Enriching the Software Development Process by Formal Methods*

Manfred Broy, Oscar Slotosch
Institut für Informatik, Technische Universität München
80290 München, Germany

Abstract. We describe a software development process designed for an integration and usage of formal methods into practical software process models in a scalable way. Our process model is an extension of the V-model, and allows the specification of critical components and the verification of crucial development steps. For different development stages we suggest user-oriented description techniques, based on a common formal semantic. Furthermore we outline methods for the verification of critical development steps. We illustrate our process by developing a small example with some critical aspects.

1 Introduction

The development of software systems is a difficult and error prone task. This is certainly true if systems get very large and complex. However, this may even be true in cases where small to medium size programs have to be developed that are based on complex algorithms, data structures, or patterns of interaction.

Today software development in practice is almost always done in a non-scientific manner [Hoa96] based on pragmatics and heuristics. The development process is structured into phases like:

- analysis and requirements engineering,
- specification,
- design,
- implementation and testing.

In most cases, large parts of the development work are done informally and the correctness of the developed systems relies mainly on the intuition and experience of the developers received by extensive inspections, testing, and prototyping. The debate is still going on whether there is a cost effective alternative to this heuristic approach to software development including and applying so-called formal methods.

In the early days of computing science there was not even a theoretical alternative. The scientific foundation of programming computer systems were not

* This work was supported by the German Information Security Agency (BSI) within the project Quest.
understood. Programming languages were seen as pragmatic, operational notations to control the machine, used on the basis of an intuitive understanding. For many of the programming concepts a theoretical scientific foundation was not available.

Today the theoretical situation has changed. Over the last 30 years the scientific community of computer scientists has developed a solid foundation of software and systems engineering [BJ95, Bro95]. Today there is nearly no programming construct or development concept for which a scientific basis is not available. Of course, still a lot of notation is used in practice for which such a scientific foundation is not worked out explicitly. But this is not due to the lack of theory but rather it has not been done since those people suggesting the notation to a large extend have not yet recognized the virtue of a scientific foundation. For an improvement of the formal basis of software development, it is therefore highly important to have a clear idea what formal methods are good for and how they are combined with informal ones in a way to get the best benefit out of this combination.

Our paper is organized as follows. After a clarification of the term “formal methods” (Section 2), we describe the role of formal methods in the development process and how formal techniques can be combined pragmatically within formal methods. In Section 3 we start with a short presentation of the V-model, and show how formal methods can be integrated into the development process in a scalable way by using them only for the critical parts and for important development steps. In the rest of this paper we illustrate the development process by means of a small example which is presented in Section 4. Section 5 describes different views of the system during the development process, especially in the requirement, design and implementation phases. Furthermore we present graphical description techniques that can be used in a co-development with formal and conventional methods since all description techniques are based on the same semantic model. In Section 6 we present methods that support the engineer in proving the correctness of development steps between different views, and we analyze tool support for different refinement situations.

2 What is a Formal Method

In spite of the enormous amount of work spent on formal methods the notion of a “formal method” is not always made sufficiently precise [Bro96]. We therefore clarify this to begin with. First of all, let us define what we call a method in computing science.

A method consists of description concepts, rules for constructing and relating descriptions, and development strategies explaining how and when to apply the method in a goal directed manner. We therefore refer in the following to three ingredients of a method:

D: description concepts,
R: rules, and
S: strategies.