Faculty of Automation and Computers

Bachelor Program: Calculatoare si Tehnologia Informatiei

# XCore: Framework for Software Analysis Tool Development

**Master Thesis**

Alexandru Ștefănică

*Supervisor*
Asistent Profesor
Dr. Ing. Petru-Florin Mihancea

Timişoara June, 2015

# Contents

**1 Introduction** **3**
1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
1.2 Previous Work . . . . . . . . . . . . . . . . . . . . . . . . . 5
1.3 Current Development . . . . . . . . . . . . . . . . . . . . . 5
1.4 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . 5

# Chapter 1

# Introduction

Any fool can write code that a computer can understand. Good programmers write code that humans can understand

Martin Fowler

## 1.1 Motivation

One of the goals of any software developer is to write quality code. Code that respects company policy, that has a large test coverage percentange, that uses well defined mechanism thus making it portable on different operating system . . . etc. In order to establish the quality of encode and enforce different standards analysis tools where build. Some of the most used tools are:

**NullTerminator** It represents a pseudo-automatic refactoring tool which eliminates the null checks by use of the **Null Object Design Pattern** [8]

**Wala Tool** Developed by IBM Research it implements a series of dataflow and type analysis algorithms.

**FindBugs** Widely used tool to detect common programming language hacks

**Intellij IDE** A platform developed by JetBrains which implements over 100 code inspector tools

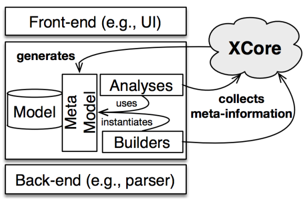

Figure 1.1: Generic Backend for analysis tools [9]

**CodePro** A platform which implements software metrics developed by LOOSE Research Group. Goes by the name of INCODE, also [4]

Most analysis tools share a generic back-end which can bee seen in 1.1. The back-end is responseble for parsing the code in the desired projects and extracting a low-level abstractions / artefacts of the code (e.g the AST of the code). The extracted low-level artefacts, though usefull, are too granular for most analysis tools, it needs to be processed furthere, resulting in high-level abstractions / artefacts called a **model**. This model is structured according to a **meta-model** defined and implemented by the developers of the tool. [9]

The construction of the **meta-model** is a recuring task when it comes to building analysis tools. **IPlasma** [7] uses the **Memoria** meta-model in order to compute metrics. Wala [10] builds its own meta-model in order to preform analysis. Eclipse, as Wala, provides its own meta-model called JDT [11] which is used by most of the Eclipse analysis plugins.

Building such a meta-model and the necessary back-end is a tedious task, at best. Another major problem that arises when building such a meta-model, and any other program for that matter, is the evolution in time. We want a meta-mode that can be easily extensible over time, but also have the ability to integrate / inter-operate with other tools / meta-models in order to obtain complex tools. All of these requirements are difficult to design and implement and, different approaches have been made propoused.[2] [9] In order to obtain the mentioned goals, some tools such as inCode[4], have introduced architectural faults which compromised the type safety of the program. This ultimately lead to poor code which is hard to maintain and understa. [1]

## 1.2 Previous Work

Previously we have build an Ecliplse plugin, called XCore which solved the previously mentioned problems. [1]. The tool works by providing the software developers with the means to describe the meta-model of their analysis tool directly into the source code of the tool, without the need to actually implement the meta-model. This is done by providing a higher level of abstraction called meta-meta-models. The meta-meta-model is implemented by using the java annotations system.[1] [3]

When the tool is compiled, the java compiler will notice the annotations and invoke the XCore annotation process before actually processing the entire project. Our annotation process will generate, based on the information provided, the appropriate meta-model for the tool and will enforce all the necessary restrictions on meta-model entities. This is perfectly type safe because after the code generation phase is over the java compiler can fully typecheck the entire system !

## 1.3 Current Development

## 1.4 Related Work

In essence XCore is a tool for meta-programming. It contains meta-meta-model which allows the user to describe any meta-model respecting the constraints. Thus is bears some similarity with other tools which try to solve the problems metioned above, like it such as:

**Rascal** [5]

**Soot** [6]

**Fame** [2]

Rascal is a meta-model programming language. It allows a user to express any meta-model for the development / integration of analysis tools. To the best of my knowledge there is no way to interact wit other concrete meta-models like those from Wala and Jdt. This is one it's major disadvantage, another being the time and resources need to fully learn the language before using it. Also, at time of writing, there is also the problem of running time. According to [5] the runtime environment is pretty slow.

Soot is java framework developed for code analysis. It directly analyzes jvm bytecode and can represent in several low level representations. Unlinke XCore, which provides support for high analysis tools, the goal of the framework is to provide the necessary tools in order to develope optimization for applications. Also, to the best of my understanding Soot is limited to the analysis of java byte code, whereas XCore can integrate with and language library backend, provided that the library is written in Java (or C and which can be integrated in Java applications).

Fame is similar in scope and idea of development with XCore. It uses the FM3 meta-meta-mode [2] and which is handeled by the JavaFame Tool. The meta-model for the developed tool is described in a msfe in file. By contract, XCore allows de developers to describe the meta-mode directly in the java source by use of annotations. The main disadvantage in using external files in order to provide configurations is the lack of support for refactorings (i.e easy change of the meta-model).

# List of Figures

1.1 Generic Backend for analysis tools [9] . . . . . . . . . . 4

# List of Tables

# Bibliography

[1] S. Alexandru and P. F. Mihancea. Xcore: Framework for software analysis tool development. June 2015.

[2] Stéphane Ducasse, Tudor Gîrba, Adrian Kuhn, and Lukas Renggli. Meta-environment and executable meta-language using smalltalk: an experience report. *Software and Systems Modeling*, 8(1):5–19, 2009.

[3] Bruce Eckel. *Thinking in Java*. Prentice Hall, 2006.

[4] LOOSE Reasearch Group. `http://loose.upt.ro/reengineering/research/codepro`.

[5] Paul Klint, Tijs van der Storm, and Jurgen Vinju. *EASY Meta-programming with Rascal*, pages 222–289. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[6] Patrick Lam, Eric Bodden, Ondrej Lhoták, and Laurie Hendren. The soot framework for java program analysis: a retrospective. 2011.

[7] Cristina Marinescu, Radu Marinescu, Petru Florin Mihancea, and R Wettel. iplasma: An integrated platform for quality assessment of object-oriented design. In *In ICSM (Industrial and Tool Volume)*. Citeseer, 2005.

[8] Ş. Medeleanu and P. F. Mihancea. Nullterminator: Pseudo-automatic refactoring to null object design pattern. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 601–603, Oct 2016.

[9] Alexandru Ştefănică and Petru Florin Mihancea. Xcore: Support for developing program analysis tools. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*, pages 462–466. IEEE, 2017.

[10] IBM Watson. Watson libraries for analysis, wala. sourceforge. net/wiki/index. php. *Main Page*.

[11] J Wiegand et al. Eclipse: A platform for integrating development tools. *IBM Systems Journal*, 43(2):371–383, 2004.