

# Overview over the Project



Oscar Slotosch

Institut für Informatik, Technische Universität München  
80290 München, Germany

**Abstract.** The software development project Quest provides tools to build correct software, especially for embedded systems. It connects the CASE-tool AUTOFOCUS to the formal development tool VSE II, and the model checker SMV. To increase quality of non-formally developed programs a test environment is realized, containing a connection to the test-case classification tool CTE.

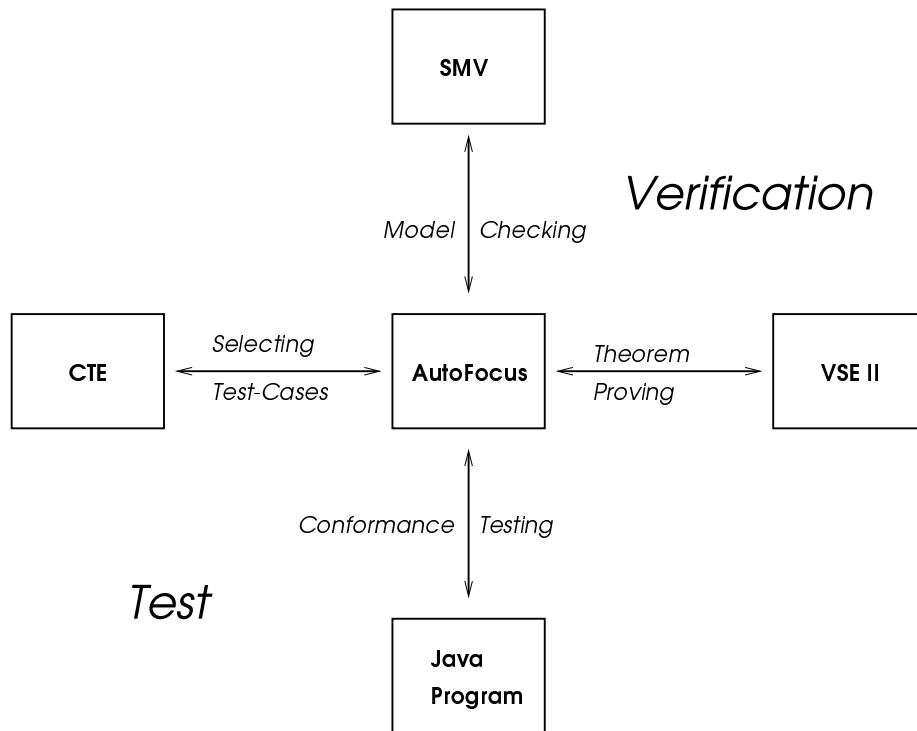
Proving the correctness of the emergency closing system of a storm surge barrier is a running example within the project Quest.

## 1 Structure of Quest

The project Quest is carried out for the German “Bundesamt für Sicherheit in der Informationstechnik” (BSI). One of the main tasks of the BSI is to increase the assurance and trustworthiness of security and safety critical systems. Therefore the aim of the project Quest is to enrich the practical software development process by the coupling existing methods and tools using formal techniques in order to ensure the correctness of critical parts of systems. For the development of large systems a scalable concept is required. To achieve this goal we choose hierarchical, graphical description techniques (supported by CASE-tools) and a compositional logic with theorem prover support. For small systems model checkers can be used to verify critical properties automatically.

The combination with traditional software engineering methods is achieved by specification-based test methods to generate test-cases for non-critical components. The tools developed within Quest translate the graphical concepts into a logical formalism and systematically generate test-cases. To support an integrated, iterative development process a graphical visualization of some formulas is planned by retranslating them to the CASE-tool.

The main phases of the project Quest are: the analysis phase, where the tools AUTOFOCUS (CASE-tool), VSE II (theorem prover), SMV (model checker) and CTE (test case assistant) have been selected, the design phases, where the translation concepts have been fixed, and the implementation phase where the translators are realized and integrated into the tools. Quest started out at 1.10.1997 and will end at 30.9.1999. The project Quest and also this paper are structured into four parts: the connection from AUTOFOCUS to VSE II, the connection from



**Fig. 1.** Tool Structure of Quest

AUTOFOCUS to SMV, the test environment and the case study. An integration of the Quest tools into the software development process using the V-model is described in [BS99].

Figure 1 shows the structure of the tools of the project Quest. The boxes represent the selected tools, and the arrows between them denote the connections that are designed and implemented within the project Quest. We integrate the tools by defining interfaces and (sometimes partial) translations between the used concepts. The interfaces could also be used by other tools. This paper shortly describes the connections between the tools, and the case study that is developed using tools.

The project is carried out at the Technical University of Munich (Group of Prof. Broy), together with the DFKI (German Research Center for AI), Daimler Benz Research, and ist (innovative software technologie). The following persons are contributing with the author to the success of Quest: F. Koob, M. Ullmann, S. Wittmann (BSI), W. Stephan, A. Wolpers, G. Rock, H. Mantel (DFKI), P. Baur, T. Plasa, M. Popall (ist), T. Klein, S. Sadeghipour (DB), M. Broy, S. Merz, J. Philipps, O. Müller, I. Krüger, H. Lötzbeyer, P. Braun, R. Sandner, P. Gierl, T. Sprenger, G. Wimmel, and D. Chibisov (TUM).

## 2 The Connection between AUTOFOCUS and VSE II

The CASE-tool AUTOFOCUS [HMR<sup>+</sup>98,HMS<sup>+</sup>98] has been designed for the development of correct embedded systems. It bases on the formal concepts of FOCUS [BDD<sup>+</sup>92]. AUTOFOCUS offers graphical description techniques for structure, behaviour, and interaction. The descriptions use functional data types, and allow pattern matching in the definition of functions and in the transitions of the behavioural view. All views can be structured hierarchically, and consistency of the descriptions can be checked. AUTOFOCUS has a synchronous simulation<sup>1</sup> to validate the specifications by rapid prototyping. AUTOFOCUS is extended within the project Quest to store properties of components.

The VSE II system [RSW97] is an extension of the VSE [HLS<sup>+</sup>96] system by concepts of TLA, hence it has asynchronous, action oriented semantics. VSE II uses, like AUTOFOCUS, functional data types. To define correct translations from AUTOFOCUS components to VSE II a scheduler is introduced within VSE II to enforce synchronization of the components. Since AUTOFOCUS components can be hierarchic, the translation introduces one scheduler for every level of abstraction. The communication channels between components are modelled by combining the input and output channels of the components. This translation is also applied to properties of specifications that are translated to VSE II.

There is also a partial retranslation from VSE II to AUTOFOCUS. It inverts the above translation and can be used to visualize structures of VSE II specifications. Furthermore changes made within VSE II can be incorporated into the original AUTOFOCUS model of the system.

This connection between the CASE-tool AUTOFOCUS and VSE II supports a user-friendly description of systems and allows to prove arbitrary properties using the general theorem prover of VSE II.

## 3 The Connection between AUTOFOCUS and SMV

We chose the SMV system [McM92] because it offers an efficient checking procedure for synchronous models. Therefore no scheduler is required. The translation to SMV generates a module for every component and combines the modules using variables for message channels. This translation is quite natural, however since SMV has neither functions (except boolean and integer arithmetic) nor functional data types integrated it is necessary to eliminate these data types during the translation from AUTOFOCUS to SMV. Of course the static elimination works only for finite data types.

The retranslation from SMV to AUTOFOCUS, depends on the model checking result. In the case the property is valid, this information is passed to AUTOFOCUS. In the case SMV produces a counterexample, this is translated into an

---

<sup>1</sup> The simple (not object-oriented) component semantics of AUTOFOCUS and the synchronous execution model characterize the main application domain of AUTOFOCUS: embedded systems.

event trace that shows the interaction between the components that leads to a contradiction of the desired property.

Furthermore abstraction techniques are integrated to check properties of large systems by reducing systems to simpler ones that can be checked. The correctness of this simplifications has to be proved (for every property, and every abstraction) using the VSE II system.

## 4 Test Environment

The test environment is part of the project Quest, because testing is important to increase quality of software, especially if the development is not completely formal. Extended Event Traces (EETs) are a graphical notation of AUTOFOCUS to visualize interactions of components with other components and the environment. EETs are used to represent test-cases.

The test environment computes a “transition tour” i.e. a minimal sequence of transitions that covers all possible transitions of the state transition diagram. If the behaviour description contains no variables, the input/output sequences for the test run can be generated automatically. Otherwise the user has to select appropriate input values for the variables of the transitions to construct the input/output sequences for the test run. The CTE tool [GWG95] supports the user in managing classifications of input variables, basing on the types of the variables.

For the execution of test runs a driver is implemented that passes the values of the EET to the Java program under test, and compares the results with the output values of the EET.

The test environment of Quest supports a systematic, specification-based test generation and execution. Furthermore a graphical notation for test-cases is given.

## 5 Case Study: Emergency Closing System

To demonstrate the benefits of Quest for the practical development of safety critical systems, a real-world case study has been selected. The case study is carried out together with the Dutch SIMTECH company, that is currently designing the system. For this project a high level of integrity is required that can only be reached by a formal correctness proof.

The task is to correctly develop a realization for the emerging closing system of the storm surge barrier in the Eastern Scheld, that prevents the Netherlands from floodings like 1953. The system has six sensors for the inside and outside water levels and determines if the barrier has to be closed, and if it can be reopened. The correctness of the open- and close-signals has to be verified formally.

The informal specification is textual with some graphical descriptions of the systems structure and behaviour. The current state of the case study is that a

complete formal model has been built using AUTOFOCUS and the translation into the SMV systems resulted into a checkable system. Verification and the generation of tests is ongoing work.

## 6 Conclusion

The project Quest integrates formal methods and practical software development into a framework, that allows to develop correct software for embedded systems. Instead of general software development (like UML), we focus on embedded systems, because the formal foundations for this application domain are ready for practical applications.

Within the Quest framework it will be possible to develop correct software and to certify a high level of integrity for the developed systems.

For comments on previous versions of this paper we thank Heiko Lötzbeyer, Robert Sandner, and Stefan Wittmann.

## References

- [BDD<sup>+</sup>92] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T.F. Gritzner, and R. Weber. The design of distributed systems - an introduction to FOCUS. Technical Report TUM-I9203, Technische Universität München, Januar 1992.
- [BS99] M. Broy and O. Slotosch. Enriching the Software Development Process by Formal Methods. 1999. this volume.
- [GWG95] M. Grochtmann, J. Wegner, and K. Grimm. Test Case Design Using Classification Trees and the Classification-Tree Editor. In *Proceedings of 8th International Quality Week, San Francisco*, pages Paper 4-A-4, May 30-June 2 1995.
- [HLS<sup>+</sup>96] D. Hutter, B. Langenstein, C. Sengler, J. Siekmann, W. Stephan, and A. Wolpers. Deduction in the Verification Support Environment(VSE). In Marie-Claude Gaudel and James Woodcock, editors, *Proceedings Formal Methods Europe 1996: Industrial Benefits and Advances in Formal Methods*. Springer-Verlag, 1996.
- [HMR<sup>+</sup>98] F. Huber, S. Molterer, A. Rausch, B. Schätz, M. Sihling, and O. Slotosch. Tool supported Specification and Simulation of Distributed Systems. In B. Krämer, N. Uchihira, P. Croll, and S. Russo, editors, *Proceedings International Symposium on Software Engineering for Parallel and Distributed Systems*, pp. 155-164, ISBN 0-8186-8467-4, pages 155–164. IEEE Computer Society, Los Alamitos, California, 1998.
- [HMS<sup>+</sup>98] F. Huber, S. Molterer, B. Schätz, O. Slotosch, and A. Vilbig. Traffic Lights - An AUTOFOCUS Case Study. In *1998 International Conference on Application of Concurrency to System Design*, pages 282–294. IEEE Computer Society, 1998.
- [McM92] K.L. McMillan. The SMV system, Symbolic Model Checking - an approach. Technical Report CMU-CS-92-131, Carnegie Mellon University, 1992.
- [RSW97] G. Rock, W. Stephan, and A. Wolpers. Tool Support for the Compositional Development of Distributed Systems. In *Proc. Formale Beschreibungstechniken für verteilte Systeme, GI/ITG-Fachgespräch*. GMD-Studien Nr. 315, ISBN: 3-88457-514-2, 1997.