Supporting Systems Engineering with Methods and Tools: A Case Study

Jock Rader and Leslie Haggerty

Hughes Aircraft Company and H&A System Engineering

Abstract

Many projects have applied the Hatley-Pirbhai real-time structured analysis method to the definition and analysis of system/software requirements. Application of the method results in the generation of a functional requirements model, which includes data/control flow diagrams, plus process and control specifications. Although infrequently applied, the method also defines an architecture model, which specifies the physical components and the physical interconnect channels for the system.

After first providing a brief overview of the architecture and requirements models, we discuss how the Hatley-Pirbhai methodology is being used by system engineers on a real-time embedded avionics program. We also discuss the set of automated tools used to support the methods, and how both methods and tools were tailored and enhanced. Lastly, we describe operational experience and some difficult lessons learned.

1. Hatley-Pirbhai methodology

Structured analysis techniques were developed in the 1970's to improve the system specification process. The techniques are based on the idea that systems can be more clearly and easily described by using pictures rather than words. Derek Hatley and Imtiaz Pirbhai [1] have built on the work performed by Edward Yourdon [2], Tom DeMarco [3] and others to develop a comprehensive method for specifying real-time systems.

The Hatley-Pirbhai methodology (HPM) separates the system specification into two models: requirements and architecture. The requirements model describes what the system is supposed to do, while the architecture model describes the physical entities in the system and the allocation of requirements to these entities. Examples of HPM diagrams for both requirements and architecture models are provided in Figure 1.

Requirements model. The requirements model combines traditional structured analysis and finite state modeling techniques to define the system function. Data flow diagrams (DFDs), process specs, and a dictionary are all borrowed directly from structured analysis. HPM adds control flow diagrams and control specs to structured analysis to specify the systems finite state behavior. Control specs use traditional finite state elements such as decision tables and state transition diagrams.

In addition, HPM introduces the concept of enhancing requirements. The core requirements of the system which are independent of specific technology are captured in the core model. These requirements are then *enhanced* to add requirements for input, output, user interface and maintenance/self test processing. The goal is to isolate the core model from the technology dependent aspects of the system which are more likely to change as technology advances. An architecture template provides separate regions for each of the four categories of enhanced requirements.

Architecture model. The architecture model is adapted from functional block diagrams and describes the physical partitioning of the system. The architecture context diagram (ACD) shows the physical boundaries of the system. The architecture flow diagram (AFD), shows the physical entities, called modules, in the system. An architecture module may be a system, subsystem, unit, unit module, component, etc., depending on the level of detail of the model. The architecture flows on the AFD show the transfer of information between modules.

The interconnection of modules in the system is shown on the architecture interconnect diagram (AID). The physical channels through which the information passes between modules are shown as architecture interconnects. Architecture interconnects may be electrical wires, busses, waveguide, mechanical connections, or any physical channels used in the system. The type of line used to represent an interconnect depicts the type of channel. The AFDs and AIDs are pairs of diagrams each showing the same modules. Also, the architecture flows and interconnects are included in the dictionary. Dictionary entries identify the allocation of data flows to architecture flows as well as the allocation of architecture flows to interconnects.

The connection between the architecture and requirements models is achieved via a traceability matrix with the help of superbubbles. The system engineer decides what functionality will be accomplished by each module and assigns the requirements accordingly. To assist in this process, superbubbles are drawn around groups of processes, control specs and stores on the enhanced data/control flow diagram (EDFD) to show the allocation to architecture modules. Processes which cannot be fully allocated to a module, must be decomposed further to permit allocation of whole processes to modules. The traceability matrix documents the allocation of requirements to architecture.

Once the requirements and architecture models are complete, the process is repeated for each module in the architecture model. The processing allocated to a module becomes the core requirements model at the next level. The requirements model is then enhanced to include processing necessary for intermodule communication. As with the system, the enhanced processing is partitioned into modules and the whole process begins again. In this way, a complicated system is specified one layer at a time with more detail being added at each level. Ultimately, each requirement is partitioned to either hardware or software.

At each level, the system's functionality is described in the detail appropriate to the level. For example, at the level of a radar system, the detailed processing required to generate an excitation waveform need not be specified. It is sufficient to state that waveforms with certain characteristics are required to accomplish a stated purpose and leave the details to a lower level in the total model.

2. Experience and lessons learned with the Hatley-Pirbhai methodology

For applying HPM to a complex radar system, several adaptations were made to the method. The first adaptation concerns the flows on the AFD. In HPM, the flows between modules on the AFD are the flows crossing the superbubble boundaries on the enhanced requirements data/control flow diagram. These architecture flows are then allocated to interconnects from the AID. A difficulty arises when the modules from the system are expanded into subprojects. The method states that the core requirements for a module are enhanced to handle intermodule communication. The enhancing is necessary when data must be converted to a communication format for transmission across a bus. The result of enhancing is that the data flows leaving a module are no longer the flows shown on the AFD. In fact, the flows allocated to interconnects at the system level do not flow across the interconnects. The enhanced flows from the module subproject are what flow between modules via interconnects.

To resolve this inconsistency, the author made a modification to the method. The flows on the AFD are taken from the enhanced requirements for a module. Now, the flows leaving a module will be shown on the AFD, the

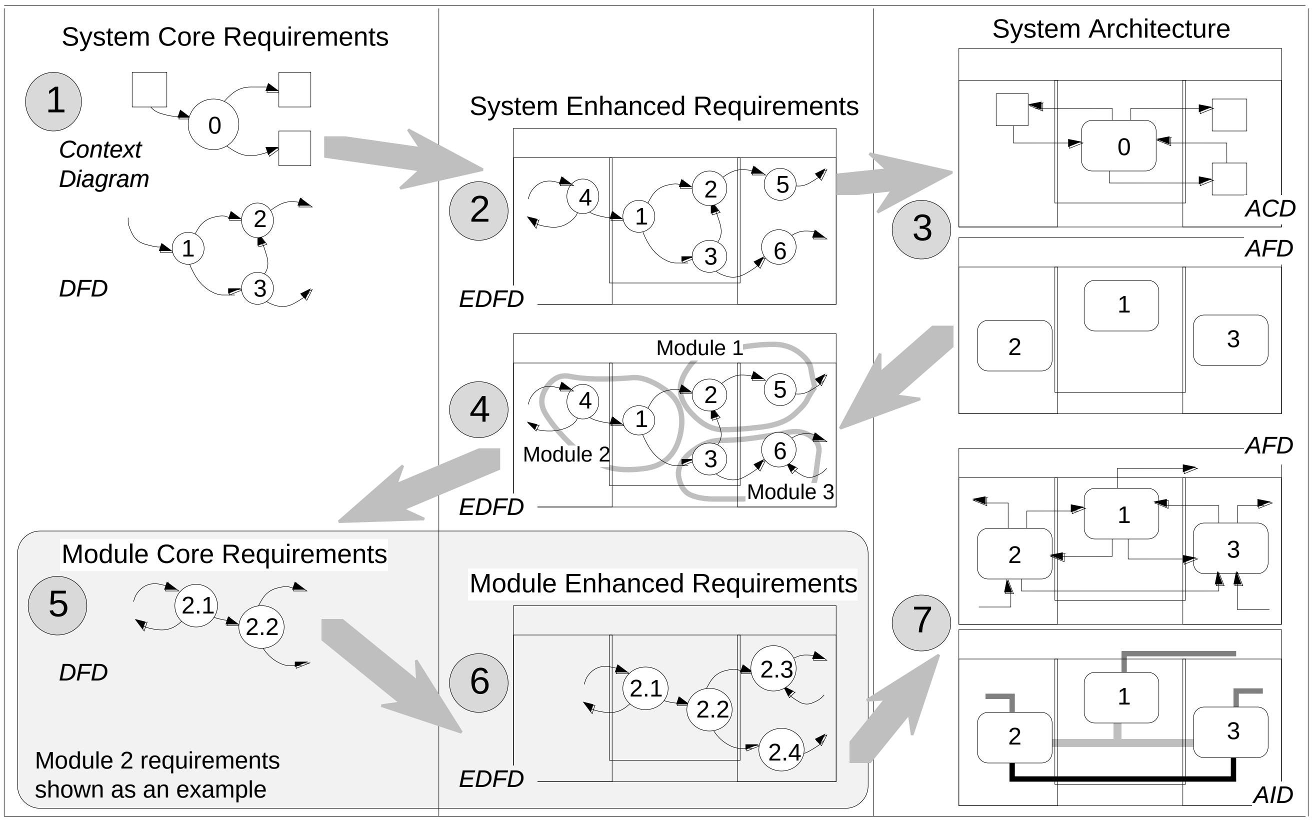

Figure 1 Model development process. Enhancing requirements models for each architecture module before drawing flows on the AFD provides consistent allocation of architecture flows to interconnects.

AFD for a module will balance with the parent AFD, and the allocation of flows to interconnects will remain consistent throughout the total model. The penalty for this modification is that the AFD and AID cannot be completed until the enhanced requirements for the modules are complete. This lag did not pose a significant problem for the author's project, especially in light of the gains in consistency achieved by the modification. The development process for the adaptation, illustrated in Figure 1, is as follows:

1. Develop system core requirements including context diagram, DFDs, CFDs, control specs, process specs and dictionary. (Context and top-level DFD shown)
2. Enhance the core requirements.
3. Determine the architecture modules for the system and draw the ACD. (ACD and AFD with only modules shown)
4. Allocate processes, control specs and stores from the enhanced requirements to architecture modules using superbubbles and the traceability matrix.
5. Draw the core requirements for each module in the system, taking the requirements directly from the traceability matrix. (Module 2 core requirements shown as an example. Processes 4 and 1 from the system are renumbered as 2.1 and 2.2, respectively.)
6. Enhance the module core requirements as necessary for intermodule communication, new user interfaces and maintenance/self-test. (Note: enhanced processing is not required for the inputs to process 2.1 because 2.1 is an enhanced process from the system model.)
7. Draw the flows and interconnects on the AFD and AID based on the boundaries of the enhanced module requirements. (AFD and AID shown)

The second adaptation of HPM was done in the interface between hardware and software. The method states that when requirements are partitioned between hardware and software, the hardware and software core requirements are enhanced to handle hardware/software communication as well as intermodule communication. In radar/processor applications and many others, hardware and software communicate via registers. This mechanism lends itself to being modeled as a store. As shown in Figure 2, using stores alone or enhanced processes that produce flows into stores is an effective technique for modeling hardware/software communication. Stores are especially effective for communicating control flows between hardware and software because HPM prohibits transformation of control flows within a process spec.

In Figure 2, hardware process 1 produces a control flow into store B which is accessed by software process 2. Hardware process 3 is an enhanced process which transforms the output of hardware process 1 into flow A which is accessed by software process 1 via store A. The output flows for software processes 3 and 4 must be transformed in hardware for communication to other modules since software cannot talk directly to an external module. Hardware processes 4 and 5 accomplish the interface processing for software, receiving their inputs via stores.

A final variation from traditional HPM was the addition of an architecture interconnect context drawing. The AID shows the interconnect channels between the system and external terminators. However, the AID does not show the destination or source of the external interconnects. Furthermore, in complex systems, the AID is very busy so external interconnects are not easily identified. The architecture interconnect context diagram is a valuable tool in communicating the system boundary with the customer

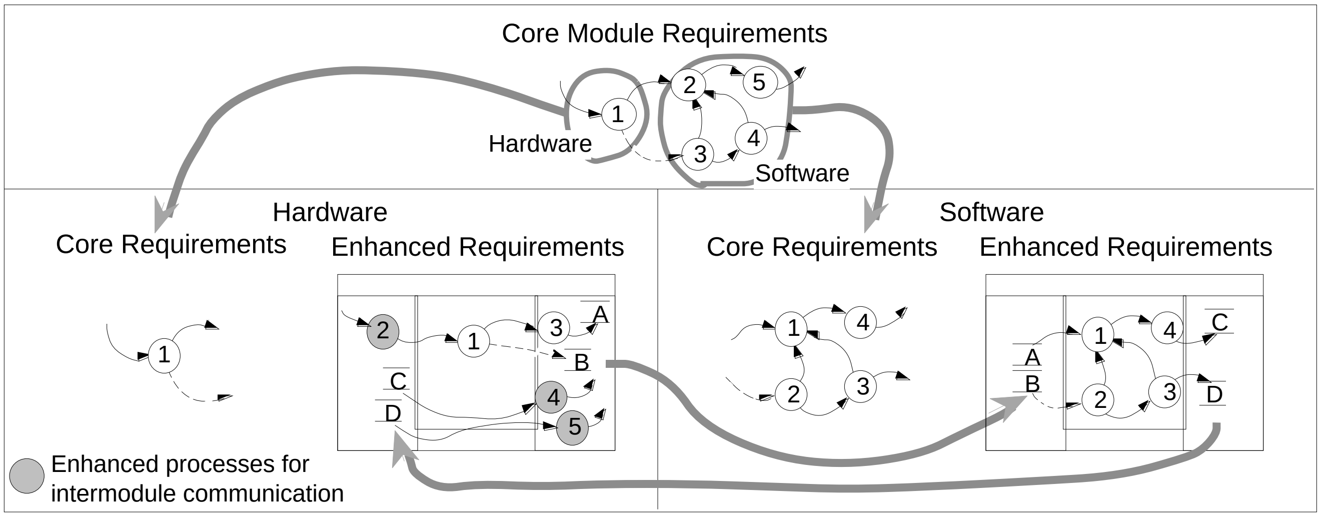

Figure 2 Hardware/software interface modeling using stores. Hardware processes produce flows into stores which are accessed by software processes. Software processes produce flows into stores which are accessed by hardware processes and transformed for intermodule communication.

as it isolates the external interconnections along with their source or destination on one drawing.

3. Commercial off-the-shelf tools

The commercial off-the-shelf (COTS) tools were chosen to facilitate generation of the documents and reports required by the customer, and to satisfy resource constraints. The state of our tools environment during the first nine months of 1994, is shown in Figure 3. The environment is distributed across both Unix and Macintosh platforms, which is a tool integration challenge.

Originally, the project expected to use a Unix tool to support the HPM architecture model and provide easy integration with the requirements traceability tool. Unfortunately, support for the architecture model was not provided when expected. Thus, the decision was made to use TurboCASE/Sys, which supports the architecture model and which is hosted only on the Macintosh. In addition, the requirements database tool, Requirement Traceability Management (RTM) did not support hierarchical requirements definition, so Microsoft Word was used for the initial document development. The role of automated tools is as follows.

Radar requirements specification. The text of the requirements and the boilerplate is captured in Word. The HPM requirements model is captured in TurboCASE/Sys and exported. The graphics are then touched-up in MacDraw and pasted into the specification in Word.

Architecture specification. This specification contains the complete requirements and architecture models, along with supporting text. The same tools are used to generate it as for the previous specification.

Trace reports. The main tool for the various trace reports is RTM, a requirements management and tracing tool. To generate trace reports with RTM, you strip the customer requirements specification into the database such that each requirement becomes a separate object in a class

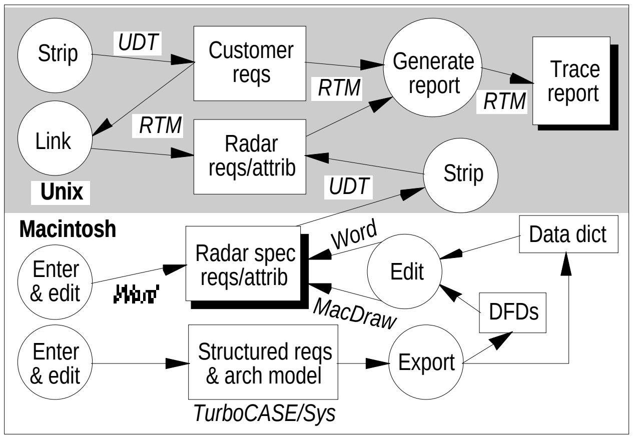

Figure 3 Operational tool use. Both Unix and Macintosh tools support project tasks.

of objects. You also strip the requirements from the requirements specification into a second class. Finally, you establish links between objects of the two classes to show the dependence of Hughes requirements on customer system requirements.

The tool allows you to associate a set of attributes with the members of a given class. Two of the attributes in use on our project are priority and rationale. It now becomes straightforward to generate a report listing all high priority requirements in the customer specification, and to show for each which requirements in the requirements specification combine to satisfy that customer requirement. Additionally, you can include in the report the rationale for the choice of each requirement.

4. User developed and enhanced COTS tools

The nature of the current generation of Computer Aided System Engineering (CASE) tools is such that for operational use on a real project, it is necessary to perform some degree of tailoring, enhancing, and integrating to the COTS tool set. This results in a set of tools and utilities which are either enhanced versions of tools provided by the vendor (enhanced COTS) or new tools written by your staff (user developed). The following examples illustrate the need for the services of a competent toolsmith.

Enhanced COTS tools. Referring again to Figure 3, we focus on the bubble labeled "Strip" in the middle of the diagram. This enhanced COTS tool strips requirements from a specification generated with Word into the RTM database. A very coarse, but configurable, strip utility is provided with RTM. After some training and experience, you can learn how to tailor and enhance the basic utility for most of your own needs. We found to our disappointment that even with help from the vendor we were unable to fully satisfy our needs. Our solution has been to resort to some unattractive manual steps to compensate, while continuing to search for automated means.

A second enhanced COTS example is provided by the bubble labeled “Generate report.” RTM provides a report writing tool, which can pull objects from the database and format them into reports, of the type described in the preceding section. An expert user employs a scripting language to specify the format of the report and the objects and attributes to be included. Several of these report generators have been developed for the project.

User developed tools (UDT). The strip bubble is actually a sequence of three filters, the last of the three being the enhanced COTS tool described just above. The first two filters preprocess the file to be stripped in order to prepare it for the last of the three. For this purpose two small programs were written by our toolsmith. These are simple examples of user developed utilities.

A more substantial user developed tool, not shown in the figure, is called AutoDoc. The planned functionality for AutoDoc is to automatically generate the requirements specification from the RTM database and the HPM tool database. In October 1994, we expect to generate the textual portions of new specifications from the requirements database. This eliminates a major source of inefficiency and error – maintaining the requirements text in two places.

5. Operational lessons learned

Methods and CASE plan. We found the use of a CASE implementation plan to be very helpful. The plan includes the project goals being addressed by CASE, the methods to be used and how the tools will support the methods. The plan provides the link between the program goals and the daily tasks. Without a clear understanding of what the project is trying to accomplish, it is easy to become caught up in performing tool directed tasks which may not support project goals. With a CASE plan, when tool limitations require work-arounds, they can be judged against project goals, which then can be consciously modified if deemed appropriate.

Infrastructure. The tool and method infrastructure needs to be in place and stable before the project begins. Our project staff and our CASE team experienced considerable inconvenience due to our shortcomings in this arena. To accomplish the project tasks, the following infrastructure tasks need to be completed very early:

1. Select tools and host platforms
2. Set up work area
3. Purchase workstations
4. Purchase software licenses
5. Install licenses
6. Develop CASE implementation plan
7. Conduct methods training
8. Conduct tools training
9. Develop CASE tool extensions.

Most of these infrastructure tasks are not technically difficult, however, they are all time consuming and if they are not complete before the project begins, it introduces significant risk into the project.

Management sponsorship. It is important to get management endorsement and support for the use of new methods and tools. Change is uncomfortable for people especially when it means more up-front work. Thus, a strong management stance is necessary to help overcome resistance. It is to be expected that the use of new tools and methods initially will slow down project progress, so schedule must be adjusted to reflect this. However, new tools and methods make a complex task more manageable when used correctly. For database related methods and tools, the databases provide a wealth of readily available information which would not be available otherwise. Management must be committed to achieving the benefits of methods and tools and provide support in overcoming the initial challenges.

CASE team. The CASE team is a group of individuals which is responsible for making the use of methods and tools a success. Some of the necessary roles for team members are as follows.

1. **Methods leader/facilitator.** The project needs a methods leader/facilitator to adapt the methods (e.g., HPM) to the specific application and to help engineers apply methods to the specific application.
2. **CASE infrastructure leader.** Especially during the start up phase, a CASE leader is needed to ensure the infrastructure foundation is ready. The CASE leader is also needed to guide the toolsmiths and focus attention on problems that arise.
3. **Toolsmiths.** CASE tools are still immature, and most tools promise more than they can deliver today. Toolsmiths are needed to tailor and enhance and connect the tools.
4. **Expert tool users.** An expert tool user for each tool provides support to other tool users, communicates user needs to the CASE leader and the toolsmiths, and is a good point of contact with the tool vendor.
5. **System administrator.** Workstation based tools are complex so a system administrator is needed to maintain licenses, set up accounts, make backups, and resolve tool problems with the host platforms.

6. Summary

Structured methods, such as HPM, bring structure and rigor to complex projects, making the project more manageable, higher quality, as well as providing a wealth of project information. Automated tools are required to support the capture of the diagrams and data, and to assure consistency. A CASE team is required to maintain both methods and tools, and to support users. The probability of success is greatly improved by deploying a proper infrastructure ahead of time, developing a clear plan for the use of CASE, providing strong management endorsement, and assigning the right people to the task.

References

[1] D. J. Hatley and I. A. Pirbhai, *Strategies for Real-Time System Specification*., Dorset House, New York, 1988.
[2] E. Yourdon and L. L. Constantine, *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*., Prentice-Hall, Englewood Cliffs, 1979.
[3] T. De Marco, *Structured Analysis and System Specification*., Prentice-Hall, Englewood Cliffs, 1978.