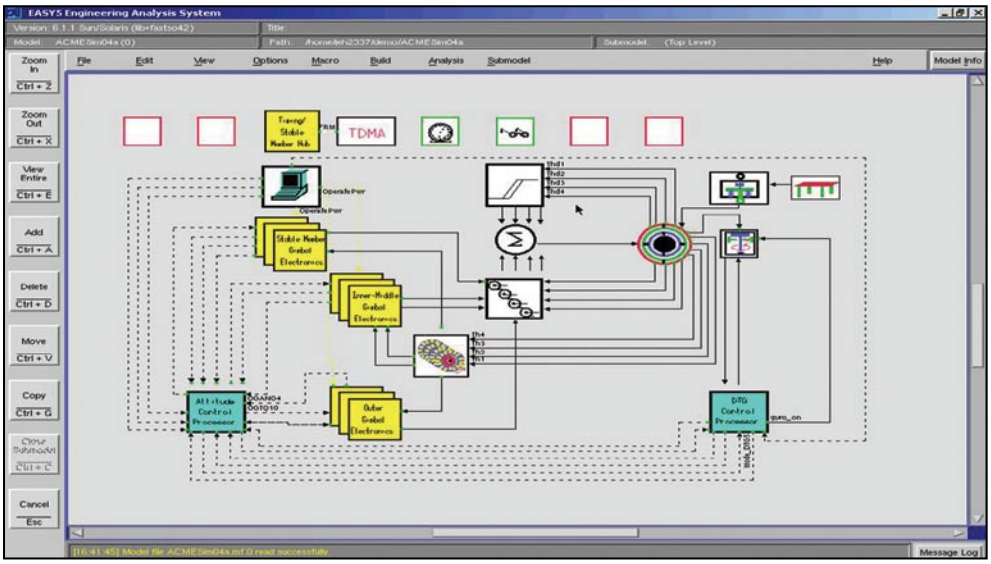
# Keeping Missile Guidance Systems on Track




> *"Our simulation run times have gone from a week or more to as little as eight hours."*

**Customer:**
Draper Labaratory
Cambridge, Massachusetts
www.draper.com

**Software:**
MSC.EASY5™

**Summary:**
Draper Laboratory used MSC.EASY5 to provide the graphical user interface (GUI) and infrastructure for a collaborative Virtual Product Development environment in the development of missile guidance systems. The open architecture of MSC.EASY5 improved the collaboration by providing a direct link to other popular simulation tools, such as MATLAB/Simulink. Using MSC.EASY5, Draper Labs reduced the high costs of testing very complex electromechanical systems by cutting simulation time from weeks to hours.

Physically testing a missile guidance system is time-consuming and expensive. Add to this the need for collaboration between prime contractors and subcontractors, and the process becomes very complex and even more time-consuming. According to John Harrison, Task-Lead, Virtual System Simulation, Strategic Systems at Draper Laboratory, "The problem we faced was the high cost of testing very complex electromechanical systems. We needed to find ways to do validation and verification testing of our software without building a real device and testing it in the field. MSC.EASY5 was selected to provide the graphical user interface and much of the infrastructure for our virtual product development environment."

Guidance system software is very complex, utilizes multiple processors, and must respond in very harsh environments. The main goal was to provide a Virtual Product Development (VPD) environment using system simulation for software developers to write and completely test software, leading to a zero-prototype requirement. Harrison says, "We have created a simulation environment which allows software developers to verify that their implementation of control algorithms is correct. In addition, the environment supports our software validation and verification testing in that we can execute the actual binary image of our flight software as if it was running in a real processor. To achieve this we used, in part, the "timed-switch-state" feature in MSC.EASY5 which allowed us to accurately and efficiently model the digital portion of our system and still use variable-step solvers for thecontinuous system dynamics."

The simulation uses very detailed processor models, accepting binary code like an embedded processor and replicating its behavior. The rest of the system can be modeled around the processor. When a simulation is run, the software developer can actually look at how many clock cyclesit takes to execute the code.

"The instruction set simulators are slow because they include a 25MHz clock," says Harrison. "Initially we were trying to run simulations with two processors and a plant model. This gave us run times of about 120 seconds per second of simulated time, which was too long. We couldn't afford to wait almost a week for the results. We had to find a way to execute these simulations in the 30- to-1 range. As a result of architectural changes to our infrastructure and vendor-provided enhancements to our simulation tools, our simulation run times have gone from a week or more to as little as eight hours."

MSC Software

*"I can take a relatively inexperienced person and in 10-15 minutes show them how to exercise a simulation of a system and have the results presented to them in a useful way."*

Because designing a missile guidance system is a collaborative process, there are several types of engineers involved in the simulation-based design process. Harrison says, "We have engineers at different firms involved in this project. We are modeling a very complicated system and our dynamists are representing their equations in their favorite languages, including Simulink, MATLAB, FORTRAN, C, and MSC.EASY5. We developed a way for the engineers to model in any language they want and to integrate the model into one system simulation environment."

The embedded software engineers are the users of the VPD environment. Many of the programmers involved with the guidance software have little, if any, knowledge about system simulations. But they all seem to have one thing in common - the desire to use simulation. Harrison notes, "I can take a relatively inexperienced person and in 10-15 minutes show them how to exercise a simulation of a system and have the results presented to them in a useful way."

Draper Laboratory's subsystem model development process requires the modeler to maintain a certain level of documentation and engage in peer reviews. Using this process, the VPD software environmenthas been evaluated against the Software Engineering Institute's (SEI) Capability Maturity Model and met the requirements for practicing at SEI Level 3. "We have a process for assembling system simulation, where there are formal requirements for what the simulation must do," explains Harrison. "We look at the requirements and pull pieces of the system from a repository of validated models. We have lots of models with varying levels of detail for a given problem set."

Collaboration between engineers at different subcontractors and locations throughout the United States, all needing access to the simulations and data, requires a great deal of control, and configuration control is especially critical for collaboration. The VPD environment at Draper Laboratory has an automated process for reporting problems and recording change requests. The VPD environment utilizes a tool called PVCS Dimensions (Merant), built on top of Oracle (Oracle). It is used for traditional version control paths, checking data in and out. It also allows program management to assign roles to users and provides both a work breakdown structure and a tool for describing how a project should operate. Users are assigned roles at any given point in the process. "You can automate the execution of a process," states Harrison. "For example, the problem reporting process starts with an unapproved report appearing in my 'in box.' In many cases, I determine that the problem needs to be solved and send the report to the team leader responsible for building the simulation. The team leader gets the problem report and then opens up a change request that's related back to the problem report. Whoever is assigned a developer role for the part affected by the requested change also gets notified."

"The developer then checks out the files relative to a change request, and so on," he continues. "So there is a complete history of everybody who was notified and everything that was done. In addition, developers can only do things in response to a change request that's been assigned to them. For example, if a developer is driving home one day, and remembers he forgot to put something in the design, he can't put in the next day unless there is authorization."

Probably the most difficult task users face is modeling how the embedded software responds to power abnormalities in the system. This is tested by running a system simulation to a specific point in time, stopping the system, perturbing the operating point to change the state of the system, and then restarting the simulation. When the simulation is restarted, it has to be very accurate, as if the simulation had never been stopped. "If I run a system simulation for 10 seconds, I have to be able to simulate the model for 5 seconds, stop it, and a week later pick it up at 5 seconds and go to 10 seconds.

Then take the two plots and connect them together," says Harrison. "They have to be exactly the same as if I had done a 10 second simulation straight through. More importantly, I need the ability to modify any part of the stopped simulation's operating point prior to restarting the simulation, as you can always integrate a simulation to the operating point you really need to be at. It may sound trivial, but it is very difficult to do this. We're pushing the technology a lot in this area and have had good success in developing a fully restartable simulation environment with state manipulation." Computational speed is another critical component of Draper Laboratory's VPD environment. With two processors, simulations were running at a ratio of 120 to one, which is considered very long. Each additional processor simulator bumps simulation time up 50 CPU seconds. Mr. Harrison said, "We were facing 300 CPU seconds per second run times, which were unusable. Now we run the simulations on a Sun Fire 6800 shared-memory machine and have cut our simulation run times down by an order of magnitude as a result of distributing processes and improvements in our simulation infrastructure. To simulate our full system it would have taken over a week, now we can do in about 8 hours."

Harrison concludes, "Simulation is an art. You constantly face problems that demand clever solutions, more computing power, and/or higher-fidelity modeling approaches. The problem set facing Draper Laboratory is pushing the limits of technology in a number of areas and the tools we have chosen are an important part of our solution path."

**Corporate**
MSC.Software Corporation
2 MacArthur Place
Santa Ana, California 92707
Telephone 714 540 8900

www.mscsoftware.com

**Europe, Middle East, Africa**
MSC.Software GmbH
Am Moosfeld 13
81829 Munich, Germany
Telephone 49 89 431 98 70

**Asia-Pacific**
MSC.Software Japan LTD.
Shinjuku First West 8F
23-7 Nishi Shinjuku
1-Chome, Shinjuku-Ku
Tokyo, Japan 160-0023
Telephone 81 3 6911 1200