The Space Congress® Proceedings

1988 (25th) Heritage - Dedication - Vision

Apr 1st, 8:00 AM

Software Engineering Development Environment For The Launch Processing System

Marcia W. Burch

Debra K. Moyer

SOFTWARE ENGINEERING DEVELOPMENT ENVIRONMENT FOR THE LAUNCH PROCESSING SYSTEM

Marcia W. Burch **Debra K. Moyer**

INTRODUCTION

Tasked with supporting a progressive Shuttle launch rate, Lockheed Engineering and Software Production set out in 1984 to address the need to increase software productivity. Attention was focused on innovative tools since existing computer development systems were being reallocated for Shuttle operational testing and launch activities. It became apparent that due to the highly integrated nature of software production activities, a solution involving a local area network of engineering workstations was required. After prototyping and proving the design for increasing productivity, Lockheed procured and installed a networked computing system which generated a state-of-the-art environment for software engineering. The introduction of this new technology not only brought about new methods of implementing software changes, it resulted in a culture change for nearly everyone involved in the development cycle.

HISTORY

To investigate the concept, a prototype network of 12 individual workstations and 4 file servers was established to determine methods for improving productivity and to provide development facilities separate from the Launch Processing System (LPS) firing room equipment. In April of 1985, the prototype team demonstrated to Lockheed and NASA management the capabilities inherent in the system. Following the proof-of-concept an acquisition plan was generated and competitive procurement initiated. The contract was awarded and delivery of Phase 1 of the acquisition plan equipment was completed in December, 1985. The final phase of the equipment purchase was installed in March, 1987.

EQUIPMENT DESCRIPTION

The local area network, referred to as the LPS Software Development Network (LSDN), is a 12 megabit-per-second token passing ring architecture connecting individual nodes in a series. Each node contains a 32-bit VLSI CPU (integrated MC68020 processor and MC68881 floating point co-processor) and 4M bytes of main memory. Most nodes are equipped with at least an 86M byte Winchester disk and either a 5-1/4" floppy disk drive or a 1/4" cartridge tape drive.

The network's distributed file system is based on storage objects accessible from anywhere on the network. This distribution of resources allows sharing of programs, data and peripherals from anywhere on the network without the overhead of sharing computing power. A node can be a file server, a gateway to external communications, or a personal workstation. In addition, each workstation includes a high resolution display subsystem utilizing bit-mapped raster graphics which facilitates display of output from multiple programs simultaneously in multiple windows.

The native operating system provides a comprehensive set of standard computing operations such as compiling, binding and copying files/directories, as well as commands, pipes, filters and shell program interpreter. Additionally, the system offers a UNIX System V operating environment with mapping to the native operating system environment.

NETWORK TOPOLOGY

The LSDN spans several buildings within the boundaries of KSC as shown in Figure 1. Cable connections between buildings are fiber optics. Within each building workstations are connected to the token ring network with coaxial cable laid out in subloop rings and separated by network switches. Each subloop is configured based on the following guidelines:

- Each subloop contains an average of 10 workstations
- One server node or file server is placed in each physical ring with a full complement of the operating system plus all optional operating system software
- One node in each subloop contains a site registry of user accounts which will allow users to log on to the system even if the LSDN master registry becomes unavailable due to switching out a subloop
- The server node or file server in each subloop contains the UNIX permission files so that UNIX utilities are not affected by switching the subloop out
- All working directories for users residing within the boundaries of the physical subloop are located on the server node for that loop
- A low-capacity printer is provided in each physical subloop

Figure 2 illustrates a typical subloop configuration. Subloops are configured in this manner to allow all users to continue to be operational within the bounds of their working directories in the event of a failure elsewhere on the network. In the event of a failure on that subloop, it can be physically switched out of the network. Only the users of the inoperative loop are affected; whereas all other network users remain unaffected.

In addition to the physical subloops for user groups, the LSDN support group maintains a subloop of mass storage devices and peripherals which accommodate network-wide resources and databases.

Each node is loaded with a standard operating system based upon its configuration within the physical loop. Deviations from the standard are based on user requirements for the node. The configuration of the system software on the node can only be altered by a network system administrator.

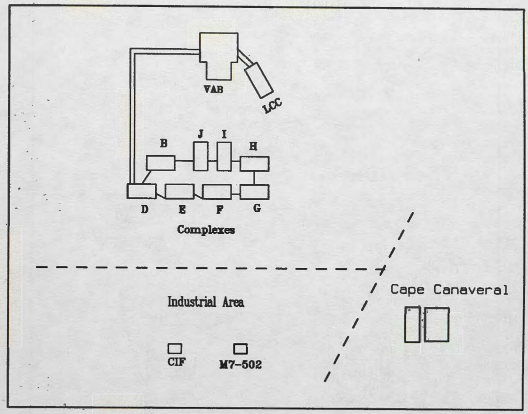

Figure 1 KSC LSDN Topology

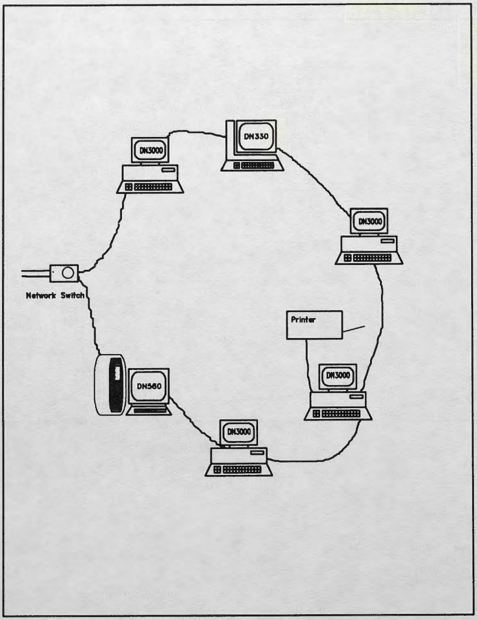

Figure 2 Typical LSDN Subloop Configuration

SYSTEM IMPLEMENTATION

During the prototype and first phase of LSDN implementation there were several projects initiated which demonstrated the capability to develop portable software, communicate with other computer systems and maintain access and configuration control over projects on a distributed bases. The following paragraphs describe some of the projects developed on the new system:

GOAL COMPILER

Previously all applications software to support Shuttle launch and ground operations testing was generated on terminals connected to the Central Data Subsystem (CDS) mainframe. These jobs were processed in batch mode and would be queued typically for 4 to 8 hours. In addition, engineers would take turns traveling to the building which housed the mainframe to pick up truck loads of compile listings which they would then sort and distribute to the software engineering department.

With the advent of the LSDN, a GOAL compiler was generated which could execute from any node on the network. While maintaining all source code in the existing CDS, software developers began transferring their code over a communications link to the LSDN, editing the code using the full screen editor, and then compiling the code with immediate test results before running the final compile on CDS. The result was an environment whereby the workload became distributed down to an individual level, giving more time to concentrate on the design of the code rather than on clerical tasks. Because of the multi-tasking capabilities of the system, developers could also take advantage of executing multiple compiles concurrent with debug activities. Additionally, the full screen edit cut-and-paste capabilities generated a time saving mechanism for copying common information between programs which had been typed laboriously in the past.

CCMS DEVELOPMENT SUPPORT

In the past, all development and testing of the Checkout, Control and Monitor Subsystem was performed on a software development laboratory of MODCOMP 11/45's. Since the lab was being disassembled to support the firing rooms, these developers were being forced to schedule time in the firing rooms to do code changes and compiles. One of the first tasks on the new system involved providing this group with a tool to perform their work.

A cross-compiler was developed and implemented whereby the engineers could maintain their source code on the host, control and track it automatically as they transferred it to the LSDN for changes, then return it in a controlled manner to the host. This has allowed the developers to work on a normal schedule and perform multiple concurrent tasks.

SHUTTLE CONNECTOR ANALYSIS

One of the most expensive tasks of the SPC has been insuring the flight readiness of the wiring within the Shuttle. There are over 7000 connectors containing more than .25 million connector pins involved in each orbiter. Each vehicle must undergo from 400-1000 connector demates,

mates and retests per launch due to changing payloads and other requirements. The effects of every such operation have to be thoroughly analyzed and extensively tested. The traditional method of handling these requirements has been manual analysis using engineering drawings.

Upon the procurement of the LSDN a team of senior engineers was employed to generate an on-line system for tracking changing configurations in pin connections. The system uses a frame based knowledge base with an "inference engine" and rule-like functions coded directly in LISP. The intent was to mirror the thinking of systems analysts as they trace shuttle wiring, while remaining fast enough to permit rapid analysis of a complex situation involving a vast knowledge base. Ultimately, each workstation on the network is able to access the latest status of any one of the Shuttle vehicles being processed. Furthermore, each node can independently perform "what if" and minimized impact approaches to work under consideration, or assist in troubleshooting vehicle test problems.

COMPUTER AIDED GRAPHICS

Prior to the introduction of the LSDN, drafting to support LPS Engineering was performed entirely on the drafting board by hand. Eighteen months after loading a CAD package onto the system, 50% of the engineering drawings were being maintained on-line. This time included training users who had virtually no experience with computer equipment.

Although the drafting group reports that training is a continuing effort, they enjoy working on the computer and learning new and easier ways to accomplish their tasks. They expect to continue to increase their ratio of drawings generated and maintained on the system to at least 90% since they have already discovered marked increases in throughput.

CONFIGURATION MANAGEMENT

To support change status accounting, implementation status accounting and baseline status accounting, development was begin to generate the Shuttle Data Systems Configuration Management System. Its purpose is to provide a single source of configuration accounting data for management of Shuttle Data System baselines. Prior to SPC, this function was managed by various contractors using individualized tracking systems, some of which were manual systems.

Because of the increased capabilities of the LSDN the following support can be afforded the users of the system:

- o Start to finish configuration accounting data to the end item identifier level
- o Automated extractions (via a user friendly interface) of requirement status reports supporting vehicle flows and/or milestones
- o Cross-referencing, audit trails, and backward/forward paper traceability
- o Integration of change package approval and implementation processing into a single database
- o Automation of as-designed versus as-built accounting

Because the system is generated and implemented on the LSDN, the developers and users are sharing both the environment and resources for a fully integrated tool.

SWITCH CONTROLLER PROTOTYPE

The Math Model group of LPS Application Software developed a model switch controller prototype to run on the LSDN. This controller is a graphic simulation of the Shuttle Orbiter cockpit switches and talk-back instrumentation.

The workstation Mouse is used to select any one of the Orbiter switch panels. A replica of the switch panel then provides the current status of each panel switch and talk-back. Mouse movement highlights the selected switch and button depression changes the switch position by sending the appropriate command via the serial port to the math model which is executing resident in a Honeywell 66/80. Switch position indicators within the model then change to reflect the current switch positions, enter the telemetry path, are decoded by Front End Processors, and arrive at the Firing Room buffers for display on color CRTs.

The graphics of the switch controller was coded using a software package available to all developers on the network. Logic code for math model communication was originally coded in Pascal, with the final version to be converted to the "C" language to ease porting to a standard UNIX system.

APLM CONFIGURE REQUESTS

In order to automate a wholly manual paper system, the System Build group of LSOC created an computer based APLM/Configure Request system. This allows an engineer using any of a couple hundred engineering workstations located throughout the Launch Complex 39 area to request a Firing Room application program to be installed in the Application Program Library and/or one of many Test Configuration IDentifiers (TCID). A TCID is the collection of System Software, Data Structures, and Application programs specifically combined to provide support for Vehicle testing/Launch or software development.

The ACR system validates the user's request using several criteria such as: Valid ACR access, proper TCID name and sub-level partition, and correct Support Software version identifier. The user provides data to the ACR through a user friendly Dialogue interface provided by the network environment.

The ACR routes the request through the proper groups for approval and finally presents the request to the System Build group for action. When the request is complete, the System Build group can send the completed status back to the originator. The request is also logged for historical retrieval. At any time, any valid user may status a request to see where it is in the processing chain.

Future enhancements of ACR will automate the integration of several requests and as a background process, connect to a host computer, activate the necessary routines to effect the user request, all without the need for human intervention.

DOCUMENTATION SUPPORT

Prior to SPC, documentation of software had been performed by multiple contractors on a variety of word processing systems, some of which were obsolete. Under the LSDN proof-of-concept it was demonstrated that software documentation could be transferred to the LSDN for on-line access, maintenance and review. Software was purchased that supports "what you see is what you get" full page editing with the capability to generate and integrate graphics on-line. The programmability of the software also allowed for generation of standard templates to support government publishing standards and graphics conventions.

This capability, distributed across the network, allows developers to generate documentation changes concurrently with their code changes using one tool. The documentation staff has also substantially reduced the need for outside assistance in the generation of graphics. Furthermore, the manual processes of generating table of contents, indices and cutting-in of artwork have been virtually eliminated.

To further illustrate this concept, this paper was generated with the documentation software loaded on LSDN using diagrams which are shared with other documents. Users located at different workstations around the network submitted information electronically to be included as ideas for the paper. The paper was printed on one of the laser printers which is accessible by any user on the network. All of this was accomplished while the author continued to perform other tasks to support the network concurrently with writing and editing of the paper.

CULTURE CHANGE

Because most of the functions performed in the Software Engineering environment had been primarily manual, the transition to the new networked environment proved to be a challenge.

Training

Since one of the primary objectives of the new environment was to provide an industry standard tool, UNIX was selected as the primary operating system in hopes of taking advantange of software engineers available in the marketplace. Because the majority of the existing work force consisted of developers skilled on the existing proprietary systems, most had to be trained for the new system. This impact was negligible, however, since the engineers were enthusiastic about learning new skills.

Standard Configuration

In implementing the new system, the network support group realized that the LSDN was different from most "industry standard" distributed networks in the area of configuration management. Typically, other sites do not have the requirement to support a controlled effort such as vehicle processing and therefore their nodes are "owned" by the developers who use them and can be configured in a variety of states. Since users of the LSDN need to expect a specific environment across the board, a standard had to be developed for controlling the configuration of all devices on the network.

This has been accomplished by creating a generic master load of the operating system and copying the master or a subset of it to all nodes on the network. By protecting the top level directories against user update, only system administrators are able to implement changes to the system. Engineering Support Requests are required for any changes in configuration, including the addition of optional software and peripheral devices. Configuration data is stored in a database maintained on the network.

NEW TECHNOLOGY

Probably the most significant change has been in dealing with new ideas. Along with the introduction of new tools came a plethora of new ideas on ways to use them. Therefore the largest burden in the entire upgrade has been for management to determine which innovations to implement and when.