TRIAKIS CORPORATION

Using a Virtual System Simulation Environment for the Development of Avionics Systems & Software

– A Triakis White Paper –

For further information, please contact:

Ted Bennett
Triakis Corporation
Tel: (425) 558-4241
Fax: (425) 558-7650
Ted.Bennett@triakis.com

or

Paul Wennberg
Triakis Corporation
Tel: (425) 861-3860
Fax: (425) 558-7650
Paul.Wennberg@triakis.com

Table of Contents

1 INTRODUCTION

Triakis Corporation was founded in 1993 to offer companies greater efficiencies and improved quality in the software (SW) development process through the application of IcoSim©, a unique environment-simulation tool developed by founder Paul Wennberg. Triakis finished its first complete avionics embedded system simulation in 1995 and has since created numerous additional simulations for commercial- and military-avionics development efforts. A key factor in the success of IcoSim© has been its ability to run the actual avionics software object code on the simulated avionics CPU. This has enabled SW developers to achieve significantly higher levels of SW validation and verification (V&V) than economically feasible using traditional methods of V&V.

Being a flexible, hierarchical tool, IcoSim© has proven itself to be useful as a system-level concept development and validation vehicle. With a demonstrable, validated system design modeled in IcoSim©, all of the information and detail necessary to describe the design has been captured. Forming the basis of detailed design, the system modules developed in IcoSim© unambiguously described module-level requirements for each of the components functioning within the validated system. Through this realization, the Triakis team has developed its concept of concise, accurate executable specifications derived directly from demonstrable, validated avionics system designs.

1.1 Background

Traditional methods of aircraft- and avionics-systems development employ arduous, expensive and time-consuming processes relying with varying degrees on a variety of tools from manual analysis, algorithmic modeling and simulation, and limited SW development to support concept proofing. Much of this process is performed using disparate tools by a variety of people, often working in separate departments. As a result, the process of large, complex system-development efforts is highly error prone – requiring major efforts to validate and verify overall system architectures.

The problem is further complicated by the fact that once a system design has been validated, the design of each individual module must be accurately captured in manually generated system specifications. This opens the door for the introduction of additional errors both in the process of translating the design into an English-language format and in the unintentional creation of ambiguities open to interpretation by the implementation team.

Upon completion of the module, typically the avionics line replaceable unit (LRU) development, its functionality must be validated and verified that it conforms to the originally intended system design. Despite the best efforts of, and time invested by development and verification teams, system and SW faults are frequently found late in the development process and often during integration testing when they are most expensive to correct. It is not uncommon for SW faults to remain undetected until after the end product has entered production and is in service with a customer. As a general rule, a SW fault detected after a product has entered production and service is 10

times more costly to fix than when discovered during integration testing, and a hundred times more expensive to fix than when detected during SW module testing.

As the complexity of systems and SW has grown, the time, manpower and equipment required for thorough V&V testing has grown to represent a sizable portion of development budgets. It is precisely these increasingly problematic methods of error-prone system specification and V&V that IcoSim© was developed to address nearly 10 years ago.

In an effort to make significant improvements in quality, while reducing the costs of the avionics systems and module development, the concept of executable specifications (ES') has been widely discussed as possessing the greatest potential for realizing these goals. An ES is intended to provide a standardized means of accurately and unambiguously communicating subsystem behavior to another group so that the subsystem component is developed exactly as the system designer specified and fits as intended into the overall system. Having the ability to make sure that subsystem specifications are complete and accurate before being distributed can eliminate the costs and delays inherent in the cycle of correcting problems due to specification errors and misinterpretations.

1.2 The Future of Aircraft and Avionics Systems Design, Validation and Verification

The new paradigm in aircraft- and avionics–systems development incorporates a primary software tool enabling the system architect to develop new system concepts, simulate them and validate them all on his PC. At this level, communication protocols, message traffic, data throughput, display formats, human factors, control algorithms, interaction between electrical/hydraulic/mechanical systems and all other system-related issues can be conceived, tested and validated. Test scripts are created or derived from real-world recorded events to exercise the system in a manner closely mirroring real-world scenarios.

In many cases, new system designs incorporate subsystem elements that have been developed and proven in previous designs. Once a subsystem element has been proven, it becomes a standard library component available for use in subsequent or collateral designs. Library components may also be used to form the basis for derivative designs where modified or extended functionality is desired. Subsystem elements in the library, managed under a configuration control system, are available for use by other parties involved not only in this development effort but in the development of other systems incorporating like functionality.

Once a system design has been thoroughly tested and validated, each simulated subsystem element is used as an ES for dissemination to the party responsible for developing the real-world counterpart (e.g. an avionics LRU). The ES and any test scripts used specifically to validate its subsystem element form an ***unambiguous*** description of the part functionality. Combined with a brief textual description and additional standard detail intentionally omitted from the simulation (e.g. ARINC 600 4MCU package, connector type, spare ports, processor and memory-reserve

capacity, environmental specs, etc.), the full requirements are captured in a consistent, concise manner thereby eliminating interpretation errors.

The component developer uses the same simulation environment as the systems designer and creates a design in the simulator that matches the functionality of the ES. This part simulates the hardware and runs the actual executable software object code that will run on the avionics subsystem being developed. When the test scripts developed during the system-validation phase are used to exercise the detailed component simulation, the same results should be produced as those generated using the original system-level module from which the ES was derived.

2 AVIONICS DESIGN AND DEVELOPMENT WITH ICOSIM©

As with conventional approaches, an IcoSim©-based design begins at the system-level with the group responsible for the overall system design (usually the airplane or avionics manufacturer). The system designer creates a simulated system comprising one or more interconnected executable specifications as system size and complexity dictates. With a suite of functional system tests created or derived from real-world recorded events, the ES-based system design is then tested and debugged until the system architecture is validated and functionally verified. Once validated and verified, each individual ES forms an unambiguous functional representation of the system element that it simulates.

The ES contains all of the functional requirements from which the embedded avionics SW will be developed. At this point, a hardware (HW) architecture is selected that will support the functional and reserve-capacity requirements of the module, as well as other requirements such as built-in test, environmental, etc. The selected avionics HW design is then simulated in IcoSim©, forming the basis of the detailed-executable (DE) module and providing the platform upon which the embedded avionics SW is tested and verified. Key to the success of IcoSim© is the ability of the DE to be used in place of the ES in the system simulation. The embedded avionics SW executing within the DE is tested in the actual environment in which the ES was developed, using the same suite of functional system tests.

Executing the actual object code developed for the avionics HW, system simulations using the DE should perform virtually the same as the ES it replaces. There is no better method presently available to verify avionics component and system functional performance.

2.1 The Executable Specification

The ES concept has been applied to a variety of engineering disciplines such as ASIC design, SoC design, industrial design and avionics design, but its definition will vary depending on the realm in which it is used. In the avionics realm, an ES is a program that runs in a simulator, with the functional behavior of an avionics box or LRU at the

level of detail necessary to support interconnection with the rest of the system (other ES') and perform all of its functions. A truly useful ES must employ the following key elements:

- Unambiguously specify the functional performance of a bounded system or subsystem
- Be directly replaceable by a DE module (i.e. plug-in compatible)
- Define the functional behavioral model with a high-level language
- Function in a part-oriented, hierarchical simulator

The subparagraphs below examine these elements in greater detail.

2.1.1 Bounded System or Subsystem

The ES specifies the functional characteristics and performance of a physical piece of equipment that will be designed and built. The ES, therefore, must be limited in scope to the boundary monitored or controlled by the physical equipment, i.e., signals flow in and out of ES part boundaries, and information within the part should not bypass its boundary to appear in other parts. Enforcing the boundary rules of a part is an important aspect of the IcoSim© simulation allowing different implementations of a part to be plug-in replaceable. This mimics the physical world where the same boundary discipline must be applied to the design of avionics LRU systems.

2.1.2 Plug-In Compatible

Of prime importance to the success of the IcoSim© ES concept is the ability of the DE to plug into the system simulation in place of the ES. The DE part is simply a higher-fidelity implementation of the ES part executing real avionics SW object code in a simulated HW environment that, together, have been designed to produce the ES functional characteristics. This feature enables testing of the DE in the same system environment in which the ES was developed, using the same test scripts and conditions, and interacting with other ES' or DEs (as they are developed), thereby verifying that the DE has been correctly designed. Because the DE runs real avionics SW in a high-fidelity simulation of its host HW environment, the subtle variations in signal timing, data latency and response times that occur in the actual HW are reproduced, and their impact on overall system performance can be assessed.

The ability to easily verify the module design within the IcoSim© system simulation virtually all but eliminates system functionality related unit rejections at the integration verification & test phase of the development project.

2.1.3 High-Level Language

Another essential element of a successful ES tool is that it must support ease of ES component creation in a manner that prevents ambiguities. The IcoSim© ES concept achieves this through the use of C++, a widely accepted high-level programming language through which the behavior of both the ES and DE are encoded. As IcoSim©-based ES projects progress, function and class libraries are created and expanded, thereby increasing the ease of use and

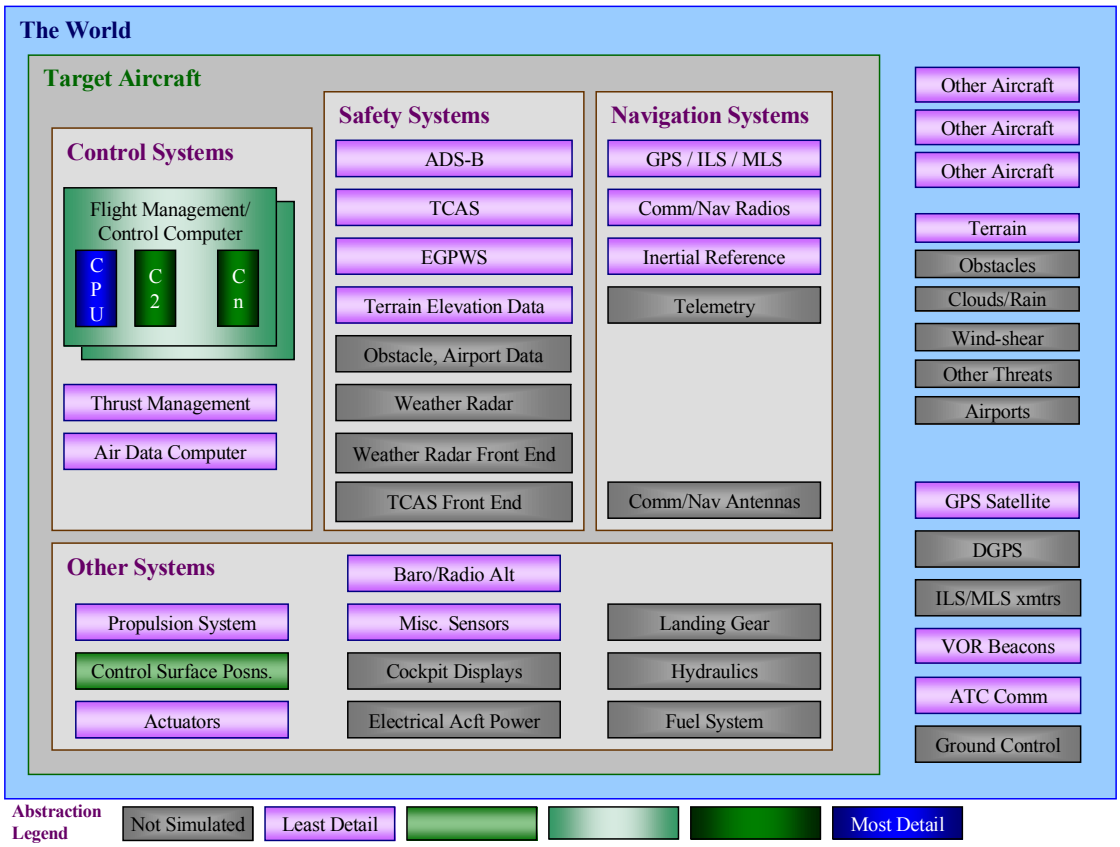
shortening the development time of subsequent projects. The object-oriented nature of IcoSim© is a natural extension to that of Microsoft Visual C++, making the learning process a relatively easy step for those familiar with C++ or C.

For the system designer not familiar with C++, it will be necessary to coordinate some parts of the system-design activity with a programmer, or avail themselves of Triakis' reasonably priced simulation support services. As ES parts are developed and validated, they are added to the parts library for use in other simulations where needed. Triakis has several avionics ES parts available such as a Digital Flight Data Acquisition Unit (DFDAU), a Digital Flight Data Recorder (DFDR), etc. Triakis may be able to negotiate the licensing of additional avionics ES parts such as a Proximity Switch Interface Unit (PSEU), Slat/Flap Control Computer, Power Control Panel, Power Switching Unit, etc., whose development has been independently funded by avionics companies.

2.1.4 Part-Oriented

A viable ES tool must be part-oriented so components can be developed, validated and cataloged for reuse. This eliminates wasted time recreating functions for new system designs that have already been developed and tested on

Figure 1: Example of IcoSim© Simulator Hierarchy



previous programs. IcoSim© was implemented with an object-oriented architecture for precisely this capability as well as a host of other benefits conferred by this approach. All models are based on a class called "Sim," and all system parts created are of "Sim" type, thereby enforcing this part-oriented approach.

System implementations must be hierarchical to allow for the ability to cleanly contract or subcontract parts of functional components as size, complexity and schedule demands necessitate. At the top of an IcoSim© system simulation exists "The World," a part consisting of subparts such as "Airplane," "Weather," "VORs," "Terrain," "Airports," etc. ad infinitum. The part "Airplane" might comprise subparts such as "Avionics," "Landing Gear," "Control Surfaces," "Engines," "APU," "Electrical Power," "Fuel System," "Hydraulic Power," etc. Each of these parts, in turn, might consist of zero to 'n' subparts as required to develop a simulation within which the system under design will interact. For example, avionics parts may contain subparts implementing data-bus protocols (e.g. ARINC 739, MIL STD 1553, TTP, ASCB, CSDB, etc.) at the system level, or microprocessors, memory, and other integrated circuits and discrete components at the HW level. The diagram shown in Fig. 1 illustrates one possible example of how a hierarchical simulation might be implemented for the purpose of developing a Flight Management/Control Computer module.

2.2 The Detailed Executable

With all of the functional requirements validated, verified and unambiguously described within the ES, it then becomes the basis from which the embedded avionics SW will be developed. A HW architecture is developed to meet all of the functional and reserve capacity requirements of the avionics module (or LRU), as well as other requirements such as built-in test, environmental, etc. While an avionics HW design is being created and simulated in IcoSim©, a SW design is created and much of the high-level protocol and algorithm code can be developed concurrently with the development of the simulator. The HW simulation forms the core of the DE module and provides the platform upon which the embedded avionics SW will be further developed, tested and verified.

In order for the DE to function within IcoSim© as a SW-development environment, the target HW must be simulated with sufficient fidelity to execute the SW object code being developed. The most challenging part of this task can be implementing an accurate simulation of the microprocessor itself. Triakis has addressed much of this challenge by developing simulations of many of the popular microprocessors in use today such as the Motorola PowerPC™, MPC 555, MC68000, MC68332, DSP56005, DSP56302, DSP56309, and Intel 80196, 8051, 8096, 8097, 8798, et. al. Triakis adds new parts to the IcoSim© parts library as they are developed and makes them available through the Triakis Web site. A more complete listing of IcoSim© microprocessor and supporting interface parts that have been developed can be found at www.triakis.com/CodedPartList.asp.

Once the simulation of the microprocessor is complete, simulation of the remainder of the design proceeds fairly quickly with many of "glue" parts being available on the Web site listed above or in the IcoSim© SW-developers kit.

The parts library includes some of the standard avionics interface chips such as the HI8282 ARINC 429 transceiver and the Honeywell ASCB interface, for example. Also provided are a variety of standard and generic supporting parts such as power supplies, A/D and D/A converters, logic gates, bus drivers and latches, RAM, ROM, diodes, resistors, capacitors, inductors, transistors, relays, motors, etc. The completed HW-design simulation will feature the same inputs and outputs as the ES, making it plug-in compatible with the system simulation in which the ES was developed and verified. For the design team interested in developing in-house IcoSim© expertise, Triakis offers reasonably priced simulation support services for going on-site to develop a custom simulation together with the design engineers.

With the HW-design simulation complete, each engineer developing the embedded avionics SW now has a complete Virtual Systems Integration Lab (VSIL) available on his or her individual computer. IcoSim©, acting as a virtual microprocessor emulator, can be interfaced with an integrated debugger/editor (IDE) application to create a seamless interactive simulator development environment with full symbolic debugging support. IcoSim© has been tested and used with the Green Hills IDE compiler as well as the Altium TASKING PowerPC™ EDE and CrossView Pro debugger. SW modules are written and first tested on the simulated HW using built-in emulator and debugger controls along with simulated oscilloscopes, waveform generators, etc. SW modules can then be tested using the entire system, or selected elements of the system environment, with test scripts exercising the specific functions that the module was designed to address. Engineers developing the SW need not wait for limited laboratory resources to exercise and test their code.

Formal SW verification testing with structural coverage analysis is conducted next in support of certification. The DE is complete when the embedded SW has been fully developed and verified, and passes all of the functional tests used to validate the ES. At this point, the DE can be used interchangeably with the ES in the system simulation and can be added, along with the ES, to the IcoSim© parts library for use as required on other projects.

2.3 Software Verification

What makes IcoSim© such an effective tool for SW development and verification is that the entire world in which the target SW is being developed to interact with is simulated relatively easily from the highest system-abstraction level to the minutest HW-component level as required. At the core of this IcoSim© virtual world is a fully functional simulation of the target HW into which the SW object code itself is loaded and executed, interacting with the virtual world through simulated I/O HW exactly as it would in the real world. Verification at the CPU level is ultimately the only way to ensure that the actual SW implementing the ES functionality is thoroughly tested with all the attendant delays and constraints inherent in execution at the object-code level.

2.3.1 Module-Level

There is no better way to verify SW functionality and performance than to run it in the environment in which it was designed to interact. Assembling a real-world avionics environment (i.e. simulation/integration laboratory) is very expensive and creates a resource bottleneck as programmers vie for testing time. Using an IcoSim© virtual world, however, every programmer has access to a copy of the virtual avionics/aircraft/etc. environment at their desktop computer in which to develop and test their SW.

As with the system-design verification level, test scripts remain an important part of the development process at the SW-module level. SW modules are written to implement individual functional requirements specified in the ES and then debugged and verified through the use of low- and later system-level test scripts to verify that the developed subsystem-element functionality performs as dictated by the corresponding ES requirement.

The Triakis simulation environment uses C++ for creation of low-level component- or function-exercise commands in support of SW-module development and debugging. The resulting C++ exercise commands are combined as required into automatic tests that are linked with the simulation library and executed for debugging and verification. In order to enhance the use of the virtual environment as a test vehicle, IcoSim© comes with simulations of standard laboratory test equipment such as oscilloscopes, signal generators, etc., along with the functional debugging capability of microprocessor emulators. Access to every part and node in the system is available for determining the state of any part, controlling part behavior, and controlling fault injection. The simulator also has control of time to a very fine level and can stop the entire system at any point to gather and analyze data.

Verification to DO-178B level A requires documentation showing that each conditional jump instruction has been fully exercised, that there is no unused code, etc. In support of reducing the testing and documentation effort to meet this requirement, IcoSim© can be configured for automatic generation of a variety of reports such as, but not limited to the following:

- Modified Code Decision Coverage (MCDC)
- Path Coverage
- Timing and Throughput Analysis
- Reserve Throughput Analysis
- Memory Marker (identifies unused variables)
- Memory Reserve
- Subroutine Interrupt Trace
- Execution Trace

Iterative generation and analysis of these reports throughout the SW development process will ensure that the software is well tested, and that the embedded-SW engineers and simulator-part writers have a matching understanding of the system. When groups working on different aircraft subsystems run the simulator with their

combined work, overall system understanding is enhanced and system incompatibilities are quickly found and remedied.

2.3.2 System-Level

Since SW-module development is performed in the same IcoSim© VSIL environment designed to create the ES, most of the system-level verification can be performed concurrently with module-level verification. As SW modules are developed to implement specific functional requirements in the ES, they can generally be tested as a partially functioning DE using existing test scripts created to exercise the related function at the ES level. In this manner, the programmer verifies that his or her code actually implements the ES requirement correctly. When the programmer is satisfied that the SW module functions properly, it may be passed to the systems engineer for further evaluation. When all SW modules have been developed, integrated and tested together, the DE is complete and can be used interchangeably with the ES in the system simulation.

Test scripts remain an important part of the development process to verify that each developed subsystem element performs equivalently to the corresponding ES function. All test commands and scripts created in support of the DE development are typically saved and used during final testing and for certification support documentation.

2.4 Systems Integration

Prior to receiving actual HW, the systems integrator receives the DE from the avionics developer responsible for implementing the ES. The systems integrator then replaces the ES from which the DE was developed and runs the system using the same set of functional system tests originally used to verify the system design. Passing this test is proof that the avionics SW and HW design conforms to the functional requirements of the ES. Ultimately, the systems integrator will run the simulation with all ES' replaced with their corresponding DEs to verify that any timing or other irregularities introduced in individual DEs do not create undesired consequences when interacting at the system-integration level.

The ability to catch problems in a system simulation using DEs before avionics are delivered for testing results in a great deal of time, effort and money saved by both systems integrator and avionics vendor. Further, product quality is significantly improved and the certification effort is substantially reduced.

2.5 Acceptance Testing

With avionics-SW and simulated-HW functionality verified through DE testing by the systems integrator, all that remains for final acceptance of the product is verifying that non-ES requirements have been met. This is accomplished through traditional means of analysis and laboratory tests to verify that environmental, reserve-capacity, mechanical, etc. requirements have been met. Finally, all system-avionics components should be

connected together in the systems integrator's laboratory to verify that the end products perform the same as their corresponding DEs. The few problems likely to be encountered at this stage should be minor, causing little or no program impact.

2.6 Certification

2.6.1 Simulator

Certification of the simulator created with IcoSim© is relatively straightforward since the simulation does not actually generate any code used in the final avionics product. The FAA requires that all test tools used in support of avionics development comply with the verification requirements of DO-178B §12.2 as a SW-verification test tool. Tool qualification data must be provided per DO 178B §12.2.3.

The FAA also requires that the test procedure demonstrate all functionality of the part or system being certified. In order to support this requirement, Triakis' simulator-testing process compares real HW breadboards to the simulated DE part behavior. Further, when a new CPU simulation is being developed, a full and complete suite of tests is used to verify the CPU instruction set. Triakis typically performs the following tasks for verification of the simulators it creates:

1. **Simulator Configuration Control:** The simulator SW is kept under configuration control during verification.
2. **CPU Instruction Coverage:** The CPU simulation is verified by exercising all instructions in all addressing modes and boundary conditions.
3. **CPU Interface Simulation:** All CPU interfaces simulated are verified by automatic tests that exercises all control and monitor functions for each simulated part.
4. **Cross-Environment Demonstration:** Selected software is demonstrated on both the simulator and a representative target platform. For cross-environment demonstrations, Triakis is responsible for conducting the test on the simulator, and Triakis' customer is responsible for providing test data from the HW platform. Analysis of cross-environmental demonstration test data is the joint responsibility between Triakis and its customer.
5. **Test-Data Configuration Control:** All simulator test data is kept under strict configuration control and archived by both Triakis and its customer.
6. **Problem-Report Tracking:** Triakis maintains a list of problems reported during the use of the simulator.

2.6.2 Avionics

While the documentation requirements may vary according to the level of avionics criticality, the following RTCA DO178B documents would typically be required whether or not IcoSim© was used in the development of the subject

avionics: SW Design Document (SDD), SW Requirements Document (SRD), SW Test Procedure (STP), SW Test Report (STR) and the Version Description Document (VDD). The STP and STR rapidly become the focus of most certification scrutiny because they document how and to what level of detail the avionics functionality was tested. These two documents reveal in detail the thoroughness and rigor with which the IcoSim© simulator has been able to verify that every line of code has been tested and traced to a requirement in the ES. Virtually all the contents of these documents are automatically generated by the simulator, thereby saving a great deal of time in the process of preparing for the certification process.

Without exception, every avionics project (or subproject) on which Triakis has applied IcoSim© for development and V&V use has passed OEM acceptance testing with no functional-problem reports, and easily received FAA certification at DO178B level B, C or D (depending on the project). IcoSim© has been developed and used for V&V to the requirements of DO178B criticality levels A, B, C, & D. Following is a sample list of avionics-development project simulations for which Triakis has used IcoSim©:

Avionics Systems Simulated to Date Using IcoSim©:

- Hawker Horizon Landing Gear Control & Indication System
- Crane Aerospace Weight & Balance System
- A380 Doors & Emergency Slide Sensing System
- B747-400 Proximity Switch Electronics Unit (PSEU)
- Bombardier CL-604 PSEU
- Northrop B-2 Proximity Switch Logic Unit
- Boeing 717 (MD-95) P5 DSP (sub-unit of PSEU)
- Boeing 757-300 P5 DSP (sub-unit of PSEU)
- Boeing 757-300 PSEU {Framework}
- Boeing 747X DC Standby-Power Power Module
- Lockheed C-5 Slat Proximity Controller {ES sim}
- Embraer ERJ-170 Proximity Switch System
- R&D Simulator for Anti-Skid and Linear Position-Sensing System
- ARINC 717 Digital Flight Data Acquisition Unit (DFDAU) for the Boeing 737
- ARINC 615 Airborne Data Loader (ADL) Protocol

NASA awarded Triakis a 1-year research grant in 2002, and a 3-year research grant in 2003 to study how its simulation technologies may be used to improve software assurance.

2.7 Support

No matter how much research went into an avionics design, some changes will invariably need to be implemented for reasons of enhancement, changes to the system environment or obsolescence in the years following entry into service. This is another area where the simulator originally developed for the unit design can easily return big dividends. Traditionally, HW- and SW-design changes would require the recreation of some or all of the original laboratory-equipment setup – often at a substantial cost in material and labor. Through the use of the original

system simulation, functional changes can be made at the ES level and verified using the original or modified test scripts.

Modifications can be simulated at the DE-level and fully tested in the original system environment to verify that intended functionality has been implemented without negative side effects. With this capability, service bulletins can be developed with the highest degree of quality assurance at a cost below those developed using standard methodologies. Since IcoSim© runs in the ubiquitous Windows® environment, it will be relatively easy through the use of standard configuration and archive management procedures to support the 20 - 30 year typical product service life.

3 SUMMARY

While it can be of great benefit in the development of embedded avionics software alone, IcoSim© is most cost-effective when used from the very beginning of a project where system-level functionality is being conceived and developed. IcoSim© is a most effective tool for avionics development because the entire world, from the highest system level to the minutest hardware component, in which the target software is being developed to interact can be simulated relatively easily. This virtual world contains a fully functional simulation of the target hardware on which the embedded SW object code itself is loaded into simulated ROM, executed on the simulated CPU, and interacts with the virtual world through simulated I/O hardware exactly as it would in the real world. V&V at the CPU level is ultimately the only way to ensure that the actual SW implementing the subject functionality is thoroughly tested with all the attendant delays and constraints inherent in execution at the object code level.

IcoSim's© ability to execute object code in it's virtual target environment ensures that the final software is tested regardless of how it was created – whether by traditional methods or by the use of code generators based on behavioral models. Simulation of peripheral systems, sensors, and vehicle dynamics ensures that realistic data is always being sent to and received from the SW under test as opposed to traditional methods of creating test pattern sequences for SW stimulation. ***This unique combination of capabilities provides for unprecedented levels of SW V&V confidence, and allows for the simulation of external system, sensor, and actuator faults as well as factors external to the aircraft, for the express purpose of validating and verifying the response of the total system.***

Simulation of high-level systems has been shown to reduce development costs and improve the testing, reliability and quality of embedded software. When software can be run in the simulated target HW environment for which it was developed and further, in the simulated aircraft itself, large gains in productivity can be realized. For instance:

- Each programmer uses a copy of the entire simulated target environment during development thereby reducing the dependency upon laboratory hardware resources.
- By facilitating concurrent design of hardware and software, the use of IcoSim© shortens project schedules and reduces associated costs.

- Since IcoSim© test scripts are based on a complete environment model, more realistic and thorough testing is accomplished than can be achieved by traditional test methods. Applications developed with IcoSim© are tested to a high degree of confidence reducing the need for repeated testing in expensive system integration facilities.
- Modeling of hardware failure modes is easily accommodated with IcoSim©, eliminating the need to build custom hardware failure emulators.
- IcoSim's© method of performing system tests, software verification tests, and module-level tests promotes high levels of consistency and re-use across multiple versions and products.
- Automatic path coverage analysis can direct the development of additional module-level tests to exercise untested paths, insuring complete code test coverage. Exercising untested paths is accomplished by commanding external device models to force the application code down those paths.
- The simulator can be used to prototype new software features and enhancements
- The IcoSim© test environment is virtually immune to obsolescence issues that typify special hardware equipment traditionally used for testing.

There is no better way to verify SW functionality and performance than to run it in the environment in which it was designed to interact. However, assembling a real world avionics SIL is very expensive and creates a resource bottleneck as programmers vie for testing time. Using an IcoSim© virtual world, however, every programmer has access to a copy of the VSIL at his or her individual computer in which to develop & test his or her SW. To enhance the use of the VSIL as a test vehicle, IcoSim© comes with simulations of standard laboratory test equipment such as oscilloscopes, signal generators, and the functional capability of microprocessor emulators.

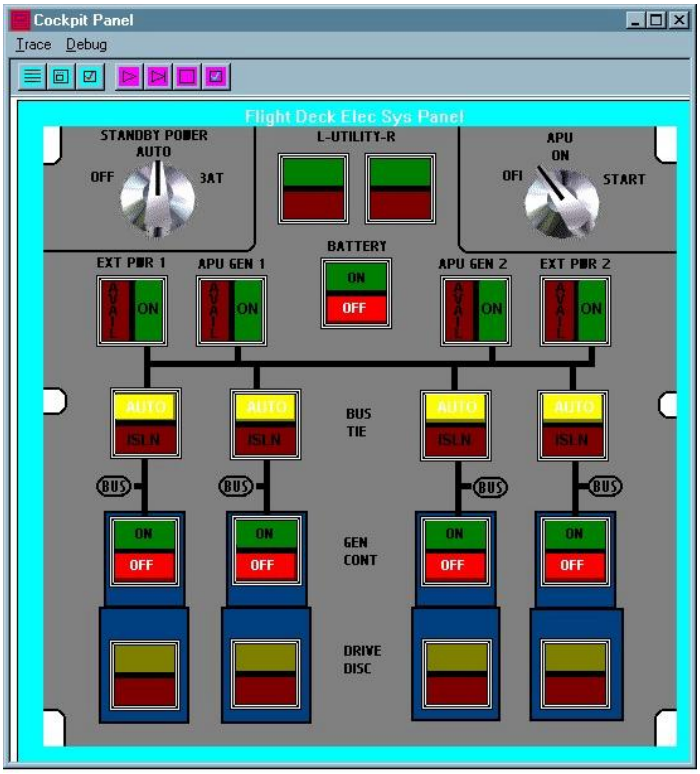
The IcoSim© virtual world is hierarchical so objects at the lowest level of abstraction are interconnected to make successively higher levels of functional objects until the required environment is complete. This modular approach enables the highest possible levels of SW reuse for subsequent development projects.

Creating your host embedded avionics SW development environment with IcoSim© following the Triakis ES model ensures that:

1. Maximum quality is achieved by testing in the VSIL environment throughout the entire product development and service life cycle.
2. Lowest overall development and support costs are realized through unprecedented fault detection coverage at the least costly phase of the development process.

APPENDIX

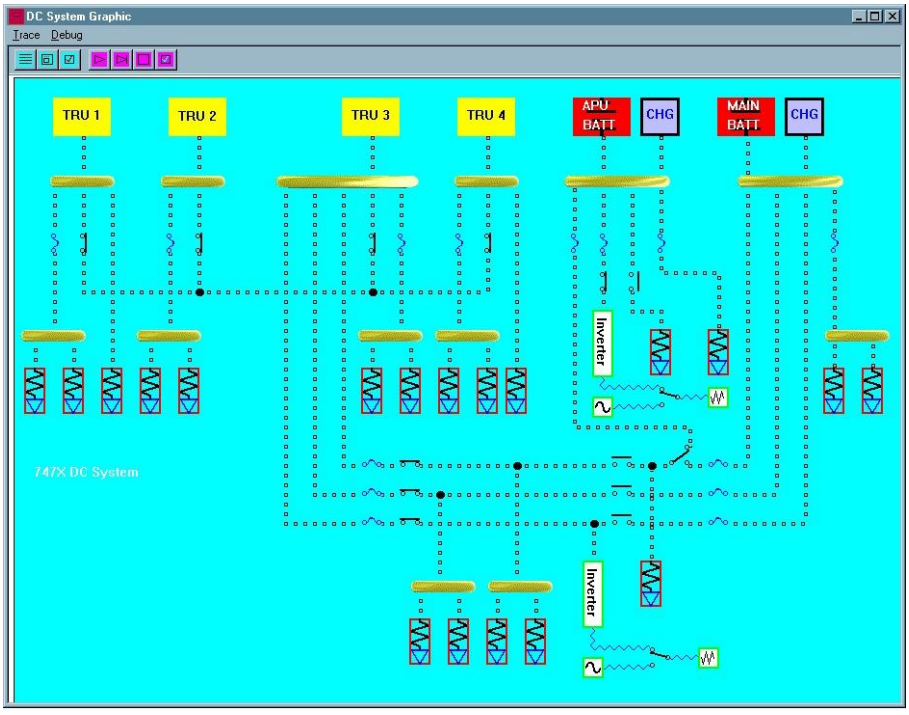
Simulated Boeing 747 DC Standby-Power Cockpit Control Panel (courtesy of ELDEC Corp.)



The figure at the right is a screen shot of a simulated 747 DC Standby-Power Flight Deck Electrical System Control Panel. This panel is simulated at the ES-level and functions in its virtual aircraft environment as it would in the real aircraft environment. When the simulation is running, the knobs and pushbuttons are operated with the mouse. The switches initiate their corresponding actions and the lamps illuminate to reflect their corresponding system status.

The figure below is a screen shot of another module that runs within the same simulation that dynamically represents, in schematic form, the DC standby-power system circuitry onboard the 747 aircraft. While the simulation is running, the dotted lines representing the main power buses indicate both the magnitude and direction of current flow in an animated fashion. Relay contacts and circuit breaker states can be observed dynamically responding to manual input from the control panel, or automatic input from the DC Standby-Power Control Computer (DC SPCC).

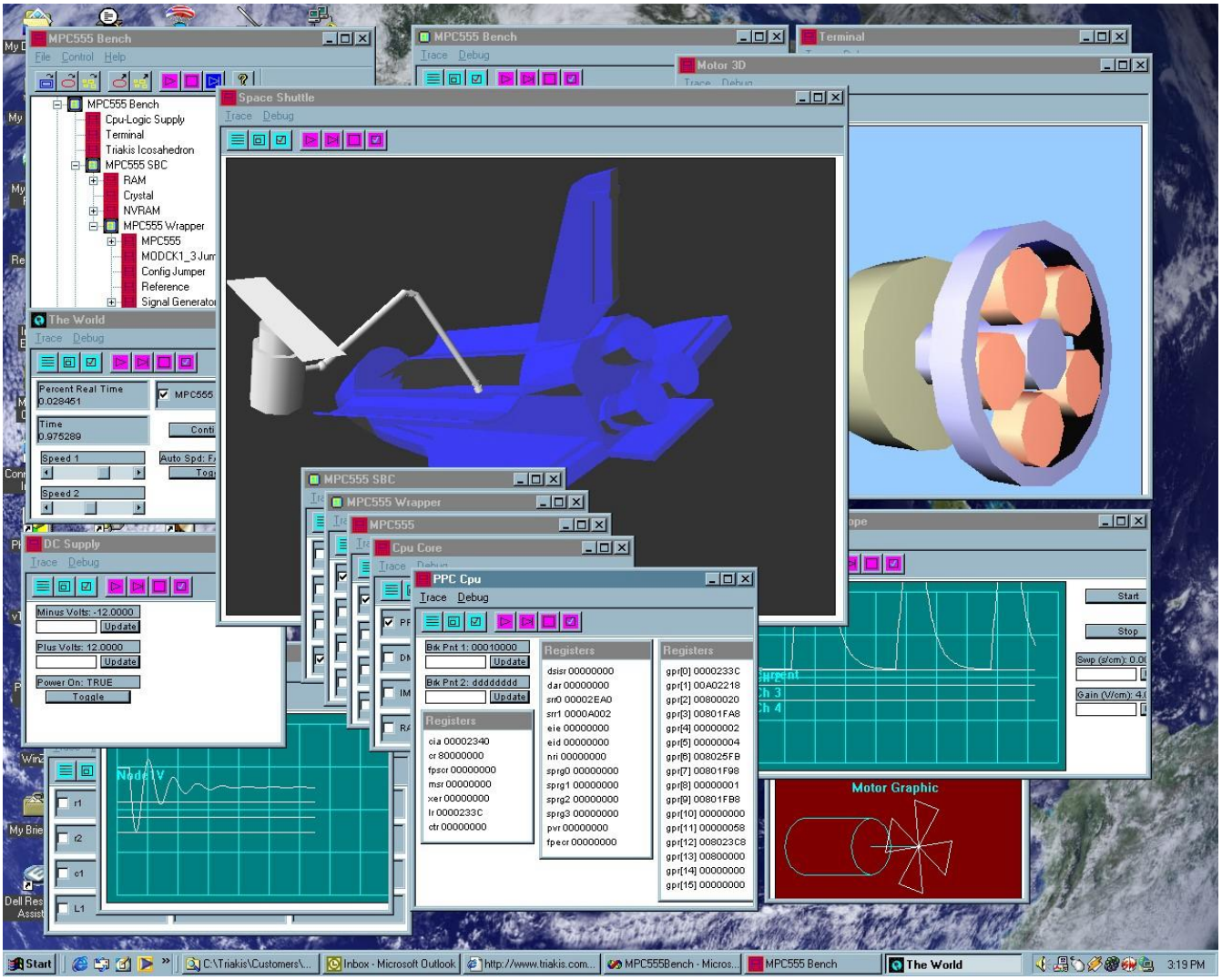
Dynamic System-Level Simulation of Boeing 747 DC Standby-Power Control (courtesy of ELDEC Corp.)



Not shown here are the DC SPCC module which has been simulated at the ES-level, as well as the APU, Integrated Drive/Generators (IDGs), etc. that have been simulated with enough fidelity to realistically represent the interface signals and timing with which this system interacts.

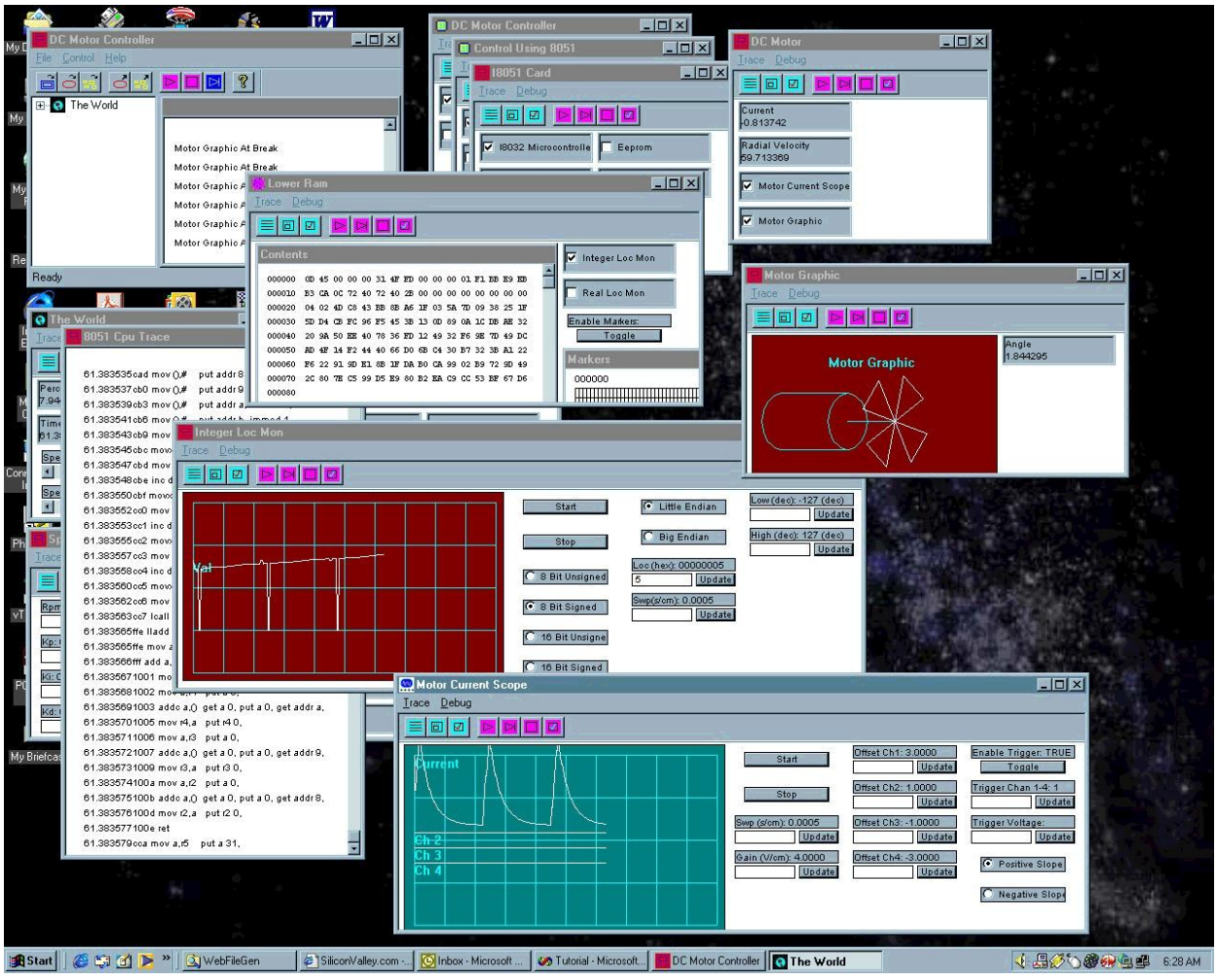
The screenshot below depicts a simulation of the space shuttle robotic arm currently in development by Triakis to enhance IcoSim© with Newtonian dynamics capability. The CPU chosen for use in this simulation is the MPC 555 PowerPC™.

Space Shuttle Robotic Arm Control Simulator



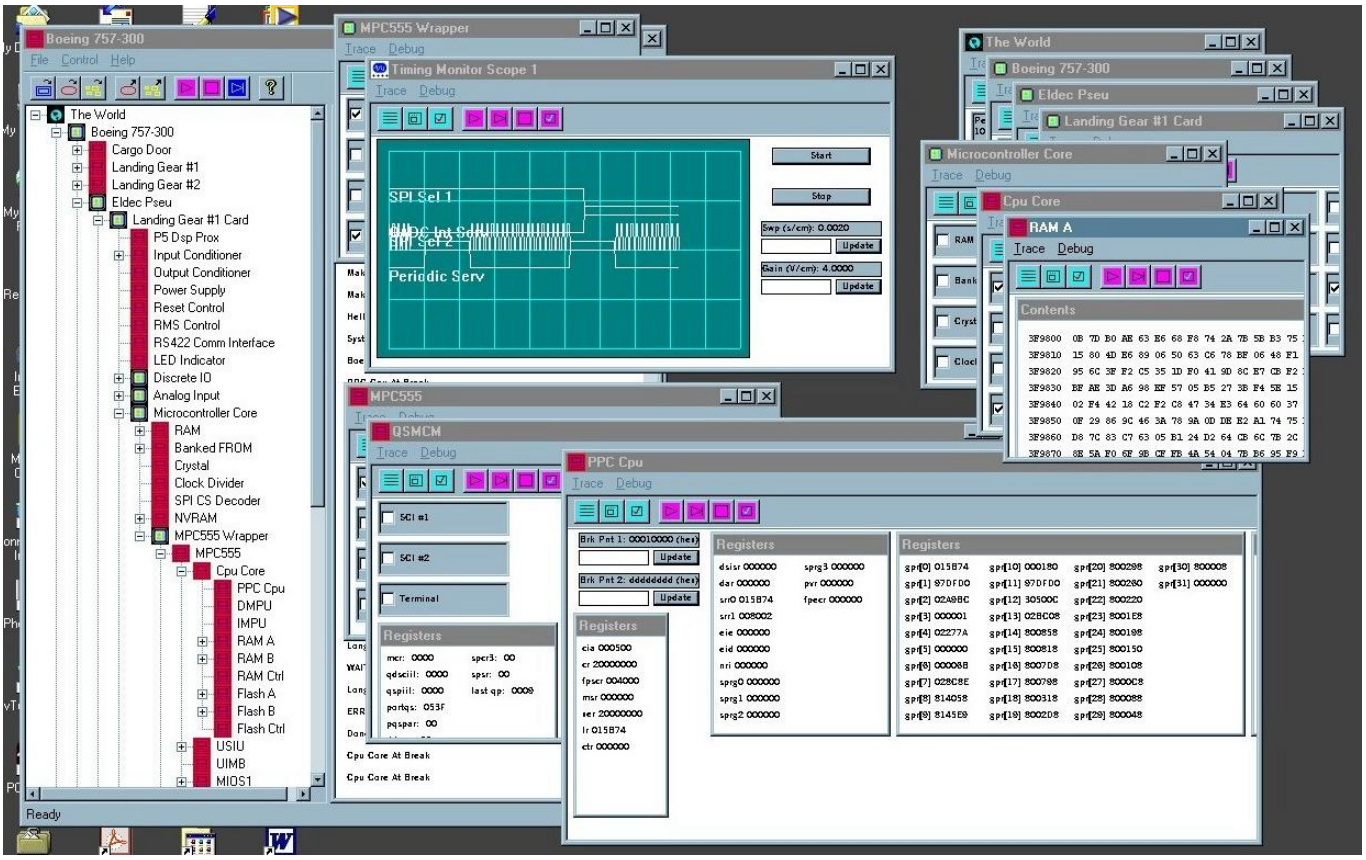
This is a screenshot from Triakis' motor speed controller tutorial simulation. This simulation exemplifies how a DE has been developed from an ES, and how the ES and DE can be used interchangeably in the simulation.

Tutorial Application DC Motor Controller with Memory Location Scope



This screen shot shows a hierarchical subset of the Boeing 757-300 from the Airplane level down to the CPU level. The PowerPC™ registers are shown along with an oscilloscope monitor displaying periodic interrupt signals.

Triakis Custom Boeing 757 Sensing System Simulator (courtesy of ELDEC Corporation**)**



This is a screen shot displaying some diagnostic views of the ELDEC Corp. 757-300 PSEU simulator. This simulator is running SW object code implementing an adaptive control filter algorithm on a simulation of the highly complex Motorola DSP56302 digital signal processor. This simulator simultaneously runs object code on another simulated microprocessor that serves as the primary controller for the PSEU.

Simulated Motorola DSP56302 running adaptive control filter algorithm SW
(courtesy of ELDEC Corp.)