SNiFF+ Talks to Rational Rose

Interoperability using a Common Exchange Model

Sander Tichelaar and Serge Demeyer,
Software Composition Group, University of Berne, Switzerland,
{tichel,demeyer}@iam.unibe.ch

Abstract

Nowadays development environments are required to be *open*: users want to be able to work with a combination of their preferred commercial and home-grown tools. TakeFive has opened up SNiFF+ with a so-called "Symbol Table API"; Rational has opened up the UML tool Rose via the so-called "Rose Extensibility Interface (REI)". On the other hand, efforts are underway to define standards for exchanging information between case-tools; CDIF being a notable example. This paper reports on our experience to generate UML diagrams in Rational Rose from the symbol table in SNiFF+ using a standard CDIF exchange format.

More information on the model and the SNiFF+ extractor is available at: http://www.iam.unibe.ch/~famoos/InfoExchFormat/.

All comments are welcome: famoos@iam.unibe.ch.

1) Introduction

Tool interoperability is becoming of more and more importance. Developers want their favourite environment for their favourite programming language and might use in parallel some external tools, for instance, for program analysis, visualisation, task automation, etc.

SNiFF+ provides an open development environment: users are able to use the environment and seamlessly integrate their preferred debuggers, compilers, versioning system, editors, etc. Apart from that SNiFF+ provides an API to access the symbol table of SNiFF+ projects. This allows for external programs to retrieve information such "what are the classes in this project?", "what is the superclass of this class?", etc.

We use this information to produce UML [Booc96a] diagrams in Rational Rose. An extractor called sniff2cdif extracts the needed information using the API and puts it in an industry standard representation for exchanging information between CASE tools called CDIF [CDIF94a]. This intermediate format is then converted into VisualBasic calls to Rational Rose's Extensibility Interface (REI).

The intermediate format is based on a model to describe object-oriented source code in a language independent way. We use this model to integrate a whole set of tools for visualisation, metrics & heuristics and other analysis tasks.

In this paper we first show how we use Rational Rose to produce UML diagrams of SNiFF+ projects. Afterwards we give a quick introduction to the Information Exchange Model that is used for the intermediate format and its representation in the industry standard CDIF.

2) From SNiFF+ to Rational Rose

We have developed a program, sniff2cdif, that extracts the information from SNiFF+ by querying its symbol table. The SNiFF+ Symbol Table API provides a set of C functions that give access to the

symbol table data structure. Partly there are a set of pre-defined queries, partly the data structure of the symbol table itself can be walked through. Using the API the main information for the model is extracted and put in the CDIF format. We feed this CDIF format into an other program, in our example a script that transforms the information to something Rational Rose understands (see Figure 1).

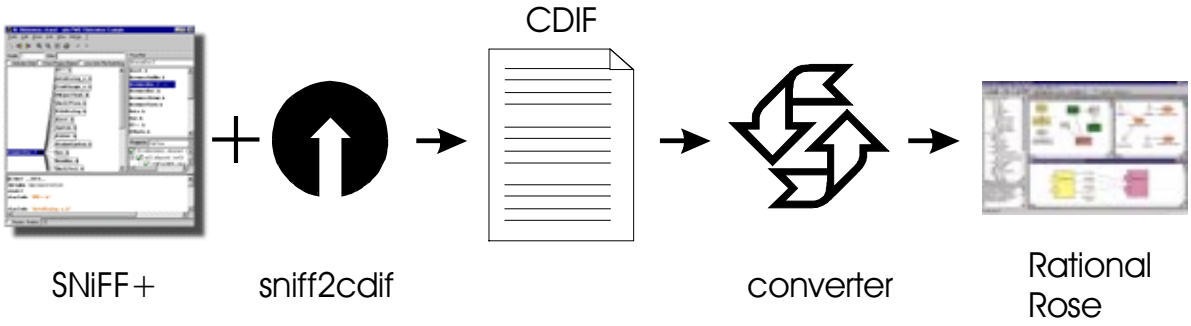


Figure 1: SNiFF+ connection to Rational Rose

The information we extract from the symbol table is based on a model that describes object-oriented source code in a language-independent way (see section 3)). The SNiFF+ symbol table provides a lot of the information that is described in this model. Especially convenient is the fact that SNiFF+ provides queries to extract information about which method calls with method and which attribute is accessed by which method (which is actually the information SNiFF+ uses itself for its reference browser).

The output of sniff2cdif is a file that contains the data in CDIF, a conveniently queryable ASCII format, which is an industry standard for exchanging information between CASE tools. Once we have this language independent format we either feed it to a tool that understands the format, or convert it into a format a tool understands. In our example we convert it to VisualBasic calls that the UML tool Rational Rose understands. Running this VisualBasic in Rational Rose produces a UML diagram of the SNiFF+ project.

3) The Information Exchange Model

2.1 The Conception

Many tools have been and are being developed to support developers in performing activities such as visualisation, metrics & heuristics and system reorganisation. However, systems are written in different implementation languages (C++, Ada, Smalltalk and even Java). To avoid equipping all of the tools with parsing technology for all of the implementation languages, we have developed a model for information exchange between reengineering tools.

The model is a language independent representation of object-oriented sources and should provide enough information for the reengineering tasks of the tools (see Figure 2).

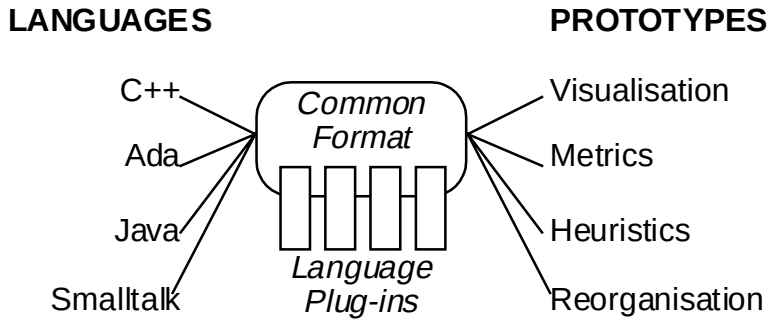


Figure 2: Conception of the Exchange Format

However, we cannot know in advance all information that is needed. Therefore, the model is extensible in a couple of ways. First, tools may need to work on the language specific issues of a system. Therefore, we allow *language plug-ins* that extend the model with language-specific items, but don't break the language independent tools. Second, we allow *tool plug-ins* to extend the model so tools can, for instance, store analysis results or layout information for graphs, again without breaking other tools.

2.2 The Core Model

The core model consists of the main OO entities, namely `Class`, `Method`, `Attribute` and `InheritanceDefinition` (see Figure 3). For reengineering we need the other two, the associations `Invocation` and `Access`. An `Invocation` represents the definition of a `Method` calling another `Method` and an `Access` represents a `Method` accessing an `Attribute`[1]. These abstractions are needed for reengineering tasks such as dependency analysis, metrics computation and reengineering operation. Typical questions we need answers for are: "are entities strongly coupled?", "which methods are never invoked?", "I change this method. Where do I need to change the invocations on this method?".

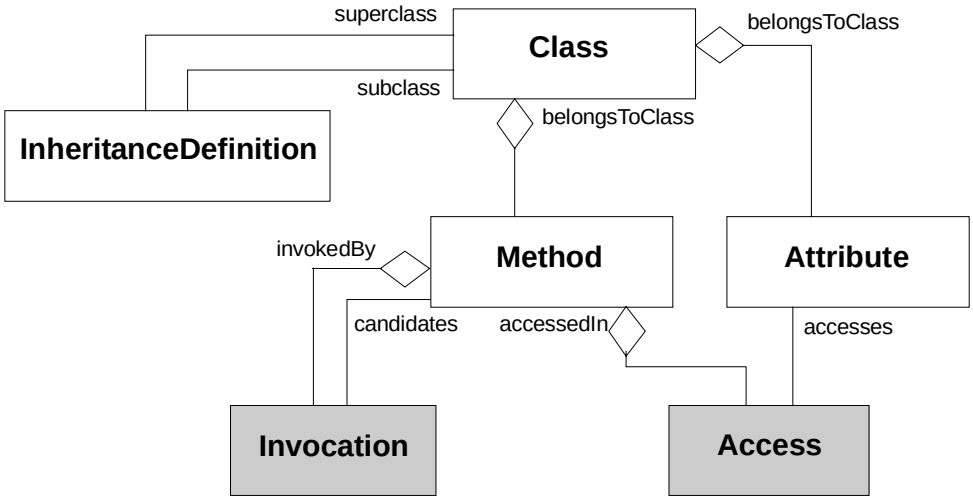


Figure 3: The Core Model

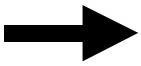
A complete description can be found in [Deme98].

[1] Actually, the complete model is more general: an Invocation is about behavioural entities (such as methods and functions) calling other behavioural entities and an Access is about a behavioural entity accessing a structural entity (such as attributes and global variables).

4) CDIF as Interchange Format

We have adopted CDIF [CDIF94a] as the basis for the information exchange of information in the FAMOOS exchange model. CDIF is an industrial standard for transferring models created with different tools. The main reasons for adopting CDIF are, that firstly it is an industry standard, and secondly it has a standard plain text encoding which tackles the requirements of convenient querying and human readability. Next to that the CDIF framework supports the extensibility we need to define our model and language plug-ins. An example:

```
package gui;
class Widget {
  int wTop;
  …
  void print() {
   System.out.println(
       “Top = “ + wTop);
  }
}
```



```
(Class FM1
  (name “Widget”)
  (uniqueName “gui::Widget”)
)
...
(Attribute FM5
  (name "wTop")
  (uniqueName “gui::Widget.wTop”)
  (belongsToClass “gui::Widget”)
)
...
(Method FM8
  (name “print”)
  (uniqueName “gui::Widget.print()”)
  (belongsToClass “gui::Widget”)
)
...
(Access FM12
  (accesses “gui::Widget.wTop”)
  (accessedBy “gui::Widget.print()”)
)
```

5) Conclusion

In this paper we have shown how we use the SNiFF+ symbol table API to extract information from SNiFF+ projects. We have used this information to produce UML diagrams in Rational Rose. The exchange of information between those two tools is based on a language-independent representation of object-oriented source code. Actually, any tool that can understand the intermediate format can be integrated with SNiFF+. Besides the connection with Rational Rose, our group is currently working on metrics and graph tools that work on the same intermediate format and therefore can also be used together with SNiFF+.

Acknowledgements

This work has been funded by the Swiss Government under Project no. NFS-2000-46947.96 and BBW-96.0015 as well as by the European Union under the ESPRIT IV programme Project no. 21975 (FAMOOS, see http://www.iam.unibe.ch/~famoos/).

References

[Booc96a] Booch, G., Jacobson, I. and Rumbaugh, J, The Unified Modelling Language for Object-Oriented Development. See http://www.rational.com/.

[CDIF94a] CDIF Technical Committee, "CDIF Framework for Modeling and Extensibility", Electronic Industries Assocaiation, EIA/IS-107, January 1994. See http://www.cdif.org/.

[Deme98] Serge Demeyer and Sander Tichelaar, "Definition of the FAMOOS Information Exchange Model - Version 1.1", Technical Report, 1998. See http://www.iam.unibe.ch/~famoos/InfoExchFormat/.