Research Project

Temporary Title

Automated support for modeling software systems: model synthesis

Context and State of the Art

Models are increasingly recognized as an effective means for elaborating requirements and exploring software designs. Model-driven elaboration and analysis of requirements and designs call for rich system models that cover the intentional, structural, and behavioral dimensions of the system [Lam00a]. Goals, scenarios, and state machines form a golden triangle for modeling systems along those dimensions.

- *Goals* are prescriptive statements of intent whose satisfaction requires cooperation among the agents forming the system. Goal models are AND/OR graphs that capture how goals contribute to each other. Such models support various forms of early, declarative, and incremental reasoning for, e.g., goal refinement and completeness checking, conflict management, hazard analysis, threat analysis, requirements document generation, and so forth [Lam04]. On the down side, goals are sometimes felt too abstract by stakeholders. They cover classes of intended behaviors but such behaviors are left implicit. Goals may also be hard to elicit in the first place and make fully precise.
- *Scenarios* capture typical examples or counterexamples of system behavior through sequences of interactions among agents. They support an informal, narrative, and concrete style of description. Scenarios are therefore easily accessible to stakeholders involved in the requirements engineering process [Jar98]. On the down side, scenarios are inherently partial and cover few behaviors of specific instances. They leave intended system properties implicit. Scenarios may also entail premature design decisions about event sequencing and distribution of responsibilities among system agents.
- *State machines* capture classes of required agent behaviors in terms of states and events firing transitions. They provide visual abstractions of explicit behaviors for any agent instance in some corresponding class. State machines can be composed sequentially and in parallel, and are executable. They can be validated through animation and verified against declarative properties [Mag06]. State machines also provide a good basis for code generation. On the down side, state machines are too operational in the early stages of requirements elaboration. Their elaboration may turn to be quite hard.

In view of the complementary benefits of goals, scenarios and state machines, a rich system model should provide multiple facets covering the intentional, structural and behavioral dimensions in a coherent way. Building and verifying such models turns out to be quite hard for complex systems. Efforts were therefore recently made to automate parts of this process, notably, by synthesizing behavior models from scenarios of interactions between the software-to-be and its environment.

For example, a labeled transition system (LTS) model can be synthesized from (high-level) message sequence charts (MSC and hMSC) taken as positive examples of system behavior [Uch03]. UML state diagrams can be generated from sequence diagrams capturing positive scenarios [Whi00, Mak01]. Goal specifications in linear temporal logic can also be inferred inductively from MSC scenarios taken as positive or negative examples [Lam98].

Objectives

This thesis, together with its companion by Christophe Damas (Automated support for modeling software systems: model analysis) aims at discovering tool-supported techniques that help the building and the analysis of models in the structural/behavioral/intentional triangle. The contributions can be summarized as follows:

1. A coherent framework for multi-view modeling that allows modeling and reasoning in terms of scenarios, state machines and goals. Our work extends existing approaches allowing to mix together state-based abstractions and event-based models. In particular we introduce a formal way to use guards in MSC, hMSC and LTS leading to guarded-MSC, guarded-hMSC and guarded-LTS (contribution shared with the companion thesis)
2. A deductive approach for generating guarded-LTS and LTS from guarded-hMSC.
3. An inductive approach for synthesizing LTS models from MSC and hMSC models. This inductive technique is based on grammar induction algorithms known as RPNI and BlueFringe. These algorithms are used and extended in several ways:
    a. We show how the induction search-space can be pruned through state-based model abstractions. Interestingly, these abstractions are LTS decorations generated using a symbolic execution algorithm contributed by the companion thesis.
    b. QSM (Query-driven State Merging) is an interactive version of BlueFringe allowing to prune the induction search space through membership queries submitted to an end-user. In the context of software model synthesis, these queries are additional scenarios to be labeled as positive or negative software behaviors.
    c. We also show how control information can be used both to prune the induction search space. This leads to the ASM algorithm (Automaton State Merging) which allows inducing LTS models from h-MSC models (relaxing the constraint of providing MSCs only). This algorithm can also be used to generalize LTS models, typically derived deductively using the technique presented in 2.
    d. Lastly, we show how to take goals into account in the induction process. Injecting goals has the additional benefits of pruning the search space and ensuring the coherence of synthesized models in the golden triangle. Identifying goals may be an hard task for complex systems and the companion thesis shows how automatically discovering goals from scenarios.
4. The induction techniques will be evaluated both on typical software engineering case studies and synthetic data. Tool support will also be discussed and open-source libraries provided as an additional contribution.

Approach and Work Plan

This thesis will be based on the following papers, most of them being already submitted and accepted for publication:

[Dam2010] C. Damas, B. Lambeau, F. Roucoux and A. van Lamsweerde, Abstractions for Analyzing Decision-Based Process Models, submitted for acceptance to ICSE'2010: 32th International Conference on Software Engineering, Cape Town, South Africa, May 2-8, 2010.

[Dam2009] C. Damas, B. Lambeau, F. Roucoux and A. van Lamsweerde, Analyzing Critical Process Models through Behavior Model Synthesis, Proc. ICSE'2009: 31th International Conference on Software Engineering, Vancouver, Canada, May 16-24, 2009.

[Lam2008] B. Lambeau, C. Damas and P. Dupont, State-merging DFA Induction Algorithms with Mandatory Merge Constraints, Lecture Notes in Artificial Intelligence No. 5278, Springer, pp. 139-153, 2008, 9th International Colloquium on Grammatical Inference, St Malo, France, September 22-24.

[Dup2008] P. Dupont, B. Lambeau, C. Damas and A. van Lamsweerde, The QSM Algorithm and its Application to Software Behavior Model Induction, Applied Artificial Intelligence, Vol. 22, 2008, 77-115.

[Dam2006] C. Damas, B. Lambeau and A. van Lamsweerde, Scenarios, Goals, and State Machines: a Win-Win Partnership for Model Synthesis, Proc. FSE'06: Intl. ACM Symposium on the Foundations of Software Engineering, Portland (OR), November 2006.

[Dam2005] C. Damas, B. Lambeau, P. Dupont and A. van Lamsweerde, Generating Annotated Behavior Models from End-User Scenarios, IEEE Transactions on Software Engineering, 31, 12, 2005, p. 1056-1073.

This work has been previously supported by the Regional Government of Wallonia (ReQuest project, RW Conv. 315592 and GISELE project, RW Conv. 616425). The thesis author is currently supported by the MoVES project (PAI program of the Belgian government), for which 24 months funding is available.

Work remaining mainly includes the redaction of the thesis itself. Some work also remains on the tool support and evaluation.

References

[Gia03] D. Giannakopoulou and J. Magee, “Fluent Model Checking for Event-Based Systems”, *Proc. ESEC/FSE 2003*, Helsinki, 2003.

[Jar98] M.Jarke and R. Kurki-Suonio (eds.), Special Issue on Scenario Management, *IEEE Trans. on Software. Engineering*, Vol. 24 No. 12, Dec. 1998.

[Jef98] R. Jeffords and C. Heitmeyer, “Automatic Generation of State Invariants from Requirements Specifications,” Proc. Sixth ACM Symp. Foundations of Software Eng., 1998.

[Lam98] A. van Lamsweerde and L. Willemet, “Inferring Declarative Requirements Specifications from Operational Scenarios*”, IEEE Trans. on Software. Engineering*, Vol. 24 No. 12, December 1998.

[Lam00a]A. van Lamsweerde, “Requirements Engineering in the Year 00: A Research Perspective”. Keynote Paper, *Proc. ICSE’2000: 22nd International Conference on Software Engineering*, 2000, 5-19.

[Lam00b]A. van Lamsweerde and E. Letier, *Handling Obstacles in Goal-Oriented Requirements Engineering,* IEEE Transactions on Software Engineering, *Special Issue on Exception Handling*, Vol. 26 No. 10, October 2000, 978-1005.

[Lam04] A. van Lamsweerde, “Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice”, Keynote Paper*, Proc. RE’04, 12th IEEE Joint Intl. Requirements Engineering Conf.*, Kyoto, Sept. 2004, 4-8.

[Mak01] E. Mäkinen and T. Systä, “MAS – An Interactive Synthesizer to Support Behavioral Modeling in UML”, *Proc. ICSE’01 – Intl.Conf. Soft. Engineering*,, Toronto, Canada, May 2001.

[Mag06] J. Magee and J. Kramer, *Concurrency: State Models and Java Programs*. Second Edition, Wiley, 2006.

[Uch01] S. Uchitel, J. Kramer, and J. Magee, “Detecting Implied Scenarios in Message Sequence Chart Specifications,” Proc. ESEC/FSE’01— Ninth European Software Eng. Conf. & ACM SIGSOFT Symp. Foundations of Software Eng., Sept. 2001.

[Uch03] S. Uchitel, J. Kramer, and J. Magee, “Synthesis of Behavioral Models from Scenarios,” IEEE Trans. Software Eng., vol. 29, no. 2, pp. 99-115, Feb. 2003.

[Whi00] J. Whittle and J. Schumann, “Generating Statechart Designs from Scenarios”, *Proc. ICSE’2000: 22nd Intl. Conference on Software Engineering*, Limerick, 2000, 314-323.