The Space Congress® Proceedings

1998 (35th) Horizons Unlimited

Apr 28th, 2:00 PM

Paper Session I-B - Modeling Methodology: The Use of Vehicle Control System Modeling for the International Space Station (ISS) Program

Scott C. Haase
Boeing

Jimmy L. Williams
Boeing

Modeling as Methodology:

The Use of Vehicle Level Control System Modeling for the International Space Station (ISS) Program

Scott C. Haase (Boeing), Jimmy L. Williams (Boeing)

This paper presents a new control system software development approach entitled, RAPid Modeling and Analysis Philosophy (RAPMAP), and presents an example of the application of this approach on the International Space Station (ISS) program. RAPMAP embraces control system behavior modeling as an integral and ongoing part of the nominal development and integration process. Through a process called 'Predictive-Verification,' the RAPMAP approach ensures that development proceeds based on known and demonstrable performance and behavior throughout the development life cycle. The RAPMAP approach provides early identification of system integration issues and support for test and operational procedure development and training long before development products, which normally support these activities, are available. Section 1 of this paper introduces the concept of control system behavior modeling. Section 2 describes the RAPMAP approach and Section 3 provides a description of the the RAPMAP approach including a detailed description of its application on the ISS program. Section 4 summarizes the benefits of the RAPMAP approach.

1. Control System Behavior Assessment

In most large system developments, technical performance measures (e.g., weight, logistics usage, computer throughput, processor memory usage) are selected and models of the system are used throughout the definition and design phases to predict the performance of the integrated system against corresponding required levels. As products become available, actual test and measurement data replaces model data, and the analysis becomes a part of the system verification. Similarly, specialty engineering analyses (e.g., safety, reliability, electromagnetic compatibility) are performed using either system models or actual products. Control system software, particularly real-time embedded avionics development currently relies on such measures as Input/Output (I/O) throughput, memory usage, and Source Lines of Code (SLOCs) to predict and assess software viability. These measures are, at best, only tangentially related to the viability and acceptability of the required, designed and encoded software behavior. There is currently no corresponding standard or effective method for performing early validation and integration of control system behavior. However, recent technological modeling advancements provide new mechanisms for accomplishing predictive analysis of control system behavior. This paper presents the RAPMAP development approach which incorporates these advancements into the standard control system development to achieve the benefits of early and ongoing systems integration and analysis.

2. The RAPMAP Development Approach

The RAPMAP approach incorporates control system behavior modeling to provide tangible and ongoing evidence that the integrated control system will perform its allocated function(s) acceptably and as intended. In RAPMAP, models of the control system hardware, software and environment are built and maintained throughout the development and model-based analyses are used to:

- Analyze system control concepts as early as possible without expending the time and resources generally required for more formal up-front system prototyping
- Engage both the developer and customer in validating system control (pre-) conceptions or identifying necessary changes to eventual product behavior during more or less formal interactive analysis sessions

- Communicate desired control behavior between the system engineering organization and the design and code organization
- Assess behavior during selected system stressing conditions through scripted and repeatable behavior analyses
- Perform concept, requirement and design trade-studies during normal development or when evaluating change implementation options
- Perform early training or system operations familiarization exercises
- Develop product test and usage procedures
- Perform, given appropriate expansion of fidelity, full-blown training or operations analysis
- Interactively demonstrate system behavior at major program reviews to help provide a rational basis for approving the developing control system to proceed into subsequent development phases

1 RAPMAP Modeling Tool

In order to maximize these benefits, in our experience, the toolset selected to implement the RAPMAP models must perform at reasonably close to real-time, provide both interactive and script driven controls, and allow the models to reflect the actual system definition status. An interactive interface supports early developer, customer, and user familiarization with the defined system, later operations usage (e.g., procedure development and training), and allows for informal exercising (and more formal demonstration) of the control system during its development. Scriptable controls (along with automatic data gathering capabilities) allow the execution of strictly controlled and repeatable scenarios which are necessary to support system stressing behavior analyses.

In order for the RAPMAP models to reflect the system definition status, the toolset must provide:

a) Flexible, easy model modification - Allows model to be refined throughout the development as the system definition evolves without requiring wholesale model redevelopment.
b) An ability to interface with other tools or software - Allows integration of non-host tool models or even development products wherever the host tool or model is determined to be inadequate. Note that this is particularly crucial for the Product-based RAPMAP implementation discussed below.
c) Support for mixed fidelity operation - See the discussion below.
d) Support for immature functional definitions without compromising model execution or validity - Allows model to reflect the actual state of the system definition, regardless of its completion status. See the discussion below.

The modeling toolset must simultaneously support models at varying degrees of fidelity. System definitions do not mature as a unit, therefore, it is important that models of these systems allow for parallel evolutionary fidelity growth (as opposed to requiring revolutionary upgrades for every minor system change). Initial, low fidelity models and analysis results give engineers a general feel for the functional behavior of the system. But, as the life cycle progresses and the increase in system definition is progressively reflected in the model, the confidence resulting from the analysis and the insight into the integrated performance of the system grows.

A subset of the variable fidelity modeling requirement is that the toolset must not force the user to model a ‘fully realized’ control system. It is crucial that the RAPMAP models reflect the actual status of the system definition. Many available modeling tools are actually built to directly support the definition of the system. As such, in an effort to apply the strictures of good systems engineering practices, these modeling tools often force a complete system definition long before the system is mature enough to support this sort of analysis. Thus, modelers using these tools are stymied until system definition inadequacies are resolved, or in an effort to make forward

modeling progress, models becomes riddled with assumptions that nullify the value of the resulting analyses.

2 RAPMAP Implementation

There are two basic approaches to implementing RAPMAP: Analysis-based and Product-based. Analysis-based RAPMAP implements a control system behavior modeling analysis effort which is separate but parallel to the standard software development. In this implementation, the model is primarily used as an analysis tool and results are used to inform the ongoing software development. Product-based RAPMAP is an implementation in which the final product flows directly from the initial modeling.

Analysis-based RAPMAP is summarized along development life cycle lines in Table 2.2-1. This table lists each development phase along with its associated modeling and analysis activities. This table implies serial and distinct model content and usage based on development phases; however, in practice this is not the case. The variability of model fidelity, a central RAPMAP tenet discussed above, allows the model and its associated analyses to simultaneously benefit multiple development phases. Further, the modeling efforts are not independent, the modeling required in each development phase relies on the modeling effort accomplished for the previous phase.

Product-based RAPMAP wholly encompasses the Analysis-based implementation processes, but goes one more step by merging the model and control system software development. In this implementation, initial models are assembled and, after proving out control system performance, pieces of the model are replaced with progressively higher fidelity models, ultimately including the actual product. For example, the toolset used on the ISS program, MATRIX , supports the following progression: from initial low fidelity logic models to progressively higher fidelity models to automatically generated code versions to hand-coded software. This approach provides a gradual top-down development of the product which supports ongoing demonstration and examination of the functional and performance behavior of the control

Table 2.2-1 RAPMAP Modeling Approach

Development Phase	Inputs	Primary Analysis	Outputs	Modeling Development Task	Other Model Uses
Concept Definition	• System Control Concepts • Concept scenarios	Concept validation and tradeoffs	• Concept scenario analysis results • Concept issues	• Model host selection • Initial modeling and process development	Support SRR/SDR with system operation demonstration
Requirements Development	• SRR Validated system control concepts • Specifications, ICDs • Behavior scenarios	Requirement validation and tradeoffs	• Behavior scenario analysis results • Concept and Requirement issues	• Update models to reflect new requirements • Increase model fidelity	Support SSR with system operation demonstration
Design	• SSR Validated requirements • Design documents • Behavior scenarios	Design validation and tradeoffs	• Behavior scenario analysis results • Requirement and design issues	• Update models to reflect latest requirements and design • Increase model fidelity • Integrate system performance models as available	Support PDR and CDR with system operation demonstration

Table 2.2-1 RAPMAP Modeling Approach					
Development Phase	**Inputs**	**Primary Analysis**	**Outputs**	**Modeling Development Task**	**Other Model Uses**
Implementation	• Requirements and design updates • Behavior scenarios • Draft test procedures	• Test procedure development • Operations training	• Behavior scenario analysis results • Requirement and design issues • Test procedures	• Update models to reflect latest requirements and design • Increase model fidelity • Integrate development products as available	-
Integration and Test	• Requirements and design updates • Behavior scenarios • Draft procedures	• Test procedure development • Operations procedure development • Operations training	• Behavior scenario analysis results • Requirement and design issues • Test and operations procedures • Training	• Update models to reflect final requirements and design • Increase model fidelity • Validate model performance and predictions • Begin structured, informal CM process.	-
Operations	• System Performance data • Anomaly reports • System upgrade options	• System behavior performance analysis • Anomaly isolation • Upgrade tradeoffs • Operations training	• System performance predictions • Anomaly isolation data • Upgrade evaluation data • Training	• Validate model performance and predictions • Maintain model fidelity • Implement formal CM processes	-

system starting very early in the development process. Additionally, there are benefits to the test program. A complete control system model includes models of the hardware and the environment. As these portions of the control system model are gradually increased in fidelity and are validated against actual product performance data, they become the test suite for the product software without having to go through a separate development process.
Notably, in Product-based RAPMAP there are significant organizational implications. Most obvious is the merging of the Systems Engineering and Software Engineering functions. Traditionally, these organizations interface primarily through requirement allocation documentation; however, using a Product-based RAPMAP implementation, the model which defines the System behavior (functionally a Systems Engineering product) becomes the skeleton upon which the Software Engineering function progressively builds its product. Obviously, because of the difficulty of maintaining and coordinating separate models for separate organizations, this will be best accommodated if these organizations are the same.

To maximize the benefits of the RAPMAP approach, the modeling effort must be allowed flexibility not normally found in product development, Ideally, stringent documentation, program and customer review, and configuration management which must be present in the development of fielded (flight or operational) products should not be imposed on the model development process until very late in the life cycle. In an Analysis-based RAPMAP implementation, resources should be diverted from model development and use to formal model validation, documentation and configuration management only when real-time anomaly resolution/performance prediction is intended. However, in a Product-based RAPMAP implementation, this ideal is not possible. Since the model becomes the product, it must also be formally documented, controlled and reviewed. As a result, in this implementation, a balance needs to be found which maximizes the flexibility of the modeling but also provides the necessary control and insight into

the model development. This balance may entail new documentation approaches which diverge from the standard concentration on the structure and content of hand-coded software and concentrate instead on functional control system behavior. In the current engineering environment where emphasis is being placed on cheaper, faster, better process performance, exploring the Product-based implementation implications is likely to be a direction taken by future control system developments.

While commitment to a cogent development philosophy which relies on control system behavior modeling has the potential for producing significant rewards, an early, overall implementation of the RAPMAP process is not the only option. As experienced on the ISS, even a gradual or partial implementation can benefit overall development quality with the potential additional benefit that modeling resource expenditures are reduced when compared to a pervasive modeling methodology. In a partial, most likely Analysis-based, implementation of RAPMAP however, early emphasis should be placed on identifying an appropriate modeling methodology and maintaining a clear picture of the resulting processes to accomplish the overall product development goals.

2.3 Predictive-Verification

The classical approach to control system software development and review, the software engineering "Waterfall," (see Figure 2.3-1) is characterized by a progressive stepwise development of products starting with concepts and requirements and flowing through design, implementation, test and integration onward to product delivery. The basis for the development half of the 'waterfall' approach (requirements-implementation) is a 'Top-Down' methodology which layers progressively more detailed definitions on top of previous definitions which are presumably more mature but of a broader scope. Conversely, the verification half of the waterfall process proceeds in the opposite, 'Bottom-Up' direction demonstrating the performance of progressively more broad-based functionality as the affected portions of the system are integrated and tested. As a result, the system's most broad-based, horizontal functional layers are actually demonstrated last because they can only be examined once the necessary large functional system segments become available. Clearly, this can be a risky approach since flaws in the broad System and Subsystem functional basis for the product may not be found until very late in the development cycle.

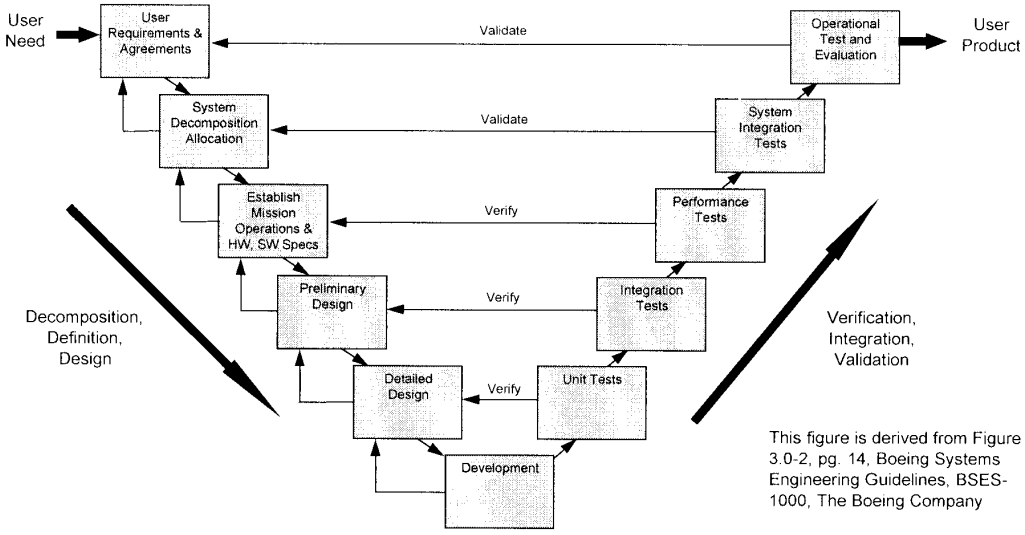

Figure 2.3-1 Program Development Cycle

The RAPMAP approach addresses this risk by implementing a continuous Top-Down 'Predictive-Verification' of eventual system behavior. As the development proceeds, the models built as part of the RAPMAP approach are used to predict and validate the eventual behavior and performance of the integrated system. As seen in Figure 2.3-2, Top-Down Predictive-Verification complements the later development phase Bottom-Up verification strategy, and ensures that, during the early development phases, the program proceeds based on proven system behavior.

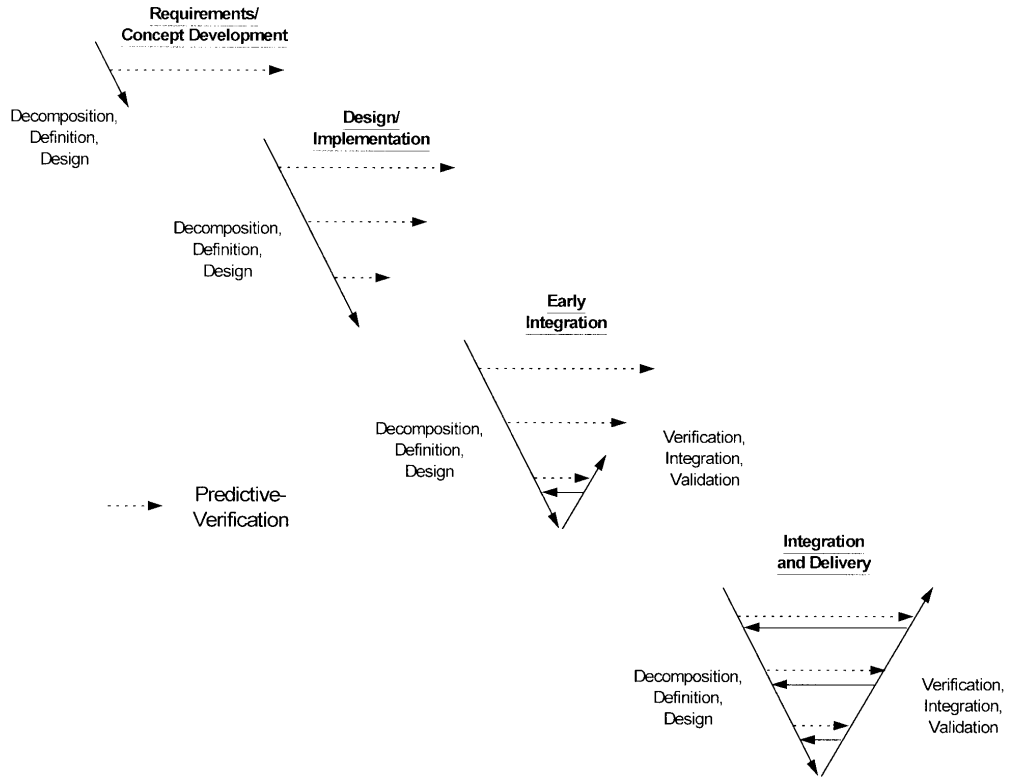

Figure 2.3-2 RAPMAP and Predictive-Verification

'Predictive-Verification' of control system behavior is accomplished through the modeling effort itself and through model use during control system behavior analyses, trade-study efforts and program reviews. Similar to the benefits associated with early product prototyping, the modeling effort exposes issues with the system definition by forcing modelers to examine, understand, and translate the available system definitions into coherent executing control system models. Issues are provided to the system definers for the product in question and result in either modifications to the system definition or corrections to modeler interpretations. 'Predictive-Verification' is also achieved when the models are used to assess system behavior through pre-scripted scenario-based analyses or more informal interactive analysis sessions.

One of the less tangible, but most notable benefits of the Predictive-Verification process is an increased level of technical communication. During a product development, customers, systems and design engineers often miscommunicate due to their differing perspectives, concerns and roles. However, in our experience, interactive model execution sessions between these groups often lead to quick understanding of system operation and the issues and options inherent therein. This communication benefit is achieved in both small group technical inter-

changes and program review demonstration forums. It is simply easier to understand and discuss a system when it can be seen running than when it is only found in a paper document, or even worse, when it is only realized in someone's mind.

3. Station Management and Control (SMC) Prototype Top Level Architecture and Purpose

The ISS Command and Data Handling (C&DH) architecture is a highly complex and distributed MIL-STD-1553B network comprising hundreds of processors with functionality which includes Power generation and distribution, Guidance, Navigation and Control, maintenance of the internal habitable environment, Communications and Tracking, and control of robotic operations and payload experiments. During the development of the software requirements for the two highest level processors in this architecture it became apparent that their complexity and interaction required some sort of early functional concept validation. This realization provided the genesis for the model now called the SMC Prototype. The use of this model led to the development of the RAPMAP approach and demonstrates the value of control system behavior modeling as an adjunct to the standard program development processes.

As a general statement, the SMC Prototype is a model of ISS vehicle-level autonomous functionality including related subsystem and cross-subsystem behavior. The primary goal of the model is to evaluate the viability of the vehicle-level control concepts and required behavior and to assess the interaction of these behaviors with affected subsystems. In terms of the ISS, vehicle-level control behaviors include Station Modes, Vehicle Failure Detection, Isolation and Recovery (FDIR) and Safing, Power and Thermal Load Shedding, and Crew Interfaces.

As can be seen in Figure 3-1, at the top level, the SMC Prototype architecture mirrors that of the actual space station. Within each station element are the software and hardware component models and their appropriate connectivity and control loops that are designed to be a part of that element on the ISS. In general, elements are simply turned on or off to simulate the progressive on-orbit assembly of the ISS. However, where necessary, assembly-stage specific functionality and connectivity has also been modeled

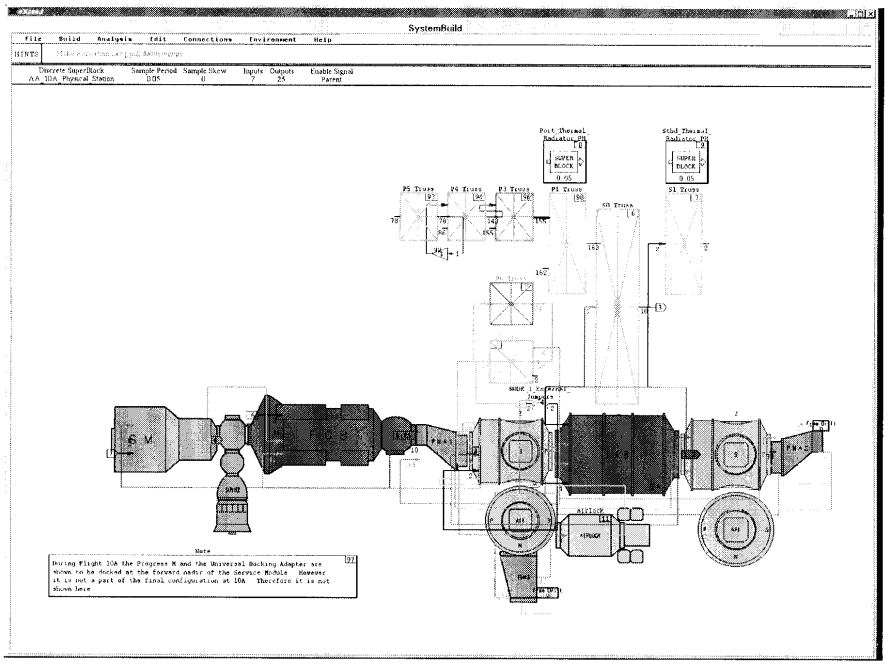

Figure 3-1 Stage 10A Top Level Vehicle Model Diagram

[Insert Figure 3-1 Stage 10A Top Level Vehicle Model Diagram]

The behavior modeled by the SMC Prototype is primarily based on ISS program requirements and design documentation including:

- System, Segment and End Item requirements documents and Interface Control Documents (ICDs)
- Software Requirements Specifications and Software ICDs
- Software Design documentation
- Instrumentation, Program and Command List (IPCL)
- Architecture team documentation including Architecture Description Documents (ADDs) and Schematics
- assorted less formal program information sources

Where externally developed models of appropriate fidelity and functionality were available that could be integrated with minimum impact, these models have been included. Additionally, actual Crew/Ground control displays - including Caution and Warning panels, alarms and displays - have been incorporated into the model to support interactive assessments of vehicle operability.

Where necessary to support analysis of the interaction of the vehicle-level control behavior, vertical subsystem functionality has also been modeled. For example, since the majority of the vehicle-level control behavior affects the on/off state of ISS devices, the whole of the Secondary Power Distribution system (connectivity and steady state power draw) is included in the model; and, since the orientation of the model is on control behavior, the Command and Data Handling (C&DH) connectivity, functionality and command and data flow has also been modeled.

Many of the conditions that are necessary to stress the vehicle-level control concepts involve failure scenarios. The SMC Prototype accommodates these scenarios by allowing for the injection of selected failure conditions. These conditions, selected based on assessments of failure likelihood and expected cross-functional impact, are modeled to the extent that they affect vehicle-level control concepts. When a failure is injected, related intra- and inter-subsystem reactions to the failures are captured through modeling of failure effects to the environment and to device communications, power usage, and connectivity.

The SMC Prototype model also provides an ability to inject environmental conditions. These conditions usually take the form of singular sensor readings, but may be more complex conditions (e.g., ‘Fire’) which normally involve a series of readings. Again, environmental aspects are controllable to the extent that they elicit or effect a vehicle-level control concept or selected failure or alarm condition.

At the Vehicle level, the SMC Prototype was used as part of an Analysis-based RAPMAP implementation. However, several of the SMC Prototype components provided by external organizations are or will directly produce actual flight products, and so are part of Product-based RAPMAP implementations. The following sections detail examples of the use of the SMC Prototype model. As noted, each of these uses exposed issues during the early development phases, long before the traditional development process would nominally have identified them.

3.1 Integrated Subsystem Requirements Allocation

The SMC Prototype model was utilized to map all of the ISS software and hardware it controls, to the Elements in which they reside along with the necessary power, thermal and data utility connectivity and flow required to make a viable system. This map was then used to determine where functional requirements should be levied in the Element specifications. As a result, many of the program specifications were updated to reflect the necessary subsystem performance, control, and connectivity allocations. Notably, this analysis did not require a

functioning model, however, the existing functional behavior models and the interactive functional analyses being performed for other reasons increased the accuracy of the software to hardware to Element correlation.

3.2 Computer Failure Recovery Logic

The ISS has a multi-tiered computer system with the Command and Control computer at the top of the hierarchy. The C&C computer (Tier I) controls numerous Tier II Remote Terminals (RTs) including the Guidance, Navigation, and Control (GNC), the Power Management and Control Unit (PMCU), and the Internal (INT) computers. A static analysis of the failure recovery logic, data flow, and resource connectivity for each individual Tier II computer indicated a fully functional and complete concept and requirements set. However, when multiple computers where brought down through injection of a power channel trip in the SMC Prototype model, the redundant GNC computer did not recover properly.

Analysis of the data recorded during the Prototype scenario revealed that the backup GNC computer was powered through a switch which was in turn controlled by the INT computer. When both the INT and the GNC primary computers failed due to loss of the power channel, the C&C computer initiated the GNC and INT recoveries simultaneously, as then required. However, because the INT was still being recovered, the command to provide power to the backup GNC never progressed to the controlling switch. By the time the INT had completed its recovery and was ready to process a GNC switch command, because the backup GNC had failed to recover as commanded, the C&C had already determined that the backup GNC was inoperable, designated it as failed, activated appropriate Caution & Warning alarms, and halted its attempt to recover the GNC processor.

Of particular note for this issue is the likelihood that it would not have been identified until very late in the verification process since integrated testing would only occur after completion of all of the applicable components. Individual component tests and connectivity analyses would not have revealed this issue.

3.3 Operational Initialization

During the execution of 2A stage assembly scenario scripts, the entire set of primary heaters for the Node 1 module shell were reported as failed. Since heater control functionality is one of the first automated software functions to be enabled on the Vehicle, the immediate failure of the entire primary string was of significant concern.

Analysis of recorded data revealed two problems. The first was that the specified heater failure levels neglected to account for the expected shell temperatures at heater control initialization. The second was that because the required failure recovery logic did not specify removal of power from the failed heaters, and because once a string was declared failed, the nominal control logic that would have turned them off when the shell temperatures reached the upper limits would not execute, the integrated primary heater string control system demonstrated an unexpected and unhandled failed-on failure mode.

Because these errors were discovered during the execution of operational scenarios against a model that simulated the on-orbit conditions, it is highly likely that these problems would not have been detected until the on-orbit assembly operations. However, because the SMC Prototype was being used to ‘Predictively-verify’ the assembly operations, this error was discovered and corrected early in the design phase of the program.

4. Conclusion

The overall results currently experienced on the International Space Station program due to the application of the RAPMAP approach benefit a wide range of development areas, including:

- Early identification of concept deficiencies
- Rapid functional review of software requirements
- Early operations procedure development and review
- Early software-based Hardware/Software Integration analysis
- Software functionality implementation trade study
- Early operator training
- Software test procedure development and review
- Customer system operations familiarization
- Software to software interface analysis

Another realized benefit, which is less tangible but still of note, is an improvement in system communication with the customer, particularly at a technical level, and the corresponding increase in confidence exhibited in the customer regarding integrated product analyses and reviews. Based upon the benefits achieved on the ISS program from an evolution into the RAPMAP process, implementation of this philosophy from program inception throughout the development cycle has the potential to yield even greater cost reduction and engineering excellence benefits to the development of large software control systems.