EXPERT SYSTEM DEVELOPMENT METHODOLOGY AND THE TRANSITION FROM PROTOTYPING TO OPERATIONS: FIESTA, A CASE STUDY

Nadine Happell and Steve Miksell
Stanford Telecommunications, Inc.*+

Candace Carlisle‡
NASA/GSFC

ABSTRACT

A major barrier in taking expert systems from prototype to operational status involves instilling end user confidence in the operational system. End users want assurances that their systems have been thoroughly tested, meet all their specifications and requirements, and are built based on designs which are reliable and maintainable. For most software systems, the waterfall life cycle model can provide those assurances. However, this model is inappropriate for expert system development, where an iterative refinement approach is commonly employed. This paper will look at different life cycle models and explore the advantages and disadvantages of each when applied to expert system development. The Fault Isolation Expert System for TDRSS Applications (FIESTA) is presented as a case study example of development of an expert system. FIESTA is planned for use in the Network Control Center (NCC) at Goddard Space Flight Center in Greenbelt, Maryland. The end user confidence necessary for operational use of this system is accentuated by the fact that it will handle real-time data in a secure environment, allowing little tolerance for errors. The paper discusses how FIESTA is dealing with transition problems as it moves from an off-line standalone prototype to an on-line real-time system.

*1761 Business Center Dr., Suite 400, Reston, VA 22090

+FIESTA development has been undertaken by Stanford Telecommunications, Inc. as a subcontractor to Computer Sciences Corp. under NASA contract NAS5-31500.

‡Code 532.3, NASA/Goddard Space Flight Center, Greenbelt, MD 20771.

1.0 DEVELOPMENT LIFECYCLE MODELS

When discussing software development methodologies and lifecycle models, there are two which are readily seen as the most common: the waterfall model, and iterative refinement. Of these, iterative refinement is by far the most common for expert system development. In fact, some experts claim the use of iterative refinement for expert system development "seems inescapable" [1].

The basic approach behind iterative refinement is to start off with a basic concept of what the system should be like, build a prototype, and evaluate what modifications need to be made to this prototype to bring it closer to the desired system. Modifications are made, and the system is continuously reevaluated, until a satisfactory system exists. This approach is generally used because hard, fast requirements for the system can not be specified ahead of time. Basically, the developer and user must tinker with the system, experimenting to see what will work the best. An advantage to this approach is that it provides a rapid operational capability. The very first prototype could potentially be of limited use to the end user, even though it may be lacking in many areas. And each prototype provides a realistic base on which to build the next round of improvements.

There are problems to this approach however. After numerous modifications and reworking, the code tends to lose its integrity, resulting in the expert system equivalent of spaghetti code [2]. Also, quick, temporary fixes have a tendency to become permanent before long-range architectural concerns are addressed. These problems can make the system difficult and expensive to maintain.

The classic answer to these problems has been to use the waterfall model, which advocates the stagewise development shown in Figure 1. Additional steps like cost analysis, hardware studies, and feedback loops can be added, but the basic approach remains the same. In the waterfall model, the requirements must be defined before the system is designed, the design must be complete before coding starts, and the system must be thoroughly tested before going operational. This model has become the standard for most conventional software development because it provides the end user with confidence that the system meets his requirements, has been thoroughly tested, and is based upon a design which facilitates maintenance. However, most expert system applications can not complete one of the first steps, requirements definition, until several extensive prototypes have been built. And the prototypes can't be adequately tested until they are put in an operational environment.

There are software engineers who are exploring new models in order to balance out the advantages and disadvantages of these two approaches, the most notable of which is Barry Boehm's Spiral model [3]. However, this model still has some problems, and is not yet ready for general application outside the research environment. So for the present, developers are restricted to using iterative refinement, the waterfall model, and derivatives thereof.

For applications in which the expert system can be tested in an operational environment without impacting existing operational systems, the iterative refinement approach may be sufficient, even if it isn't optimal. However, there are applications, such as the Fault Isolation Expert System for TDRSS Applications (FIESTA), which can not be tested on-line without the potential for affecting existing operations, and therefore require a more rigorous approach. This paper will look at the FIESTA project and how it is trying to balance the need for an iterative refinement development approach with the end user's need for a trustworthy system before going on-line.

2.0 FIESTA

2.1 BACKGROUND

2.1.1 FIESTA/Problem Domain

The purpose of the Fault Isolation Expert System for TDRSS Applications (FIESTA) prototype is to automate fault isolation and service monitoring for the Space Network (SN). The SN, illustrated in Figure 2, provides NASA's primary means of communication with satellites and the space shuttle. It is composed of Tracking and Data Relay Satellites (TDRS), the NASA Ground Terminal, the White Sands Ground Terminal, and the Network Control Center (NCC). Data from spacecraft are relayed through the TDRS, then sent through White Sands and the NASA Ground Terminal to each spacecraft's control center. The NCC at Goddard is responsible for scheduling use of the space network's resources, monitoring the quality of the services it provides, and diagnosing problems. High-speed messages called schedule orders (SHOs) transmit the space network schedule from the NCC to White Sands. Fault Isolation Monitoring System messages, operations messages, and operations data messages are sent between White Sands, the NASA Ground Terminal, and the NCC; console operators in the NCC can use the information in these messages to monitor service quality and diagnose problems. Console operators currently do this job by viewing cluttered, number-filled displays, detecting when a problem occurs, and determining an appropriate course of action. These tasks are known as service monitoring and fault isolation, and are extremely labor-intensive.

This application of human expertise currently provides acceptable levels of service monitoring and fault isolation. However, providing for projected growth in network loading, while retaining the current levels of performance requires automating many of these functions.

Translating expertise into conventional software requirements is extremely difficult.

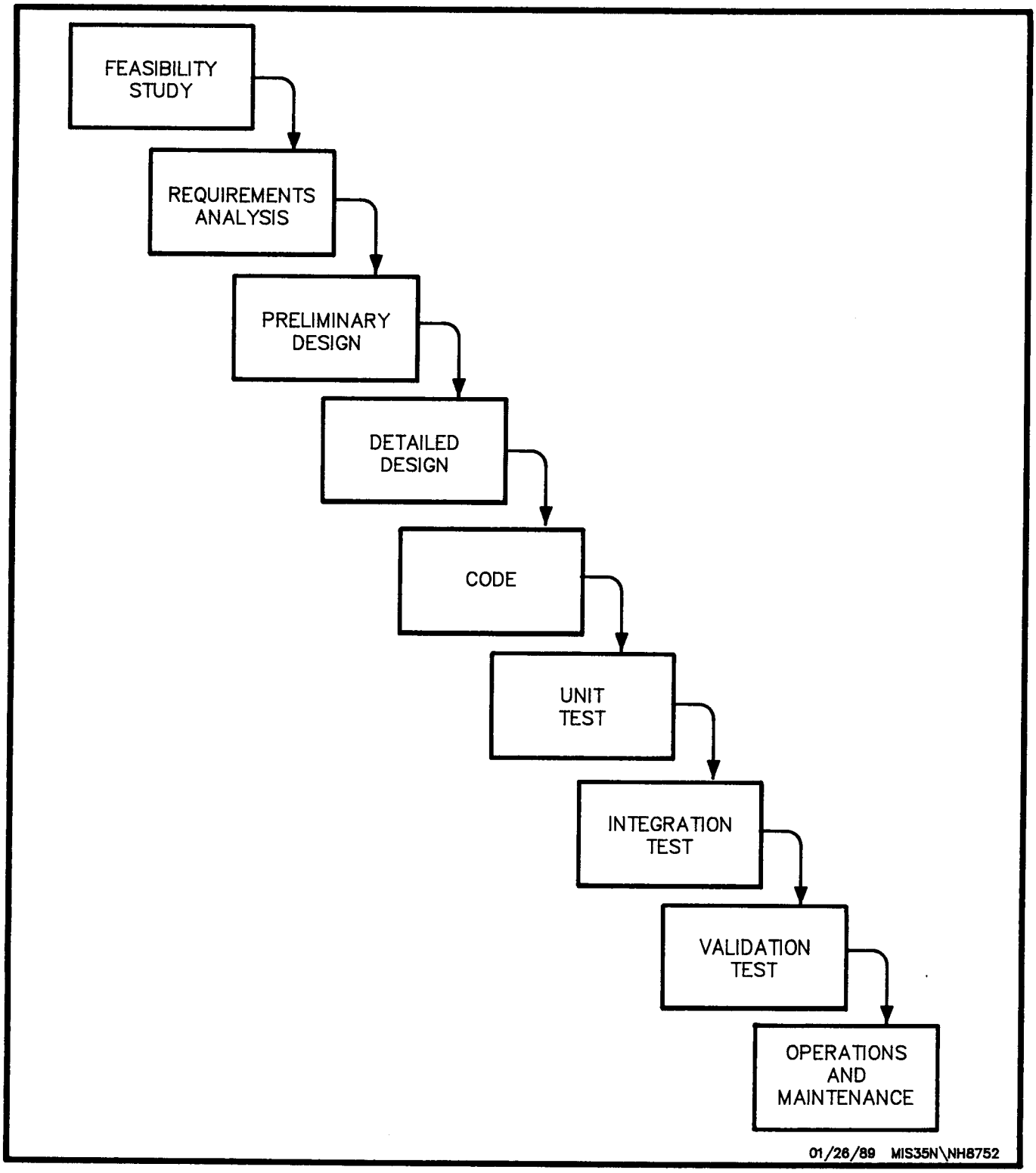

FIGURE 1: WATER FALL MODEL

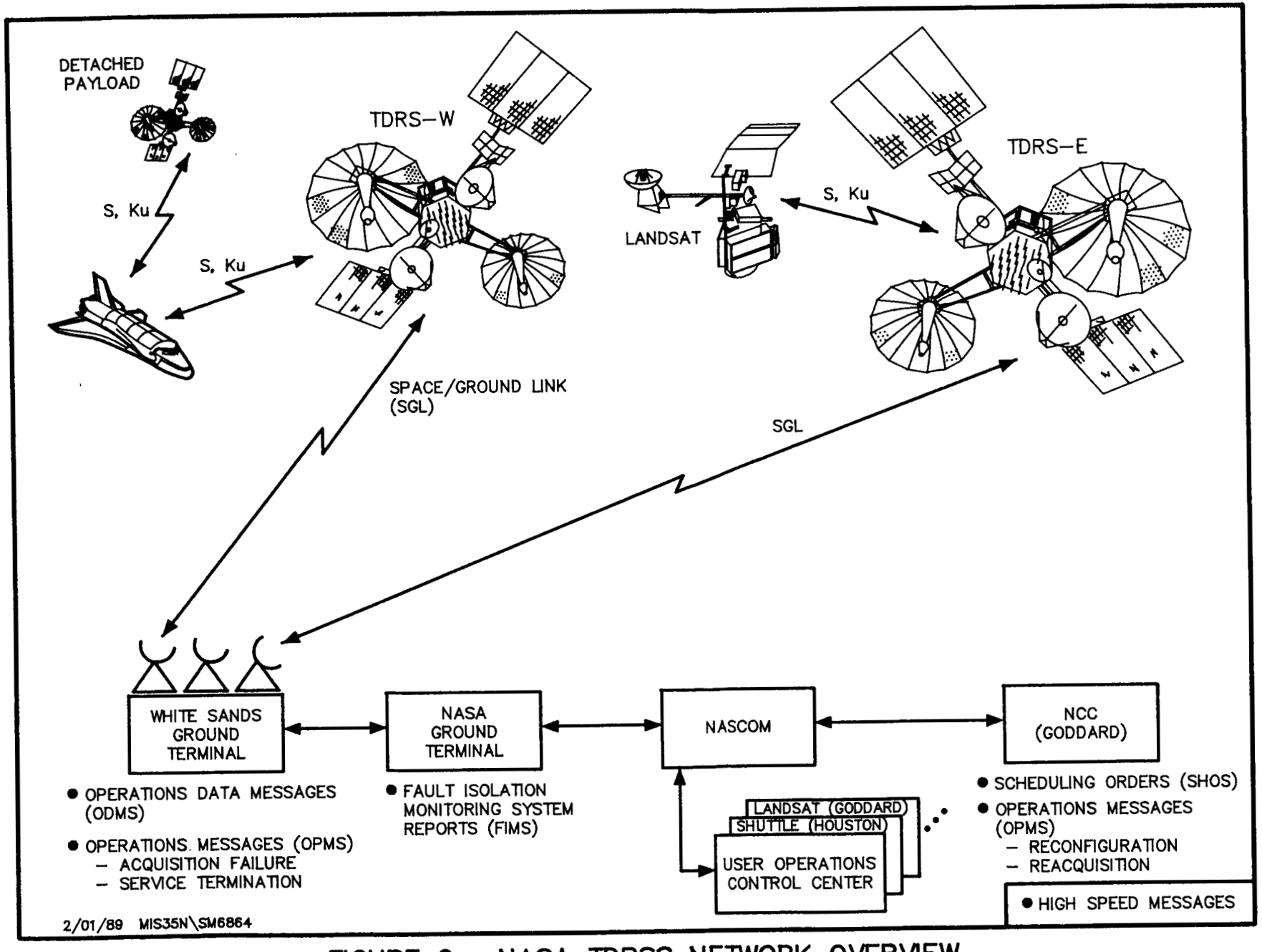

FIGURE 2: NASA TDRSS NETWORK OVERVIEW

However, fault isolation and service monitoring are good candidates for expert system automation because a "rules of thumb" diagnostic process is used, and experts are available.

2.1.2 The FIESTA Approach to Expert Systems Automation

Since an expert system has never been used in the NCC environment, integration of a fault isolation/service monitoring expert system into the NCC was considered a high risk project. To minimize this risk, it was decided to use a 2-step prototyping approach. First, a standalone off-line prototype would be developed by Stanford Telecommunications (STel) on a Symbolics lisp machine. This prototype would receive its input from old NCC messages, archived on "log" tapes. This step would provide a proof-of-concept. Second, the expert system prototype would be put on-line in the NCC, receiving its message input in real-time. The FIESTA second stage prototype would then be used as a tool by NCC console operators.

The operators' hands-on experience would allow requirements and an operations concept for an operational service monitoring and fault isolation system to be developed.

The first stage of the prototype has already been developed and successfully demonstrated to operations personnel. The prototype has been used to diagnose old shuttle, as well as services for spacecraft such as ERBS, SME and LANDSAT. FIESTA's results have been compared to the real-time diagnoses made by the console operators. Currently, the second stage (on-line) prototype is being implemented. A System Requirements and Critical Design Reviews have been held. These reviews are typically held for all Goddard software projects.

In order to bring FIESTA on-line, the Symbolics will have to interface with the NCC's Communication and Control Segment (CCS) computer via a local area network (LAN). Existing code on the CCS computer will be modified to provide messages to FIESTA, code must be installed in the CCS to translate messages into s-expressions, and LAN interface software for the Symbolics will be written. Thus, on-line FIESTA software consists of 4 components: modified CCS software, expert system, LAN interfaces and new CCS software.

2.2 FIESTA LIFECYCLE

2.2.1 Overall Development Approach

The FIESTA expert system implementation has emphasized iterative development and the refinement of requirements based on frequent demonstrations allowing operations feedback. Structure has been imposed on this process by the use of "builds". Successive builds, each with clearly defined requirements and objectives, were characterized by increasing levels of complexity. Figure 3 provides an overview of this development approach.

The first three builds served to refine the operational concept and supported the development of realistic requirements. This phase also included the establishment of the development configuration illustrated in Figure 4. Two features of this development approach have been particularly significant in the transition to operational deployment. These were (1) the use of actual (archived) data to drive development and (2) the creation of an independent front end processor (FEP) to access the archived data and emulate the NCC environment.

Use of actual data provided clearly defined requirements in terms of inputs to the system. Accessing this data via the standalone FEP (emulating the NCC) allowed expert system development to proceed as if it were actually in the target environment. For the expert system component, changes required for operational deployment were concentrated in areas where emulation differed from reality.

The final three builds focused on the transition to the operational environment. Performance Issues (Build IV) were concerned with operation in a real-time environment. Detailed results of this effort are presented in [4].

The expanded mission support (Build V) was accomplished using a replicate and merge approach [5]. This stage also provided the opportunity to restructure the rule organi

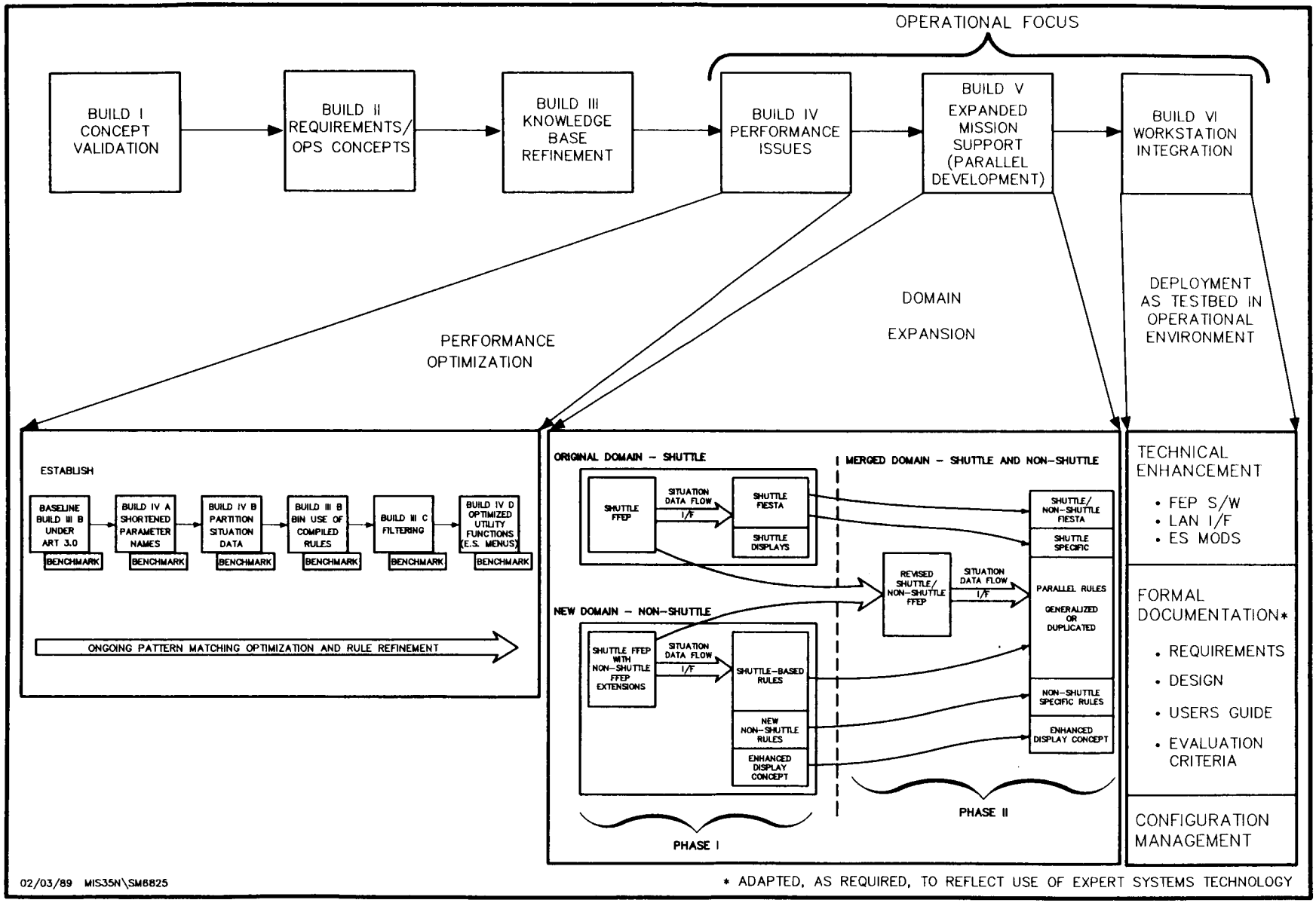

FIGURE 3: FIESTA DEVELOPMENT APPROACH

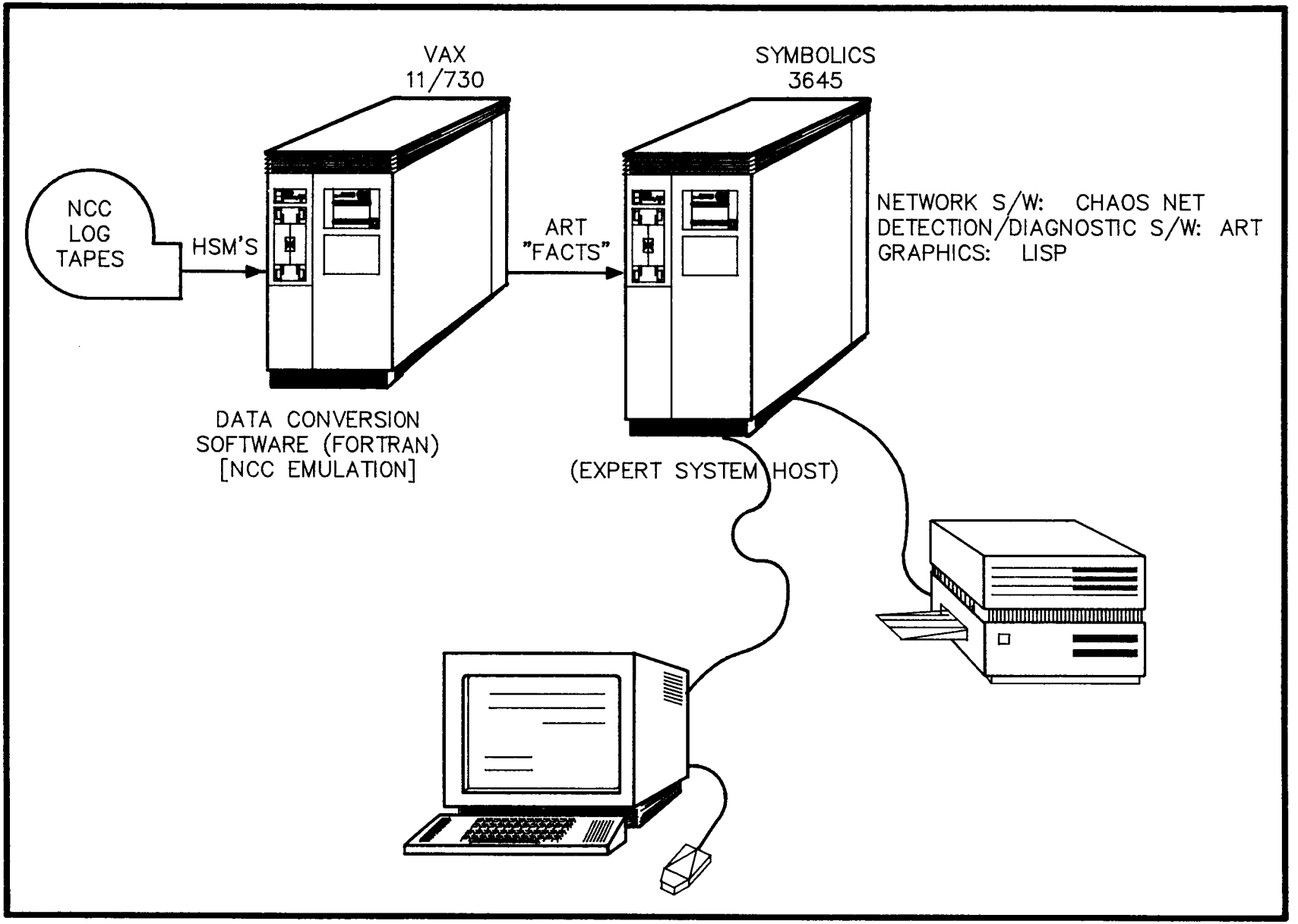

FIGURE 4: DEVELOPMENT CONFIGURATION

02/03/89 MIS35N\SM8761

zation. As noted earlier, spaghetti code within an ad hoc structure often plagues development based on iterative refinement. Build V allowed some of these deficiencies to be corrected. FIESTA development is currently completing Build VI, integrating the workstation into the operational environment. This stage is discussed in the section which follows.

2.2.2 Work in Progress (On-Line Integration)

The FIESTA expert system task is currently well into Build VI, on-line integration. FIESTA is being installed as an on-line testbed to support final assessment of the concept and application of expert system technology. Despite the testbed designation, the mission-critical nature of the NCC environment remains applicable to the FIESTA implementation.

Major technical elements of the on-line integration include:

- Formal specification of functions for the On-line Testbed
- Allocation of functions to available processors
- Interface/access to the live data (LAN and CCS development)
- Modifications to the expert system component to account for "actual vs emulation" differences.

Supporting documentation that was recognized as important for the on-line operation included:

- Implementation Plan
- Transition Plan (from stand-alone prototype to on-line testbed)
- Functional Requirements Document
- Systems Requirements Document
  - Conventional format for interfaces and algorithmic processing
  - Modified format for expert system component
- Systems Design Document
  - Same distinction as for Systems Requirements
- Users Manual/Operational Scenarios
- Operations Concept for the On-line Testbed
- Evaluation Criteria.

Configuration management was also identified as a necessary element of deployment in an operational environment.

While the above items are also associated with conventional software engineering efforts, the application of expert system technology introduced the need for adaptation. In the following sections, the manner in which FIESTA methodology was accommodated are presented.

3.0 TESTBED DEVELOPMENT

3.1 FIESTA DEVELOPMENT STANDARDS

Usually, prototyping is associated with a "quick and dirty" software development approach. This approach was used for the off-line prototype. Operational software for the NCC is typically developed using formal standards, since the NCC must reliably operate in real-time, 24 hours a day. For the FIESTA on-line prototype, a balance has been struck between the formal and "quick and dirty" approaches. This "rapid software development" approach is tailored to meet the productivity goals of FIESTA, without compromising the NCC's real-time reliability. FIESTA documentation is pared down to technical details, eliminating boilerplate and redundant information. Reviews and walkthroughs are held by small working groups of technical personnel. Formal testing is limited to interfaces between the Symbolics and the CCS computer (not the expert system's knowledge base or rules). Three different levels of software development formality exist, based upon the level of risk associated with each software area.

Modifications to existing CCS software are done using the formal NCC development and Configuration Management (CM) standards. This approach is needed since CCS software is critical to the NCC's real-time operations.

New CCS software and Symbolics LAN Interface development use modified unit code,

prolog, unit testing, and development certification standards. This software is of low risk since it can be deactivated, but retains some formality because it interfaces with the NCC LAN and/or resides on a critical real-time computer. Standard NCC CM techniques are employed in this area.

The FIESTA expert system has been developed by iterative refinement and demonstration. Minimal formality has been imposed on the expert system since it can simply be disconnected from the LAN should problems occur. However, system requirements and design have been documented using formats developed and tailored for the FIESTA expert system. A CM approach appropriate for the expert system component is also being developed for the Symbolics software.

3.2 DESIGN DOCUMENTATION

Even though there was no explicit design stage in the development of the FIESTA expert system component, we decided to document the design of the prototype [6]. By documenting the design we can facilitate its maintenance. We also found that in the effort to document the design, we were able to clean up the system, thus restoring most of the integrity which had been lost by numerous reworkings. The methods we used to document the design helped identify areas that needed to be cleaned up, and facilitated the restoration.

3.2.1 Structure Charts

When working with a knowledge base the size of FIESTA (600+ rules), the easiest thing to lose track of is the rule interactions. The rules most likely to interact in FIESTA are those which are related functionally. Therefore, we tried to keep functionally related rules together in the source code. However, with several developers working on the system over several years and many modifications to the system, it was inevitable that these groupings became less and less cohesive over time.

We used structure charts showing the functional hierarchy of FIESTA to help restore these groupings. In the process of shuffling rules, we were able to identify redundant rules, and rules which could be generalized to cover a higher-level function. Therefore, we were able to streamline the number of rules. In addition, we used codes to indicate which lower-level functions represented individual rules, and which higher-level functions represented different types of files of source code, and placed these indicators in the appropriate structure chart boxes. As a result, the structure charts can be used as a table of contents to the source code, identifying which functions are performed, and exactly which rules perform them. Therefore, the structure charts can be a valuable tool for future maintenance.

3.2.2 Rule-Relation Lists

We determined that a utility built into the expert system building tool (we used ART) provided an excellent means of keeping track of rule interactions. ART keeps a list of all rules, and another list of fact relations (generic fact formats). These lists are cross-referenced, so that the developer can view all of the rules which either reference a relation (using that relation as a pattern on their left-hand side), or assert facts defined by that relation into the knowledge base on the rule's right-hand side. These lists are useful in many ways, and can serve some of the same functions as data flow diagrams.

First they identify rules which could potentially impact one another. If a modification is made to a relation in one rule, the rule-relation list can be referenced to identify which other rules may be affected by that change. Secondly, they can be used to help prevent superfluous facts from being asserted into the knowledge base. If there is a relation which has no rules which reference it (use it on the rule's left-hand side), then all of the rules which assert facts defined by that relation are adding information to the knowledge base which is never used. Likewise, useless rules can be identified by examining relations which have no rules to assert facts of that type. If these facts are not asserted at system initialization, all of the rules which reference that relation will never fire, because no facts of that type are ever asserted.

3.2.3 Viewpoint Structure

The third most critical tool we used to document the expert system design was a mapping of relations to viewpoints. Viewpoints are ART-provided utilities which allow the developer to partition the knowledge base into related areas. See the design document [6] for a discussion of the viewpoint network used in FIESTA. Each relation was defined to be used at different levels of the viewpoint network. Asserting or retracting a fact from the wrong viewpoint level could have major and unexpected effects on the program's execution. This documentation most closely parallels the documentation of a database design.

Although we were unable to apply standard waterfall model methods of design documentation to FIESTA, we were able to establish means to document the design which served many of the same purposes. This documentation should aid in future maintenance efforts and help to retain the integrity of the system.

3.3 CONFIGURATION MANAGEMENT

Configuration Management (CM) has been employed in an informal manner throughout the FIESTA Life cycle. Directories are associated with each of the Builds (and sub-Builds), and changes within files have been documented with comments in the preface describing the modifications and when thye were made, as well as identifying the responsible individual(s). This approach will be further formalized for operational use to assure that the following key requirements of CM are satisfied:

- Identification and availability of all source corresponding to the operational system
- Clearly defined procedures for introducing and tracking changes in the system (i.e., a complete audit trail)
- The ability to revert to an earlier version of the system.

Furthermore, CM will operate in conjunction with Quality Assurance Procedures (modified for rapid software development) to maintain a viable, dependable product. Central to the CM approach is the logging of all update procedures and the use of existing utilities (on both the CCS development machine and the Symbolics) to identify and archive files.

SUMMARY

FIESTA development can be reasonably described as a process of iterative refinement. Use of the Waterfall Model was precluded by the very factor (i.e., ambiguous requirements associated with the automation of expertise) which supported the use of expert system technology. Nonetheless, many of the features, milestones, and particularly documents which characterize the Waterfall Model are important for the effective development, implementation and maintenance of a computer-based system. Generation of such documentation and the specification of milestones similar to those found in the Waterfall Model have characterized the transition of FIESTA from prototype to operations. Even though adaptation has been required to accommodate expert system technology, the intent of the documentation and the various reviews corresponds closely to that associated with conventional software systems.

ACKNOWLEDGMENTS

The FIESTA development described in this paper was performed by Stanford Telecommunications, Inc. as a subcontractor to Computer Sciences Corp. (CSC) under Contract NAS5-31500. NASA and CSC personnel are also providing expertise required for the CCS and LAN modifications mentioned in the paper. NASA and CSC responsibiliies and participation involves overall CCS software development; their technical involvement in the integration of FIESTA in the NCC has been essential.

The ongoing encouragement and support of Paul Ondrus, Anthony Maione and Dawn Lowe of NASA have been appreciated and are acknowledged. Roger Werking (CSC-NCC Program Manager) and his support and technical personnel also deserve our thanks.

REFERENCES

1. Hayes-Roth, F., Waterman, D.S., and Lenat, D.N., Building Expert Systems, Addison-Wesley, Reading, Mass., 1983.

2. Soloway, E., Bachant, J., and Jensen, K., "Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-Base", Proceedings AAAI-87 Sixth National Conference on Artificial Intelligence, Seattle, Washington, 1987.

3. Boehm, B., "A Spiral Model of Software Development and Enhancement", Computer, May, 1988.

4. Wilkinson, W., Happell, N., Miksell, S., Quillin, R., Carlisle, C., "Achieving Real-Time Performance in FIESTA", Proceedings of the 1988 Conference on Space Application of Artificial Intelligence, NASA CP 3009, May 24, 1988.

5. Happell, N., Miksell, S., "Domain Expansion (with Maintenance Implications) of a Diagnostic Expert System: The FIESTA Example", In Preparation, Stanford Telecommunications, Inc., MIS845, 1989.

6. "FIESTA On-line Testbed Design - Volume II: Expert System Design", Stanford Telecommunications, Inc. Technical Report, TR880135, Oct. 1988.

FIESTA ON-LINE TESTBED DOCUMENTATION

"Functional Requirements for the Fault Isolation Expert System for TDRSS Applications (FIESTA) On-Line Testbed", TR870114, Revision B, 25 October 1988.

"Fault Isolation Expert System for TDRSS Application (FIESTA) Front End Process Requirements", TR870114.

"FIESTA On-Line Testbed Design-Volume II: Expert Sytsem Design," Stanford Telecommunications, Inc. TR880135, October 1988.

"User Manual for the Fault Isolation Expert System for TDRSS Applications (FIESTA) Testbed", MAN38ATP, 30 January 1989.

"Evaluation of the Fault Isolation Expert System for TDRSS Applications (FIESTA) On-Line Testbed", TR880137, 7 November 1988.

"Fault Isolation Expert System for TDRSS Applications (FIESTA) Implementation Plan", FIESTA Task Planning Document (C. Carlisle/532), 30 September, 1988.