Motivations for Model Checking of a Fault Protection System

Martin S. Feather
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109, USA
Martin.S.Feather@Jpl.Nasa.Gov

Stephen S. Fickas
Computer Science Department
University of Oregon
Eugene, OR 97403, USA
fickas@cs.uoregon.edu

Model checking of spacecraft systems

Model checking is a powerful analysis technique that has emerged from the formal methods research community. Over the last decade or so it has been successfully applied to analyze a variety of hardware and software designs.

At NASA, model checking has been employed to analyze spacecrafts' critical control software. For example, Schneider et al. used model checking on the checkpoint-rollback scheme of a spacecraft's fault protection system (Schneider et al. 1998). The NASA-Ames Software Engineering Group http://ase.arc.nasa.gov/ applied model checking to the DS-1 spacecraft's real-time executive, a component of an artificial intelligence based autonomy architecture. In ongoing work they are extending model checking to work directly from programs written in Java (Visser et al., 2000).

We are about to embark on a modest study (approximately 5 work-months in total) that seeks to apply model checking to the core of the fault protection system of a new spacecraft. Fault protection systems are good candidates for model checking. Furthermore, this particular fault protection system shares some characteristics with "intelligent" systems – its design is formally specified and the responses it commands are formally specified. These make it particularly amenable to model checking.

The purpose of this summary is to outline the motivation for, and nature of, this study. By the time of the workshop the authors hope to be near to completing their work, and have specific results to report.

Rationale for model checking fault protection systems

Model checking is just one of a range of techniques that could be applied to validate aspects of a system. The choice of model checking should be driven by a combination of factors, notably:

- **Criticality** of the system aspects being validated by model checking,
- **Effectiveness** of model checking in comparison to alternative validation methods, and
- **Tractability** of model checking vis-à-vis the size of the search space.

Here we present rational for why fault protection systems exhibit the above factors, and are thus suitable candidates for model checking.

The fault protection system of a spacecraft monitors the health of the spacecraft's hardware and software, and coordinates and tracks responses to faults that it detects. It is obvious that the correct operation of the fault protection system itself is **critical** to the successful operation of the spacecraft.

The fault protection system operates concurrently with the spacecraft's other systems. Faults can occur at any time, including the time during which the fault protection system is responding to a previously detected fault. This concurrency leads to a plethora of possible execution sequences. Conventional means of assurance are inadequate to handling such a large space of possibilities (e.g., testing can cover but a small fraction of the space; human insight is apt to miss certain cases). It is in situations like this that model checking has proven to be especially **effective**.

Finally, fault protection deals with an abstraction of the system that it is protecting. Between the fault protection system and the spacecraft are monitors. These monitors abstract from the detailed operation of the spacecraft to yield fault symptoms. The fault protection system then maps these symptoms to the faults that would explain their presence. Symptoms and faults are an abstraction of the detailed operation of the spacecraft. Model checking's susceptibility to the combinatorial explosion of large search spaces has traditionally been quelled by working with abstractions. Thus a fault protection systems, because it deals with abstractions, is likely to exhibit the **tractability** that model checking requires.

Specifics of our model checking study

Our study focuses on model checking the fault protection system for a new spacecraft, whose control software is currently under development. The fault protection system is an adaptation of one used successfully in the past. From a

validation standpoint, one of the key changes made in adapting to the new spacecraft concerns the means by which the fault protection software interacts with the rest of the spacecraft. In the original design, fault protection communicated *synchronously* with the spacecraft. In the adapted design, communication is *asynchronous*. This switch to asynchrony raises concerns about race conditions, and their possible effect on fault protection requirements. Of course, the designers have considered this change, and crafted their design accordingly. Nevertheless, the criticality of this software warrants thorough validation and verification. The purpose of our study is to ascertain whether or not certain key requirements of fault protection are preserved in the adapted design. The critical requirements that we will focus on are:

- "Fault Protection shall map reported symptoms to faults and start the execution of the response"
- "Fault Protection shall avoid running a response unnecessarily"

Our task is made easier by the availability of a formal description of the core of the fault protection system. This takes the form of statecharts that specify the design. The "engine" of the fault protection system – the piece that maps symptoms to faults, decides upon responses, and manages execution of those responses – is specified by means of a statechart. Additionally, each of the responses is specified as a statechart, the execution of which is managed by the engine. Finally, even the monitors are specified by statecharts, (but operate outside of the fault protection system). This extensive use of statecharts in fault protection was pioneered on the Deep Space One project, allowing for the use of automatic code generation (Rouquette, Neilson & Chen, 1999).

For our purposes of model checking, the availability of these statechart descriptions is most helpful. The anticipated cost of extracting model checkable description is considerably less than would be the case if only natural language system descriptions were available. Similarly, there is far more confidence that the statechart is an accurate specification, and hence that analysis based on this will reflect the true state of the design.

We have selected Holzman's model checker SPIN http://netlib.bell-labs.com/netlib/spin/whatispin.html, given what we perceive as a good match with the asynchronous nature of the task. In the coming months we will be hand translating the fault protection engine statechart into PROMELA , (SPIN's input language). While we could consider the use of automatic translation, e.g., (Mikk, Lakhneck & Siegel, 1998) or (Bose 1999), the potential costs of a one-time hand translation are not large, and allow us maximum control of the PROMELA representation. We will also express the relevant aspects of the asynchronous communication between fault protection and spacecraft as PROMELA. Finally, we will express the key fault protection requirements as temporal properties. We will use SPIN to ascertain whether the fault protection engine in its operational context meets those requirements.

It is interesting to note that many "intelligent" systems exhibit these same characteristics of readily available formal descriptions of the objects they reason with, and of the reasoning engines themselves. We believe these are key to the practical application of advanced validation techniques such as model checking.

The goals of this research are to (1) validate the critical requirements of this fault protection system, (2) ascertain the cost and effectiveness of model checking when formal design descriptions are available, and (3) indicate those areas where further research (e.g., into application of automatic translation) would further decrease costs of validation.

Acknowledgements

The research and development described in this paper is being carried out at the Jet Propulsion Laboratory, California Institute of Technology, and the University of Oregon, Eugene, under a contract with the National Aeronautics and Space Administration. The authors thank Jeff Levison, Nicolas Rouquette and Chi Lin their help in establishing this effort.

References

Bose, P. 1999. Automated Translation of UML Models of Architectures for Verification and Simulation Using SPIN. *Proceedings of the 14th IEEE International Conference on Automated Software Engineering*, 102-109, Cocoa Beach, Florida, October 1999.

Mikk, E.; Lakhneck, Y.; and Siegel, M. 1998. Implementing Statecharts in PROMELA/SPIN. *Proceedings of the 2nd IEEE Workshop on Industrial-Strength Formal Specification Techniques*, 90-101, Boca Raton, Florida, October 1998.

Rouquette, N.F.; Neilson, T.; and Chen, G. 1999. The 13th Technology of Deep Space One. *Proceedings of the 1999 Aerospace Conference*, 477-487 vol.1.

Schneider, F.; Easterbrook, S.M.; Callahan, J.R.; and Holzmann, G.J. 1998. "Validating Requirements for Fault Tolerant Systems using Model Checking", *Proceedings, 3rd International Conference on Requirements Engineering*, 4-13, Colorado Springs, Colorado, April 1998.

Visser, W.; Brat, G.; Havelund, K.; and Park, S. 2000. "Model Checking Programs", *Proceedings of the 15th IEEE International Conference on Automated Software Engineering*, 3-11, Grenoble, France, September 2000.