
Technical Tracks associated with paper (indicate 1, delete those which don't apply)

* Modelling and Tools

Areas of Special Interest associated with paper (indicate 1, delete remainder)

Model driven design

Author's Information: **(for Brian Mar Student Award)**

Professional

Intended Audience (*indicate all that apply, delete others*):

Experienced Practitioner
Trainers
New Practitioner

Academics

Researcher

Recommended Expertise of Reviewer (indicate one or more, delete others):

Architecture
Requirements Mgmt
Tools
Process

SE Mgmt
Systems Analysis

Using COTS tools in the Development of a Model Based Avionics Architecture Tool

Jet Propulsion Laboratory
4800 Oak Grove Dr., Pasadena, CA 91109, USA

I-Logix Inc.
Three Riverside Drive, Andover, MA 01810, USA

Abstract. Information and discussion of the development of an Avionics Architecture Tool that is model-based using commercial off the shelf (COTS) products will be provided. Statemate Magnum from I-Logix Inc. was used in this work. Five areas of interest will be discussed. How to allow Architecture trades in a cost-effective manner with environments that are frequently changing. How to allow easily modified "Plug and Play" architectural models. How to utilise a common standard interface to address all technical concerns. How models generated using the tool can be used to evaluate and produce lower level requirements. Finally, how to integrate tool-generated models with existing or externally generated subsystem models for spacecraft level simulation.

INTRODUCTION

System engineering plays a more important role in spacecraft design than most of the other applications. This is because space missions are expensive and risky, and space environments are very unforgiving. Any mistake made in system engineering would easily result in millions of dollars of budget overrun or even the loss of the spacecraft. Over the years, the Jet Propulsion Laboratory (JPL) has developed a system engineering process for spacecraft development. This process includes mission conceptual development, system definition, and architecture design. While this process has been successfully applied to many flight missions, it has been costly and lengthy because it requires many meetings of system engineers in order to establish the requirements and specifications. Worse yet, these traditional textual requirements and specifications are error prone, often ambiguous and sometimes contain many oversights and mistakes. Therefore, JPL has initiated a series of efforts to develop a set of system engineering tools to improve the process. One of

these tools is the Avionics System Architecture Tool (ASAT).

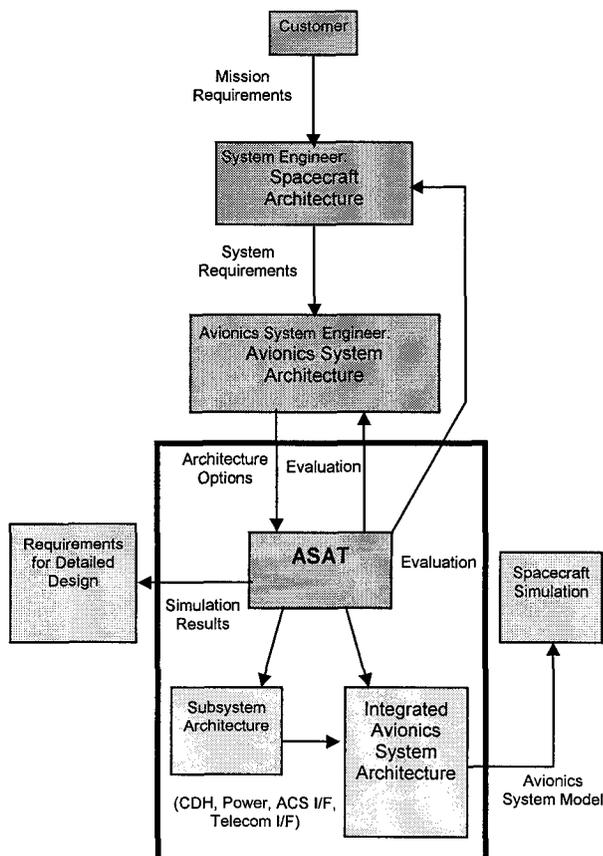
ASAT focuses only on the avionics system of the spacecraft design. The process of avionics system design is similar to, and in fact driven by, the spacecraft design process. Hence, in the *mission concept phase*, the avionics system engineer has to estimate mass, power, volume, and cost of the avionics system based on the mission concept. The avionics system engineer may have to negotiate with the spacecraft system engineer if the estimates exceed the amounts allocated. Eventually, these estimates become the mission level requirements for the avionics system. In the *system definition phase*, more detailed mission requirements are developed. These requirements are evaluated by the avionics system engineer, who then creates a high-level avionics architecture. The mass, power, volume, and cost estimates are revisited based on the high-level architecture. The system performance and reliability are estimated and compared against the mission requirements. In the *architecture design phase*, details of the avionics architecture are developed. Trade studies are employed to consider different architecture options. Detailed requirements are generated for implementation. All estimates and functional correctness of the architecture are verified through design reviews [2].

An experienced avionics system engineer can complete the mission concept and system definition phases in a relatively short time with the help of simple tools such as spreadsheets. This is because these two design phases can tolerate a rather large margin of error in the estimates and because many details of the spacecraft system have not been defined. In contrast, because the architecture design phase involves generating details of the architecture, functional correctness is a major concern. The margin of error in the estimates also has to be

tightened because it has direct impact on the cost and schedule in the implementation phase.

The activities of the architecture design phase lend themselves to using executable architecture models as an analysis tool. Executable architectures can verify architecture functions and estimate accuracy. They can also allow the engineer to explore system dynamics such as interactions between components and time dependent behaviours. However, executable architectures and related development tools have their own requirements. In order to support trade studies, the architecture model should be flexible and modular, made up of components can be easily added, removed, or exchanged while verifying the consistency of the component interfaces. Such flexibility can also support reuse of architecture models. That is, once an architecture model is captured, the model itself or its components can be used to model other avionics systems with only minimal changes. Furthermore, the simulation results of the architecture model should be easily captured and formatted to guide the development of the implementation requirements. Finally, since the architecture model is executable, it should be able to be integrated with other subsystem models to support spacecraft level simulations, which are often used to verify command sequences in mission operations. These requirements suggest a more sophisticated tool than a simple spreadsheet.

The role of ASAT in the spacecraft development process is depicted in Figure 1. The objective of ASAT is to help the avionics system engineers to develop architecture models to meet these needs.



APPROACH

In order to prove the concept is achievable, ASAT is being developed in the context of an X2000-like avionics architecture model. X2000 is a technology program at JPL funded by NASA to develop system architecture and technologies for future spacecraft. The X2000 architecture is highly scalable and adaptable to a wide range of space missions. A typical instance of an X2000 architecture is shown in Figure 2. The ASAT prototype will demonstrate the capability to capture this architecture and then replace components or modify the architecture with minimum effort.

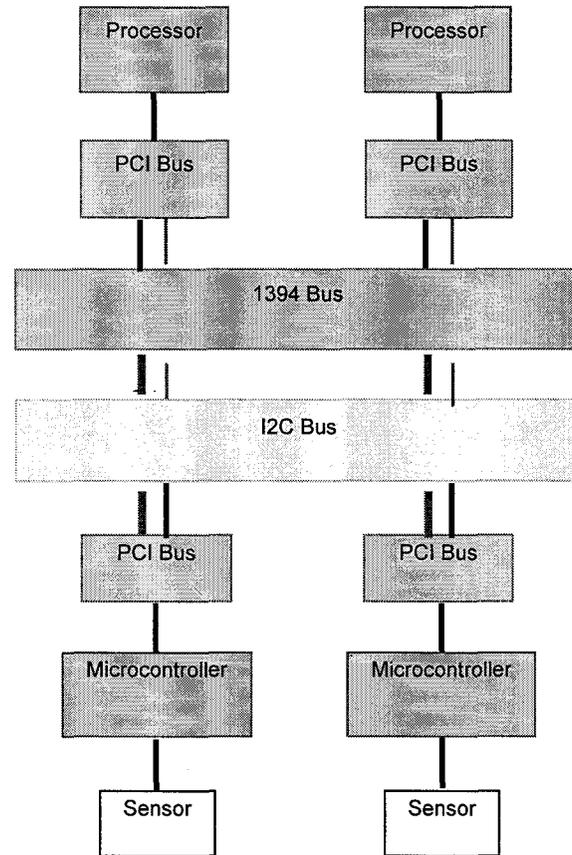


Figure 2: X2000 Architecture Model

In addition to the demonstration effort, ASAT development includes capturing and implementing an array of needs users might have for a system engineering oriented tool suite. Interviews with members of the user community helped develop sets of needs in four main areas: ease of use, capability (vis-à-vis competing tools), validated components, and cost effectiveness. These needs are being analyzed to determine sets of effectiveness measures in each area. These effectiveness measures are, in turn, evaluated to determine interrelationships, rank, and cost. The result being a set of requirements on ASAT.

The approach to meeting users needs in each of the four areas is described in Table 1.

| User Need | Satisfaction |
|--------------------|--|
| Ease of Use | <ul style="list-style-type: none"> Design and implement technologies common to all components that support user concepts of ease of use, including: <ul style="list-style-type: none"> Plug and Play Data Driven |
| Capability | <ul style="list-style-type: none"> Design and implement architecture components of appropriate accuracy and fidelity Design and implement support tools that allow users to do things competing tools do not support Use development tools that support analysis of architecture dynamics |
| Validation | <ul style="list-style-type: none"> Design and develop components based on standard specification (e.g. IEEE Std 1394a) |
| Cost Effectiveness | <ul style="list-style-type: none"> Components are reusable Architectures are reusable Reduce architecture development time Reduce trade study time |

Table 1: The ASAT Approach to Meeting User Needs

ASAT – THE TOOL

Based on feedback from the user group sessions discussed previously, it was clear that ASAT should initially focus on ease of use and cost effectiveness. The team developed two essential features to address these user concerns: plug-and-play and a standard data exchange infrastructure between core X2000 components. Additionally, the team wanted to use an existing System Engineering modelling tool that could provide the necessary capabilities addressed by the users.

The Statemate Magnum tool set from I-Logix allows System Engineers to quickly describe the requirements and specification of their system in a

non-ambiguous graphical representation. This graphical model is a 'formal specification' that represents the functionality (Activity Chart) and behaviour (Harel [1] Statecharts, truth tables, time-continuous charts, flow charts and pseudo code) of the system. Once the model is built, the tool can check the specification for static correctness and completeness issues.

Statemate fulfils the need to build executable models by providing a powerful simulation engine. This allows the engineer to verify the dynamic system behaviour. The tool can run test- or use-cases through the model to fully validate that the system accomplishes its purpose. The tool allows the engineer to interact with the model through Statemate Magnum's graphical panels. These represent user interfaces to the physical components of the system.

In addition, Statemate Magnum provides a documentation tool to produce useful documentation in text form. Finally, the tool can generate behavioural 'C' and ADA code for a number of different platforms.

For these reasons, and the fact that Statemate Magnum had already provided valuable contributions to the System Engineering process at JPL, the team chose Statemate Magnum to be the primary modelling engine for ASAT. Having selected the tool set, the team looked at existing modelling approaches.

Traditionally, the Jet Propulsion Laboratory has adopted a component based approach that required the user to define a set of explicit relationships describing the very detailed interfaces between all components used in the architecture model. Although this approach is not incorrect, subsequent experience has shown it is prohibitively time consuming as it does not lend itself to rapid design trades that take place in the architectural design phase.

A new approach was needed. This new approach utilized a standard interface, communications protocol, infrastructure support functionality, an architecture configuration file, and scenario files to drive architecture simulations.

The standard interface and protocol allow the user to initially ignore explicit details of connecting components. Implementation of the standard interface and protocol raised the possibility of a common architecture for all component models. Every component model in ASAT consists of three layers. The *ASAT_IF* layer contains all functionality to allow plug-and-play between components. The *Architectural_IF* layer contains all interface related functionality required to place the specific component in an architecture, such as parallel to serial conversion. The *Component* layer contains all functionality of the actual component. For example, consider the implementation of the IEEE 1394 serial bus as an ASAT component. The ASAT_IF layer is

as described above, the Architectural_IF layer contains JPL developed hardware to interface the 1394 bus, the PCI bus, and the I²C bus, and the Component layer contains the implementation of the IEEE 1394 standard.

Figure 3 depicts a prototypical ASAT component and communication between the its three layers.

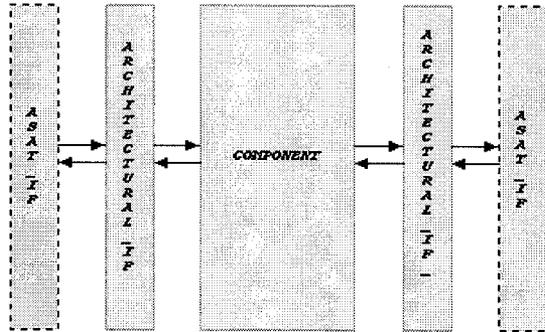


Figure 3: A standard ASAT component implementing the interface layering approach

The infrastructure support functionality provides the user with the development environment in which architecture models are developed and interfaced. The architecture configuration file defines how components are connected to each other. This means that a *usable* architecture is fully defined by a collection of components (the architecture graphical model) and a configuration file. This also means that multiple architectures can use the same graphical model and different configuration files. At the beginning of a simulation run, the components will automatically determine how they are connected and interact with each other. The integrated architecture model will be stimulated by a the scenario file and any measurements generated by the simulation will be captured in a file for future analysis.

This approach makes the process of building and testing an architecture into five steps:

1. Place existing components in the architecture development environment
2. Connect each component to the architecture support infrastructure
3. Build an architecture configuration file describing component connectivity
4. Build scenario files to exercise the architecture
5. Run the scenario files through the architecture

Based on analysis of the simulation results the user may repeat these five steps to consider trade-offs between various architectures.

This approach has several benefits:

1. Reduces the time it takes to build an avionics architecture
2. Reduces the time required for a given trade study
3. Allows component models built in compliance with ASAT standards to be easily plugged into any of our architectures.

The advantage of the ASAT approach is that it allows system engineers to focus on the architecture of the system they are designing and evaluating rather than getting caught up on low-level modelling and tool-specific issues.

System engineers would and should not accept component models unless they were known to be correct. This makes model verification and validation critical. Although the certification process is not fully implemented, it has been developed and is currently being tested. ASAT component models are verified through systematically testing all features and behaviours. Once models are proven to be properly implemented (i.e. they do what we intended them to do), it is necessary to validate that our models yield accurate results. This is done by comparing the results of the simulation against a hardware testbed. The comparison test involves generating many scenarios and running those scenarios through various ASAT-generated architectures as well as their testbed twins. Results are then compared, and discrepancies addressed. Each new component model added to the ASAT component suite is required to undergo this verification and validation process.

RESULTS

The plug-and-play and common data exchange infrastructure has proven to be very effective. Tables 2 and 3 show component development time X2000 architecture integration time. Table 2 shows that the average time to build a component model was about 12 weeks (assuming the IEEE 1394 bus model will be completed in two more months). Experience has shown it would take at least that much time to integrate components into architectures. However, Table 3 shows that the integration time for the PCI and I2C versions of the architecture model took 2 to 3 days. The bottom line: ASAT has shaved months off the time it takes to build an avionics architecture model.

| | PCI | I2C | PROCESOR | IEEE 1394a |
|--------------------------|-----------|---------|----------|------------|
| COMPONENT COMPLEXITY | High | Medium | Medium | Very High |
| INTENDED FIDELITY | High | High | Medium | Very High |
| LEARN/SPECIFICATION TIME | 2-3 Weeks | 1 Month | 2 Weeks | 3 Months |

| | | | | |
|-------------------------------------|------------------|----------|---------|-------------------------|
| BUILD TIME | 1 Month | 2 Months | 3 Weeks | 1 Month (incomplete) |
| LEVEL OF MODELLING EXPERTISE | Very Experienced | Beginner | Expert | Very Experienced |

Table 2: Component Model Development Time

| | Communication among Modelling Engineers | Integration | Test |
|---------------------------------|---|-------------|----------|
| PCI Architecture Example | 0 ^{note} | 0.5 Day | 1 Day |
| I2C Architecture Example | 0.5 Days | 1 Days | 1.5 Days |

Table 3: X2000 Architecture Model Integration Time

Note: The PCI model was used during the development of the plug-and-play infrastructure. Therefore, no learning of the PCI model is required in the architecture example.

NEXT STEPS/FUTURE WORK

Work on ASAT is not complete, and future work involves several important issues.

Although the plug-and-play infrastructure has significantly reduced the time to integrate components in an architecture model, the development of individual models still takes several months. Opportunities to reduce component development cycle time must be found and exploited.

Plug-and-play is not fully developed. Ways to standardize protocols between components in the architecture layer, such as data buffering and handshaking, must be investigated.

ASAT scenario files have been developed to be similar to the output of existing spacecraft simulations at the lab. In order to be more readily accepted, ASAT must be able to be more tightly linked with these simulations

Although a verification and validation process has been created and is being tested, it is in no way complete. A component validation/certification process must be completed, proven, and approved by the user community to garner institutional acceptance of ASAT.

A detailed analysis of the whole range of effectiveness measures in the users' areas of concern is underway but by no means complete. This would lead to a robust set of requirements of an integrated system engineering tool such as ASAT.

CONCLUSION

In this paper, we have described the process of system engineering in avionics design for spacecraft and why a tool that can capture system dynamics is required. We have identified users' concerns about the usefulness of ASAT and responded by concentrating on tool ease of use and cost effectiveness.. The application of our approach in the architecture prototype demonstrates the efficacy of the approach. Results show that plug-and-play reduces architecture development and modification time and the capture and demonstration of system dynamics can lead to correct, highly detailed avionics architecture models. Although a currently a proof of concept, ASAT has demonstrated some promising directions for system engineering at JPL.

ACKNOWLEDGMENT

This work has been funded by the Develop New Product program at the Jet Propulsion Laboratory, California Institute of Technology in association with I-Logix Inc. We would also like to acknowledge and thank Ted Bharami, Mike Dickerson, Wai-Chi Fang, Sue Johnson, Mark Kordon, Carol Lo, Tyson Thomas and Terry Wysocky for their contributions to the ASAT project.

REFERENCES

- [1] Harel, David: Statecharts, A visual Formalism for complex systems in "Science of Computer Programming", 8 (1987) 231-274
- [2] Tooraj Kia and Doug Bernard, Avionics Flight System Engineering, JPL, 1998