Introducing the *Quick Fix* for the Petri Net Modeling Tool Renew

Jan Hicken, Lawrence Cabac, Michael Haustermann

University of Hamburg, Faculty of Mathematics, Informatics and Natural Sciences, Department of Informatics
`{1hicken,cabac,haustermann}@informatik.uni-hamburg.de`

1 Extended Abstract

Many modern integrated development environments (IDEs) such as Eclipse [2] support developers by providing a *quick fix* feature. If the application detects syntax errors in the source code, it can propose fixes addressing the error by providing context-sensitive alternatives the developer can choose from interactively.

Considering Petri net modeling tools, similar functionality can be useful for inscriptions for high-level nets. We want to transfer the functionality to Renew [1], where the Java-based inscriptions may be treated analogously to source code in IDEs. This includes a mechanism for detecting and highlighting errors for developer on the one hand and algorithms for suggesting possible fixes to these errors on the other hand. Furthermore, the feature shall apply these fixes automatically, so that the developer has to adjust as little as possible source code manually.

The first category of suggestions consists of Java fields: The choices consist of the declared and accessible fields for a denoted class in the source code. If an entered field is unknown, the quick fix can provide a list of fields using the Java Reflect API. However, this can lead to a lot of choices being presented to the developer, when the class defines a great amount of fields. In order to address excessive lists of suggestions, lists may be filtered to fields having the same prefix as entered by the developer.

Similar to the fields, methods can also be searched by using the Java Reflect API. Filtering the list of methods declared by a class may also be done by comparing prefixes. In addition to that, the entered parameters need to be considered when providing suggestions to the user. A method giving the wrong parameter types, the feature can suggest possible method overloads.

Variables, that are used in the Petri net can be declared in Renew's declaration node. If the net contains a declaration node with at least one declared variable, all used variables have to be declared. An undeclared variable causes the quick fix to suggest possible types for that variable and adding the corresponding statement in the declaration node.

When assigning a value to a variable, the value can either be a literal, a newly constructed object or the returned value from a method call. Determining the type in the first case is trivial, because the corresponding literal's primitive type can be evaluated by the compiler immediately. To determine the type of a newly constructed object, the constructor's class name has to be well-known, which means the class name is either fully qualified or an already imported class. Giving the case of a well-known class name, the corresponding declaration statement can also be generated easily. Moreover, the information of a fully-qualified class name can be used to construct an import statement for that class or its whole package right away. The last case is also rather trivial, because the return type of a method and its corresponding class object can be determined using the Java Reflect API.

With the quick fix feature, we achieved a further integration of modeling support, which helps the developer reaching his goals. Errors in the source code are explicitly highlighted and possible fixes can be applied interactively. In addition to the management of development errors, the feature can be used similar to an autocomplete feature. The developer does not have to look up every class member he wants to use in the API documentation but can choose from any alternatives the quick fix provides. We streamlined the development process by enabling a much faster development in an integrated environment, which is easily extensible.

When suggesting fields or methods for a class, the quick fix can provide all known members. If the amount of class members is high, the filtering only applies to suggestions with the same prefix as the entered text. It is possible to extend this feature to filter for matching types and parameters, which also enables a prioritization of suggestions. The suggestions for variables are reliable when it comes to assignments, where the right part of the expression is parseable and the type is known. However, when it comes to arc inscriptions and variable types, which have not been imported, the current algorithm does not give sufficient results. The former case may require an analyzation of other in- and outgoing arcs regarding the particular place or transition and its inscriptions. Supporting unimported types depends highly on the ability to automatically import classes found in the classpath. In addition to that, the class hierarchies are not part of the suggestion mechanism at all.

Furthermore, the quick fix feature not only is applicable to Java source code within inscriptions but also affects modeling errors. Common Petri net properties are yet verifiable through algorithms. These may be used to address unwanted discrepancies between wanted and actual net properties using the quick fix.

References

1. Kummer, O., Wienberg, F., Duvigneau, M.: Renew – the Reference Net Workshop (Jun 2015), `http://www.renew.de/`, release 2.4.2
2. The Eclipse Foundation: Eclipse: The Eclipse Foundation open source community website (Jun 2015), `http://www.eclipse.org/`