Limits of Formal Methods

Ralf Kneuper
Philipp-Röth-Weg 14, 64295 Darmstadt, Germany

Keywords: Formal methods; Software quality; Correctness; Trustworthiness; Formal specification

Abstract. Formal methods can help to increase the correctness and trustworthiness of the software developed. However, they do not solve all the problems of software development. This paper analyses some limitations of formal methods.

1. Introduction

When discussing the possibilities and limitations of formal methods, some people (often the ‘academics’) take either a highly optimistic view, stressing possibilities and ignoring limitations, or (often developers from ‘the real world’) a highly pessimistic view, describing certain limitations of formal methods and deducing that, since formal methods do not solve all our problems, they are useless. However, during the last few years, more people have started to promote a realistic view of the applicability of formal methods (e.g. in [CDH96]).

The main goal of this paper is to support this realistic view of the possibilities and limitations of formal methods (concentrating, as the title suggests, on their limitations, because the readership of this journal will already know all (well, almost) about their possibilities). To some extent, these limitations will seem quite obvious once stated, but in the author’s experience, some parts of the formal methods community still do not fully realize them.

Such personal experience led the author to write the current paper. It stems from working in a formal methods research group for several years (cf. [JIL91, Kne91], where he developed a rather optimistic view concerning the applicability of formal methods. Working first in the quality management group of a major software house and now in the IT department of a large company, he soon had
to appreciate that there is much more to producing 'good' software products than using formal methods. This experience led to the current paper.

1.1. Formal Methods

There are two fundamental views of formal methods as a discipline:

- as a branch of pure mathematics, an intellectually challenging research field which may or may not have any application in the 'real world', or
- as a branch of software engineering which is concerned with the design and application of a certain set of development techniques and tools to create better software systems

Both of these views are legitimate and useful. Problems arise if, for funding and similar reasons, the first view hides behind the second. This paper will be based on the second view since the limitations discussed refer to the practical applicability of formal methods (and, admittedly, this is what the author is most interested in today).

Software engineering is the technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost estimates. [Fai85, p. 2]

As for the term formal methods, we use the following definition:

... a formal method is a set of tools and notations (with a formal semantics) used to specify unambiguously the requirements of a computer system that supports the proof of properties of that specification and proofs of correctness of an eventual implementation with respect to that specification. [HiB95b]

Typical techniques used in formal methods are invariants, proof obligations, and a calculus for refining specifications or proving properties about specifications and implementations, and the relationship between a specification and its implementation.

The emphasis here will be on formal methods concerned with the functional requirements and correctness, e.g. VDM or Z, but there are also formal methods concerned with other quality characteristics, such as for dealing with performance requirements.

Formal methods can be applied at different levels, ranging from 'only' writing formal specification for small parts of a system, via the rigorous approach of expressing but not usually discharging proof obligations, to providing formal proofs of program correctness with respect to the specification, or even proving that the compiler, its environment and the hardware satisfy their specifications (under stated assumptions). Formal specifications are usually considered the most important and most useful aspect of formal methods:

From an economic point of view, therefore, the most important part of a formal development is the system specification. For many projects, this is the only part of the development that is formal. [Hal90, p. 13]

1.2. Goals of Using Formal Methods

As a branch of software engineering, formal methods are concerned with the systematic production and maintenance of software products, on time and within