Static Analysis for Industrial Automation Systems

Michael Moser and Rudolf Ramler
Software Analytics and Evolution
Software Competence Center Hagenberg GmbH
Softwarepark 21, A-4232 Hagenberg, Austria
+43 7236 3343 {814, 872}
{michael.moser, rudolf.ramler}@scch.at

ABSTRACT
Static analysis is a highly successful measure when applied in software development. However, there is a lack of corresponding methods and tools in context of industrial automation systems. In this talk we give an overview of different applications of static analysis and the opportunities they provide for software development in industrial automation.

Categories and Subject Descriptors
D.2.3 [Software Engineering]: Coding Tools and Techniques

General Terms
Measurement, Documentation, Verification.

Keywords
Static Analysis, Industrial Automation.

1. INTRODUCTION
What is static analysis? Static analysis works by examining the structure and the elements of a program without actually executing it. It is mainly based on the source code of a program or an intermediate representation thereof [1]. Techniques and tools for static analysis are widely used in the development of software systems, for example to improve the quality of program code or the system design in early development stages. Furthermore, static analysis can also provide many valuable insights about the dynamic characteristics of software systems like performance or reliability, since they often have their causal roots in the program’s static structure. By means of automation, even complex analysis techniques can be effectively applied for software systems with a large code base.

2. APPLICATIONS
How can static analysis be applied? Static analysis techniques form the basis for a wide range of subsequent analysis applications supporting software development.

Defect Detection – Static analysis is a widely used quality assurance measure for detecting a variety of defects, for example, violations of coding conventions and programming rules, array boundary overruns, division by zero, infinite loops, data races, and dead code [2]. A prominent and widely used example for a static analysis approach is bug pattern detection, where structural models of the program code are searched for patterns that typically indicate a defective implementation.

Architectural Conformance Checking – Deviations of the actual implementation from the initially defined architecture throughout a software system’s development and evolution lead to architectural erosion. Continuous automated conformance checking can be an effective counter measure [3]. Thereby, static techniques are used to extract the actual architecture for evaluating it to the reference architecture.

Documentation Generation – Accurate program documentation is indispensable for software maintenance and evolution. However, manually written documentation often becomes outdated when the software is changed. This challenge can be tackled by generating program documentation from source code following the idea of literate programming. For instance, ForDoc [4] generates technical documentation from source code using a minimal set of additional annotations to control the generation process.

Program Comprehension – Static code analysis can be used to facilitate program comprehension in several ways. For instance, we have developed a toolkit [5] to propagate domain concepts as well as physical dimensions to cryptic local and global program variables along the data flow of a program. This approach facilitates comprehension of legacy systems, for which neither documentation nor the original authors are usually available anymore.

3. CONCLUSIONS
We have successfully applied static analysis in context of many software projects, including the analysis and evolution of technical applications and the development of industrial automation systems. Currently we are working on an analysis platform that integrates different static techniques for various programming languages and software technologies.

4. REFERENCES