Teaplet, Weekly Report, 14 March 2011

This week, we have done some generic for the output of JavaML component, which is classname.xml. This way, we will get rid of if-elses that we have written for xml parsing.

An example is given below:

```
<xs:group name="expr">
    <xs:choice>
        <xs:element ref="send"/>
        <xs:element ref="new"/>
        <xs:element ref="new-array"/>
        <xs:element ref="var-ref"/>
        <xs:element ref="formal-ref"/>
        <xs:element ref="field-ref"/>
        <xs:element ref="field-access"/>
        <xs:element ref="array-ref"/>
        <xs:element ref="paren"/>
        <xs:element ref="assignment-expr"/>
        <xs:element ref="conditional-expr"/>
        <xs:element ref="binary-expr"/>
        <xs:element ref="unary-expr"/>
        <xs:element ref="cast-expr"/>
        <xs:element ref="instanceof-test"/>
        <xs:element ref="literal-number"/>
        <xs:element ref="literal-string"/>
        <xs:element ref="literal-char"/>
        <xs:element ref="literal-boolean"/>
        <xs:element ref="literal-null"/>
        <xs:element ref="this"/>
        <xs:element ref="super"/>
    </xs:choice>
</xs:group>
```

“expr” is a “xs:group” kind, which can include many other kinds, such as <xs:choice>, list of <xs:element> in this example.

Looking into “expr” element, first of all, we see a <xs:choice> element, which informs us to select one of the followings. First of the followings are is an <xs:element> kind, which is composed of various other components. But if we look into an <xs:element> type, we see something like below:

```
<xs:element name="send">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="target" minOccurs="0"/>
            <xs:element ref="arguments"/>
        </xs:sequence>
        <xs:attribute name="message" type="xs:string" use="required"/>
        <xs:attribute name="idref" type="xs:string"/>
        <xs:attribute name="idkind" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

The “send” component is composed of a <xs:complexType>, which may include many other types, but if we look into <xs:complexType>, we see that it has 1 <xs:sequence> component and 3 other <xs:attribute> component. <xs:attribute> components are usually the most basic ones. They are mostly composed of String types, which are relatively primitive for other complex objects.

By using this generic approach, we started to write <xs:*> elements and most basic <xs:element> and <xs:attribute> components. By this way, we plan to be able to map every element and its tags to an object directly. Although it seems pretty straightforward, DTD and XSD of JavaML tool are highly complex. We will follow this approach this week and meanwhile, we also need to map Java Applet objects to JSF ones, because this part is only for java parsing. It is responsible from the parsing process of the whole source code.

Anıl Sevim

Özge Tokgöz

Berkan Kısaoğlu