Ground Systems Development Environment (GSDE) Interface Requirements Analysis Final Report

N-62-TM

p-106

Victor E. Church
John Philips
Ray Hartenstein
Mitchell Bassman
Leslie Ruskin
Computer Sciences Corporation

Alfredo Perez-Davila
University of Houston-Clear Lake

June 1991

Cooperative Agreement NCC 9-16
Research Activity No. SE.34

NASA Johnson Space Center
Mission Operations Directorate
Space Station Ground Systems Division

ricis

Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information systems in 1986 to encourage NASA Johnson Space Center and local industry to actively support research in the computing and information sciences. As part of this endeavor, UH-Clear Lake proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a three-year cooperative agreement with UH-Clear Lake beginning in May, 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The mission of RICIS is to conduct, coordinate and disseminate research on computing and information systems among researchers, sponsors and users from UH-Clear Lake, NASA/JSC, and other research organizations. Within UH-Clear Lake, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business, Education, Human Sciences and Humanities, and Natural and Applied Sciences.

Other research organizations are involved via the "gateway" concept. UH-Clear Lake establishes relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research.

A major role of RICIS is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. Working jointly with NASA/JSC, RICIS advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research, and integrates technical results into the cooperative goals of UH-Clear Lake and NASA/JSC.

Ground Systems Development Environment (GSDE) Interface Requirements Analysis Final Report

Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Computer Sciences Corporation in cooperation with the University of Houston-Clear Lake. The members of the research team were: Victor E. Church, John Philips, Ray Hartenstein, Mitchell Bassman and Leslie Ruskin from CSC and Alfredo Perez-Davila from UHCL. Mr. Robert E. Coady was CSC program manager for this project during the initial phase. Later, Mr. Ray Hartenstein assumed the role of CSC program manager. Dr. Perez-Davila also served as RICIS research coordinator.

Funding has been provided by the Mission Operations Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Thomas G. Price of the ADPE and Support Systems Office, Space Station Ground Systems Division, Mission Operations Directorate, NASA/JSC.

CSC/TM-91/6102

Ground Systems Development Environment (GSDE) Interface Requirements Analysis Final Report

Prepared for

The University of Houston-Clear Lake
Houston, Texas

by

Computer Sciences Corporation
System Sciences Division
Beltsville, Maryland
and
Systems Engineering Division
Falls Church, Virginia

under

Subcontract No. 075
RICIS Research Activity No. SE-34
NASA Cooperative Agreement NCC 9-16

June 1991

Prepared by:

Victor E Church 6/20/91

V. Church Date

M. Bassman
J. Philips
L. Ruskin

Quality Assurance:

[illegible] 6/20/91

D. Long Date

Approved by:

[illegible] 6/20/91

R. Hartenstein Date

Abstract

This report presents a set of procedural and functional requirements for the interface between software development environments and software integration and test systems used for space station ground systems software. The requirements focus on the need for centralized configuration management of software as it is transitioned from development to formal, target-based testing. This report is concerned with application and presentation level interface questions, and does not address physical interface issues.

This report concludes the GSDE Interface Requirements study. It builds on earlier reports of the study and provides a summary of findings concerning the interface itself, possible interface and prototyping directions for further investigation, and results of CSC's investigation of the Cronus distributed applications environment.

Table of Contents

List of Figures and Tables

Section 1 - Introduction

As part of the Space Station Freedom Program (SSFP), the Mission Operations Directorate (MOD) at the Johnson Space Center (JSC) is developing a Space Station Training Facility (SSTF) and a Space Station Control Center (SSCC). The software components of these systems will be developed in a collection of computer systems called the Ground Systems Development Environment (GSDE). The GSDE will make use of tools and procedures developed by the SSFP Software Support Environment (SSE) contractor. Both the SSTF and the SSCC will be developed using elements of the GSDE.

Configuration management (CM) of SSTF and SSCC software will be performed using the government-furnished SSE CM tools residing on the GSDE Amdahl computer in building 46 at JSC. The GSDE Amdahl currently serves as the Ground Systems/Software Production Facility (GS/SPF) host computer. The Mission Systems Contract (MSC) contractor and the Training Systems Contract (TSC) contractor will make use of these tools from the start of formal testing onward. The GSDE CM system will be used to store and manage source code, documentation, objects, executable images, and software test resources for both the SSTF and the SSCC. The GS/SPF host may also be used to store object code and load images of operational software following qualification testing.

At JSC's request, the Research Institute for Computing and Information Systems (RICIS) at the University of Houston-Clear Lake (UHCL) has performed an analysis of the interface between the GSDE and the SSCC and SSTF Integration, Verification, and Test (IV&T) systems. The study was performed in cooperation with RICIS by Computer Sciences Corporation (CSC) under subcontract to UHCL.

This is the final report on the GSDE Interface study; it documents the requirements definition phase of the study. The interfaces of concern are those between the software *production* environments and the software *integration, validation, and test* (IV&T) systems. The requirements defined address the configuration management of software as it is moved back and forth between the two environments, data collection across the interface for test activity recording, and the operational aspects of file and information transfer over the interface.

The two ground system contractors provided detailed and responsive comments to CSC's presentations and working papers. Personnel from both MSC and TSC were cooperative and open in discussing plans for software development. MSC personnel in particular provided detailed comments and information on their plans and on our analyses. Software configuration management is regarded as an important element of the total quality management effort at NASA; the level of information provided to us makes it clear that though their approaches differ both contractors are very serious about CM.

1.1 Purpose and Scope

This final report presents the CM and functional interface requirements developed in this study. It is based on CSC's analysis, and on extensive discussions with RICIS and JSC personnel and with personnel from the SSE, SSCC, and SSTF development teams. The report presents specific requirements based on those discussions, critically analyzes the procedures proposed by the MSC and TSC contractors, and includes recommendations developed during the study that may be of value during implementation of the requirements.

The scope of this report includes the definition of requirements for functionality in the interface between ground system software production environments (SPEs) and IV&T systems. It also includes requirements for procedures and data transfer needed to support centralized configuration management of ground system software during and after formal testing, and provides information on the context and analysis of these requirements in order to facilitate their implementation.

1.2 Scope and Organization

Following this introductory section, Section 2 describes the context of the requirements defined in the report. It identifies the high-level CM requirements that we derived and used as a basis for the detailed analysis; the formal statement of these high-level requirements is an assumption on CSC's part based on direction from NASA. Section 2 also provides an overview of the computer systems and operations planned for the SSCC and SSTF developments, based on current information.

Sections 3 and 4 present the GSDE interface requirements developed in this study. Section 3 presents the requirements for configuration management of controlled software that crosses the GSDE-to-IV&T interface. Section 4 presents the requirements for functional support of operations and CM which the interface must provide. Each section presents an overview of the interface from the appropriate perspective, and provides the requirements statements and explanatory material. Requirements for CM policies and procedures are related to the high-level requirements stated in Section 2. Functional interface requirements were derived from TSC and MSC responses to the operations scenarios developed during this study (documented in CSC/TM-91/6061), and from the CM requirements in Section 3.

Section 5 summarizes supporting and related information included as appendices to this report. These appendices include information provided by different ground system software development contractors together with analyses used in defining the requirements in sections 3 and 4.

Appendix A lists the configuration management information fields required in the GSDE CM system that are relevant to this interface. Appendix B describes the CM and operational interface procedures planned by the MSC and TSC contractors, and is based primarily on material provided by those contractors. Appendix C reports on analysis of the proposed SSCC and SSTF information flows in the context of the requirements described in Sections 3 and 4. Appendix C also presents information developed during the interface analysis that may be of value in implementing the requirements.

Appendix D consists of the detailed responses from the TSC contractor, the Flight Simulation Division of CAE-Link Corporation, concerning the interface operations scenarios in CSC/TM-91/6061. Appendix E consists of the software development scenarios developed by the MSC contractor, the Space Information Systems division of Loral Corporation.

Appendix F consists of a report on the findings of the study concerning the applicability and implications of using the Cronus distributed applications environment.

1.3 Statement of the Problem

The Mission Operations Directorate at JSC is responsible for the development of ground support computer systems, the SSTF and the SSCC, for the Space Station Freedom Program. The software in these systems is being developed in the GSDE. Within the GSDE, the GS/SPF provides resources developed as part of the SSFP SSE. The GS/SPF includes a host computer (currently an Amdahl mainframe), several Rational R1000 Ada development computers, and a local area network (LAN) with various workstations (Unix-based, MS DOS-compatible, and Apple Macintosh, at a minimum) and some special-purpose devices attached.

The *target* environments for this ground system software will be composed of computers, workstations, and special-purpose devices that are specific to the operational purposes of the two systems. Software will be developed in software production environments and transferred to the target for integration and system testing.

The integration, verification, and testing of SSCC and SSTF software will be performed on target computers which closely resemble the operational environment. Some of the IV&T computer systems, in fact, may become operational systems. The interface between development and target computers, and the need for configuration and change management in these distributed systems, pose new challenges to the software development process.

In order to address these challenges, which involve both the SSTF and the SSCC, the MOD has requested RICIS to perform this interface study. Neither the SSCC developer nor the SSTF developer is specifically tasked to perform overall process integration for the GSDE, or for the GSDE CM process, and the study is complementary to software environment research being conducted at UHCL. This study has analyzed plans for both ground systems and the development environment they share.

CSC has investigated the interface and configuration management aspects of the GSDE-to-IV&T interface. The study included identifying and documenting interface requirements, reviewing the software development and configuration management plans for the SSCC and the SSTF (in the context of this interface), proposing operations scenarios of CM procedures, and developing requirements for the interface. Information and comments were received from both the MSC and TSC contractors at several points during the study. The preliminary analysis and the operations scenarios were documented in previous reports, identified in the list of related documents and references.

This study task focuses on the *interfaces* between the development and IV&T environments. Those interfaces include communications between SPEs and IV&T systems, transfer of files and command scripts, and reports on formal testing performed on the target. The goal of this effort is to define interface requirements that prescribe support for configuration management and test documentation.

1.4 Related Documents and References

BBN Systems and Technologies, Cronus Release 2.0, March 1991 (preview notice)

CAE-Link/Flight Simulation Division, Space Station Training Facility/Ground Software Development Environment (GSDE) Usage Concepts, 17 August 1990 (briefing)

Computer Sciences Corporation, *Digital Systems Development Methodology*, May 1990

Computer Sciences Corporation, *Ground Systems Development Environment (GSDE) Interface Requirements Analysis: Operations Scenarios*, CSC/TM-91/6061, February, 1991

Computer Sciences Corporation, *Ground Systems Development Environment (GSDE) Interface Requirements and Prototyping Plan*, CSC/TR-90/6155, March, 1991

Federal Information Processing Standards Publication 151, *Portable Operating System Interface Standards (POSIX)*

Johnson Space Center/T. Price, Ground Software Development Environment, April 1990 (briefing)

Johnson Space Center/S. Hinson, Ground Systems Development Environment, October 12, 1990 (briefing)

Lockheed Missiles and Space Company, *SSE Detailed Requirements Specification (DRLI 72)*, LMSC F255472, July 1989

Loral/Space Information Systems/D. Sundermeyer, Space Stations Control Center/System Functional Design review/Ground Systems Development Environment, 15 November 1990 (briefing)

McKay, C., "Portable Common Execution Environment (PCEE)", UHCL Report

NASA/SSE System Project, CM system for OI-5, 16 August 1990 (briefing)

NASA/SSE System Project/C. Michaels, OI 6.0 DRR Concepts/Configuration Management, 18 October 1990 (briefing)

NASA/Software Management and Assurance Program (SMAP), *Software Assurance Guidebook*, SMAP-GB-A201, September 1989

Vintner, S., "Integrated Distributed Computing using Heterogeneous Systems", *Signal*, vol. 43:10, June 1989 (overview of Cronus)

Section 2 - GSDE Interface Overview

The focus of this report is the interface between the GS/SPF and the IV&T systems used in development of SSCC and SSTF software. Specifically, this report addresses the interface requirements for the transfer of files and communication of status that is needed to support configuration management. The interface requirements must be understood in the context of the overall requirements for formal CM which exist independent of the GS/SPF to IV&T interface. Section 2.1 discusses the high-level requirements for configuration management of ground system software.

In addition to the procedural and information requirements of CM, there are practical concerns of how information is transferred between systems. Section 2.2 describes the physical and logical environments for the development, integration, verification, and test of the SSTF and SSCC software.

2.1 High-level Configuration Management Requirements

This section describes the high-level requirements for formal CM that formed the basis for this study. For purposes of analysis, CSC restated the direction provided by NASA in terms of specific requirements. These high-level CM requirement specifications are presented as the basis for the detailed requirements in Section 3. They are included here to establish context, and should not be interpreted as actual statements of requirement placed on the SSTF and SSCC developers. Rather, these are formal statements of the informal direction provided by JSC concerning the CM of ground systems software.

The interface requirements presented in this report are based on the asssumption that requirements for formal CM, similar to those described in this section, will be levied on the MSC and TSC at some point.

2.1.1 SSE-defined CM Capabilities

The SSE has requirements to provide tools that will support configuration management of space station software. The requirements are defined in the Space Station SSE document *SSE Requirements Specification* (DRLI 72), as amended by various formal Change Requests (CRs). The tools and support capabilities are described in various briefing documents and materials that characterize the CM capabilities to be provided with each operational increment (OI), specifically OI 5.0 and OI 6.0.

These requirements and support capability descriptions form the basis for the detailed requirements for formal CM that are presented in this report.

PRECEDING PAGE BLANK NOT FILMED

In greatly simplified terms, the SSE-provided CM capabilities are supported on an Oracle relational database management system (on DEC or IBM mainframes) with application software to perform data entry, modification, verification, and reporting. These applications are part of the SSE CM toolset, and are integrated with the Oracle database.

The database consists of data structures and relationships, allowable values and ranges, and procedural constraints to insure data integrity. The database includes both pre-defined and user-definable elements and attributes. To support CM, the database includes elements to support configuration identification, traceability between versions and related elements, and relationships between elements. It also supports activity logging (e.g., tests performed) on configuration items, and both standard and customized reporting capabilities. The CM system provides security measures to insure that no changes are made to configuration items in the absence of valid change instruments auhtorizing the changes.

A brief description of CM-related fields in the SSE database is included as Appendix A.

The SSE-provided CM support also includes an interface to the Rational Ada Development Facility (ADF), to make use of the Rational-based Code Management and Version Control (CMVC) system for development support. Developers can use the Rational ADF for creation and editing of source code. Following acceptance testing at the component level, new code can be exported from the Rational and promoted to the mainframe CM database. Changed code that has been checked out and modified can likewise be uploaded back to the mainframe CM database, where (with appropriate authorization) it is checked back in to the CM system. In addition, the Rational ADF subsystem archive capability (which stores entire subsystems instead of independent components) can be used in conjunction with the mainframe CM system.

A Rational ADF subsystem can be stored as an "archive" file and uploaded to file storage on a mainframe. However, outside the Rational the archive file is not able to be interpreted or modified; the internal structure is specific to the Rational, and cannot readily be processed by other computer systems. As a consequence, the mainframe CM system cannot directly examine the components of a subsystem and determine which, if any, have been modified. This agglomeration of components into a monolithic subsystem archive file presents an obstacle to detailed configuration control. Figure 2-1 shows this process.

The solution, developed on the Rational by the SSE System Project, is to accompany the exported archive file with extracted text of components in the subsystem. The archive file and the text files are transferred to a "landing area" on the mainframe using standard SSE file transfer mechanisms. Upon subsequent checkin to the CM system, these text files are compared with controlled versions in the mainframe CM database. The text of unchanged components is discarded. For changed components, a change instrument must be on file to approve the change. The text of approved changed components is promoted into the CM database, and the entire subsystem file is stored as a single controlled object.

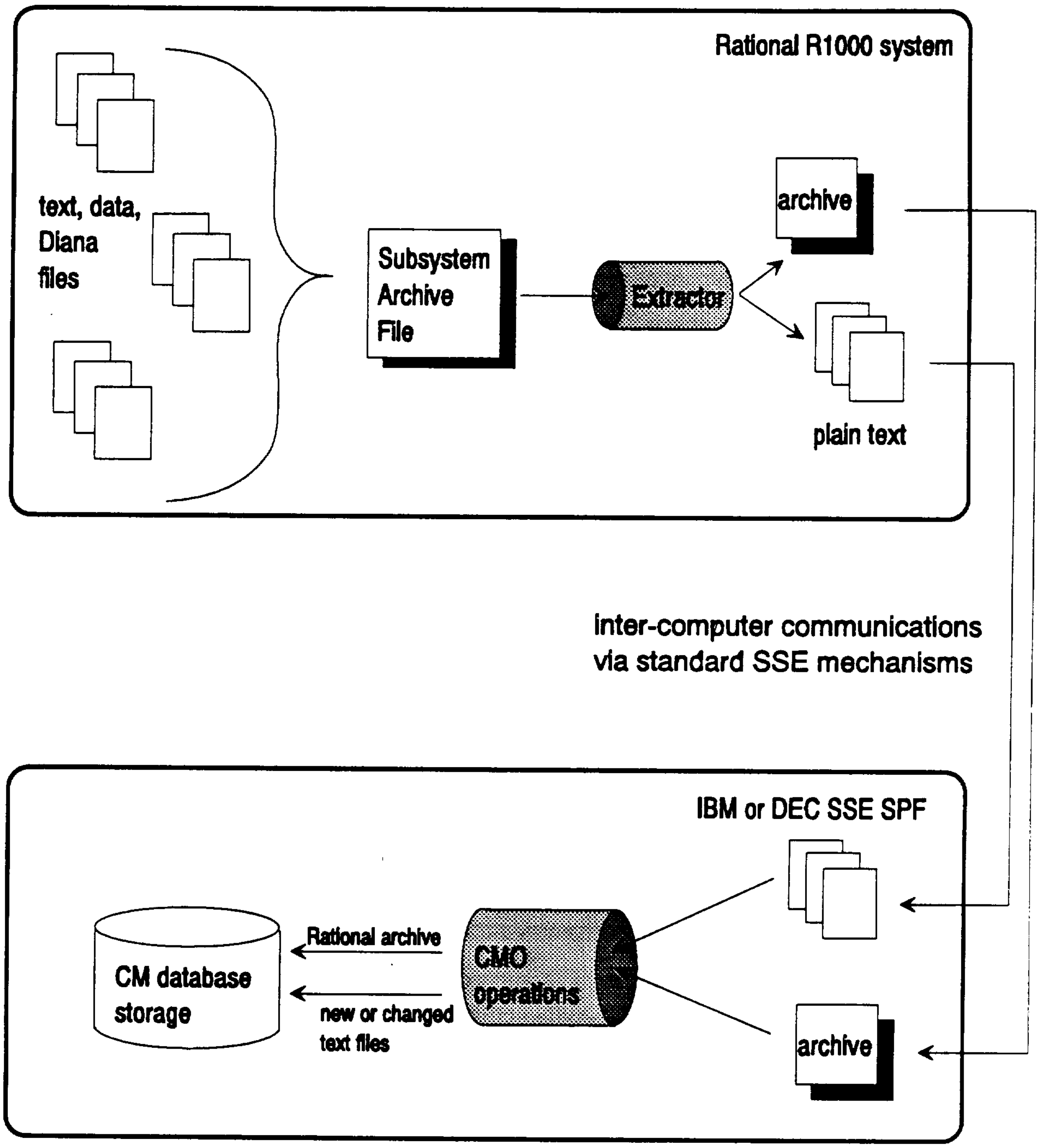


Figure 2-1 Rational-to-SPF CM Interface

When a subsystem is checked out and transferred back to the Rational ADF, the source text is uploaded from the GS/SPF along with the subsystem archive file. If the source code still resides on the Rational and has not been changed, there is no need to expand the archive file to restore the subsystem.

Some of the details of this process will change with the release of OI 6.0, but the basic capability is available with OI 5.0

2.1.2 High-level Requirements for Formal CM

The assumed requirements for formal CM of SSFP ground system software are presented below. The specifications of the requirements are shown in sans-serif text. The descriptions and explanations that follow each requirement in standard text are for information only, and are not part of the formal requirements.

1. Formal configuration management (CM) of SSTF and SSCC software will make use of the SSE-provided CM object storage, database, tools, and reporting capabilities on the GS/SPF host computer system.

   This assumption says that the GS/SPF will be the repository of CM information and controlled software, and that the SSE tools will be used. It anticipates the use of NASA-supplied standard tools instead of (or in addition to) any contractor-defined tools.

   The traditional requirements of CM (e.g., "record all changes") are embodied in the SSE CM system. By specifying the use of the SSE-provided system, the specific requirements are incorporated without excessive detail.

   The term *formal CM* is used to distinguish the CM process supported on the GS/SPF host and operated by NASA from the CM systems which the SSTF and SSCC developers will use for source code control during development. This usage is consistent with the difference between formal testing, which leads to acceptance of a CI, and informal testing, which is part of the development process.

2. All software which is required to be placed under formal CM will be submitted to the formal GSDE CM system prior to any level of formal testing in the target environment.

   For purposes of quality control it is essential that software be controlled and monitored during the integration and testing process. This specification ensures that all formal testing on target systems is performed on controlled software. This does not preclude formal acceptance testing of software in the SPE prior to submission to the GSDE CM system.

3. All deliveries of software to operations will be made from controlled file storage (i.e., from files that are under formal CM), or from compiled

object code that has been directly generated, using controlled process scripts, from such controlled storage.

This specification assumes that complete traceability and accountability must exist between delivered operational software and software components that are under formal CM. It expects that either the compiled objects themselves will be placed under formal control, or that controlled source code will be used in a script-controlled process to generate object code for the purpose of a specific delivery.

The reason for the requirement to use controlled process scripts is to ensure that no deliberate or inadvertant changes are made (for example, changing the optimization level on a compiler between certification and delivery) that would make the *delivered* code different from the *tested* code.

4. After a component has been placed under formal CM: all operations on, tests of, and/or changes to that component will be documented in the formal CM system.

This specification assumes that once a component enters the formal CM system, *everything* that is done with it is recorded. It is intended to ensure that there is complete traceability of the history and use of any component, and that all operations are visible to NASA as well as to the developer.

If there is a need for less formal testing and redevelopment, the components can be checked out of formal CM back to the SPE.

5. The information that is provided to document any operations on, tests of, or changes to a component will include all information about those operations, tests, or changes that are reported by the standard SSE-defined CM reports of configuration item identification and history.

This assumption ties the reporting of component activity to the data fields defined in the standard reports. This tie-in requires (for example) information from the tools used to perform operations, the user IDs of operators, and authorizations for any changes.

Appendix A provides a description of CM fields which are relevant to the CM processes addressed in this study (i.e., those involving the GS/SPF-IV&T interface).

2.1.3 Discussion of High-Level Requirements

The primary purpose of the formal CM system is to ensure the integrity and maintainability of operational software. A major secondary purpose is to accumulate information on the development process to support metrics-based process improvement. These purposes can be met by the requirements specificed in section 2.1.2.

The integrity and maintainability of operational code is ensured by documenting the following two relationships:

- Controlled source code satisfies all relevant requirements, as demonstrated by formal testing of the software.
- Operational software is derived from controlled source code with no changes to the code or to the procedures used in performing formal testing.

Requirements 1, 2, and 3 provide the mechanism for documenting a traceable history from requirements through source code to delevered, operational software.

The accumulation of process data is supported by requirements 1, 4, and 5. By requiring a complete accounting of integration and test activities, these requirements ensure that the collected data provides an accurate representation of the process.

2.2 General CM Operations Description

This section describes, in general terms, the CM system that will be provided by the SSE as it pertains to supporting configuration management for the integration and testing of software. This section consists of several sub-sections describing the different aspects of the CM system.

The CM system provided for the GS/SPF host by the SSE is an Oracle-based system of tools and a tracking data base. This system has some (but not all) interfaces needed to collect data throughout the life cycle of any software. In addition, the system has tools that allow the developer or tester to construct information reports that will be used throughout the software life cycle.

The SSE CM system is based on the principles of **configuration identification, control, status accounting**, and **traceability** (auditability). There are specific definitions of what objects (hardware and software) are considered to be *configuration items (CIs)*: these include software at every life-cycle stage, scripts used to process software, test data, change instruments, and status reports. Control involves requiring that any change to a CI be authorized by a recorded *change instrument* (e.g., an approved Change Request), and performed by an authorized user. Status accounting ensures that all activities affecting the status of a CI (e.g., passing or failing a formal test) are recorded in the records that describe the CI. Traceability is established by maintaining relationships among related and derivative CIs, such as sequential versions of source code.

The SSE CM system provides rigorous checking of transactions affecting the CM database, thus assuring the internal consistency of the data. However, procedures for use of the SSE CM system are determined independently for each SPE. Policy questions such as what items are placed under SSE CM and when, who is authorized to approve changes, and which descriptive fields must be filled in, are addressed separately for SSTF and SSCC. The SPE-level implementation can also develop user-specified fields, values, forms, and reports that extend the SSE-provided set. The CM system provided by the SSE must be implemented and tailored to the specific requirements of each installation.

The following sections provide a brief description of the capabilities provided by the SSE-CM system.

2.2.1 Configuration Management Fields

Configuration management fields are those information fields that make up part of the SSE CM Oracle database. The fields described in Appendix A are those that are relevant to the CM processes addressed in this study. This compilation does not represent all the CM fields that are available in the SSE system. Many more fields are defined by the SSE, and are listed in the tables that include the CM tools and the CM reports fields, in the *Operations Scenarios* report (CSC/TM-91/6061).

From the perspective of software CM, the supported data fields can be classed as CI definition and description, CI activity records, and CI change records. The first set describes the CI and its relationships to other CIs, including other versions of the same software item. The second class records actions that change the CI status, but not the CI itself. Testing is the primary activity that generates this class of information. The third class records actions that change the software item, perhaps generating a new version (a new CI) of the item.

2.2.2 Configuration Management Reports

The SSE CM Oracle data base includes the capability for authorized users to generate many pre-defined reports, and to create custom reports, from the information in the database. All these reports may be edited, modified, have information added or deleted, or logged. A full listing of the predefined reports and their contents is provided in Appendix A of the *Operations Scenarios* report. The reports are summarized below.

A basic report for describing a CI is the Configuration Version Description Report. This report includes the description of the item, its status, and its relationship to other items in the CM database. The report particularly notes version relationships; placing the subject CI in context with other versions of the same article.

There are a variety of test-related reports. Several reports deal with the testbed (the software that together is used to test a particular CI) and test resources: the IT&V Current Test Resources report, the Testbed Build Report, and the Test Resource Status report. Test status is reported with the IT&V Test Status report, the Test Results report, the Non Conformance report, and the Non Conformance Closeout/Explanation report.

Change history for a CI is reported in the History of Configuration Item Changes report.

Test status for one or more CIs is reported in the reports Testing Status of Deliverable Components, Test History for a Component, Test Metric, and Configuration Items affected by a Non Conformance.

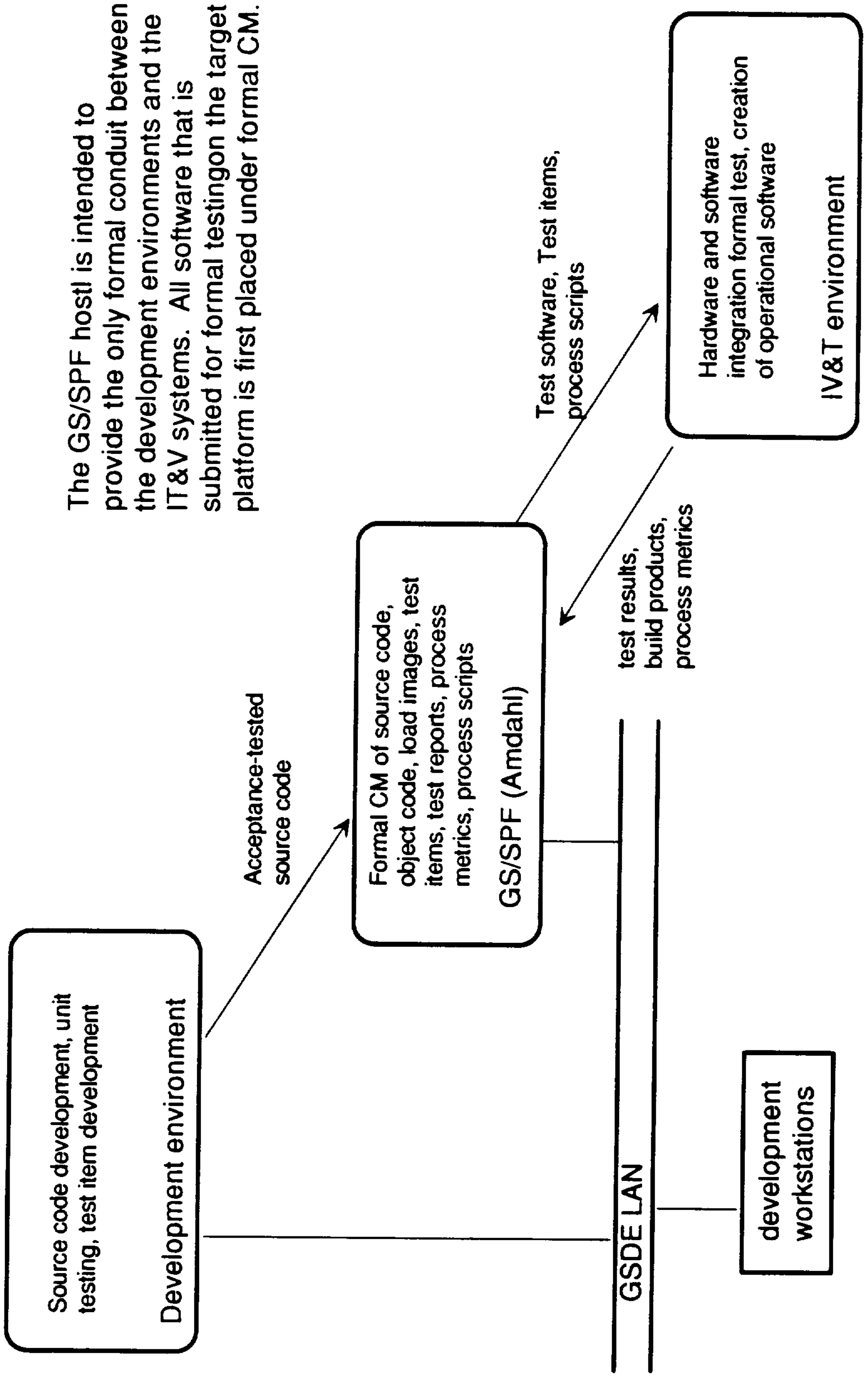
2.2.3 Configuration Management Tools

The SSE provides a number of Oracle-based tools designed to help manage configuration data. In particular, there are tools designed to support testbed definition and generation and test status reporting. The tools were developed specifically to support the development of space station flight software. The value of these tools outside of the flight software domain is not yet determined, but they provide a model for the types of tools and activities that the SSE CM system is intended to support. (Most of these tools will be available with OI 6.0).

There are also test support utilities designed to generate testbed creation scripts, testing scripts, and test resources. Other tools support the logging of information from testing, collecting all of the information needed for testing status reports. Still other tools can be used to generate reports or post test results.

2.3 GSDE Interface Architecture

Figure 2-2 shows the basic architecture of the GSDE interface to the development and testing facilites for both the SSTF and the SSCC. The architecture was developed to place the GSDE CM tool between the development and testing areas. This was done to direct all transactions through the GSDE CM in the GS/SPF host so that strict CM could be effected.


Source code development, unit testing, test item development
Development environment
Acceptance-tested source code
Formal CM of source code, object code, load images, test items, test reports, process metrics, process scripts
GS/SPF (Amdahl)
Test software, Test items, process scripts
Hardware and software integration formal test, creation of operational software
IV&T environment
test results, build products, process metrics
GSDE LAN
development workstations

The GS/SPF hostl is intended to provide the only formal conduit between the development environments and the IT&V systems. All software that is submitted for formal testingon the target platform is first placed under formal CM.

Figure 2-2 GSDE Interface Architecture

Development personnel interact with the GS/SPF host to perform configuration management of files, and of their attributes and relationships recorded in the CM system. Software configuration items (CIs) are uploaded to the GS/SPF host with appropriate processing instructions (command scripts). Processing (e.g., integration testing) occurs in the IV&T system, and may include interactions with IV&T personnel. There is no direct interactive (i.e., workstation-based) link from the IV&T system to the GS/SPF.

Products and status are returned to the Amdahl after processing. Products generated on the IV&T computers (e.g., object code) may be retained there for further use as well as being uploaded to the GS/SPF host.

Section 3 - Procedural Requirements for CM

This section provides a general description of the GSDE software configuration management context, and presents specific requirements in terms of policies and procedures necessary to achieve the intended level of configuration management.

3.1 CM Interface Overview

The fundamental interface issue in the GSDE is how to assure that all software operations in the IV&T environment are recorded in the GS/SPF host-based CM system. Figure 3-1 provides a schematic picture of the interface issue. Prior to any level of formal testing in the target environment, the source code of developed software is placed under formal configuration management using the SSE-provided CM system on the GS/SPF host. (This CM system is referred to in the requirements as the *formal GSDE CM system*. In the discussions which follow, it is abbreviated as the GS/SPF-based CM system, or *GCM*). These controlled software configuration items are tested in the IV&T using test data and procedures that are likewise controlled. Following successful testing, the software CIs are used, possibly linked with other data, to produce operational software.

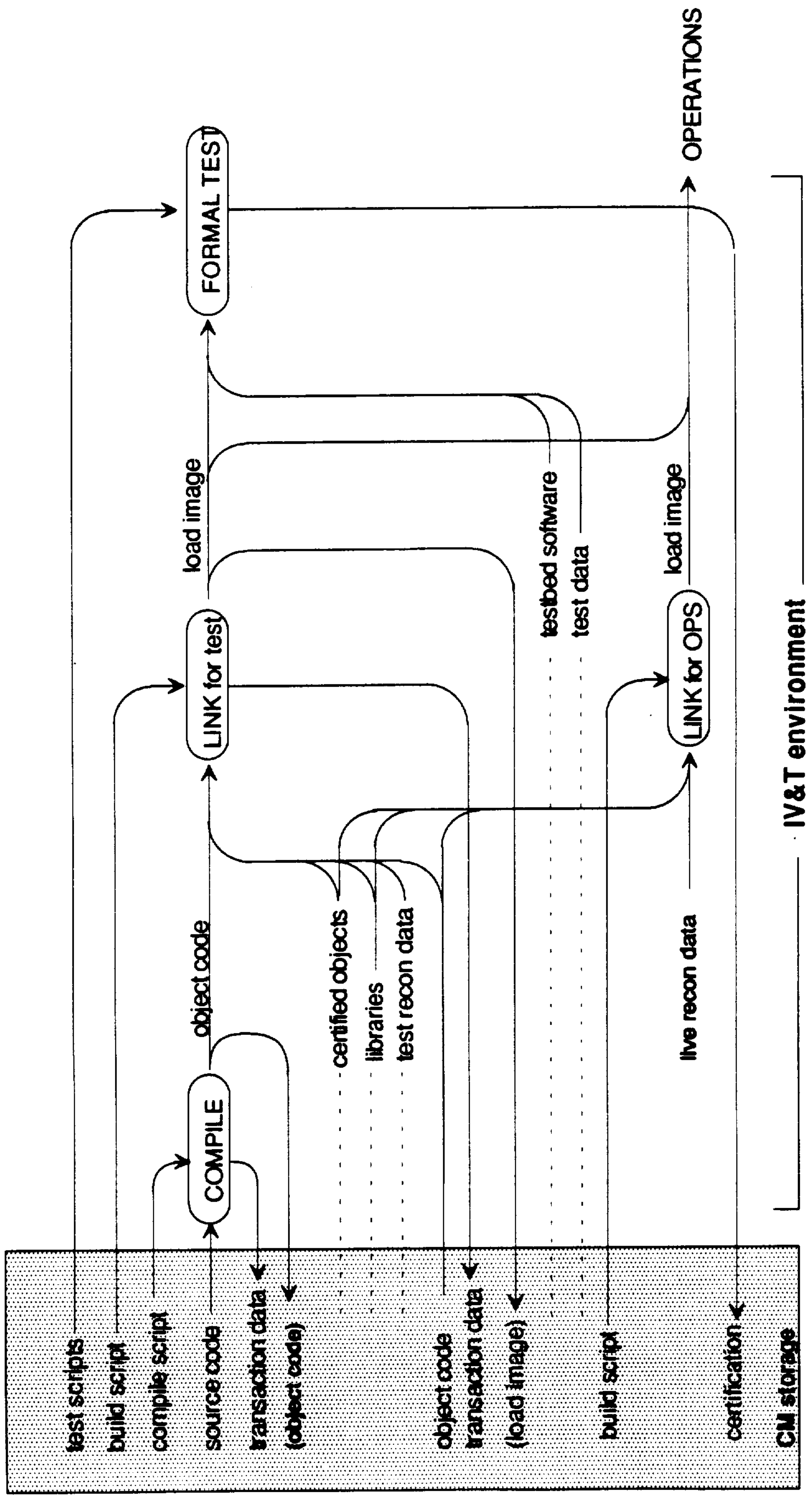
The role of CM and the GS/SPF-to-IV&T interface is to ensure that the source code, test cases, test scripts, test results, and delivered software are consistent and reproducible. Figure 3-1 traces the movement of source code and object files across the interface, and indicates the return of status information and test results. Figure 3-1 also shows the optional return of compiled and linked components to the GS/SPF host (the items in parentheses); this path is not required, though it does simplify the accountability process.

The figure also illustrates the role of stored, controlled scripts in performing operations on CIs. Such scripts provide the accountability that CM requires, by permitting any questionable product to be reproduced on demand. At the same time, these scripts facilitate development by providing a reusable resource in a controlled fashion.

The requirements statements are grouped in terms of compilation of source code, linking of object modules, and testing of executable images.

3.2 Compilation CM Requirements

The following requirements apply to the process of compiling a source code configuration item (CI) in the target environment. These requirements apply to compilation of source code which has been placed under formal GSDE CM.



With the exception of the live recon data used to generate loads for operations, all software, scripts, and data either comes directly from, or is directly traceable to, controlled CM storage.

Figure 3-1 Configuration Management Process Flow

CM-1. The object code generated by compilation of a source code CI in the IV&T environment shall be recorded in the formal GSDE CM system as an object code CI. If the object code itself is not copied to the GS/SPF host, the CM record for the object code CI shall indicate the location of the object code file. Exception: if the compilation process does not generate object code, or if the object code is discarded without being used, no CI record shall be established.

If the object code is immediately deleted, or if, by compiler directive or compilation error, no object code is produced, there is no need for a CI record to be established. There will be a record of the compilation, but no actual CI.

This may be the case when compilation testing is performed, or when an existing CI is recompiled with a new compiler to test the compiler.

CM-2. The process of compiling a source code CI in the IV&T environment shall be a script-driven process. The compilation script shall completely specify all parameters and conditions that can change the object code produced by the compilation process. Compilation scripts for source code CIs shall be placed under formal GSDE CM.

"Source code configuration item" means a single element, such as a source code file, that is treated as an entity. "Script-driven " implies that the process is not interactive, but is controlled by a stored sequence of commands and/or parameters. A script-driven process should be entirely repeatable. The intent of "completely specify...object code" is to ensure that all of the settings on the compilation system are consistent from one use to the next, so that identical object code will be produced regardless of any changes to default conditions. This does not prohibit changes in incidental parameters such as listing options.

The essence of this requirement is that the compile command sequences are subject to the same CM as the source files.

CM-3. When a source code CI is compiled in the IV&T environment, information characterizing the compilation shall be transmitted to the formal GSDE CM system. This information shall include the identification and version of the tool(s) used in the compilation, the User ID of the initiator, the platform used to perform the compilation, the outcome of the compilation, and the unambiguous description of the object code file.

The outcome of the compilation is the completion status reported by the compiler. The unambiguous file description includes the file name, size, and date of creation. This requirement addresses the need for recording the integration and test process. Informal, unrecorded compilations, if required, should be performed with development CIs.

The next two requirements specify where compilation information is to be filed in the CM database.

CM-4. If the object code produced by such a compilation is placed under formal GSDE CM (becoming a distinct CI), whether or not the object code is stored on the GS/SPF host, the compilation information shall become part of the CM records describing that object code CI.

If the object code is produced and not deleted, the object CI record will include the generation data.

CM-5. If the object code produced by such a compilation is not placed under formal GSDE CM (i.e., no CI is generated), the compilation information shall be recorded as a transaction record linked to the source code CI.

If object code is not produced or is deleted without being used, there is no need for generation data. However, the compilation record will be linked to the source code CI for process metrics analysis.

3.3 Object Linking CM

The next set of requirements applies to the creation of load images (executable programs) or object libraries, collectively termed *build products*, from controlled object code and other files.

CM-6. Any library or load image (build product) created in the IV&T environment from existing CIs shall be placed under formal GSDE CM. If the build product itself is not copied to the GS/SPF host, the CM record for the build product CI shall indicate the location of the build object file. Exception: if the build process does not generate any product, or if the product is discarded without being used, no CI record shall be established.

If the build process is performed to verify compatibliity of objects without generating a product, there is no need to define a configuration item record because there is no actual configuration item to be controlled. (Any output would have essentially the same transient status as a listing file).

If the build process generates an executable file which is then used in testing without being copied to the GS/SPF host, a CM record is required.

CM-7. The process of building a load image or library from object code, libraries, and test versions of data tables shall be a script-driven process. The build script shall completely specify all object code files or CIs, all libraries, all data tables, and all parameters and conditions that can change the build product generated by the build process. Build scripts for configuration items shall be placed under formal GSDE CM.

Most development systems routinely provide the capability to use scripts to ensure repeatability of operations. This requirement is primarily intended to ensure that those scripts are controlled.

CM-8. When a build product is generated in the IV&T environment from CIs, information characterizing the build process shall be transmitted to the formal GSDE CM system. This information shall include the identification and version of any tools used in the build, the User ID of the initiator, the platform used to perform the build, the outcome of the build process, and the unambiguous description of the build product file.

There can be many items that contribute to a build; these are defined in the build script. The build information record, together with the build script and the CI records of the elements that contribute to the build, should provide a complete definition of the build product.

CM-9. If the load image or library produced by the build process is placed under formal GSDE CM (becoming a distinct CI), even if the product(s) is not stored on the GS/SPF host, the build information shall become part of the CM records describing that build product CI.

CM-10. If the load image or library produced by such a build process is not placed under formal GSDE CM, the build information shall be recorded as a transaction record linked to the build script CI.

If a build product is not generated, or is deleted without being used, the build record will be linked to the build script CI for process metrics analysis.

3.4 Formal Testing CM

The next set of requirements apply to the formal testing of controlled software in the target environment.

CM-11. All products needed to perform formal testing of controlled software shall be placed under formal GSDE CM prior to testing in the target environment. This includes the software to be tested, any data files needed for the test, test versions of data tables used in building executable software, test setup command scripts, and test scripts for test operators.

For deliverable software, reconfiguration data will be provided to and managed by the reconfiguration data system. For testing purposes, however, the reconfiguration data must be placed under CM on the GS/SPF host so that the testing process is accountable and repeatable.

CM-12. Any tests performed on controlled software in the target environment shall be recorded in the formal GSDE CM system. The information recorded about the test shall include the configuration IDs of all CIs involved in the test, the User ID of the initiator, the identification of the specific test(s) performed, the status of each test or test step, and the status of each configuration item being tested.

It may be the case that software testbeds are created and then used for testing over an extended period. Software which is checked out for testing will not be checked out for a specific test, but for the process of testing. The first indication that a specific test was performed will be the test status report provided after the test is complete (with whatever status).

CM-13. Test reports recorded in the formal GSDE CM system shall be associated with the test performed and with the configuration ID of the build product CI that was tested.

The intent of this requirement is to ensure that test conditions and reports are consistently stored in the formal GSDE CM system.

Section 4 - Functional Interface Requirements

The functional interface between the GS/SPF host and an IV&T computer system consists of the tools and procedures used to transfer files and information between the two. The functional interface is built on the physical interface (the network interconnections) and embodies the high-level protocols used to ensure effective communication. In terms of the International Standards Organization (ISO) Open Systems Interconnect (OSI) model, the functional interface primarily involves the application, presentation, and session layers.

Six requirements categories were identified to capture the functional interface requirements. These categories describe broad functional areas of support, as listed below:

- file transfer and informational dialog support
- support for target-compatible command scripting
- support for file naming, location reporting, and verification
- tools for extracting and packaging CM information from products
- computer resource scheduling support
- remote login capability.

The more detailed requirements presented in each category provide a basis for analysis and development of actual support software. However, in many cases the design of the IV&T systems, the physical interfaces, and the operational procedures are not fully mature. In some areas there is not sufficient detail to justify requirements at the "design-to" level of specificaton. These areas are noted in the discussion below, and will require further definition as the GSDE evolves.

CSC recognizes that the implementation of these requirements entails resources that are subject to prioritization and scheduling. Our intent is to define the requirements for a functional interface that is realistic and effective from both performance and cost perspectives. We have specifically avoided describing a state-of-the-art, new technology solution to the problem. It may be necessary to defer implementation of some of these requirements based on resource availability, but we believe that the implementation should be at least a long-term goal of the GSDE.

Section 4.1 provides a brief description of the basic interface architecture between the GS/SPF and an IV&T system. The interface requirements are documented in Sections 4.2 through 4.7.

4.1 Interface Architecture

The basic architecture of the GS/SPF host -to-IV&T system interface that we have assumed is shown in figure 4-1. This is a "generic" architecture used for the purpose of describing the functional interface requirements. The specific interface architectures planned for the SSCC and SSTF systems are described in Appendix B.

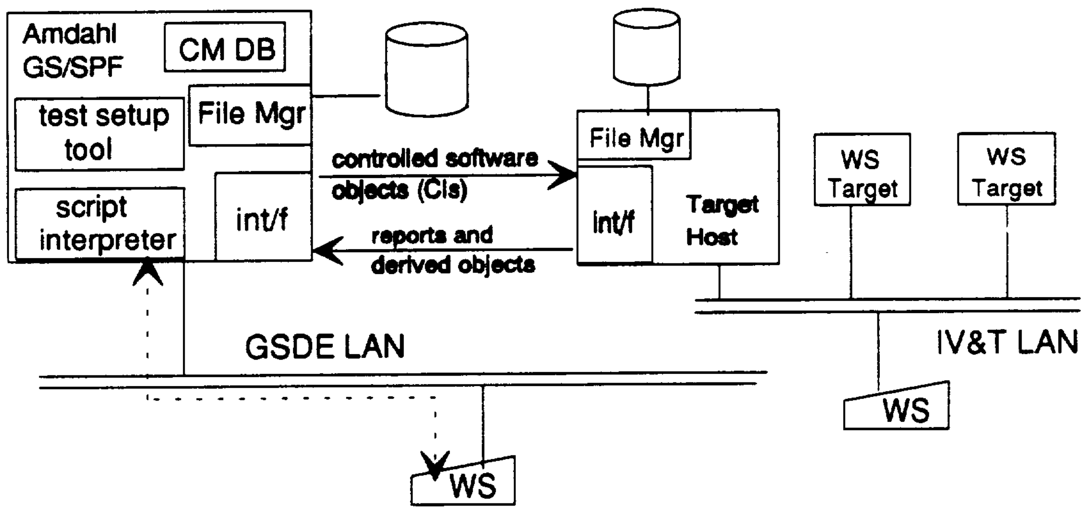


Figure 4-1 General Architecture of GSDE-IV&T Interface

The figure shows controlled storage on the GS/SPF host, with SSE-provided tools to facilitate the interface between the GSDE and the IV&T systems. Workstations are connected to the GSDE and Ops LANs. Physical communications between the IV&T and the GSDE is effected over the GSDE LAN. The primary data flows of concern are the workstation data flows (e.g., information, test scripts, control actions, CM reporting) and the file interchange between the mainframe computers.

The GS/SPF host sends controlled objects to the IV&T. These include of OADP source code, object code, testbeds, test data--anything which has been placed under formal GSDE CM to ensure accountability of software. The IV&T system returns reports on activities (e.g., test reports, tool version and tool product information, and possibly items to be controlled such as object code from target-compiled source code.

4.2 File and Information Transfer

This set of requirements addresses the transfer of data: files, information about files, status of CM processing, transfer processing status, and script operations status. As noted above, the intent of this report is to address the application-level interface rather than physical or protocol-level interface issues.

IF-1. The interface shall support the integrated transfer of data files and command scripts which operate on them, wherein the files and scripts are transferred together and the command scripts executed in a single operation. In such file-and-script processes, it shall not be necessary to transfer files in one operation and invoke the command scripts in a separate, subsequent operation.

Commercially available remote batch processing systems, such as IBM's MVS Remote Job Entry (RJE) system, provide integration of data files and commands. Such integration does not preclude separate file transfer and script transfer; it does permit file operations such as a target-based compile to be performed as atomic (indivisible) operations. This requirement is necessary to ensure repeatability of operations.

IF-2. The interface shall provide a mechanism for positive acknowledgement of file transfers at the level of the applications that send and receive the files. This mechanism will convey to the sending application any file transfer status report produced by the receiving application. This requirement is in addition to any acknowledgement provided by the network transport prototcol.

This reqquirement uses the context of the ISO model of network communications. The network file transfer protocol provides confirmation that the receiving computer actually received the transmitted file. This confirmation does not necessarily indicate that the receiving *application* was notified of the receipt, and was able to process the transmission. This requirement calls for a peer-to-peer acknowledgement mechanism.

For example, a file transmitted by the GS/SPF-based CM system to an IV&T-resident CM system would elicit acknowledgements at two levels: a file-received confirmation from the IV&T computer (transport layer), and a file-accepted confirmation from the target CM system (application layer). The latter would probably have additional, application-specific information such as the CM ID of the newly stored file.

IF-3. The interface shall provide a status inquiry and reporting mechanism for file-and-script processes. This mechanism shall provide a process-

ID facility to ensure that inquiries and reports can be matched to the correct processes. This mechanism shall, at a minimum, provide activity reports (e.g., "pending", "executing") upon request for active processes, and termination reports upon completion of processes. The mechanism shall support the transmission of whatever termination report is provided by the script processor, identified by the process ID.

Many commercially available remote batch systems provide job numbers, status reports on jobs pending or in progress, and detailed output listings in addition to the outputs of specific processes (e.g., compiler listings). This requirement specifies such a facility.

IF-4. The interface shall support a semi-automatic test status reporting mechanism, wherein checklists tailored to specific testing situations can be used by testers to report the completion status of each step in a test sequence. This mechanism shall incorporate data on the testbed and test environment as well as on the tests to be performed.

This requirement is based on the requirement for comprehensive test reporting to be conveyed from the IV&T system to the formal CM system. Most of the needed information is available as a byproduct of the testbed construction process. The determination of test status is not automated, so the best that can be expected in this area is a semi-automatic process. A mechanism that produces a printed checklist which is completed by the tester and entered manually into the GS/SPF CM system would be considered a partial satisfaction of the requirement.

4.3 Target-oriented Command Script Support

On the GS/SPF side, the CM system can provide generic scripts for compiling, building, and testing CIs. These generic scripts are suitable for the SSE-supported SPF host systems only. Some mechanism is needed to support tailoring those scripts to fit the script processors (command language processors) in ground system target platforms. On the IV&T side of the interface, there may be need to fill in parameter values (such as which computer to use for a specific test), and to pass scripts from one processor to another to perform distributed system building and testing.

The requirements in this section describe a jobstream processing system that provides support for distributed processing with centralized job initiation, control, and reporting.

IF-5. The interface shall support the tailoring of generic processing scripts for compilation, linking, directory maintenance, and execution of applications as appropriate to each target platform in an IV&T system.

This tailoring can be performed on either side of the interface, or with distributed support. If the tailoring is not automated (and therefore repeatable), the modified scripts must be submitted to the GS/SPF for CM.

IF-6. The interface shall support the distribution of command scripts and files to allocated processors, with dynamic subsititution of values for parameters as required by the distribution.

There are a variety of unique identifiers that must be supplied for script processing, particularly for testing. These include processor names, channel addresses, console addresses, etc. This requirement says that some executive process or network addressing scheme must exist to route scripts to their targets; and that the scripts can be dynamically completed in the routing process.

4.4 Support for File Directory Services and Verification

To maintain (or establish) traceability between file objects on both sides of the interface, mechanisms must exist to positively locate and identify files. The fact that there will be multiple versions of files, and multiple target processors, adds to complexity of the problem. The SSE traceability tools are expected to be provided with OI 6.0, scheduled for March 1992.

There is no requirement that file storage be provided in the IV&T system; but if files *are* stored and used from IV&T storage, the following requirements describe the constraints on their use. After the procedural details of the IV&T systems are developed, following the specification of the OADP platforms, further elaboration of these requirements will be appropriate.

IF-7. The interface shall support a mechanism for positive confirmation, that a file which is identified in the GSDE CM is identical to the file that was created in or moved to the IV&T system and recorded in the GSDE CM system.

Mechanisms for "positive confirmation" are defined in subsequent requirements. The phrase "stored or recorded" is intended to permit a controlled item to be available for use without actually being copied to the GSDE CM system, as long as a unique description of the item has been created in the GSDE CM record of the CI, and the item can be regemerated from controlled files that do reside in the GCM.

IF-8. Positive confirmation of file identity may be provided by use of an approved file security system that either prevents changes to a file in its control, or invariably records the fact of any change to such a file. To provide such positive confirmation, three conditions must hold:

a) The file must have been placed under control of the file security system as part of a controlled process such as compilation, file verification, or file transfer.

b) A complete, unambiguous file description of the stored file must have been recorded in the GSDE CM system as part of the process of placing the file under security control.

c) The description of the file being confirmed must match the description recorded in the GSDE CM system and included in the process script that directs the use of the file.

This requirement is intended to authorize the use of file security systems (e.g., RACF) as long as the controlled file is moved from creating process to secure storage to using process without any gaps where file corruption could occur. The three conditions prescribe that the file must be valid when stored and unchanged when retrieved.

IF-9. The interface shall support a mechanism such that, when a file is transferred to the GSDE CM system from storage in an IV&T system, there is positive confirmation that the file has not been changed since it was created in the IV&T system and recorded in the GSDE CM system.

The CM requirements specify that, when a file is created in a controlled process, it is recorded as a CI even if the file itself is not immediately moved to the GS/SPF host. If, at some later time, it is desired to move the file to GS/SPF host storage, there must be verification that the correct, unchanged file is being transferred.

4.5 CM Information Extraction and Packaging

Some of the CM information required to characterize activities in the IV&T system is available from the tools used in those activities. Mechanisms are needed to automatically extract such information, and to make it available to the GS/SPF-based CM system. Depending on the tools (including CM tools) available in the IV&T systems, the interface may consist of program-to-program interfaces or extraction of data from tool outputs (such as compilation listings).

Further elaboration of these requirements is needed following tools specification and the selection of the OADP platforms. Particularly, the specification of any IV&T-based CM tools will impact the elaboration of these requirements.

IF-10. The interface shall provide the capability to record information about the process of creating derivative files from controlled CM items. This capability may involve controlling the process, or extracting information from process outputs (e.g., compiler listings). The specific information to be recorded includes the following items:

a) name and version number of tool

b) completion status of operation

c) identification of any ancillary files or data used (e.g., system parameters)

d) date and time of operation

e) location (i.e., which computer) of operation

The recordation capability shall be able to be invoked from the same command script which directs the process about which information is to be recorded.

Complete repeatability of software processes demands that the *tools* used in producing object and load image files be *themselves* controlled. The control of tools (which often are commercial products that cannot be modified) is achieved by recording the name and version of the tool. This information is commonly available as part of the listing product of a compiler or builder.

IF-11. The interface shall provide the capability to combine all relevant and available CM information concerning an operation on a CI in a single record that can be used to create or update a CI record in the formal CM system. Ths record shall include:

a) CI identifier

b) user ID of person initiating operation

c) date, time, and location of operation

d) description of operation (e.g., information extracted from tool output)

e) completion status of operation.

The record shall be packaged for submission to the formal CM system as an update transaction.

This requirement closes the loop on extracting information from processes and using that information to update the records in the GS/SPF CM system. "Relevant" information is information defined in the SSE CM system for the particular operation and CI. "Available" information is whatever is available from the process that was used (e.g., from the script-processing system or from a tool-specific extractor). "Packaged...transaction" means that a programmatic interface to the CM system can read the record and submit it to the CM system without operator intervention.

4.6 Resource Scheduling Support

Particularly during testing, the scheduling of IV&T resources is a complex matter. This interface is not concerned with the actual scheduling and allocation. It is concerned with *requests* for resources that are based on CM-controlled test specifications, and with conveying resource allocations to the target scripts and test status reports.

IF-12. The interface shall provide a mechanism for requesting the allocation of resources for a process, and for dynamic resolution of generic resource allocations in a process. The mechanism on any given IV&T system shall be compatible with the resource scheduling and allocation procedure, automated or manual, that is used on that system.

Requesting allocation data could involve displaying a list of dynamic resources, or attempting to open a parameter file where the allocations are stored. Dynamic resolution involves replacing dummy parameters with specific allocations and access addresses. The compatibility requirement is intended to insure that this requirement does not drive the IV&T resource scheduling system design.

IF-13. If the resource request and allocation mechanism involves manual editing of a controlled process script, the editing capability shall be restricted to changes to resource allocations in the script.

To preserve accountability and repeatability, there must not be a free-text editing procedure in the execution sequence. A restricted "forms-filling" procedure would be appropriate, and would address both the request and the resolution aspects of the requirement. A standard text editor would violate this requirement.

IF-14. All dynamic resource allocations performed in a controlled process shall be recorded as CM information describing the process.

This requirement directs that any resource allocation changes to a controlled script be reported in the process status output. Such reporting is a common feature of remote batch execution systems, which would probably (depending on specifics) satisfy this requirement.

4.7 Remote Login Support

During the IV&T of software, there must be provision for testing and debugging. This is not generally desired in an operational setting, but for these software systems the operational target environments will be used for IV&T. Accordingly, the interface between the GS/SPF and the IV&T system must support interactive test sessions on target platforms from development workstations. To maintain the independence of the IV&T and development systems, this support should be provided through interconnections already defined.

IF-15. The interface shall provide a mechanism whereby interactive sessions can be established between target platforms and development workstations for the purpose of testing CIs. These interactive sessions shall be automatically recorded as activity reports related to the CI or CIs being tested. These sessions shall be restricted to testing activities, and shall not be used to transfer or modify controlled software on the target platforms.

This requirement supports remote login from development areas to the IV&T systems during testing and debugging. Restrictions on use of this capability should be embedded in the remote session software as much as possible, but can be imposed procedurally. The logging of interactive sessions (including who logged on and what software they tested) must be automated for the sake of accountability.

Section 5 - Requirements Implementation Notes

The purpose of this study was to define the requirements presented in Sections 3 and 4. In the process of drafting the requirements, characterizing the environment, and reviewing with contractors, the study team collected and developed information that may contribute to understanding and implementing these requirements. The study team also reviewed the software development plans for both the SSCC and the SSTF, and tried to assess the requirements in terms of procedures that the two contractors had planned or already in place. This information and these assessments are included as appendices to the study report.

Appendix A describes in some detail the CM-related contents of the Oracle-based database designed as part of the SSE. More detail and specifics on the use of this information can be found in the SSE CM Users guide.

Appendix B describes the development architectures of planned by the MSC and TSC contractors. This information was based on material available during the study, and is subject to change as the space station project evolves.

Appendix C reports on the analysis of the two development architectures by the study team, and includes recommendations and sugestions for implementing the requirements in this report. The requirements for CM support are assessed in terms of anticipated compliance; the interface requirements are discussed in terms of applicability based on current development environment plans.

Appendices D and E contain information provided by the MSC and TSC contractors as part of the information gathering procvess of the task. We wish to note, again, that the responses by both contractors demonstrated how seriously they took the importance of software configuration management.

Appendix F contains a final report on the Cronus network application environment, which the study team investigated as a candidate for mechanizing the interfaces within the GSDE.

Appendix A - Configuration Management Data Elements

The information in this appendix identifies the information (attributes, or "fields") that the SSE-provided CM system can record and track about a configuration item.

Configuration identification fields

The SSE requirements document describes a large number of classes of items which can be placed under configuration control. This analysis is only concerned with software, which still includes a respectable list. Configuration items can be life-cycle products, support files, documentation, test data, test plans, or aggregates of items. The list below consists of fields used to identify items under configuration control.

Configuration Item (CI) identifier

Configuration Item name

Configuration Item description

Configuration Item version

Software integration hierarchies descriptions

Configuration identification sensitivity levels:

0-Negligible impact
1-Minimal Impact
2-Adverse Impact;
3- Irreparable Impact

Security snformation

1-Personal
2-Financial, Commercial, Trade Secret
3- NASA Internal Operations
4- Investigation,Intelligence Related, Security
5- Other Federal Agency
6- Unclassified National Security-Related
7- National Resource Systems
8- Mission Critical
9- Operational
10- Life Critical
11- High or New Technology
12- Other Unclassified

PRECEDING PAGE BLANK NOT FILMED

Configuration Item modification fields

Change control is an essential element of CM. The following fields describe the record-keeping needed for changes to controlled items.

Configuration ID information of the CI affected

Status of CI before modification

Status of CI after modification

USERID of the person performing the modification

Date and time of modification

Reason for modification

Test transaction information

From an IV&T standpoint, a most impoirtant category of controlled information is test transaction data. The following list describes attributes of tests or items being tested.

Testbed software configuration

Type of transaction performed

Configuration ID information of the test resource affected

Status of test resource before transaction

Status of test resource after transaction

USERID of person performing transaction

Date and time of transaction

Testing process information

Test activities are recorded, both to assist with analysis of testing and for process improvement with IV&T process metrics.

Test tools

Test data

Test scripts

Configuration ID information of the test invoked

Functional requirements implemented

Name of analyst

For test results: type of transaction performed

Configuration ID information of the test whose results are being posted

Configuration ID information of the CIs tested

Testbed resource information

Support for testing is provided via controlled tools, testbeds, and recording. The fields in the following list relate to the context of test execution.

Test resource class

Resource class/relationship value

Resource class/attribute value

Configuration ID of CIs being tested

Status of all CIs before testbed build

Status of all CIs after testbed build

USERID of person building testbed

Date and time of testbed build

Status of all affected CIs before posting

Status of all affected CIs after posting

USERID of person posting results

Date and time results posted

Optional remarks

USERID of person executing test procedure

USERID of person authorizing bypass of the previous test in a test sequence

Date of test execution

Time of test execution

Configuration IDs of productstested

Current Status of Test Resources:

Under Development
In Test
Ready for Test
Completed Test
Ready for Test with Bypass
Failed Test

Test Procedure Identifier

Component placed in Test

Period the Components Were Under Test

Number of Test Failures For Each Component

Test Results once the Testbed is Successfully Built

- Passed Test
- Failed for Rework
- Failed for Retest
- Failed with Bypass
- Test Bypassed
- Defective Test

Appendix B - Contractor Interface Architectures

This appendix describes the connectivity and functionality planned by MSC and TSC to support the IV&T process.

The following sections describe the procedures and methodologies that will be used by the MSC contractor and the TSC contractor for the configuration management of software being integrated and tested for the respective contracts. (Development-phase CM is generally outside the scope of this study and will be described only as necessary for clarity). Figure 3-1 describes a general CM flow of information that is necessary for a centralized CM. This description is based on the requirements given by NASA for CM of ground systems software.

The emphasis of the discussions is on the use of the SSE-provided CM tools within the GS/SPF host (currently the GSDE Amdahl computer located in building 46), and on the connectivity that is employed to use these tools. The GS/SPF-resident CM system is referred to in this report as the *GS/SPF-based CM system*, or GCM. It should be noted that these SSE provided tools will primarily be used for the management of software that has been through development and unit testing and has been submitted for testing in the IV&T (formerly known as the Integration, Verification and Test Environment).

In addition to the CM tools provided within the GS/SPF host, both contractors have other or additional CM tools that reside in computers other than the GS/SPF. Those CM tools will be mentioned in the following subsections only in the context of where they fit in the CM scheme for the individual contractor.

It is important that all CM information be accurate as this information will be a source of software metrics information to be gathered throughout the SSCC and SSTF programs. This information will aid not only the contractors but will be used by NASA in determining cost and schedule parameters for these as well as other projects and programs.

B.1 Mission Systems Contract Configuration Management Architecture

Figure B-1 describes the software development-to-CM testing configuration proposed by the MSC contractor. The figure shows three distinct areas involved in the life cycle of the software. These are: the software development area which is located within the contractor's facilities: the GS/SPF area located within building 46 at JSC; and the testing area located in building 30S at JSC.

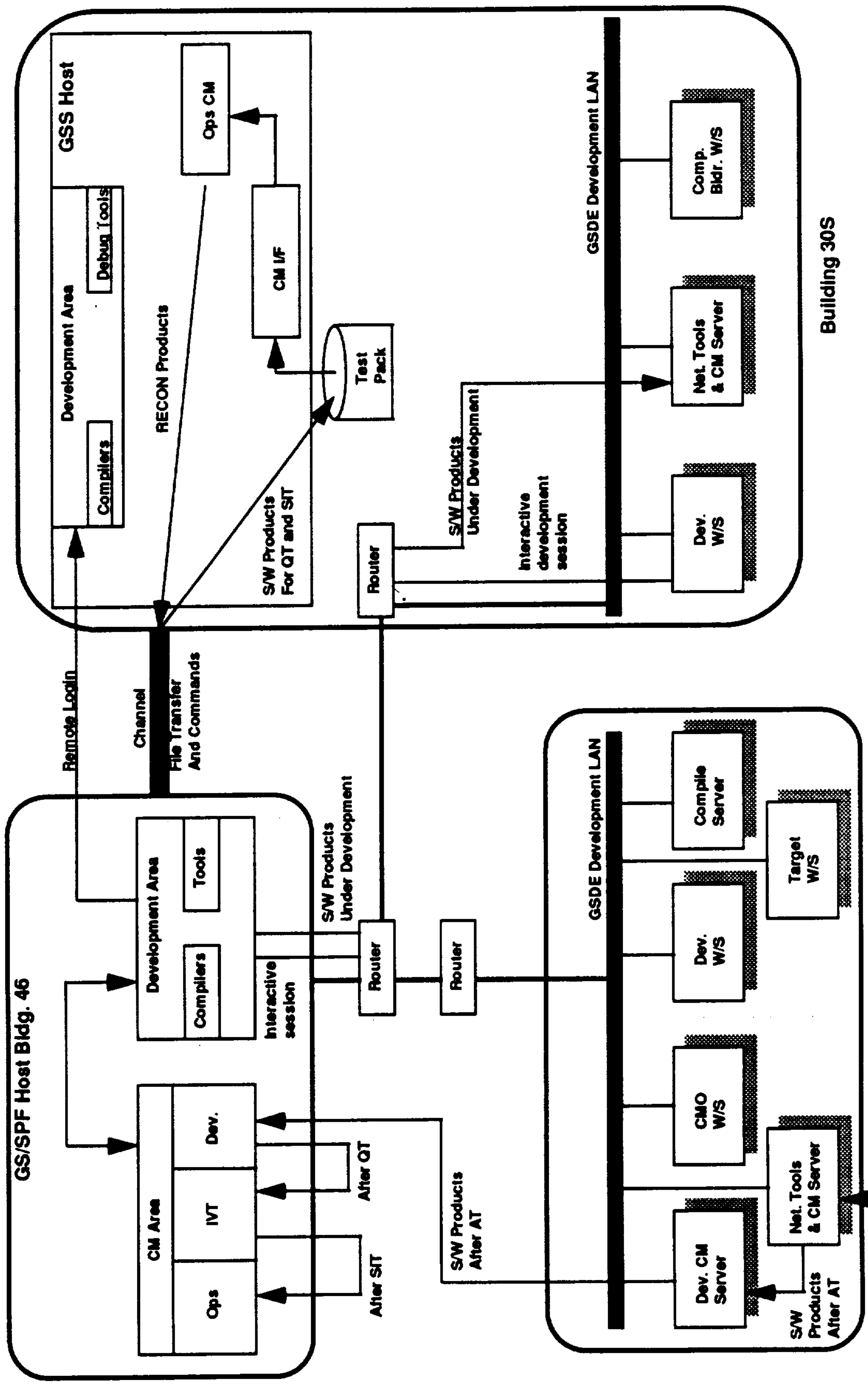


Figure B-1 MSC Configuration Management Architecture

It is anticipated that the contractor will make full use of the SSE provided CM tools located within the GS/SPF host. In addition to the SSE provided CM tools, the MSC contractor plans to use other CM tools throughout the life cycle in support of the SSE tools.

The following subsections describe the CM methods to be used by the MSC contractor.

B.1.1 MSC CM of Workstation Software

The MSC development area will be distributed across several physical locations (as shown in figure B-1) but connected by the GSDE development LAN. This LAN exists not only in the MSC contractor facilities but also within the testing area in building 30S at JSC. Connected to this LAN are development workstations, target workstations, compile engines, and CM servers. This is both the area for the development of code by the MSC contractor and the entry point for acceptance-tested code that is developed by sub-contractors in locations not attached to the development LAN.

At this time the MSC contractor does not plan to use the SSE provided CM tools located in the GS/SPF host for CM services during the early stages of development (through code unit testing). SSE personnel confirmed that the SSE-provided CM system, with its rigorous control and authorization requirements, was not designed for the development phase and might be cumbersome in that role. (The MSC contractor is considering the use of a copy of the SSE CM tool that would be located within the development facilities for use as an early development CM tool).

The CM servers within the development areas for workstation code development are the focal point for communication between the development areas and the SSE CM tool in the GS/SPF host. There are several instances when information will flow between these servers and the CM tool within the GS/SPF host. Described below are such instances.

- The development area CM servers will receive (or generate) CM information as well as actual source and object code when code is accepted from a subcontractor or some source other than the development facility. Acceptance testing will be done and the code along with the necessary CM information will be uploaded to the GS/SPF host for submission into the formal CM system.

- Code that is developed within the development area and has gone through unit and acceptance testing will be uploaded to the CM tool within the GS/SPF host.

- Source code that is downloaded from the GCM to the development area for compilation (this is code that is destined for target workstations) will be compiled in the development areas. The workstations that serve as development platforms are essentially equivalent to the target platforms, but for this role are connected to the development system. Information about the compilation will be uploaded to the CM within the GS/SPF host.

It should be noted here that the distribution of configuration-controlled materials (i.e., source code) is not controlled by the GS/SPF host CM but rather is controlled (with automated procedures as far as possible) within the development area. In addition, information and products that will be placed back under CM control within the GS/SPF host will be manually entered in the development area; this interface is not an automated function of the GS/SPF host-to-development area interface. Because this is a non-automated interface, there exists a potential loss of CM control.

As described by the MSC contractor, all compilation and unit testing of target workstation software will be done in the development areas. (This includes the target platforms that serve as development workstations when connected to the development area that is co-located with the test environment in building 30S).

The manual connection between the development area and the CM within the GS/SPF host will be effected by use of a workstation connected to the GS/SPF host. Manual entry will be made through this connection to supply CM information to the CM system as well as manually transfer files to the GS/SPF host containing the materials to be placed under CM control.

B.1.2 MSC CM of Mainframe Software

Other than the workstation testing that is done within development areas as described in the preceding paragraphs, a formal testing area exists for the testing of mainframe software to be developed for the SSCC. This testing area is located with building 30S of JSC and is channel-connected to the GS/SPF host.

The testing environment is principally a mainframe host computer, the Ground Support System (GSS) computer, that houses both a host software development area and a test environment (both internal to the mainframe). The MSC contractor described the separation of these two "facilities" within a single machine as being protected areas of memory and DASD with password security used to keep CM "clean" between development and testing.

There are several instances of CM controlled materials and information being exchanged between the GCM and the GSS host. Those instances are described below.

- Software that is developed for the host and has gone through unit and acceptance testing will be transferred to the GS/SPF host and placed under CM control. Included in this transfer will be source and object code as well as CM information that will be used to track and control the software.

- Software materials as well as CM information used in the testing of mainframe software will be downloaded from the GS/SPF host to the GSS mainframe. In this transfer, the GSS hosts a facility that will accomplish several CM functions. One of these functions is to distribute the software products for testing as well as

CM-provided information to the appropriate testing or operations target. This operations CM is not a replacement for the GSDE CM but is a focal point within the GSS for collecting and distributing software materials and CM information.

- Software materials and CM information that is collected within the GSS will be uploaded to the CM within the GS/SPF host. This material will also include reconfiguration products that will be used in testing. These reconfiguration materials will be produced outside of the GSDE, but the MSC contractor has planned the storage of some of these products in the GS/SPF host under the control of the GSDE CM.
- For test builds, the MSC contractor has described the used of the GSDE CM to upload software materials (including some reconfiguration products) and CM information to a test pack through the GSS mainframe. This test pack will be physically secured and will contain the entire test package to be used. Once the test is completed the software materials and the CM information will be uploaded back to the GSDE CM in the GS/SPF host.

As is the case for the target workstation software, mainframe software and the accompanying CM information is transferred by manual means through the use of a development workstation located in the testing area. As is noted in figure B-1 there will be a connection between the development workstation within the testing area in building 30S and the GSS host development area.

B.2 Training Systems Contract Configuration Management Architecture

Figure B-2 describes the software development-to-CM testing configuration for the TSC contractor. The figure shows three distinct areas involved in the life cycle of the software. These are the software development area which is located within the contractor's facilities, the GS/SPF area located within building 46 at the JSC, and the testing area located in building 5/5A at JSC.

From discussions with the TSC contractor, information was obtained about the methods and tools to be used for the configuration management of software developed, tested and delivered by the TSC contractor. (TSC contractor responses to questions about CM are attached to this document in Appendix D). In general, the TSC contractor plans to use CM tools provided by the Rational Ada development environment and CM tools located within the reconfiguration host (located in the testing/operations environment) to provide the bulk of the CM needed throughout the software life cycle. However, this does not mean that the SSE provided tools located within the GS/SPF host will not be used.

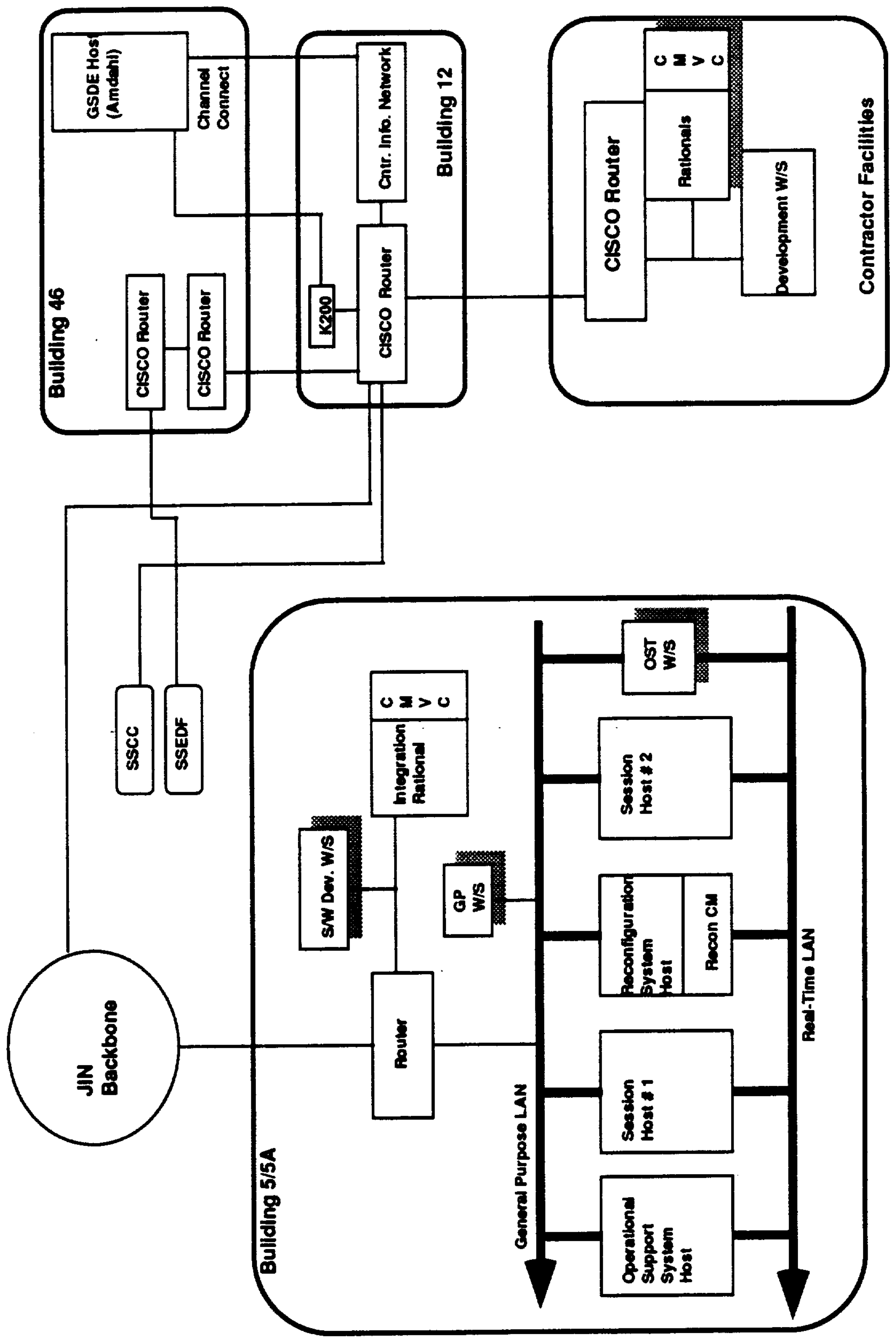


Figure B-2 TSC Configuration Management Architecture

The following subsections will describe the CM flow as proposed by the TSC contractor.

B.2.1 TSC CM Through Development

The TSC development area is really distributed across two locations. There are the Rational Ada development environments and development work stations located in the contractor facilities, and a Rational Ada development environment machine (referred to as the "integration Rational") located in the testing area. (This Rational is directly connected to both the development area within the contractor facilities and the test/target machines).

As described by the TSC contractor, development and non-real-time testing will be accomplished on the Rationals and the configuration management for this code is done with the local CM tools (most of the local CM is handled by the Rational provided tool, CMVC). At the point that the code is ready for delivery, data items are copied to the SSE CM system within the GS/SPF host. Preliminary real-time testing is performed on the IV&T systems without the involvement of the GS/SPF host or the GCM. The TSC contractor has stated that "neither the SSE CM tools nor the Amdahl will be a necessary part of moving software to the IV&T environment for preliminary testing". The user at a software development workstation will be able to file-transfer new software to the integration Rational without accessing the GS/SPF host.

The SSE will provide a tool interface between the SSE-developed GCM system and the Rational CMVC. This interface will permit software configuration items to be placed under GCM. The interface will support the transfer of image files that can only be interpreted on a Rational system (that is, files which cannot be listed or edited on the GS/SPF host). It will also support the transfer of text versions of the same meterials, so that file-level CM can be imposed. This support is available in OI 5.0, and will be enhanced in OI 6.0. (The Rational image files contain entire subsystems, and do not lend themselves to lower-level control.)

B.2.2 TSC CM during Testing

The TSC contractor plans to use the Rational CMVC tool, the GCM system, and a TSC-supplied CM tool hosted in the reconfiguration computer (see figure B-2) for software CM during IV&T. The use of the three systems is described in the sequence that follows.

1. Code that has been prepared for formal testing will be checked into the GCM system. If this is Ada code developed on a Rational, there will be text files and CM information along with a Rational image file that cannot be processed on the GS/SPF host. If this is the first time this software has been uploaded to the GS/SPF host, it will be entered into the GCM system. If this is a subsequent upload, the text files will be compared with those already stored; any

discrepancies must be authorized by some change instrument. No comparison test will be performed on the Rational image file.

2. The Rational image file is downloaded to the integration Rational for interpretation, along with any necessary test data objects, such as test definition files and test scripts which are controlled items stored in the GCM system.

3. The Reconfiguration system is invoked to create a loadable image file from the Rational image file. This loadable image is flagged within the Reconfiguration CM system as an unverified image. This sequence of operations is essentially invisible to the GCM system.

4. Software testing is performed in the IV&T system. If minor errors are found or minor changes required, the image on the Integration Rational is modified. Traceability from source to tested load image is provided by the Reconfiguration CM system.

5. After successful testing, the loadable image is flagged as a verified image. Test results are recorded in the GS/SPF host from a development workstation; the software CIs are marked as "tested". The loadable image may or may not be uploaded to the GS/SPF host. The reporting and optional uploading of information is a manual process controlled by an authorized user from a development workstation.

6. Delivery to operations is achieved by using the Reconfiguration system, to link operational data and software with the compiled Rational image that may either be stored in the integration Rational or downloaded from the GCM system.

It is not expected that operational software will be stored in the GCM system. Operational load images will be managed by the operational CM system.

The interrelationships of the various CM systems are shown in figure B-3.

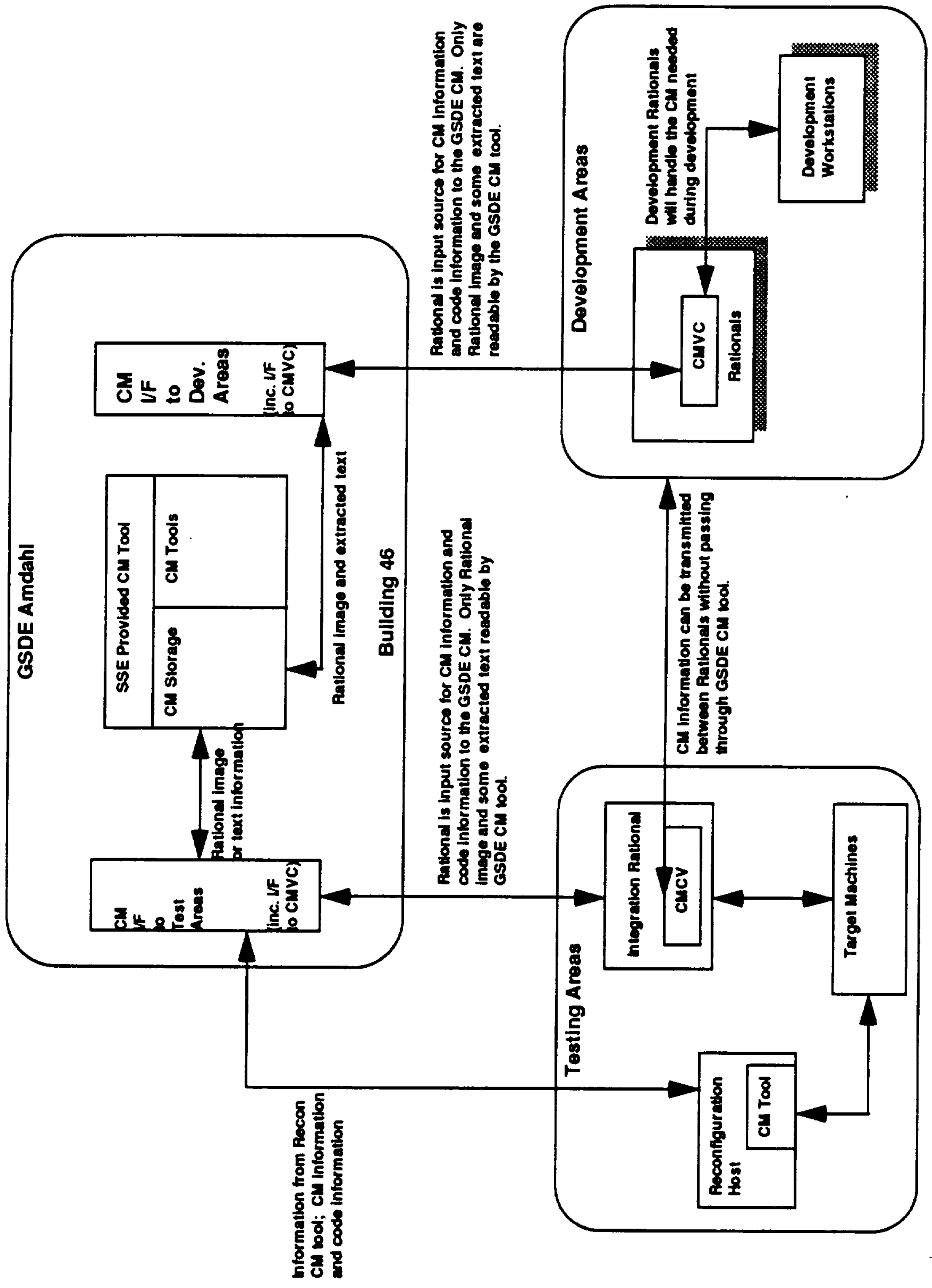

GSDE Amdahl
SSE Provided CM Tool
CM Storage
CM Tools
CM I/F to Dev. Areas (inc. I/F to CMVC)
CM I/F to Test Areas (inc. I/F to CMVC)
Rational image or text information
Rational image and extracted text
Building 46
Rational is input source for CM information and code information to the GSDE CM. Only Rational image and some extracted text are readable by the GSDE CM tool.
Development Areas
Development Rationals will handle the CM needed during development
Development Workstations
CMVC
Rationals
CM information can be transmitted between Rationals without passing through GSDE CM tool.
Rational is input source for CM information and code information to the GSDE CM. Only Rational image and some extracted text readable by GSDE CM tool.
Testing Areas
Integration Rational
CMCV
Target Machines
Reconfiguration Host
CM Tool
Information from Recon CM tool; CM information and code information

Figure B-3 TSC CM Configuration

Appendix C - Interface Architecture Analysis

CSC's intent in analyzing the SSCC and SSTF software development plans is to contribute to the development of quality software in a controlled, monitored process. The two systems are both large and complex, involving new technology in both hardware and software. There is increasing recognition within the NASA community of the critical importance of total quality management, in which CM plays a major role. Our intent is not to criticize the efforts of either contractor, but to point out areas where additional attention to the development process may provide significant benefits in terms of quality. It is clear from the level of detail provided to us that both contractors take CM very seriously. Our goal in this entire analysis is to contribute to the successful development of total configuration management efforts for ground systems software.

C.1 Analysis of MSC and TSC CM Plans

The requirements detailed in sections 3 and 4 were developed, in part, from the software development plans and procedures described by the MSC and TSC contractors. It is important to recognize that those plans were developed prior to this list of requirements.

Nonetheless it is useful to compare this general set of requirements with the specific plans developed by the two contractors for their respective development efforts. This appendix provides a lists of the CM and interface requirements and an assessment of how closely the proposed development plans appear to satisfy them. These assessments are intended to suggest areas for further development of those plans.

Table C-1 lists the configuration management interface requirements presented in section 3. While the specific requirements listed in this report are newly stated, the general requirements for software configuration management are well known and long established. The MSC and TSC plans can fairly be assessed in terms of fundamental principles of software CM. Subsection C.1.1 presents the basic principles used as a basis for assessment. Subsections C.1.3 and C.1.4 analyze the two plans in that context.

PRECEDING PAGE BLANK NOT FILMED

Table C-1 Requirements vs. GSDE CM system (GCM) utilization matrix

Reqt	Statement	MSC GCM utilization		TSC GCM utilization	
		use	assessment	use	assessment
CM-1	object code becomes GCM CI	yes	GS/SPF will be primary CI storage mechanism	no	object code storage in GCM not a mandatory process
CM-2	compile script is GCM CI	yes	specific details are TBD	no	compile process is manual, Rational-based, emphasizes flexibility rather than control
CM-3	compilation info goes to GCM	yes	specific details are TBD	???	information not provided
CM-4	compilation info becomes part of object code CI record	yes	shares mechanism with CM-1	???	information not provided
CM-5	no-code-output compilation recorded	yes	all operations on CIs will be recorded	no	results only reported, not all operations
CM-6	load image is GCM CI	yes	GS/SPF will be primary CI storage mechanism	no	GS/SPF usage not integral to process
CM-7	build script is GCM CI	yes	specific details are TBD	no	process is manual, controlled in Rational and Recon system
CM-8	build process info goes to GCM	yes	specific details are TBD	???	information not provided
CM-9	build info becomes part of load image CI record	yes	shares mechanism with CM-8	???	information not provided
CM-10	no-load-image build process is recorded	yes	all operations on CIs will be recorded	???	information not provided
CM-11	test items are GCM CIs	yes	GS/SPF will be primary CI storage mechanism	no	GS/SPF usage not integral to process
CM-12	tests recorded in GCM	yes	all operations on CIs will be recorded	no	results only if test successful
CM-13	test reports linked to build product CI	yes	all operations on CIs will be recorded	no	reporting not integral to process

In Table C-1, the GCM usage columns include a brief note on CSC's determination of expressed intent to use the GSDE Configuration Management system for configuration management, with an explanatory note to show the basis for the assessment. In some cases we could not make a determination due to lack of information (often because the system designs are not yet mature). Where there is an expressed intent to use the GCM, but some question as to the availability of resources to provided automated support, the early stage of development is noted.

C.1.1 Principles of Software CM

The basic principles of software configuration management are well understood and are embodied in most software management standards and plans. The principles stated here are those identified in CSC's *Digital Systems Development Methodology* (and many other guides).

Configuration identification--any object (or *configuration item*, CI) that is to be controlled must be uniquely identified. If a CI is a composite, all of its elements must also be CIs. The identification of a CI includes its origin and history. Different versions of a CI must be uniquely identified in a manner that shows the evolutionary relationship of the CIs.

Configuration control--configuration items can be changed or deleted only with authorization specific to that CI. A change to a CI results in a new version of that CI. If a CI is used to produce a derivative CI (e.g., source code is used to produce object code) the original CI must be locked to change or deletion for the lifetime of the derivative CI.

Configuration tracking and status accounting--all changes to a configuration item must be recorded. The change record must indicate what change was made, when it was made, and what authorization existed for the change. If a CI is derived from another CI (e.g., object code is derived from source code) the record of the derivative CI must show how it was derived and from what CI or CIs.

Configuration auditability--There must be complete traceability from a configuration baseline to the current configuration of a system; the traceability must be adequate to support an audit of all changes to the baseline that are necessary to achieve the current baseline or baselines.

C.1.2 Interface Requirements Implementation

The interface requirements presented in section 4, unlike the CM requirements discussed in table C-1, are quite specific to the operational interfaces to be constructed by the two contractors. An assessment of compliance with these requirements will not be possible until the design process is more mature and details of hardware and software components have been defined.

C.1.3 MSC CM Assessment

CSC's overall assessment is that the MSC contractor intends to meet the CM and functional interface requirements in all areas where information is presently available. There are some concerns as to the availability of resources to implement some of the interface mechanisms, but there is full agreement with the CM requirements.

One area where MSC's plans do not adhere to the strict GSDE CM approach is in the development and test of some workstation software. Some of the workstation software will be developed and tested on the same machines with connectivity which bypasses the GSDE CM in the GS/SPF host. This deviation to CM control has been discussed with NASA and negotiations are continuing.

One other area of concern is in the use of test packs to store test images and data during a testing period. Current MSC plans do not describe a mechanism whereby the files on those test packs can be either secured from modification or verified before use.

C.1.4 TSC CM Assessment

CSC's overall assessment is that TSC falls short of complying with the GSDE CM requirements in that the traceability and auditability of the system is undefined. The audit trail goes through a TSC-controlled CM process that allows anything to be changed, and that breaks the hard connection between testing and certification. Figure C-1 shows this process. Information from the TSC contractor makes it clear that there will be CM imposed on all software, via the integration Rational or the reconfiguration CM system; but NASA's visibility into the integration and test process is severely compromised.

There are three basic problems with the TSC-described method of CM for the SSTF. The first is that there are three (GSDE, Rational, and Reconfiguration) distinct CM systems for use during the software life cycle. This inherently leads to duplication of effort, CM systems holding different or contradictory information, and a loss of centralized accountability. The second problem is that the GSDE CM tool in the GS/SPF host can be circumvented at each step within the life cycle. This poses a potential loss of CM information and control. The third problem is that under the TSC plan. the GSDE CM will be used to mostly store "completed" code. This will mean that the history and metrics of the code will be absent or only partially available.

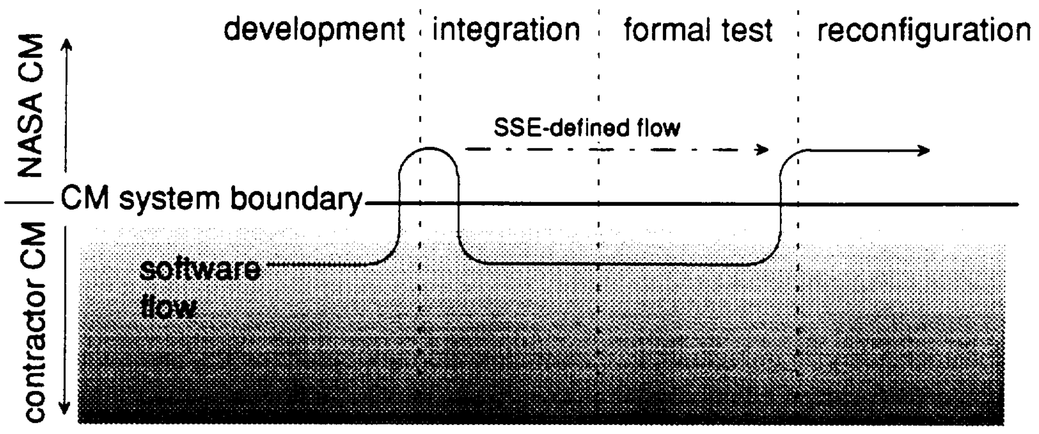


Figure C-1 Visibility into TSC Software CM

C.2 Implementation findings

As stated in the beginning of this section, both the MSC and TSC contractors have taken a serious look into the problems of configuration management of these very large scale software development activities. This section presents some suggestions in general for improvements to the CM process as well as some suggestions for the individual contractors.

In general, there are two broad areas for improving the software CM of both the SSTF and the SSCC. These are areas in which more automation of the CM process should be implemented, and where specific requirements and direction should be given to the SSE for support for CM in ground software development activities. It is understood that both these areas discussed below are not so much technology issues as budget issues. These areas are described here to suggest priorities for a "budget wish list".

The first priority should be given to the automation of some of the CM processes. In detail this means developing the necessary tools that will allow the CM tools within the Amdahl (provided by the SSE) to communicate with the target test machines directly and interactively. These tools include command script processors, compile report generators, test results generators, *et al.* These needed items are part of the SSE charter, but no requirements have been written for them and no funding has been provided.

The second priority should be given to automating the information input required of the user to get the correct CM information from the test or development areas to the GS/SPF-based CM system. This means that there should be automated facilities that present pre-defined screens to the user to make sure that all information needed by the

CM tool in the GS/SPF host is collected and in the correct format. This automation will allow the GSDE CM tool to make checks of the information for correctness and completeness.

It can not be stressed enough that manual CM procedures dilute the intent of CM. The more the CM process can be automated, the more human error will be removed from the CM process. This will serve to give confidence not only in the CM information but also in the software metrics that are based on the CM information.

C.2.1 Specific Suggestions for the MSC Contractor

The MSC plan for CM appears to be very well thought out and shows a lot of understanding about the development of large scale software systems. But in reviewing the planned CM procedures and methodology of the MSC contractor, a few areas for "tightening" the CM process were uncovered. The following suggestions are made to help close potential gaps in the CM process.

As was stated in the general suggestions for improving the CM process for both the MSC and TSC, automation of CM information and code materials between the GS/SPF host and the development and target areas is a must. The MSC should look into defining these areas for the SSE.

Another area of potential CM gaps is in having a development area, CM system (other than the GSDE CM), and target all within one machine (GSS host). This combination makes it very difficult to protect the integrity of CIs and can lead to the corruption of CM information. Although CM information generated within the GSS host is transferred to the GSDE CM there is still the possibility that developed or edited code might pass from the development area to the test area without being checked into the GSDE CM. It appears from reading the MSC procedures for this situation that there are manual procedures designed to eliminate this problem, but that the potential for human error has not been adequately factored into the design.

A third area of concern is that there is a third CM tool in the picture. This is the SSCC CM. Although this system is mostly used for the delivery of operational software to the operations environment, this CM tool is a focal point for CM information and code materials that pass between the GS/SPF host and the development and target machines. This second CM tool presents an area for possible CM corruption.

C.2.2 Specific Suggestions for the TSC Contractor

There are some areas within the TSC plan for CM that caused some concern for the integrity of the CM information to be captured in the GSDE CM. The following are some suggestions that are made to help resolve some of these concerns.

As with the MSC, the TSC contractor should look into methods to automate as much of the CM interface between the GSDE CM and the development and target areas. These requirements should be levied on the SSE.

The main area of concern for CM integrity is the physical connection between the development areas and the testing areas which bypasses the CM tool within the GS/SPF host. This connection through the "integration Rational" provides a bypass that circumvents the audit trail of the GCM (see figure C-1). Although it is understood that the Rational has CM capability, we suggest that the "integration Rational" be physically and logically connected to the development side of the GCM tool and not connected to both development and testing sides of the CM.

The second suggestion is that the Reconfiguration CM be used only for the development of operational loads, and not for the CM for the development and testing of software prior to system integration testing. Dividing the development and testing CM responsibilities among three different CM systems with different CM tools leads to a potential for corruption of the CM information in both systems, and will make metrics collection more difficult.

Appendix D - TSC Responses

The material on the following pages was provided by Training Systems Contract personnel in response to the Operational scenarios developed during this study.

Ground Software Development Environment (GSDE) Interface Requirements Analysis: Operations Scenarios

comments from TSC/Link

contents:

1. Questionable assumptions in CSC document

The CSC GSDE-IVTE analysis document is based on several assumptions that do not reflect how TSC plans to do business in building the SSTF. These assumptions are basic enough that the document as a whole requires reassessment. The specific listed questions and assumptions from section 4.6 of the CSC document are addressed in section.4 of this document.

Following is a list of the questionable assumptions in the CSC document. This list does not include the basic assumption that an IVTE will exist, even though there is no longer a separate identified set of hardware that makes up an IVTE. The processes of integration, verification and test will have to occur in a more loosely-controlled environment than the formal training environment. While this environment will exist on the same hardware as the training environment, it will have to be a functionally distinct environment. Thus, the basic assumption that there will be a functionally separate IVTE is accepted.

The questionable assumptions are:

- Most target compilation is hosted in the SPE.

  No target compilation will be hosted in the SPE. This is one reason that the SSTF developer needs more open access to the IVTE than CSC assumes.

- The software development user on a development workstation will not be able to log onto the IVTE machines; all interactions with IVTE machines from the development environment will be via the Amdahl host computer.

  TSC expects the user at some software development level 3 workstations to be able to access the IVT environment and the Reconfiguration system. These workstations will be, in fact, the primary access to the IVT environment. Tests will be conducted, test data collected, and test reports generated using these software development workstations. The current level of design does not address the question of whether the limitations on this access will be purely based on user identification, or whether the IVT and Recon capabilities will be limited to a subset of the level 3 workstations.

- The SSE Configuration Management (CM) tool on the Amdahl will be the only conduit through which source code can be moved from the development environment to the IVTE.

GSDE Interface Rqts Analysis/Scenarios — TSC/Link comments

This will only be true for formal training loads. TSC plans to provide more flexibility for development and testing that needs target compilation and execution. Neither the SSE CM tool nor the Amdahl will be a necessary part of moving software to the IVT environment for preliminary testing.

The Rational TBU facility, on the integration Rational connected to the Amdahl host computer, will be used as the source for all code to be compiled and for compilation batch scripts. The Rational already has the capabilities in place to manage selection and downloading of source code and creation of batch scripts for compilation and linking. This is in accordance with SSE guidance concerning use of SPF facilities.

TSC anticipates that this facility will be tailorable for non-Ada source code, as well as for Ada source code. If this turns out to not be the case, more traditional MAKE tools will be used.

The user at a software development workstation will be able to file-transfer new software to the integration Rational without accessing the Amdahl.

- There is a security barrier between the software development environment and the IVTE.

  According to SSTF level A requirements, the entire SSTF is considered a single level 3 data processing facility. Individual machines will require user access authority (logins), but there is no security concern that keeps software development users out of the IVTE, or controls their actions there to any greater degree than they are controlled in the development environment. There is no security barrier between a software developer and the IVTE.

- Automation of the collection of code for target compilation, of target compilation, and of managing the compiled/linked loadable images is a concern of the software development environment (GSDE).

  Target compilation and linking is a function of the SSTF Reconfiguration system; this is outside the GSDE. All automation of the target compilation process occurs inside the Reconfiguration system. The Reconfiguration system will be hosted in the target environment, close to the target compilers it must control.

  TSC has intentionally deferred some automation of the compilation/linking process until enough experience with the process is collected to indicate what can reasonably and profitably be automated, and what parts of the process require so much flexibility that they must remain under user control.

- CM of the GSDE-IVTE interface, including management of the interface between the Development CM system and the operational CM system, will be performed by the GSDE.

GSDE Interface Rqts Analysis/Scenarios — TSC/Link comments

> The SSTF Reconfiguration system manages data coming across this interface, manages the data objects during the compilation and load build process, and tracks their submission into operational CM and their return to Development CM.

In short, CSC appears to envision a centralized, automated process to be used for all S/W DEV - IVTE interactions, hosted on and controlled by the Amdahl. It is necessary to change conceptual environments, if not physical workstations, to test target-compiled software in the IVTE. Only compiled code is passed to the IVTE, which is apparently only used for formal testing and integration. CSC assumes that target hardware is available in the development environment for informal and unit testing.

SSTF does not have target compilation or target hardware duplicated in the development environment. TSC plans to use the IVTE facilities for informal and unit-level hardware-dependent testing as well as for formal testing and integration. The Amdahl acts as a CM repository for unit-tested versions of code, but the CM repository is not the sole source for target compilation in the IVTE, and target compilation in the IVTE is available as part of a less-formal process during development. Target compilation for deliveries will, of course, remain a controlled and managed process.

2. Timing of information request in light of current design phase

Much of the information requested in this document is at a level of detail much deeper than TSC's current design level (SFDR). Some of the higher-level decisions that will drive these lower-level decisions are still in flux, so we cannot provide authoritative answers to these questions.

The simulation facility envisioned is of a significantly greater complexity than most that have been built or operated in the past, by Link or by anyone. The simple fact is that no one — Link, NASA, CSC, or anywhere — has the base of experience to authoritatively and fully define the process of using this facility, let alone define exactly how that process should be automated. Link will develop and provide automated processes in later deliveries that will encapsulate what is learned about running the facility in earlier deliveries. Because we do expect to learn, we are choosing to reduce our up-front automation of processes that are bound to change.

3. Current Link operational scenarios

The following sections discuss our current operational scenarios. They are stated in terms of Figure 3-1, a functional diagram of the SSTF network connectivity. Specific details of network connections (routers, T1 lines, ethernet vs. other networks, etc) are left out of this functional diagram.

These scenarios assume that access to the IVTE hardware (now the STE hardware being used for IVT) from software development workstations is limited to those workstations that physically reside in building 5.

page 3

GSDE Interface Rqts Analysis/Scenarios — TSC/Link comments

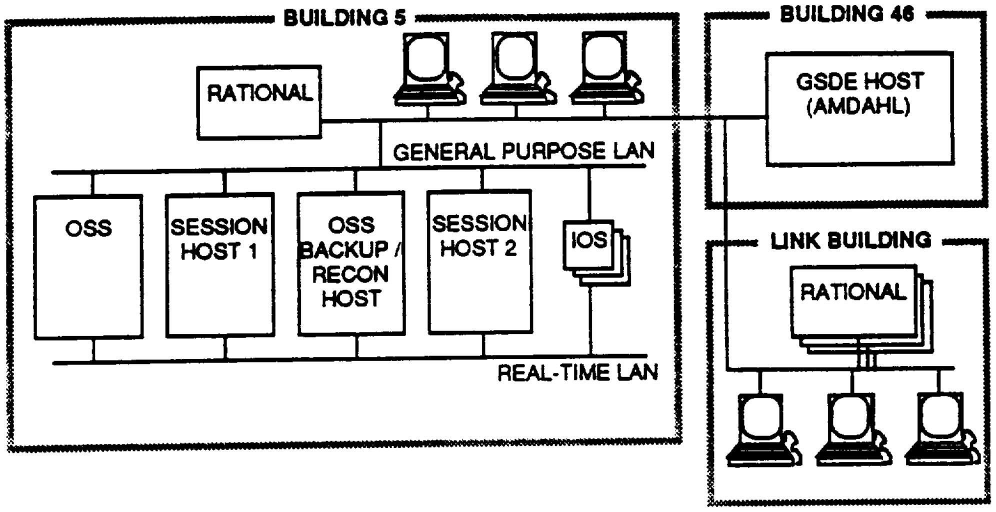


Figure 3-1. Functional diagram of SSTF development network connectivity

3.1 Development of new code

Requirements analysis and preliminary design are done on the software development workstations in the Link building. Data is managed with local CM tools.

Ada detailed design, implementation, and non-real-time testing and integration are done on Rationals in the Link building, accessed from the software development workstations in the Link building. Non-Ada detailed design, implementation, and non-real-time testing and integration are done on the software development workstations in the Link building. Data is managed with local CM tools.

Non-real-time testing is testing that does not depend on the target hardware, or on target environment timing characteristics. In some cases, non-real-time testing is sufficient for code delivery.

At delivery, data items are copied to the SSE CM system on the GSDE Host. Source code is entered into the SSE CM system after the developer has tested it to his/her own satisfaction, but prior to formal testing. Formal testing is always performed on controlled copies of code checked out of the SSE CM system.

If real-time testing is needed, the developer will perform informal real-time testing to ensure that the code is ready for formal real-time unit test.

Real-time testing prior to formal unit test is done as follows:

- The code is networked to the integration Rational (the Rational in Building 5).

GSDE Interface Rqts Analysis/Scenarios — TSC/Link comments

- From any software development workstation, the Reconfiguration system is invoked to compile the code from the integration Rational and create a loadable image. This loadable image is flagged in the operational CM as an unverified image.
- From a software development workstation in building 5, a user with privileges for real-time testing loads the unverified image into a set of simulator hardware that is not scheduled for training use (this scheduling may be subject to automated verification). The software is informally tested. The unverified image in the simulation hardware is discarded.
- From any software development workstation, the user can discard the unverified image from the Reconfiguration system.

Formal real-time testing is done as follows:

- The code will have been checked into the SSE CM system as part of submission for formal testing. It is checked from the SSE CM system for testing, and networked to the integration Rational along with any test objects provided in the SSE CM system (e.g., scripts, data files).
- From any software development workstation, the Reconfiguration system is invoked to compile the code from the integration Rational and create a loadable image. This loadable image is flagged in the operational CM as an unverified image.
- From a software development workstation in building 5, a user with privileges for real-time testing loads the unverified image into a set of simulator hardware that is not scheduled for training use (this scheduling may be subject to automated verification). The software is put through formal testing. The unverified image in the simulation hardware is discarded. The results of formal testing are networked to the GSDE Host. This may include a copy of the compiled image.
- From any software development workstation, the user can discard the unverified image from the Reconfiguration system.
- From any software development workstation, The results of formal testing are checked into the SSE CM system on the GSDE Host. If all testing has been succesfully completed, the status of the object in the SSE CM system is entered into the acceptance process.
- Once the results have been examined and verified by the cognizant authority, the status of the object in the SSE CM system is upgraded from any software development workstation.

Once formal testing is completed and the source code is considered verified, the system is target-compiled and integrated. This is done from a software development workstation in building 5, by a user with privileges for verified load building, using the Reconfiguration system. If the compiled version was copied into the SSE CM system as a testing output, it may be checked out instead of being recompiled. The linked and verified load is networked back to the GSDE Host. It is checked into the SSE CM system from any software development workstation by a user with privilege to do so.

If errors are found in the source code during formal testing or integration, they are not handled in the target environment; rather, the code is copied back into the development environment, or an existing copy in the development environment is used and then submitted to SSE CM as a new version. Minute changes may be done using GSDE Host-based editing tools, but most if not all code changes are expected to be done on Rationals or software development workstations. Note that the workstations in building 5 are complete software development workstations, and editing may be done there.

The SSE CM system cannot be the only conduit to the target systems without creating a severe burden in informal real-time test, where many versions are expected to be produced, tested, and superceded.

3.2 Maintainence of existing code

If the data products to be edited (requirements documents, design documents, code, data files, etc.) still exists in a software editing environment (on a Rational or workstation), that copy may be edited, tested, and submitted as a new version. Otherwise, the code is checked out of the SSE CM system and edited on a Rational (Ada) or a workstation (non-Ada).

4 Response to assumptions and questions

Table 4-1 shows our response to the specific assumptions, issues and questions listed in section 4.6 (pages 35-39) of the CSC document. Note that many of these assumptions and questions are irrelevant because of the questionable overall assumptions.

GSDE Interface Rqts Analysis/Scenarios — TSC/Link comments

Table 4-1. Responses to assumptions and questions

Sect / item	Item (assumption, question, etc.)	Response
4.6.1	**Assumptions**	
1	IVTE mainframe will act as IVTE host and buffer interface with Amdahl.	True; but interface with Amdahl is not the whole story. This mainframe will interface with integration Rational and workstations; other target environment machines will also interface with these.
2	SSE CM-access tools used in SPE.	True.
3	IVTE platforms will host "remote batch" capability.	Lower-level design decision. Some processes may be accomplished via remote login instead.
4	Code development through unit test occurs in SPE.	Only true for software that does not require target hardware for unit test.
5	IVTE platforms used for some compilation and load building.	True; target platforms are used for all target compilation and load building, controlled by Reconfiguration system.
6	Some duplicate compilation and test products stored in IVTE.	Lower-level design decision.
7	Duplicate storage in IVTE will be tracked from Amdahl.	If there is duplicate storage, CM facilities local to it will be used. These will be coordinated with the Amdahl CM, probably procedurally. This will be under the control of the Reconfiguration system.
8	Processing transactions over Amdahl-IVTE interface will have unique IDs.	Lower-level design decision. Further, assumes automated interface instead of a developer using remote log-in facilities.
9	SSE file verification facilities will be available on Amdahl and IVTE.	SSE is not committed to delivering file verification facilities on the unidentified OADP architecture.
10	Testbed definitions are controlled on the Amdahl.	True for formal testing; provided by SSE CM toolset. Reconfiguration system will provide the ability to define configurations. Ops CM in IVTE will control the current configuration in the target environment.
11	All software and data for testing are in controlled storage on Amdahl.	True for formal testing; provided by SSE CM toolset.
12	Mechanism exists for recording that a configuration item is "in use" in a testbed.	Probably true; depends on SSE CM toolset. Note that software configuration items may be included in many configurations at one time.
13	Scheduling IVTE resources is not part of Amdahl-IVTE interface.	True.
14	DR process is not part of Amdahl-IVTE interface.	True.
15	Testing user uses a workstation located in IVTE but networked only to Amdahl.	False.
16	Execution of a test session can occur logically disconnected from Amdahl.	True. However, not disconnected from Software Development System workstations.
4.6.2	**Data flows**	
1	Configuration items Amdahl->IVTE	True; however, not the only conduit from SPE to IVTE.
2	Configuration data Amdahl-> IVTE	Lower-level design decision.
3	File transfer IVTE->Amdahl	True. Configuration items.

page 7

GSDE Interface Rqts Analysis/Scenarios — TSC/Link comments

Table 4-1. Responses to assumptions and questions

Sect / item	Item (assumption, question, etc.)	Response
4	Status of processing, IVTE->Amdahl.	Probably false.
5	Process outcome reports, IVTE ->Amdahl.	False. From workstations.to CM on Amdahl.
6	Test outputs and new config items IVTE -> Amdahl.	True. Also from Rational and workstations to Amdahl.
7	Configuration descriptive info IVTE -> Amdahl.	Probably from workstations to Amdahl.
4.6.3	**Interface functionality**	
1	Bidirectional file transfer including format translation, confirmation, directory inquiries	No format translation. Probably no directory inquiries, since user can directly log to Amdahl and IVTE mainframe from workstation; however, this may be available.
2	IVTE execute Amdahl-generated command scripts.	False.
3	IVTE host able to format CM records and reports.	Lower-level design decision; however, this will probably be handled by the workstations, either as a remote standalone tool (as suggested here for IVTE) or by logging on to Amdahl.
4	Amdahl able to process IVTE-provided CM records and reports.	Lower-level design decision; however, probably not.
5	Record and recall session data.	Automated non-interactive sessions not anticipated in design.
4.6.4	**Interface questions**	
1	Amdahl-IVTE transactions will require a command language.	No such automated transactions (except file transfer) anticipated. A file transfer protocol such as FTP will be used.
2	Requirements for IVTE command language(s)	COTS operating system will be used.
3	Amdahl-IVTE command procedure processing questions.	No such procedures.
4	Identification of CM-controlled items while in IVTE.	Lower-level design decision; IVTE CM system and its interaction with SSE CM toolset not yet defined. Manual process is likely.
5	What level of control will be exercised over files and command scripts downloaded to IVTE?	These will be created on IVTE platform, downloaded from integration Rational, or downloaded from workstation. User is expected to control appropriately.
	Will such files be under [SSE] CM?	Not until formal testing.
6	Can command scripts be tailored without formal CM?	Yes, until formal testing.
7	What mechanism will be used to tailor command scripts?	Lower-level design decision. Probably a workstation tool.
8	How will transaction identifiers be tracked?	No automated non-interactive transactions as envisioned in this document. Native OS remote job entry and remote log-in facilities will be used.
9	Where in transaction process is CM data gathered?	This transaction process does not exist for SSTF.
10	Which computers can initiate a transaction?	Transactions as envisioned do not exist.

page 8

GSDE Interface Rqts Analysis/Scenarios — TSC/Link comments

Table 4-1. Responses to assumptions and questions

Sect / item	Item (assumption, question, etc.)	Response
11	Can Amdahl pre-verify existence in the IVTE of items required in a testbed?	Probably. How is a lower-level design decision. Procedural techniques are another alternative.
12	Acknowledgement mechanism from IVTE to SPE user.	Actions are performed by remote login.
13	How will scripts and instructions be provided to testers?	For formal testing, stored in SSE CM; can either be examined on the Amdahl (via a remote login) or downloaded to workstation.
14	How will test status reports be returned to Amdahl?	Lower-level design decision affecting workstation-Amdahl interface. Probably directly entered into SSE CM tool.
15	Assembly of resources during remote IVTE transactions.	No such transactions.

Appendix E - MSC Software Development Briefing

The material on the following pages was provided by Mission Systems Contract personnel in response to the operational scenarios developed during this study.

GROUND SYSTEMS DEVELOPMENT ENVIRONMENT (GSDE)

SOFTWARE DEVELOPMENT SCENARIOS

April 2, 1991

MSC Loral Team

Preface

The following Software Development Scenarios were developed to describe the processes involved in using the Ground Systems Development Environment (GSDE) system to develop software and data products for the Space Station Control Center (SSCC)

This package is intended to convey high-level development procedures in a step-by-step fashion to aquaint the reader with both the equipment in the GSDE, and the interaction among different MSC organizations involved in the development process. These scenarios are still under revision and are not yet in final form. This package is simply a record of the concepts and agreements at this point in time.

In these scenarios, the term *Developer* indicates an MSC contractor or sub-contractor while the term *User* indicates either an OSC or NASA employee. *I&T* represents the MSC Integration and Test organization, *SE&A* is the MSC Subsystem Engineering and Acquisition organization, and *QA* is the MSC Quality Assurance orrganization.

Comment and suggestions on these scenarios are welcome and should be sent to:

David P. Sundermeyer
Loral Space Information Systems
1322 Space Park Drive M/S F851A
Houston, TX 77058
(713) 335-6676

GSDE SOFTWARE DEVELOPMENT SCENARIOS

This package contains operational scenarios for the following activities:

- Workstation Software Development
  - Development through acceptance test (AT)
  - Software promotion
  - Development between AT & QT
  - Qualification testing (QT)
  - System integration test (SIT)
- OADP Software Development
  - Development through acceptance test (AT)
  - Software promotion
  - Development between AT & QT
  - Qualification testing (QT)
  - System integration test (SIT)
- Display Builder Software Development
- Computation (COMP) Builder Software Development
- TCATS Software Model Analysis

Workstation Software Development Scenarios

Workstation S/W Development Through Acceptance Test (AT)

(1) Reusable libraries are first searched for potential software packages that may satisfy some or all of the requirements of the software module to be developed. Reusable library packages are copied from the GS/SPF and placed on a development CM server. Any modifications are accomplished on the GSDE Development W/S.

(2) The developer identifies resources on the GS/SPF Host that are needed for compilation and testing and request a copy to be sent to the local development CM server.

(3) New source code is entered on the Development W/S and stored on a development CM server.

(4) Source code is compiled and debugged on the Development W/S for such things as syntax checking, library identification, and basic stand-alone functionality.

(5) After initial testing, the developer defines a load build script identifying all resources needed for checkout of W/S code for a specific target W/S.

(6) The script is executed with the local development CM server sending the source code, compilation instructions, and any other identified data files needed to compile the source code to the target W/S compile server.

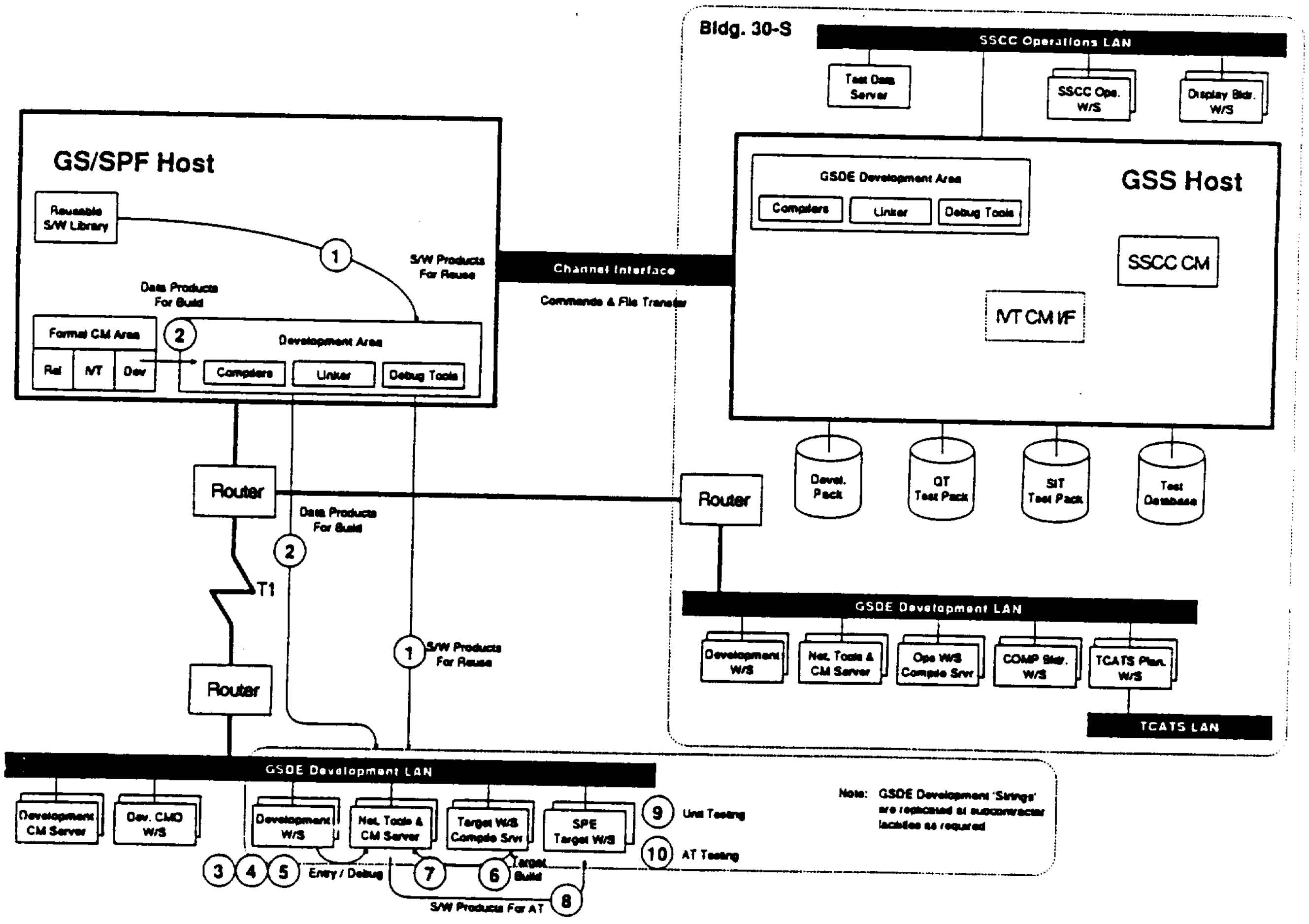
(7) After compiling the source code, the W/S compile server links the object modules into an executable S/W load module for the appropriate target W/S. The object code and executables are returned to the local development CM server.

(8) The software developer logs onto the SPE target W/S and requests a download of the executables for initial hardware/software integrated unit testing.

(9) The developer performs initial hardware/software integrated unit testing on the SPE Target W/S using the workstation's stubs and drivers. These stubs and drivers interact with other GSDE resources to simulate as much of the SSCC services as possible

(10) After integrated unit testing is completed, an Acceptance Test is performed on the SPE Target W/S with SE&A, I&T, and QA in attendance.

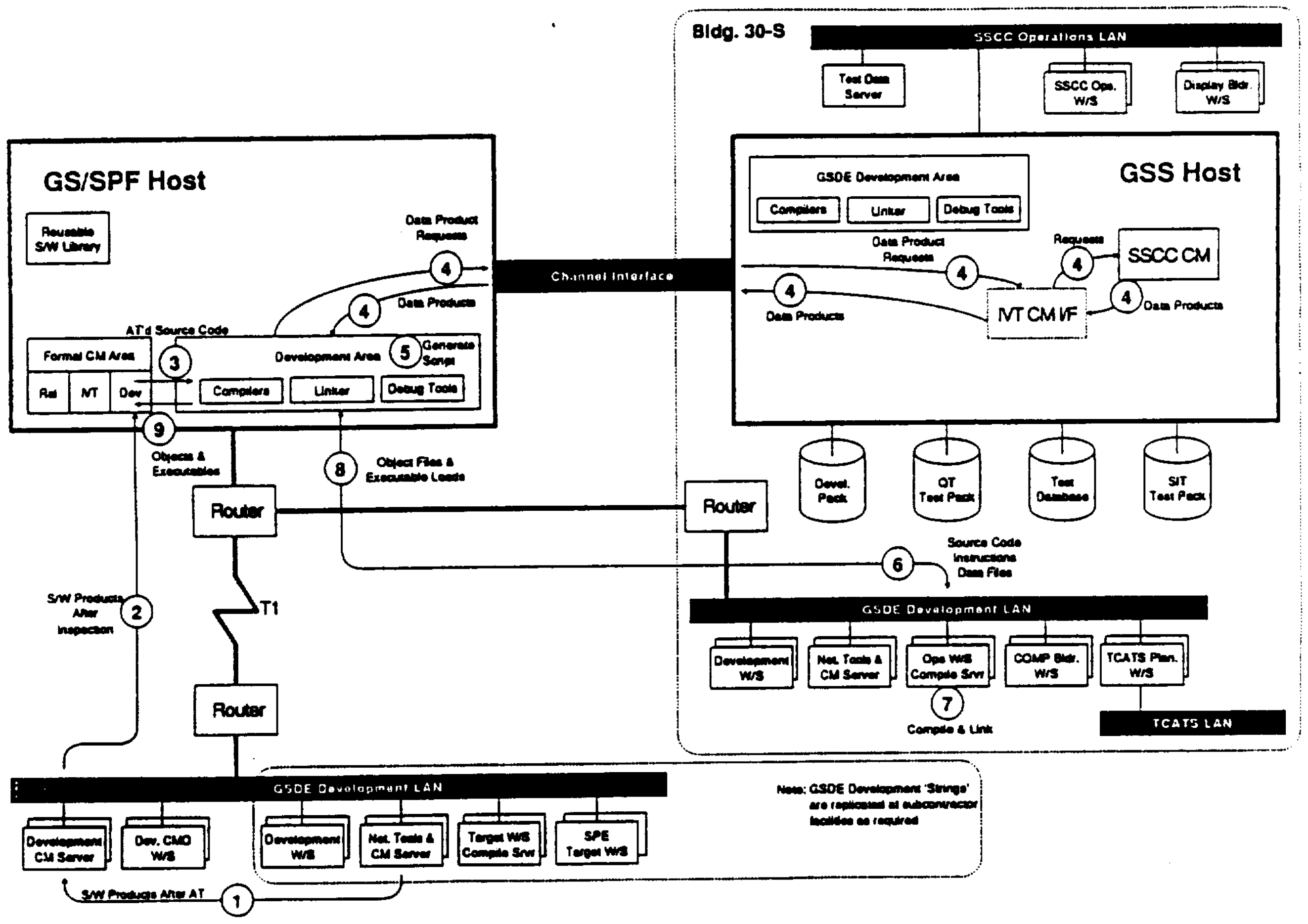
NOTE: Workstation software AT could also occur on a Development Workstation if desired.



Workstation S/W Development Through AT

Workstation Software Promotion

1. After the AT review, the assigned Configuration Management Officer (CMO) transfers the software products to the master development CM server.
2. The CMO inspects the S/W products and performs a CHECK-IN to the Development portion of the GS/SPF Formal CM area. Software products consist of source code, object code, executables, test plans, test procedures, test results, test data products, and all other documentation required for traceability and manageability.
3. Software source code is copied from the GS/SPF Formal CM Area to the software Development area on the GS/SPF Host.
4. Any products required for the Ops W/S that reside in the SSCC are requested from the GSS Host CM. These products may consist of Test RECON products, special tables, etc.
5. A build script is created containing a list of all required software products (with version numbers) required to build an executable load for an SSCC Operational Workstation (Ops W/S).
6. The source code, compilation instructions and any required data products are sent from the GS/SPF to the appropriate Ops W/S compile server in Building 30-S.
7. The source code is compiled and the new objects are then linked into an executable load module.
8. The new objects and executables are transferred to the GS/SPF software development area.
9. SE&A performs a CHECK-IN of the new objects and executables into the Development portion of the GS/SPF Formal CM area as the Ops W/S objects and executables. The original objects and executables from the AT are also maintained.



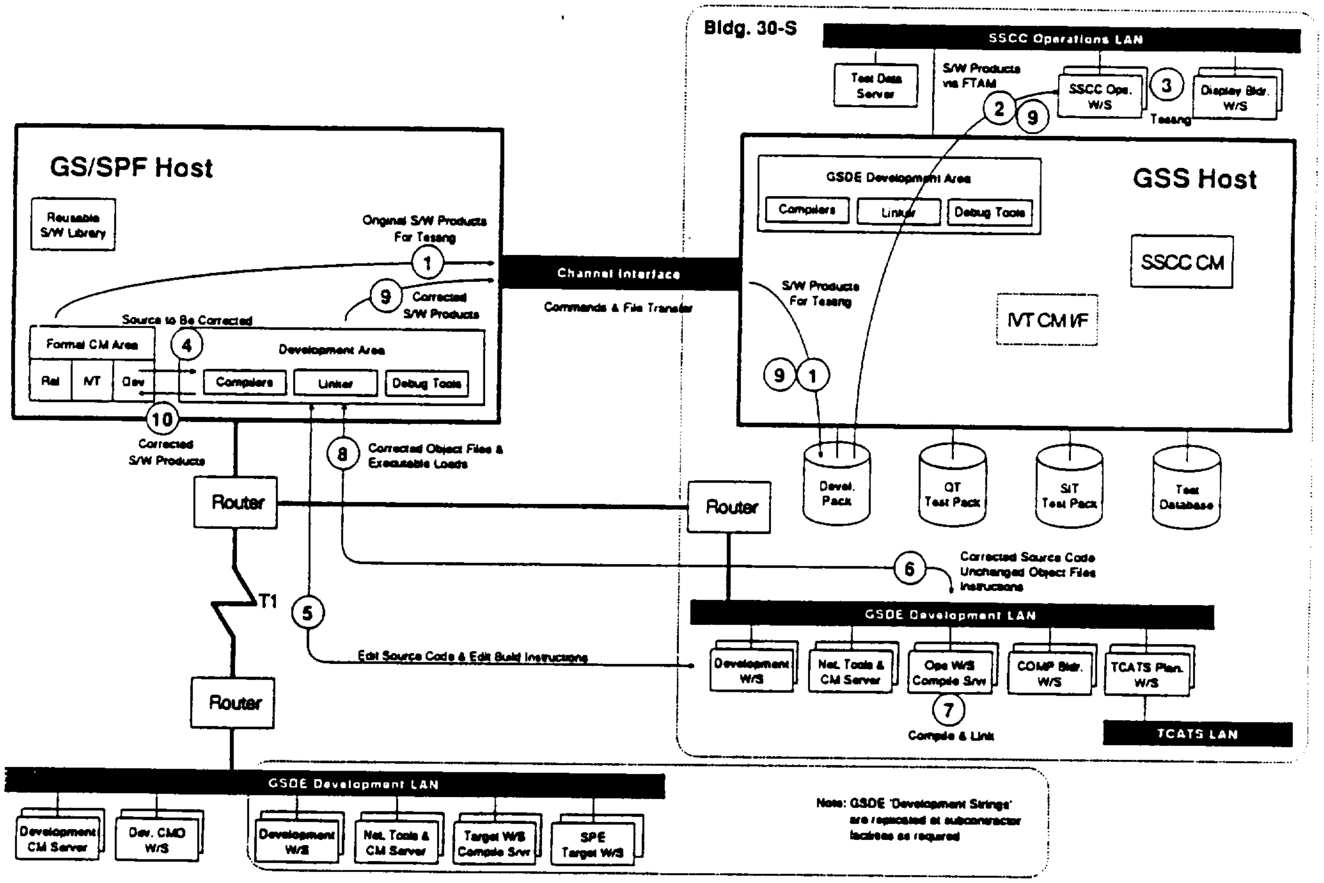
Workstation S/W Promotion

\projects\gsde\scn-wswp

March 12, 1991 -- D. Sundermeyer

Workstation S/W Development Between AT & QT

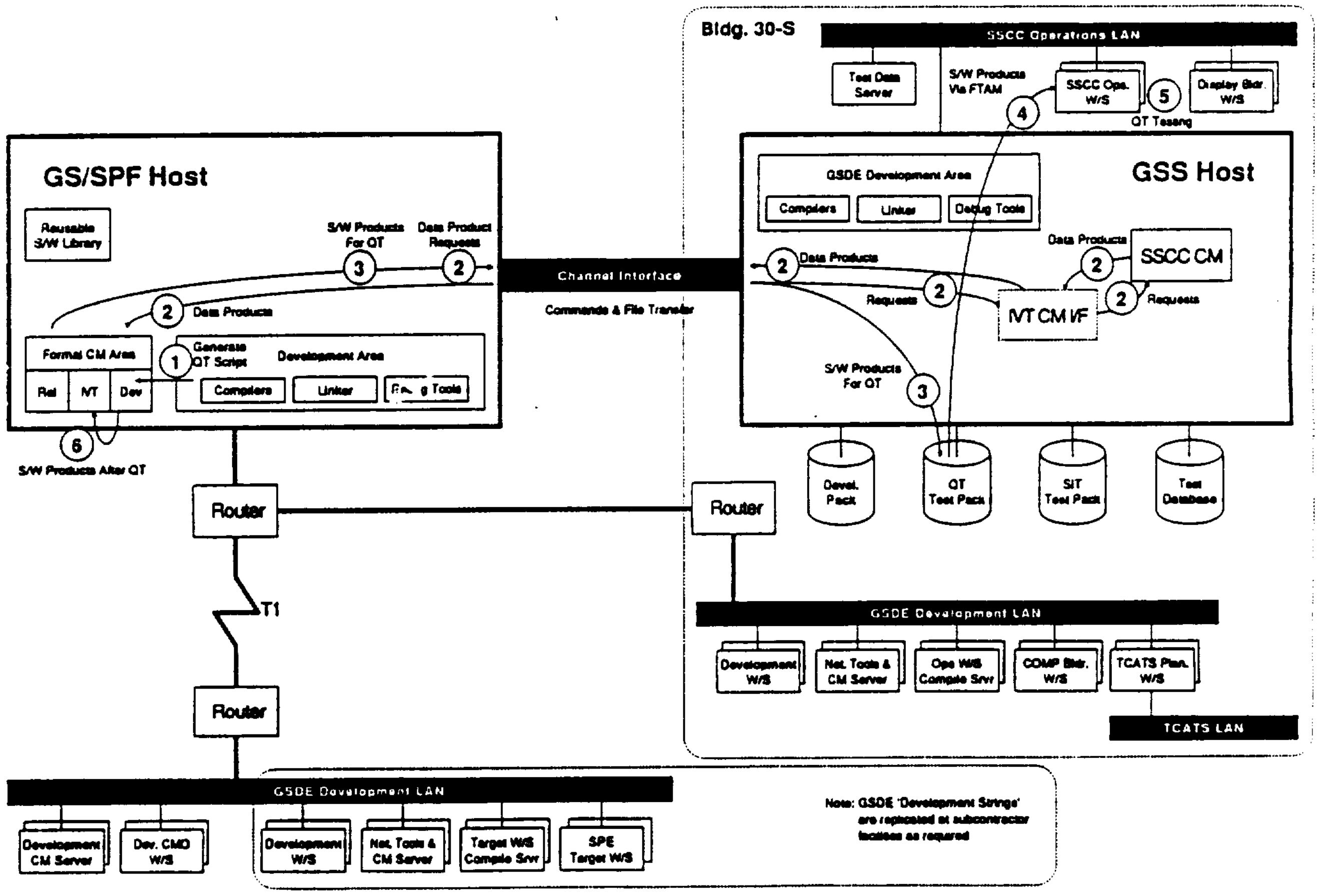
1. The Ops W/S executable, Test RECON products, and any other required data tables are downloaded from the GS/SPF Formal CM to a Development Pack on the GSS Host.
2. The executable W/S software products from the Development Pack are transferred to an Ops W/S via the SSCC Ops LAN using the File Transfer Access Methodology (FTAM) services available on the Ops network.
3. Testing is accomplished by the developer on the Ops W/S using a combination of stubs, drivers, and actual SSCC services and interfaces. If errors are detected, step 4 through 10 (below) are followed. Otherwise, Workstation QT testing is begun.
4. Any errors discovered will be documented with a Internal Discrepancy Report (IDR). The developer, using a GSDE Development W/S in the SSCC, will CHECK-OUT the appropriate file(s) from the GS/SPF Formal CM using the IDR number(s) as the authorizing change instrument(s). The source code is placed in the development area of the GS/SPF Host.
5. Corrections are made using the Development W/S's in Building 30-S. The developer will modify a copy of the build script that was created in the Workstation Software Promotion process.
6. The build script will be executed, sending the correct source code, unchanged object files, and modified compile instructions to the appropriate Ops W/S compile server.
7. The source code is recompiled and the new object linked to an executable load module.
8. The new object and executables are transferred to the GS/SPF development area.
9. The new executable is sent to the Ops W/S in the process described above and the product retest.
10. When the IDR is corrected, the source, object and executables are CHECKED-IN to the GS/SPF Formal Development area by the assigned CMO.



\projects\gsde\scn-wdqt **Workstation S/W Development Between AT & QT** March 12, 1991 -- D. Sundermeyer

Workstation S/W Qualification Testing

1. SE&A generates a QT build script specifying the executable loads and data products that are necessary to run the QT.
2. Any data products required for the QT that reside in the SSCC are requested from the GSS Host CM. They are CHECKED-IN to the GS/SPF Formal CM to maintain a permanent record of the data products used in the QT. These products may consist of Test RECON products, special tables, etc.
3. Products necessary for the QT are downloaded from the GS/SPF Formal CM area to the QT Test Pack.
4. The executable W/S software products from the QT Test Pack are transferred to an Ops W/S via the SSCC Ops LAN using the File Transfer Access Methodology (FTAM) services available on the Ops network.
5. QT testing is accomplished by SE&A on the Ops W/S using a combination of operational systems, Ops W/S stubs and drivers, core data, and test playback recorded data. Any errors discovered will be documented with a Discrepancy Report (DR). The MSC Sustaining Engineer will correct the DR's using the same process described in the Workstation Development From AT To QT scenario.
6. Upon successful completion of the QT, the software products are promoted to the IVT portion of the Formal CM area on the GS/SPF Host.



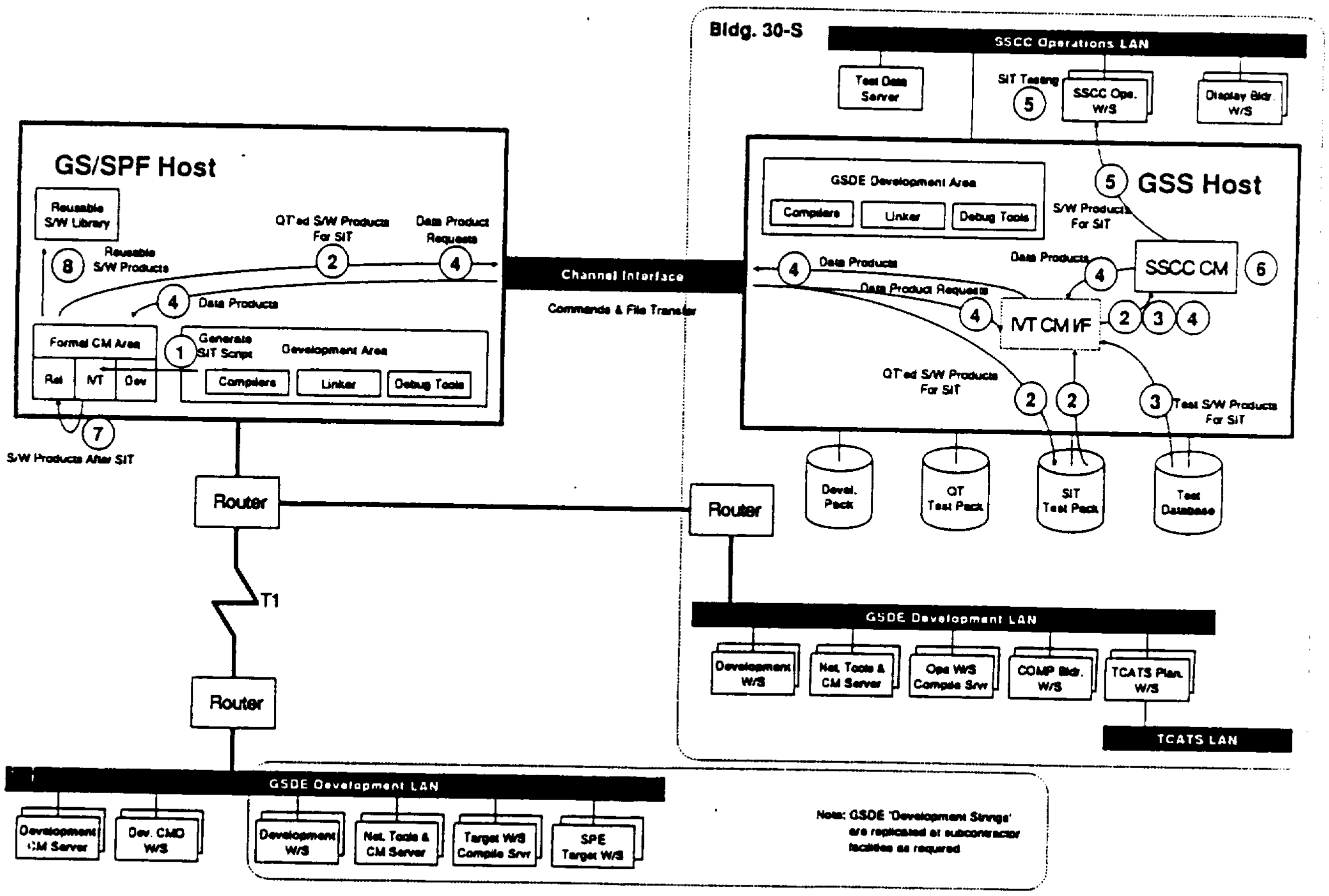
Workstation S/W Qualification Test (QT)

\projects\gsde\scn-wqt

March 12 1991 D. Sunderme[illegible]

Workstation S/W System Integration Test (SIT)

1. I&T generates a SIT script specifying the executable loads and data products that are necessary to run the SIT.
2. Upon successful completion of the QT, I&T downloads all QT'ed executables for the Ops W/S to an SIT Test Pack on the GSS Host and then uploads them to the SSCC CM using the IVT CM interface program on the GSS Host.
3. Test products residing on the Test Database are uploaded into the SSCC CM for testing.
4. I&T requests a copy of all test data from the SSCC CM needed for the SIT. The data products are CHECKED-IN GS/SPF Host formal CM to maintain a permanent record of all data products used in the SIT These data products will include the data transferred from the Test Database and data from the SSCC CM.
5. SIT testing is accomplished by I&T on the Ops W/S using real-time data and recorded data for inputs. Errors will be DR'ed for the Sustaining Engineering team to provide corrections. Critical DR's or major updates will require an RQT procedure when completed.
6. Upon successful completion of the SIT, the executables for the Ops W/S are promoted to the SIM level of SSCC CM by the Operations Support Contractor (OSC).
7. The software products in the IVT portion of the GS/SPF Formal CM Area are promoted to the Released portion of the GS/SPF Formal CM area by MSC.
8. If the new software products are determined to be reusable, the reusable aspects of the products are documented in the GSDE Reusable Software Library.



sscpec sigsJeleacn-wsit

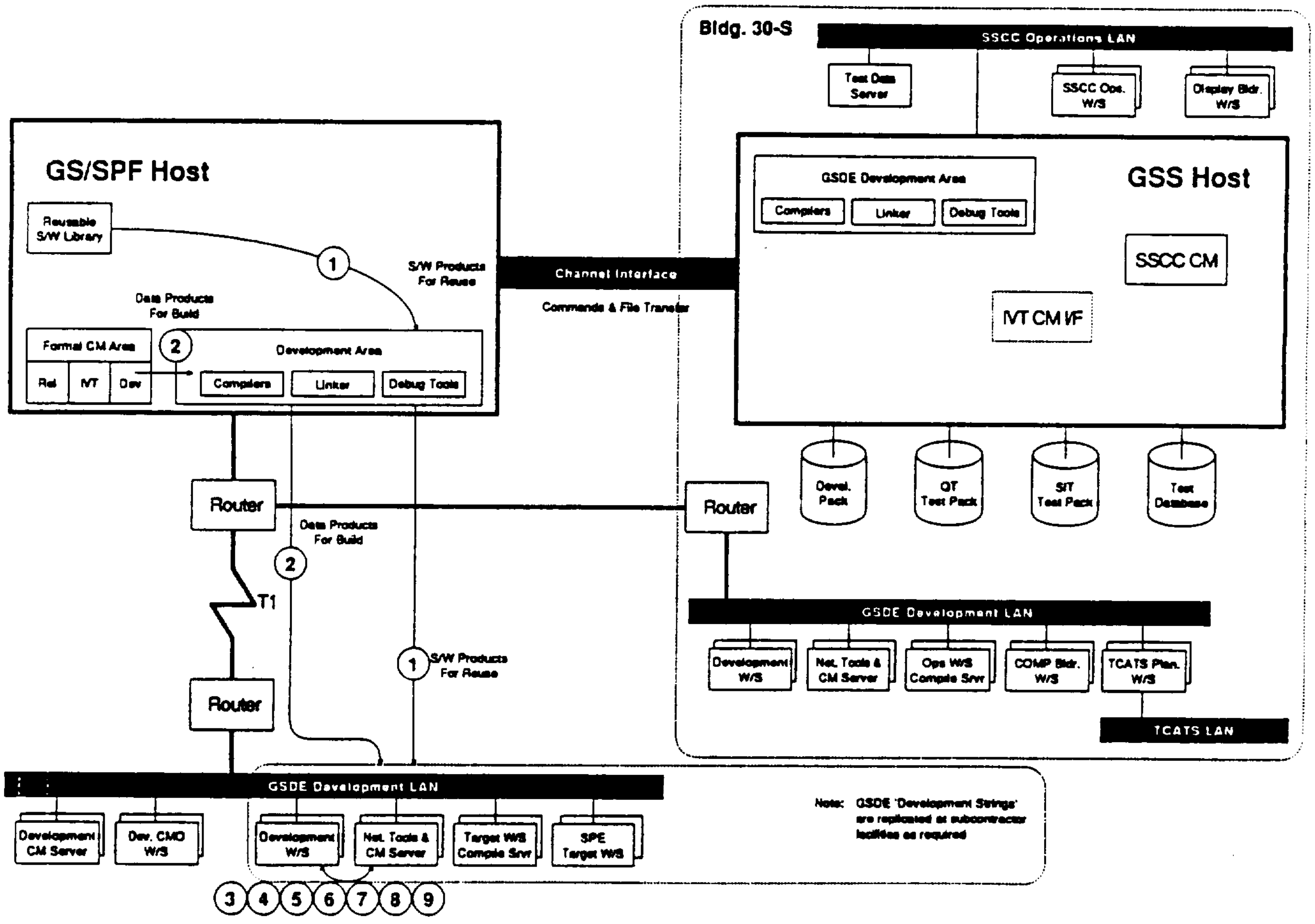
Workstation S/W System Integration Test (SIT)

March 12, 1991 D. Sundbergwsve

OADP Software Development Scenarios

OADP S/W Development Through Acceptance Test (AT)

1. Reusable libraries are first searched for potential software packages that may satisfy some or all of the requirements for the software module to be developed. Reusable library packages are copied from the GS/SPF and placed on the development CM Server. Any modifications are accomplished on the GSDE Development W/S.
2. The developer identifies any resources on the GS/SPF Host that are needed for compilation and testing and requests a copy to be sent to the local development CM server.
3. New source code is entered on the Development W/S and stored on a development CM server.
4. Source code is compiled and debugged with limited testing on the Development W/S for such things as syntax checking, library identification, and basic stand-alone functionality.
5. After initial testing, the developer defines a load build script identifying all resources needed for integrated checkout of OADP code on a Development W/S.
6. The script is executed with the local development CM server sending the source code, compilation instructions, and any other identified data files needed to compile the source code to the Development W/S
7. After compiling the source code, the Development W/S links the object modules into an e xecutable software load for the Development W/S. The object code and executables are returned to the local development CM server.
8. The software developer performs integrated unit testing on the Development W/S using OADP stubs and drivers. These stubs and drivers interact with other GSDE resources to simulate as much of the SSCC OADP services as possible.
9. After unit testing and Pre-Acceptance testing are completed, an Acceptance Test is performed on the Development W/S with SE&A, I&T, and QA in attendance.



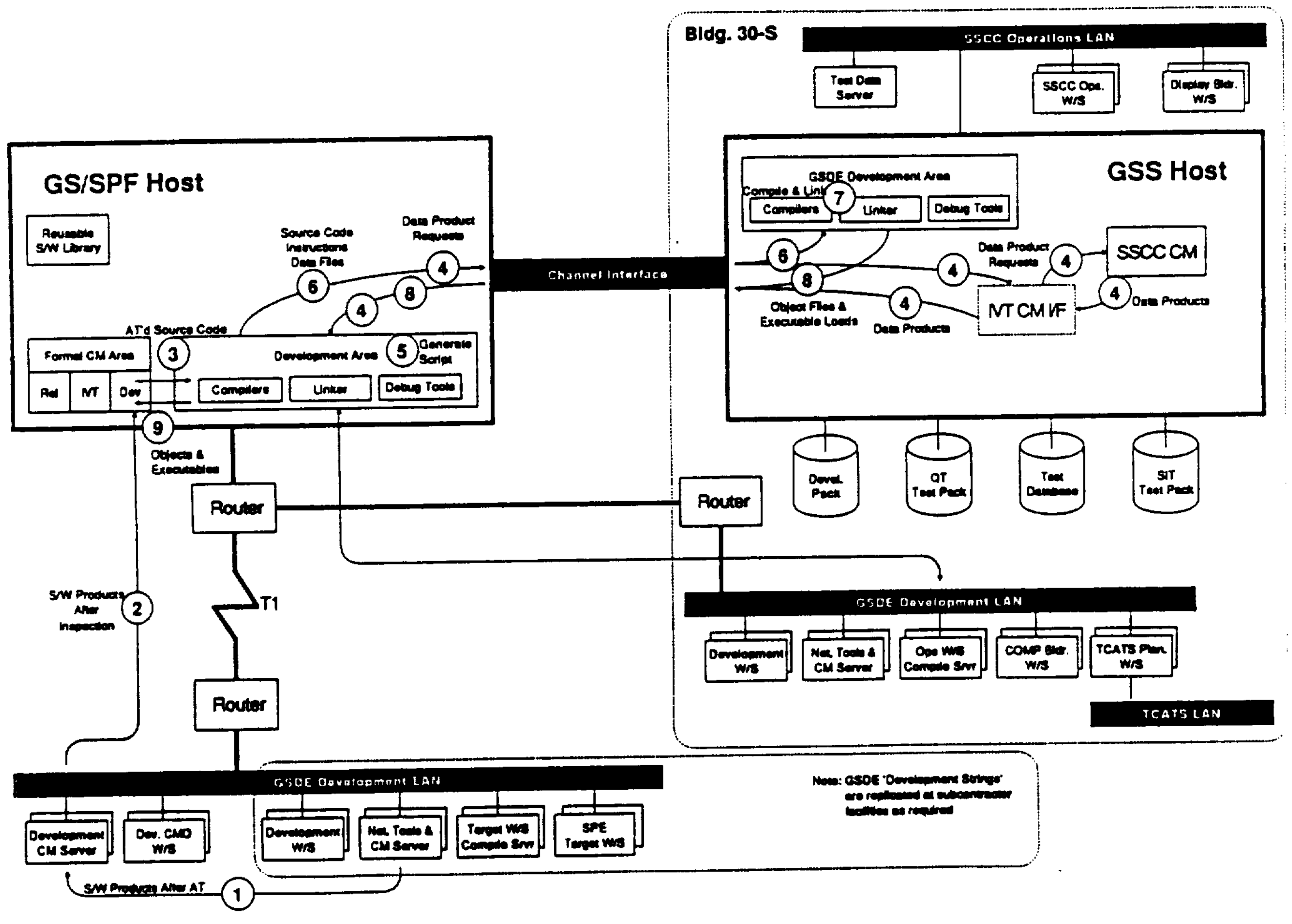
projects\gsde\scn-odat

OADP S/W Development Through AT

March 12, 1991 – O. Sundermeyer

OADP Software Promotion

1. After the AT review, the assigned Configuration Management Officer (CMO) transfers the software products to the master development CM server.
2. The CMO inspects the S/W products and performs a CHECK-in to the Development portion of the GS/SPF Formal CM Area. Software products consist of source code, object code, executables, test plans, test procedures, test results, test data products, and all other documentation required for traceability and manageability.
3. Software source code is copied from the GS/SPF Formal CM Area to the software development area on the GS/SPF Host.
4. Any products required for the OADP machine that reside in the SSCC are requested from the GSS Host CM. These products may consist of Test RECON products, special tables, etc, and are placed in the GS/SPF Host for CM.
5. A build script is created containing a list of all required software products (with version numbers) required to build an executable load for an SSCC.OADP computer.
6. The source code, data products, and the compilation instructions are then sent to the GSS Host.
7. The source code is compiled and the new objects are then linked into an executable load module.
8. The new objects and executables are returned to the GS/SPF software development area..
9. SE&A performs a CHECK-IN of the new objects and executables into the Development portion of the GS/SPF Formal CM area as the OADP objects and executables. The original objects and executables from the AT are also maintained.



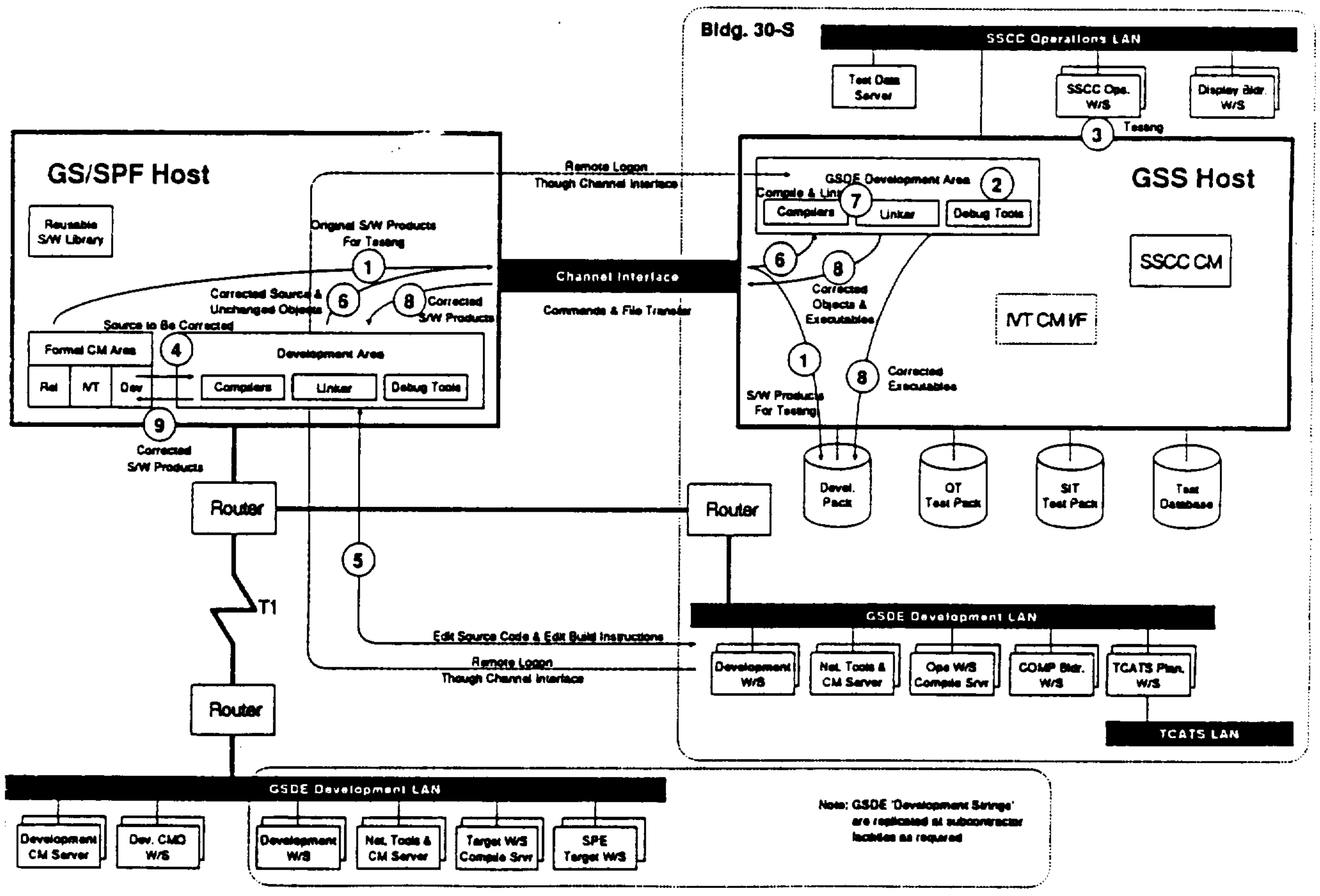
OADP S/W Promotion

March 12, 1991 – D. Sundermeyer

\projects\gsde\scn-oswp

OADP S/W Development Between AT & QT

1. The OADP executables, Test RECON products, and any other required data tables are downloaded from the GS/SPF Formal CM area to a Development Pack on the GSS Host.
2. The executable OADP software products from the Development Pack are used directly for testing on the GSS Host. If testing is desired on the Real-Time Host (RTH), the disk pack is physically transferred or logically switched to the RTH via disk farm controls.
3. Testing is accomplished using a combination of OADP stubs, drivers, and actual SSCC services and interfaces. If errors are detected, steps 4 through 9 are followed. Otherwise, OADP QT testing is begun.
4. Any errors discovered will be documented with a Internal Discrepancy Report (IDR). The developer, using a GSDE Development W/S in the SSCC, will CHECK-OUT the appropriate file(s) from the GS/SPF Formal CM using the IDR number(s) as the authorizing change instrument(s). The source code is placed in the development area of the GS/SPF Host.
5. Corrections are made using the Development W/S's in Building 30-S. The developer will modify a copy of the build script that was created in the OADP Software Promotion process.
6. The build script is then executed, sending the corrected source code, unchanged object files, and modified compilation instructions to the development area of the GSS Host.
7. The modified source code is compiled and the new objects are linked with the unchanged objects to create a new executable load module.
8. The new objects and executables are returned to the GS/SPF Host and the executables are transferred to the Development Pack and the software is retested.
9. When the IDR is corrected, the source, object and executables are CHECKED-IN to the GS/SPF Formal CM area by the assigned CMO.



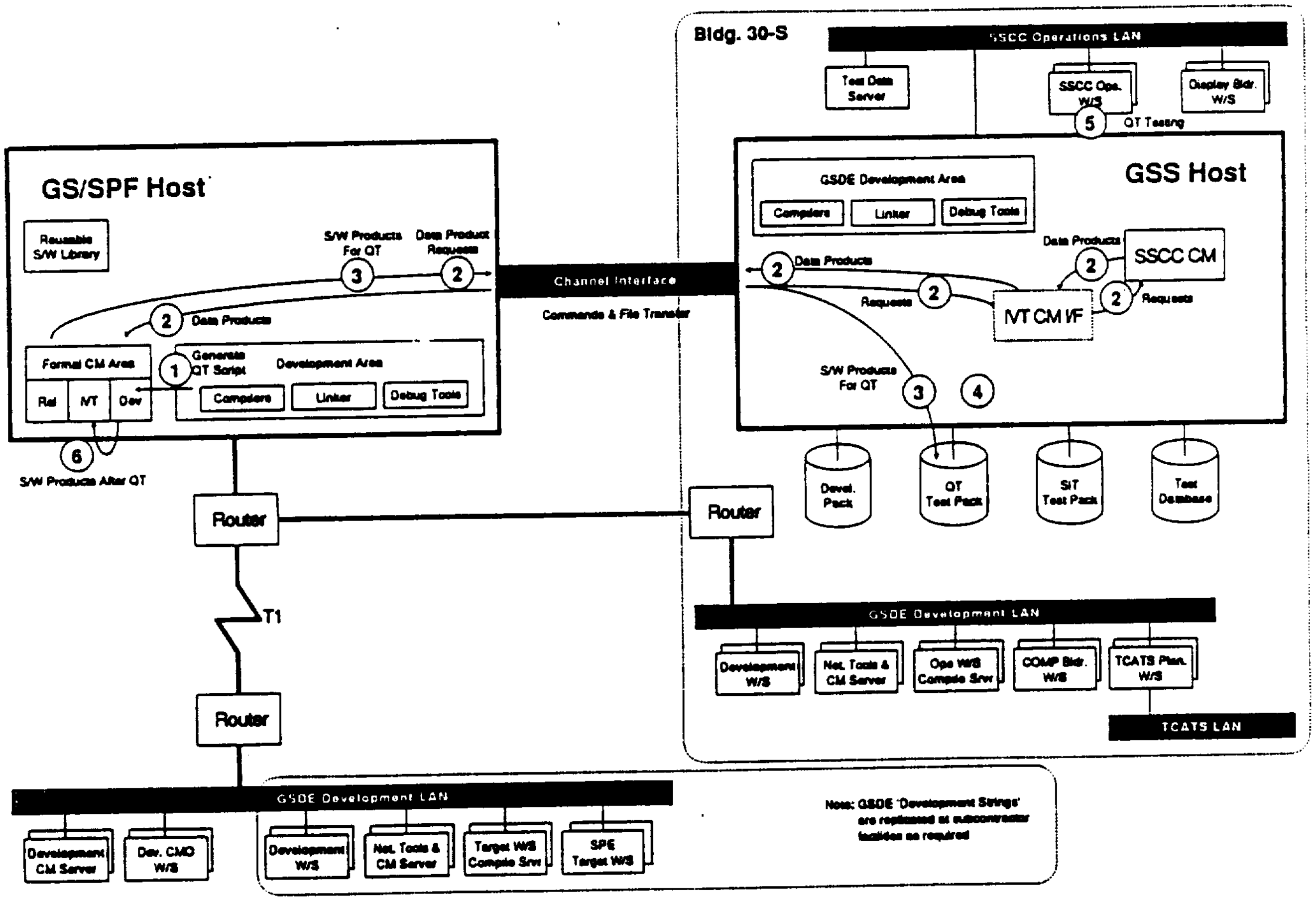
\projects\gsde\scn-odqt

OADP S/W Development Between AT & QT

March 12, 1991 -- D. Sundermeyer

OADP S/W Qualification Testing

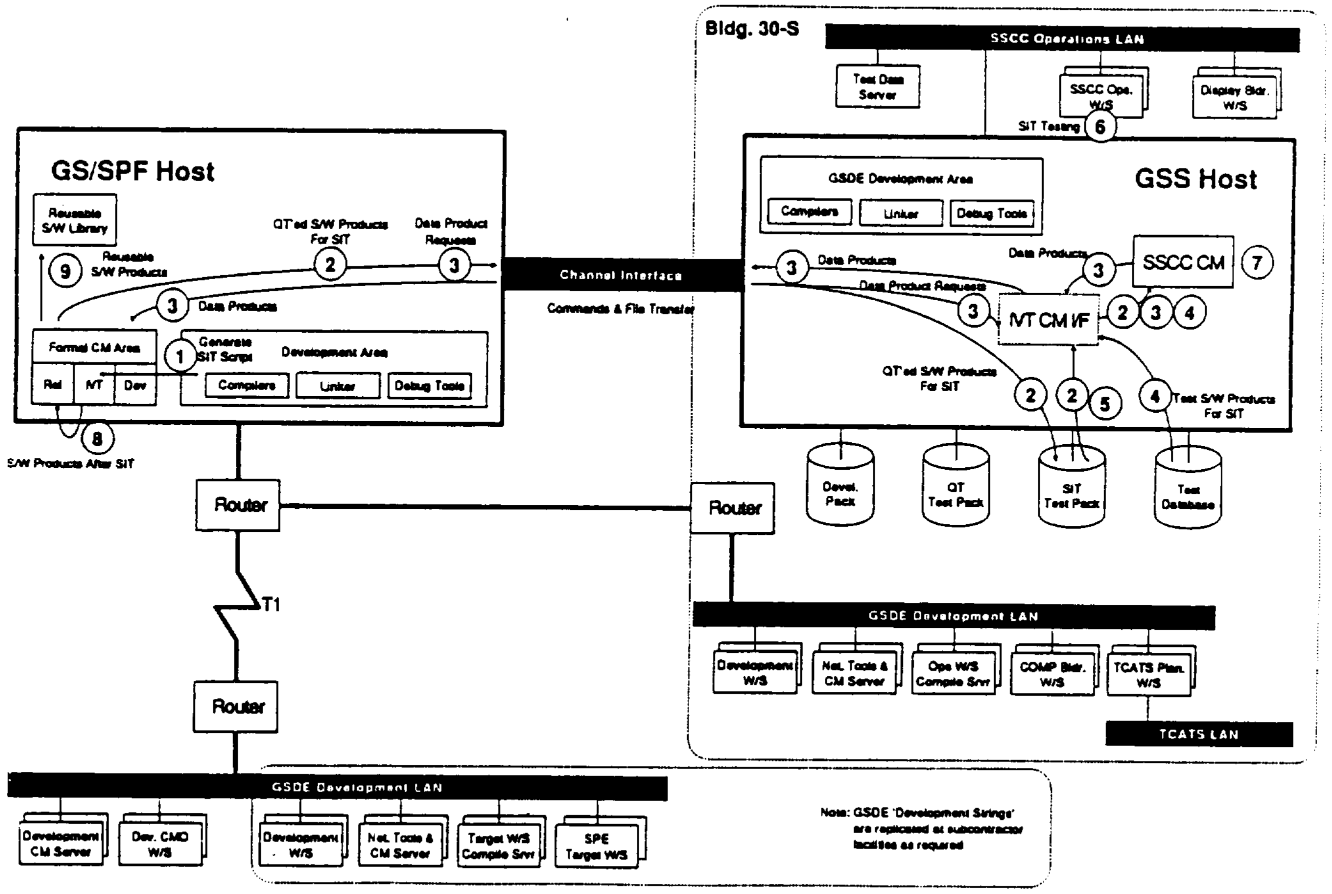
1. SE&A generates a QT build script specifying the executable loads and data products that are necessary to run the QT.
2. Any data products required for the QT that reside in the SSCC are requested from the GSS Host CM. They are CHECKED-IN to the GS/SPF Formal CM to maintain a permanent record of the data products used in the QT. These products may consist of Test RECON products, special tables, etc.
3. Products necessary for the QT are downloaded from the Development portion of the GS/SPF Formal CM area to the QT Test Pack.
4. The executable OADP software products from the QT Test Pack are used directly for the QT on the GSS Host. If testing is desired on the Real-Time Host (RTH), the disk pack is physically transferred or logically switched to the RTH via disk farm controls.
5. QT testing is accomplished by SE&A on the OADP machine or Ops W/S using a combination of operational systems, stubs and drivers, core data, and test playback recorded data. Any errors discovered will be documented with a Discrepancy Report (DR). MSC Sustaining Engineering will correct the DR's in the same method described in the OADP Development Between AT & QT scenario.
6. Upon successful completion of the QT, the software products in the QT portion of the GS/SPF Formal CM Area are promoted to the IVT portion of the GS/SPF Formal CM Area.



OADP S/W Qualification Test (QT)

OADP S/W System Integration Test (SIT)

1. I&T generates a SIT script specifying the executable loads and data products that are necessary to run the SIT.
2. Upon successful completion of the QT, I&T downloads all QT'ed executables to an SIT Test Pack on the GSS Host and then uploads them to the SSCC CM via the IVT CM interface program on the GSS Host.
3. Any data products required for the SIT that reside in the SSCC are requested from the GSS Host CM. These are CHECKED-IN to the GS/SPF Formal CM to maintain a permanent record of the data products used in the SIT. These products may consist of Test RECON products, special tables, etc.
4. Any products necessary for the SIT that are resident in the Test Database are uploaded into SSCC CM using the special IVT CM Interface program.
5. The executable OADP software products on the SIT Test Pack are used directly for the SIT on the GSS Host. If testing is desired on the Real-Time Host (RTH), the disk is physically transferred or logically switched to the RTH via disk farm controls.
6. SIT testing is accomplished by I&T on the OADP machine or Ops W/S using real-time data and recorded data for inputs. Errors will be documented with a DR for the Sustaining Engineering team to provide corrections. Critical DR's or major updates will require an RQT procedure when completed.
7. Upon successful completion of the SIT, the executables for the OADP are promoted to the SIM level of SSCC CM by OSC personnel.
8. The software products in the IVT portion of the GS/SPF Formal CM Area are promoted to the Released portion of the GS/SPF Formal CM Area by MSC.
9. If the new software products are determined to be reusable, the reusable aspects of the products are documented in the GSDE Reusable Software Library.

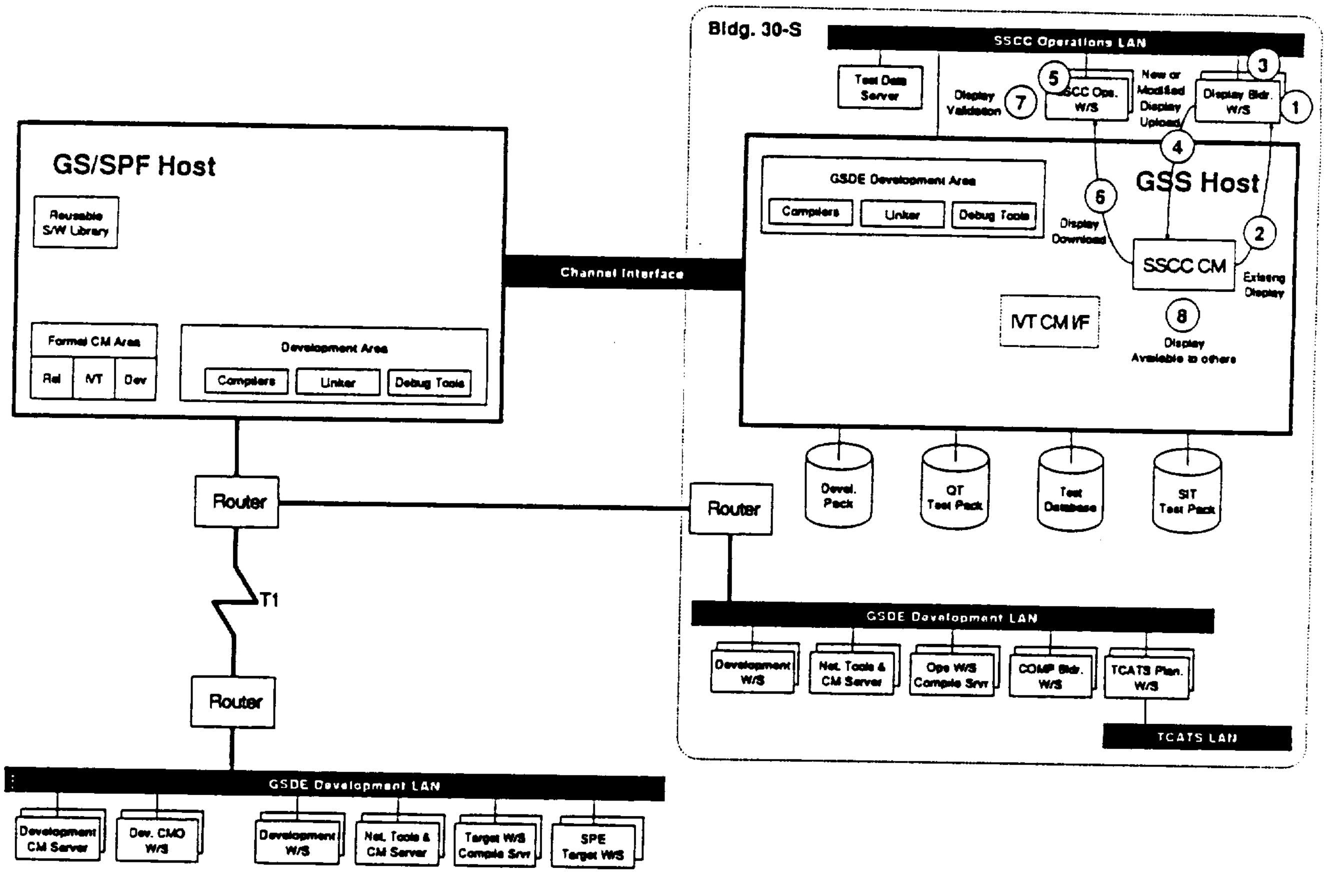


OADP S/W System Integration Test (SIT)

Display Builder Software Development Scenario

Display Build or Modification

1. A user logs on to a Display Builder W/S in the SSCC to develop new displays or to modify an existing display for use on the SSCC Ops W/S.
2. If a existing display is to be mo·.:ified, it is downloaded from the SSCC CM.
3. The user creates or edits an existing display file using the Display Builder application programs.
4. The new or modified display is uploaded inth the user's personal area of the SSCC CM.
5. The user then logs onto an SSCC Ops W/S and requests a download of the new display definition.
6. The display definition is downloaded from the SSCC CM to the requesting SSCC Ops W/S.
7. The new display is validated by the user with MSC's I&T personnel providing support and coordination assistance
8. The new display is now available for use by other users through SSCC CM



\projects\gsde\scn-disp

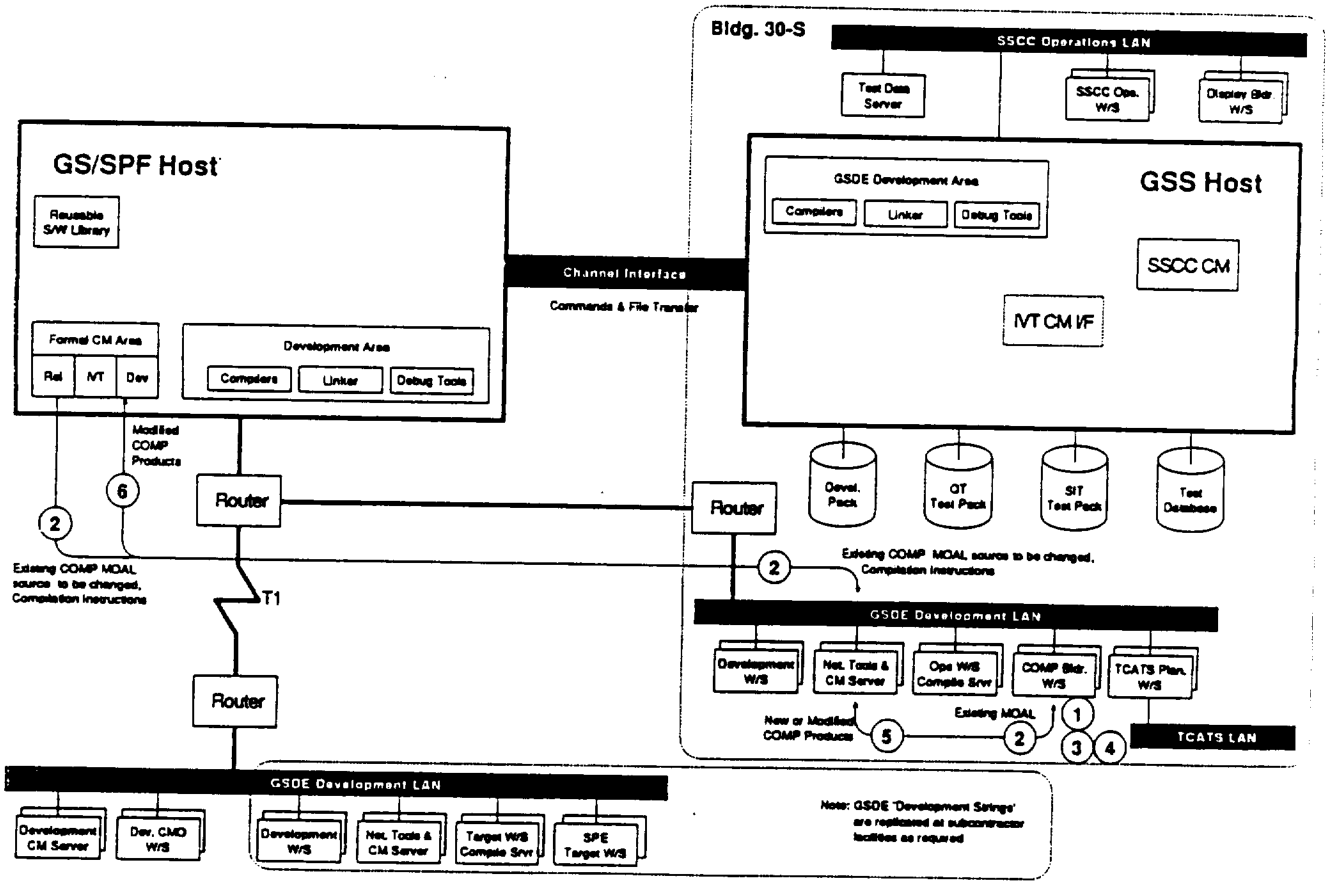
GSDE Display Build Scenario

March 12, 1991 – D. Sundermeyer

Computation (COMP) Builder Software Development

COMP Build or Modification

1. Developer logs on to a COMP Builder Workstation located in Building 30-S. COMP's are built by the COMP Builder application programs producing a "C" language source code product.
2. If an existing COMP is to be modified, the Mission Operation Application Language (MOAL) source file for that COMP is CHECKED-OUT from the Released portion of the GS/SPF Formal CM and placed on the development server in Building 30-S.
3. The developer either creates a new MOAL source file or modifies the existing one.
4. The COMP Builder application translates the MOAL into "C" source code which is then compiled on the COMP Builder W/S. Limited testing is performed on the COMP using available OADP stubs and drivers, and COMP Builder test tools.
5. When initial testing is complete, the assigned Configuration Management Officer (CMO) is notified that the products are ready for promotion to the GS/SPF Formal CM. The CMO transfers the product from the COMP Builder W/S to the development CM server.
6. The CMO then inspects the COMP's products and places them in the Development portion of the GS/SPF Formal CM area. The software products include the MOAL source code, "C" source code, object code, executables, and compilation instructions.
7. Normal OADP S/W developr^ent procedures are followed for S/W Promotion, Development Between AT & QT, Qualification Testing, and System Integration Testing. See the appropriate OADP Scenario for procedures.



\projects\gsdelscn-comp

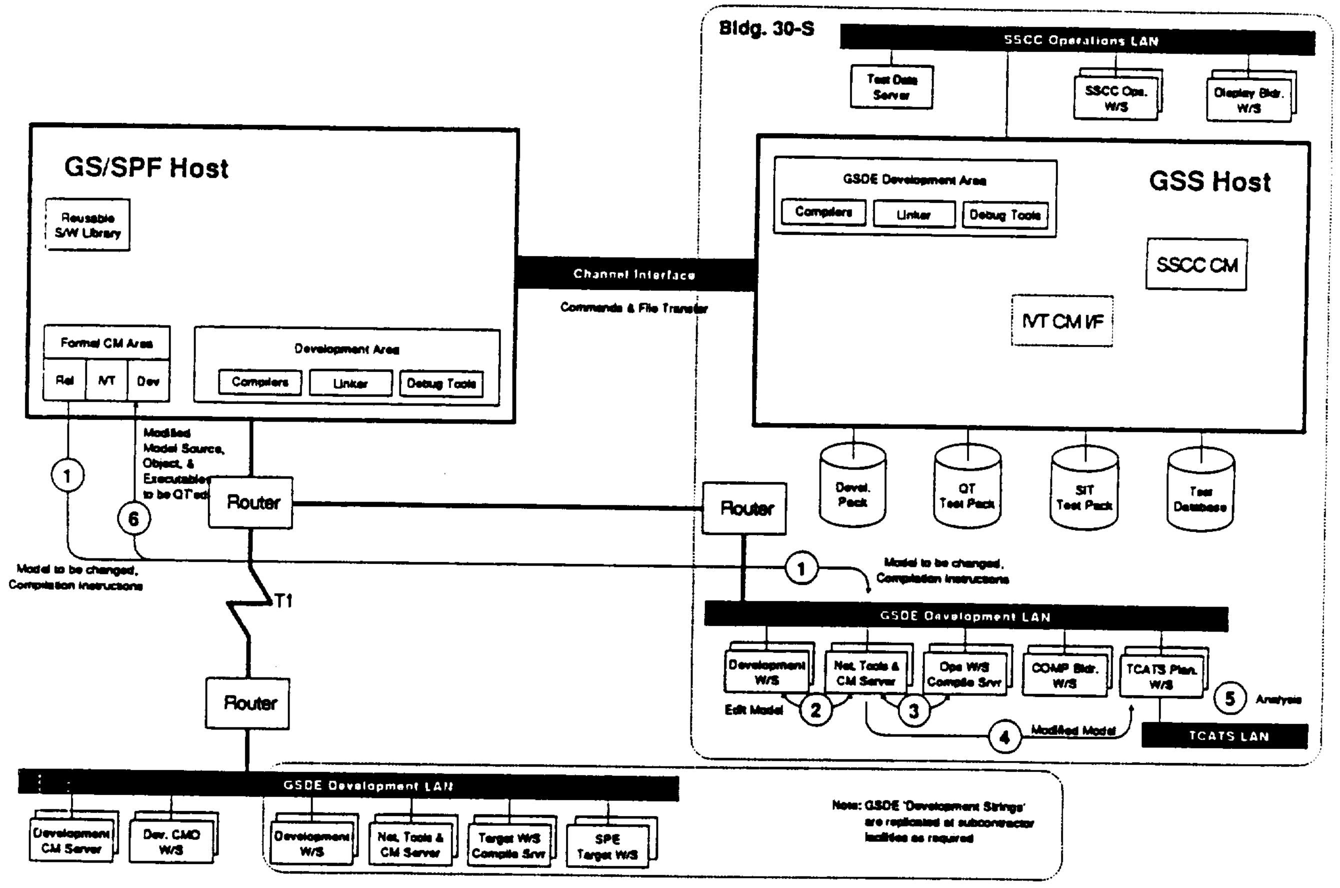
COMP Building Scenario

March 12, 1991 -- D. Sundermeyer

TCATS S/W Model Analysis

TCATS S/W Model Analysis

1. The TCATS model to be modified and its associated compilation instructions are copied from the Released portion of the GS/SPF Formal CM area to the development CM server in Building 30-S.
2. The model source code is modified and compiled on a Development W/S. Limited testing and debugging is performed on the Development W/S for such things as syntax checking, library identification and basic stand-alone functionality.
3. The source code and necessary compilation instructions are sent from the development CM server to the TCATS Ops W/S compile server. The source code is compiled and linked into an executable format. The new objects and executables are returned to the development CM server.
4. The developer requests a download of the new executables at a TCATS Planning W/S. The development CM server sends the requested files to the TCATS Planning W/S using the GSDE LAN.
5. The TCATS S/W developer executes the model on the TCATS Planning W/S using the TCATS LAN for data retrieval. The results of the test are collected and analyzed.
6. If the modified TCATS's model is to be baselined, the responsibility is given to the MSC development organization. The MSC contractor will assign the products to a Configuration Management Officer (CMO) who will inspect the products and take the necessary steps to perform a CHECK-IN of the products to the Development portion of the GS/SPF Formal CM area.
7. The product will then undergo normal Workstation S/W development procedures for QT and SIT as described in the Workstation S/W Development Scenarios.



TCATS Model Analysis

\projects\gsdelecn-lcal

March 12, 1991 -- D. Sundermeyer

Appendix F - Summary of Cronus Evaluation for the GSDE

F.1 Introduction

This report summarizes CSC's evaluation of the Cronus distributed application environment as a candidate to support the requirements of the distributed heterogeneous Ground Software Development Environment (GSDE). CSC installed Cronus on computers in our STAR*LAB computer research facility and began to assess Cronus in the context of GSDE requirements. A plan was developed for prototyping some elements of the GSDE on Cronus to provide specific, quantified answers to questions of applicability. At the direction of the customer, the evaluation of Cronus was suspended before the work was fully completed.

Before work was suspended, CSC did begin to assess the probable costs in using Cronus, but did not begin any GSDE-specific performance testing or application modeling. Details of the planned effort are provided in the *Requirements Analysis and Prototyping Plan* report. A further effort of two to six staff-months would be required to formulate preliminary answers to the questions raised in our investigation; the value of further effort would be determined by those preliminary findings.

This appendix reports on the work which was performed, and provides a partial assessment of Cronus based on that work, on other Cronus investigations underway at CSC, and on information (documents and discussions) provided by the developers of Cronus. Although not as detailed or specific as would be provided by the planned prototype effort, this assessment does provide a general picture of Cronus.

F.2 What is Cronus?

Cronus is an environment for developing and executing survivable distributed applications. The Cronus software development environment includes a comprehensive set of tools to assist in development of these applications. Cronus is layered on top of native operating systems (e.g., Unix and VMS) and provides value-added functions that simplify the construction of heterogeneous network-based applications. This layered approach allows application developers to select both native mechanisms (provided by the native OS) and remote resources (available via Cronus), as appropriate.

The architecture of Cronus supports the sharing of machine and device resources over networks, thereby allowing applications to be distributed over a wide collection of resources. The underlying Cronus system, and Cronus-based applications, are built using an object-oriented model. Using tools provided as part of Cronus, application developers design and implement classes of application-specific objects and the necessary network servers for those objects.

Operations on these objects can be invoked uniformly within a single machine or across machine boundaries. Functionality provided by Cronus handles issues of data translation between machines of different architectures, then uses low-level Cronus routines to move raw binary data from client to server via Cronus's interprocess communication (IPC) system.

Cronus supports load-balancing across a cluster by automatically routing operation invocations to the appropriate server. That server's underlying operating system then performs local scheduling.

Internally, Cronus is built upon:

- A distributed, fault tolerant mechanism for generating unique identifiers for all Cronus objects such as files and processes.
- An IPC facility based on standard network protocols and machine- and language-independent message formats defined in terms of byte strings. The IPC facility transports Cronus operation invocations on objects transparently from machine to machine.
- A set of Cronus processes (user and system) running on each host that execute the operations sent via the Cronus IPC.
- An object-based approach to support for client-server computing.

F.2.1 Cronus Object Model

The architecture of the Cronus application environment is its *object model* view of processes and resources. Within the Cronus environment, each system resource (e.g., file, device, database) is seen and acted on as a typed object. To support application processing, Cronus implements a *manager/client* model. An object manager ("server" in standard client-server terminology) is a program which runs on a Cronus host and is responsible for a Cronus particular object type. An object client is a program which invokes operations on the object. Managers and clients communicate by messages using the Cronus IPC system, and need not reside on the same Cronus host.

Several integral parts of the Cronus system, such as the Directory Manager and the Type Definition Manager, are fully implemented object managers themselves. For most applications, however, a Cronus application developer will need to define new object types and develop the managers which implement them. The procedure for creating an object manager is straightforward, and much of the basic framework of manager construction is automated with the Cronus Manager Development tools.

After writing a manager for a particular type of object, the developer can write client programs which call on the manager. The manager can be implemented separately from the client programs, providing a mechanism for freezing the object interface and splitting up the development of distributed applications.

F.2.2 History and Status

Cronus was developed by BBN Systems and Technologies Corporation (BBN) under contracts to the Rome Air Development Center (U. S. Air Force) and the Naval Ocean Systems Center (U. S. Navy). It has been hosted on a variety of machines (primarily Unix platforms) that use TCP/IP as their network protocol. Cronus is currently in use in a number of environments for distributed applications including satellite telemetry processing and real-time simulation support.

Cronus includes support resource leveling and load-sharing in a cluster, and enhances process survivability. It includes interfaces using a subset of SQL to several commercial databases (Informix, Oracle, and Sybase). It supports network security with an Authentication Manager that is compatible with recent system security research.

Support for Cronus is available from BBN.

F.3 Background of this Evaluation

F.3.1 Original Direction

CSC was directed to investigate environments which would support the data and tool interoperability requirements of the distributed heterogeneous GSDE. It was planned that CSC would evaluate Cronus by attempting to interface a CSC-developed interoperability utility, implemented in the Ada programming language, with Cronus's native interoperability capabilities. Development of prototype software, in Ada on the Rational R1000, was considered to be an important part of the study. CSC delivered a Prototype Software Development Plan describing the approach.

F.3.2 Initial Results and Redirection

Cronus presently provides support for application development in the C and Common Lisp languages. CSC attempted to interface the C-language routines in Cronus with the Ada-language interoperability suite already developed by CSC. The task proved to be impractical due to time constraints and the complex nature of the data structures and language characteristics involved. (Interfaces have been developed between Ada and Cronus; the CSC interoperability suite poses a particularly complex interface problem.)

CSC initiated discussions with the BBN staff regarding the interface problems, with two results. First, CSC learned that BBN was developing an Ada implementation of Cronus which would minimize the language interface problems. Second, the nature of CSC's work was discussed in order for BBN to offer work-around suggestions to the problems. Through these discussions, CSC realized that the interoperability prototype effort was not the best way to evaluate Cronus because Cronus inherently provides robust interoperability support. RICIS and CSC realized a change in direction of the Cronus evaluation would prove more fruitful in determining whether Cronus would satisfy the GSDE needs.

The effort was not wasted, though it was not as productive as had been expected. As part of this effort, CSC also studied the internal structure of Cronus, and this knowledge, along discussions with BBN, provided important insights into Cronus's full potential for the GSDE context (discussed below).

The prototyping plan (in CSC/TR-90/6155) is intended to provide more detail as well as qualitative assessment of Cronus for the GSDE. However, the prototyping effort was discontinued at the customer's direction. CSC was redirected to assemble its findings and discuss capabilities from experience to data and from information provided by BBN.

F.3.3 Current Status of Cronus in STAR*LAB

To date, CSC has obtained versions of Cronus written in C and Ada for the Sun 3/260 and a C version for the MicroVAX running VMS, both located in STAR*LAB. Both C versions have been installed and tested using demonstration applications provided by BBN. The C versions are supported by BBN; the Ada version is a prototype provided by BBN as a courtesy.

At CSC's invitation, BBN visited STAR*LAB to provide further insight into the optimal use of, and future enhancements to, Cronus. BBN is in the process of finishing a prototype Ada implementation of Cronus for the Air Force. Discussions were started between BBN and CSC on the feasibility of porting the Ada prototype of Cronus to the Rational. These discussions are on hold pending a resumption of funding and the direction to begin prototyping efforts.

F.4 Cronus Support For Requirements

F.4.1 Amdahl-IV&T Interface

The GSDE is a development environment that is a heterogeneous network of computers. Developing software in this environment requires that some tools, such as the configuration management capability, operate over the network, and may require that some data, such as software test data, be moved between machines on the network. Specifically, some of the distributed network services include the following:

- name service: providing system-wide unique identifiers for products and processes
- directory services: tracking the location of specific products and copies of products (e.g., IV&T-resident copies of Amdahl-controlled files)
- distributed transaction services: processing command scripts that involve operations on more than one platform (e.g., TBU scripts)
- dynamic message addressing services: supporting location independence for processes that involve network communications. For example, application A talks to application B on a different platform, where the locations of A and B are dynamically determined.
- distributed database interaction services: interacting with the Amdahl-based CM system from other platforms that are networked to the Amdahl.

F.4.2 Capabilities to Support GSDE Requirements

Since no actual prototyping has been completed, the assessment in this section is based on analysis of documentation, limited testing, and discussions with BBN.

F.4.2.1 Name Service

Cronus provides a distributed, fault-tolerant generator that can provide unique identifiers far in excess of any anticipated demand (the total number of unique identifiers is 2^{48}). Compound identifiers are generated for objects, indicating the host as well as the object being tracked. This approach provides for location-specific identification and version management.

Identification service support is hierarchical, and can be partitioned so that specification of identifiers (suffixes, actually) can be performed in Cronus hosts without network interaction.

F.4.2.2 Directory Service

Cronus's name space is used to identify all user-defined objects (e.g., files and processes) known to Cronus. Thus, Cronus supports location transparency for user defined objects. Furthermore, Cronus supports object replication, i.e., multiple copies of a resource are kept on different hosts. Replication is a method of improving the availability of resources in a distributed computing environment.

F.4.2.3 Distributed Transaction Services

Cronus provides communications and messaging services among Cronus applications wherever they are in a cluster. The underlying communications protocol of the network (TCP/IP is currently supported) is used for actual data transfer between hosts, while intra-host transfers use either the Unix User Datagram Protocol or a special-purpose large-message protocol. Support for distributed transaction processing would require specific application development, but the basic tools are in place.

F.4.2.4 Dynamic Message Addressing Services

Cronus provides location independence for manager-server interactions through the Directory service and process-forwarding mechanisms. Cronus does not directly support client-to-client messaging.

F.4.2.5 Database Interaction Services

Cronus includes database access servers for several commercial relational database management systems (Informix, Oracle, Sybase). These servers implement a subset of Structured Query Language (SQL), allowing client applications to interact with remote databases. Multiple Cronus applications can gain access to a single database with the use of multiple instances of database session manager applications (DBMS front-ends).

F.4.2.6 System Approach to Reliability

Cronus provides a monitoring and control system that can "check the heartbeat" of each processor in a Cronus cluster and can restart Cronus services that have failed on a given processor. It is not clear that Cronus can reboot failed processors.

As stated in the Cronus documentation, replication of objects is one of the key concepts in implementing survivable Cronus applications that are able to continue functioning in the presence of failures. Cronus provides facilities for propagating changes made to one object instance to the other copies of that object. Furthermore, because the replication needs of Cronus applications differ, several replication strategies are supported.

F.4.2.7 Interoperability Support

Cronus supports data interoperability between different architectures for several canonical data types (e.g., the 16-bit integer). It allows users to define object types that are then managed through Cronus services just like predefined Cronus objects.

F.4.2.8 Security

Cronus implements security with the use of application-specific access controls and a user authentication mechanism. The security approach is based on the Kerberos network user authorization system developed on MIT's Project Athena for use in distributed computing environments where neither hosts nor users can be trusted.

F.4.3 Potential Cronus Deficiencies

F.4.3.1 Interface Concerns

A prototype Ada version of the Cronus computing environment is being implemented by BBN. The native Ada implementation should eliminate many of the interface problems experienced in the initial study of interfacing an Ada utility to the C implementation of Cronus. However, since no pure Ada Cronus kernel exists today, there may be Ada/Cronus interface issues similar to Ada/SQL interface issues.

F.4.3.2 Performance

Communications through the layers of the Cronus computing environment necessarily entail overhead on the process. The magnitude of this overhead cannot be determined without some testing, preferably on a realistic prototype of some GSDE applications. BBN is concerned about the problem, and has re-written the Cronus kernel and IPC service for better performance. However, there is no easy way to assess performance without actual testing.

F.4.3.3 Maintenance Support

Cronus is offered for sale by BBN to both government and commercial customers, the former without run-time license fees. A variety of Cronus support services are available from BBN, including maintenance, hotline support, installation, and training.

F.4.3.4 Support for Porting Cronus to GSDE Platforms

Cronus provides a machine-independent interface for process and file management that allows the same tools to be executed across a heterogeneous set of computers. If Cronus is used as a component of GSDE, then it must support the platforms and development tools required by GSDE. The Cronus evaluation therefore must eventually answer the following questions regarding the support Cronus offers to GSDE platforms and tools.

- How many Cronus ports must be made to support GSDE? What is the estimated cost per port?
- What provisions does Cronus make for supporting software development tools built on the underlying operating systems? For example, using compilers hosted on native operating system to develop and debug Cronus applications.
- How many tools would have to be ported to or built on top of Cronus to support GSDE?
- For tools to be ported onto Cronus, what operating system support must Cronus provide? Does Cronus provide the required support?

CSC inquired of BBN as to the typical cost of porting Cronus to a new platform. The response was that there are a large number of factors to be considered, and no simple general answer. BBN will provide an estimate for a specific machine and operating system, but they need to have a significant amount of information about the computer system to make the estimate.

The computer and operating system combinations currently supported are as follows:

Hardware	**Operating System**
Alliant FX/80	Concentrix 5.0
BBN Butterfly GP1000	Mach 1000
DEC RISC	Ultrix 3.x and 4.x
DEC VAX	Ultrix 3.x and 4.x
DEC VAX	VMS 5.x
Encore Multimax	UMAX 4.2
Masscomp 54xx and 55xx	RTU 4.x
Sun 2	SunOS 3.4, 3.5, and 4.x
Sun 3	SunOS 4.x

C-2

Sun 4	SunOS 4.x
Sun 386i	SunOS 4.x
Sun 3	Mach 2.5
Symbolics	Genera 8.0.1

F.4.4 PCEE Requirements and Cronus

The Portable Common Execution Environment (PCEE), conceptualized by Dr. Charles McKay at UHCL, has been proposed as a model to specify the requirements of the space station and ground support systems. Some of the requirements specified by the PCEE may also be applicable to the Amdahl-IVTE environment. These topics and their relationships to Cronus are introduced in this section. The first four issues were raised by Dr. McKay at a meeting in Houston.

F.4.4.1 Object Transactions

PCEE requires the support of atomic and nested object transactions, where object transactions refer to the manipulation of objects via other objects. Analogous to database transactions, it may be desirable to nest object transactions into one atomic transaction which provides the capability of rolling back a sequence of object manipulations if the entire sequence is not successful. This functionality is not explicitly addressed by Cronus, but could probably be provided by implementing a transaction manager shell. The Common APSE Interface Set (CAIS-A) specifies such a mechanism.

F.4.4.2 CIFO Support

The Catalogue of Interface Features and Options (CIFO) for the Ada Runtime Environment is the Ada Runtime Environment Working Group's (ARTEWG) catalogue of proposed interfaces to Ada runtime environments. CIFO is expected to improve the effectiveness of Ada applications and their supporting runtime environment implementation. Cronus would be even more efficient if it were to make use of these extensions and if the Ada implementation used to compile Cronus supported them.

F.4.4.3 RPC versus Ada Rendezvous Paradigm

Ada provides semantics which support inter-task communication rendezvous. These semantics could be extended to an object model as part of a distributed operating system. The Cronus inter-object communication model is based on vanilla RPC semantics. Preliminary investigations have not yet confirmed the flexibility of Cronus to support these semantic extensions.

F.4.4.4 Interoperability Support

Cronus takes significant advantages of its underlying target-specific operating systems, potentially compromising Cronus's abilities to achieve effective interoperability. However, since Cronus achieves interoperability via canonical data representation transformations, the support for data interoperability may be adequate. This issue may need extensive further investigation in the context of specific platforms and applications.

Cronus generally relies on the least-common-denominator features of the operating systems on which it is layered. The C version of Cronus, for example, consists almost exclusively of ANSI C and POSIX constructs. The Common Lisp implementation follows ANSI Common Lisp standards, as well as emerging standards for CLOS (Common Lisp Object System) and the CLIM (Common Lisp Interface Manager). To the extent that GSDE platforms support standard interfaces, problems using Cronus would be minimized.

F.4.4.5 The Object-Oriented Paradigm

The PCEE describes a very robust object-oriented paradigm. Cronus's object model, discussed above, supports this requirement.

F.4.4.6 Cluster Component Flexibility

Cronus supports most of the distributed requirements specified by the PCEE, including on-the-fly cluster configuration. New components of an existing Cronus cluster can be added at any time. Similarly, components can be removed from a cluster without interfering in the operation of a cluster. Furthermore, because of robust intra-cluster communication, components can fail without bringing down the entire cluster. Also, components will have the capability to be restarted after a failure because Cronus is typically brought up automatically when a system is booted.

The present version of Cronus does not address communication between clusters, and it is not apparent whether inter-cluster communication is a GSDE requirement. However, it will be addressed in future Cronus versions.

F.4.4.7 Dynamic Software Modifications

PCEE poses the requirement of dynamic software changes. This implies that software be able to be changed while it is executing. No operating system currently provides full support for this capability. In an interpretive object-oriented environment, this feature could probably be implemented, as it is in many object-oriented languages. Dynamic modification may, therefore, be a limitation of the underlying operating system rather than one of Cronus.

F.5 Summary

From our investigation into the GSDE interface problem and Cronus capabilities, we have concluded that Cronus is at least a viable candidate to provide GSDE interface functionality. It seems clear that the capabilities of Cronus match some of the unresolved requirements of distributed software management within the GSDE. It is also clear that application software would have to be developed to make those capabilities available in fact. (Supporting the development and use of such applications is the normal mode of use for Cronus; the GSDE is not unique in that respect).

Major questions would have to be answered before a decision was taken to use Cronus in the GSDE. These questions involve cost and schedule of porting Cronus to the GSDE, cost and schedule of developing needed applications, and performance costs of using Cronus. The first question is partly dependent on the platforms selected for the GSDE in the OADP process, and partly on the difficulty involved in creating a full Ada implementation. The second and third problems would be addressed by the proposed prototyping effort.

Glossary and Abbreviations

Ada	Ada programming language; Ada is a registered trademark of the US Government, Ada Joint Program Office
ADF	Ada Development Facility (a Rational product, part of the SSE)
APSE	Ada Programming Support Environment
ARTEWG	Ada Real-time Environment Working Group
build products	object code, executable images, or load libraries generated from configuration-controlled source code
CAIS-A	Common APSE Interface Set-version A
CI	configuration item, any item which has been placed under configuration control
CIFO	Catalog of Interface Features and Options (for realtime Ada)
CLIM	Common Lisp Interface Manager
CLOS	Common Lisp Object System
CM	configuration management
CMVC	Component Management and Version Control system (a component of the Rational ADF)
COTS	commercial, off-the-shelf (i.e., commercially available hardware or software products not requiring SSFP-specific development
CR	change request, a document used to request and possibly authorize a change to a controlled baseline
Cronus	distributed network operating system, developed at Rome Air Development Center by BBN Systems
CSC	Computer Sciences Corporation
DBMS	database management system
DEC	Digital Equipment Corporation
download	to transfer a file from a remote computer to the initiating computer (see also upload)

PRECEDING PAGE BLANK NOT FILMED

DSDM	Digital Systems Development Methodology, a trademark of the Computer Sciences Corporation; a detailed set of project management guides and procedures
GCM	GSDE-based configuration management system
GS/SPF	Ground Systems Software Production Facility
GSDE	Ground Systems Development Environment
GSFC	Goddard Space Flight Center
GSS	Ground Systems Support computer, part of the SSCC
IBM	International Business Machines
IPC	inter-process communication, an element of distributed computing architectures
IV&T	integration, verification, and test
JSC	Lyndon B. Johnson Space Center
LAN	local area network
MCC	Mission Control Center
MOD	Mission Operations Directorate
MSC	Mission Systems Contract
NASA	National Aeronautics and Space Administration
OADP	Office Automation, Data Processing procurement, an umbrella procurement of computer systems for NASA JSC
OS	operating system
PCEE	Portable Common Execution Environment
POSIX	Portable Operating System Interface (standard)
QA	quality assurance
RACF	Remote Access Control Facility, a security package on IBM mainframe systems
RJE	remote job entry

RPC	remote procedure call, an element of distributed computing architectures
RICIS	Research Institute for Computing and Information Systems
SMAP	Software Management and Assurance Program, a set of guidelines developed by NASA for safety, reliability, maintainability, and quality assurance of software
SPE	software production environment, a collection of LAN-linked computers and workstations used for software development prior to formal target-based testing
SPF	software production facility
SQL	Structured Query Language, an access language for relational database management systems
SSCC	Space Station Control Center
SSE	software support environment
SSFP	Space Station Freedom Program
SSTF	Space Station Training Facility
TBU	Target Build Utility; a component of the Rational ADF
TCP/IP	transmission control protocol/internet irotocol; a network interface standard
TSC	Training Systems Contract
UHCL	University of Houston-Clear Lake
upload	to transfer a file from the initiating computer to a remote computer (see also download)
WAN	wide area network