ImageNetDiff: Finding Differences in Models

Lawrence Cabac, Kolja Markwardt and Jan Schlüter

Department of Informatics, University of Hamburg, Germany

Abstract. In this paper we propose a method and present a tool as plugin for RENEW that supports the process of discovery of differences in possibly conflicting versions of all kinds of supported diagrams. These diagrams can be either semi-formal UML diagrams, Petri net models or simple JHotDraw drawings. Instead of searching for differences on the syntactical or even on the semantical level, we choose to find differences on the visual level.

1 Introduction

During development of applications developers frequently encounter (and have to deal with) different and/or conflicting versions of model artifacts. Especially in shared projects where modeling artifacts are shared through source code management systems (SCM) such as the Concurrent Versions System (CVS) or Subversion conflicts frequently appear and have to be resolved manually by the developer. This is especially true for Petri net-based applications, since here the models are the code base of the system and are thus treated as usual code with all attributes, such as collective code ownership. In the evaluation of the code (Petri nets) of other modeling artifacts the main problem is the identification of the syntactical differences or equalities. However, on the one hand formally it is very hard to verify graph equality and even harder to determine the minimum of parts that are different. The graphical representation, on the other hand, may contain valuable hints for the mentioned problems but may also differ without change in the syntax. The merging of changes is usually a manual task, even if only different parts of the nets have been modified. In contrast, when text-based source code is used, merging of non-conflicting concurrent changes is possible. To our knowledge no tools exist so far that manages the merging to some extent or even supports the developer in this task. Even if a string representation of the net code exists, usually this code is not handleable by common tools such as *diff* [3] (or windiff). This means that models in source code management systems are treated as binary files, even if the file representaiton of the diagram (model) is text-based, such as XML – as with SVG (Scalable Vector Graphics).

In this paper we propose a simple but efficient method that can simplify the task of the discovery of differences under certain conditions. To this means we exploit the graphical representation of the nets and transfer the problem to finding differences in the visual image of the Models. We also present an implementation of the method as plugin for RENEW [5, 6], a multi-formalism tool whose graphical engine is based on JHotDraw (www.jhotdraw.org) and supports all kinds of modeling techniques (e.g.: Petri nets, use case diagrams, sequence diagrams, class diagrams) and drawings (including import

Cabac L., Markwardt K. and Schlüter J. (2009).
ImageNetDiff: Finding Differences in Models.
In *Proceedings of the 7th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 156-161
DOI: 10.5220/0002202101560161

and export possibilities). In Section 2 we describe the method, its implementation and integration within RENEW. Section 3 presents seveal examples to illustrate the method, the tool and possible appications.

2 Discovery of Net Differences

The development of models within development groups frequently leads to conflicting models. Even if the system models are decomposable in many parts, still the problem persists – as with all source code – that within one design artifact (Petri net or UML diagram) several changes can occur concurrently and have to be merged. In this situation two tasks have to be performed. First, the differences have to be identified. Second, the changes have to be included. If conflicts occur in text-based source code developers are supported by powerful tools and techniques, such as diff tools, versioning systems, etc. For models (Petri nets) these tasks usually have to be performed manually.[1] We believe that tool support for the discovery of net differences can accelerate the development of (net system) models significantly.

Scenarios. We can distinguish at least two different scenarios in which the tool can be utilized: the *similarity check* and the *difference discovery*. In the similarity check a developer does not know, whether two models (Petri nets) or two versions of the model (Petri net) own the same code (are syntactically/semantically equal but may differ visually). For text-based code there exist code beautifiers that manage to unify the style of code as a preparation for the differences tools. Restricted layout possibilities which could have the same effect as code beautifiers are usually too restrictive for model designers. Often model elements or text inscriptions have been moved in the diagram by another developer and this has been committed to the repository resulting in a conflict. If the models (or the model versions) contain only small differences (e.g. only one element has been moved) the ImageNetDiff image will show instantly that the models are syntactically equal. The checking of the equality of the models is thus reduced to the checking of the graphically differing parts.

In the difference discovery the visual areas of the model that own differences can be easily spotted by the developer. Again, if small changes have been made in the model, such as the removal or the addition of elements, the ImageNetDiff image will directly and clearly show the differences. Removed object are highlighted as red elements in the diff image and additions are highlighted blue. If this is not the case and substantial changes have been made, at least the ImageNetDiff image points out the net areas which are of concern to the developer and which parts have not changed.

Technique. The tool (see also [2]) makes use of the internal export function of RENEW and the ImageMagick[4] tool kit. For the production of the differences image in the format of Portable Network Graphics (PNG) or alternatively Encapsulated Postscript (EPS) first the nets are being exported to the file system as image. Then the exported images are passed on as arguments to the imaging tool to compute the differences image, which will also be stored in the file system. The resulting image will feature light

[1] An alternative strategy is the avoidance of concurrent changes.

grayish drawing elements for the parts of the original images that are equal and two different shades of red for the additional and removed graphical parts Finally, for the convenience of the user the image is displayed by RENEW once the computation of the differences image has finished. Sources of models that are to be compared can be either drawings (diagrams) that are opened within the editor of RENEW or files from the file system. Also command line commands exist to quickly access the functionality of the plugin without loading the whole graphical editor of RENEW. On the command line it suffices to define the two comparing files as argument. Thus the tool can also be included in scripts. As a support for Subversion the tool is able to directly compare the current working version of a model with the locally stored code base file. This allows the developer to use the `renew diff <file>` command in the same manner as `svn diff <file>`. Especially if no (real) change has been done (i.e. involuntary saving of the model during inspection) the equality check can help to prevent superfluous check-ins. However, some limitations of the presented method exist that result from the used tools.

- For a flawless comparison the compared images must have the same size.
- The comparison can not be customized, yet.
- For instance, the color scheme is fix.
- The results for models in which all graphical elements have been moved are not satisfying, yet, because the images are compared coordinate pixel against coordinate pixel.
- However, a simple move of *all* elements do not effect the result, since the images are clipped before export.
- There is no integration with the model representation in RENEW, yet. Thus, the discovery of changes is supported but the knowledge has to be transferred to the model manually by the developer.

3 Examples

The presented method and tool is able to compare a broad variety of supported models and drawings. Here we present as example the results of the tool for a Petri net model.

A Petri Net: The Mulan Knowlege Base. As an example for the presentation of the method we present a Petri net from the developing of multi-agent systems with MULAN: the knowledge base net of the MULAN standard agents. The two nets differ – pragmatically – in the fact that they support two different property files formats: simple properties (*kb*) and XML notation (*kbe*, kb enhanced).

The net that supports the enhanced representation is built upon the simple version, thus they are comparable. To find the similarities and differences of the implementation we present fragments of both nets in Figures 1 and 2. The fragments show the initialization of the net with the initial knowledge parts of the agents interface to the knowledge base and the interface that handles the initialization of decision components (active knowledge). Figure 3 then shows a screenshot of the resulting difference image (similar fragment).[2]

[2] The dashed squares and ellipses are added manually.

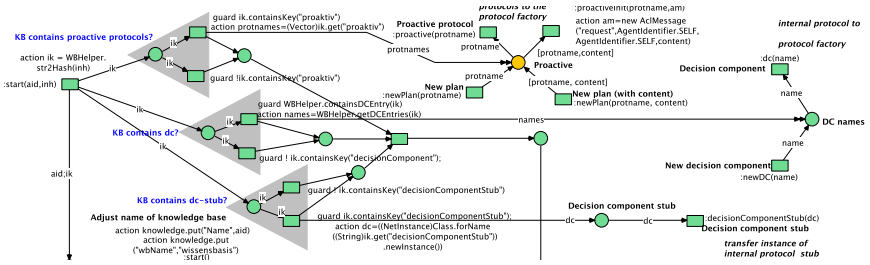

Fig. 1. Knowledge base net template of a MULAN agent.

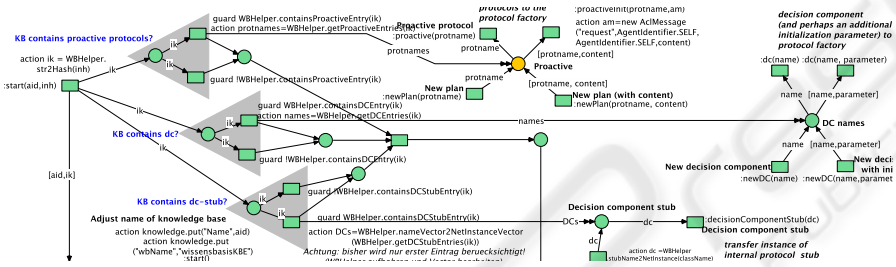

Fig. 2. Enhanced knowledge base net template of a MULAN agent.

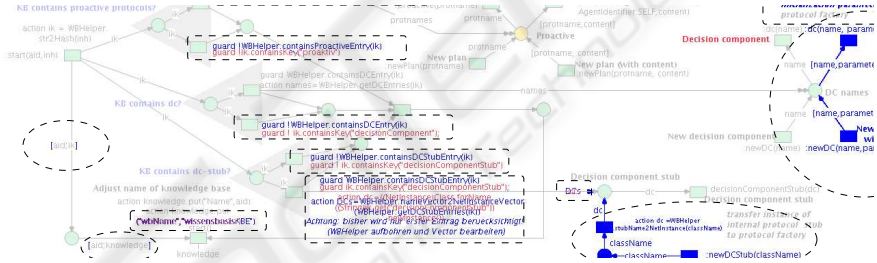

Fig. 3. Screenshot showing differences of the two Petri nets.

The developer's awareness is instantly attracted by the bright red and bright blue net elements and inscriptions. One can see simple additions – manually marked in the image by dashed outlined squares – and also changes to the code / inscriptions – ellipses – that have been made. The image shows clearly that all of the old net structure has been preserved. Only additional net elements and inscriptions have been added and some inscriptions have been altered.

In a scenario of a shared development, if a developer is confronted by a concurrent change of the net, which results in a conflicting version of the net code, the tool can help the developer to decide whether the code has been manipulated, the syntax has not been changed and/or if the changes have been made in the same areas of the net. Thus, the manual act of merging the code or model can be significantly simplified and accelerated.

Comparing (Embedded) Images. The tool is even able to show differences in (embedded) images. As a second example Figure 4 shows a (constructed) image (PNG, Image 1) and a minimally altered version (Image 2). Usually the difference is not even detectable. However the difference image to the right shows clearly the difference of the two images. This possibility is not very surprising, since this is the original application domain of the ImageMagick *compare* tool.

Fig. 4. Differences of embedded images (PNG).

4 Conclusions, Discussion and Outlook

Although the approach is rather simple, the results are effective and surprisingly efficient. Developers of (Petri net) models have the means to check for differences in their graphical code by the means of visual support. Clearly a code beautifier for Petri nets and other models would improve the results of the ImageNetDiff plugin considerably. At least for Petri nets net components [1] could help to impose a conventionalized structure upon the nets.

The presented approach makes use of the graphical representation of the diagrams such as UML diagrams or Petri nets, the export to an image format and the power of the graphical framework ImageMagick. There are, however, several other possibilities to tackle the presented problem. One could compute equality of Petri nets on the ground of the formal representation including node and arc ids or develop a PNML (Petri net XML representation) diff tool.

The presented method and the tool leaves room for many improvements. By choosing different color schemes and maybe also opaqueness in the diff images the readability could still be improved significantly. However, since the used tool's main purpose of comparing images is not concerned with graph representations, it does not support this feature and a reimplementation or switch to another tool could – with some effort – produce better results. The interpretation of the graphically highlighted elements could lead to an integration of useful information within the Petri net editor to further support the merging of concurrent changes.

In principle, with the presented method the results from image processing have to be re-transferred to the application domain. Alternatively, similar differences can be computed and presented to the developer on the direct analysis of Petri net structures. Here, additional information could support the process of matching elements in Petri net versions. For instance, id-tagged net elements (in RENEW transitions and places have ids) could be matched. However, this would not solve the problem of constructs that have different ids but are syntactically equal. A method based on a Petri net (or model)

representation is also less general than the presented method, which can be applied to other graphs such as UML diagrams.

References

1. Lawrence Cabac, Michael Duvigneau, and Heiko Rölke. Net components revisited. In Daniel Moldt, editor, *Proceedings of the Fourth International Workshop on Modelling of Objects, Components, and Agents. MOCA'06*, number FBI-HH-B-272/06, pages 87–102, Hamburg, Germany, June 2006. University of Hamburg.
2. Lawrence Cabac and Jan Schlüter. Imagenetdiff: A visual aid to support the discovery of differences in petri nets. In *15. Workshop Algorithmen und Werkzeuge für Petrinetze, AWPN'08*, volume 380 of *CEUR Workshop Proceedings*, pages 93–98. Universität Rostock, September 2008.
3. Gnu diff utilities. online, 2008. http://www.gnu.org.
4. Imagemagick homepage. online, 2008. http://www.imagemagick.org/.
5. Olaf Kummer, Frank Wienberg, and Michael Duvigneau. Renew – the Reference Net Workshop. Available at: `http://www.renew.de/`, July 2008. Release 2.1.1.
6. Olaf Kummer, Frank Wienberg, Michael Duvigneau, Jörn Schumacher, Michael Köhler, Daniel Moldt, Heiko Rölke, and Rüdiger Valk. An extensible editor and simulation engine for Petri nets: Renew. In Jordi Cortadella and Wolfgang Reisig, editors, *Applications and Theory of Petri Nets 2004. 25th International Conference, ICATPN 2004, Bologna, Italy, June 2004. Proceedings*, volume 3099 of *Lecture Notes in Computer Science*, pages 484–493. Springer, June 2004.