N94-21343

DEVELOPING A SPACE NETWORK INTERFACE SIMULATOR: THE NTS APPROACH

Gary E. Hendrzak
Booz•Allen & Hamilton Inc.
7404 Executive Place
Suite 500
Seabrook, Maryland 20706
Tel: (301) 805-5423
Fax: (301) 805-9535

ABSTRACT

This paper describes the approach used by Booz•Allen & Hamilton to redevelop the Network Control Center (NCC) Test System (NTS), a hardware and software facility designed to make testing of the NCC Data System (NCCDS) software efficient, effective, and as rigorous as possible prior to operational use. The NTS transmits and receives network message traffic in real-time. Data transfer rates and message content are strictly controlled and are identical to that of the operational systems. NTS minimizes the need for costly and time-consuming testing with the actual external entities (e.g., the Hubble Space Telescope (HST) Payload Operations Control Center (POCC) and the White Sands Ground Terminal). Discussed are activities associated with the development of the NTS, lessons learned throughout the project's lifecycle, and resulting productivity and quality increases.

INTRODUCTION

NASA's Spaceflight Tracking and Data Network (STDN) provides continuous telecommunications coverage for low-earth orbiting spacecraft such as the Space Shuttle, the HST, and the Gamma Ray Observatory. The NCC, located at the Goddard Space Flight Center (GSFC), serves as the interface between the STDN and its customers, who primarily use the network to retrieve science and telemetry data from these spacecraft. The NCC consists of automated planning, scheduling, fault isolation, performance monitoring, communications, and display

241-242
PRECEDING PAGE BLANK NOT FILMED

systems (collectively called the NCCDS) that manage and control the network's resources.

Although the STDN offers a set of standard services to all science users, the addition of new users and new STDN elements requires modifications to the 438,000 lines of source code in the NCCDS. The most recent major change to the NCCDS was driven by the integration of a new Tracking and Data Relay Satellite System (TDRSS) earth terminal in White Sands, New Mexico, the Second TDRSS Ground Terminal (STGT).

The STGT integration requires intensive testing of new NCCDS software as well as tests of the changes to the interfaces between the NCCDS and each STDN user. The existing test system developed prior to 1983 was coded in assembly language and could not fulfill these test requirements. In addition, the user interface was cumbersome and supported only a single tester. The alternative was to test with the operational sites, which posed unacceptable risks to ongoing support of high profile user missions. Recognizing this situation, GSFC management commissioned the development of a new NTS in late 1989.

Figure 1 portrays the role of the NTS in the context of the STDN. Testers use the NTS to simulate and test all external interfaces to the NCCDS. The NTS can also validate operational scenarios, provide an off-line test platform for new NCCDS software releases, and collect a wide variety of test data for analysis by developers and operations personnel.

The testing process consists of three phases shown in Figure 2. The first phase involves the development of test scripts, messages, and timing delays to simulate actual operational scenarios. For example, schedule requests from the HST POCC would be sent to the NCCDS, validated, and acknowledged. In the second phase, the test scripts are transmitted over actual NASA communication (Nascom) lines to the off-line NCCDS, while logging all message traffic. In phase three, the message traffic is analyzed to verify that test objectives were met.

FIGURE 1
NTS IN CONTEXT

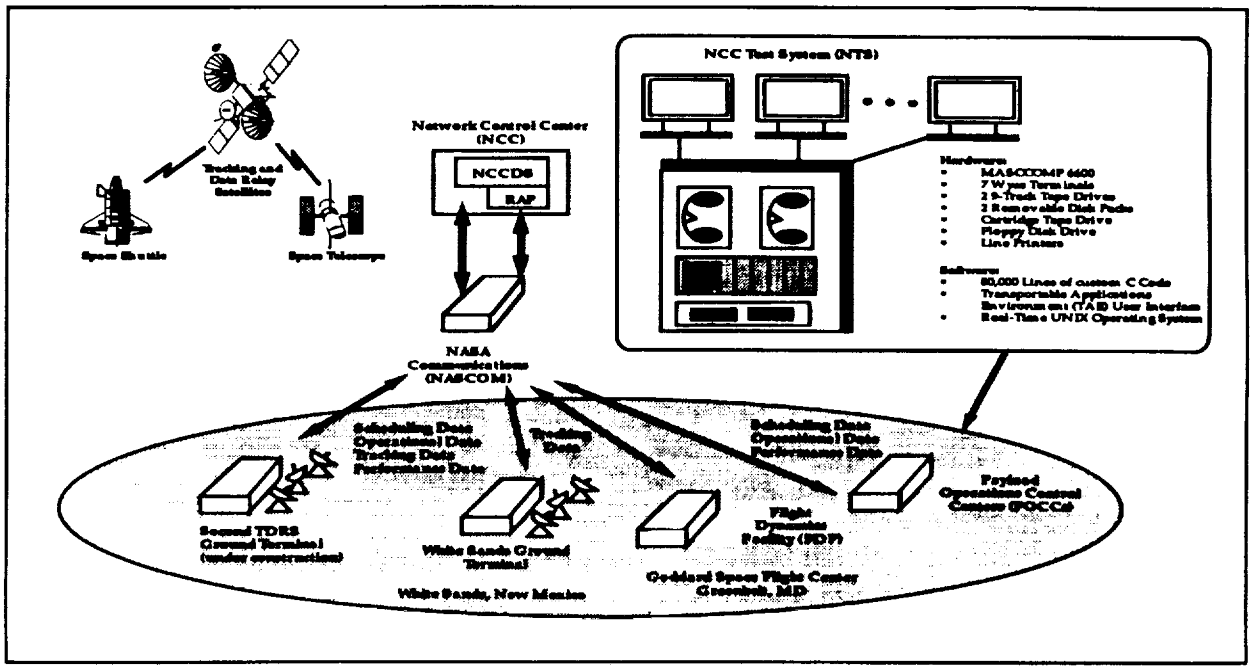
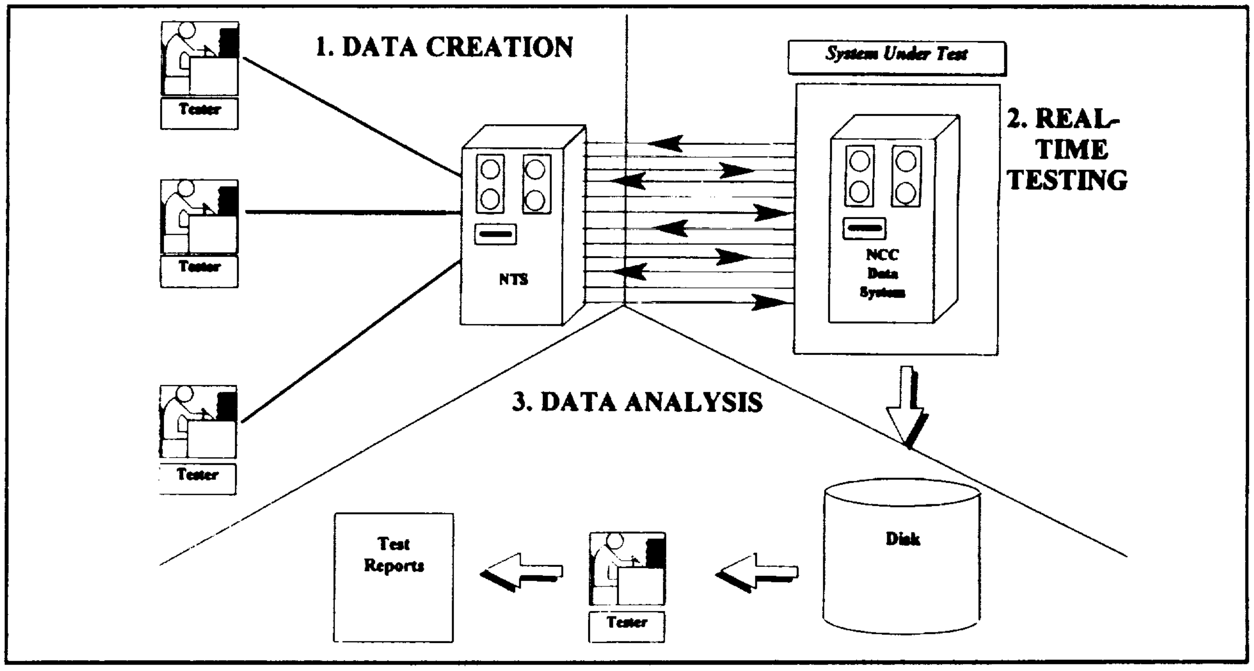

Tracking and Data Relay Satellites
Space Shuttle
Space Telescope
Network Control Center (NCC)
NCCDS
RAF
NCC Test System (NTS)
Hardware:
MASSCOMP 6600
7 Wyse Terminals
2 9-Track Tape Drives
2 Removable Disk Packs
Cartridge Tape Drive
Floppy Disk Drive
Line Printers
Software:
80,000 Lines of custom C Code
Transportable Applications Environment (TAE) User Interface
Real-Time UNIX Operating System
NASA Communications (NASCOM)
Scheduling Data
Operational Data
Tracking Data
Performance Data
Tracking Data
Scheduling Data
Operational Data
Performance Data
Second TDRS Ground Terminal (under construction)
White Sands Ground Terminal
White Sands, New Mexico
Flight Dynamics Facility (FDF)
Payload Operations Control Centers (POCCs)
Goddard Space Flight Center Greenbelt, MD

FIGURE 2
TESTING PHASES



DEVELOPMENT OBJECTIVES

In addition to those requirements documented in the System Requirements Specification [NTS93], several project goals were set. The system had to meet the schedule for the implementation of STGT-related changes in the NCCDS software. In addition, existing functionality and command syntax had to be replicated to minimize the learning curve required for NCCDS test teams. The human-machine interface had to be user-friendly and permit concurrent use. Finally, the NCCDS testing process had to be made more efficient by automating as many functions as possible. Likewise, the NTS development approach had to meet a set of objectives that the development team believed were vital:

- To perform only those tasks that directly added value and directly contributed to the success of the project
- To have end-users play a vital role throughout the project life-cycle
- To develop a system that promotes encapsulation, maintainability, modularity, extensibility, and re-use
- To define a process that is measurable, manageable, and repeatable.

To meet these goals and objectives, a phased implementation schedule was selected. The first release of the system met these major goals: duplication of the present functionality, implementation of the new functionality required for STGT-related modifications, as well as a new human-machine interface. Subsequent releases were used to continually automate the testing process, save time, and support more rigorous scenarios.

DEVELOPMENT APPROACH

The NTS development approach followed the traditional waterfall model in which the process cascades from one level to the next in a smooth progression [SEL92]. While this is neither unique nor groundbreaking, the model was deemed sufficient to meet the project's goals. However, the development team realized that a number of inefficiencies embedded in the development process had to be justified or eliminated to stay on schedule. Examples include excessive amounts of documentation, inefficient configuration management procedures, extended review and approval cycles, and responding to issues not relevant to the project. The development team believed that a more streamlined and flexible approach was preferable to the rigid, structured approach prescribed by the waterfall model. The philosophy of "Lean Software Development" described by Basili [BAS92] and based upon the work of Womack, et. al. [WOM90], seemed a perfect fit. This concept involves tailoring the development process to the needs of the product. Additionally, the Plan-Do-Check-Act cycle of Continuous Process Improvement espoused by Deming [DEM86] was applied to the development process, rather than to the product. As the project progressed, the entire process was continually refined and lessons learned were incorporated into subsequent development cycles.

Another key element in the approach was to include the NTS users in weekly functionality discussions and demonstrations aimed at specifying and clarifying new NTS requirements. The results of these meetings were captured and documented. Through numerous discussions, the NTS development team gained an in-depth understanding of the users' needs. This knowledge and first-hand experience allowed both developers and users to recommend and refine a number of enhancements that saved time during test sessions, increased the quality of testing, and decreased the amount of human-intensive analysis that was common to the testing process. Not only was testing more efficient in the NCC, but the new NTS eliminated most of the preliminary testing sessions with each of the 34 external entities.

The development team also determined that the content of the design reviews was not directly adding value to the project. All too often, no substantive issues were raised at the reviews, mainly because the attendees were users concerned with *what* the system would do, and not *how* it was to be implemented. With the approval of GSFC management, the number and content of the reviews were tailored to explain the system from a user's perspective. System features were discussed, followed by a brief overview of their implementation. Finally, an operations concept of the feature was presented using ToolBook^{TM}, a PC-based animation tool. These reviews, coupled with frequent human-machine interface demonstrations and a full day of hands-on training produced a system that exactly matched user expectations.

The development team also selected the Transportable Applications Environment (TAE) Classic for the user interface. TAE, developed for GSFC and maintained by Century Computing, consists of an interface that interacts with the user and manages the execution of application programs, while shielding the user from the host operating system. TAE provides a hierarchical menu system, on-line, context-sensitive help, parameter range checking, and a tutor mode to help new users build valid command strings. Thus, the users were required to learn only the NTS interface and not concern themselves with the operating system. By using TAE, the development team saved an estimated 630 staff days (approximately $250K) of development effort. The single user problem was alleviated by hosting the system on a Masscomp 6600 computer. The Masscomp is a Unix-based timesharing system that supports 16 concurrent users.

The software, developed in C, was designed with reuse in mind. Various standalone programs were developed to assist the user in developing test data, changing the system configuration, and analyzing test results. The human-machine interface to all of these tools is common and contains over 7000 lines of reused software. In addition, a library of common functions was developed, containing over 3500 lines of code. In total, over 18% of the software was reused in subsequent releases to implement new functionality.

Due to the development team's close working relationship with the user group, problems were usually resolved and tested on the development system *prior* to receipt of the official documentation describing the problem. In addition, the team foresaw a problem associated with dual mode use (classified vs. unclassified). Sanitization of over 1.3 gigabytes of disk storage would require 4 hours. The team recommended removable disk drives, resulting in sanitization time being reduced to only 5 minutes.

RESULTS

The results of the principles applied on this project can best be described quantitatively. Figure 3 presents software error rates for the three development cycles. Table 1 presents software productivity metrics, based upon the philosophy of Putnam and Myers [PUT92]. These statistics suggest that continual refinement of the development process

FIGURE 3
SOFTWARE ERROR RATES

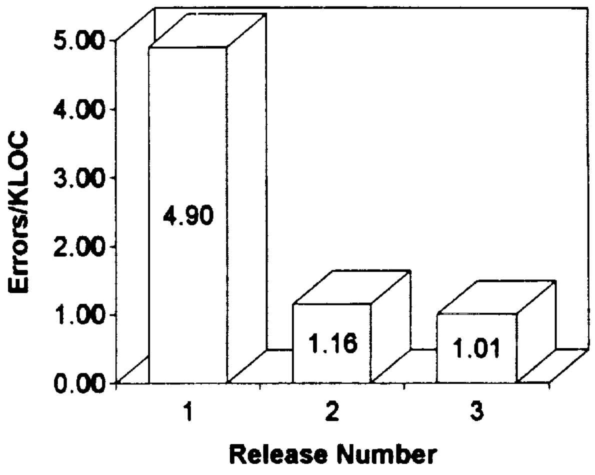


TABLE 1
SOFTWARE DEVELOPMENT METRICS

Version	Lines Of Code[1] (LOC)	Staff Months[2] (Effort)	Calendar Months[2] (Time)	Productivity Parameter[3] (PP)	Productivity Index[4] (PI)
Release 1	22,023	83.0	14.0	5315	9
Release 2	26,773	89.5	11.0	8690	11
Release 3	21,836	53.0	9.5	10261	12

NOTES

1. Booz•Allen-developed software only.
2. Effort and Time are calculated from the beginning of the design phase (following the Software Requirements Review) until conclusion of the code/unit test phase.
3. The Productivity Parameter (PP) is calculated according to the following equation:

$$PP = (LOC)/(Effort/B)^{(1/3)}\ (Time)^{(4/3)}$$

B, the special skills factor, is a function of size. For all releases, B = 0.18.

4. The Productivity Index (PI) is obtained from Table 2.3 of [PUT92].

resulted in higher productivity and lower error rates. User satisfaction ratings of the tool's functionality, completeness of the User's Manual, and the quality of training continues to be high. Suggestions for system enhancements and additional functionality were prioritized and addressed in each new release with no impact to cost or schedule.

In May of 1993, the NTS development effort was selected as one of five representative software projects to be part of a Booz, Allen, corporate-wide software process assessment based upon the criteria developed by the Software Engineering Institute (SEI). The Institute has developed an instrument to assess an organization's software development process. The results of this self-assessment showed that the NTS development team was functioning as a Level 2 organization while exhibiting many of the qualities characteristic of a Level 3 organization. These results are significant because the SEI process assessment procedure is geared more toward larger, more functionally segmented organizations (e.g., those having separate configuration management, quality assurance, test, and document preparation teams), whereas the NTS development team never consisted of more than 10 members.

CONCLUSIONS

Applying the principles of Lean Software Development and Continuous Process Improvement resulted in an increase in productivity and quality. This increase allowed for the delivery of additional functionality at no additional cost. Even though each release was successful, the development team continued to look for ways to improve and streamline the development process. Getting the user community involved from the very beginning and soliciting their input throughout the entire development process is a key strategy for success. Software development is by nature a dynamic process, constantly evolving and maturing. Change is a part of that process, and is not only necessary, it should be required.

The approach described here has resulted in the delivery of three separate releases of NTS software totaling 80,000 lines of source code. Each of these releases was delivered on or ahead of schedule and 3-5% under budget. The NTS is functioning as intended, allowing testers to perform more robust and exacting tests on the target software. In fact, during the first few weeks of operational use, testers using the new NTS uncovered several defects in the NCCDS software that had not been discovered by its developers or during any of the previous independent test phases.

The NTS provides a significant increase in tester productivity over the previous system, permitting simultaneous test data creation, test execution, and results analysis. The system was designed and documented to support future growth and changing requirements. It is a user-friendly test tool, decreasing the overall certification time of the NCCDS software, while greatly improving testing accuracy.

ACKNOWLEDGMENTS

The author wishes to thank Clint Provenza, Hellmut Scheel, and Bill Brooks for their comments on earlier drafts of this paper. In addition, the input and guidance of Keiji Tasaki and Roger Clason, GSFC project managers, and of course, the NTS user community, greatly contributed to the success of this project.

REFERENCES

[BAS92] Basili, V., The Experience Factory: Can it make you a 5?..., Proceedings of the 17th Annual Software Engineering Workshop, Goddard Space Flight Center, Greenbelt, MD, December, 1992.

[DEM86] Deming, W., Out of the Crisis, MIT Center for Advanced Engineering Study, MIT Press, Cambridge, MA, 1986.

[NTS93] NASA/GSFC, NTS Requirements Document, 530-SRD-NTS, February, 1993.

[PUT92] Putnam, L. and W. Myers, Measures for Excellence, Yourdon Press, Englewood Cliffs, NJ, 1992.

[SEL92] NASA/GSFC Software Engineering Laboratory, Recommended Approach to Software Development, SEL-81-305, August, 1986.

[WOM90] Womack, J., et. al., The machine that changed the world: based on the Massachusetts Institute of Technology 5-million dollar 5-year study on the future of the automobile, Rawson Associates; New York, NY, 1990.