Moose: an Agile Reengineering Environment

Stéphane Ducasse ducasse@iam.unibe.ch
Tudor Gîrba girba@iam.unibe.ch
Michele Lanza michele.lanza@unisi.ch

Introduction

Software systems are complex and difficult to analyze. Reengineering is a complex activity that usually involves combining different techniques and tools.

MOOSE is a reengineering environment designed to provide the necessary infrastructure for building new tools and for integrating them. MOOSE centers on a language independent meta-model, and offers services like grouping, querying, navigation, and meta-descriptions.

Several tools have been built on top of MOOSE dealing with different aspects of reengineering like: visualization, evolution analysis, semantic analysis, concept analysis or dynamic analysis.

The Architecture of Moose

Moose uses a layered architecture as shown in Figure 1.

Information is transformed from source code into a source code model. MOOSE supports multiple languages via the FAMIX language independent meta-model. In the case of VisualWorks Smalltalk models can be directly extracted via the built-in parser. In the case of other Smalltalk dialects, the code has to be first ported to VisualWorks and then imported into MOOSE. For other source languages like C/C++, Java MOOSE provides an import interface for CDIF and XMI files.

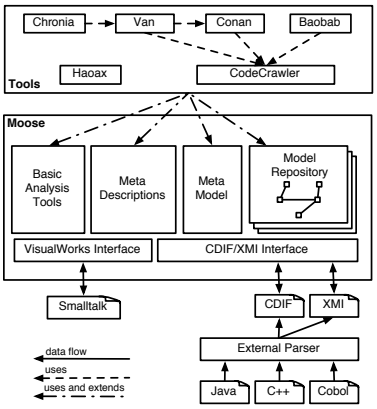

Figure 1: The architecture of MOOSE.

Every model contains entities representing the software artifacts of the target system. Every entity is represented by an object, which allows direct interaction and querying of related entities, and consequently an easy way to query and navigate the model. MOOSE can maintain and manipulate several models in memory at the same time via a

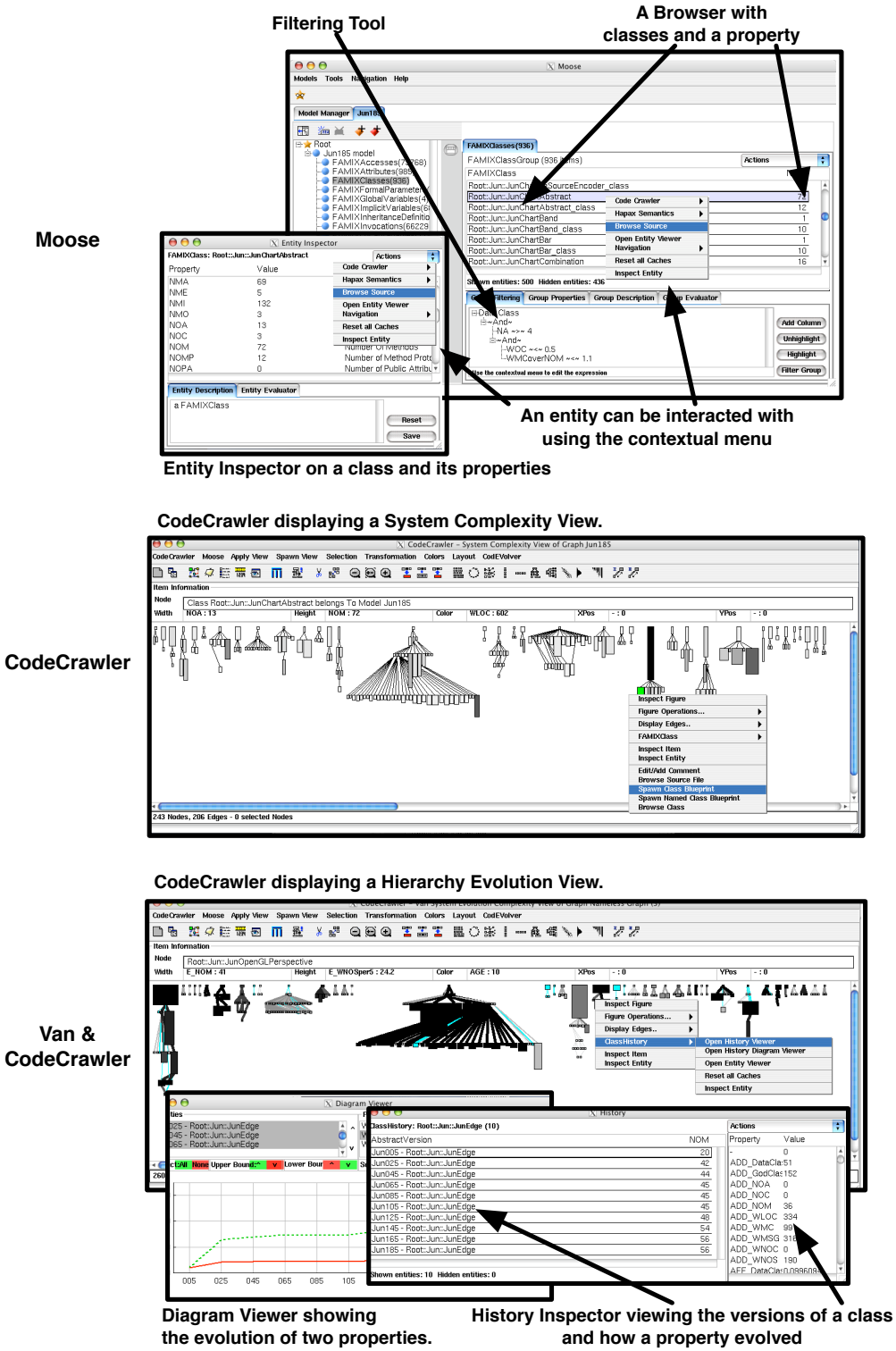

Figure 2: Screenshots of MOOSE, CodeCrawler and Van.

model repository. Every entity is described by a meta-description, which is then used by the environment to display user interfaces or load/save entities. These meta-descriptions are extensible by other tools and are used by different tools. Examples of the supported meta-descriptions are: description of related entities, menu, description of properties.

MOOSE also provides basic tools that use the are generic by using the meta-descriptions (see Figure 2): Browser, Entity Inspector and Filtering Tool.

Tools Built on Top of Moose

CodeCrawler. CodeCrawler is a visualization tool implementing polymetric views which is based on a graph notion where the nodes and edges in the graph can wrap the entities in the model. For example, in Figure 2 we see a screenshot of CodeCrawler displaying a hierarchy of a system called Jun. In Figure 1 it is shown that CodeCrawler is used by different tools for different visualizations.

ConAn. ConAn is a concept analysis tool and its target is to detect different kinds of patterns in the model based on combining elements and properties. ConAn uses CodeCrawler for visualization purposes and supports analyses like: X-Ray views for understanding the internal of classes, identification of recurring code patterns, and views for hierarchy understanding.

Van. Van is a tool for analyzing the evolution of systems. At its core, it defines the Hismo meta-model which is based on the notion of history. In Figure 2 we show how Van uses CodeCrawler to display the evolution of the class hierarchies in the Jun system. Van offers other analyses like history measurements, change characterization.

Chronia. Chronia is a tool that bridges MOOSE with versioning systems like CVS and it enables analyses of how developers change the system.

TraceScraper. TraceScraper analyzes the dynamic traces from different perspectives. For example it offers measurements and visualizations for dynamic traces.

Hapax. Hapax is a semantic analysis tool. It makes use of the comments and names of the identifiers from the code to recover the domain information and it also offers clustering of different parts of the system based on how they use the same terms.

Moose Availability

MOOSE is completely implemented in VisualWorks Smalltalk under the BSD license: it is free and open source software. A demo package containing the 3.1 alpha release can be downloaded from:

www.iam.unibe.ch/∼scg/Research/Moose/download/Moose31AlphaJun.zip

Further information can be obtain from the official webpage located at:

www.iam.unibe.ch/∼scg/Research/Moose/

Acknowledgments. We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project and "RECAST: Evolution of Object-Oriented Applications" (SNF Project No. 620-066077, Sept. 2002 - Aug. 2006).