Verification Support Environment

Frank Koob, Markus Ullmann, Stefan Wittmann

Bundesamt fuer Sicherheit in der Informationstechnik
Godesberger Allee 183
D-53133 Bonn, Germany
Tel.: x49-228-9582154
Fax.: x49-228-9582455
Email: vse@bsi.de

Abstract. Formal methods are recognized as the most promising way to produce high assurance software systems. In reality this fact is not enough to convince industry to use them. Formal methods must be applicable and usable in several areas (security, safety), engineers have to accept a change in software development work but should not be asked to give up the environment they are used to and bosses must realize that higher effort during the design phase can save money and time later. This paper describes the recently completed formal specification and verification tool Verification Support Environment (VSE). An advantage of the design of the VSE tool is the possibility of using formal and semiformal development methods combined in a unique working environment. After official release of the VSE-system March 1995 several pilot projects were carried out with industry. The paper gives an overview of the VSE-system and describes the results of the pilot applications.

1 General

This paper briefly describes the functionality of the Verification Support Environmet System (VSE), its integration into the high assurance development process and an outline of the result of an industrial pilot project recently carried out.

In 1991, the Bundesamt fuer Sicherheit in der Informationstechnik (German Information Security Agency) initiated a CASE tool project with emphasis on formal specification and verification. Now, after four years of work, the Verification Support Environment (VSE) Tool has attained a status that permits industrial application. This tool combines the two traditions of semiformal and formal methods within a unique framework. Therefore the benefit of the VSE tool is both to support migration to formal methods where needed and, where it is sufficient, to rely on semiformal methods ([1]).

2 Components of VSE

1. specification language VSE-SL
   - formal language describing abstract data types and abstract state transiton machines, using terms of First-Order Predicate Logic with Equality and Dynamic Logic
   - syntax controlled editor, consistency and type checking
2. graphical interactive development presentation
3. import/export functions
4. dual interactive verifier subsystems
   - Dynamic Logic: KIV (Karlsruhe Interactive Verifier)
   - Predicate Logic: INKA (INduction prover KArlsruhe)
   - predifined proof strategies, tactics, heuristics
5. verification management system and database (multi-use capability)
6. standard design interface (ODS) to conventional CASE-tools

3 Functionality

The VSE-system supports the software development process from analysis to code generation. During analysis it has to be determined which parts of the future system are security (safety) critical. The non-critical parts can be developed conventionally within the regular production environment. For the critical parts VSE provides a specification language (VSE-SL) to structure them in a way to support later proof activities. The top level specification formally describes the functionality on an abstract level. The security (safety) model defines characteristics of the objects that have to be fulfilled. With means of refinement the top level specification is modified stepwise to abstract programs and, using the code generator, ADA sourcecode.

The VSE prover (verifier) subsystem automatically generates proof obligations out of the specification and the refinement process.

- The top level specification fulfills the security (safety) model.
- The entire refinement process guarantees that the generated code has the same functionality as the top level specification.

All proof obligations have to be verified in order to be able to call the source code correct in accordance to specification and security (safety) model. The functionality of the deduction subsystems includes the following features ([2] and [3]):

- semi-automatic switch between Dynamic Logic (KIV) and Predicate Logic (INKA)

- provision of proof tactics and empirically predifined heuristics running automatically
- provision of pre-selected deduction rules applicable to the current proof goal in an interactive mode
- proof protocol (text and graphics), replay mode, restart at any proof step

4 Applications

Two major case studies (disposition control and system and nuclear power plant access control system) were carried out within the VSE-project to show that VSE is applicable even on large projects (4100 verified lines of code, 20000 proof steps, average automation rate 80 was used by eight industry companies and government institutions from Germany and Italy for pilot applications dealing with traffic control (railway), space flight, smart cards, hospital administration and secure message handling. The pilot application presented is a drug administration component as a part of a hospital administration system. This part is based on SEMA.

The product Health System 2000 (HS2000) was developed by SEMA Group Ismaning, Germany. It is a hospital information system to administrate patient files, medical means and accounting of a hospital. The subject of the pilot project was an additional part to the existing HS2000 to control and check access to and applicability of drugs in a hospital pharmacy. The security-critical kernel was to be developed with the VSE-system. From April 20, 1995 until June 1, 1995 one expert from the University of Ulm and one company representative worked on the project. The task of the VSE-expert was to give a basic training in the VSE development method and the VSE-tool and to support the company representative only as much as needed.

5 Conclusion

The pilot projects reached almost all the goals that had been defined before the start. Concerning the size of the problems and time/personal constraints the results were surprising. This pilot project showed that formal methods are applicable in industrial environments and they significantly improve the quality of software systems. Facts like less misinterpretations and incompatibilities in the requirements and the reusability of verified components strengthen the trustworthiness of the systems developed with the formal VSE method. Besides that re-specifications helped to detect errors in existing systems and showed the limitations of conventional software development.

Nevertheless six weeks of work with formal methods and the VSE-tool are not enough to be prepared for independent formal development. There is still a need for intensive training and support of experts to handle the specification language and the verifier subsystems. But the cooperation of software engineers and VSE-specialists turned out to be a very promising way to introduce formal

methods to industry and to transfer the new technology. New fields of industrial services like formal design or proof engineering do not belong to science fiction.

All partners realized that formal methods solve a lot of problems in the area of critical software development. A mathematically based engineering including verification, all supported by a tool that minimizes the practical work showes the way to a new dimension of software quality. But especially non-governmental partners cannot ignore the rules of the market. Formal development, even using VSE, takes considerably more effort. The result might be the better product but it also might be too late and significantly more expensive than conventionally developed products. The solution to this problem still has to be worked out. Software applications that have to fulfill high security/safety standards (e.g. avionic systems with the safety level 'catastrophe') already need an enormous amount of quality assurance measures like tests, code inspections and simulation). Both additional efforts for formal methods and conventional QA-activities should be evaluated and compared. There was no time to do that during the pilot projects but the results might be interesting. Not only the producers of high assurance software have to be convinced that formal methods are applicable, improve the quality and save money on testing and warranty acticities; potential clients have to realize that it is worth to spend more money on a product developed with means that guarantee high reliability.

VSE Version 1 is the first successful step to open the market for formal methods in software development. Nevertheless the pilot projects showed that VSE has to be modified and improved: better means of structurizing, batch mode for proofs, better integration of the verifiers, applicability on embedded, reactive systems, code generation C(++), interfaces to Z and model checkers, extension of the interface to the CASE-tool TEAMWORK, etc. Major modifications and improvements will be realized in a follow-on project VSE-II starting summer 1996. The Bundesamt fuer Sicherheit in der Informationstechnik will continue to keep the public informed about changes, experiences and new developments.

References

1. Koob, F., Ullmann, M., Wittmann, S.: The Formal VSE Development Method - A Way to Engineer High-Assurance Software Systems. Eleventh Annual of the COMPUTER SECURITY APPLICATIONS Conference (1995) 196–204
2. Reif, W., Schellhorn, G., Stenzel, K.: Interactive Correctness Proofs for Software Modules Using KIV. Proceedings of the Tenth Annual Conference on Computer Assurance (1995) 151–162
3. Hutter, D. et al: Deduction in the Verification Support Environment (VSE). Springer LNCS **1051** (1996) 268–286