Chapter 6
Model Evolution Leads by Users Interactions

Charles-Georges Guillemot
Upper Alsace University, France

Frederic Fondement
Upper Alsace University, France

Michel Hassenforder
Upper Alsace University, France

ABSTRACT

Reuse has long proven its interest in software engineering. It is also useful while modeling: reusable elements can be concepts in a meta-model, or modeling elements in libraries, as libraries of components, or of activities to be used in frameworks. Choosing among those elements implies a compromise between modeling needs and provided services. Some authors have proposed to describe such services by means of models. However, this description is rarely available in practice. We propose here to let users improve these specifications by themselves in a participative way by merely tagging the reusable elements. Tags express how the elements can be used, or how to compare the elements together. To do so, questions, which depend on a model, are automatically asked to the user while working. Answers, provided under the form of tags or keywords, are processed in order to evolve the model.

INTRODUCTION

Modelling is an activity that everyone performs everyday. Indeed, it is the normal functioning for our brain: We all associate a mental image with each person or item around us. This image is a model, made of concepts and relations between them, which describes that entity in order to mentally simulate its behaviour. Einstein (Einstein & Infeld, 1967) illustrates the model notion in the form of a metaphor:

"Physical concepts are free creations of the human mind, and are not, however it may seem, uniquely determined by the external world. In our endeavor to understand reality we are somewhat like a man trying to understand the mechanism of a closed watch. He sees the face and the moving hands, even

DOI: 10.4018/978-1-61350-438-3.ch006

Figure 1. Simple image preparation with a view to counting the number of cells

hears its ticking, but he has no way of opening the case. If he is ingenious he may form some picture of a mechanism which could be responsible for all the things he observes, but he may never be quite sure his picture is the only one which could explain his observations. He will never be able to compare his picture with the real mechanism and he cannot even imagine the possibility or the meaning of such a comparison."

The creation of such models is an intuitive step, and is not necessarily made explicit by describing it in a document written in a formalized or even natural language. However, such formalizations may become necessary as soon as one needs to reuse or share such models. So in recent years, Model-Driven Engineering (MDE) (Schmidt, 2006) tends to play an increasingly significant role in various domains, and in particular in the software development. Indeed, all software aspects are defined by models and thus when a software system needs evolve, his models must evolve accordingly. Although many model transformation languages (MOF, 2010) (Jouault and al., 2008) have been developed, this task remains time-consuming, tedious, and error-prone. Consequently, if a user of one software has a need which is not describes in the model, he will have to choose an other software or wait an update, hoping that this update takes account of his need.

We will illustrate this situation with a simple example of image processing (Figure 1). This process prepares a picture with a view to counting the number of cells which are inside. The first action allows to delete objects which are not cells and keeps only good candidates. Secondly, we successively erode and expanse all remaining objects to remove all latest artifacts. Then the picture is ready to be analyzed. But this scenario is completely determined by the source picture. A modification of this picture type could modify a part or all of the process.

For instance, we will apply the same processing over a different picture, which include intensity skips. In this picture type the previous process will not produce correct results, because intensity skips could interfere with filter actions. So, various activities (Figure 2) must be applied to delete these skips, and get a correct result.

Figure 2. Modified previous scenario