3-The Econet project

3.1- Introduction

3.1.1- Context of the project

Component-based software engineering is now a common approach in many areas of software development. This approach permits to make well-constructed software, it is based on a structural separation. A component must be autonomous and dialog with other components through his interfaces.

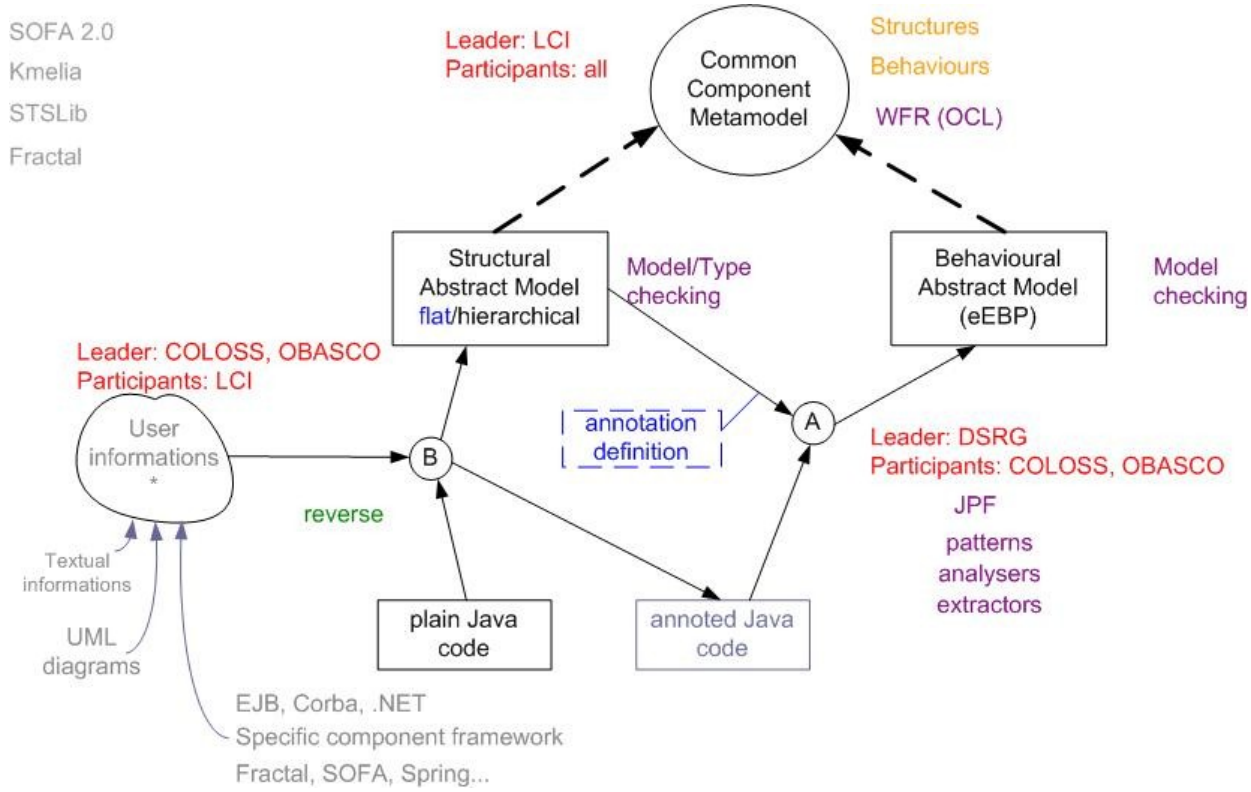
Today, there is no solution to check if a software is well-constructed or not. For example, we can use classes diagrams to describe an architecture, scenarios to describe behavioral features and the Java language to implement a solution but it is very difficult to check if the implementation is components oriented or not. It is difficult to check if the components are independent from each others. UML [0] specification or academic solution like ACME [1] are based on an abstract model when EJB [2], CCM [3], OSGi [4] or other industrial solutions are focusing on implementation. It is difficult to make links between these two specifications.

In the next part we will explain a solution proposed by the COLOSS team [5] to resolve this problem by using reverse engineering technologies and the MDA approach.

3.1.2- The Econet project

The solution proposed by the *COLOSS team* is called the *Econet project*. First we present the global scheme of this project.

Global scheme :



This project is divided in three independents parts :

- A component metamodel, described with the EMF [6], standard permits to represent components and dialogs between them without concerns about implementation solutions (EJB, OSGi, ...)
- **Process B** : this is the part that we will study in this project, as we can see on the scheme, plain Java code is in entry of this process and produce a structural abstract model conforms to a component metamodel. User informations can be added to simplify the extraction of components.
- Process A : This process analyse the structural abstract model given by the *process B* and extracts a dynamic behaviour specification. We don't explain this part of the *Econet project*.

3.1.3- Why our project ?

An implementation of the *process B* was proposed by the *COLOSS team* but it is just an experimental solution : all the functionalities are not implemented. The result produce by this process is not yet a model conforms to the EMF metamodel but textual informations that represents the structure of the application studied. This implementation is an eclipse plug-in.

We have to enhance and complement this implementation and to make possible the creation of the model that will be used by the *process B* in the future. For this work, the main API is Java Development Tools (JDT). This API is not used a lot and, therefore not documented. We have also to write a documentation which explain and detail the structure of this API and how to use it. This documentation is available in the part X of this report.

If all this work was done we can implements a functionality that will permit to represent component structure with a graphical representation. We suppose that we can use the language *dot* [7] because it is very simple to generate a graph with this language or the java plug-in *prefuse* [8] to make prettiest graphs with more possibilities of representation.

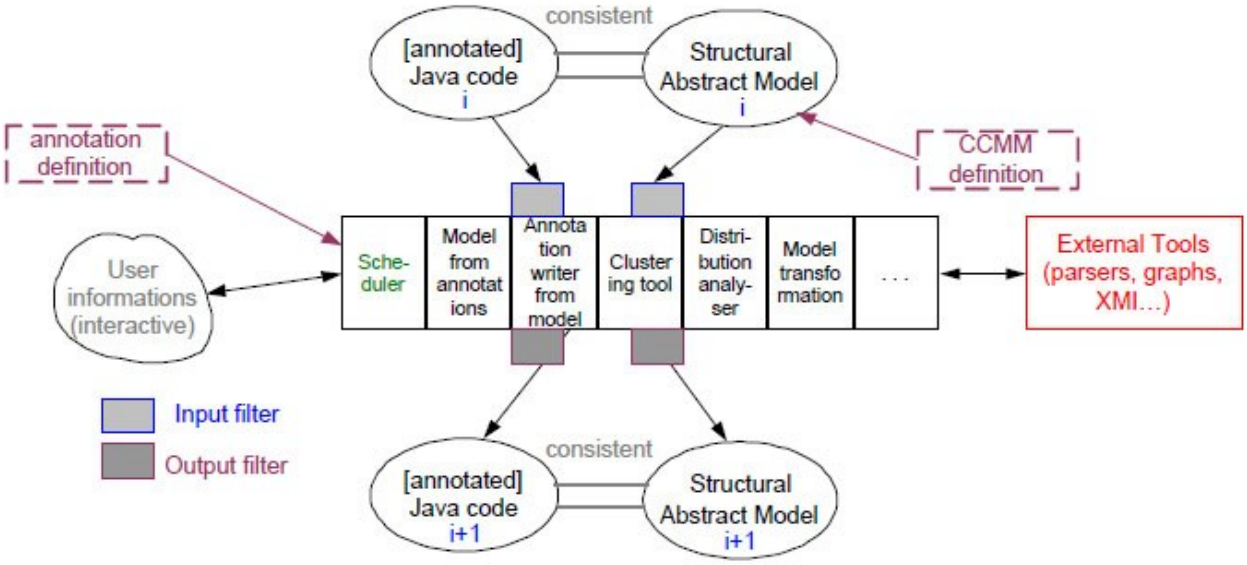
3.2- The process B

3.2.1- Introduction

As we explain in the introduction, the *process B* permits to extract an abstract model conforms to a component metamodel from plain Java code. In this part we describe this process more precisely.

3.2.2- Architectural view

Scheme :



Details :

A plain Java code (with possibles annotations) and a structural abstract model (not necessary) are in entry of the *process B*. Annotations permit to make links between the plain java code and the abstract model. As we can see on the precedent scheme, the *process B* is an iterative process represented by the markers *i* and *i+1*. Each iteration of this process visits one box and completes the java code and the abstract model. The java code and the structural abstract model are always consistent at the input and the output of the process. The process is a *Rules Based System* (RBS). We describe the RBS in the part X of this report. Naturally, rules involve an order of application of them, without this order, rules could negate the work done by another. To resolve this problem we have to find the best application order for the rules, we will explain this in the same part. It is impossible to make the process fully automatic so, it will ask the user for additional informations to check which rules can be applied.

Détails of the toolbox :

- Model from annotations : permits to generate a model conforms to the EMF metamodel with an annotated java code in entry
- Annotation writer from model : permits to annotated a plain java code though a component model
- Clustering tool : takes a primitive abstract component based model in entry and produce a more precise one with the notion of components composition.
- Distribution analyser : enables the possibility to explore XML model and makes a components based model
- Model transformations : models transformations in general, patterns analyser or automatics transformations, etc.

4- References

[0] – http://fr.wikipedia.org/wiki/Unified_Modeling_Language

[1] – http://www.cs.cmu.edu/~acme/

[2] – http://fr.wikipedia.org/wiki/Enterprise_JavaBeans

[3] – http://www.ibm.com/developerworks/webservices/library/co-cjct6/

[4] – *OSGi Service Platform Core Specification release 4*, August 2005, The OSGi Alliance

[5] – http://www.sciences.univ-nantes.fr/info/perso/permanents/attiogbe/COLOSS/

[6] – http://www.eclipse.org/modeling/emf/

[7] – http://www.graphviz.org/

[8] – http://prefuse.org/