Introduction to Subject Area “Verification”

Frank Ortmeier, Wolfgang Reif, and Gerhard Schellhorn

Lehrstuhl für Softwaretechnik und Programmiersprachen,
Universität Augsburg, D-86135 Augsburg
{ortmeier, reif, schellhorn}@informatik.uni-augsburg.de

Over the last two decades the use of software in technical applications has dramatically increased. Almost all real-world systems are now embedded systems consisting of hardware and software components. Just think of modern automobiles; every new car comes equipped with computers that have many tasks in almost all parts of the car: fuel injection rates of the engine, airbags, anti-blocking systems (ABS) for brakes or the anti-theft device are some examples.

With the use of software the complexity of such systems and therefore the risks associated with failures have increased too. Failure of the ABS can result in bad injuries or even fatalities. But cars are only the tip of the iceberg. There are far more critical embedded systems in our environment like air planes, high speed railways or nuclear power plants. Such systems must not fail.

To assure safety thorough analysis is required. This can be done by the integration of techniques of modern software specification with engineering techniques - the topic of this volume and the DFG focus area program 1064. In this part we will cover one such technique in particular: verification.

Verification allows to rigorously prove that a certain property holds for a formal system model. Application of verification on critical embedded systems is the strongest analysis techniques available to ensure safety. However the task of formally analyzing a complex embedded system requires some effort and specialized techniques. This task can be split in three parts:

1. Building a formal model of the system
2. Identifying the important safety properties
3. Verification of the properties for the given system

The first step towards applying any verification technique is to build a formal model of the system. There exists a wide variety of possible modeling languages. Generic languages are based on algebraic (e.g. CASL [1], CSP [2]), model-oriented (e.g. VDM [3], Z [4], B [5]) or state-based (e.g. Statecharts [6], SDL [7], TLA [8], ASM [9], hybrid automata [10]) approaches, and there is an abundance of problem-specific languages too. Which one is the best, depends on the characteristics of the problem at hand. Are the data structures more important or are dynamic sequences of actions and reactions the central part of the system? Is a continuous time model necessary or can all aspects be modeled in discrete time? Such questions help in choosing the most appropriate modeling language. Typically, a specification language provides adequate support and intuitive models only for a few of these aspects. But often industrial systems have many facets, so several models are required. Therefore one of the important research tasks in supporting formal system specification and verification is to provide approaches that correlate or integrate different views of systems. One, but not the only attempt at integration is to define a formal semantics for the different models of UML [11]. The invited paper [12] by *D. Bjorner*, *A. Haxthausen*, *C.W. George*, *C.K. Madsen*, *S. Holmslykke* and *M. Pěnička* discusses this and gives other approaches for the domain of railway systems: The RAISE specification language (RSL) and its integration with UML class diagrams, Petri Nets, Live Sequence Charts and State charts is considered.

Integrating specification formalisms is also the topic of the contribution [13] of *M. Kardos* and *F.-J. Rammig* of the *ISILEIT* group. Their approach is targeted towards software for distributed production control systems. SDL is used to model component structure and signal flow, while UML state machines describe the individual components. An integrated semantics of both specification languages is defined in terms of AsmL models. These also serve as the basis for model checking to verify system properties.

The second step is to determine and formalize the safety properties the system is expected to fulfill. Determining safety properties is not an easy task and safety analysis techniques like FTA [14], FMEA [15], Hazop [16] from engineering can help to find them. Combining safety analysis techniques from engineering with a systematic process of formal specification and verification is the aim of the *ForMoSA* project (*F. Ortmeier*, *G. Schellhorn*, *A.Thums* and *W. Reif*). Their approach [17] starts with failure-sensitive specification to detect hazards. Fault tree analysis (FTA) is then used to analyze the hazards. In combination with a formal model, FTA can be given a formal semantics that can be used to verify the completeness of FTA as well as to validate the formal model. FTA is used for qualitative as well as quantitative analysis of the model, and combined with safety optimization. The approach is evaluated using the industrial case study "height control in the Elbtunnel".

Describing safety relevant properties of specifications with the graphical formalism of live sequent charts is discussed in the paper of the *USE* group (*M. Brill*, *W. Damm*, *J. Klose* and *B. Westphal*). The contribution [18] shows how to integrate these into the development process of the widely used V-model, and how to provide effective verification support. The results are exemplified using the reference case study "Funkfahrbetrieb".

After accomplishing the first two steps, the third step is clear. Prove the properties of step two for the model obtained at step one. Two possibilities exist to prove a property: interactive verification with tools like Isabelle [19], PVS [20], ACL2 [21], KIV [22] and model checking, using tools such as SMV [23], Spin [24], Uppaal [25] or the verification environment of statemate [26]. Interactive verification is more powerful, but depends heavily on interaction with a human expert. The big advantage of model checking is that proofs are fully automatic. However the technique is limited to finite-state systems.

Development of efficient verification systems is a complex problem in itself. One solution is to use existing tools, and to use a translation of the domain-specific specification language to the language of the verification tool. This approach was used in the *SFC-Check* project by *N. Bauer*, *R. Huuck*, *S. Lohmann*, *B. Lukoschus*, *O. Stursberg* and *S. Engell*. Their contributrion [27] is concerned with the development of model checking support for programmable logic controllers (PLCs), which are in widespread industrial use. They present a systematic approach to develop PLC programs from specifications using sequential function charts (SFCs). At the core of their approach are translations of SFC models to automata as used in the model checking tools Cadence SMV and Uppaal that allow to prove safety properties of PLC programs formally.

Sometimes using existing tools is too inefficient. Therefore the group *J. Ruf*, *R. J. Weiss*, *T. Kropf* and *W. Rosenstiel* developed an efficient real-time model checker, called RAVEN, themselves. RAVEN's architecture, specification language and logic CCTL are described in the final contribution [28]. They use simulation techniques to reveal critical states which are then used in model checking. This enables them to analyse larger designs at the cost of reduced coverage. The reference case study of a holonic production system is used to demonstrate the approach.

Summarizing part "verification" of this volume we think that the papers of this part are an important contribution towards a better understanding of the task involved in formal specification and verification of embedded systems. Within the focus area program formalizations for several application domains have been developed. New techniques to state safety properties intuitively and to do safety analysis on a formal basis have been provided, as well as proof support has been implemented. This results in intuitive modeling techniques, better understandable formal analysis methods and easier proofs.

References

[1] CoFI (The Common Framework Initiative): CASL Reference Manual. LNCS 2960 (IFIP Series). Springer (2004)

[2] Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall (1985)

[3] Jones, C.B.: Systematic Software Development using VDM. 2nd edn. Prentice Hall (1990)

[4] Spivey, J.M.: The Z Notation: A Reference Manual. 2nd edn. Prentice Hall International Series in Computer Science (1992)

[5] Abrial, J.R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press (1996)

[6] Harel, D.: Statecharts: A visual formalism for complex systems. Science of Computer Programming **8** (1987) 231–274

[7] International Telecommunications Union (ITU): ITU-T Recommendation Z.100, Specification and Description Language (SDL). (2002) available at HTTP://WWW.SDL-FORUM.ORG.

[8] Lamport, L.: The temporal logic of actions. ACM Transactions on Programming Languages and Systems **16** (1994)

[9] Gurevich, M.: Evolving algebras 1993: Lipari guide. In Börger, E., ed.: Specification and Validation Methods. Oxford University Press (1995) 9 – 36

[10] Henzinger, T.: The theory of hybrid automata. In: Proceedings of the 11th LICS, IEEE Comp. Soc. Press (1996) 278 – 292

[11] The Object Management Group (OMG): OMG Unified Modeling Language Specification Version 1.5. (2003) available at HTTP://WWW.OMG.ORG/TECHNOLOGY/DOCUMENTS/FORMAL/UML.HTM.

[12] Bjørner, D., George, C.W., Haxthausen, A.E., Madsen, C.K., Holmslykke, S., Pěnička, M.: "UML–ising" formal techniques. In: this volume. Springer LNCS (2004)

[13] Kardos, M., Rammig, F.J.: Model based verification of distributed production control systems. In: this volume. Springer LNCS (2004)

[14] Vesley, W., Dugan, J., Fragole, J., II, J.M., Railsback, J.: Fault Tree Handbook with Aerospace Applications. NASA Office of Safety and Mission Assurance, NASA Headquarters, Washington DC 20546. (2002)

[15] Reifer, D.: Software failure modes and effects analysis. IEEE Transactions on Reliability **28** (1979) 147 – 249

[16] Fenelon, P., McDermid, J., Nicholson, A., Pumfrey, D.: Experience with the application of HAZOP to computer-based systems. In: Proceedings of the 10th Annual Conference on Computer Assurance, Gaithersburg, MD, IEEE (1995)

[17] Ortmeier, F., Thums, A., Schellhorn, G., W.Reif: Combining formal methods and safety analysis - the for mossa approach. In: this volume. Springer LNCS (2004)

[18] Brill, M., Buschermöhle, R., Damm, W., Klose, J., Westphal, B., Wittke, H.: Formal verification of LSCs in the development process. In: this volume. Springer LNCS (2004)

[19] Paulson, L.C.: Isabelle: A Generic Theorem Prover. LNCS 828. Springer (1994)

[20] Owre, S., Rushby, J.M., Shankar, N.: PVS: A Prototype Verification System. In Kapur, D., ed.: Automated Deduction - CADE-11. Proceedings. LNAI 607, Berlin, Saratoga Springs, NY, USA, Springer (1992)

[21] Kaufmann, M., Moore, J.: An industrial strength theorem prover for a logic based on common lisp. IEEE Transactions on Software Engineering **23** (1997)

[22] Thums, A., Schellhorn, G., Ortmeier, F., W.Reif: Interactive verification of statecharts. In: this volume. Springer LNCS (2004)

[23] McMillan, K.L.: Symbolic Model Checking. Kluwer Academic Publishers (1990)

[24] Holzmann, G., Holzmann, G.: The Spin Model Checker: Primer and Reference Manual. Addison Wesley (2003)

[25] Amnell, T., Behrmann, G., Bengtsson, J., D'Argenio, P.R., David, A., Fehnker, A., Hune, T., Jeannet, B., Larsen, K.G., Möller, M.O., Pettersson, P., Weise, C., Yi, W.: UPPAAL - Now, Next, and Future. In Cassez, F., Jard, C., Rozoy, B., Ryan, M., eds.: Modelling and Verification of Parallel Processes. Number 2067 in Lecture Notes in Computer Science Tutorial, Springer–Verlag (2001) 100–125

[26] Bienmöller, T., Damm, W., Wittke, H.: The STATEMATE verification environment – making it real. In Emerson, E.A., Sistla, A.P., eds.: CAV'00: 12th international Conference on Computer Aided Verification. Number 1855 in LNCS, Chicago, IL, USA, Springer (2000) 561–567

[27] Bauer, N., Engell, S., Huuck, R., Lohmann, S., Lukoschus, B., Remelhe, M., Stursberg, O.: Verification of PLC programs given as sequential function charts. In: this volume. Springer LNCS (2004)

[28] Ruf, J., Weiss, R.J., Kropf, T., Rosenstiel, W.: Modeling and formal verification of production automation systems. In: this volume. Springer LNCS (2004)