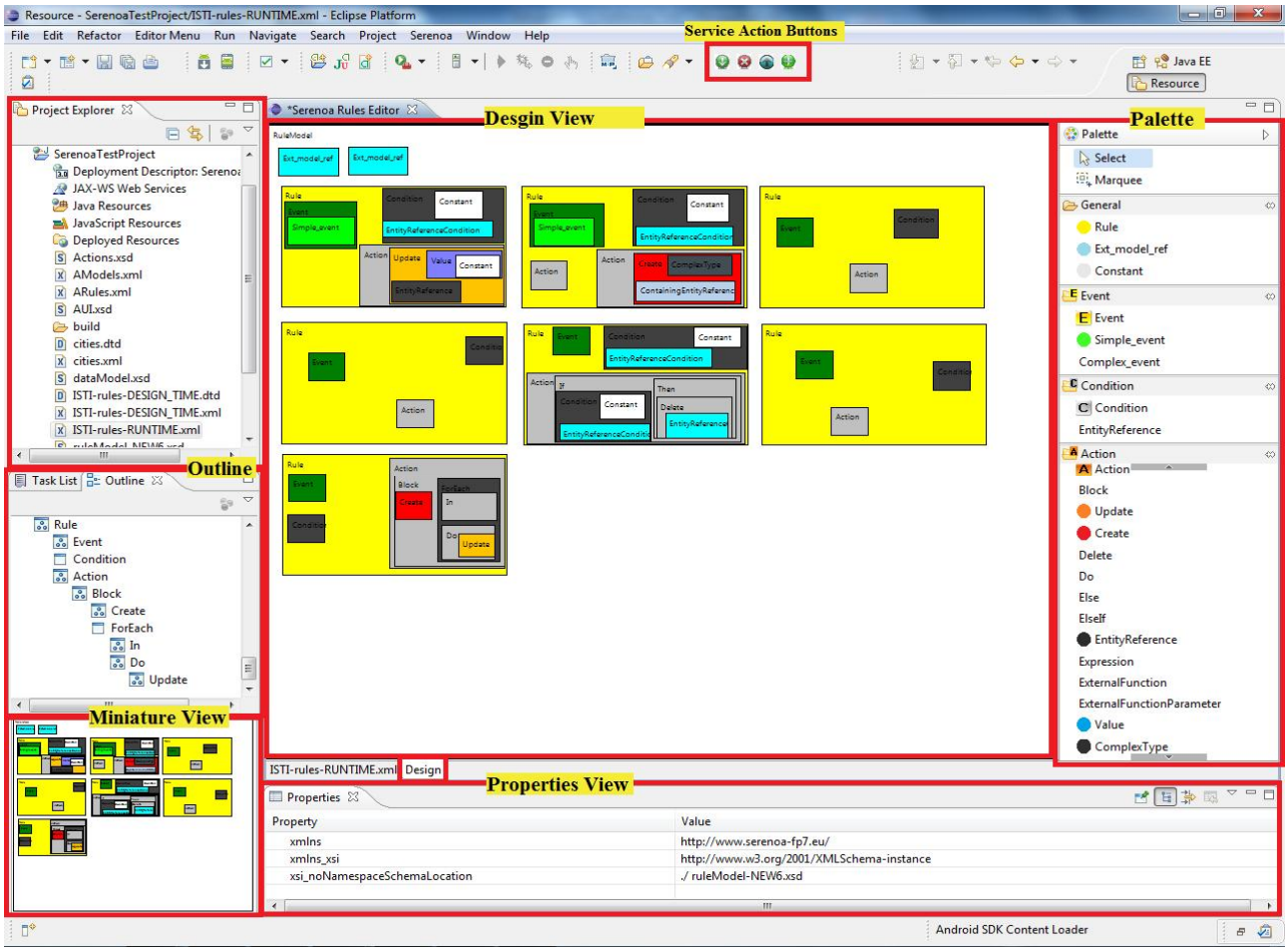

Figure 2: The *Design View* of the Rules Editor of the authoring tool plugin. On the right hand side, the *Palette* block shows the elements that can be dragged and dropped on the design canvas in the middle. These elements include, mainly, *Rule, Event, Condition* and *Action*, where all the elements can be dropped inside a *Rule* element. Multiple rules can be created, as shown in this figure. In the bottom, the *Properties View* shows the respective properties which a user can set by selecting an element. On the left hand side, The *Project Explorer* shows the project and all the files of rules and UI definitions created in it. Also, the *Outline* area shows the tree view of the rules and the *Miniature* View shows the miniaturized view of the design view

repository by pressing *Shift + D* as well.

Retrieval of Rules and Service Definition

The adaptation rules or service definitions can be retrieved or downloaded from a remote repository. First of all, the user selects the project, where the Service should be downloaded to, and then presses *Shift + G*.

Updating Rules and Service Definition

After the retrieval of adaptation rules or service definitions, the user can make changes to them. After saving it locally, the user can update them on remote repository by pressing *Shift + U* and then selecting the updated rules and service definitions files.

CONCLUSION

We have developed the first version of Eclipse-Plugin based authoring tool. The current implementation comprises of two editors, namely the *Service Editor* and the *Rules Editor* to create the service definitions and context rules, respectively. Currently, the Service Editor can be used only to create *Abstract*-level service descriptions, while the Rules Editor supports creating the context-dependent transformation rules using the *Design* and/or *Code* views. The authoring tool is able to communicate with a remote repository via RESTful Web Services based interface and support all the CRUD (Create, Retrieve, Update, and Delete) operations on the service definitions and context-dependent transformation rules.

FUTURE WORK

In future, the authoring tool will support the editing of SFEs at *Concrete* and *Final* levels along with testing of SFE designs on a variety of target environments, e.g. through a means to emulate those environments (e.g. mobile, tablet, desktop, etc.) for different interaction modalities, e.g. graphics, voice, touch etc.

ACKNOWLEDGMENT

This work received funding from the European Commission within the context of Serenoa Project (http://www.serenoa-fp7.eu/) under grant agreement number 258030 (FP7-ICT-2009-5).

REFERENCES

1. Fabio Paternó; Model-based Design and Evaluation of Interactive Applications; Springer Verlag, November 1999, ISBN 1-85233-155-0
2. Puerta, A.R; A Model-Based Interface Development Environment; IEEE Software, 14(4), July/August 1997, pp. 41-47.
3. Fabio Paternó; Model-based Tools for Pervasive Usability; Interacting with Computers, Elsevier, May 2005, Vol.17, Issue 3, pp. 291-315.
4. EU FP7 Serenoa Project; http://www.serenoa-fp7.eu/
5. Model View Controller (MVC) Design Pattern; http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller