C++ Support in Envision

Bachelor Thesis Description

Lukas Vogel
luvogel@student.ethz.ch

March 12, 2013

Introduction

Envision is a development environment, that represents source code as a combination of visual and textual components. It features a structured editor that can represent source code using graphical notations. This approach is meant to offer several advantages for developers such as easier code comprehension and improved navigation. For example using Envision's ability to display a visual overview of a whole project can be benificial for developers getting familiar with the project.

Envision currently supports a subset of Java. This includes a model for object-oriented programs and flexible visualizations for this model. Furthermore these visual components are already fully interactive. Envision aims to be an IDE supporting the object-oriented paradigm for multiple languages and therefore support for additional languages is desired.

This thesis focuses on extending Envision to support C++ code. More specifically the goal is to import C++ code. Regarding the large amount of features C++ offers, this thesis aims to support only subset thereof. The focus lies on simple C++ object-oriented programs. Some basic preprocessor functionality should also be supported.

To achieve this task the project will use an existing parser framework, namely clang[1]. The framework is nicely structured making it possible to use only parts of it which is suitable for this project. Thanks to a big community clang is kept up to date and supports the newest features of C++. The framework is used in commercial environments and in the FreeBSD operating system. In addition using this framework will also help to make this project more extensible in the future.

[1] http://clang.llvm.org/

Core Tasks

- Get familiar with the clang framework and Envision.
- Add basic C++ code import functionality to Envision, this includes the following tasks:
  - Implement support for simple C++ language constructs such as classes, methods, variables and code which is semantically comparable to Java.
  - Implement support for the following preprocessor directives: #include , #pragma once.
  - Design the plugin/code such that it is easy to extend with more language or preprocessor features.
  - Handle unsupported features such that they result in warnings.
- Create and import a simple C++ program with several classes to show and test the plugin's functionality. This should cover all of the implemented features.

Possible Extensions

- Add support for conditional compilation, this includes finding a suitable tree representation of this inherently textual mechanism.
- Handle simple template constructs which are semantically comparable to Java generics.
- Assure that unsupported code still will be translated in some internal representation to be able to losslessly import and afterwards export code to/from Envision.

Schedule

A tentative time schedule for the project (The week numbers represent the ISO week numbers[2]):

- Week 10: Finish the project description and administrative tasks.
- Week 11: Get familiar with Envision and clang.
- Week 12-14: Implement a plugin which will work with simple C++ code.
- Week 14-15: Take semester break.
- Week 16-21: Finalize plugin to achieve all core tasks.
- Week 22-26: Work on Extensions.
- Week 26-32: Write the report and finalize the project and its extensions.

[2] http://en.wikipedia.org/wiki/ISO_week_date