

On Your Six

Spring 2000

D-Six Newsletter
Bihrl Applied Research
www.bihrl.com

We would like to welcome everyone to the first issue of our D-Six newsletter. The purpose of this publication is to keep our users and other contacts up to date with the latest developments and applications concerning D-Six, since the software has undergone a lot of evolution in the past few years. We hope that you will find this informative and useful, and look forward to any comments and feedback.

Table of Contents

- D-Six Synopsis
- Newest Developments
 - PC graphics
 - VME Interface
 - Network Interface
- Application Spotlight: Pilatus engineering simulator
- Operator and Programmer Chat (Tips, stuff on the horizon)
- Commentary: Power to the People!

D-Six Synopsis

Since this is the first issue, and some of the readers may not be aware of the current iteration of D-Six, we wanted to do a recap of the software and it's capabilities before launching into a review of some of the newest features. For those of you who are already familiar with everything – go on to the new stuff!

Anyway, as most of you hopefully know, D-Six is a fully model independent simulation environment for the PC platform. Originally developed as a tool for model development and validation, the efficiency of the database operations code, along with the rapid pace of PC hardware development, has allowed the software to evolve into many different **real time** simulation applications. D-Six is now in use in virtually every type of flight simulation application, from desktop simulation development tool, to flight model host for hardware in the loop simulation, to flight model host for full cockpit simulators.

The reason that this simulation development tool has been able to evolve into this wide range of applications starts with it's original design as a structured, fully object oriented, native C++ code application. Rather than an adaptation or rewrite of an older simulation application, this fresh start approach has enabled D-Six to take full advantage of the modularity and code re-use this

design philosophy offers. As a result of full integration of the Windows Component Object Model (COM) specifications, D-Six enables the user to dynamically link language independent modules to provide a broad range of simulation functionality. These range from easy integration of legacy model code to graphical, run time assignment and manipulation of hardware interfaces. Because of the simulation's structure, the addition of new functionality or hardware is easily accommodated with new modules or device drivers as these are developed or the needs arise. The bottom line to all this (and the overriding purpose to the software's deployment) is that models developed on the desktop can propagate smoothly and easily to and from virtually any aspect of analytic or manned simulation. There are no D-Six legacy models – the modular integration of the latest features easily allows every model ever developed in D-Six to evolve to whatever simulation requirement the user has.

Another important issue that arises in the use of anyone's simulation environment is the structural requirements of the environment on the model itself. Several other simulation applications require the structuring of the simulation model to their predefined model structure and format, which can lead to difficulties in the propagation of models

if certain simulation capabilities are not easily supported. Because BAR originally developed D-Six as an in-house tool for use in the simulation development and support services that we provide to the industry, we have been required by the wide range of model formats and structures encountered to make D-Six as flexible and transparent to these requirements as possible. As a result, a user can use all of D-Six's model development tools in a native hosting of their model in D-Six, or they are free to include all (or portions) of their simulation code and data as a compiled library, with no changes to their model. This approach has been used to easily host models as old as a 70's version of the engineering simulation of the McDonnell F-4, as well as the flight controls of the F-18E/F, and a complete AV-8B operational flight trainer simulation. This important capability is another one of the software's key features that separates it from the other simulation applications available.

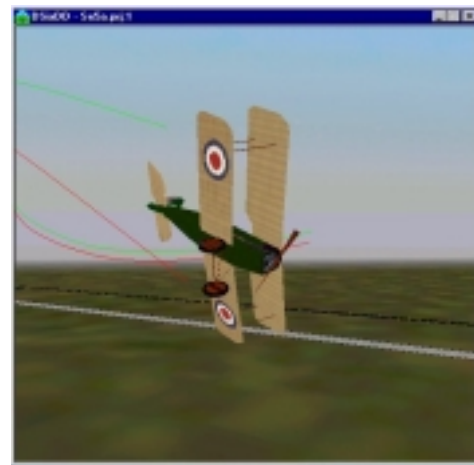
While BAR has been focusing significant development efforts on the hardware interface capability and it's functionality in recent months, the basic environment has also been undergoing considerable update. Virtually all the visualization interfaces have been revised and updated for greater functionality, including a new graphics application that optimizes the latest PC based 3-D graphics accelerator cards for a wide variety of vehicle visualization capabilities. If you'd like more detailed information on any of the simulation's capabilities or applications, please call for additional info or a demo, or check our current web presentation at www.bihrie.com/dsix.htm. By the way, we are putting an all new web site together specifically for D-Six, www.d6sim.com, that will have many more features and a more detailed presentation of the software. We'll send you a notice as soon as it's up and running.

New Developments

P.C. Based 3-D Graphics

D-Six has always supported the deployment of external out-the-window graphics in the past, either through the use of the Primary Image family of PC based 3-D graphics cards (i.e., P10, Piranha, Barracuda, etc.), or through the interface with other external image generators such as SGI and Evans and Sutherland graphics hardware. The downside of even the Primary Image solutions was the expense of the hardware that kept external graphics visualization from many users. Because of the explosion of PC based graphics hardware and capability, we wanted to move the ability to utilize this important visualization feature to any user that had a current 3-D graphics card installed. In 1999, the state of the art PC graphics cards had progressed to a point that a \$200 card could be expected to provide an acceptable level of external graphics capability at a minimum 30hz frame rate. BAR initiated a development program to access this capability, and as the graphics power has since accelerated, these low cost cards are poised to provide an astounding level of capability.

What the graphic capabilities of these cards has done is to allow D-Six to offer as part of the



basic environment, a windowed graphic interface that essentially replaces the old icon based simulation view that came up in the engineering screen. This new window is highly configurable to allow the user to view the aircraft from numerous external views as well as an out-the-window view. The external views are dynamically configurable, with one of the Gods eye view slewable from the keyboard or mapped hardware device. Configuring the display window terrain, external view configuration and other parameters are all handled



through a tabbed interface (e.g., fig_). The airplane external view models can also be dynamically loaded, and can have actions mapped to appropriate features of the model, such as moving surfaces, gear retraction, etc. The out-the-window view can be configured from a number of supplied terrain configurations, or can be customized (more on that later). The user can activate wing tip streamers for flight path visualization, and can opt to display the velocity vector on the external view as well. Many other dynamically configurable options are available as well.

The beauty of this system is the efficiency of the graphics code and its performance on current hardware. Using Pentium II and III in the 300 to 450mhz range, and a current 3-D card (not the top of the line) we are getting a fixed 30 to 60hz frame rate in the windowed graphic presentation, **while running the simulation software**



concurrently on the same platform. This includes the massive F-18E/F simulation that we like to quote, with approximately 1.4 million tabular data points and over 30,000 lines of FORTRAN code for the flight control system alone. The effect of having the graphics and the simulation running on the same platform is virtually zero graphics latency. This is definitely a good thing.

While this windowed graphics presentation will be included as part of the basic D-Six simulation environment, BAR is also completing the development of a full screen graphics mode that will utilize the full monitor (or projector) area for display of external graphics. This mode is actually more efficient than the windowed mode, because the graphics no longer has to keep track of the screen window configuration. This allows much more complex graphics, and we've been playing with importing National Geologic Survey Digital Elevation Measurement (DEM) data for terrain, as

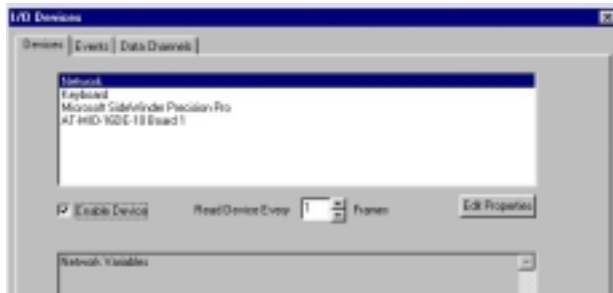


well as adding other graphics features (see figure below for an example of DEM terrain data for the Luray, Virginia area). Another feature that using Windows 98 as the operating system provides is a multi-monitor capability. Windows will support up to 9 graphics adapters, so we can give users a highly flexible graphics display capability. This expanded graphic capability will be available as an add-on module to interested users. If you'd like to see some actual recorded examples of the graphics in use, call or e-mail (graphics@bihrl.com) for a free CD-ROM with some MPEG files showing the graphics used in several simulation applications

I/O Devices Interface: VME Hardware

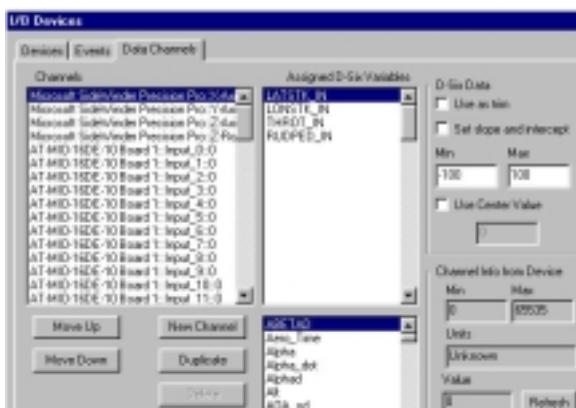
Several years ago, BAR developed an Input/Output Device (IOD) interface that removes the need for the user to develop and embed custom hardware interfaces and controls in the simulation code. The IOD interface has been very successful in giving the simulation user complete interactive control on his assignment of simulation variables and events to any hardware device attached to the system. This hardware mapping can be saved as part of the simulation project if desired, or can be dynamically loaded and modified during the simulation session. This dynamic reconfiguration allows a single set of hardware devices to become truly reconfigurable, with no downtime as the simulation is reconfigured or modified as part of the user requirements. BAR has been supporting the evolution of hardware devices and interfaces

BAR has developed a number of device drivers at this time, ranging from digital and analog I/O PC cards to drivers for a wide range of digital



through the development of device drivers that report their available input output channels or states to IOD for the assignment of user selectable variables. The figure below shows an example of

and analog PC input devices. The most recent development came as BAR needed to interface with an electrically loaded set of controls for the NAVAIR 4.3.2.4 reconfigurable cockpit. This SCT control loader system was hosted on a PC that communicated through a VME bus controller. In order to allow D-six to communicate to the control loader system, BAR had to develop an interface to the IOD system for the selected PC-to-VME communication hardware.



the selection and configuration of the device I/O channels for mapping to selected simulation variables, in this case a game joystick. Another interface allows the user to map and control simulation events as well, such as switch settings for flap deflection or simulation restart, etc.

BAR selected the SBS bit3 hardware family (www.bit3.com) for the PC-to-VME hardware interface, specifically the PCI Model 673 interface card paired with the VME Model 616 chassis card. The 616 card was configured with a 1 meg shared memory window and connected to the 673 card with fiber optic cable. A device driver was developed for the 673 card and a card configuration interface that allows the user to set up and map variables between the loading system. This interface has been successfully used to develop and demonstrate a number of reversible and



irreversible control loading simulations in conjunction with high fidelity flight models and graphics, all run and controlled by a single PC. Again, the dynamic loading capability of D-Six project and configuration files enables a rapid and seamless reconfiguration from one simulation model to another.

The net result is that BAR can now offer D-Six users a VME interface to a local shared memory

D-Six Network Interface

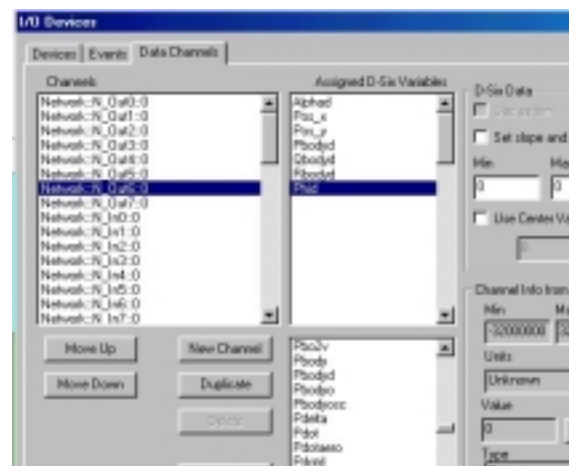
In the conduct of several projects, the requirement to operate multiple simulation stations in a common environment. A typical application would be the evaluation of multiple UCAV simulations, the evaluation of the tactical capability of a piloted configuration in combat with other vehicles, or the modeling of an air traffic control situation with actual piloted simulations interacting with a controller station. In any of these situations, the need to connect at least one simulation to another with user controllable awareness is required.

In order to enable these types of applications, BAR developed a D-Six network module that enables the interaction of many separate instances of D-Six in a common local area network. D-SixNet is designed to minimize system latency when supporting the high fidelity, real time models that D-Six has always allowed. In addition, the network interface was also designed to enable the easy manipulation of the network parameters for quick hosting of network simulation object, and maximum flexibility in the set up of communication variables.

D-SixNet works across a Local Area Network/Ethernet environment and, with the use of high speed network base code, allows for network speeds in excess of 200 Hertz and update speeds in the tens of milliseconds and lower. In addition, the network is created to be as easy to configure as possible. The base network is an add-on module to the D-Six product. Once the network module is loaded, the module works with D-Six's internal IOD interface (see discussion above for description of IOD) to allow for the network variables to be configured as easily as you can configure a joystick or keyboard. Configuring the data passing between network simulation objects can be done at the click of a mouse. The Network additionally supports code-based means of accessing internal data. So not only can you easily read and write in variables

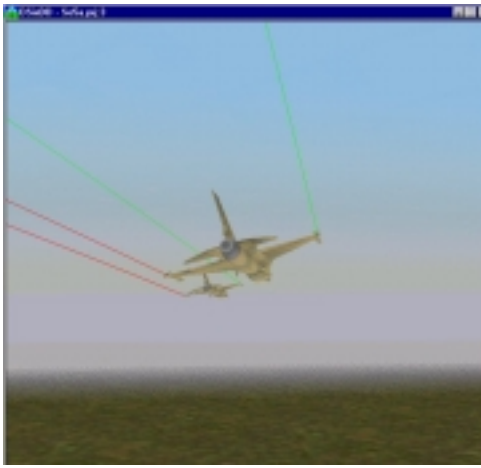
configuration. Since this is the defacto standard of many of the large scale simulation platforms, D-Six users can now easily configure and utilize their PC based simulations as flight model hosts and controllers of complex simulation hardware. For SCRAMNET users, BAR has already demonstrated compatibility with this hardware interface as well.

with a mouse click, but you can also code in internal network structures to your D-Six Project.



The D-Six network is based off low level UDP networking protocol operating in a client server configuration for maximum performance and network control. This configuration allows for reduced network over head and during maximum network throughput. The D-Six network has been tested at 200+ Hz without loss of data or significant latency. In fact, the network can run faster- it's currently only limited by the frame rate of the D-Six Simulation Objects that operate over it as it runs.

The network module was originally designed for close formation UCAV simulation. With multiple UCAV's flying together, high frame rate networks are required for development and evaluation of group and individual control schemes. Without the high performance of the network, along with the ability to dynamically map the simulation variables of network objects, scenario development and evaluation of close formation UCAVs becomes difficult to impossible. The D-Six network allows for absolute minimization of network latency such that virtually no dead reckoning is required for the prediction of vehicle positional information on the



network, a situation unacceptable in the evaluation of close formation work. Obviously, this system performance is highly effective in other network applications that require high network performance, such as air-to-air tactical evaluations of aircraft and

weapons, formation flight simulation and re-fueling tasks, etc.

The flexibility of the D-Six network has also been used for real time data display and review over multiple computer systems. Using a specialized D-Six simulation project called SimView, it is possible to establish a remote viewer for observation of the network simulation objects from a base station. This application allows the display both 3D visualization (from external views to pilot's out-the-window view) of the Simulation Object along with real time strip charts or other plotted data. This allows for multiple stations in other locations to observe the same exact simulation object(s). Go to [___](#) for a detailed description of how SimView was used to evaluate and plan the flight test program of a general aviation manufacturer.

Product Application Spot Light: Pilatus Aircraft Engineering Simulator

In order to highlight some of the diverse simulation applications that D-Six has been used in, we'd like to present a review of particular installations in this section of the newsletter. Anyone interested in having their application included in this section, or with questions concerning the presented application, please let me know.

Pilatus Aircraft, located in the shadow of Mount Pilatus in beautiful Stans, Switzerland, has been building fine trainer, STOL and transport aircraft for over 60 years. The need to assemble the required elements of an in-house reconfigurable engineering simulation capability led them to the review of D-Six as a candidate for the simulation environment. D-Six was chosen as the flight model host environment for a number of reasons, ranging from cost to application flexibility. Since D-Six is optimized to operate at real time on the cost effective PC platform, Pilatus was able to extend their budget to include such additional components as a Fokker electric stick loader (www.fokkercontrol.com), and SGI based external graphics. These were integrated into the software to provide a high fidelity simulation environment for the pilot. The flexible model development and model hosting features that D-Six provides expedited the development and deployment of multiple simulation flight models. The simulation and model complexity that Pilatus was able to develop in a very short period of time (particularly when viewed in a typical simulation development

time scale!) was such that they were able to accurately demonstrate the complex spin characteristics of a particular trainer application. The fidelity of the simulation, along with the simulation refinement they were able to achieve in the time and money expended, was a testament to the company's engineers, as well as to the flexibility of D-Six. The fact that this was done with no previous company simulation experience is even more noteworthy.

Besides providing configuration development and flight test support, the simulation capability is also used to demonstrate the flight characteristics of their aircraft to visitors as well. Potential future updates include the change to D-Six native PC graphics (the SGI will go to CFD duties), and the continued incorporation of avionics hardware and software for even greater fidelity of the operational experience. Please check out the company's web site at www.pilatus.com for more information about this excellent airframe developer, and look for pictures of the simulator under the Engineering tab.

Operator and Programmer Chat

In this part of the newsletter, we'd like to provide you with some insight to what we're continuing to develop, any technical issues or comments, and some presentation and/or discussion on techniques developed by our in house users. Feedback is especially welcome!

D-Six Info:

- We are currently under a Phase II with the Air Force to develop a number of new D-Six capabilities. In support of a multi-UCAV (Uninhabited Combat Air Vehicle for civil types) simulation capability development, D-Six will be evolving new multiple model hosting and interface capabilities that will provide a range of new features for all users. In addition, new tighter integration with Matlab model development tools, more extensive than the current set provided in D-Six's Aeroport model porting tool, will also be a result of the development work conducted under this SBIR. Other new capabilities supporting this update are also under development. We'll keep you informed of the future developments.
- BAR is working with a local company, AeroTech Research (ATR) to try and incorporate some of their atmospheric modeling capability as a module in D-Six. ATR has been conducting research into high fidelity modeling of atmospheric phenomena for many years for NASA and other government agencies, and has developed a number of models to provide highly realistic turbulence and wind shear encounters. These models provide mathematically appropriate 6-DOF effects that are added to the vehicle as U,V,W and moment perturbations, not like most atmospheric encounter "effects" models that are currently in use. We hope to be making these high fidelity, computationally efficient models available as add on modules for our customers soon.
- A completely new set of plotting analysis and editing tools are currently being beta tested in our office. These new plot interfaces will dramatically expand the plotting functions with many new configuration and editing features, all exportable to any Windows software. Virtually all of D-Six's plotting interface will be updated, and the dynamic editing capabilities will be further expanded with loadable editing features (w/ a provided wizard so users can develop their own). We'll be featuring the plot update in the next newsletter, and this package will be available to all supported users as a free update. Keep your support contract active!
- BAR is investigating the potential of putting together a hardware package that will essentially be a turnkey 3-channel external graphics package. It will include the PC based D-Six graphics capability (see presentation above) hosted on a 3 graphics board PC that will be capable of interfacing with any other simulation output using Ethernet or other communication interface. This very low cost hardware solution would come with several generic terrain packages and would provide 60hz, external texture mapped graphics that could be plugged into D-Six or any other sim application for quick, inexpensive three channel viewing. Coupled with 3 monitors, projectors or panels, the user could have highly functional 3 channel application quickly up and running for demo, developmental or specific application purposes. Other specific models and terrain could be provided if desired. Envisioned to sell in the \$6 - \$10 k range for the hardware, we think that this could fill a niche in the market. If you have any feedback on this, please let us know.

Tech Tip:

Some D-SIX simulation projects may require specialized models and functions that perform the same tasks performed by D-Six model independent functions. These tasks consist of atmospheric computations, aircraft equations of motion, initialization, simulation looping, etc. The functions may be replaced with user-defined functions by including new code in the model dependent dynamic link library.

Users are encouraged to use model template code as a starting point when creating user-defined functions. These templates are located in `\ModelTemplate\Advanced Source` in the DSIX directory. The

template code in this directory contains identical source code to the default functions used by D-Six. A list of these functions is provided below.

Template File	Function
Atmos.cpp	Computes atmospheric parameters.
InitMainLoop.cpp	Initialization code for MainLoop function
MainLoop.cpp	Simulation loop and equations of motion
MainOverdriveLoop.cpp	Overdrive function call stack and computations
R_k.cpp	Runge Kutta integration
Set_init_val.cpp	Initial conditions for computations

To replace one of the functions defined above, the user includes the appropriate template file in the MS Visual Studio that generated the model dependent dynamic link library. Once in the project, the user can modify code as needed. In order for D-Six to recognize the new user defined code, the function name must be added to the model definition, or `Mdepdll.def` file. One application of creating and using a user-defined function to replace a D-Six default function is provided below.

User Defined Initial Conditions: Replacing `Set_init_val()`

With the use of a user defined `Set_init_val()` function, a user can define specific initial conditions for a particular simulation models. `Set_init_val()` is called prior to every simulation run. The default function executes a series of computations that ensure units consistency between default variables found in the initial condition dialog box.

Normal simulation operation allows the user to add or delete variables from the default list of variables that appear in the initial conditions dialog box by simply changing variable properties in the **Edit>Variable List** dialog box. Clicking the radio button adjacent to the desired property modifies the property of a variable. In addition, once the initial condition variables are chosen, the user may also provide user defined code to perform specific operations to set initial conditions for the simulation.

As stated above, this code must be added to the users D-SIX project in the MS development environment. A template for the `Set_init_val()` can be found in the **Advanced Source** folder. In addition to adding the `Set_init_val()` template to your project, the user must modify the `Mdepdll.def` file in the project, by adding `Set_init_val` to the function list. This registers the user-defined `Set_init_val()` with D-SIX.

Once in the D-SIX project, the user can modified the template code as required. An example of this is provided below here. The example code, listed below, allows the user to define the initial velocity of the aircraft one of three ways, defining Mach number, calibrated airspeed, or true airspeed. In doing so, the algorithm calls several user defined functions to perform the appropriate computations.

```
void Set_init_val()
{
    // Initial condition hierarchy for velocity
    // Mach      Mach Number
    // Vcal      Calibrated Airspeed kts
    // True      Airspeed
    if      (Mach != 0.0f) CalcVcalVt();
    else if (Mach == 0.0f && Vcal != 0.0f) CalcMachVt();
    else if (Mach == 0.0f && Vcal == 0.0f && Vt != 0.0) CalcMachVcal();
    else{
        Mach = 0.6f;
        CalcVcalVt();
    }
}
```

Commentary: Power to the People

This is the part of the newsletter that I really looked forward to, since a forum for an opinionated engineer is always welcome. For my first editorial, I wanted to look back on the “bad old days” of simulation and try and visualize how it will change.

I started my career as a co-op at Johnson Space Flight Center in Houston, moving to General Dynamics in Fort Worth Texas after graduation. In both cases I worked in Vehicle Dynamics (or something similarly named), primarily analyzing the stability and control of various configurations. One of the things that immediately struck me at both facilities was the inability of the various simulation users and developers to get convenient access to the simulation themselves. At JSC and GD both, the various project engineers would assemble the required datasets needed to build the model, “throw it over the fence” to the simulation engineers who would mechanize the model and the data in their arcane code and computer interface, and then send you back validation cases to check the mechanization. Our use of the simulation went downhill from there.

The level of frustration with this division of labor was evident in that each group had their own reduced versions of the models in their group just so they could get stuff done. This became particularly apparent when the first desktop computers began to arrive – my lead engineer built an entire shuttle landing simulator that ran in 64k of HP 9820 (early desktop computer for you younger guys) memory and chugged along most of the night to complete a single run. The fact that he and other co-workers ran their own nose gear unstick, performance, handling qualities, engine, flight control, etc. models on whatever was locally available was not just a testament to the anal retentive characteristics of engineers. This was primarily a response to the lack of access and easy use of the “official” simulation of the particular airframe, and their inability to use this environment for developmental purposes. Sure, some of us persistent young engineers with lots of time and no social life actually decoded and became proficient at using a given simulation, but recognition of that capability usually meant that you’d be working on that simulation for the rest of your career. What became obvious over time was that the lack of easy access to the simulation and its databases forced the proliferation of other separate models were aligned with the “official” simulation only under great pain (this in itself is the subject for a future diatribe).

This division persists to this day. The complexity of the latest aircraft and their models have further isolated the full fidelity, complete model to particular groups, and access remains a difficult issue for many reasons, ranging from configuration control to hardware costs. Never the less, the current computer hardware and software evolution has really developed to a point that many of these issues can be addressed or eliminated. Obviously, PCs are cheap and incredibly powerful, but the real advantages occur in the development of new programming languages and architecture that enable the integration of model component modules. This technology, in use in many of the current PC consumer applications, greatly simplifies the dynamic linking of component modules into a core environment. Used in the aerospace simulation field, this technology will allow users to develop the particular simulation module (the engine model and control components, for example) and link them into the complete – or partial – simulation. In this form, the developers can update and modify their pieces as desired, and using the latest configuration control software, propagate the updates, if desired, to the rest of the simulation users. In this fashion, all the developers will have more direct control of their portion of the simulation, and will always be operating in the presence of the most recent and appropriate simulation modules. But unlike other earlier attempts at “modularizing” simulation, this new approach will force few, if any, conventions on the developer. Using common interface structures such as Windows Component Object Model (COM), these components can be language independent and portable to other applications or simulation models. The models themselves will become encapsulated structures that can be propagated easily to all users, alleviating the “model diversity creep” that typically occurs as a project ages and only the diligent bother (or are funded) to update their simulations. Other simulation users, such as training, tactical evaluation, and operational analysis, typically farther down the simulation food chain from the engineering simulation developers, will benefit from earlier availability of the most recent models and the increased fidelity available. Exchange and update of the components or the models themselves will become much more transparent.

You don’t have to be a child of the 60’s to see where this is going. With the same effect on distributing development and access that the introduction of desktop PCs had on the centralized computational facilities, distributed simulation development and application will ultimately provide more “power to the people”, in this case the engineers. This wouldn’t be a product newsletter if I didn’t mention the fact that this has been the developmental philosophy behind the evolution of our D-Six simulation environment, nevertheless, this is the way simulation will evolve. Let us help you get there.