THE MANAGEMENT IMPLICATIONS OF THE MODULAR SIMULATOR CONCEPT

by

James Brown
and
William Tucker
Boeing Aerospace and Electronics
Huntsville, Alabama

ABSTRACT

The Air Force has, with Tri-service support, contracted for research, development and demonstration of the modular simulator concept known as HAVE MODULE. Reactions to the concept, as developed by this program, have ranged from frank disapproval to open acceptance, but the most common is "What can HAVE MODULE do to help me with my problems?" In this paper,an attempt is made to answer this question. A dream of an ideal simulator development program is contrasted to often dismal realities. The contributions of the modular simulator concept that help achieve the ideal are discussed from a practical point of view, with emphasis on subcontracting. Some problems are described that the concept can help avoid and some that it will not. Lessons learned from the application of the concept to the demonstration, which was 75% subcontracted, and other projects in the specification stage are discussed. Recommendations are made for the future HAVE MODULE based programs.

INTRODUCTION

The HAVE MODULE architecture defines a logical grouping of generic simulator requirements into logical modules and defines interfaces between them. A standard hardware interface is provided if needed as in those cases where the logical modules might be assigned to physically separate computers, perhaps of different types. The requirements are grouped, as shown in Figure 1, to minimize coupling between modules, maximize cohesion within modules, and yet closely resemble classic simulator partitioning. The modules are defined in generic specifications and the data interfaces are defined in Ada language compilable constructs. Both are tailorable to any simulator program. This concept can strongly affect the way all phases of a simulator program are conducted. It is the opinion of the authors that careful application of the HAVE MODULE standards will ease each of these phases and reduce cost and schedule risk. In this paper we will present a dream of an ideal program and then describe how HAVE MODULE can help make this dream true.

THE DREAM PROGRAM

In the exciting time when a new contract is received, the front end analysis needed to prepare a strong technical base is often bypassed in the haste to "get on with the program". Long lead parts need to be procured, subcontracts let, drawings released, and software design started. There is no time to do esoteric functional analysis or preparation of detailed interface specifications. We start this job like we have all the others, knowing that there will be integration problems when the suppliers deliver, but rightfully confident that our good engineers will solve them. After all, they did last time.

But this time, we tailored the generic HAVE MODULE specifications to our application during the proposal phase and used them in our supplier negotiations. We updated the interface definitions during technical negotiations and included them in the agreement. Strangely, after contract award we're still busy, but the procurement people complain less and the software manager doesn't show up to complain about systems engineering quite so soon. Soon we finish releasing our requirements and go into the next phase.

Development

As the software designers attempt to implement systems engineering's usually cryptic requirements, they are tempted to once again allow their imagination free rein to design the system they like, but find themselves bound by the Ada language interface definition and the detailed functional specifications. From time to time they gleefully report an interface exception to the program office, but are quickly provided a repaired coordinated interface. When they exercise the static and dynamic test cases provided with the specification, they realize that they have no choice; they have to meet the requirements. They are comforted by the realization that the models they have now turned over for integration are similar to what they have done before and will be a good contribution to their store of off the shelf routines.

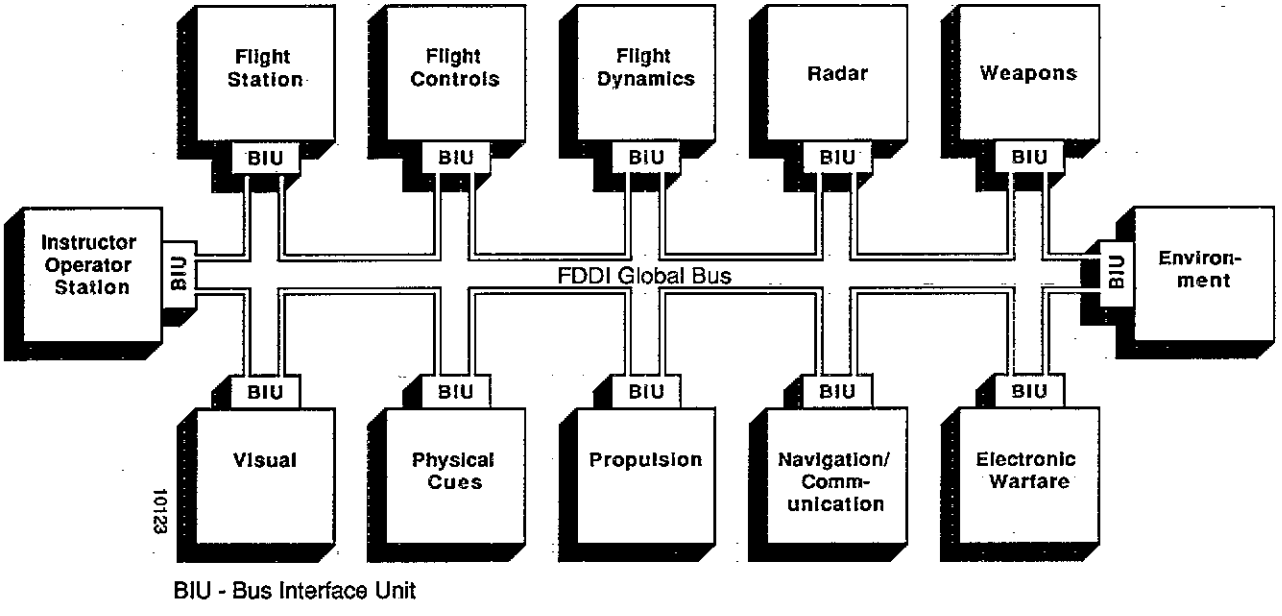

FIGURE 1
Modular Simulator Architecture

Integration

Now is the time for the integration engineers to shine. All of the suppliers and in house builders have passed their ATPs and all of their subsystems are ready to play together. In the bad old days, now we would start wishing we had done better systems engineering. But, since we published our functional and interface specifications at contract award, maintained them through development and enforced them at supplier acceptance, we suddenly experience instant integration. Shocked, we begin test.

Test

After all the successes we have had in our ideal program, we enter into test confident that we will again have a record setting success. However, success at this point is more a function of how well we allocated the top specification requirements, how well we met them, and how well we designed the test procedures to prove them. However, we will assume that the systems and test engineers used some of the time and effort they saved at proposal time and integration time to meet these goals. Therefore our testing proceeded at the same startling pace as our previous efforts.

Conclusion

Now, congratulated by our peers, rewarded by our superiors, and appreciated by our customer, we look back at our ideal program. We realize that we've been dreaming after falling asleep reading a boring RFP. None the less, we plan to closely examine the concept to gain some of these promised benefits. At the office the next morning, resolved to learn more about this approach, we drop by our friendly local HAVE MODULE ISWG representative whose presentation precipitated our dream. When the representative ran down after talking about loosely coupled standard modules and global data busses, we asked "What does that mean to me ?"

"Basically," said our loyal representative, " HAVE MODULE provides you with a standard simulator architecture, comprising twelve separate logical groups, documented in a set of specifications. Included in the specification is a generic interface description, written in Ada, and tailoring instructions which describe how to adapt the provided specification and interfaces to your simulator. Test software is available to test the completed modules along with software tools to manage the Ada language interface and keep the Bus Interface Unit software and test software current with the released interface definition."

"The specification has been successfully adapted to two programs separate from the demonstration project. The lessons learned from those adaptations along with experience gained in the demonstration project is implemented in the tailoring guide as down to earth practical instructions on how to modify the requirements to meet a particular need. The support software is available in an "as is" condition from the Air Force. Use of the software system with a limited set of modules (multiple segments in a module) and limited module sets (like systems without weapons or radar simulations) have

been proposed. Further study is required to validate the use of the concept with multiple crew station simulators. A concept has been developed and documented to interconnect devices using the Distributed Interoperability Standard (DIS) Protocols. Figure 2 shows the research status in these areas."

Technology Goal / Program Component	Conceptual Design	Generic Specification	Tailoring Guide	Demonstration	Independent Verification	
Baseline Partitioning, Interfaces, and FDDI Bus	√	√	√	√	√	
XTP Protocol	√	√	√			
Limited Module Set	√	√	√	√		
DIS Interface Module	√	√				
Multiple Logical Segments in a Physical Module	√					
More Fully Functional F16C	√	1				
Multiple Crew Stations						

1 Application specific specification

M2 '22/45/LS

Figure 2
Have Module Research Status

THE REAL PROGRAM

The research that has been done has demonstrated that benefits are available from the HAVE MODULE concept as shown in Figure 3. These benefits were demonstrated not only by the HAVE MODULE program, but by two independent agencies who tailored the generic system/segment specifications for their particular applications. Phil Peppler [1] of Williams Air Force Base Armstrong Laboratory, and Terry Snyder [2] of Grumman's Simulation/Training Programs provided their experience with tailoring the specifications during the sixth HAVE MODULE Interface Standards Working Group. The results were: The HAVE MODULE architecture allows for a straightforward design and development and is complementary with Ada; It allows engineers to focus on the simulation requirements rather than "specmanship"; It encourages reusable modular designs; It does not force a particular design approach ,but allows design to be determined by analysis, judgement,and resources and; It was a good vehicle for flowing down requirements. The HAVE MODULE concept has also been selected for use on the Army's Advanced Distributed Simulator Technology program.

Tailoring these specifications to provide module level requirements and interfaces at proposal time will greatly ease the pain of program startup by providing a stable engineering baseline. This has always been a goal but the relative ease to tailor generic specifications makes it an achievable goal.

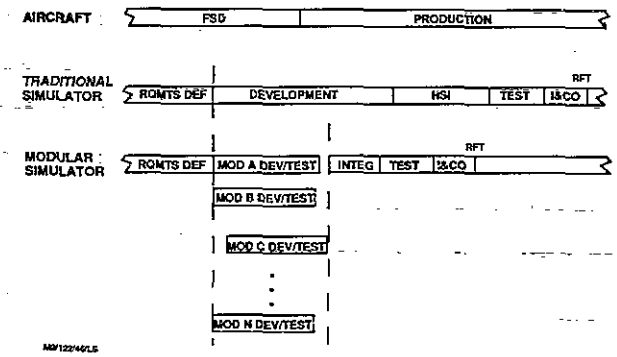

Figure 3
Representative Have Module Schedule

Development

When the development phase starts with well defined requirements and interfaces, the application engineers can focus on the application software design. We found this phase of the program to contain the most critical need for interface meetings/telecons between the module designers. During the HAVE MODULE demonstrator development, the module designers identified a very small number of interface changes. Because the HAVE MODULE concept extends tight interface control deeply into the *program from* the very beginning, a required revision to the interface must be developed and coordinated very quickly. The BIU and the Module Tester software must be revised and recompiled and distributed to the module designers in a very short time. Therefore, special software tools were developed to allow very rapid coordinated *changes to be made to the interface.* Because each change required each module builder to recompile his load, changes were scheduled for block updates. Other than the need to continuously coordinate the interface, the HAVE MODULE concept allows for highly parallel and independent individual development and test of individual modules.

Module Test

The module test phase is the most critical phase of this type of program. A principal advantage of the HAVE MODULE concept is that once a module has passed module test it will integrate smoothly into the system with few problems. The thoroughness of module testing is directly proportional to the ease of integration. This fact became painfully obvious during the integration of the demonstration device. *Some modules required tuning* (flight dynamics) and rework (IOS and weapons) during integration. In retrospect, the test cases for these modules were not thorough enough to adequately test the modules. Test data may not be available at the proper level to qualify a given module, but if the mode and state transitions are tested along with logical *extremes and representative* worst cases, integration will occur smoothly.

Integration

Integration of successfully tested modules can be a rewarding experience. By properly scheduling the modules, you can enter integration with confidence that the modules will communicate without problems. This allows you to focus on typical simulator tuning problems sooner. Even though the HAVE MODULE demonstration program experienced more difficulties than anticipated, integration was still successfully performed in much less time than a typical simulator program (18 days).

Test

At the system level, test of a modular simulator is no different than any other simulator test. However if system level requirements were adequately assigned to the module level and properly tested, the system testing will be painless. One key advantage of the modular architecture is that, once a problem is isolated to a module, that module can be pulled off line from the simulator and placed in a module test mode for troubleshooting while the rest of the device continues integrated testing of unaffected systems.

To assist this isolation process, the HAVE MODULE program created software integration tools that capture messages and data on the FDDI bus. Additional tools such as a FDDI Bus Analyzer and an enhanced IOS data display pages would be very beneficial. A real time debugger would be essential to quickly troubleshoot problems in a deliverable trainer environment.

Using the existing software test tools and the additional tools mentioned above, testing will require less time than traditional simulator programs.

CONCLUSION

The HAVE MODULE architecture can help a simulator development program to reduce cost and schedule and improve supportability by lowering technical risk. It does this by providing an early firm specification and interface baseline readily derived from an easy to tailor generic specification.

REFERENCES

[1] Peppler, Phil Presentation at The HAVE MODULE ISWG #6 July 11-12, 1990.

[2] Snyder, Terry Presentation at The HAVE MODULE ISWG #6 July 11-12, 1990.

ABOUT THE AUTHORS

Mr. James Brown served as the engineering subcontracts manager for the modular simulator demonstration program. He has also performed as engineering subcontracts manager for the United Kingdom E-3 simulator program and on various avionics and weapon systems programs. He received his Bachelors degree in electrical engineering from the University of Arkansas in 1986.

Mr. William Tucker was the project manager on the demonstration project. He has also been project manager for the KC-135 production contract, the UK E-3 project and other programs. He has worked in simulator software and systems design since 1977 after serving two years as a field artillery officer in the US Army. He received a Bachelors degree in electrical engineering from Wichita State University in 1975.