Teaplet Weekly Report, 23 March 2011

Anıl:

This week, Anıl continues constructing our base classes of AJCON to put the resulting XML data of JavaML into the memory.

Beside thinking on after usage of memory structure, Anıl thought on the topic "How to construct those memory structures".

With the help of the XSD's and DTD's, Anıl continued his work.

- Constructed the base classes stated below
- Developed an algorithm to bind objects in the memory

This algorithm is like:

- From the most general object to the most specific object, we keep every smaller object in the fields.
- But on the other hand from most specific to most general one:
    + We will keep a stack of every element.
    + When a start-element call made by the XML parser, we will push the newly created object to the stack
    + When a end-element call made by the XML parser, we will pop an element from the stack

...

And so on, when we create an object, we will know the parent of the object.

Also we will create the object from it's name like:

```
Class.forname("ClassName");
```

We will give the class name as string -as we read from the XML file. And then we directly create the related object.

We will not make any if-else comparison.

Özge and Berkan:

We discussed on several algorithms about Translation part according to the possible results of Anıl's work. As a result, we come with an algorithm.

After Anıl's work, we have all the classes that maps to an element in a Java source hierarchically. To be more clear, we will have such classes after Anıl's work:

```
public class CFormalArgument {
        private CModifiers modifiers;
        private CType type;

        ... // Some other fields

        public CFormalArgument()
        {
name = null;
```

```
        id = null;
        }

                public CModifiers getModifiers() {
                return modifiers;
        }

        public void setModifiers(CModifiers modifiers) {
        this.modifiers = modifiers;
        }

        ... // Some other getter and setter functions

        }


        public class CModifiers {
        private List<CModifier> modifiers;

        public CModifiers()
        {
        modifiers = new ArrayList<CModifier>();
        }

        public List<CModifier> getModifiers() {
        return modifiers;
        }

        public void setModifiers(List<CModifier> modifiers) {
        this.modifiers = modifiers;
        }
        }


        public class CType {
           private String primitive;
        private String name;

        ... // Some other fields and constructor

        ... // Getter and setter functions

   }
```

At last we will end with a general object that contains all other objects as fields in it. After generating such a memory structure, we are ready to traverse the structure to manipulate it.

Manipulation part will be something like:

- Take the root object
    + Other objects are fields of the root object

- Look all the fields of the objects recursively
    + Define related UI elements and push them to an array
- Process all the UI elements and after processing, set them null to mark them "processed"
    + Process operation contains constructing of new objects in the memory
    + New UI classes that will be written into a JSF page. For example:

```
public class Combobox extends UIelement{
        Vector<ComboboxItem> comboboxitems;
        String objectid;

        ... // Some other fields

        public Combobox(){
                comboboxitems = null;
                objectid = null;

                ... // Set some other fields to null
        }

        ... // Getter and setter functions
}


public class ComboboxItem{
        String objectid;
        String value;
        String name;

        ... // Some other fields

        public ComboboxItem(){
                objectid = null;
                value = null;
                name = null;
        }

        ... // Getter and setter functions
}
```

- This time we will produce new objects according to the JSF UI elements.
- After processing all the elements in the memory, we will traverse objects again, but this time objects contains JSF UI elements also.
- While traversing, we will construct the related files.