Modelling and Validation: AUTOFOCUS and Quest

Oscar Slotosch

Technical University of Munich, Germany.

Abstract. We describe the graphical description techniques, their underlying semantics and the validation techniques that have been applied to the FM99 example; furthermore we sketch the model and discuss the validation results we have obtained. AUTOFOCUS is the modelling tool, and the Quest environment connects the model to a broad spectrum of validation tools, including model checkers (SMV, SATO), theorem provers (VSE) and test tools (CTE). The original contribution, further publications, and AUTOFOCUS can be downloaded from <http://autofocus.in.tum.de>.

Keywords: Event traces; Interactive theorem proving; Model checking; State transition diagrams

1. Modelling Technique

AUTOFOCUS [HMR98] offers a comprehensive set of graphical notations to specify distributed systems from different points of view. All description techniques are hierarchic and refer to a common system model. *System structure diagrams (SSDs)* describe the components of the system, their interfaces and their connections. Components can be refined by assigning a substructure (again given as an SSD) to them; they can also refer to a behaviour description in the form of a state transition diagram (see below). Functional *data type definitions (DTDs)* describe the types, values and functions the model is based on. Pattern matching can be used to define functions. The behaviour of components is described using *state transition diagrams (STDs)*, refer to the interface of components and operate on the data specified in the DTDs. Transitions consist of input patterns, preconditions, output patterns and actions (to update local variables). Interactions between components of the system are described using *extended event traces (EETs)*. EETs show interaction sequences of components. EETs are used to capture requirements, to document simulation runs, to represent test cases, and to visualise counter-examples obtained from model checking.

The formal semantic basis of the descriptions is a mathematical model consisting of time-synchronously operating components with a single element buffer for communication. Thus neither micro steps nor instantaneous feedback are needed. This simple semantics is well suited for the verification task and forms a clear basis for the software engineering process.

2. Model of the Banking System

Our model for the banking system comprises two tills that are connected to a central banking system via insecure communication channels. The central system is modelled with two drivers for the channels and a database, consisting of two parallel processes to read data from the clients' accounts. The DTDs contain descriptions of the data structure for credit cards. Cards store the pin code, the account number, the date of the last withdrawal and the amount that has been withdrawn on this day. Furthermore, types for the interactions between users and tills and messages to describe transactions are defined. The tills gather the required information, and send it in the form of a composite transaction, to the central. If the connections are down sending is repeated until an acknowledge signal is received. The decision whether the client can withdraw money or not is modelled non-deterministically and can be refined later. The STDs of the till, the connections and the processes are reused for both tills, connections and processes, to show that the model can easily be extended with additional tills.

3. Validation Techniques

Several validation techniques can be applied to the model. *Consistency checks* [HSE97, BLS00] allow us to express static consistency of the different views. This enables detection of modelling errors like type errors, or inconsistencies between different abstraction levels and different views. The graphical *simulation* allows us to validate the systems behaviour. Consistency checks and simulation are the main features to improve the adequacy of the (formal) model. A simple temporal logic is used to describe safety-critical system properties. These properties can be *model checked* using SMV [McM92] and *bounded model checked* using SATO [Zha97] (see [PhS99]). For larger or infinite systems *abstraction techniques* can be applied to reduce complexity [Mul98, CGL92]. For each property a proof obligation is generated ensuring that the property is true in the concrete system, provided it holds in the abstract system. These proof obligations and other properties can be verified using the *interactive theorem prover* VSE (verification support environment) [HMR99]. Several *specification-based test case generation* methods can be applied to test implementations of the system or a hardware realisation. Some of those methods are: classification of variables using the classification tree editor CTE [GrG93], a transition tour that covers all transitions of an STD, and systematic approaches to testing the combination of several units [WLP00].

4. Validation Results

We corrected several formalisation and modelling errors that have been found using consistency checks and simulation. We model checked the property that the driver drives the connection correctly. Applying bounded model checking, we proved that invalid cards are rejected. The complete model was checked abstracting from the data types. Counter-examples to negated properties were used as test cases, for example to show how money can be withdrawn.

Acknowledgements

We thank TU Munich, and German BSI for financial support, and the whole development team, especially Peter Braun, Franz Huber, Heiko Lötzbeyer, Alexander Pretschner, Guido Wimmel and Bernhard Schätz. For comments on this paper we thank Ingolf Krüger, Robert Sandner and Bernhard Schätz.

References

- [BLS00] Braun, P., Lötzbeyer, H., Schätz, B. and Slotosch, O.: Consistent integration of formal methods. In *Proc. 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'00)*, 2000.
- [CGL92] Clarke, E. M., Grumberg, O. and Long, D. E.: Model checking and abstraction. In *Proceedings of the 19th ACM Symposium Principles of Programming Languages*. ACM Press, New York, 1992, pp. 343–354.
- [GrG93] Grochtmann, M. and Grimm, K.: Classification trees for partition testing. *Software Testing, Verification, and Reliability*, 3:63–82, 1993.

- [HMR98] Huber, F., Molterer, S., Rausch, A., Schätz, B., Sihling, M. and Slotosch, O.: 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*. IEEE, 1998, pp. 155–164.
- [HSE97] Huber, F., Schätz, B. and Einert, G.: Consistent Graphical Specification of Distributed Systems. In J. Fitzgerald, C. B. Jones, and P. Lucas, editors, *Industrial Applications and Strengthened Foundations of Formal Methods (FME'97)*, LNCS 1313. Springer Verlag, 1997, pp. 122–141.
- [HMR99] Hutter, D., Mantel, H., Rock, G., Stephan, W., Wolpers, A., Balser, M., Reif, W., Schellhorn, G. and Stenzel, K.: VSE: Controlling the Complexity in Formal Software Development. In D. Hutter, W. Stephan, P. Traverso and M. Ullmann, editors, *Proceedings Current Trends in Applied Formal Methods, FM-Trends 98*, Boppard, Germany, 1999. Springer-Verlag, LNCS 1641, pp. 351–358.
- [McM92] McMillan, K. L.: *The SMV system, Symbolic Model Checking: an approach*. Technical Report CMU-CS-92-131, Carnegie Mellon University, 1992.
- [Mul98] Müller, O.: *A Verification Environment for I/O-Automata Based on Formalized Meta-Theory*. PhD thesis, Institut für Informatik, Techn. Univ. München, 1998.
- [PhS99] Philipps, J. and Slotosch, O.: The quest for correct systems: Model checking of diagrams and datatypes. In *Proceedings IEEE Asian Pacific Software Engineering Conference (APSEC'99)*, 1999, pp. 449–458.
- [Slo98] Slotosch, O.: Quest: Overview over the Project. In D. Hutter, W. Stephan, P. Traverso and M. Ullmann, editors, *Applied Formal Methods (FM-Trends 98)*. Springer LNCS 1641, 1998, pp. 346–350.
- [WLP00] Wimmel, G., Lötzbeyer, H., Pretschner, A. and Slotosch, O.: Specification Based Test Sequence Generation with Propositional Logic, 2000. Submitted to Software Testing, Verification & Reliability (STVR): Special Issue on Specification Based Testing (submitted).
- [Zha97] Zhang, H.: SATO: An e-client propositional prover. In W. McCune, editor, *Proceedings of the 14th International Conference on Automated deduction*, (vol. 1249 of LNAI), Berlin, July 13–17 1997. Springer, pp. 272–275.

Received June 2000

Accepted in revised form December 2000