Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 28 (2014) 146 – 154

Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

Abstractions for Executable and Checkable Fault Management Models

Corrina Gibson[a]*, Robert Karban[b], Luigi Andolfato[b], John Day[a]

[a]*Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91011 USA*
[b]*European Southern Observatory, Karl-Schwarzschild-Str. 2 85748 Garching, Germany*

Abstract

The work presented in this paper describes an approach used to develop SysML modeling patterns to express the logical behavior of fault protection (FP), test the model's logic via fault injection simulations, and verify the system's logical design via model checking. A FP model was architected with collaborating Statecharts that captures interactions between relevant system components (error monitors, FP engine, devices) and system behavior abstractions. Development of a method to implement verifiable and lightweight executable FP models enables future missions to have access to larger fault test domains and verifiable design patterns.

Keywords: Model Checking; Model execution; Java Pathfinder; SysML; Statechart; Fault Protection; Design Validation

* Corresponding author. Tel.: +1-626-660-5107
E-mail address: Corrina.L.Gibson@jpl.nasa.gov

doi:10.1016/j.procs.2014.03.019

1. Introduction

Spacecraft with highly complex Fault Protection (FP) systems emphasize the need to explore improved methods of developing, testing, and validating FP systems of the future. To investigate model-based FP engineering, a behavioral model of a FP system was developed in an environment that can be executed and model checked. The model provided a basis for checking FP design versus the defined failure space, running fault injection tests, and analyzing methods for validation of the FP logical design.

In order to gain confidence in the validation and verification of the model-based design and its implementation, the following questions must be addressed:

- Does the model represent the system?
- Do the generated artifacts for model checking represent the model?
- Do the generated artifacts for model checking represent the final software system implementation?

Simulation of the model caught initial modeling and design translation errors. The ability to inject a variety of inputs to the FP model during execution enabled the autonomous behaviors of the system to be tested and lead to confidence in the logical design of the model. It became clear, as more error monitors and responses were added to the model, that it would not be possible to manually run simulations for all of the possible sequences of the model – a model checker is necessary to formally and exhaustively verify the model for all possible sequences. It is assumed that the generated artifacts for model checking represent the model. However, this could be mitigated by comparing model simulation results with the execution of the generated code for model checking.

2. Background of Fault Management

Fault management is the operational capability of a system to contain, prevent, detect, isolate, diagnose, respond to and recover from conditions that may interfere with nominal mission operations[4]. Fault protection is a common term used to describe the elements of fault management that are part of the flight system design. FP consists of protective functions that are defined to influence system behavior in the presence of off-nominal (unexpected or unintended) conditions. These functions are implemented as a set of mechanisms, which are part of the flight system or ground system. These typically include flight software (FSW) error monitors and fault responses, but also include other mechanisms such as contingency plans and specific hardware (e.g., fuses, opto-isolators). FP is responsible for actions such as health and redundancy management, system fault integration, announcement handling, and response execution.

Validation of FP designs is historically problematic, with many examples of inadequate resources (people, time, and budget) and/or unexpected problems. Many factors contribute to these issues, but the problem can be traced to a lack of appreciation of system complexity. When considering a system, there are significantly more ways the system can fail (contingency paths) than ways it can succeed (nominal paths)[5]. As NASA continues to develop more complex and capable spacecraft, the behavior state space will increase, stressing the ability of teams to properly understand the behavior.

In the work described herein, a subset of the Soil Moisture Active Passive (SMAP[7]) FP design[1], consisting of error monitors, local and system fault responses, and logic for activation of system responses, is represented as a set of collaborating Statecharts. A representative description of these elements is shown in Fig. 1. These abstractions, along with other necessary abstractions of spacecraft behavior are used to capture the necessary behavioral characteristics of the SMAP system. The patterns for developing the abstractions are particularly relevant to assuring the integrity of the model.