Using Medical Devices to Teach Formal Modeling

Orieta Celiku
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
orietac@cs.cmu.edu

David Garlan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
garlan@cs.cmu.edu

Abstract

Formal modeling can be used as an effective technique to improve the quality and reliability of software-intensive systems in general, and medical devices in particular. However, for formal modeling to be accessible to practicing engineers and domain specialists, suitable educational materials need to be developed. We report on the development of educational materials designed to give students the necessary experience to infuse formal modeling into practice. A core component of this effort is a set of modeling tasks drawn from the medical device domain.

1. Introduction

Over the past decade there has been considerable progress in the development of formal methods to improve our confidence in complex systems [1]. Today the use of such methods in certain fields, such as hardware design, or nuclear power control systems, is de rigueur, with commensurate improvements in quality and reliability.

Regrettably, however, the use of formalism in the medical device domain is relatively sparse. This is due in large part to the perceived difficulty of using formal methods by ordinary engineers and domain specialists, and by the lack of training in how best to apply existing tools to solve the problems faced in that domain.

Over the past few years we have been developing educational materials to help bridge this gap. Specifically we have developed a course in formal modeling for practicing engineers. A core component of this effort is a set of modeling tasks drawn from the medical device domain, which are used to

a) show how formal modeling can be used as an effective technique to improve quality and reliability of software-intensive systems

b) provide guidelines on selecting appropriate modeling approaches for the problem at hand

c) give students hands-on experience in modeling and tool-assisted analysis

In this paper we outline our use of medical device challenge problems in achieving these goals. We argue that such exercises (and the underlying concepts) can go a long way towards bridging the gap between theory and practice, and could be used more generally to improve the state of the practice in developing high-confidence systems, in general, and medical devices, in particular.

2. Formal Models for Software Systems

Formal Models for Software Systems is a core course in the professional Masters in Software Engineering curriculum at Carnegie Mellon University [2].[1] Students generally have about 5 years experience in the software industry, and plan on returning there at the end of the 1.5 year program.

The course is designed to give students the necessary experience to infuse formal modeling into practice. As such, it emphasizes the engineering value of formalisms (as opposed to simply the theoretical elegance), as well as the need to judiciously pick both the formalism and the level of detail to match overall goals of a software project. To achieve this students are exposed to a number of specific notations and tools, and are constantly challenged to justify decisions about choice of system features to model and the level of detail that they have chosen, and to compare strengths and weaknesses of different approaches.

3. The Modeling Tasks

A core component of the course is a set of formal modeling tasks. Key features of these include:

[1]Details on this course can be found at `http://www.mse.cs.cmu.edu/Courses-17-651.html`.

- Teams: Students work in groups of 4-5 to complete the task over a period of about 2 weeks. Teams reinforce the idea that formalization can be used as a communication vehicle, there are tradeoffs in level of detail and feature, and that work on formal modeling can be partitioned across multiple roles.

- Extension: Each modeling task requires students to extend a given system. By providing a base system, students start from an exemplary model, and can develop much more complex models than if they were starting from scratch. Moreover, the use of the same base system permits direct comparison between the approaches used by different teams.

- Presentation: Solutions to each task are presented to the class by a subset of the teams. Presentation emphasizes the need to use formalism as an effective communication vehicle and the need to explain complex models in simple ways.

- Reflection: In addition to producing a formal model, students must reflect on why they chose that particular model, and what tradeoffs they considered.

All tasks center on modeling various versions of an infusion pump. Infusion pumps convey fluids, medication, or nutrients into a patient's circulatory system, fed intravenously through one or more infusion lines. Infusion pumps are used to administer fluids in ways that would be impractical or unreliable if performed manually by nursing staff.

Infusion pumps are an ideal basis for formal modeling because they represent complex systems for which formalism can provide tangible benefits in improving reliability. In particular, because of a history of serious failures there is considerable documentation on infusion pump problems[2], as well as numerous concrete commercial examples of these devices. Students are given access to these source documents, and are expected to become familiar enough with the domain that they can make intelligent choices about what problems are most critical, and which of these problems might be amenable to formal modeling and analysis.

3.1. Modeling Notations and Tasks

Each of the modeling tasks requires students to model an infusion pump in a specific formal notation, highlighting different aspects of system complexity. The three notations used in this course are based on state machines, abstract specification languages, and concurrent processes.

[2]See, for example, JCAHO Environment of Care News Jan 2003 (`http://www.jcrinc.com/3686/`).

- State machines capture the idea that a system progresses through a set of states by performing or responding to a set of events. They provide simple abstractions of complex systems, and are the foundation of all other formalisms. State machines have intuitive graphical representations, which makes them suitable for modeling abstract operational requirements. They also have tools for visualization, simulation, and analysis of safety conditions.

- Abstract specification languages, based on set theory and predicate logic, enable the description of the desired effect of operations on the state space. The building blocks of specifications represent states, and operations on states. Operations on states are represented as relations between predicates over initial and final states. Emphasis is placed on composing larger specifications from smaller ones and incrementally refining abstract specifications into more concrete ones.

- Concurrent processes are suitable for modeling interaction in concurrent systems. They demonstrate the use of concurrency to manage complexity, separate concerns, and model reality. Concurrent processes enable analysis of important liveness properties (such as absence of deadlocks) in addition to safety properties.

Students are initially given a state machine model of infusion pumps that captures only the most rudimentary behavior of a single-line pump, including the overall flow of processing (device set- up, operation, termination) and alarms under certain conditions (such as when the device runs out of fluid; see Figure 1). Students must then extend the specification to account for other kinds of behavior such as occlusions in the delivery lines and power failure. They have considerable freedom, however, in deciding what additional functionality to model, based on their understanding of the device and its safety-critical nature.

When modeling with abstract specification languages students are asked to extend their model of the single-line infusion pump to model pumps with multiple lines, but which execute in an interleaving fashion. The emphasis is on the abstract specification of operations, capturing the pre-conditions for safe operation, specifying remedies in the presence of failures, and building larger specifications from smaller ones.

Finally, students are asked to model a 2-line infusion pump using concurrent processes. Emphasis is placed on using concurrency to factor the model into parts that represent different concerns, such as power system, an individual line, alarms, and an interface to set-up and control the pump.

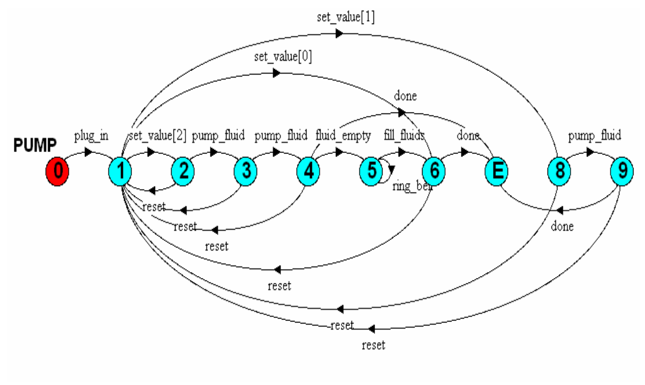

Figure 1. Simplified Description of Base System State Machine.

3.2. Model Analysis and Reflection

State machines and the concurrent process notation used in this course are also supported by tools for visualization, simulation, and analysis of certain properties. Using these tools, or predicate logic inference, students are asked to analyze their specifications for conformance with respect to certain properties. Moreover, they are asked to reflect on the modeling experience. Some of the questions they answer are

- Which aspects of the pump did you choose to model; which did you choose to leave out?
- What ambiguities in the English description of the infusion pump does your specification resolve?
- State some general properties that your pump guarantees (for example, the alarm will always sound if a line becomes clogged, the system never deadlocks), and say why they are guaranteed.
- Which recorded failures of real infusion pumps does your model address? Does your pump preclude some of them from happening?
- What are the strengths and weaknesses of the notation and tools used? Under what situations would you recommend their use? Under what situations would you not recommend their use?
- With respect to each notation, what is the single most-important future development that would be needed to make it more generally useful to practitioners?
- Reflect on the experience of developing a model in a group; concentrate on how the formal model helped or hindered you in understanding the infusion pump.

4. Conclusion

As we have outlined above, the use of medical devices (such as infusion pumps) provides an ideal basis for teaching practicing engineers the benefits and techniques of formal modeling. By grounding the course in practical exercises inspired by this domain students learn to apply formal modeling and analysis in a realistic setting, helping to bridge the gap between theory and practice. Although we have not carried out formal evaluation of the impact of the course, anecdotal evidence gathered over the past five years indicates that students find that they are able to later apply the formal techniques in their own industrial context, matching the use of formalism to the needs of system development.

While this course indicates the promise of applying formalism in medical device domains, and suggests that this domain can effectively motivate the use of formal methods, there remain many avenues for improvement. In particular, there is a notable lack of examples from which to draw realistic problems. Second, we have yet to find a good textbook that covers a broad enough base of formal approaches that students can learn to make good engineering decisions in their choice of formalism, and their use of it on a particular problem. Finally, we see this course as an important first step towards a much broader goal of deepening the outreach of computational thinking [3] into practical domains.

Acknowledgements

This work was supported by grants from the Army Research Office through grant number DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems"), the National Science Foundation under grant number CCR-0205266. We would like to thank Insup Lee and his research group for suggesting medical devices as a focus for formal modeling, and for providing us with documentation about such systems. We would also like to acknowledge contributions to the design and content of Models of Software Systems from Daniel Jackson, Daniel Kroening, Dawn McLaughlin, Anthony Hall, Jeannette Wing, and the numerous teaching assistants and students who participated in the course over the years.

References

[1] E. M. Clarke and J. M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

[2] D. Garlan, D. Gluch, and J. Tomayko. Agents of change: Educating software engineering leaders. *IEEE Computer*, 30(11):pages 59–65, 1997.

[3] J. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, Mar. 2006.