

# **A DEMONSTRATION OF THE INTEGRATED SUPPORTABILITY ANALYSIS AND COST SYSTEM (ISACS+)**

Helena L. Weaks  
James D. Barrett

NYMA, Inc.  
Engineering Services Division  
4027 Colonel Glenn Hwy., Suite 445  
Dayton, OH 45431-1672, U.S.A.  
e-mail: hweaks@earthlink.com, jbarrett@erinet.com

## **ABSTRACT**

This paper describes the Integrated Supportability Analysis and Cost System (ISACS+) and the features which will be demonstrated. ISACS+ is a distributed, client/server system for evaluating operation and support characteristics of weapon systems and, in future builds, commercial aircraft systems. This tutorial briefly describes the ISACS+ Concept of Operations and how Operation and Support studies are conducted using the ISACS+ tool. This tutorial also describes the architecture used to distribute the ISACS+ software over multiple computing tiers in support of the concept of operations.

## **1 INTRODUCTION**

The Integrated Supportability Analysis and Cost System (ISACS+) was originally developed for General Electric Aircraft Engines (GEAE) to perform operation and support analysis for the U. S. Air Force F-16 engine program. ISACS+ is currently being rehosted and extended under a Space Act Agreement between GEAE and NASA Lewis Research Center.

Both commercial and military system developers know that there is a competitive advantage to improving customer satisfaction through improved reliability and supportability at minimum cost. Military systems must comply with requirements for maintenance and supportability in order to maximize system availability to assure mission success. This concept was deemed so critical that numerous standards have been developed to assure that reliability, maintainability and supportability are an integral part of the design process. Maximum system availability at low cost and assured safety are a must for commercial airlines as well, to ensure continuing revenue generation.

In general, weapon system customers require systems that will operate for long periods of time with minimum down time; in which any required maintenance actions are quick, simple and inexpensive. However, such maximum supportability does not come without a price. In general, trade studies must be conducted to balance the supportability requirement against other system parameters, such as acquisition costs or performance.

In order to ensure that supportability requirements are met, the offering of warranties by manufacturers is a common practice in both the military and commercial environments. Like supportability, warranties incur a cost of implementation, which is typically passed on to the customer. The manufacturer's ability to estimate warranty cost determines whether or not a warranty will be profitable; and the ability to minimize warranty costs for the customer without incurring a loss is a competitive advantage to the system manufacturer. When warranty parameters and costs are well-defined and understood, use of warranties can yield a win-win situation for both the manufacturer and the customer.

## **2 THE SCOPE OF ISACS+**

The ISACS+ model is a full life cycle model, which supports studies conducted during concept exploration (paper engine) and preliminary design phases through full-scale production and fielding of weapon systems. Figure 1 shows how the accuracy of weapon system data increases as the design moves through these various developmental phases.

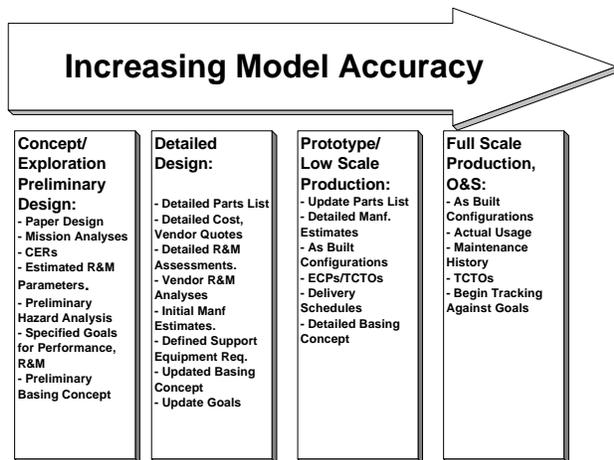


Figure 1 - Data Accuracy within the Product Life Cycle

**2.1 Concept Exploration and Preliminary Design**

Supportability considerations begin during the concept exploration/preliminary design phases since these phases are when materials, complexity, durability, and accessibility of hardware are defined. Changes made during this phase are relatively inexpensive, since large investments in equipment, hardware, and manufacturing processes have not yet been incurred. Goals for reliability and maintainability (R&M), acquisition cost, performance, etc. should be set based on perceived or directed customer requirements. These goals often conflict with one another, requiring that on-going trade studies should be conducted to ensure an optimal design solution which best satisfies all customer requirements. In order to support trade studies, high level information is collected, such as preliminary hazard analyses, results of cost estimating relationships (CER), and comparisons to similar systems.

**2.2 Detailed Design**

As the design becomes fixed, data accuracy increases. Detailed parts lists, cost assessments, reliability/maintainability analyses, support equipment requirements and customer usage become better defined. Low level hardware purchases are made for prototype testing. Design changes during this phase should be assessed using an integrated engineering approach, in which all areas of impact are identified and assessed. Figure 2 provides an example of the interrelationships between design changes and supportability. Design decisions related to an improvement in a performance parameter may have an impact on the life or durability of hardware; which in turn may impact scheduled and unscheduled maintenance frequency. Changes in

maintenance frequencies would likely have an impact on spares demand and provisioning requirements. Design changes may also have an impact on repair or maintenance procedures which affect skill level specifications, materials used, or support equipment requirements. ISACS+ baselines can be updated during this phase to reflect this increased accuracy in data. The model can then be used for making intelligent decisions based on trade studies which analyze impacts of design modifications or to assess impacts of various customer usage scenarios.

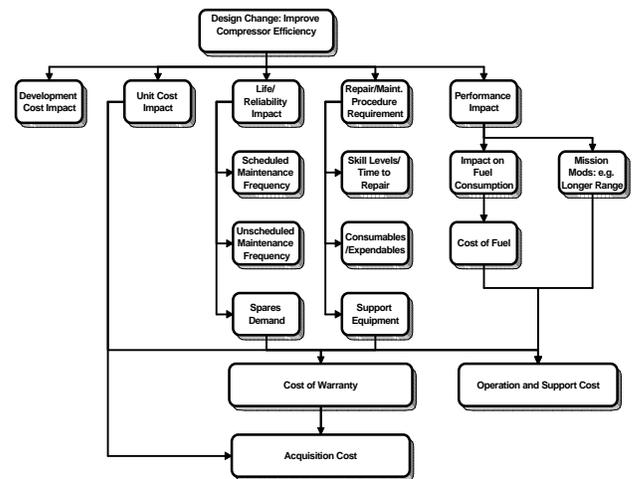


Figure 2 - Impact of a Design Change on Supportability

**2.3 Prototype or Low Scale Production**

As system designs becomes fixed and the pre-production phase begins, maintenance and support programs must be defined, including identification of maintenance tasks, facilities, manpower requirements, support equipment needs, and spares provisioning. ISACS+ can help the logistician assure that the correct resources are at the right location at the right time in order to support the system being deployed. ISACS+ does this by forecasting manpower, facility and equipment utilization; as well as forecasting spares, consumables and expendables demands at each of the facilities under study.

The customers may also wish to negotiate warranty contracts. ISACS+ can provide the warranty analyst with input to the cost drivers relative to warranty so that the planners can provide justification for the cost of warranty during warranty negotiations.

**2.4 Full Scale Production/Operation and Support**

Design changes can and do occur after systems have been shipped. This being so, one must plan for introduction and support of design changes to fielded systems at a minimum cost and with minimum impact to fleet operations. ISACS+ can aid the analyst in determining the cost, system availability and resource utilization impact of the introduction of new replacement hardware into the field.

Once a weapon system is fielded, field history data may be collected to assess the effectiveness of system design and maintenance programs in order to identify safety hazards, improvements, or opportunities for cost reductions. Warranted removals are monitored to assure that warranty commitments are met without incurring a loss to the manufacturer.

**2.5 Organizational Roles**

The disciplines within an organization that contribute to operation and support studies are Reliability, Maintainability, Logistics Support, Life Cycle Cost and Warranty analysts. In general, these disciplines are highly dependent on one another for information. ISACS+ integrates these disciplines by providing a common data source and analysis tool, thus reducing

data and application redundancy while enforcing consistency in results. These disciplines typically follow standard practices for military applications (SAE RM&S Committee, 1989) which may be modified for the commercial environment (Weeks, 1996).

**3 ISACS+ CONCEPT OF OPERATIONS**

The ISACS+ concept of operations supports the natural flow of a typical analysis, as exhibited in Figure 3 (Weeks and Barrett, 1996). The basic steps for conducting an analysis are as follows:

1. Development of a baseline;
2. Storage and retrieval of baselines by multiple users;
3. Duplication and modification of baselines for trade studies and sensitivity analyses;
4. Automated downloading of the currently active baseline for input to the simulator;
5. Execution of the simulator, using multiple iterations to get a measure of variation for the output; and
6. Automated database storage of simulator output, for analysis and flexible reporting.

Each of these steps is described in more detail in the following sections.

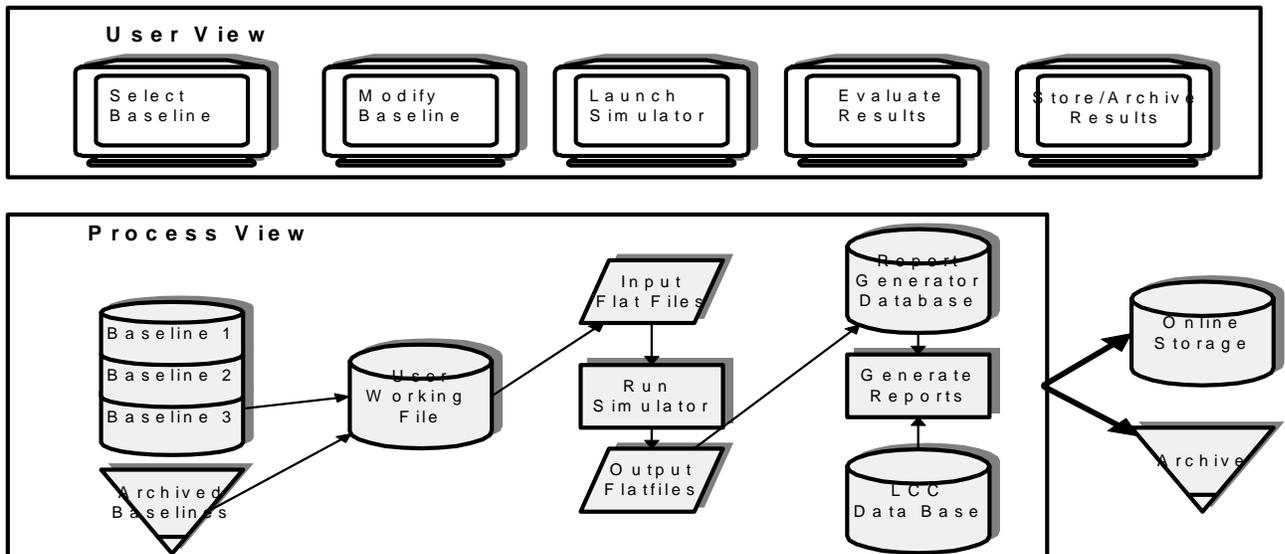


Figure 3 - ISACS+ Concept of Operations

### 3.1 Development of Baselines

ISACS+ contains a flexible, object-oriented simulation model that simulates the operation and support environment for a fleet of aircraft (Weeks and Barrett, 1996). ISACS+ offers the user over one hundred features for modeling a weapon system operation and support environment, including the following:

1. The capability to model both military and commercial 'basing' structures where the user may identify multiple locations with multiple facilities, where each facility may have different maintenance capabilities and capacities.
2. Modeling of multiple weapon system (aircraft) configurations with different subsystems (engines, electronic controls), where each subsystem may be defined with unlimited levels of indenture. The subsystems are allowed to dynamically change within the simulation through the modeling of time compliance technical orders (TCTOs).
3. Modeling of flight line activities such as fixed and scheduled inspections, as well as scheduled removals with the capability to "overfly" schedule limits.
4. In-shop activities, such as 'workscoping', may identify opportunistic maintenance. Other maintenance requirements may be discovered during teardown and inspection of a system.
5. Failures are modeled using three parameter Weibull distributions for durability failures, or using parameters for reliability growth. Hardware may have multiple failure modes of any type. Failures can be modeled at any level of indenture.
6. Each hardware item may have life limit or other scheduled removal activity requiring specific maintenance actions.
7. System level failure modes such as performance degradation based on the degradation of subsystems can be modeled.
8. Maintenance tasks leading to scrap, repair or further inspection may be modeled. Future enhancements will include modeling of rework.
9. Warranty requirements may be modeled at the system level.

Development of a baseline is a time-consuming process. Each commercial or military organization typically has its

own data sources stored in heterogeneous systems and formats. Data may also be provided by multiple groups within an organization. The process of developing a baseline may require extracting and manipulating large amounts of such data, especially for fielded engine studies where there is much historical data. Therefore, preprocessing routines for automated import of data are source-dependent.

Older simulation models often mandate use of application-unique files that require the user to count fields or use delimiters to locate where the data should fall for accurate load into the simulator. Often, there is no indication of what the fields mean, and the user must constantly refer to the user documentation to recall the field names and lengths. This increases the tedium of an already time-consuming process for setting up large data files. ISACS+ uses a graphical user interface (GUI) that provides an intuitive approach for viewing and modifying simulator inputs, which requires minimal training of the user. To verify the usability of the ISACS+ GUI, GEAE analysts were provided with no user documentation and received minimal training, yet there have been few questions relative to the input of the simulator using the GUI.

ISACS+ uses the relational data base engine, Oracle, for storing input data. Relational data bases allow the developer to set up intuitive relations between entities that are easy to understand by the user. For example, an entity such as a "component" could have multiple "inspection requirements" or multiple "failure modes" associated with that "component". The relational data base works in conjunction with the GUI framework to allow the user to enter data in a manner best suited to the user's work habits, without understanding the developer-defined relationships being implemented and enforced by the GUI. Additionally, the use of a relational data base engine allows the user to create ad hoc reports from the input data, eliminating reliance on hard coded reporting features often found in simulation models. Relational data bases are also relatively easy to load from other external data sources. For example, much of the test data generated for ISACS+ was originally prepared on a PC in FoxPro, down loaded to comma-delimited ASCII text files using FoxPro features, and then uploaded into Oracle using simple Oracle SQL\*Loader scripts. The GUI was then used to review the results.

### 3.2 Modification of Baselines for Trade Studies

Trade studies typically involve making small changes to a specific element of a baseline to determine the impact on system behavior. ISACS+ allows the user to select and load a baseline into a working space owned and password protected by the user. As the baseline is being modified, the ISACS+ system ensures that the business rules and data constraints required by the simulator are adhered to. The user's ability to enter bad data (i.e., data that might cause the simulator to "crash") is minimized through the use of warnings, data constraints, integrity checks, and provision of acceptable defaults.

### 3.3 Download to the Simulator

Once the user is satisfied with the baseline or baseline modifications, the data may be automatically downloaded to files required by the simulator. The user's desired view of the data differs a great deal from the needs of the simulator. The user wishes to see the data in terms of characters and intuitive, meaningful relationships. However, character representation within the simulation environment reduces available memory, in turn restricting the size of the model itself and increasing the complexity of the algorithms within the simulation model. Integerized data is used within the simulator to reduce memory requirements, enhance performance and simplify algorithms.

The simulator is a stochastic model using Monte Carlo techniques to randomly generate events. Each time an event occurs, a comma delimited record is written to a text file to record that event; such event recording is designed to ensure that desired user views of the output could be supported. Each iteration of the simulator is identified by recording a time stamp and the random number seed used to initiate the iteration with each of the output records. This provides the necessary information for calculations of means and variances across multiple iterations of the simulation outputs.

### 3.4 Simulator Execution

There are currently sixty-six (66) relational tables used to define the maintenance environment and hardware items for the ISACS+ model. The user only needs to load data for a small subset of those (typically 15 to 18 tables), depending on whether the system is a paper engine or a fielded system. The remainder of the tables are filled in based on the modeling needs of the user and increase the complexity of the environment being modeled. The simulator "knows" which features are being modeled based on whether or not a given table has data.

### 3.5 Storing Simulation Results in a Reporting Database

Once the simulator has completed execution, the comma delimited output files produced by the simulator are loaded into a relational database, where the user can analyze the results and generate reports in a secure environment. The use of a relational database facilitates report generation and modification using the many available reporting tools. The user is able to create ad-hoc reports by using the existing report data, without modifying or rerunning the simulator. A reporting database offers numerous views of the data, and allows the user to drill down into the data to investigate, in detail, numerous patterns. The user is also able to relate output data to external information. For example, maintenance events generated by the simulator can be related to support equipment usage, skill levels, manpower requirements, cost parameters, or usage of consumables and expendables.

## 4.0 THE ISACS+ TECHNICAL ARCHITECTURE

ISACS+ was originally hosted on a UNIX platform using X Windows/Motif, Ingres Relational Database Management System, Ingres Windows/4GL, and C++ for the simulator module. It was necessary to keep these considerations in mind while defining a technical architecture which would prove more cost effective, and would yield a flexible, extensible system.

### 4.1 An Approach for Software Distribution.

The environment defined for ISACS+ is one in which a user at a desktop workstation can attach to the ISACS+ system to perform database maintenance, initiate the simulator to run trade studies, generate reports from simulator output, and archive study results to a backup medium. To avoid the problem of bogging down a single server with application, GUI, and database processing, a client/server implementation was selected. This supports distributing system functions over multiple computing platforms while allowing access to a central data repository (Vaughn, 1994). All of this is largely transparent to the end user, who is free to perform tasks without regard to where the various system components are being vended.

### 4.2 Open Systems Guidelines

The architectural approach followed for implementation of ISACS+ was facilitated by adopting the principles defined in the National Institute of Standards and Technology (NIST) Application Portability Profile (APP) (NIST Special Publication 500-210, 1993). The NIST

APP is, in turn, based upon the Institute of Electrical and Electronic Engineers (IEEE) Open Systems Environment Reference Model (OSE/RM), defined by the IEEE Portable Operating System Interface for Computer Environments (POSIX) Working Group P1003.0 (IEEE P1003.0, 1992).

**4.3 The IEEE OSE/RM.**

The IEEE POSIX Working Group OSE/RM defines a layered architecture that divides system components into three general areas.

1. Application Software - data, documentation, training and application programs;
2. Application Platform - hardware and software components that provide system services used by the Application Software layer; and
3. External Environment - system elements external to the application software and platform (human interface and interface to other application platforms).

These layers are joined together by two layers of interfaces.

1. Application Program Interface (API) - the interface between the application software and application platform layers which supports portability of the application software;
2. External Environment Interface (EEI) - the interface between the application platform and platform external environment layers which supports data interchange between human users, external data stores, and other application platforms.

**4.4 The NIST APP Profile Applied to the OSE/RM.**

The NIST APP modifies the IEEE OSE/RM by applying standards that are of interest to the U.S. Government and defining services at the various layers and interfaces. The following description of the IEEE OSE/RM has been extracted from the NIST APP.

“The services defined in the APP tend to fall into seven broad service areas:

- operating system services
- human/computer interface services
- data management services
- data interchange services
- software engineering services
- graphics services
- network services

“Each of the APP service areas addresses specific components around which interface, data format, or protocol specifications have been or will be defined. Security and management services are common to all of the service areas and pervade these areas in one or more forms.”

The APP Service areas, as applied to the IEEE OSE/RM, are displayed in Figure 4.

**4.5 Applying the NIST/APP & IEEE OSE/RM to the ISACS+ Problem Domain**

In general, it is desirable to follow the NIST APP and the IEEE OSE/RM models to the greatest extent possible (Department of Defense Joint Logistics Systems Center, 1995). Doing so will minimize dependence on proprietary platforms, operating systems, and telecommunications protocols by establishing an application which is portable

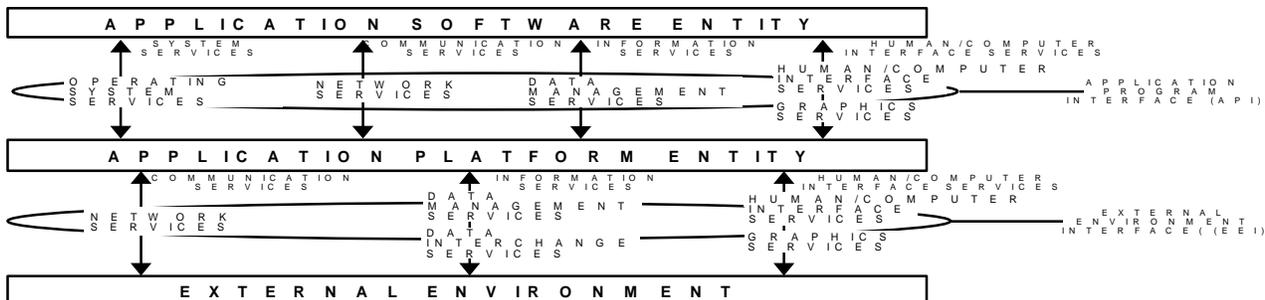


FIGURE 4 - NIST APP Service Areas Applied to the IEEE OSE/RM

across various platforms and databases. However, in reality, various platforms, operating systems, and vendor extensions to “standard” languages such as C++ and SQL often result in applications that are only portable in a very narrow context, or require extensive rework to port. While it is desirable to isolate various layers as per this model, it is also worth noting that vendor-unique extensions usually provide value-added capabilities for which the tool in question is chosen. The problem, then, is to evaluate the degree of adherence to standards against the available toolsets.

#### 4.6 The ISACS+ Development and Deployment Environments.

In order to best define and examine tradeoffs for tool selection, it is first necessary to define the environment in which the ISACS+ application will be run. The definition of such an environment has at once been simplified by advances in computing technology and complicated by the large number of choices available for developing distributed systems (Wheeler, 1992). The use of C++, without extensions, makes the ISACS+ Simulation Module portable to virtually any operating environment with a native C++ compiler. Likewise, use of the Ingres SQL database engine, and therefore any truly standard relational database manager, makes the database table definitions largely portable, with a moderate level of modification. However, the original GUI code is largely written in a proprietary language, Ingres Windows/4GL, and induces dependence on the Ingres engine. The GUI code turns out to be the least portable component, and must be replaced. This replacement is complicated by the fact that there are at least two “competing” de facto industry standards for GUI development - the MS Windows Software Developers Kit (SDK) and the X-Window/Motif tool kit.

The development and deployment runtime environments and tools ultimately selected for development of ISACS+ are defined in Table 1. These tools were selected to best meet the needs of the ISACS+ architecture as defined by the ISACS+ Technical reference Model (TRM). Specifically, the ISACS+ TRM is a derivative of the NIST APP and focuses on nine major service areas required to support existing and planned ISACS+ functionality (Barrett, 1995). These are given below.

1. Operating System Services
2. Application Platform
3. Software Engineering Services (Development and Maintenance)
4. Data Management Services
5. Data Interchange Services
6. User Interface Services

7. Distributed Computing Services
8. Network Services
9. System Management Services.

Table 1 - Tool Selection and the ISACS+ TRM

	Client Application	Server Application
<b>Application Platform - Development</b>	120 MHz Pentium Laptop, 40 MB RAM	120 MHz Pentium Laptop, 40 MB RAM
<b>Application Platform - Deployment</b>	120 MHz, 32 MB RAM	166 - 200 MHz Pentium Pro Processor, 64+ MB RAM
<b>Operating System Services</b>	Windows NT 3.51 Workstation or Windows 95	Windows NT 3.51 or 4.0 Server
<b>Software Engineering Services</b>	Delphi 2.0 Client/Server Development Suite & Borland C++ 5.0 Development Suite	N/A (Maintenance only)
<b>Data Management Services</b>	Oracle SQL*Plus 3.2 and Personal Oracle 7	Oracle 7.2 Workgroup Server for Windows NT
<b>Data Interchange Services</b>	Borland SQL Links for Oracle & Oracle SQL*Net Client V2.2	Oracle SQL*Net Server V2.2
<b>User Interface Services</b>	Win 32	Win 32
<b>Distributed Computing Services</b>	Windows 32 TCP/IP Stack	Windows NT 3.51 or 4.0 TCP/IP Stack
<b>Network Services</b>	Windows 32 TCP/IP Stack & Oracle SQL*Net Client V2.2	Windows NT 3.51 or 4.0 TCP/IP Stack & Oracle SQL*Net Server V2.2
<b>System Management Services</b>	N/A (Managed from NT & Oracle Servers)	Windows NT 3.51 or 4.0 Server & Oracle 7.2 Workgroup Server for NT

#### 4.7 Distributing the Application

The planned distribution of the ISACS+ components over computing tiers has evolved with time. After several large trial runs, the development team decided that the Simulation Module, initially targeted for implementation on the server machine, would disrupt a workgroup much less if vended to each individual client machine. With this change, it seemed reasonable to run the Simulation Module as a 32-bit DLL and assign it an API signature. Pushing large loads through embedded calls to the database (in both C++ and Object Pascal) turned out to be orders of magnitude slower than direct calls to the Oracle engine to natively load the output of the simulation runs for the Report Generator, and so a Remote Procedure Call (RPC) Server was developed to

perform this task. Like the simulation module, the RPC server is invoked as a Win32 DLL from the client machine. The Report Generator application relies heavily on the use of Triggers and Stored Procedures, and the processing is split between the database and the GUI. The GUI which provides all of this capability to the user runs on the user's machine. This distributes the ISACS+ capabilities logically over computing tiers, providing functionality where it is needed as it is needed. Finally, by providing the user with a logical view which treats the entire network of PC's as a single system, ISACS+ allows users to retrieve and store data from any machine, thus allowing the data to be distributed as best meets the needs of the users of ISACS+.

Figure 5 shows the final distribution of components over computing tiers as the model is currently built.

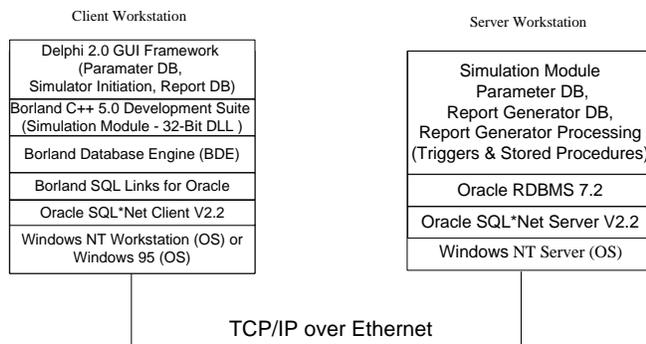


Figure 5 - Distribution of ISACS+ Components Over Computing Tiers

## REFERENCES

- Barrett, James D. 1995. *ISACS+ Concept of Operations - Technical Reference Model*. NYMA, Inc. for NASA Lewis Research Center, Cleveland, OH.
- Department of Defense Joint Logistics Systems Center (JLSC). 1995. *JLSC Integrated Technical Architecture and Common Operating Environment*. Wright-Patterson Air Force Base, Dayton, OH.
- International Electrical and Electronics Engineers, Inc. 1992. *Guide to the POSIX Open Systems Environment. Draft 15*. IEEE P1003.0, Washington, DC.
- National Institute of Standards and Technology (NIST). 1993. *NIST Special Publication 500-210 Application Portability Profile*. Systems and Software Technology Division, Computer Systems Laboratory, Gaithersburg, MD.
- SAE RM&S Committee .1989. *Reliability, Maintainability, & Supportability Guide Book, First Edition (Draft)*, Society of Automotive Engineers Continuing Education, Committee (G-11).
- Vaughn, Larry T. 1994. *Client/Server System Design & Implementation*. McGraw-Hill, New York, NY
- Weaks, Helena .1996. *Requirements Analysis Plan, Integrated Supportability Analysis Cost System (ISACS)*; NYMA, Inc. for NASA Lewis Research Center, Cleveland, Ohio.
- Weaks, Helena and James D. Barrett .1996. *Build 1 System/Software Requirements Document Enhanced Integrated Supportability Analysis and Cost System (ISACS+)*; NYMA, Inc. for NASA Lewis Research Center, Cleveland, Ohio.
- Wheeler, Tom. 1992. *Open Systems Handbook*. Bantam Books, Inc., New York, NY.

## AUTHOR BIOGRAPHIES

**JAMES D. BARRETT** is an Engineering Specialist with NYMA, Inc., in Dayton, Ohio. He is responsible for selecting and installing the ISACS+ development environment and for development of the Simulation GUI, the Parameter Database GUI, the Main Menu GUI, and all C++ development. In addition to ISACS+, he has developed simulation models for factory and laboratory facilities planning. He is currently a member of the IEEE, IEEE Computer Society, the Association for Computing Machinery, and the Society for Computer Simulation.

**HELENA L. WEAKS** is an Engineering Specialist with NYMA, Inc. in Dayton, Ohio, under contract with NASA to support a Space Act agreement between NASA and GEAE to develop the ISACS+ system. She is one of the original developers of the ISACS+ system. The original effort took place at Belcan Engineering, from 1992 to 1994, as development of a software prototype based on requirements defined by General Electric Aircraft Engine (GEAE) systems. She received her Ph.D. in operations research from the University of Toledo in 1986. She received her masters in mathematics in 1979, and bachelors in mathematics in 1977, both from the University of Toledo.