AN ANNOTATION TOOL FOR ENHANCING THE USER INTERFACE GENERATION PROCESS FOR SERVICES

Paoli Izquierdo, Jordan Janeiro, Gerald Hübsch, Thomas Springer and Alexander Schill
Department of Computer Science, Institute of System Architecture, Chair of Computer Networks
Dresden Technical University
Dresden, Sachsen 01187, Germany
E-mail: paoli.izquierdo, jordan.janeiro, gerald.huebsch, thomas.springer, alexander.schill {@tu-dresden.de}

Abstract – The user interface generation process is still a complex issue. The manual creation process is time consuming and complex because it requires the combination of the application developer work and the user interface designer work. Therefore, many approaches investigate the automatic user interface generation. However, currently such approaches are not able to take usability aspects into account, due to the lack of information for such a process. Thus, we propose a tool which intends to ease and enhance the automatic user interface generation process for services by introducing service annotations that are attached to services and provide additional information for the process of automatic user interface generation.

I. Introduction

The creation of user interfaces is still a complex problem. Some statistics state that 50% of the time for building an application is due to the user interface development [1]. Therefore, some approaches propose the idea of generating the user interface automatically [6].

Many of these approaches focus on the generation of user interfaces for services. A common way to generate user interfaces for services is their inference from services description, like WSDL or WADL files, because it saves development time and helps to avoid errors.

Given that data types can be matched to specific graphical controls, the inference mechanism to create GUI forms can be straightforward. However, the inference mechanism is limited to a certain degree, because the developer may need to include more detail to the controls on the form that cannot be inferred from purely technical descriptions. The annotation tool we present in this paper includes both the GUI inference mechanism and the annotation functionality that is required to improve the process of automatic GUI generation. The annotation tool generates an initial version of the user interface from a WSDL file through inference. The innovation of our work is the introduction of a further step in which a UI expert can modify the appearance of the generated interface and store the modifications separately in service annotations. The service annotations are made available to consumers of the service and can be exploited to accelerate the development and to increase the quality of user interfaces for interactive applications that utilize the annotated services.

a. Use Case

In our use case the user selects a web service description file to be imported. For example, in Figure 1 we see one of the web service files that *Amazon* [2] uses for the operations of adding, removing and modifying items in the shopping cart.

The annotation tool displays the file name in the "WSDL Files" section of the main window of the program.

Once the user clicks on the Amazon web service, the operations found in it are listed in the "WSDL Operations" section and when he clicks on the name of an operation, the inferred UI for it is displayed in the center of the window.

The center of the window is the main section where the user can customize details of the automatically generated user interface. All the possible properties which can be edited for each UI component are presented in the right area of the tool, called "Properties". Selecting the option "Save Properties" button, the tool stores the annotations related to certain web service.

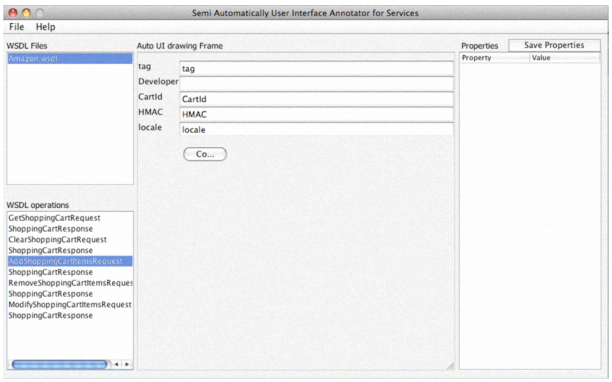

Figure 1. Annotation Tool Main Window

The user can also edit the text of the label of any of the controls by clicking on the label itself in order to view its content and language properties.

Finally, when the user is ready to save all the annotations he has done, the user can export them to an XML file.

In this paper, we first describe the process of extracting the information from the web service description file, and then we explain the implementation details of the process. Later on we refer to other related works to automatic UI generation process and annotation tool, and finally disclose the conclusions.

II. Web Service Information Extraction

The Figure 2 presents the overall information extraction process to generate the user interface. It begins with the information extraction, by analyzing a WSDL file. During this phase, the tool first identifies all the information to automatically generate the user interface for the service, namely: data types, operations, parameters and return values.

In the next step, the inference mechanism will generate a graphical control for each parameter based on the data type extracted from the web service file and a form is created for each operation.

III. Implementation

The annotation tool was written in Java version 1.5 with Swing elements for the graphical interface because they offer more sophisticated UI controls than AWT.

Even though WSDL is a type of XML file, we made the decision to not only use DOM for the information extraction, but to complement it with the WSDL4J library [3]. We can exploit the features that this API offers in order to locate only the information we need out of the web service file, in our case the operations and parameters, with minimal effort.

As part of the user interface generation process, our tool creates a series of java objects, which represent the operations, parameters and annotations. These objects

2009 19th Int. Crimean Conference "Microwave & Telecommunication Technology" (CriMiCo'2009). 14-18 September, Sevastopol, Crimea, Ukraine
 ISBN: 978-966-335-244-2. IEEE Catalog Number: CFP09788

mimic the relationship between operations and parameters, which is established in the web service file. They also store information found on the web service file.

On the other hand, the annotation objects serve to store information that is not present in the WSDL file, which makes the automatically generated UI more detailed and tailored to each operation. For example, an operation to sign in to a service typically requests a username and password, since both of these parameters are of data type string, a text box can be used for each of them. However, even though the same control is used, the properties are different depending on the parameter, because a text box for a password should not display what the user is typing, but a mask of asterisks or some other character instead, in order to conceal the sensitive content.

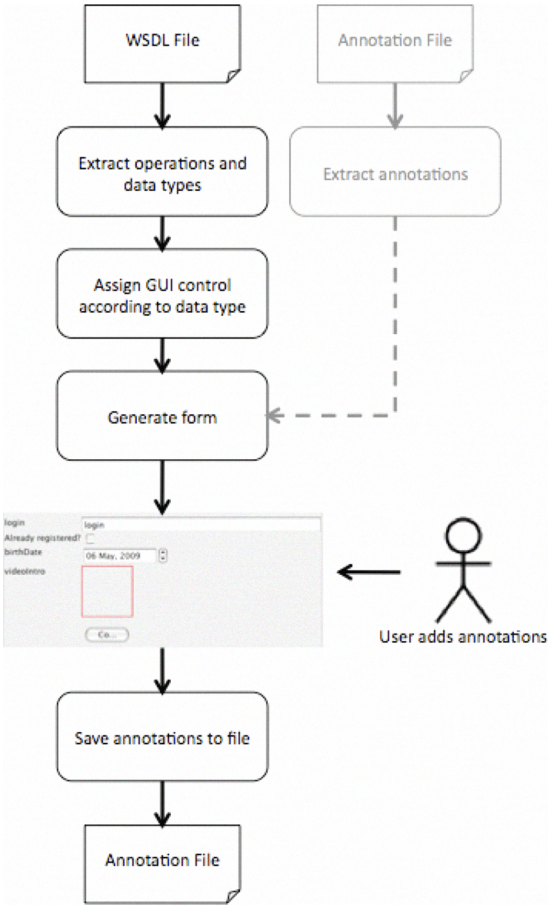

Figure 2. UI Generation and Annotation Process

Other examples of details that can be annotated are: defining the length of the parameter data, setting predefined content, establishing whether a field is mandatory, specifying the media type of a parameter, setting the text alignment, among others.

The Annotation objects are created at the same time as Parameter objects are, this way, if an annotation file already exists, the information obtained from it during parsing is stored in these objects. However, if an annotation file is not present, then the object will be ready to store information when modifications are made later on.

There are different types of annotation objects that the system can create. Based on the data types of the parameters, the tool will create either a *PanelAnnotation*, a *TextAnnotation*, a *SpinnerAnnotation* or a *CheckboxAnnotation* object depending on the data type of the parameter.

The data types that our annotation tool supports are the standard data types for WSDL [4] according to the W3C specification:

- *string*
- *anyURI*
- *date*
- *datetime*
- *time*
- *gYearMonth*
- *gYear*
- *gMonthDay*
- *gDay*
- *gMonth*
- *duration*
- *boolean*
- *float*
- *decimal*
- *double*

If the data type is either string, float, decimal or double, a TextAnnotation object will be created, thus displaying a text box. We choose a text box control for these datatypes because it is a generic user interface widget which is normally used by users to input values to forms.

Given that the data types date, datetime, time, gyearmonth, gyear, gmonthday, gday, gmonth and duration are predefined data types, the tool creates specific widgets to handle such data types based on a spinner control widget concept, which allows the user to iterate over a set of values and select them instead of specifying it as text. In the case of all of these data types we know exactly how many subelements a piece of data has, for example, gYearMonth describes the year and month data. We also know with a high level of accuracy the information they hold because even though we may not constraint the lower and upper bounds of the years for the user to choose from. For example, we know exactly how many and what are the months of a year and how many days there are in a month, therefore we can prepare such values in a list to allow a user to choose between them. A Spinner control, according to the Java documentation [5], presents to the user a set of items to choose from but it can also allow the user to input their own data as long as it is a legal value, thus making this type of control the better suited for date related data types. For storing the annotation details a SpinnerAnnotation object is created by our tool.

In the case of Boolean data types, a checkbox is generated and a CheckboxAnnotation object is created. A checkbox is what best represents Boolean data because it is able to display only two possible states, checked or unchecked. Besides, it is also a control which non-expert computer users are familiar with. For this specific type of control it is also very important to edit the label content in a way that allows the user to easily understand which kind of input is expected from them.

Finally, if the data type is anyURI, then a PanelAnnotationObject is created and a Panel control is displayed on the screen. We use this control as a placeholder, because the resource that the URI points to could be of a different type depending on the web service, like a picture, a video or an animation.

We point out that the annotation objects are subclasses from the Annotation class. The Annotation class is capable of storing information common to all types of annotations, such as: a reference to the parameter they annotate, the label content, label language and the type of UI component associated to it. This gives the tool flexibility to add new annotation classes which might be necessary in the future by creating specific new subclasses.

 ISBN: 978-966-335-244-2. IEEE Catalog Number: CFP09788

After all the before mentioned objects have been created, the user interface can be displayed. In addition to all the controls, a label is also generated for each one of them.

The user can now annotate the displayed UI elements as well and the tool can store the information entered in the Annotation objects, which it will read in order to create the annotation file.

The annotation file is an XML file with the structure shown in Figure 3. To create this file, the tool will go through all existing operations and parameters objects of a given WSDL file in order to save the information it finds in the annotation objects. Should any of the annotation properties be blank, the property will not be considered. The data is first loaded into a DOM tree and, when the data in all the annotation objects has been processed, the content of this tree is written to a file. We've chosen as a convention, to give the XML annotation file the same name as the WSDL file. This way, whenever a WSDL file is selected to be imported with the tool, it will search in the same directory for a file of the same name but with the .xml extension as well, if found, it will suggest the user to import that file too. The tool does, however, allow the possibility to browse to a different annotation file in case none is found or the user prefers to use a different one.

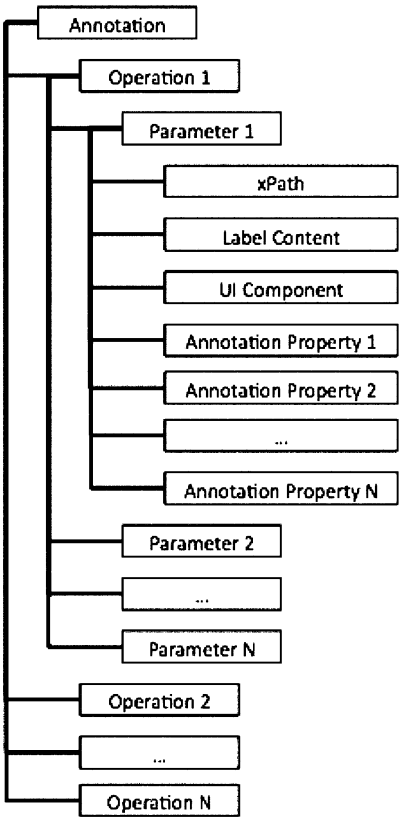

Figure 3. Tree diagram of XML Annotation file

IV. Related Work

In [6] an inference mechanism based on XML Schema and XForms is presented. This work is able to retrieve data from XSD files and generate a User Interface with XForms controls. Although it is not designed specifically to work with web service files, it can extract information from them as well. They do not go into detail as to how they handle information extraction from WSDL files so we are not able to draw comparisons with our work in that respect. However in terms of the technology used for the user interfaces, our solution can in future versions allow for a larger variety of UI controls because XForms doesn't offer as vast a selection as Java Swing.

The TERESA XML [7] project is able to generate user interfaces for a range of different devices. Instead of inferring the UI from XSD or web service files, they work with Task Models (TM) and with Abstract/Concrete User Interfaces (AUI/CUI). This work is not based on data types to generate user interfaces and, when working with web service files, the developer still needs to have either a task model or an abstract/concrete user to use this tool.

The OpenXava [8] project is an application framework, which is capable of generating user interfaces automatically from a data model. From a Java class, this tool can generate a graphical user interface with create, read, update and delete behavior. For more complex applications or to add detail, this framework supports JPA annotations as well as its own. However the annotations have to be coded in the java class file and it cannot perform the information extraction and inference based on web service files.

V. Conclusion

We have developed an annotation tool which supports the automatic generation of User Interfaces for web services through the inference of graphic controls based on the data types of operation parameters.

After the generation of a user interface, developers often need to add extra details to enhance it and this tool successfully aids them in that respect, by allowing them to annotate the generated form with such additional information.

Even though this tool supports a range of graphic controls, it can further be extended to support many more by adding new annotation classes.

VI. References

[1] *Myers, B., Rosson, M.*: Survey on User Interface Programming. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 195-202. ACM Press, New York (1992)

[2] *Amazon Web Services LLC*, Amazon Web Services, http://aws.amazon.com, 2009.

[3] *Freemantle P., Duftler M.,* Java Specification Requests JSR110: Java APIs for WSDL 22 Sep 2006.

[4] *Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.*, Web Services Description Language (WSDL) 1.1, March 2001

[5] *Sun Microsystems,* Java 2 Platform, Standard Edition, v. 1.4.2 API Specification, 2003

[6] *Spillner, J., Schill, A.,* Analysis on Inference Mechanisms for Schema-driven Forms Generation, Tagungsband XML-Tage, September 2007, pp. 113-124.

[7] *Berti, S., Correani, F., Paternò F., Santoro, C.*, The TERESA XML Language for the Description of Interactive Systems at Multiple Abstraction Levels. Proceedings Workshop on Developing User Interfaces with XML UIXML: Advances on User Interface Description Languages, May 2004, pp.103-110.

[8] *Paniza, J.*, Automatic User Interface with OpenXava: an Evolutionary Option for GUIs, June 2008.

2009 19th Int. Crimean Conference "Microwave & Telecommunication Technology" (CriMiCo'2009). 14-18 September, Sevastopol, Crimea, Ukraine
 ISBN: 978-966-335-244-2. IEEE Catalog Number: CFP09788