The Network Operations Control Center Upgrade Task: Lessons Learned

T.-L. Tran and S. Lee
Software Product Assurance Section

J. S. Sherif
Software Product Assurance Section
and
California State University, Fullerton

This article synthesizes and describes the lessons learned from the Network Operations Control Center (NOCC) upgrade project, from the requirements phase through development and test and transfer. At the outset, the NOCC upgrade was being performed simultaneously with two other interfacing and dependent upgrades at the Signal Processing Center (SPC) and Ground Communications Facility (GCF), thereby adding a significant measure of complexity to the management and overall coordination of the development and transfer-to-operations (DTO) effort. Like other success stories, this project carried with it the traditional elements of top management support and exceptional dedication of cognizant personnel. Additionally, there were several NOCC-specific reasons for success, such as end-to-end system engineering, adoption of open-system architecture, thorough requirements management, and use of appropriate off-the-shelf technologies. On the other hand, there were several difficulties, such as ill-defined external interfaces, transition issues caused by new communications protocols, ambivalent use of two sets of policies and standards, and mistailoring of the new JPL management standard (due to the lack of practical guidelines). This article highlights the key lessons learned, as a means of constructive suggestions for the benefit of future projects.

I. NOCC Background

The Network Operations Control Center (NOCC) facility is one of three major components that comprise the Deep Space Network (DSN). The other two components are the Deep Space Communications Complexes (DSCCs), where the antennas and other front-end equipment are located, and the Ground Communications Facility (GCF), which provides communication links between the DSCCs, the NOCC, and flight project operations control centers (commonly referred to as POCCs). An example of such a POCC at JPL is the Space Flight Operations Control Center (SFOC) of the Multimission Operations Support Office (MOSO).

The NOCC is divided into two major subfacilities. One subfacility directly supports the DSN controllers and provides the day-to-day focal point for DSN operations-to-flight project interface. The other

produces operations support data for the Operations Engineering and Analysis (OEA) Group and the DSCCs and provides selected data products (e.g., tracking orbit data files) to its customers (i.e., flight projects and their respective organizational elements). The former is named NOCC-RT, since it deals mainly with real-time monitor-and-control operations. The latter is referred to as NOCC-NRT (nonreal time); this includes the NOCC support subsystem (NSS), the navigation subsystem (NAV), and the NOCC very long baseline interferometry processor subsystem (NVP).

The existing NOCC-RT is composed of old MODCOMP computers that can no longer be maintained at a reasonable cost. Moreover, their very limited memory capacity can no longer handle the increasing computing load. The existing software was written in MODCOMP assembly language and is not salvageable.

II. NOCC Upgrade Task

The scope of the NOCC upgrade task involved the redesign of the NOCC-RT and the replacement of the NOCC-RT hardware and software in its entirety. Installation of the hardware and software was divided into two main phases: Phase I and Phase II. Phase I hardware installation began in November 1990, and Phase II ended in July 1993. During this period, the existing MODCOMP real-time monitors (RTMs) continued to support old missions and whatever new missions were required.

The main thrust of the first phase involved the replacement of a subset of existing RTMs with Sun SPARC-1 workstations and new application software written in C language and utilizing a MOTIF-based graphical user interface (GUI). Phase II continued the transition to the high-performance Sun workstations and the replacement of the old MODCOMPs.

A. Elements of Successful Projects

Successful projects always exhibit an overall plan with shared purpose and goals and management's commitment to these goals. Also, and of equal importance, are the presence and support of dedicated and skilled process people. Tran summarizes the criteria for in-depth assessment of projects and gives attributes of projects that are considered successful.[1] These attributes include (1) consistent visibility of requirements, (2) well-enforced configuration control, (3) involvement of sponsors, users, and customers throughout the development life cycle, (4) support of a dedicated and skilled development team, (5) effective team communications, and (6) compliance to sound process policies or guidelines (for a disciplined and cost-effective development process).

Successful projects are those that meet valid functional (or design) requirements as well as user expectations; adhere to the spirit of process standards that promote rigor, discipline, and continuous improvement; and are accomplished on time and within budget.

B. NOCC-Specific Success Factors

Phase I of the NOCC upgrade task met 93 percent of the allocated functional requirements and was accomplished within budget and schedule. Additionally, the NOCC-RT system transfer agreement identified all known liens, and the delivered products met with user satisfaction, both upon initial transfer to operations and today. Details of the requirements traceability matrices for the NOCC-RT,[2] for frequency

[1] T.-L. Tran, "SSORCE In-Depth Assessment Worksheet," JPL Interoffice Memorandum LT:522-92 (internal document), Jet Propulsion Laboratory, Pasadena, California, p. 11, April 20, 1992.

[2] Ibid.

and timing,[3] and for telemetry and monitor[4] are contained in JPL internal documents. In summary, the NOCC upgrade task is considered a very successful project. NOCC-specific success factors include: (1) top management support, (2) focused ownership of requirements, (3) the team's can-do attitude, (4) use of enabling and extensible commercial technologies, and (5) a proactive software assurance function, supported and owned by program (i.e., TDA), project, and line management.

Top management support resulted in successful implementation of the project by assuring project visibility while minimizing organizational uncertainties. This support conveyed a perception of constancy in overall project purpose, and of mutual dependency—which united the entire team for a deeper commitment to the project plan and objectives.

Explicit and sustained ownership of requirements by system engineering throughout the product development cycle enabled a more systematic approach to requirements management, especially for better control of risk items, such as unexamined assumptions, neglected constraints, overspecification (or "gold plating"), misplaced or misassigned priorities, and excessive changes (or requirements-creeping syndrome). This focused ownership also facilitated the sharing of a single philosophy and a unified set of concepts, thereby contributing to easier identification of requirements-related problems and faster problem resolution or disposition. The real benefit of requirements management is a visible, common context of customer-driven needs and requirements, so that technical trade-offs between locally and globally optimized design alternatives can be made concurrently by different individuals or different subsets of the project team. Thus, the prescription of the overall architecture of the NOCC-RT, the layered approach to monitor-data presentation schemes, and spacecraft-and-link-oriented user interaction strategies were more openly debated, weighed, and ultimately more consistent with one another. (Refer to Ellman and Carlton [2] for more detail on NOCC-RT user interface design.)

Dedication of cognizant personnel and their enthusiasm during the two phases of the NOCC upgrade were major factors in the project success, but were even more significant to the first phase. The team's resilience in working with and around two sets of standards, namely the new JPL D-4000 and the old Telecommunications and Data Acquisition (TDA)/810-10 standard; its ability to upgrade the NOCC while keeping the old NOCC operational; and the can-do attitude, including toward incorporating unplanned, yet necessary, design changes, all contributed to a successful delivery of Phase I (and of Phase II).

The use of extensible technologies was and will remain a critical success factor for the NOCC-RT's overall flexibility and its access to different migration or evolutionary paths.

C. NOCC-Specific Difficulties

Yet, the road to "success" was not without hurdles. A few difficulties encountered by the NOCC upgrade task included the following: (1) ill-defined monitor-data interfaces with flight projects, (2) transition issues caused by new communications protocols, and (3) ambivalent use of two sets of policies and standards.[5]

Ill-defined data flow interfaces could be partially attributed to the fact that three DSN facilities (namely, the NOCC, SPC, and GCF) were being upgraded simultaneously. This complicated the devel-

[3] B. Falin, *Network Operations Control Center Subsystem Functional Requirements Document, Frequency and Timing Subsystem (1990–1995)*, 822-021 (internal document), Jet Propulsion Laboratory, Pasadena, California, January 15, 1989.

[4] D. L. Ross, *Network Operations Control Center Subsystem Functional Requirements Document, Telemetry and Monitor Subsystem (1991–1995)*, 822-25, JPL D-6101 (internal document), Jet Propulsion Laboratory, Pasadena, California, February 1, 1989.

[5] T.-L. Tran, *Network Operations Control Center Subsystem Functional Requirements Document, Monitor and Control Subsystem (1990–1995)*, 822-23, JPL D-5783 (internal document), Jet Propulsion Laboratory, Pasadena, California, June 30, 1989.

opment and coordination of the interface specifications with the 26-m subnet projects and one primary DSN customer (i.e., Mars Observer).

The transition issues caused by the dual use of commercial off-the-shelf (COTS) protocols and DSN new communications protocols were extremely problematic and required extra effort and time to be resolved properly. This diverted project resources (equivalent to a full calendar month) into the dispositioning of this problem.

The ambivalent use of two sets of policies and standards, namely the new JPL D-4000 standard and the old TDA 810-10, caused some confusion. This is due to the fact that the JPL D-4000 is a rather high-level process model that lacks practical application guidelines. Also, the old TDA 810-10 standard calls for reviews and milestones that differ in rigor and scope from those of JPL D-4000.

Frequent unavailability of the old NOCC for testing is a somewhat uncontrollable problem, since the facility is required to be operational all the time. This problem may have delayed the project integration schedule. However, this delay was not significant.

III. Lessons Learned

The success of Phase I of the NOCC upgrade capped a major transition step for the DSN, going from an environment of memory-limited, single-sourced computing systems that are kept alive with cannibalized parts to a distributed environment of Unix-based, reduced instruction-set computer (RISC)-architecture workstations connected with one another via the transmission control protocol/user datagram protocol/internet protocol (tcp/udp/ip) protocol suite. In a period of budget constraints and rapid technological change, the upgraded NOCC continues its move toward an open-system support environment—while undergoing a few application-level adjustments. The decision to adopt the "open system" architecture, and to use commercially available communications protocols, has propelled the NOCC into the 1990's and will have a profound impact on the way the futures of the NOCC and JPL ground data systems evolve. Additionally, there are some aspects of the product development process that deserve special analysis. In order to make the lessons-learned more relevant, this section proposes to present "findings" that reflect the local particularities of the NOCC upgrade and associated "recommendations" that generalize from the findings and make them portable to other project or application settings.

A. Findings and Recommendations on Requirements Management

The NOCC experience entails software development "in the many," i.e., major modifications to 8 subsystems and 25 major interfaces and integrating these changes into the operational environment—and all of this without impacting real-time spacecraft support. (Refer to Fig. 1 for a context diagram of the NOCC-RT.) Against the background of high DSN-availability requirements for around-the-clock nominal and emergency spacecraft tracking, the DSN/NOCC functional requirements documents have shown that system-level and subsystem-level requirements can support, qualify, constrain, or presuppose one another. Furthermore, some requirements can be hidden assumptions. Others are design goals, and a few others are implementation or technology goals disguised as implementation constraints. Such requirements contamination prevails and represents one end of the spectrum, although the extent to which it occurs varies from one subsystem (or application) to another. This finding had also been confirmed by a field study, as discussed in [5,6]. It was found that the traditional way of requirements capture by prose-like, unstructured, obscure, and somewhat ambiguous statements is no longer effective. New techniques and tools for requirements engineering are advocated by Goguen and Linde [3], Smith [7], Ohnishi and Agusa [6], and Macaulay [5]. At the other end of the spectrum, requirements can be orthogonal to one another. This high level of rigor is sometimes worthwhile pursuing, sometimes not.

In the absence of formal techniques or without the advent of some expert tools, specifying valid or orthogonal requirements remains an elusive goal. Research in the knowledge-based systems and tools for formal specification is now undergoing a revival of great significance, as noted in [4].

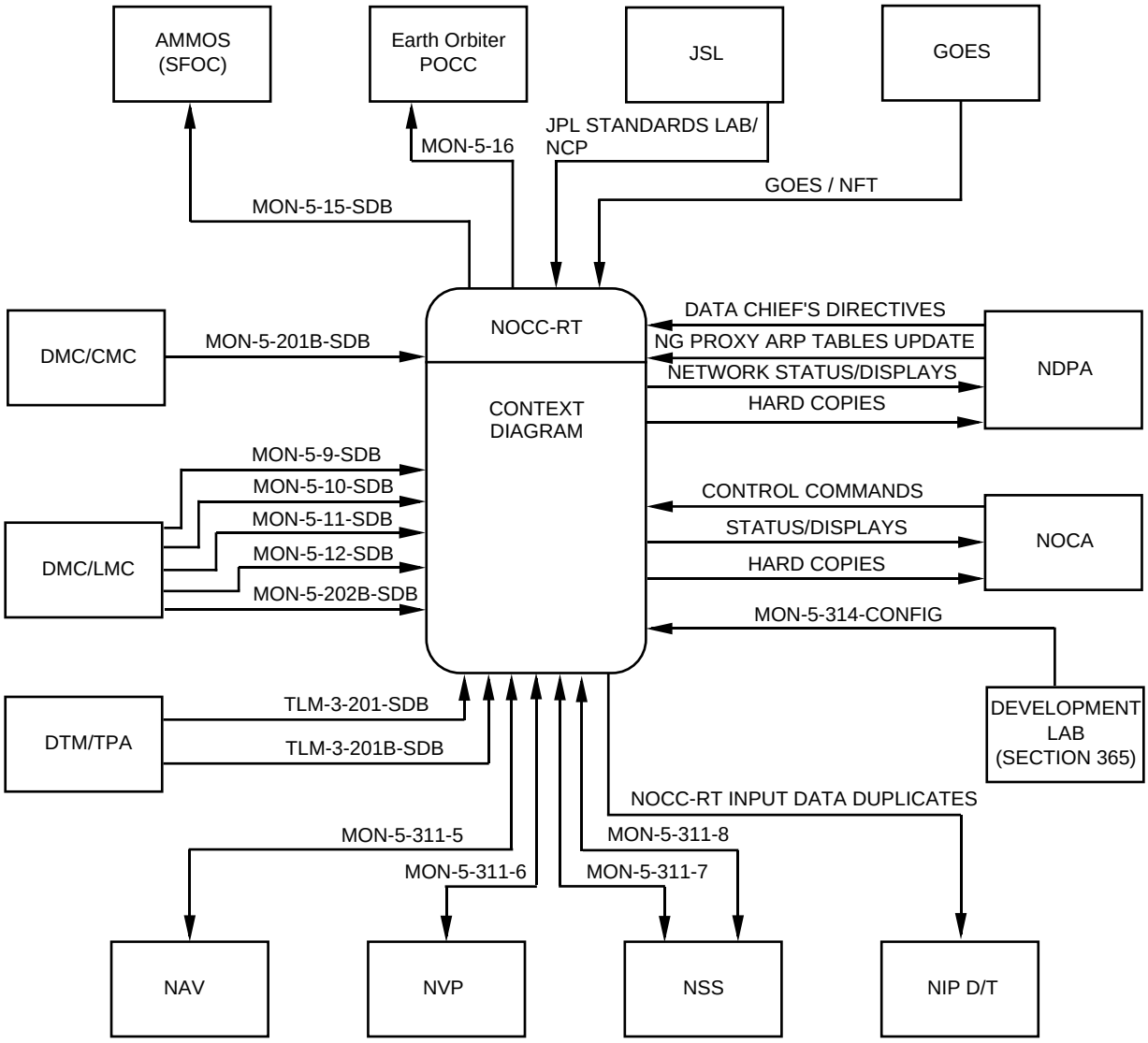


Fig. 1. NOCC-RT context diagram (Phase I).

The recommendations for requirements management include the following:

(1) There must be an automated facility or tool to support and enforce the requirements management process, thereby making it more cost effective.

(2) The customers, users, system engineers, developers, and test engineers should all work as a team in validating the requirements up front. Of equal importance is the thorough assignment (and reassignment, as appropriate) of requirements priorities. And, the system engineer (instead of a manager) must be the person in charge of sharing this information about requirements, up front and continuously, with all team members.

(3) The requirements management process must be owned by a single individual, and preferably by the system engineer (or his or her delegate, such as a subsystem engineer or a process assurance analyst). This owner must also be responsible for requirements baselining and requirements-database configuration control to ensure a minimum number of hand overs.

(4) Since the user representatives (e.g., DSN operations engineers) are key players in acceptance testing, they must be responsible for reviewing and concurring with the acceptance criteria specified for each and every high-level requirement (or "customer acceptance criteria"). This buy-in concept is a critical success factor for alignment between user expectations and planned capabilities.

B. Findings and Recommendations on Team Management

Many factors make team management of software projects different from other engineering fields. Software is generally more complex than other engineering projects. It requires a disciplined environment that enables creativity rather than a regimented environment where accountability and commitment to quality are lost. Quality is understood here in the context of total quality management, where the product meets user expectations at acceptable costs. As previously pointed out, effective team communications have always been an element of overall project success [1].

The recommendations on team management include the following:

(1) At the outset, project management should establish a project-wide communications infrastructure. Such an infrastructure includes physical connectivity as well as a hundred-percent person-to-person connectivity. Central to this infrastructure is the existence of a common information repository that is accessible to each and every team member. This would encourage and enhance team building by enabling the sharing of information on requirements changes, design changes, etc., among the whole team, sponsors, customers, and users. It is essential to recognize that hard wiring team members is an alternative (e.g., via collocation), but should not be the only alternative. Soft wiring of all team members and project management via electronic mail is a must.

(2) Also critical to effective communications is the establishment of an agreed-upon product-development process model. Such a model must have clear definitions of the control process (including any hand-over points), well-identified decision-making points, risk factors, cost constraints, and schedule agreement.

(3) Built-in mechanisms must be in place to monitor the state of alignment between management's understanding and that of the team members regarding the goals, objectives, and constraints of the project; these include (a) what is to be built, (b) who or which subteam is responsible for performing what piece of the to-be-delivered product, (c) who the collaborators are for each individual or subteam, (d) how the work is to be performed (i.e., in how many phases and with how many deliveries), (e) what the reporting and control requirements are, and (f) the providing of adequate training on the selected development techniques and tools (e.g., object-oriented analysis and design methodology and tools for source code control and requirements traceability).

(4) Project management must optimize the team size in proportion to the effort required for each phase of the life cycle of the project. Baseline estimates compiled from industry show that it is more economical to invest in quality up front (e.g., in the preproject or requirements phase) than in the back end of the process (e.g., in the coding/full-scale production phase or integration-and-test phase). This is, indeed, consistent with the tenets of total quality management, which have reaffirmed that error prevention and process quality provide more value for the money than error correction, late error correction, and work arounds (via product inspections) [4].

C. Findings and Recommendations on Process Policies

Implementation should be consistent with the model and standards that are agreed upon by project management from the start of the project. For example, following the spiral model of software development is different from following the waterfall model [1]. The major difficulty with NOCC-RT process

policies was the dual use of standards that come from two different inheritances, as was pointed out earlier. The recommendations on process policies include the following:

(1) Do not select a standard that is too general, too difficult to tailor, or too difficult to interpret.

(2) Use industry standards that conform to the task at hand and that can provide application guidelines with specific and concrete examples for various software activities (e.g., software requirements analysis or design), for various application families (e.g., general purpose versus embedded), and for various methodologies (e.g., structured analysis, object-oriented analysis and design, or data analysis and design).

(3) The effort allocated to implementation should be based on concrete estimates and quantified uncertainties and be aligned with the effort estimates suggested by an industry baseline [1].

(4) Every process policy must be supported and enforced by some automated tool. Otherwise, drop that policy. In other words, have as few laws as possible, but make them sensible and enforce them.

D. Findings and Recommendations on the Integration-and-Test Process

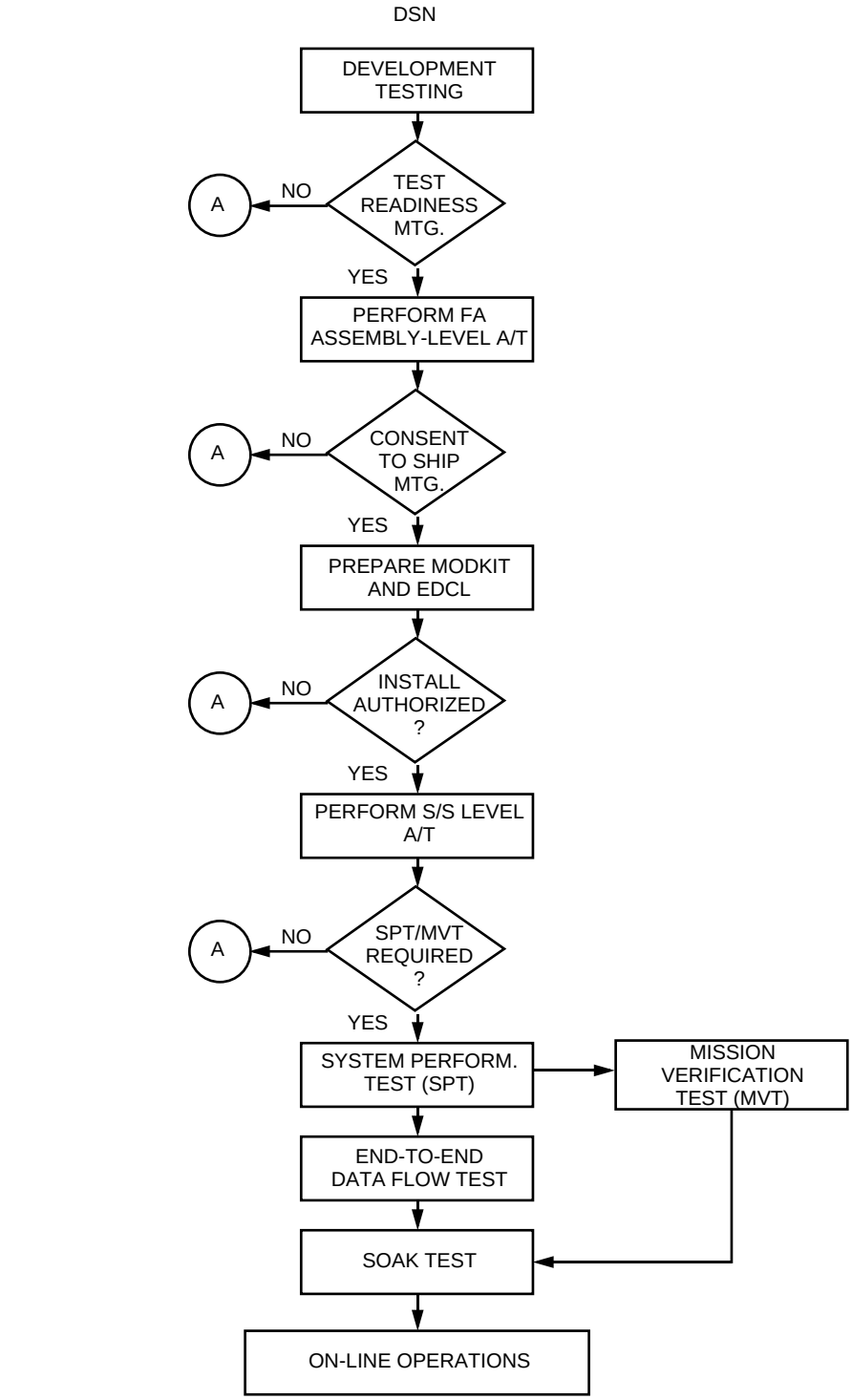
Integration and test (I&T) is a recognized critical aspect of product development. The DSN's I&T process has various levels of formality, which ultimately prepare it for mission-support readiness. These levels include unit (or module) test, assembly-level integration, preacceptance test, acceptance test, DSN data flow test, and mission verification test. An overview of this process is depicted in Fig. 2. In its current state, the DSN process is hampered by an excessive number of hand overs, a paperwork-intensive sign-off subprocess that is always out of synchronization with its objectives; also, its ownership is distributed over TDA's system engineering (responsible for defining functional requirements and associated acceptance criteria), TDA's implementation engineering (responsible for allocation of budget and approval of project implementation schedule and management), project management from the technical organization (responsible for overall project achievement, including development testing through acceptance testing), TDA Operations (responsible for system-level testing and assuring operational readiness of the DSN), and TDA Operations' main contractor (responsible for actually performing the system performance/mission verification, DSN end-to-end data flow, and soak testing). It is also worth noting that the current culture behind testing does not recognize the importance of ultimate customer acceptance and satisfaction as a driver. Accordingly, testing has been considered to be a "less critical" appendage of system engineering and has become a de facto back-end process focused on identifying problems and liens.

The recommendations on I&T include the following:

(1) Test planning must start at the beginning of the project, concurrently with requirements analysis or capability planning, or as early as possible. Unless there is a specific test-plan sample to emulate, do not follow to the letter a standard or guideline that is too general or asks for too many test-design details too early (such as test-case detailed objectives). This recommendation is especially applicable to new implementations (as opposed to minor upgrade or sustaining projects, which have the convenient knowledge of an existing architecture and configuration).

(2) Acquire or draw up the test process diagram that is imposed on the project (as a matter of policy), and share it with all team members. Ensure that system engineering, test engineering, and project management have a common understanding of the hand-over points and sign-off requirements along the I&T process. Do not assume that some "old-timers" know it all. If it is not in writing, the likelihood that their knowledge is not aligned with the de facto practice is extremely high (if not certain).

(3) The effort allocated to I&T must reflect the major steps of the process diagram, the hand-over points, and the decision points for go/no go (or pass/no pass). Accordingly, the work breakdown structure and schedule for I&T must be consistent with these steps and points.



A RETURN DEPENDENT UPON CIRCUMSTANCES/DIRECTION OF REVIEW BOARD, e.g.,
(a) EXECUTE FURTHER TESTING AT SYSTEM OR SUBSYSTEM LEVEL
(b) RETURN TO DEVELOPMENT FOR REDELIVERY
(c) REMOVE OPEN LIENS

Fig. 2. Test process flow diagram.

IV. Conclusions

The NOCC upgrade task met all functional and design requirements as well as user expectations and adhered to the spirit of process standards that promote rigor, discipline, and continuous improvement; it also was accomplished on time and within budget.

Acknowledgments

The authors would like to express their sincere thanks to Dr. Joseph H. Yuen, Editor, *The Telecommunications and Data Acquisition Progress Report*; Dr. Li Tien-tien, Distributed Computers, Advanced Laboratory for Parallel High-Performance Applications; and Patricia A. Ehlers for comments and suggestions that greatly improved this article.

References

[1] B. Boehm, *Software Engineering Economics*, Englewood Cliffs, New Jersey: Prentice Hall, 1981.

[2] A. Ellman and M. Carlton, "Deep Space Network (DSN), Network Operations Control Center (NOCC) Computer-Human Interfaces," *Proceedings, Second Int. Symposium on Ground Systems for Space Mission Operations*, Pasadena, California, pp. 561–564, November 16–20, 1992.

[3] J. A. Goguen and C. Linde, "Techniques for Requirements Elicitation," *Proceedings, IEEE Int. Symposium on Requirements Engineering*, San Diego, California, pp. 152–164, January 4–6, 1993.

[4] K. Ishikawa, *What is Total Quality Control? The Japanese Way*, Englewood Cliffs, New Jersey: Prentice Hall, 1985.

[5] L. Macaulay, "Requirements Capture as a Cooperative Activity," *Proceedings, IEEE Int. Symposium on Requirements Engineering*, San Diego, California, pp. 174–181, January 4–6, 1993.

[6] A. Ohnishi and K. Agusa, "CARD, A Software Requirements Definition Environment," *Proceedings, IEEE Int. Symposium on Requirements Engineering*, San Diego, California, pp. 90–93, January 4–6, 1993.

[7] T. J. Smith, "READS: A Requirements Engineering Tool," *Proceedings, IEEE Int. Symposium on Requirements Engineering*, San Diego, California, pp. 94–97, January 4–6, 1993.