Methods and Techniques to Analyze (source, intermediate, or target) Code, to extract and handle Application Components

Pedro Rangel Henriques
www.di.uminho.pt/~prh

CCTC – DI / Univ. Minho
Academic Year 2007/08

1 Introduction

This document is a proposal for a three year Ph.D. work in the general areas of *language processing*, *code analysis*, and *code verification*, for the MAP-i students.
The work will be held in University of Minho, Braga, **co-supervised** by:

- Pedro Rangel Henriques (pedrorangelhenriques@gmail.com) – CCTC;
- Jorge Sousa Pinto (jsp@di.uminho.pt) – CCTC.

The proposal will be integrated in the FCT research project Rescue, hence part of the work (mainly the one concerned with program verification and proof carrying code) will run in collaboration with FCUP and UBI.
Moreover, it is expected that this project will be developed in strict collaboration with prof. Árpád Beszédes from the Dep. of Software Engineering of University of Szeged (Hungary).

Keywords

Program Understanding, Program Comprehension, Slicing, Program Analysis, Bad-smells in code, Program Metrics, Program Verification, Proof Car-

rying Code

2 Context

A lot of work has been done in the past in code analysis to help in the various tasks of software re-engineering and maintenance, including program comprehension activities.

Many good results have been reached. However, there is still need for a system that can handle programs at different levels of encoding.

Suppose that we have a lot of application modules just in binary code, and only some of them in the original source language. It would be desirable to analyze and detect application components (or slices) that can be scattered over any of those modules, no matter the format they are available.

Or suppose that we need to analyze all the modules of a multi-language application to identify slices that are traversal to all of them; the most convenient solution would be to compile all of them into a common intermediate representation, and then work them out in this middle-level format.

3 Description and Objective

The aim of this Ph.D. work is to investigate efficient methods, techniques and tools to analyze source (high-level), intermediate (abstract middle-level) or target (low-level/machine) code, under the same environment, in order to extract static and dynamic information that can be visualized in a uniform way (independent of the analysis level) to help in the analysis, comprehension, and verification of a program.

Slicing or similar techniques will be explored to enable the identification and isolation of program components responsible for certain program facets; program metrics will be defined to help characterizing those components.

One main application area for the approach so far drawn, that is intended to be investigated under this work, is the so-called proof-carrying code architecture for program certification. The analysis techniques developed in this work are expected to be particularly useful for *proof compilation* tasks (from proofs of properties at source level to proofs at machine level through intermediate representations), as well as in helping scale up the whole PCC platform, through the use of component-wise verification techniques.