

Engineering Complex Embedded Systems with State Analysis and the Mission Data System

Michel Ingham, Robert Rasmussen, Matthew Bennett, and Alex Moncada
NASA Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
Email: *firstname.lastname@jpl.nasa.gov*

Following failures of recent space missions, such as the Mars Polar Lander and Mars Climate Orbiter [1], it has become clear that spacecraft system complexity has reached a threshold of viability where customary methods of control are no longer affordable or sufficiently reliable. At the heart of this problem are the conventional approaches to systems and software engineering. Divide-and-conquer strategies based on subsystem-level functional decomposition fail to scale in the tangled web of interactions typically encountered in complex spacecraft designs, where subsystems are inevitably very tightly coupled through their need to share limited resources [2]. Furthermore, there is a fundamental gap between the requirements on software specified by systems engineers and the implementation of these requirements by software engineers. Software engineers must perform the translation of requirements into software code, hoping to accurately capture the systems engineer's understanding of the system behavior, which is not always specified explicitly. This gap opens up the possibility for misinterpretation by the software engineer of the systems engineer's intent, potentially leading to software errors.

NASA's Mars Science Laboratory mission, to be launched in 2009, is addressing the complexity problem by adopting the *Mission Data System (MDS)* as its embedded software architecture, and *State Analysis* as its systems engineering paradigm. The overarching goal of MDS is to provide a multi-mission information and control architecture for robotic exploration spacecraft that is used in all aspects of a mission, from development and testing to flight and ground operations. MDS acknowledges and leverages the intimate coupling between software and systems engineering, by elevating the notion of *state variables* to an architectural level where interactions can be addressed overtly [3]. It emphasizes the separation of application-specific knowledge, in the form of models of the system under control, from reusable general-purpose code, in the form of architectural components such as estimators, controllers and schedulers (Figure 1). Instead of issuing low-level open-loop commands, MDS has a Mission Planning and Execution component which schedules and issues goals that indicate intent in the form of constraints on the values of a state variable, over a time interval. Such goal-directed control provides intrinsic robustness to off-nominal behavior, by allowing the controllers to decide how best to achieve the goals.

The MDS framework is based on a systems engineering approach called *State Analysis*, which provides a process for capturing system and software requirements in the form of models. These include the *State Effects Model* (physical model of how state variables evolve over time, under the effects of other state variables), *Command Models* (how state

variables are affected by commands), and *Measurement Models* (how measurements are affected by state variables). The relationships between state variables, commands and measurements are graphically captured in the form of a State Effects Diagram (Figure 2). The State Effects Model compiles information traditionally documented in a variety of systems engineering artifacts, such as the Hardware Functional Requirements, the Failure Modes and Effects Analysis, and the Hardware-Software Interface Control Document. In addition to being used in the design of estimator and controller algorithms, the State Effects Model is used to specify how goals elaborate into subgoals on related states. These *goal elaborations* form the building blocks for the goal-based “sequences” that are executed onboard the spacecraft, called *goal networks* (Figure 3).

State Analysis and MDS are currently being used to design and implement a robust control architecture for MSL, and have already been deployed and demonstrated on the Rocky7 and Rocky8 rover testbeds at JPL. This paper describes how requirements for complex aerospace systems can be developed using State Analysis and implemented in the MDS framework, using examples representative of the types of models and algorithms that are being developed for MSL.

References:

- [1] Young, T., et al., Report of the Mars Program Independent Assessment Team, Technical report, NASA, March 2000.
- [2] Dvorak, D., “Challenging encapsulation in the design of high-risk control systems”, Proceedings of the 17th ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'02), 2002.
- [3] Dvorak, D., Rasmussen, R., Reeves, G., and Sacks, A., “Software architecture themes in JPL's Mission Data System”, Proceedings of the AIAA Guidance, Navigation, and Control Conference, paper number AIAA-99-4553, 1999.

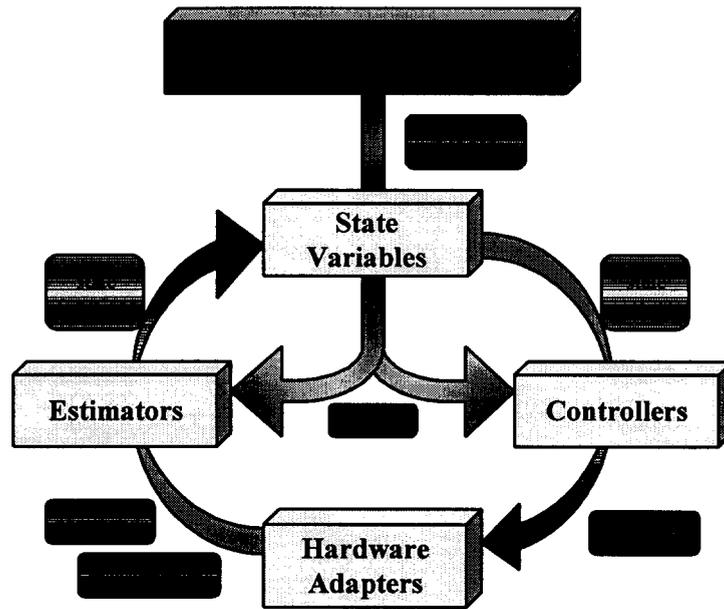


Figure 1. The MDS State-based Software Architecture

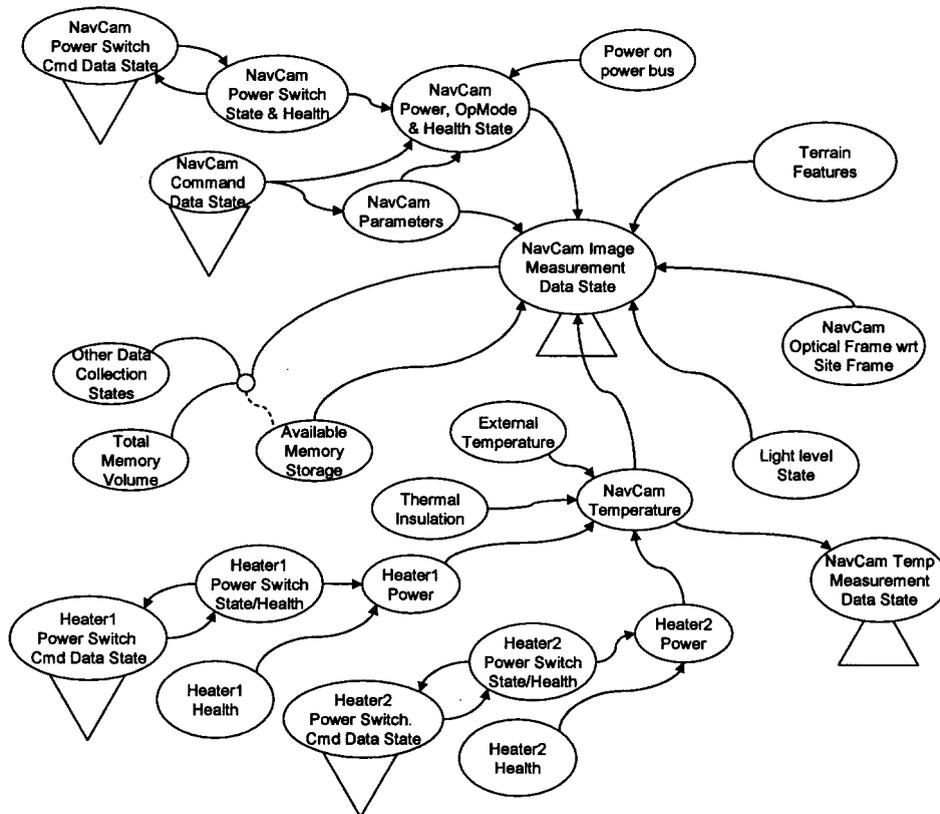


Figure 2. Excerpt from an example State Effects Diagram

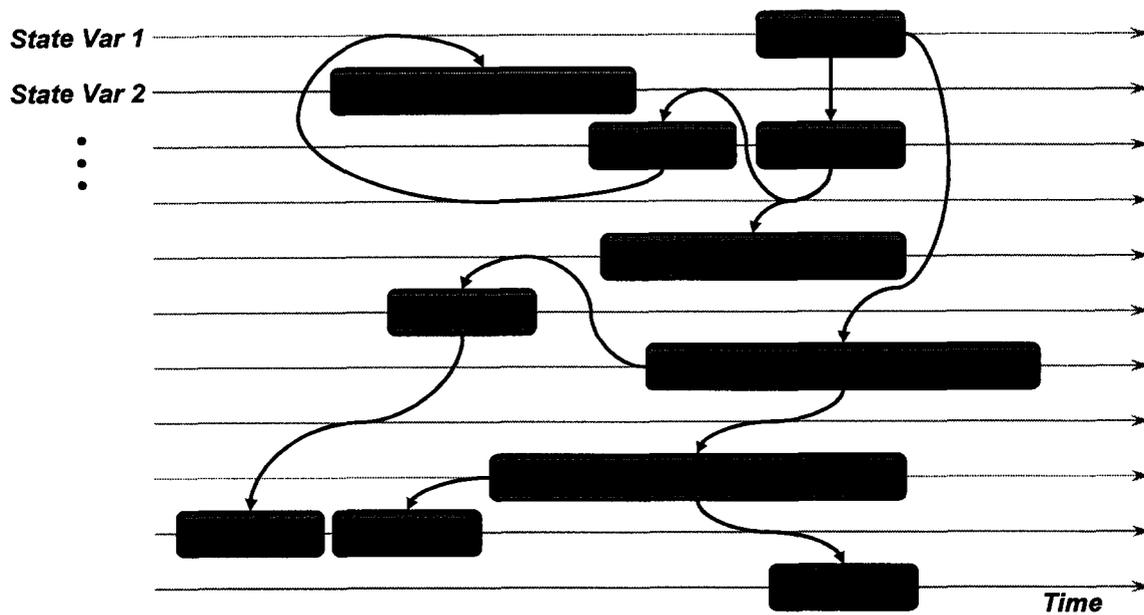


Figure 3. An MDS Goal Network