VSE: formal methods meet industrial needs

Serge Autexier, Dieter Hutter, Bruno Langenstein, Heiko Mantel, Georg Rock, Axel Schairer, Werner Stephan, Roland Vogt, Andreas Wolpers

German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany; E-mail: ⟨last name⟩@dfki.de

Abstract. The Verification Support Environment (VSE) is a tool to formally specify and verify complex systems. It provides the means to structure specifications and supports the development process from the specification of a system to the automatic generation of code. Formal developments following the VSE method are stored and maintained in an administration system that guides the user and maintains a consistent state of development. An integrated deduction system provides proof support for the deduction problems arising during the development process.

We describe the application of VSE to an industrial case study and give an overview of the enhanced VSE system and the VSE methodology.

Key words: Formal software development – Automated theorem proving – Interactive theorem proving – Temporal logic – Modular proof development – Compositionality

1 Introduction

The reliability of complex software systems is becoming increasingly important for technical systems. Malfunctioning of software systems caused by design flaws or faulty implementations may lead to loss or garbling of data, breach of security, danger to life and limb, and, in almost all cases, to severe economic losses. These failures are quite likely to occur if the software has been developed without a general design and programming methodology. Therefore there is a need for a standardised software development process in order to move to an engineering of software based on specific methods and methodologies.

CASE-tools provide automated tool support for software engineering. Since the early 80s, a wide range of these tools has been developed. They provide support for different process activities like specification, design, implementation, and testing. While in the 80s these tools were based on informal specification techniques like SADT or SSADM, the improvement of formal specification and verification technologies promoted the application of strong formal methods within academic and even industrial CASE-tools (see [6, 30] for an introduction to formal methods and associated tools). They allow for a formal comparison of specifications and programs and the proof of properties about them. These tools are usually limited in a sense that either they are only applicable to particular classes of problems (e.g., the restriction to finite state problems allows for the use of model checking as a decision procedures like in SMV [20] or FDR [25]) or that they produce, in general, only partial results (e.g., when using methods based on testing and approximate analysis techniques by abstract interpretations). Other tools, like Specware [14] or METAFrame [29], aim at the composition of reusable specifications and programs.

In contrast to model checking, theorem proving can deal directly with infinite state spaces. A large range of theorem provers like PVS [21], STeP [4], ACL2 [15], or Eves [16] has been used for strong formal methods, but they all lack thorough support in all phases of a formal development methodology.

The VSE-I tool [12], developed on behalf of the German federal agency for IT security (BSI) from 1991 to 1994, provides such assistance and allows for an industrial development of software according to the highest levels of IT security criteria (see ITSEC or CC). Similar to the B-tool [18] it supports a formal development, starting with a formal specification and ending with an automated program code generation from executable specifications. It provides adequate management of software developments, an elaborate proof construction mechanism, and a sophisticated user interface. Unlike the B-tool or the
VSE-I system which only supports the development of sequential programs, the VSE-II system is extended with respect to comprehensive methods in order to deal with distributed and concurrent systems [26]. It also improves VSE-I with respect to an even more efficient and uniform proof support which makes use of the implicit structure of the arising proof obligations. A refined correctness management supports evolutionary software development which allows changes without the need to prove everything from scratch.

After a period of evaluation in various large applications (e.g., a booking system for radio transmissions, an access control system for nuclear power plants, and a security model for digital signatures), since 1997 VSE [12] has been applied in commercial formal developments. Again, the development of VSE-II is accompanied by case studies like the re-development of the safety critical parts of a robot control system (ROBERTINO) working in a fusion reactor [27].

In this paper we focus on the applicability of VSE to industrial size developments and on the novel concepts of VSE-II. The case study (ROBERTINO) is introduced in Sect. 2 which also serves as the running example in this article. The VSE methodology is described in Sect. 3. VSE allows us to combine specifications in various ways. The structure of a specification is represented by a development graph as explained in Sect. 3.1. VSE supports specifications with functional (Sect. 4) as well as with state-based concepts (Sect. 5). For industrial applications it is essential to make use of the structure in a specification when proof obligations are tackled. The improved aspects of the proof engineering process are described in Sect. 6. These improvements include optimizations of the proof search space representation and new developments on the tactical level. Finally, we summarize the contents of this paper in Sect. 7.

2 Case study: a robot control system

In this section, we give a brief introduction to the example used in this paper: the software system used to control a heavy–robotics system in fusion reactors. The hardware and software of this system was developed at the European Community Joint Research Center (JRC) at Ispra, Italy.

2.1 The ROBERTINO facility

The ROBERTINO heavy robotics facility (shown in Fig. 1) is a 4-degrees-of-freedom gantry robot, which will be built for exchanging blanket segments in Thermonuclear Fusion Reactors. These segments are both big (up to 15 m) and heavy (up to 135 tons), requiring special technology for moving them around. In addition, access to the fusion reactor is severely limited, which requires a very exact positioning.

For practical evaluation, a 1/3 scale prototype is used which is able to move payloads of up to 6.5 tons within a range of $2.2 \text{ m} \times 3 \text{ m} \times 6.5 \text{ m}$ and it can move and position the segments with an accuracy of better than 0.1 mm and better than 0.01 deg, respectively.

There are two main modes of operation to be realized in the Robertino Control System (RCS): in the so-called Axes Position Control Mode (APCM), a workstation can send commands with target positions. The RCS software plans a route to these positions, including the selection of acceleration and deceleration, and executes it. In the Direct Axes Velocity Mode (DAVM), robot axes can be selected on a control panel, and the robot can be moved along these axes using joysticks.

2.2 The ROBERTINO Control System – RCS

The Robertino Control System (RCS) is a multi–tasking kernel system responsible for the movements along the robot axes, and the control of these movements. Due to the nature of the tasks of the RCS the software is safety–critical.

The original system, of which the safety critical part is redeveloped in VSE, is implemented in approximately