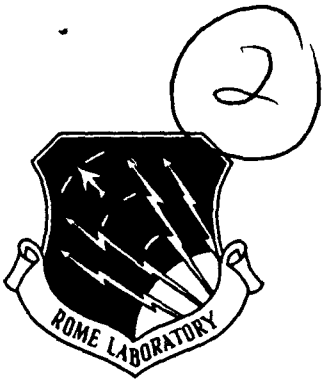


RL-TR-92-102
In-House Report
August 1992

A MAPPING OF SLCSE SERVICES TO THE NIST ISEE REFERENCE MODEL FOR CASE ENVIRONMENT FRAMEWORKS

James R. Milligan

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-92-102 has been reviewed and is approved for publication.

APPROVED: Samuel A. Dinitto

SAMUEL A. DINITTO, JR., Chief
Software Technology Division

FOR THE COMMANDER: John A. Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (C3CB) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1992	In-House Mar 91 - Aug 91

4. TITLE AND SUBTITLE
A MAPPING OF SLCSE SERVICES TO THE NIST ISEE REFERENCE MODEL FOR CASE ENVIRONMENT FRAMEWORKS

5. FUNDING NUMBERS
PE - 62702F
PR - 5581
TA - 18
WU - 47

6. AUTHOR(S)
James R. Milligan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CB)
Griffiss AFB NY 13441-5700

8. PERFORMING ORGANIZATION REPORT NUMBER
RL-TR-92-102

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CB)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: James R. Milligan/C3CB (315) 330-2054

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report describes the results of a Rome Laboratory in-house mapping exercise between the services provided by the Software Life Cycle Support Environment (SLCSE) and a Reference Model developed by the National Institute of Standards and Technology (NIST) Integrated Software Engineering Environment (ISEE) Working Group. SLCSE is a computer-based framework for the instantiation of Software Engineering Environments (SEEs) that are tailored to accommodate the specific needs of software development projects. The NIST ISEE Reference Model concentrates on characterizing and relating the services which may be found in Computer Assisted Software Engineering (CASE) environment frameworks.

14. SUBJECT TERMS
SLCSE, Frameworks, Software Life Cycle, Reference Model, NIST, CASE, Environments, ISEE, Software Engineering

15. NUMBER OF PAGES
170

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

TABLE OF CONTENTS

PREFACE

This technical report describes the results of a mapping exercise between the services provided by the Software Life Cycle Support Environment (SLCSE) and a reference model developed by the National Institute of Standards and Technology (NIST) Integrated Software Engineering Environment (ISEE) Working Group.

The author of this report would like to thank the individuals of Rome Laboratory (RL) and the NIST ISEE Working Group who have contributed to the development of this report. Special thanks go out to Deborah Cerino (RL) and Lolo Penedo (TRW/NIST ISEE Working Group).

ACRONYMS

4GL - Fourth Generation Language
ACE - ACL Entry
ACL - Access Control List
ACS - Ada Compilation System
AD - Application Development
AD/Cycle - IBM's AD Environment
ADL - Ada Design Language
AFLC - Air Force Logistics Command
AFSC - Air Force Systems Command
ALC - Air Logistics Center
ALICIA - Automated Life Cycle Impact Analysis
AMS - Automated Measurement System
ANSI - American National Standards Institute
APSE - Ada Programming Support Environment
ASCII - American Standard Code for Information Interchange
ATVS - Ada Test and Verification System
BNF - Baukus-Naur Form
CAIS-A - Common APSE Interface Set, Revision A
CASE - Computer Assisted/Aided Software Engineering
CAVS - COBOL Automated Verification System
CDI - Companion Database Interface
CE - Command Executive
CIS - CASE Integration Services
CM - Configuration Management
COTS - Commercial Off-The-Shelf
CSC - Computer Software Component
CSCI - Computer Software Configuration Item
CSU - Computer Software Unit
DCL - Digital Command Language
DEC - Digital Equipment Corporation
DGL - Document Generation Language
DID - Data Item Description
DOD - Department Of Defense
DS - Deliverable Software
DSA - Dynamic Storage Allocation
DSR - Digital Standard Runoff
ECMA - European Computer Manufacturers Association
ER - Entity-Relationship
ERA - Entity-Relationship-Attribute
ERIF - ER InterFace
ESD - Electronic Systems Division

FSD - Full-Scale Development
GAFB - Griffiss Air Force Base
GKS - Graphical Kernel System
HLERIF - High Level ERIF
IBM - International Business Machines
IFU - InterFace Utility
ISEE - Integrated SEE
J73AVS - JOVIAL J73 Automated Verification System
KBSA - Knowledge-Based Software Assistant
LQP - Line Quality Printer
LSE - Language Sensitive Editor
MBX - MailBoX
MCCS - Mission Critical Computer System
MOO - Menu Operations Organizer
NDS - Non-Developmental Software
NIST - National Institute of Standards and Technology
OBS - Organizational Breakdown Structure
OM - Object Manager
OMS - Object Management System
OS - Operating System
PCRP - Problem/Change Report Processor
PCTE - Portable Common Tool Environment
PDL - Program Design Language
PDL - Program Design Language
PDSS - Post Deployment Software Support
PMA - Project Management Assistant
QA - Quality Assurance
QBE - Query By Example
QUES - QUality Evaluation System
RDB - Relational Data Base (VAX/VMS)
RDBMS - Relational DataBase Management System
RL - Rome Laboratory
RMU - RDB/VMS Management Utility
SDDL - Software Design and Documentation Language
SDF - Software Development File
SDF - Software Development Folder
SDL - Schema Definition Language
SEE - Software Engineering Environment
SEM - SLCSE Environment Manager
SLCSE - Software Life Cycle Support Environment
SPMS - SLCSE Project Management System
SQL - Structured Query Language
UI - User Interface

V&V - Verification and Validation
VAX - Virtual Address eXtension (of DEC PDP-11)
VMS - Virtual Memory System (OS)
WBS - Work Breakdown Structure

1. INTRODUCTION

This report describes the results of a Rome Laboratory in-house mapping exercise between the services provided by the Software Life Cycle Support Environment (SLCSE) [17] and a reference model developed by the National Institute of Standards and Technology (NIST) Integrated Software Engineering Environment (ISEE) Working Group. SLCSE (pronounced "slice") is a computer-based framework for the instantiation of Software Engineering Environments (SEEs) that are tailored to accommodate the specific needs of software development projects. The NIST ISEE Reference Model [2] is an evolving document which concentrates on characterizing and relating the services which may be found in Computer Assisted Software Engineering (CASE) environment frameworks.

The Reference Model is intended to help experts describe and compare CASE environment framework technologies. It is not a standard itself, and is not intended to be used as an implementation specification nor as the basis for appraising the conformance of actual implementations. The Reference Model is intended to identify areas for developing standards, and to provide a common reference for maintaining consistency between all related standards and products.

The purpose of the mapping exercise was to validate/improve the Reference Model. The services provided by five interface technologies (CAIS-A, PCTE, CIS, SLCSE, and AD/Cycle) were mapped to a selected set of service descriptions contained in the Reference Model. A meeting between NIST ISEE Working Group members and representatives for each of the five interface technologies was held March 12-13, 1991 to set the stage for the mapping exercise. The mapping exercise was conducted, and the results were then presented and reviewed at the 5th NIST ISEE Workshop held June 1-3, 1991.

SLCSE Version 3.5 was delivered to Rome Laboratory in September 1989. Since then, the system has been used at Air Force Logistics Command (AFLC) Air Logistics Centers (ALCs) for beta testing, and by several other organizations that performed technical evaluations. In addition, SLCSE has been distributed to dozens of Government and Industry organizations for a variety of purposes. SLCSE Version 3.9 resulted from the development of the SLCSE Project Management System (SPMS) [9].

At the time of the mapping exercise, the SLCSE system being mapped (Version 3.9) resulted from advanced research and development sponsored by Rome Laboratory (i.e., SLCSE was an "advanced development prototype").

Referring to the system as a "product" would lead one to infer that the system was ready to go to market, which was not the case. Beta test reports, technical evaluations, and other forms of feedback from SLCSE users have been, overall, very positive. However, further development of SLCSE is necessary in order to obtain a framework capable of supporting the needs of the Air Force and the DOD.

As a result, Rome Laboratory, in conjunction with Electronic Systems Division (ESD), are jointly sponsoring a five year initiative entitled the "SLCSE Enhancements and Demonstration Program" [7]. The program has the following three major objectives.

- Provide the necessary functional and performance enhancements to address user concerns and issues that were documented as a result of the beta tests and evaluations.

- Establish SLCSE as an operational product, with the necessary marketing and support mechanisms.

- Support specific user needs through a user-funded task order mechanism.

The mapping exercise, in addition to contributing to the fulfillment of Rome Laboratory's long-standing technology transfer objectives, was particularly useful in assessing areas where SLCSE can be enhanced to result in a product that offers as rich a set of ISEE services as possible.

2. BACKGROUND

2.1 A Reference Model for CASE Environment Frameworks

A reference model for Computer Assisted Software Engineering (CASE) environment frameworks was prepared by the National Institute of Standards and Technology (NIST) Integrated Software Engineering Environments (ISEE) Working Group [2]. The NIST ISEE Reference Model was a revision of a European Computer Manufacturers Association (ECMA) technical report [1].

The NIST ISEE Reference Model used for the mapping exercise was a working draft strictly for internal use by the NIST ISEE Working Group. Since then, the Reference Model has been (and continues to be at the time of this writing) revised by the Working Group.

2.1.1 Aims of the Reference Model

The NIST ISEE Reference Model is intended to help experts describe and compare CASE environment framework technologies. The Reference Model is not a standard itself, and is not intended to be used as an implementation specification nor as the basis for appraising the conformance of actual implementations. The Reference Model is intended to identify areas for developing standards, and provide a common reference for maintaining consistency of all related standards and products.

Specifically, the aims of the reference model are as follows.

- Description and Comparison

The Reference Model should be suitable for being used to describe, compare, and contrast existing and proposed environment frameworks.

- Evolution of Standards

The Reference Model should provide a framework for the smooth and coordinated evolution of future standards in particular to ensure that the early standards are developed in such a way that further standards may easily achieve alignment in the sense of upward compatibility.

- Integration and Interoperability

The Reference Model should address interoperability and integration of tools.

- Degree of Generality

The Reference Model has to be able to be used to describe a wide range of CASE environment framework designs, but should balance this against a requirement to be able to define points at which useful standards can be defined.

- Education

The Reference Model should be capable of being used as the basis for educating systems engineers in the subject of CASE environment frameworks.

- Unifying Concepts

The number of unifying concepts required to describe the reference model should be small and the model should recognize the importance of the relationships which exist between its elements.

- Software Development Method Independence

The Reference Model should cover all system aspects irrespective of implementation techniques or software development methods employed by particular CASE environments or systems developed within CASE environments.

- Related Models

The Reference Model should be compatible with other appropriate reference models.

2.1.2 Reference Model Service Descriptions

The Reference Model describes a number of "services" that may be found in a CASE environment framework. These services are grouped, for notational convenience only (i.e., the groups are not intended to imply any specific functional attachments between individual services within a group), as follows.

- Object Management Services
- Task Management Services
- Communication Services
- User Interface Services
- Security Services
- Framework Administration and Configuration Services
- Integration Services

Tools are considered by the Reference Model as pieces of software that are not part of a CASE environment framework, and which call upon the CASE environment framework services listed above.

2.1.3 Reference Model Dimensions

The Reference Model describes each service in terms of "dimensions". A dimension is used to obtain a particular kind of description for a given service. That is, a service described in one dimension would be described differently using another dimension because each dimension emphasizes a distinct perspective of a service. Each dimension of the Reference Model is comprised of several items (which, for discussion purposes, will be referred to as "factors" in this report), as follows.

ICE Dimension
- *I*nternal
- *C*onceptual
- *E*xternal

ROD Dimension
- *R*ules
- *O*perations
- *D*ata

TIM Dimension
- *T*ypes
- *I*nstances
- *M*etadata

In addition to these dimensions, the Reference Model also defines three other factors for describing services:

- Degree of Understanding
- Relationships Between Services
- Justification for Including Services

2.2 Software Life Cycle Support Environment (SLCSE)

The Software Life Cycle Support Environment (SLCSE) is a VAX/VMS computer-based framework for the instantiation of Software Engineering Environments (SEEs) that are tailored to accommodate the specific needs of software development projects.

A SEE instantiated from the SLCSE framework consists of a set of integrated tools that support the Full-Scale Development (FSD) and Post-Deployment Software Support (PDSS) phases of the Mission Critical Computer System (MCCS) software life cycle. Specifically, SLCSE provides support for the various phases and inter-phase activities of the software life cycle, including requirements specification and analysis, design, coding, unit/integration testing, Quality Assurance (QA), Verification and Validation (V&V), project management, and Configuration Management (CM).

As illustrated in Figure 1, the SLCSE top-level architecture consists of four major subsystems: the User Interface, Command Executive, Database, and Toolset.

The User Interface is window-oriented and menu-driven, providing a common and consistent style of operation to all its users. While the User Interface is consistent in style, the tools and database views a user has access to are governed by the various "role(s)" the user has been assigned by the framework administrator.

The Command Executive controls all SLCSE functions (apart from those which are specific to a tool). It interfaces with the user through the User Interface, and invokes tools in the Toolset.

The Database provides SLCSE applications with an Entity-Relationship (ER) InterFace (ERIF) to an underlying relational database engine. At its highest level of abstraction, the SLCSE Database appears as a single expansive ER network of information. The current ER database schema models the data requirements of DOD-STD-2167A (the Military Standard for Defense System Software Development), and is so comprehensive and complete that each of the seventeen (17) DOD-STD-2167A data items (i.e., documents and specifications) can be automatically generated.

The Database is the most critical and important part of SLCSE. It serves not only as a repository for formal life cycle information required by DOD-STD-2167A (or potentially by any other life cycle model), but also as an integrating mechanism for tools by allowing them to share information. As data is generated and stored by users and tools, it becomes available through the Database services for use by other users and tools in subsequent activities and life cycle phases. The SLCSE integrating framework, and in particular,

SLCSE TOP-LEVEL ARCHITECTURE

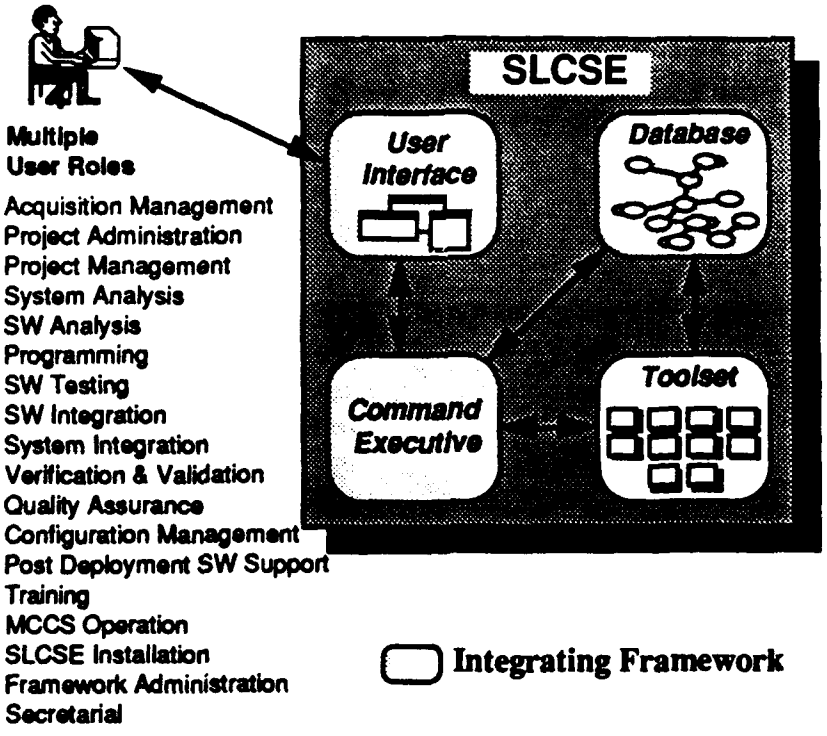


Figure 1

the interaction between tools and the Database, is what makes possible many life-cycle oriented technological opportunities and potential productivity gains.

The User Interface, Command Executive, and Database are considered to be the "integrating framework" of SLCSE, and are the three subsystems with which tools of the Toolset are integrated. A tool is invoked by the Command Executive from a tool user interface that may be "conformant" to the SLCSE User Interface, and may interact with other tools through the Database, which stores and manages all of the project data created throughout the life cycle.

SLCSE has been designed to allow virtually any number of tools to be integrated into the Toolset to support a variety of software engineering methods and overall development methodologies. This "methodology independent" tool integration concept provides for the use of Commercial Off-The-Shelf (COTS) tools, as well as tools developed specifically for use within SLCSE.

A more detailed architectural/operational description of SLCSE, as shown in Figures 2, 3, and 4, reveals that the Database subsystem is actually a hybrid implementation that includes three different data models, the primary one being the ER data model just described. The other two data models are the Infrastructure data model (used to represent all of the data objects that are necessary for the operation of SLCSE itself), and the Project Files Hierarchy data model (used to represent the file structures necessary for the storage and management of user, database, and SLCSE maintenance files). In addition, as illustrated in Figures 2, 3, and 4, it is evident that the modes of operation are quite different for: (1) SLCSE framework administration and tool integration (Figures 2 and 3), and (2) the use of an environment instantiation SLCSE framework (Figure 4).

FRAMEWORK ADMINISTRATION

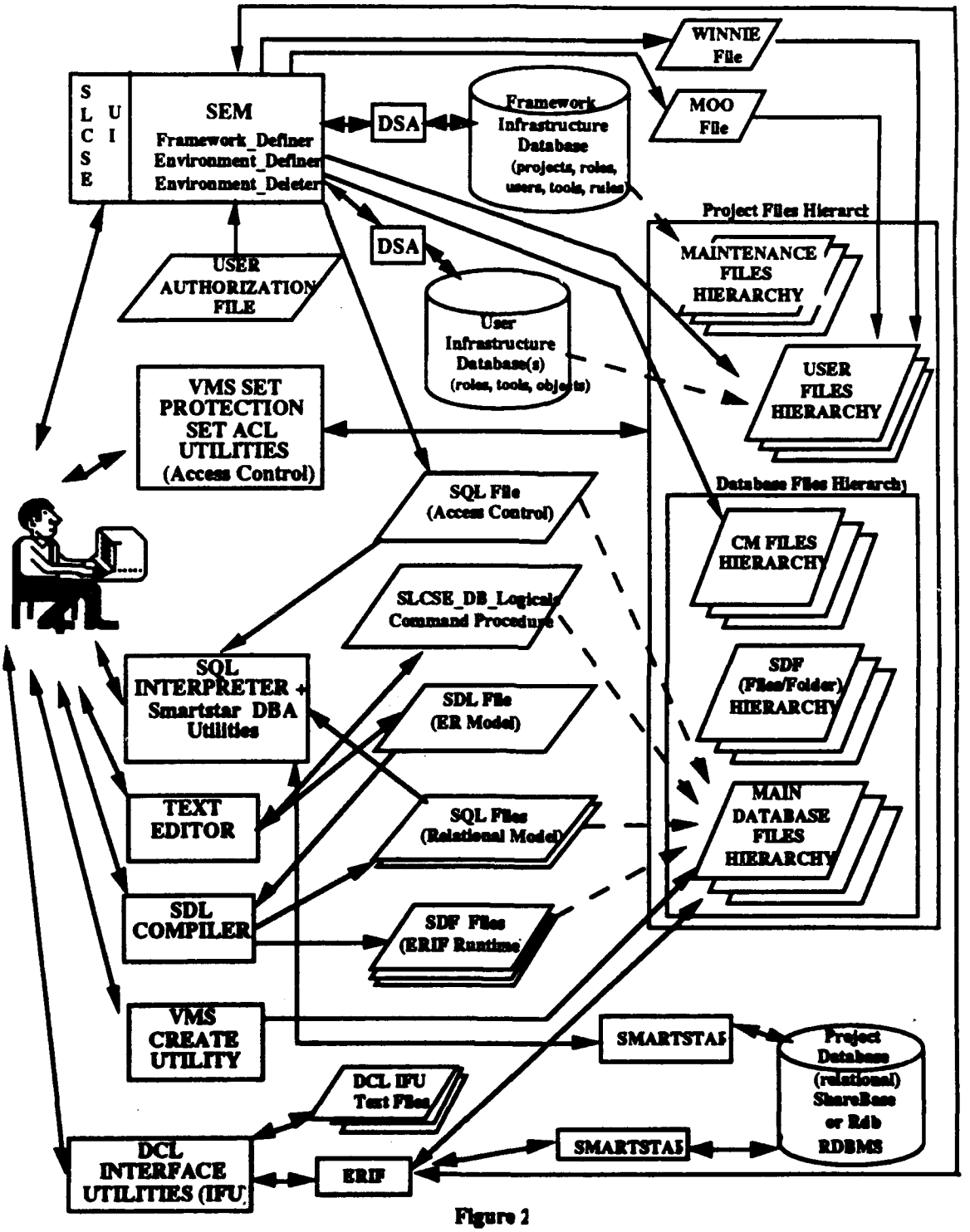


Figure 2

TOOL INTEGRATION

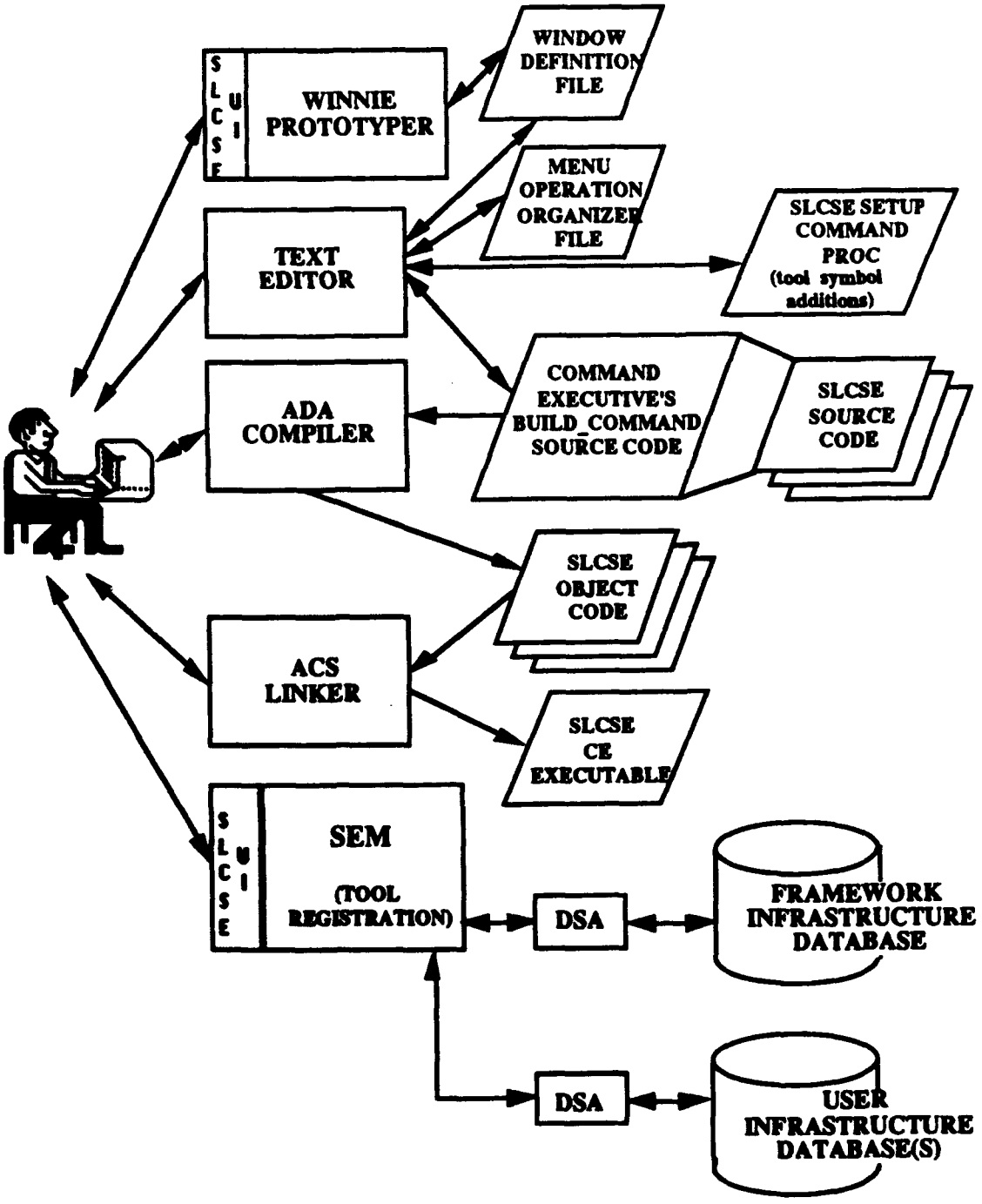


Figure 3

USER ENVIRONMENT

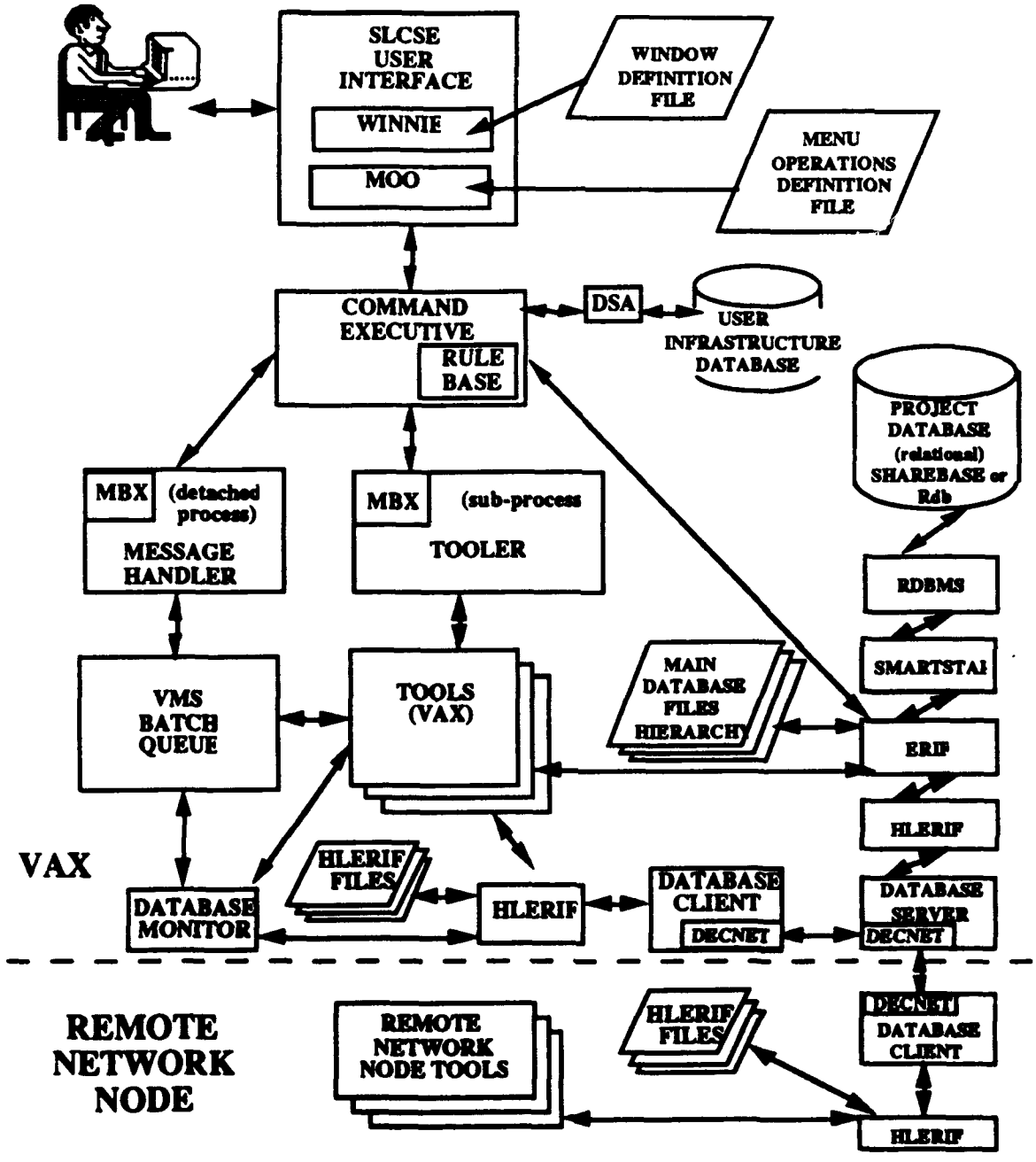


Figure 4

3. SLCSE MAPPING TO THE REFERENCE MODEL

The following sections describe the mapping of SLCSE services to the NIST ISEE Reference Model. Section 3.1 lists the Reference Model service descriptions selected for validation during the mapping exercise. Section 3.2 lists the Reference Model dimension factors (and other factors) that were recommended by the NIST ISEE Working Group to each of the mappers for describing the services provided by their standard/product. Section 3.3 describes the services of SLCSE as mapped to the selected service descriptions of the Reference Model, and section 3.4 provides comments on the Reference Model based on the SLCSE mapping.

3.1 Selected Services of the Reference Model

The following NIST ISEE Reference Model service descriptions were selected for validation during the mapping exercise.

7 Object Management Services
- 7.1 Data Model
- 7.2 Data Storage Service/Persistence
- 7.3 Relationship Service
- 7.4 Name Service
- 7.5 Distribution/Location Service
- 7.6 Data Transaction Service
- 7.7 Concurrency Service
- 7.8 Process Support Service
- 7.9 Archive Service
- 7.10 BackUp Service
- 7.11 Derivation Service
- 7.12 Replication/Synchronization
- 7.13 Access Control/Security
- 7.14 Constraint/Inconsistency Management
- 7.15 Function Attachment
- 7.16 Global/Canonical Schema
- 7.17 Version Service
- 7.18 Configuration Service
- 7.19 Query Service
- 7.20 Metadata Service
- 7.21 State Monitoring Service/Triggering
- 7.22 Sub-Environment (Views) Service
- 7.23 Data Interchange Service
- 7.24 Tool Registration

9 Tools
- 9.3 Tool Integration
  - 9.3.1 Data Integration
  - 9.3.2 Control Integration
  - 9.3.3 User Interface Integration

10 Task Management Services
- 10.1 Task Definition Service
- 10.2 Task Execution Service
- 10.4 Task History Service
- 10.5 Event Monitoring Service
- 10.6 Audit and Accounting Service
- 10.7 Role Management Service
- 10.8 Tool Registration

11 Message Services
- 11.1 Message Delivery Service
- 11.2 Tool Registration Service

13 Security
- 13.1 Security Information Class
- 13.2 Security Control Services
- 13.3 Security Monitor Services
- 13.4 Related Services

14 Framework Administration and Configuration
- 14.1 Tool Registration

15 Integration
- 15.1 Data Integration
  - 15.1.1 Object Management as a Data Integration Mechanism
  - 15.1.2 Common Data Descriptions
  - 15.1.3 Tool-to-tool Data Translators
  - 15.1.4 Tool-to-OM Translators
  - 15.1.5 OM-to-OM Exchange
  - 15.1.6 Consistency Management

3.2 Selected Dimension Factors of the Reference Model

The dimension factors (and other factors) described in the Reference Model that were recommended by the NIST ISEE Working Group to each of the mappers for describing the services provided by their standard/product

are listed below. In addition, each factor is associated with a [denotation] of how strongly each factor was recommended.

- Conceptual [required]
- Operations [required]
- Relationships Between Services [required]
- External [recommended]
- Data [optional]
- Rules [optional]
- Types [optional]
- Instances [optional]
- Metadata [optional]
- Internal [optional]

For the SLCSE mapping exercise, emphasis was placed on providing service descriptions with regard to the [required] factors (i.e., *Conceptual*: what a service does; *Operations*: how a service does what it does; and *Relationships Between Services*: what the interdependencies are between a service and other services).

3.3 Service Descriptions for SLCSE

The services of SLCSE as mapped to the selected service descriptions of the Reference Model are provided in the following sections. Each SLCSE service is described in terms of one or more Reference Model dimension factors (typically the "Conceptual", "Operational", and "Relationships Between Services" factors).

3.3.1 Object Management Services (7)

3.3.1.1 Data Model (7.1)

3.3.1.1.1 ICE - Conceptual

There are three interrelated data models in SLCSE for: (1) "project database" objects, (2) "infrastructure database" objects, and (3) "project file" objects. Each environment instantiated from the SLCSE framework is associated with a Project Database Model, an Infrastructure Database Model, and a Project Files Hierarchy Model. Each of these will be described in turn at the conceptual level.

PROJECT DATABASE MODEL: SLCSE provides an

Entity-Relationship (ER) data model which is used to define all of the data objects that are associated with a particular software development project (e.g., requirements, PDL, test cases, source code, tasks, schedules, resources, milestones, etc.). The Project Database Model is a level of abstraction above an underlying relational data model.

The Project Database Model is defined using a formal language called the "Schema Definition Language (SDL)". SDL is used to define "entity types" and "relationship types", and the "attributes" of those types of objects. In addition, SDL allows the logical grouping of entity and relationship types into "subschemas", and permits the definition of relationship types between entity types defined in different subschemas. Relationship type definitions include a "domain" entity type that has a "relation" to a "range" entity type (e.g., water (the domain) extinguishes (the relation) fire (the range)). The syntax of SDL is described in Backus-Naur Form (BNF) in Appendix 7 of [13]. A comprehensive example of a SDL-specified SLCSE Project Database Model schema for the data objects associated with a DOD-STD-2167A-compliant software development project is defined in Appendix 1 of [13]

INFRASTRUCTURE DATABASE MODEL: SLCSE provides a list-based data model which is used to define all of the data objects that are necessary for SLCSE operation (i.e., framework administration and SLCSE Command Executive information about such things as project names, peripherals, personnel, roles, tools, files and their attributes, rules, subschemas, entities, project database entity access privileges, etc.).

The Infrastructure Database Model is defined formally in an Ada programming language specification. The syntax of Ada is described in ANSI/MIL-STD-1815A [3], and the SLCSE Infrastructure Database Model Ada specification is defined in Appendix 2 of [13].

PROJECT FILES HIERARCHY MODEL: SLCSE provides a hierarchical model for the storage of file objects. This model consists of three hierarchies: (1) a "User Files Hierarchy", (2) a "Database Files Hierarchy", and (3) a "Maintenance Files Hierarchy". User file, database file, and maintenance file objects are project file objects associated with an environment instantiation of the SLCSE framework, and the Project Files Hierarchy Model provides the structure for the storage and management of file objects within the environment. This model is based on the concept of "directories", which are special files in which other file objects may be logically placed. Directories may be placed within other directories, thus forming a hierarchical layering of levels of file groups.

3.3.1.1.1.1 Example

Examples of the above data models are best described by the schemas derived from them. See the examples given for the Reference Model service called "Global/Canonical Schema" (7.16).

3.3.1.1.2 ROD - Operations

Neither the Project Database Model, Infrastructure Database Model, nor the Project Files Hierarchy Model are truly object-oriented, and, therefore, no operations are inherently associated with objects defined in either model.

3.3.1.1.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires none of the other mapped services.

This service is required by the following services which are provided by SLCSE:

TRACE ON ENTITY 7_1_data_model BACKWARD
9 LEVEL RELATIONSHIP
– AS A MEMBER OF [all] SUBSETS.

7_1_data_model (BACKWARD) [object_management]
1- 7_16_global_schema depends_on 7_1_data_model
<--2- 7_14_constraint_mgt depends_on 7_16_global_schema
<--2- 7_20_metadata depends_on 7_16_global_schema
<--2- 7_2_data_storage depends_on 7_16_global_schema
<--3- 14_1_tool_registration depends_on 7_2_data_storage
<--4- 10_1_task_definition depends_on 14_1_tool_registration
<--5- 10_2_task_execution depends_on 10_1_task_definition
<--6- 10_4_task_history depends_on 10_2_task_execution
<--7- 10_2_task_execution@ depends_on 10_4_task_history

<--6- 10_5_event_monitoring depends_on 10_2_task_execution
<--6- 10_6_audit_accounting depends_on 10_2_task_execution
<--4- 10_7_role_mgt depends_on 14_1_tool_registration
<--4- 10_8_tool_registration depends_on 14_1_tool_registration
<--4- 15_1_4_tool_to_om depends_on 14_1_tool_registration
<--5- 15_1_5_om_to_om depends_on 15_1_4_tool_to_om
<--6- 15_1_1_data_integration depends_on 15_1_5_om_to_om
<--3- 7_11_derivation depends_on 7_2_data_storage
<--3- 7_12_replication depends_on 7_2_data_storage
<--4- 15_1_6_consistency_mgt depends_on 7_12_replication
<--3- 7_13_access_control depends_on 7_2_data_storage
<--4- 13_2_security_control depends_on 7_13_access_control
<--3- 7_17_version depends_on 7_2_data_storage
<--4- 7_18_configuration depends_on 7_17_version
<--3- 7_19_query depends_on 7_2_data_storage
<--3- 7_21_state_monitoring depends_on 7_2_data_storage
<--4- 10_5_event_monitoring depends_on 7_21_state_monitoring
<--3- 7_23_data_interchange depends_on 7_2_data_storage
<--4- 7_10_backup depends_on 7_23_data_interchange
<--5- 7_9_archive depends_on 7_10_backup
<--3- 7_3_relationship depends_on 7_2_data_storage
<--3- 7_4_name depends_on 7_2_data_storage
<--3- 7_6_data_transaction depends_on 7_2_data_storage
<--3- 7_7_concurrency depends_on 7_2_data_storage
<--4- 15_1_6_consistency_mgt depends_on 7_7_concurrency
<--3- 9_3_1_data_integration depends_on 7_2_data_storage
<--3- 9_3_2_control_integration depends_on 7_2_data_storage
<--3- 9_3_3_ui_integration depends_on 7_2_data_storage

3.3.1.2 Data Storage Service/Persistence (7.2)

3.3.1.2.1 ICE - Conceptual

There are three interrelated data storage services in SLCSE which correspond to each of its data models. The data models are: (1) the "Project Database Model", (2) the "Infrastructure Database Model", and (3) the "Project Files Hierarchy Model". Respectively, the data storage services are provided by: (1) an Entity-Relationship (ER) Interface (ERIF) and a High-Level ERIF (HLERIF), (2) Dynamic Storage Allocation (DSA) routines, and (3) Operating System (OS) utilities. The Data Storage Service provided by each of these will be described, in turn, at the conceptual level.

ERIF: The ERIF provides the basic operations necessary for the creation and storage of instances of the entity and relationship types defined within the Project Database Model for a particular environment instantiation of the SLCSE framework. These instances are called "entities" and "relationships" (or "links"), respectively. Each instance created is unique, and has a number of "attributes" (defined by the type) which characterize the instance. Each attribute of an entity or a relationship has a value that falls within a range defined by the attribute's "data type" (e.g., string, text, boolean, integer, float, enumeration, etc.).

The place for storing entities and relationships is in the underlying relational database of SLCSE (i.e., the Project Database), and these instances can be observed and examined at the ER level of abstraction via the operations provided by the ERIF and the HLERIF.

HLERIF: Built on top of the ERIF is the HLERIF. HLERIF provides operations that are tailored toward the Replication/Synchronization Service (7.12) of SLCSE.

DSA: DSA routines provide all of the operations necessary for the creation and storage of instances of data objects defined within the Infrastructure Database Model for SLCSE. These instances are called "records", and are characterized by the values of their "components" (bounded by the data types (e.g., user_name_type, name_type, boolean, etc.) of the components defined in the Infrastructure Database Model).

The place for storing Infrastructure Database Model records is in a list-based data repository (i.e., the Infrastructure Database) where these objects can be observed and examined via the operations provided by DSA.

OS: The OS provides all of the operations necessary for the creation and storage of file objects within the Project Files Hierarchy Model.

User files are stored in the User Files Hierarchy, Project Database files are stored in the Database Files Hierarchy, and SLCSE maintenance files are stored in the Maintenance Files Hierarchy. These objects can be observed and examined via the operations provided by the OS.

3.3.1.2.1.1 Example

The following is an example of the Data Storage Service provided by the ERIF described at the conceptual level: (1) the creation of an entity instance NON_DELIVERABLE_SOFTWARE, whose text data type attribute

DESCRIPTION and its enumeration data type attribute NDS_TYPE are assigned the values of "MacProject II" and "COTS_Software", respectively, (2) the storage of that instance in the Project Database, and (3) the retrieval of that instance at the ER level of abstraction for observation and/or manipulation of that data object.

The following is an example of the Data Storage Service provided by DSA described at the conceptual level: (1) the creation of a PERSONNEL_RECORD record whose User_Name_Type data type component USER_NAME and Name_Type data type component PROJECT are assigned the values of "milligan" and "SLCSE", respectively, (2) the storage of that record in the Infrastructure Database, and (3) the retrieval of that record for its observation and/or manipulation.

An example of the Data Storage Service of the OS described at the conceptual level is: (1) the creation of a source code file in the work directory of a SLCSE user performing the role of a programmer, (2) the storage of that object in the user's role directory, and (3) the retrieval of that object from the role directory to the work directory for observation and/or manipulation (e.g., display or compile). Another example is: (1) the creation of a text attribute file for an entity, (2) the storage of that file in the Database Files Hierarchy, and (3) the retrieval of that object for observation and/or manipulation.

3.3.1.2.2 ROD - Operations

PROJECT DATABASE: The basic set of operations (create, query, update, and delete) applicable to this service for the Project Database are provided by the ERIF, and are:

Create:
"Insert"
"Duplicate"

Query:
"Table"
"Entity_Name_Of"
"Domain_Name_Of"
"Range_Name_Of"
"Entity_Type_Of"
"Relationship_Type_Of"
"Domain_Type_Of"
"Range_Type_Of"
"Available_Schemas"

"Entity_Types_Of"
"Relationship_Types_Of"
"Components_Of"
"Attribute_Types_Of"
"Cardinality_Of"
"Attribute_Info"
"Datatype_Of"
"="

Update:
"Update"
"Reserve"
"Release"
"Set_Text_Attribute"
"Clear_Text_Attribute"
"Grant"
"Revoke"

Delete:
"Delete"

The exact syntax, parameters, and descriptions of these operations are described in Appendix 6 of [13]. Appendix 4 of the same document describes the use of the ERIF, while Appendix 5 describes the query syntax and semantics of the query string (a proper subset of ANSI SQL) specified in the "Table" operation.

The HLERIF also provides operations built on top of the ERIF operations, and are as follows:

Create:
"Add_Monitor_Action"
"Duplicate"
"Insert"

Query:
"Attribute_Error_Message"
"Collection_Error_Message"
"Condition"
"Count"
"Finalize"
"Find_Backward"
"Find_Forward"

"First"
"Get"
"Get_Current"
"Get_Error"
"Get_Instance_Storage"
"Get_Monitor_Action"
"Get_Next_Event"
"Get_Swap_Count"
"Get_Test_Error"
"Goto_First"
"Goto_Last"
"Goto_Next"
"Goto_Previous"
"Hlerif_Error_Message"
"Image"
"Initialize"
"Instance_Error_Message"
"Last"
"Local_Collection_Exists"
"Login"
"Logout"
"More_Errors"
"More_Monitor_Actions"
"More_Test_Errors"
"Print"
"Retrieve_From_Local"
"Retrieve_From_Slcse"
"Retrieve_Monitor_Actions"
"Test_Servers"
"Value"

Update:
"Save_To_Local"
"Save_To_Slcse"
"Set"
"Set_Instance_Storage"
"Set_Matching"
"Sort"

Delete:
"Delete"
"Destroy"
"Destroy_Local_Collection"

"Remove_Monitor_Action"

INFRASTRUCTURE DATABASE: The basic set of operations (create, query, update, and delete) applicable to this service for the Infrastructure Database are provided by DSA, and are:

Create:
"Copy_Of_Dataset"
"Copy_Of_DSA_List"
"Create_DSA_Database"
"Create_DSA_Storage_Area"
"Define_DSA_Data_Type"
"DSA_Locker"
"DSA_Pointer_To"
"New_Dataset"
"New_Dynamic_Dataset"
"Save_DSA_Database"
"Template"

Query:
"Check_Folder_Vars" (New DSA routine)
"Display_DSA_Database"
"Display_Object_Folders" (New DSA routine)
"DSA_List_Membership_Count"
"Get_Next_Token" (New DSA routine)
"Initialize_Menus_From_Database" (New DSA routine)
"Is_Empty"
"Length_Of_DSA_List"
"Length_Of_Dynamic_Dataset"
"Next_Dataset_On_DSA_List"
"Obtain"
"On_DSA_List"
"Query_For_Item_Type" (New DSA routine)
"Restore_DSA_Database"
"Restore_DSAbase" (New DSA routine)
"Switch"
"Through_DSA_List"
"Validate_Keyword" (New DSA routine)
"Validate_User_Name" (New DSA routine)

Update:
"Add_Item_To_List" (New DSA routine)
"Append_DSA_List"

"Change_File_Attributes" (New DSA routine)
"Change_Tool_Position" (New DSA routine)
"Compress_DSA_Database"
"Dataset_At"
"Expand_Dynamic_Dataset"
"Lock_Dataset"
"Lock_DSA_List"
"Put_On_DSA_List"
"Reduce_Dynamic_Dataset"
"Set_Current_Role" (New DSA routine)
"Set_Tool_Parameters" (New DSA routine)
"Sort_DSA_List"
"Store_In_DSA_Locker"
"Switch_Roles" (New DSA routine)
"Take_Off_DSA_List"
"Translate_Menu_To_Keyword" (New DSA routine)
"Unlock_All_Implicitly_Locked_Datasets"
"Unlock_Dataset"

Delete:
"Delete_Item_From_List" (New DSA routine)
"Destroy_Dataset"
"Destroy_DSA_List"

New DSA routines are those that were built upon the basic DSA operations especially for SLCSE.

PROJECT FILES HIERARCHY: The basic set of operations (create, query, update, and delete) applicable to this service for the Project Files Hierarchy are provided by the OS (and used by the Command Executive (CE) of SLCSE), and are to numerous to list here.

The OS "Run-Time Library" provides many VAX/VMS system level routines. Examples are "LIB$RENAME_FILE", and "LIB$DELETE_FILE". Pre-defined Ada interfaces for many system level routines are provided in the "STARLET", "SYSTEM", and "CALENDAR" packages. Ada interfaces for some commonly used Run-Time Library routines which were not included in the pre-defined packages have been written and included in a package called "MORE_STARLET".

The Run-Time Library routines are documented in a series of reference manuals, and descriptions of the individual Run-Time facilities, along with

reference sections describing the individual routines in detail, and can be found in [20], [21], [22], [23], [24], [25], and [26].

3.3.1.2.2.1 Example

PROJECT DATABASE:

```
-- This contrived example updates a requirement by name, then
-- deletes all performance requirements, then inserts a single
-- new performance requirement.
declare
    Req : Cursor;
    Buffer : Attribute_List;
begin
-- First find and modify the requirement named "SEARCHMODE". Note
-- that this will match the most recent version of SEARCHMODE.
    Table(Universe => "Requirement",
            Attributes => "Access_Name, Description",
            Buffer => Buffer,
            Current=> Req,
            Query => "Access_Name = 'SEARCHMODE'");
    if not End_Of_Table(Req) then
       Buffer(2).V_String(1..23) := "The New Search Mode ...";
       Update(Req);
    end if;
    Commit(Req);
    Close_Table(Req);
-- Now delete all performance requirements.
    Table(Universe => "Requirement",
            Attributes => "Access_Name, Kind",
            Buffer => Buffer,
            Query => "Kind = 'PERFORMANCE'",
            Current=> Req);
    while not End_Of_Table(Req) loop
       Delete(Req);
    end loop;
-- Now, using the same table, insert a new performance requirement.
    Buffer(1).V_String(1..6) := "AVGTIME";
    Buffer(2).V_String(1..11) := "PERFORMANCE";
    Buffer(3).V_String(1..19) := "The Average Time...";
    Insert(Req);
    Commit(Req);
    Close_Table(Req);
```

```
end;
```

INFRASTRUCTURE DATABASE:

```
declare
   role_tool_pointer :
   dsa_data_structures.tool_access_type;
   tool_pointer: dsa_data_structures.tool_access_type;
begin
   role_tool_pointer := new_dataset;
   role_tool_pointer := copy_of_dataset (tool_pointer);
   role_tool_pointer.allowable := true;
-- place new tool record on tool list for this role
   put_on_dsa_list (dataset => role_tool_pointer,
                    list => role_pointer.tools_list,
                    position => bottom );
end;
```

PROJECT FILES HIERARCHY:

User Files Hierarchy:

1. SLCSE user directs CE to supply a role directory file as input to Editor (update).
2. CE directs Editor output file to work directory (create).
3. CE renames Editor output file to role directory (update).
4. SLCSE user directs CE to delete previous version of the role directory file (delete).
5. SLCSE user directs CE to display file objects in the role directory (query).

Database Files Hierarchy:

1. Text attribute of entity is examined (query).
2. Text attribute file is copied to work directory from SLCSE$TEXT directory (create).
3. Text attribute file copy is changed (update).
4. Original SLCSE$TEXT directory text attribute file is replaced by updated copy (delete and create).

3.3.1.2.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between

services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_2_data_storage FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_2_data_storage (FORWARD) [object_management]
1- 7_2_data_storage depends_on 11_1_message_delivery
1- 7_2_data_storage depends_on 15_1_2_common_data_descr
1- 7_2_data_storage depends_on 7_16_global_schema
-->2- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->2- 7_16_global_schema depends_on 7_1_data_model

This service is required by the following services which are provided by SLCSE:

TRACE ON ENTITY 7_2_data_storage BACKWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_2_data_storage (BACKWARD) [object_management]
1- 14_1_tool_registration depends_on 7_2_data_storage
<--2- 10_1_task_definition depends_on 14_1_tool_registration
<--3- 10_2_task_execution depends_on 10_1_task_definition
<--4- 10_4_task_history depends_on 10_2_task_execution
<--5- 10_2_task_execution@ depends_on 10_4_task_history
<--4- 10_5_event_monitoring depends_on 10_2_task_execution
<--4- 10_6_audit_accounting depends_on 10_2_task_execution
<--2- 10_7_role_mgt depends_on 14_1_tool_registration
<--2- 10_8_tool_registration depends_on 14_1_tool_registration
<--2- 15_1_4_tool_to_om depends_on 14_1_tool_registration
<--3- 15_1_5_om_to_om depends_on 15_1_4_tool_to_om
<--4- 15_1_1_data_integration depends_on 15_1_5_om_to_om
1- 7_11_derivation depends_on 7_2_data_storage

1- 7_12_replication depends_on 7_2_data_storage
<--2- 15_1_6_consistency_mgt depends_on 7_12_replication
1- 7_13_access_control depends_on 7_2_data_storage
<--2- 13_2_security_control depends_on 7_13_access_control
1- 7_17_version depends_on 7_2_data_storage
<--2- 7_18_configuration depends_on 7_17_version
1- 7_19_query depends_on 7_2_data_storage
1- 7_21_state_monitoring depends_on 7_2_data_storage
<--2- 10_5_event_monitoring depends_on 7_21_state_monitoring
1- 7_23_data_interchange depends_on 7_2_data_storage
<--2- 7_10_backup depends_on 7_23_data_interchange
<--3- 7_9_archive depends_on 7_10_backup
1- 7_3_relationship depends_on 7_2_data_storage
1- 7_4_name depends_on 7_2_data_storage
1- 7_6_data_transaction depends_on 7_2_data_storage
1- 7_7_concurrency depends_on 7_2_data_storage
<--2- 15_1_6_consistency_mgt depends_on 7_7_concurrency
1- 9_3_1_data_integration depends_on 7_2_data_storage
1- 9_3_2_control_integration depends_on 7_2_data_storage
1- 9_3_3_ui_integration depends_on 7_2_data_storage

3.3.1.3 Relationship Service (7.3)

3.3.1.3.1 ICE - Conceptual

PROJECT DATABASE: The Relationship Service of SLCSE allows the definition of "relationship types" in the Project Database Model, and the creation and maintenance of Project Database instances of those types (via the ERIF, and consequently, via the HLERIF), which are called "relationships" or "links". Relationships possess descriptive attribute values and cardinality (One To One, One To Many, Many To One, or Many To Many). They link a "domain" entity with a "range" entity.

It is also possible to define a relationship "union" type, and to reference relationships which are members of the union. Relationship "alias" types are also supported in SLCSE to provide relationship access by more than one name.

INFRASTRUCTURE DATABASE: Relationships between objects in an Infrastructure Database are defined by pointers established between list record components, and are maintained via operations provided by DSA.

3.3.1.3.1.1 Example

PROJECT DATABASE:

```
relationship type Person Solves Problem
   cardinality Many To One is
      Solution : text;
end relationship;
```

INFRASTRUCTURE DATABASE:

```
type tool_record is
   record
      name : tool_name_type;
      descriptor : tool_descriptor_type;
      parameters : tool_parameters_record;
      tool_list : dsa_list_type;
      item_kind : item_type;
      allowable : boolean;
      available : boolean;
      tool_deleted : boolean;
      time_last_invocation : calendar.time;
      number_of_invocations : natural;
      last_status_of_tool : cond_value_type;
      keyword_only : boolean;
      pre_rule_list : dsa_list_type; -- tool-to-rules
-- dsa relationship.
      post_rule_list : dsa_list_type; -- tool-to-rules
end record;

tool : tool_record;
type tool_access_type is access tool_record;
tool_pointer : tool_access_type;
```

3.3.1.3.2 ROD - Operations

PROJECT DATABASE: The basic set of operations (create, query, update, and delete) applicable to this service for the Project Database that are provided by the ERIF are:

Create:
"Insert"

Query:

"Table"
"Domain_Name_Of"
"Range_Name_Of"
"Relationship_Type_Of"
"Domain_Type_Of"
"Range_Type_Of"
"Relationship_Types_Of"
"Components_Of"
"Attribute_Types_Of"
"Cardinality_Of"
"Attribute_Info"
"Datatype_Of"
"="

Update:
"Update"
"Reserve"
"Release"
"Set_Text_Attribute"
"Clear_Text_Attribute"
"Grant"
"Revoke"

Delete:
"Delete"

The exact syntax, parameters, and descriptions of these operations are described in Appendix 6 of [13]. Appendix 4 of the same document describes the use of the ERIF, while Appendix 5 describes the query syntax and semantics of the query string (a proper subset of ANSI SQL) specified in the "Table" operation.

INFRASTRUCTURE DATABASE: The basic set of operations (create, query, update, and delete), applicable to this service for the Infrastructure Database, are provided by DSA routines, and are:

Create:
"DSA_Pointer_To"

Query:
"Check_Folder_Vars" (New DSA routine)
"Display_DSA_Database"
"Display_Object_Folders" (New DSA routine)

"DSA_List_Membership_Count"
"Get_Next_Token" (New DSA routine)
"Is_Empty"
"Length_Of_DSA_List"
"Length_Of_Dynamic_Dataset"
"Next_Dataset_On_DSA_List"
"Obtain"
"On_DSA_List"
"Restore_DSA_Database"
"Switch"
"Through_DSA_List"
"Validate_Keyword" (New DSA routine)
"Validate_User_Name" (New DSA routine)

Update:
"Add_Item_To_List" (New DSA routine)
"Append_DSA_List"
"Change_File_Attributes" (New DSA routine)
"Change_Tool_Position" (New DSA routine)
"Put_On_DSA_List"
"Set_Current_Role" (New DSA routine)
"Set_Tool_Parameters" (New DSA routine)
"Sort_DSA_List"
"Store_In_DSA_Locker"
"Take_Off_DSA_List"

Delete:
"Delete_Item_From_List" (New DSA routine)
"Destroy_Dataset"
"Destroy_DSA_List"

New DSA routines are those that were built upon the basic DSA operations especially for SLCSE.

3.3.1.3.2.1 Example

PROJECT DATABASE:

```
-- ERIF query on relationship.
   ERIF.Table ( Universe => "Person Solves Problem",
                Attributes => "Domain.Access_Name, " &
                              "Relation.Solution, " &
                              "Range.Access_Name"
```

```
                Buffer => The_Attribute_List,
                Current => The_Cursor );
```

INFRASTRUCTURE DATABASE:

```
declare
   role_tool_pointer :
   dsa_data_structures.tool_access_type;
   tool_pointer: dsa_data_structures.tool_access_type;
begin
   role_tool_pointer := new_dataset;
   role_tool_pointer := copy_of_dataset (tool_pointer);
   role_tool_pointer.allowable := true;
-- place new tool record on tool list for this role
   put_on_dsa_list (dataset => role_tool_pointer,
                    list => role_pointer.tools_list,
                    position => bottom );
end;
```

3.3.1.3.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_3_relationship FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_3_relationship (FORWARD) [object_management]
 1- 7_3_relationship depends_on 7_2_data_storage
 -->2- 7_2_data_storage depends_on 11_1_message_delivery
 -->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->2- 7_2_data_storage depends_on 7_16_global_schema
```

-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.1.4 Name Service (7.4)

3.3.1.4.1 ICE - Conceptual

SLCSE provides a Name Service for Project Database, Infrastructure Database, and Project Files Hierarchy objects. The Name Service for each of these kinds of objects will be described in turn at the conceptual level.

PROJECT DATABASE OBJECTS: Entity types possess names that are 12 characters maximum in length, and must start with a letter. Instances of an entity type (called an entity) are named as follows:

<entity_name> ::= <access_name>:<descr_name>:<version>
<access_name> ::= <identifier> (32 characters maximum)
<descr_name> ::= <identifier> (80 characters maximum)
<version> ::= (non-zero sequence of digits)
<identifier> ::= (sequence of letters, digits, and underscores,
starting with a letter)

Relationship types possess compound names that consist of three parts, as follows:

<relationship_name> ::= <domain_name>
<relation_name>
<range_name>
<domain_name> ::= <entity_type_name>
<relation_name> ::= <identifier> (12 characters maximum)
<range_name> ::= <entity_type_name>
<entity_type_name> ::= <identifier> (12 characters maximum)

Relationship type instances (called relationships, or links) do not actually possess names, but are accessible by the names of their types.

Entity and relationship type and instance attributes possess names that are 12 characters maximum in length, and must begin with a letter.

Both entity and relationship types may possess alias names that are 31 characters maximum in length, and must begin with a letter.

Entity and relationship type union names (12 characters maximum) may also be used to reference a set of entity/relationship types and their Subschemas also possess names that are 31 characters maximum in length, and must begin with a letter.

INFRASTRUCTURE DATABASE OBJECTS: Names of objects in a SLCSE Infrastructure Database are variable length strings maintained as components of records. However, not all Infrastructure Database records have name components, and, therefore, the Name Service in SLCSE as applied to the Infrastructure Database is limited. Names are associated with the following objects:

"entity_record"
"environment_root_record"
"file_record"
"folder_record"
"keyword_record"
"peripheral_record"
"personnel_record"
"project_root_record"
"role_record"
"sub_schema_record"
"tool_record"

PROJECT FILES HIERARCHY OBJECTS: Names are associated with file objects within the User Files Hierarchy and the Database Files Hierarchy of directories, and the Name Service is provided by the OS. In SLCSE, this basic OS Name Service is augmented in the following ways:

1. An environment instantiation for a project named, for example, "Project-X", will have the SLCSE$PRJ directory file named "SLCSE_PROJECT-X.DIR".

2. An environment instantiation for a project using a Project Database named, for example, "Project-X-DB", will have a SLCSE$DBDIR directory file named "PROJECT-X-DB.DIR", a parent directory file named "SLCSE.DIR", and subordinate SLCSE$SQL, SLCSE$SDF, and SLCSE$DATABASE directory files named "SQL.DIR", "SDF.DIR", and "DATABASE.DIR", respectively. The SLCSE$TEXT directory file(s) subordinate to the SLCSE$DBDIR directory assumes the name "TEXTn.DIR" where 'n' is a digit ranging from '0' to '9'. Directory files below the SLCSE$TEXT directory assumes a name which is the string conversion of a number ranging from '0' to '9', and directory files

below these directories assume a name which is the string conversion of a number ranging from '00' to '99'.

3. Project Database entity and relationship text attribute files are assigned a unique name, which when elaborated describes the complete path (i.e., location within the Database Files Hierarchy of directories) and file name. For example, if in a Project Database named "Project-X-DB" there is an instance of an entity type named "activity" that has an integer attribute named "key" equal to 01018 and a text attribute named "description", then the complete name of the text attribute file pointed to by the entity would be "[SLCSE.PROJECT-X-DB.TEXT0.0.18]01018CSCI.PURPOSE".

3.3.1.4.1.1 Example

PROJECT DATABASE OBJECT NAMES: The entity type named "activity" is defined that has string attributes named "access_name" and "descr_name". The first version of an instance of this type might be named "T1:Task-1:1", where "T1" is the access name of the entity, "Task-1" is the descriptive name of the entity, and "1" is the version of the entity.

INFRASTRUCTURE DATABASE OBJECT NAMES: A "project_root_record" might have a "project_name" component assigned the value "Project-X".

PROJECT FILES HIERARCHY OBJECT NAMES: See the examples provided above.

3.3.1.4.2 ROD - Operations

PROJECT DATABASE OBJECT NAME OPERATIONS: SLCSE provides all of the basic operations of create, query, update, and delete for the names of Project Database objects (as discussed in the ICE dimension, conceptual description of this service). However, it may not be possible to perform these operations at all times, as illustrated in the following two tables:

Before Environment Instantiation

OPERATIONS
(ON NAME)

OBJECT WITH NAME	Create	Query	Update	Delete
attribute_name	Yes	No	Yes	Yes
entity_alias_name	Yes	Yes	Yes	Yes

entity_name	No	No	No	No
entity_type_name	Yes	Yes	Yes	Yes
entity_type_union_name	Yes	Yes	Yes	Yes
relationship_type_alias_name	Yes	Yes	Yes	Yes
relationship_type_name	Yes	Yes	Yes	Yes
relationship_type_union_name	Yes	Yes	Yes	Yes
subschema_name	Yes	Yes	Yes	Yes

In this case: (1) Create operations are provided by a simple text editor during schema definition, (2) Query operations are provided by "SDL_Convert" and "AnalyzER" (reference [9], [10], and [11]), (3) Update operations are provided by a text editor during schema modification, and (4) Delete operations are also provided by a text editor during schema modification.

After Environment Instantiation

	OPERATIONS (ON NAME)			
OBJECT WITH NAME	Create	Query	Update	Delete
attribute_name	No	Yes	No	No
entity_alias_name	No	Yes	No	No
entity_name	Yes	Yes	Yes	Yes
entity_type_name	No	Yes	No	No
entity_type_union_name	No	Yes	No	No
relationship_type_alias_name	No	Yes	No	No
relationship_type_name	No	Yes	No	No
relationship_type_union_name	No	Yes	No	No
subschema_name	No	Yes	No	No

In this case, create, query, update, and delete operations are provided by the ERIF. It should be noted that the creation and deletion of an entity name (i.e., entity_name) is a by-product of the creation and deletion of the entity itself. That is, an entity cannot exist without a name.

INFRASTRUCTURE DATABASE OBJECT NAMES: The basic operations of create, query, update, and delete for the names of Infrastructure Database objects are provided by DSA, but only apply to records with name components. It is possible, in the case of Infrastructure Database objects, to have a record exist with a name component that has no name value assigned to the component (though this would probably be an application error).

PROJECT FILES HIERARCHY OBJECT NAMES: The basic operations of create, query, update, and delete for the names of file objects are provided by the VMS Operating System (e.g., Create, Directory, Rename, and Delete). It should be noted that the creation and deletion of a file is a by-product of the creation and deletion of the file itself. That is, a file cannot exist without a name.

3.3.1.4.2.1 Example

PROJECT DATABASE OBJECTS: The following is an example of ERIF operations used to query on the name of an entity, and to delete the name (along with the entity). This example follows the case where it is after an environment's instantiation.

```
declare
   Allocation : Cursor;
   Buffer : Attribute_List;
begin
   Table ( Universe => "Requirement",
           Attributes => "Access_Name, Descr_Name, Version",
           Buffer=> Buffer,
           Mode => MODIFY,
           Query => "Access_Name = 'Req1' and " &
                    "Descr_Name = 'Requirement1' and " &
                    "Version = 1"
           Current => Allocation );
   while not End_Of_Table ( Allocation ) loop
      Delete ( Allocation );
   end loop;
   Commit ( Allocation );
   Close_Table ( Allocation );
end;
```

INFRASTRUCTURE DATABASE OBJECTS: The following is an example of the Delete operation for a PERSON record name on a PROJECT list using DSA.

```
take_off_dsa_list (dataset => person_pointer,
                   list => proj_pointer.master_personnel_list);
destroy_dataset (person_pointer);
```

PROJECT FILES HIERARCHY OBJECTS: The following is an example of the update operation of the SLCSE Name Service provided by the OS for file objects.

```
procedure lib_rename_file (
        status : out condition_handling.cond_value_type;
        old_filespec : in string;
        new_filespec : in string;
        default_filespec : in string := string'null_parameter;
        related_filespec : in string := string'null_parameter;
        flags : in integer := integer'null_parameter;
        success_routine : in integer := integer'null_parameter;
        error_routine: in integer := integer'null_parameter;
        confirm_routine : in integer := integer'null_parameter;
        user_arg : in integer := 0;
        old_resultant_name : out string;
        new_resultant_name : out string;
        file_scan_context : in out integer );
```

3.3.1.4.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_4_name FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_4_name (FORWARD) [object_management]
 1- 7_4_name depends_on 7_2_data_storage
 -->2- 7_2_data_storage depends_on 11_1_message_delivery
 -->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->2- 7_2_data_storage depends_on 7_16_global_schema
```

-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.1.5 Distribution/Location Service (7.5)

A SLCSE Project Database is both logically and physically centralized, and therefore, there is no concept of a logically centralized database mapping to a physically distributed database in SLCSE. However, there are facilities that allow for the management of Project Database Objects on distributed computing platforms that are not the actual platform hosting a Project Database. This alternate concept is described more fully in the mapping to the Replication/Synchronization Service of the Reference Model (paragraph 7.12 of the Reference Model).

3.3.1.6 Data Transaction Service (7.6)

3.3.1.6.1 ICE - Conceptual

SLCSE provides a Data Transaction Service for Project Database Objects.

Operations on Project Database objects are transaction-oriented. When a transaction has completed, all changes are "committed" or "saved". If a "commit" involves an operation that fails, all operations of the "commit" are "rolled back" to the point before the "commit".

The Data Transaction Service of SLCSE is facilitated by a working combination of "EditER" (a general-purpose ER editing facility), the ERIF, and the underlying RDBMS.

3.3.1.6.1.1 Example

A SLCSE user modifies two entities, and creates a relationship between them in his/her local buffers. The user then "commits" the changes, and an error occurs because a mandatory attribute value specification for the relationship was not provided. The transaction is "rolled back" to the point before the "commit", and the user is informed of the problem. The user specifies the relationship attribute value, and then "commits" the changes. The "commit" is then successful.

3.3.1.6.2 ROD - Operations

The commit and rollback operations are provided for transactions in SLCSE dealing with Project Database objects. The ERIF provides the following two procedures:

```
procedure Commit (Current : in Cursor;
                  -- Table identifier.
                  Errors : in Error_Block := Standard_Error_Block;
                  -- Where to store errors.
                  );

procedure Rollback (Current : in Cursor;
                  -- Table identifier.
                  Errors : in Error_Block := Standard_Error_Block;
                  -- Where to store errors.
                  );
```

At the front end, EditER utilizes these procedures to implement the Data Transaction Service. At the back end, the ERIF utilizes the underlying operations of the RDBMS to implement these procedures.

3.3.1.6.2.1 Example

EditER's "Save_Error_Clean_Up" procedure commits any updates that were made during a transaction, and then closes the database table (a local buffer). The procedure is called if any exceptions are raised during the save process. This is to insure that changes which were made are committed, and that the buffer is not lost because of an unhandled exception.

```
procedure Save_Error_Clean_Up is
-- Current row of the ERIF table.
   current : ERIF.cursor;
begin
-- Commit the changes.
   ERIF.Commit (current);
 -- Close the table.
   ERIF.Close_Table (current);
end Save_Error_Clean_Up;
```

3.3.1.6.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on"

relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_6_data_transaction FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_6_data_transaction (FORWARD) [object_management]
1- 7_6_data_transaction depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.1.7 Concurrency Service (7.7)

3.3.1.7.1 ICE - Conceptual

The Concurrency Service of SLCSE is provided by the operations of the Entity-Relationship InterFace (ERIF) for Project Database objects.

3.3.1.7.1.1 Example

The SLCSE Project Management System (SPMS) is an example of an application that utilizes the Concurrency Service of SLCSE. When an entity is checked out from the database by user "A", an ERIF operation is used to lock entity "E". User "A" then manipulates a copy of entity "E". If user "B" attempts to access entity "E" from an application, through the ERIF, while user "A" has that entity locked, access is denied. Only when user "A" commits the changes to entity "E" will entity "E" be unlocked and accessible to user "B".

3.3.1.7.2 ROD - Operations

The ERIF provides a lock and an unlock operation, which at the application level are called "reserve" and "release", respectively. These operations apply to both entities and relationships, and are described in the ERIF specification as follows:

```
procedure Reserve (Current : in Cursor;
                   -- Table identifier.
                   Errors : in Error_Block := Standard_Error_Block);
                   -- Where to store errors.
                   );

procedure Release (Current : in Cursor;
                   -- Table identifier.
                   Errors : in Error_Block := Standard_Error_Block);
                   -- Where to store errors.
                   );
```

Both the "Delete" and "Update" operations of the ERIF use an internal "Is_Locked" boolean function (i.e., a function that is not visible to an external application) to ensure that changes are not made to Project Database objects that are locked.

3.3.1.7.2.1 Example

```
declare
   Req : Cursor;
begin
-- Find the requirement named "SEARCHMODE".
   Table(Universe => "Requirement",
         Current=> Req,
         Query => "Access_Name = 'SEARCHMODE'");
         Current=> Req);
-- Reserve the entity retrieved by the query.
   Reserve(Req);
   Commit(Req);
   Close_Table(Req);
end;
```

3.3.1.7.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between

services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_7_concurrency FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_7_concurrency (FORWARD) [object_management]
 1- 7_7_concurrency depends_on 7_2_data_storage
 -->2- 7_2_data_storage depends_on 11_1_message_delivery
 -->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->2- 7_2_data_storage depends_on 7_16_global_schema
 -->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
 -->3- 7_16_global_schema depends_on 7_1_data_model
```

This service is required by the following service that is provided by SLCSE:

```
TRACE ON ENTITY 7_7_concurrency BACKWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_7_concurrency (BACKWARD) [object_management]
 1- 15_1_6_consistency_mgt depends_on 7_7_concurrency
```

3.3.1.8 Process Support Service (7.8)

SLCSE does not provide the Process Support Service as described in the Reference Model. However, the VMS operating system upon which SLCSE is currently implemented does. The VMS operating system provides for prioritized process scheduling queues, process control blocks (for process state information), process state transition control (e.g., make wait, execute, prepare for execution, swap out, swap in, etc.), input/output process requests handling, and a number of utilities for querying about process information (e.g., the accounting utility, the show process utility, and the monitor process

utility) and for changing processes (e.g., the set utility). SLCSE provides tool integration services for the incorporation of such utilities as tools in an environment instantiation (e.g., a tool that was integrated into SLCSE was the CPU tool, which, at the VMS level, translates to the "monitor process/topcpu" utility).

3.3.1.9 Archive Service (7.9)

The VMS Operating System (OS), upon which SLCSE is currently implemented, provides a complete Archive Service for files stored on disk. Through a combination of the VMS backup (to tape from disk) utility (reference [19]), and the Backup Service of SLCSE (reference service mapping to Reference Model section 7.10, Backup Service), all data can be moved off-line, and restored.

3.3.1.10 Backup Service (7.10)

3.3.1.10.1 ICE - Conceptual

The "Digital Command Language (DCL) Interface Utilities" of SLCSE can be used to restore the Project Database (as well as the contents of text attribute files within the associated Database Files Hierarchy of the Project Files Hierarchy) for an environment instantiation after a media failure (refer to [13]).

SMARTSTAR (a 4GL RDBMS software package necessary for SLCSE operation) also provides utilities to backup a Project Database (exclusive of the contents of text attribute files within the Database Files Hierarchy). Refer to [12].

The ShareBase Server (an intelligent database machine, which is the hardware implementation option for the underlying RDBMS of SLCSE) also provides utilities to backup a Project Database (exclusive of the contents of text attribute files within the Database Files Hierarchy). Refer to [6].

The ShareBase Server and Rdb/VMS (the software implementation option for the underlying RDBMS of SLCSE) provide transaction logging for the recovery of data since the last complete backup of a SLCSE Project Database.

The Operating System (OS) supporting SLCSE provides a backup utility that can be used to restore the entire Project Files Hierarchy for an environment instantiation in the event of a media failure (this includes the

Database Files Hierarchy files for Rdb/VMS, text attribute files, Infrastructure Database files, etc.). Reference [19].

3.3.1.10.1.1 Example

For a SLCSE Project Database associated with an environment instantiation, the DCL Interface Utility "database unload" is used to dump the entire contents of the database into a predefined ASCII text format stored in a file on disk. The VMS Backup Utility is then used to archive this file, and the entire Project Files Hierarchy (including the Infrastructure Database) from disk to tape (or, alternatively, these files could be backed up to another disk). Media failure may occur to the disk holding the Project Database (be it a ShareBase Server disk or an Rdb/VMS disk), in which case, the Project Database is backed up to disk from tape (or, alternatively, to disk from disk), and then the DCL Interface Utility "database load" is used to restore the database from the ASCII text file. RDBMS utilities would then be used to recover additional data lost since the last database backup. Restoration of the Project Files Hierarchy is done simply by using the VMS Backup Utility (tape to disk, or, alternatively, disk to disk).

3.3.1.10.2 ROD - Operations

The SLCSE Backup Service operations fall into four general categories: Copy, Delete, Create, and Update. The Copy operation, in this case, is what is typically called "making a backup copy". For the Backup Service, "Copy" does not necessarily mean that a copy of an object (e.g., a database or a file) is an exact duplicate (i.e., it may be stored on a different type of media, and/or be formatted differently than the original). The Delete operation may be required to eliminate corrupted objects after a media failure. The Create operation may be required to replace deleted/corrupted objects with new ones. The Update operation is, in this case, what is typically called "restoring from a backup copy".

The primary Backup Service operations provided by the DCL Interface Utilities for a Project Database are listed below:

Copy:

"Database Unload
[/<qualifier_list>]"

Update:

"Database Load
[/<qualifier_list>]"

SMARTSTAR utilities provide the following Backup Service operations for a Project Database:

Copy:

"DUMPDBVAX" - Backup a database to a VAX file.
"DUMPDBTAP" - Backup a database to a ShareBase tape.
"DUMPTRVAX" - Backup a transaction log to a VAX file.
"DUMPTRTAP" - Backup a transaction log to a ShareBase tape.

Delete:

"DUMPDB" - Delete a database.

Create:

"CREATEDB" - Creates an empty database.

Update:

"LOADDBVAX" - Restore database contents from a VAX file.
"LOADDBTAP" - Restore database contents from a ShareBase tape.
"TRANUPVAX" - Restore and roll forward transaction log from a VAX file.
"TRANUPTAP" - Restore and roll forward transaction log from a ShareBase tape.

The ShareBase Server RDBMS provides the following Backup Service operations for a Project Database:

Copy:

"IDMDUMP" - Backup database and/or transaction log to file or tape.

Delete:

"SQL DROP" - Delete a database.

Create:

"SQL CREATE" - Create an empty database.

Update:

"IDMLOAD" - Restore database and/or transaction log from file or tape.
"IDMROLLF" - Roll forward a transaction log.

The VAX Rdb/VMS Management Utility (RMU) provides the following Backup Service operations for a Project Database:

Copy:
"RMU/BACKUP" - Backup a database and its transaction log to a file.

Delete:
"SQL DROP" - Delete a database.

Create:
"SQL CREATE" - Create an empty database.

Update:
"RMU/RESTORE" - Restore a database and its transaction log from a backup file.
"RMU/RECOVER" - Roll forward a transaction log after restore.

The VMS Backup Utility provides both Copy and Update operations (disk to tape, tape to disk, disk to disk, and tape to tape) for all files, and has the following command format:

"Backup [/<command_qualifier_list>]
<input_specifier> [/<input_specifier_qualifier_list>]
<output_specifier> [/<output_specifier_qualifier_list>]"

The VMS Delete Utility may also be used to delete corrupted files before an Update operation, while the VMS Create Utility may be used to create directories and empty files. These commands have the following formats:

"Delete [/<command_qualifier_list>] <file_specification>
"Create [/<command_qualifier_list>] <file_specification>

The VMS Create Utility is rarely used directly by the user during SLCSE Backup Service operations, since the VMS Backup Utility utilizes the VMS Create Utility to perform most of the necessary Create operations.

3.3.1.10.2.1 Example

The following is an example of SLCSE Backup Service operations:

```
$DATABASE UNLOAD /SCHEMA="UNIVERSAL" -
    /OUT="DUA0:[SLCSE.DATABASE]DATABASE.OUT"
    /LOG="DUA0:[SLCSE.DATABASE]DATABASE.LOG"
$BACKUP DUA0:[SLCSE.DATABASE...]*.*;*, -
    DUA0:[USERA.SLCSE_PROJECTX...]*.*;*, -
```

```
        DUA0:[USERB.SLCSE_PROJECTX...]*.*;* -
        MFA0:PROJECTX.BCK/LABEL=PROJECTX
$DELETE DUA0:[SLCSE.DATABASE...]*.*;*
$DELETE DUA0:[USERA.SLCSE_PROJECTX...]*.*;*
$DELETE DUA0:[USERB.SLCSE_PROJECTX...]*.*;*
$BACKUP MFA0:PROJECTX.BCK -
        /SAVE_SET -
        /SELECT=([SLCSE.DATABASE...]*.*;*) -
        DUA0:[SLCSE.DATABASE...]*.*;*
$BACKUP MFA0:PROJECTX.BCK -
        /SAVE_SET -
        /SELECT=([USERA.SLCSE_PROJECTX...]*.*;*) -
        DUA0:[USERA.SLCSE_PROJECTX...]*.*;*
$BACKUP MFA0:PROJECTX.BCK -
        /SAVE_SET -
        /SELECT=([USERB.SLCSE_PROJECTX...]*.*;*) -
        DUA0:[USERB.SLCSE_PROJECTX...]*.*;*
$SQL SYSTEM
SQL> DROP PROJECT-X;
SQL> EXIT;
$CREATEDB PROJECTX-DB
$DATABASE LOAD /IN="DUA0:[SLCSE.DATABASE]DATABASE.OUT" -
        /LOG="DUA0:[SLCSE.DATABASE]DATABASE.LOG"
```

3.3.1.10.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_10_backup FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.
```

7_10_backup (FORWARD) [object_management]
1- 7_10_backup depends_on 7_23_data_interchange
-->2- 7_23_data_interchange depends_on 7_2_data_storage
-->3- 7_2_data_storage depends_on 11_1_message_delivery
-->3- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->3- 7_2_data_storage depends_on 7_16_global_schema
-->4- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->4- 7_16_global_schema depends_on 7_1_data_model

This service is required by the following service that is provided by SLCSE:

TRACE ON ENTITY 7_10_backup BACKWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_10_backup (BACKWARD) [object_management]
1- 7_9_archive depends_on 7_10_backup

3.3.1.11 Derivation Service (7.11)

3.3.1.11.1 ICE - Conceptual

SLCSE provides for the definition of "entity views". An entity view is a representation of an entity type in a form other than the physical representation. Entity views provide a method for restriction, or abbreviation, of physical structure. Entity views can be defined in the Schema Definition Language (SDL) to enhance both the clarity of the data model and the efficiency of applications that frequently reference a given combination of attributes in an entity type.

Entity views, however, possess attributes that are derived from a combination of attributes of the entity type from which the entity view is constructed. That is, an entity view is an object which possesses a subset of the attributes derived from an entity type, and is simpler than that entity type. None the less, entity views *are* derived from entity types. When instances of an entity type, that has an entity view, are created, an instance of the entity view is automatically derived.

When the data model for a environment instantiation is defined in SDL, it is then compiled. Derived in this process are SQL files and ERIF run-time files. When the SQL files are interpreted via SMARTSTAR, the physical database for the environment is derived.

Special tools integrated into an environment may take advantage of the Derivation Service provided by the underlying RDBMS. For example, the ShareBase Server provides for the definition and creation of "views" (which are different from the "entity views" just described) and "stored commands". Execution of a stored command (similar to a macro) or a query on a view can produce conceptual objects with derived values not physically stored anywhere in the database. SLCSE does not, however, currently provide services at the Entity-Relationship (ER) level which utilize the views or stored commands provided at the relational level.

3.3.1.11.1.1 Example

The following is an example of an entity type definition in SDL from which the subsequent entity view is derived:

```
entity type Contract is
   Number : string (20);
   Effective_Date : time;
   Total_Amount : float;
   Total_Amount_Unit : Currency_Unit := Dollars;
end entity;

view type Contract_Cost of Contract is
   Dollar_Amount : Total_Amount_Unit;
end view;
```

3.3.1.11.2 ROD - Operations

The basic operations involving the Derivation Service of SLCSE are Create (i.e., create an entity view definition, create an entity view instance, create SQL and ERIF run-time files from SDL, create physical Project Database from SQL, and create view or stored command), Delete (i.e., delete an entity view definition, delete an entity view instance, delete SQL and ERIF run-time files, delete a physical Project Database, and delete a view or a stored command), Update (i.e., update an entity view definition, update an entity view instance, and update a view or a stored command), and Query (i.e., query about an entity view definition, query about an entity view instance, and query a view or execute a stored command).

It should be noted that the creation or update of an entity view is done by creating or updating the entity from which that entity view is derived.

3.3.1.11.2.1 Example

Referring to the example given for the "ICE - Conceptual" Service Dimension Form, create the entity view definition of the example in an SDL file using a text editor. Compile the SDL file to create SQL and ERIF run-time files, and interpret the SQL to create the physical Project Database. Run an application that utilizes the SLCSE EditER, and create an instance of the entity type Contract named "PROJECTX:PROJECTX:1" with a Total_Amount attribute value of 100000.00 and a Total_Amount_Unit attribute value of "Dollars". Query on the entity view instance attribute "PROJECTX:PROJECTX:1.Dollar_Amount" to discover a value of "Dollars". In frustration at discovering undesired results, delete the instance of the entity type Contract (deleting also the entity view) named "PROJECTX:PROJECTX:1", and leave the application.

Using an editor, update the entity view definition for Contract_Cost to include the entity type Contract attribute Total_Amount, but renamed for the entity view as Dollar_Amount, replacing the old Dollar_Amount attribute with this one. Delete the old SQL, ERIF run-time files, and the physical Project Database, and create new ones using the updated SDL as input. Repeat the exercise with the application, and query on the entity view instance attribute "PROJECTX:PROJECTX:1.Dollar_Amount" to discover the desired value of 100000.00 for the entity view Contract_Cost.

3.3.1.11.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_11_derivation FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_11_derivation (FORWARD) [object_management]
1- 7_11_derivation depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is required by none of the mapped services.

3.3.1.12 Replication/Synchronization Service (7.12)

3.3.1.12.1 ICE - Conceptual

SLCSE provides a Replication/Synchronization Service for Project Database Objects.

Within a heterogeneous network of computer nodes, it is possible for remote applications to "check-out" a subset of information contained in a SLCSE Project Database, manipulate that subset of information, and to "check-in" the modified information to the Project Database. Database integrity is important in a multi-user environment such as SLCSE, and therefore, it is possible to "lock" the instances that are checked-out, and "unlock" them when they are checked back in.

This facility is provided using a client-server architecture, where a client process on the remote node requests (over the network) a subset of data from a server process running on the host platform of the Project Database. Both the client and the server work through an interface to the Entity-Relationship Interface (ERIF) called the High Level ERIF (HLERIF), which is a highly portable Ada package. This package provides the capability to operate within the memory constraints of the remote computer via efficient, self-automated "swapping" operations to and from the local file space, and also provides data file formats that can be relatively easily transformed into the format required for use by a native application.

3.3.1.12.1.1 Example

The SLCSE Project Management System (SPMS) is an example of a system that uses the Replication/Synchronization Service provided by SLCSE, as described above. The SPMS contains Commercial Off-The-Shelf (COTS) project management tools implemented on a Macintosh workstation that communicates to the SLCSE Project Database over the network. The Project

Management Assistant (PMA) facet of the RL Knowledge-Based Software Assistant (KBSA) program and the QUality Evaluation System (QUES) are other examples where this service was applied for tools implemented on workstations.

3.3.1.12.2 ROD - Operations

The basic set of operations (create, query, update, and delete) applicable to this service for the Project Database are provided by the "High-Level Entity-Relationship Interface (HLERIF)", and are listed below:

Create:
"Add_Monitor_Action"
"Duplicate"
"Insert"

Query:
"Attribute_Error_Message"
"Collection_Error_Message"
"Condition"
"Count"
"Finalize"
"Find_Backward"
"Find_Forward"
"First"
"Get"
"Get_Current"
"Get_Error"
"Get_Instance_Storage"
"Get_Monitor_Action"
"Get_Next_Event"
"Get_Swap_Count"
"Get_Test_Error"
"Goto_First"
"Goto_Last"
"Goto_Next"
"Goto_Previous"
"Hlerif_Error_Message"
"Image"
"Initialize"
"Instance_Error_Message"
"Last"
"Local_Collection_Exists"

"Login"
"Logout"
"More_Errors"
"More_Monitor_Actions"
"More_Test_Errors"
"Print"
"Retrieve_From_Local"
"Retrieve_From_Slcse"
"Retrieve_Monitor_Actions"
"Test_Servers"
"Value"

Update:
"Save_To_Local"
"Save_To_Slcse"
"Set"
"Set_Instance_Storage"
"Set_Matching"
"Sort"

Delete:
"Delete"
"Destroy"
"Destroy_Local_Collection"
"Remove_Monitor_Action"

3.3.1.12.2.1 Example

On a Macintosh workstation, retrieve from SLCSE all entities of the PROBLEM entity type, passing the Retrieve_From_Slcse operation the boolean value of 'True' for the "Reserve" parameter. This locks these entities while the retrieving application operates on the local entity collection until the Save_To_Slcse operation is used with a boolean value of 'True' for this operation's "Release" parameter.

3.3.1.12.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship

information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_12_replication FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_12_replication (FORWARD) [object_management]
 1- 7_12_replication depends_on 7_2_data_storage
 -->2- 7_2_data_storage depends_on 11_1_message_delivery
 -->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->2- 7_2_data_storage depends_on 7_16_global_schema
 -->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
 -->3- 7_16_global_schema depends_on 7_1_data_model
```

This service is required by the following service that is provided by SLCSE:

```
TRACE ON ENTITY 7_12_replication BACKWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_12_replication (BACKWARD) [object_management]
 1- 15_1_6_consistency_mgt depends_on 7_12_replication
```

3.3.1.13 Access Control/Security (7.13)

3.3.1.13.1 ICE - Conceptual

SLCSE provides the service of Access Control/Security for Project Database, Infrastructure Database, and Project Files Hierarchy Objects. These shall be discussed in turn for this service.

PROJECT DATABASE: Access Control is provided on Project Database information at three levels of data granularity. A user is provided access (or denied access) to Project Database information at the level of: (1) database entry, (2) subschema entry, and (3) entity/relationship/attribute (ERA) entry. Access control is specified/modified for a user within the SLCSE Environment Manager (SEM) (at the database, subschema, and entity entry

level) before an environment is instantiated, and, at the ERA level, may be fine-tuned using the facilities provided by the RDBMS (e.g., SQL). Access control is enforced via a combination of the SLCSE Command Executive (CE) and the underlying RDBMS for a SLCSE Project Database. Information regarding access control is stored: (1) for database entry, in the relational database, (2) for subschema entry, in the Infrastructure Database, and (3) for ERA entry, in the Infrastructure Database (entity entry only) and in the relational database (ERA entry inclusive).

INFRASTRUCTURE DATABASE: Access Control is provided for an Infrastructure Database file using the access control services of the VMS Operating System (OS), as described below for Project Files Hierarchy objects. It should be noted that information for a user's access privileges to Project Database subschemas and entities, and tools integrated into an environment instantiation is recorded in the Infrastructure Database using the SEM. This information is used by the SLCSE CE and some SLCSE applications to govern access control to Project Database information and an environment's toolset.

PROJECT FILES HIERARCHY: Access Control is provided for file objects using the access control services of the VMS Operating System (OS). These services include the following two mechanisms: (1) file protections, and (2) Access Control Lists (ACLs). File protections include read, write, execute, and delete for System level access (i.e., special privilege account owners), World level access (i.e., all account owners), Group level access (i.e., all account owners with the same group number as the owner of the object), and Owner level access (i.e., the account owner who owns the object). An ACL is associated with an object, and consists of a number of ACL Entries (ACEs) that grant or deny access to a particular object. Each ACE specifies a user or a group of users who have access to a particular object (regardless of whether or not they are the owner of the object), and the type of access permitted (i.e., read, write, execute, delete, control, none). Refer to [8].

3.3.1.13.2 ROD - Operations

The basic operations Create, Update, Query, and Delete provided for this service in SLCSE are described for Project Database, Infrastructure Database, and Project Files Hierarchy objects as follows:

PROJECT DATABASE: The following operations apply to Project Database objects access control and security. SEM generates access control SQL for interpretation, and also generates the Infrastructure Database (for subschema access control) via Dynamic Storage Allocation (DSA) operations.

The SLCSE CE and applications use DSA and ERIF operations, and rely on RDBMS access control enforcement at the ERA level.

Create:
"SEM - DEFINE ENVIRONMENT"
(Default subschema access control per user role)
"SEM - DEFINE PROJECT"
(Subschema and entity access control per user of an environment instantiation)
"DSA - NEW_DATASET"
"DSA - PUT_ON_DSA_LIST"
(Subschema access control)
"SQL - GRANT"
(ERA access control)
"ERIF - GRANT"
(ER access control)

Update:
"SEM - MODIFY ENVIRONMENT"
(Default subschema access control per user role)
"SEM - MODIFY PROJECT"
(Subschema and entity access contol per user of an environment instantiation)
"DSA - PUT_ON_DSA_LIST"
"DSA - TAKE_OFF_DSA_LIST"
(Subschema access control)
"SQL - GRANT"
(ERA access control)
"SQL - REVOKE"
(ERA access control)
"ERIF - GRANT"
(ER access control)
"ERIF - REVOKE"
(ER access control)

Query:
"SEM - MODIFY ENVIRONMENT"
(Default subschema access control per user role)
"SEM - MODIFY PROJECT"
(Subschema and entity access contol per user of an environment instantiation)
"DSA - OBTAIN"
(Subschema access control)

"SQL - SELECT"
(ERA access control)
"ERIF - AVAILABLE SCHEMAS"
(Subschema access control)

Delete:
"SEM - MODIFY ENVIRONMENT"
(Default subschema access control per user role)
"SEM - MODIFY PROJECT"
"SEM - DELETE PROJECT"
(Subschema and entity access contol per user of an environment instantiation)
"SQL - REVOKE"
(ERA access control)
"DSA - DESTROY_DATASET"
"DSA - DESTROY_DSA_LIST"
"DSA - TAKE_OFF_DSA_LIST"
(Subschema access control)
"ERIF - REVOKE"
(ER access control)

INFRASTRUCTURE DATABASE: The following operations apply to Infrastructure Database objects used for subschema and tool access control and security. These operations are provided by DSA, and used by SEM, the SLCSE CE, and SLCSE applications.

Create:
"SEM - DEFINE ENVIRONMENT"
(Default subschema and tool access control per user role)
"SEM - DEFINE PROJECT"
(Tool, subschema, and entity access contol per user of an environment instantiation)
"DSA - Copy_Of_Dataset"
"DSA - Copy_Of_DSA_List"
"DSA - Create_DSA_Database"
"DSA - Create_DSA_Storage_Area"
"DSA - Define_DSA_Data_Type"
"DSA - DSA_Locker"
"DSA - DSA_Pointer_To"
"DSA - New_Dataset"
"DSA - New_Dynamic_Dataset"
"DSA - Save_DSA_Database"
"DSA - Template"

Query:
"SEM - MODIFY ENVIRONMENT"
(Default subschema and tool access control per user role)
"SEM - MODIFY PROJECT"
(Tool, subschema, and entity access contol per user of an environment instantiation)
"DSA - Check_Folder_Vars" (New DSA routine)
"DSA - Display_DSA_Database"
"DSA - Display_Object_Folders" (New DSA routine)
"DSA - DSA_List_Membership_Count"
"DSA - Get_Next_Token" (New DSA routine)
"DSA - Initialize_Menus_From_Database" (New DSA routine)
"DSA - Is_Empty"
"DSA - Length_Of_DSA_List"
"DSA - Length_Of_Dynamic_Dataset"
"DSA - Next_Dataset_On_DSA_List"
"DSA - Obtain"
"DSA - On_DSA_List"
"DSA - Query_For_Item_Type" (New DSA routine)
"DSA - Restore_DSA_Database"
"DSA - Restore_DSAbase" (New DSA routine)
"DSA - Switch"
"DSA - Through_DSA_List"
"DSA - Validate_Keyword" (New DSA routine)
"DSA - Validate_User_Name" (New DSA routine)

Update:
"SEM - MODIFY ENVIRONMENT"
(Default subschema and tool access control per user role)
"SEM - MODIFY PROJECT"
(Tool, subschema, and entity access contol per user of an environment instantiation)
"DSA - Add_Item_To_List" (New DSA routine)
"DSA - Append_DSA_List"
"DSA - Change_File_Attributes" (New DSA routine)
"DSA - Change_Tool_Position" (New DSA routine)
"DSA - Compress_DSA_Database"
"DSA - Dataset_At"
"DSA - Expand_Dynamic_Dataset"
"DSA - Lock_Dataset"
"DSA - Lock_DSA_List"
"DSA - Put_On_DSA_List"

"DSA - Reduce_Dynamic_Dataset"
"DSA - Set_Current_Role" (New DSA routine)
"DSA - Set_Tool_Parameters" (New DSA routine)
"DSA - Sort_DSA_List"
"DSA - Store_In_DSA_Locker"
"DSA - Switch_Roles" (New DSA routine)
"DSA - Take_Off_DSA_List"
"DSA - Translate_Menu_To_Keyword" (New DSA routine)
"DSA - Unlock_All_Implicitly_Locked_Datasets"
"DSA - Unlock_Dataset"

Delete:
"SEM - MODIFY ENVIRONMENT"
(Default subschema and tool access
control per user role)
"SEM - MODIFY PROJECT"
"SEM - DELETE PROJECT"
(Tool, subschema, and entity access contol per user of an environment instantiation)
"DSA - Delete_Item_From_List" (New DSA routine)
"DSA - Destroy_Dataset"
"DSA - Destroy_DSA_List"

Operations on the file itself that contains the Infrastructure Database are described below for operations that apply to Project Files Hierarchy objects (i.e., files).

PROJECT FILES HIERARCHY: The following operations apply to the access control and security of Project Files Hierarchy objects, and are provided by the VMS OS. These operations are documented in [27].

Create:
"CREATE/DIRECTORY <directory_file_specification>"
"CREATE/OWNER_UIC=<uic> <file_specification>"
"CREATE/PROTECTION=<code> <file_specification>"
"SET ALC=<ACE_list> <file_specification>"

Update:
"SET ALC=<ACE_list> <file_specification>"
"SET FILE/OWNER_UIC=<uic> <file_specification>"
"SET PROTECTION=<default_code>"
"SET PROTECTION=<code> <file_specification>"

Query:
"DIRECTORY/ALC <file_specification>""
"DIRECTORY/OWNER <file_specification>"
"DIRECTORY/PROTECTION <file_specification>""
"SHOW ACL <file_specification>"
"SHOW PROTECTION"

Delete:
"DELETE <file_specification>"
"SET ALC=<ACE_list> <file_specification>"
"SET FILE/OWNER_UIC=<uic> <file_specification>"
"SET PROTECTION=<default_code>"
"SET PROTECTION=<code> <file_specification>"

3.3.1.13.2.1 Example

The following is an example of the operations involved in establishing security in an environment instantiation.

1. Use SEM to establish the default tools and subschemas to be assigned to each user role defined by the framework.

2. Use SEM to define a project environment. Add user Jones, and use the default tool and subschema access settings defined for the framework. Set the read, write, execute, and delete privileges for each entity type in the Project Database. Save the definition. This results in access control SQL file, Infrastructure Database file, and User Files Hierarchy (of the Project Files Hierarchy) creation.

3. Use the SQL Interpreter to read the access control SQL file (which contains various GRANT commands).

4. Use the CREATE and SET operations of VMS to establish the desired file protections and ACLs for Project Files Hierarchy objects.

5. User Jones runs SLCSE to find that he is denied access to certain portions of the Project Database at the subschema and ERA levels, but has access to the appropriate portions of the Project Database, as well as to his project files.

3.3.1.13.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between

services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_13_access_control FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_13_access_control (FORWARD) [object_management]
1- 7_13_access_control depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is required by the following service that is provided by SLCSE:

TRACE ON ENTITY 7_13_access_control BACKWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_13_access_control (BACKWARD) [object_management]
1- 13_2_security_control depends_on 7_13_access_control

3.3.1.14 Constraint/Inconsistency Management (7.14)

3.3.1.14.1 ICE - Conceptual

PROJECT DATABASE: Constraint Management is provided by the definition of the data types of attributes of entity and relationship types, and the enforcement of the values associated with them within an environment instantiation. In addition, the cardinality of relationship types are also defined in the data model, and subsequently enforced.

SLCSE does not provide for Inconsistency Management except for the ability to dynamically enable and disable constraints on Project Database access control.

INFRASTRUCTURE DATABASE: Constraint Management is provided by the definition of the data types of components stored in lists of records within an Infrastructure Database. An attempt by an application to violate these constraints would cause an exception to be raised in it's run-time environment, or would totally prevent the compilation of the application prior to run-time.

Inconsistency Management for the objects of an Infrastructure Database is not provided, and should not apply, given the strong typing of Ada data types.

3.3.1.14.1.1 Example

PROJECT DATABASE:

The boolean entity type attribute called "Baselined" is constrained to the values of "True" and "False".

Access control on an entity type can be dynamically changed for a particular user from "Granted" to "Revoked", or visa versa.

INFRASTRUCTURE DATABASE:

The TOOL_RECORD record has a boolean data type component called "Available" which is constrained to the values of "True" and "False".

3.3.1.14.2 ROD - Operations

PROJECT DATABASE: Operations for Constraint/Inconsistency Management are provided by the underlying RDBMS, and are propagated to SLCSE applications through the ERIF. These operations fall into the general categories of Create, Update, Query, and Delete, but, when considering this service, might be called "guarded" operations. That is, a guarded update operation is an operation that protects against illegal updates to entities and relationships (e.g., assigning a string value to an integer type attribute).

INFRASTRUCTURE DATABASE: Operations for Constraint/Inconsistency Management are provided by the compile-time and

run-time environments of an application using DSA routines, and are provided by the Operating System.

3.3.1.14.2.1 Example

PROJECT DATABASE:

The following is an example of a guarded update operation, as expressed in SQL (much like the ERIF would translate an ER command from an application to a form understood by the RDBMS).

```
$ SQL/DBTYPE=IDM
1) OPEN DATABASE F111;
2) UPDATE CSCI
        SET VERSION = 'BAD'
        WHERE ACCESS_NAME = 'MC OFP';
```

It should be impossible to update the VERSION integer data type attribute of an instance of the entity type CSCI with the string value of 'BAD'. The ShareBase Server RDBMS for SLCSE responds as follows:

%IDM-E-IDM019, Result for field: "version" has wrong type

and the illegal update is protected against.

INFRASTRUCTURE DATABASE:

The following is an example of a guarded update operation, as expressed in Ada (as a SLCSE application might unsuccessfully attempt).

```
declare
   tool_pointer : dsa_data_structures.tool_access_type;
   bad_var: integer := 96;
begin
   tool_pointer := new_dataset;
   tool_pointer.available := bad_var;
```

It should be impossible to update the AVAILABLE boolean data type component of a TOOL_RECORD record with the integer value of 96. The Ada compiler of the VMS Operating System would respond with an error message similar to the following:

%ADAC-E-ASSIGNNERESTYP, Result type BOOLEAN in predefined STANDARD of variable tool_pointer.available at line 2 is not the same as type INTEGER in predefined STANDARD of variable bad_var at line 3 [LRM 5.2(1)]

and the illegal update is protected against.

3.3.1.14.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_14_constraint_mgt FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_14_constraint_mgt (FORWARD) [object_management]
 1- 7_14_constraint_mgt depends_on 7_16_global_schema
 -->2- 7_16_global_schema depends_on 15_1_2_common_data_descr
 -->2- 7_16_global_schema depends_on 7_1_data_model
```

This service is not required by any of the mapped services.

3.3.1.15 Function Attachment (7.15)

SLCSE does not provide for Function Attachment to objects, as none of its data models are truly object-oriented. In SLCSE, Function Attachment to objects would be done, alternatively, through tool integration into an environment instantiated from the framework. The tools would then provide the functions associated with objects.

3.3.1.16 Global/Canonical Schema (7.16)

3.3.1.16.1 ICE - Conceptual

The Global/Canonical Schema provided by SLCSE is in a combination of the instances of: (1) the Project Database Model, (2) the Infrastructure Database Model, and (3) the Project Files Hierarchy Model, for a particular instantiation of an environment from the SLCSE framework. It should be noted that the Entity-Relationship (ER) data model schema used to define all of the data objects associated with a particular software development project (e.g., requirements, PDL, test cases, source code, tasks, schedules, resources, milestones, etc.) is the primary schema for SLCSE.

PROJECT DATABASE MODEL SCHEMA: The Project Database Model is defined using a formal language called the Schema Definition Language (SDL). SDL is used to define entity types and relationship types, and the attributes of those types of objects. In addition, SDL allows the logical grouping of entity and relationship types into subschemas, and permits the definition of relationship types between entity types defined in different subschemas. Relationship type definitions include a domain entity type that has a relation to a range entity type. A comprehensive example of a Global/Canonical Schema for the data objects associated with DOD-STD-2167A, as defined in Appendix 1 of [13].

INFRASTRUCTURE DATABASE MODEL SCHEMA: SLCSE provides a list- based data model schema which is used to define all of the data objects that are necessary for operations dealing with both the instantiation of environments from the SLCSE framework (i.e., SEM operation) and and a particular environment instantiation (i.e., SLCSE operation). This information consists of such things as nodes, projects, peripherals, personnel, roles, tools, files, rules, subschemas, entities, project database access privileges, etc. A SLCSE Infrastructure Database Model Schema is defined in Appendix 2 of [13].

PROJECT FILES HIERARCHY MODEL SCHEMA: SLCSE provides a hierarchical file structure schema for the storage of file objects associated with a particular environment instantiation. This schema consists of three hierarchies: (1) a User Files Hierarchy, (2) a Database Files Hierarchy, and (3) a Maintenance Files Hierarchy. User file, database file, and maintenance file objects are all project file objects associated with an environment instantiation of the SLCSE framework, and the Project Files Hierarchy Model Schema provides the structure for the storage and management of file objects within that instantiation.

For a given environment instantiation, there may be several User Files Hierarchies depending on how many users are assigned to a given project. The User Files Hierarchy is used for the storage and management of files owned by a SLCSE user, and begins with the "SYS$SCRATCH" (or "root") directory of a particular SLCSE user. Below the root directory is a "SLCSE$PRJ" (or "project") directory (associated with a particular environment instantiation) where files essential to SLCSE operation are stored (e.g., an Infrastructure Database file). Below the project directory is one or more "SLCSE$ROL" (or "role") directories (a user can have up to 18 distinct role directories. Within a role directory are static file objects owned by the user. Below a role directory are "SLCSE$WRK" (or "work") and "SLCSE$BCH" (or "batch") directories, where dynamic file objects reside. Dynamic file objects within a work directory are subject to interactive tool manipulations, while those within a batch directory are subject to batch job (or background process) tool manipulations. Also below a role directory there may be "SLCSE$FOLDER" (or "folder") directories for the placement of static file objects. A folder directory may contain other folder directories, and are provided in this schema to allow a user-defined hierarchical organization of SLCSE user file objects. The Users File Hierarchy is described in [14].

The Database Files Hierarchy is used for the storage and management of files pointed to by the Project Database associated with a particular environment instantiation. The Database Files Hierarchy consists of three hierarchies: (1) the Main Hierarchy, (2) the Configuration Management (CM) Hierarchy, and (3) the Software Development Folder (SDF) Hierarchy.

The Main Hierarchy begins with a "SLCSE" directory which contains "SLCSE$DBDIR" directory named after the Project Database. Below the SLCSE$DBDIR directory are the "SLCSE$SQL", "SLCSE$SDF", and "SLCSE$DATABASE" directories for the storage of files essential to the operation of the SLCSE Database Subsystem (i.e., Relational Database Management System (RDBMS) and "ER Interface (ERIF)" file objects). Also below the SLCSE$DBDIR directory are one or more "SLCSE$TEXT" directories (up to 10) for the storage and management of Project Database entity and relationship type instance text attribute data type file objects. A text attribute of an entity or a relationship type instance point to unique text file objects stored within a directory hierarchy below the SLCSE$TEXT directory. The details of the Main Hierarchy of the Database Files Hierarchy of the Project Files Hierarchy are described in [15].

The CM Hierarchy consists of only one directory, "SLCSE$CM", associated with an environment instantiation, and is for the storage of PDL

and source code files associated with formal design components of configurations known to the Project Database.

The SDF Hierarchy consists of one directory, "SLCSE$SDF", with any number of subordinate directories (depending on how many SDFs are associated with whatever number of formal design components are known to the Project Database).

The Maintenance Files Hierarchy is used for the storage of files that are essential for SLCSE and SLCSE tools operation (e.g., executable files) and maintenance (e.g., source code). It begins with a "SLCSE" directory that has a "BASE_ROOT" directory below it. Below the "BASE_ROOT" directory is a "DB" directory (for files primarily related to the Project Database), a "LIB" directory (for library file objects such as sharable executables), a "TOOL" directory (for SLCSE tool file objects such as source code and executables), and a "UI" directory (for files primarily related to the SLCSE User Interface, SLCSE Command Executive, SLCSE Environment Manager, and Rule Base). Each of these directories, in turn, have one or more subdirectory structures that will not be described here. The Maintenance Files Hierarchy is described in [14].

3.3.1.16.1.1 Example

An example of a SLCSE Project Database Model Schema, defined at the conceptual level using SDL, is:

```
subschema Contract is
-- User-defined enumeration type attribute
    type Currency_Unit is (Dollars, KDollars, MDollars, BDollars);
-- entity type
    entity type Contract is
-- Predefined attributes that every entity type has:
  -- access_name : string (32);
  -- descr_name : string (80);
  -- version : integer;
  -- key : integer;
  -- created : time;
  -- creator : user;
  -- modified : time;
  -- modifier : user;
  -- locked : time;
  -- owner : user;
-- User-defined attributes:
    Number : string(20);
```

```
    Effective_Date : time;
    Total_Amount_Unit : Currency_Unit := Dollars;
end entity;

-- Relationship type (Domain Relation Range)
    relationship type Contract Defines Project
-- Every relationship type has a cardinality associated with
-- it, which may be one of the following:
  -- One To One
  -- One To Many
  -- Many To One
  -- Many To Many
        cardinality One To Many;

entity type NDS is
    Description : text;
    NDS_Type : (COTS_Software,Reusable_Software,GFS);
end entity;

-- Alias entity type
    entity type Non_Developmental_Software is alias of NDS;

-- Relationship type with user-defined attribute
   relationship type Contract Identifies NDS
        cardinality Many To Many is
-- Predefined attributes that every relationship type has:
  -- domain_key : integer;
  -- range_key : integer;
  -- created : time;
  -- creator : user;
  -- modifier : user;
  -- locked : time;
  -- owner : user;
-- User-defined attribute:
        Usage_Rationale : text;
    end relationship;

    entity type SOW is
      Number : string(20);
      Date : time;
      Description : text;
    end entity;
```

```
    entity type CDRL_Item is
      Number : string(20);
    end entity;

-- Union entity type
    entity type Attachment is union of
      SOW,
      CDRL_Item;
    end entity;
end subschema;

subschema Project_Management is
    entity type Project is
        Resource_Acquisition_Plan : text;
        Security_Implementation_Plan : text;
        Subcontractor_Mgmt_Plan : text;
        Status_Accounting_System : text;
        Configuration_Audit_Plan : text;
    end entity;
end subschema;

-- representation

-- The representations are used to uniquely define the first 12
-- characters of each named item in the schema.  Furthermore,
-- representations are used to rename items which are Smartstar
-- or RDB keywords, such as report, audit, and format.

    representation is
        use Attmnt for Attachment;
    end representation;
```

An example of a SLCSE Infrastructure Database Model Schema, defined at the conceptual level using Ada, is:

```
-- This record is created by the SLCSE Environment Manager
-- for each person available for software development projects
-- for the Environment framework; a subset of this list is
-- maintained for each project environment instantiation.

type personnel_record is
    record
        user_name:  user_name_type;
```

```
        name : name_type;
        project : name_type;
        primary_or_current_role : role_type;
-- list of roles user is entitled to play.
        assigned_roles : dsa_types.dsa_list_type;
-- for access control.
        entity_list : dsa_types.dsa_list_type;
        read_granted : dsa_types.dsa_list_type;
        insert_granted : dsa_types.dsa_list_type;
        delete_granted : dsa_types.dsa_list_type;
        update_granted : dsa_types.dsa_list_type;
        process_state : state_type;
        uaf: uaf_record;
        roles_modified : boolean;
end record;

person: personnel_record;

type person_access_type is access personnel_record;

person_pointer: person_access_type;
```

An example of a SLCSE Project Files Hierarchy Model Schema is illustrated below:

User Files Hierarchy

```
SYS$SCRATCH
            SLCSE$PRJ
                        SLCSE$ROL
                                                SLCSE$WRK
                                                SLCSE$BCH
                                                SLCSE$FOLDER
                                                      SLCSE$FOLDER

[USER]-->   [PROJECT-X]-->   [PROGRAMMER__]-->  [WORK]
                        +                   +-->  [BATCH]
                        +                   +-->  [FOLDER1]
                        +                   +-->  [FOLDER2]
                        +                         +-->  [FOLDER2-1]
                        +-->  [SOFTWARE_ANA]-->  [WORK]
                                                +-->  [BATCH]
                                                +-->  [FOLDER1]
```

Database Files Hierarchy

Main Hierarchy

```
SLCSE
        SLCSE$DBDIR
                SLCSE$DATABASE
                        SLCSE$SDF.
                        SLCSE$SQL
                        SLCSE$TEXT

[SLCSE]-->  [PROJECT-X-DB]-->  [DATABASE]
                               +-->  [SDF]
                               +-->  [SQL]
                               +-->  [TEXT0]-->  [0]-->  [00]
                                     +           +-->  [01]
                                     +           .
                                     +           .
                                     +           .
                                     +           +-->  [99]
                                     +
                                     +-->        [1]-->  [00]
                                     +           +->   [01]
                                     .           .
                                     .           .
                                     .           .
                                     +           +-->  [99]
                                     +
                                     +-->        [9]-->  [00]
                                                 +->   [01]
                                                 .
                                                 .
                                                 .
                                                 +->   [99]
```

CM Hierarchy

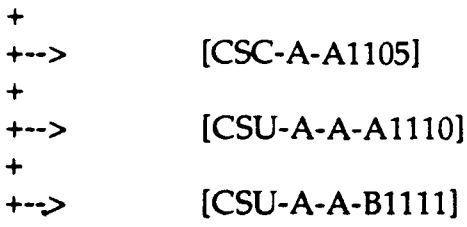
SLCSE$CM

[PROJECT-X-CM]

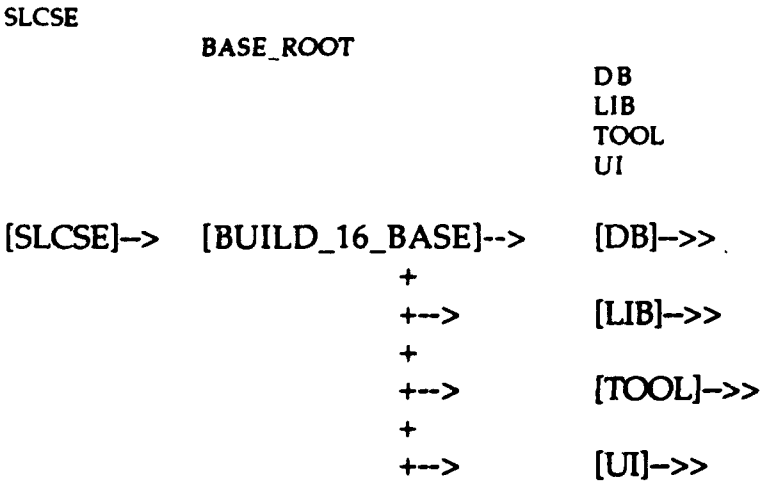
SDF Hierarchy

SLCSE$SDF

[PROJECT-X-SDF]--> [CSCI-A1101]



Maintenance Files Hierarchy



3.3.1.16.2 ROD - Operations

PROJECT DATABASE MODEL SCHEMA: Compound operations (of the basic Create, Update, Query, and Delete operations) such as "edit" (by a text editor), "compile" (by the SDL Compiler), "convert" (by the Convert_SDL tool), and "analyze" (by the analyzER tool) do apply to the Project Database Schema in support of tool integration. The "edit" operation involves creating and updating object definitions in support of the integration of tools that produce and utilize the instances of those objects.

INFRASTRUCTURE DATABASE MODEL SCHEMA: SEM provides operations on the Infrastructure Database in support of tool integration, as described in the Tool

3.3.1.16.2.1 Example

The Ada Test and Verification System (ATVS) is one example of a tool (refer to [4]) for which the SLCSE data model was augmented (refer to Appendix 1 of [13]) to include entity and relationship type definitions to support the integration of that tool with SLCSE.

3.3.1.16.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_16_global_schema FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_16_global_schema (FORWARD) [object_management]
1- 7_16_global_schema depends_on 15_1_2_common_data_descr
1- 7_16_global_schema depends_on 7_1_data_model

This service is required by the following services which are provided by SLCSE:

TRACE ON ENTITY 7_16_global_schema BACKWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_16_global_schema (BACKWARD) [object_management]
1- 7_14_constraint_mgt depends_on 7_16_global_schema
1- 7_20_metadata depends_on 7_16_global_schema
1- 7_2_data_storage depends_on 7_16_global_schema
<--2- 14_1_tool_registration depends_on 7_2_data_storage
<--3- 10_1_task_definition depends_on 14_1_tool_registration
<--4- 10_2_task_execution depends_on 10_1_task_definition
<--5- 10_4_task_history depends_on 10_2_task_execution
<--6- 10_2_task_execution@ depends_on 10_4_task_history
<--5- 10_5_event_monitoring depends_on 10_2_task_execution
<--5- 10_6_audit_accounting depends_on 10_2_task_execution
<--3- 10_7_role_mgt depends_on 14_1_tool_registration
<--3- 10_8_tool_registration depends_on 14_1_tool_registration

<--3- 15_1_4_tool_to_om depends_on 14_1_tool_registration
<--4- 15_1_5_om_to_om depends_on 15_1_4_tool_to_om
<--5- 15_1_1_data_integration depends_on 15_1_5_om_to_om
<--2- 7_11_derivation depends_on 7_2_data_storage
<--2- 7_12_replication depends_on 7_2_data_storage
<--3- 15_1_6_consistency_mgt depends_on 7_12_replication
<--2- 7_13_access_control depends_on 7_2_data_storage
<--3- 13_2_security_control depends_on 7_13_access_control
<--2- 7_17_version depends_on 7_2_data_storage
<--3- 7_18_configuration depends_on 7_17_version
<--2- 7_19_query depends_on 7_2_data_storage
<--2- 7_21_state_monitoring depends_on 7_2_data_storage
<--3- 10_5_event_monitoring depends_on 7_21_state_monitoring
<--2- 7_23_data_interchange depends_on 7_2_data_storage
<--3- 7_10_backup depends_on 7_23_data_interchange
<--4- 7_9_archive depends_on 7_10_backup
<--2- 7_3_relationship depends_on 7_2_data_storage
<--2- 7_4_name depends_on 7_2_data_storage
<--2- 7_6_data_transaction depends_on 7_2_data_storage
<--2- 7_7_concurrency depends_on 7_2_data_storage
<--3- 15_1_6_consistency_mgt depends_on 7_7_concurrency
<--2- 9_3_1_data_integration depends_on 7_2_data_storage
<--2- 9_3_2_control_integration depends_on 7_2_data_storage
<--2- 9_3_3_ui_integration depends_on 7_2_data_storage

3.3.1.17 Version Service (7.17)

3.3.1.17.1 ICE - Conceptual

PROJECT DATABASE: The Version Service of SLCSE is provided as an intrinsic part of the Project Database Model and it's run-time implementation within an environment instantiation. It is possible to have variants of entities and relationships stored within the same Project Database. Usually, one set of variants is associated with one "configuration", while another set of variants is associated with another configuration. A particular entity may have any number of versions, and the relationships between different versions of entities form the "glue" of a configuration that can be "baselined".

PROJECT FILES HIERARCHY: The Operating System underlying SLCSE provides the capability to have multiple versions of a file object, and SLCSE provides the necessary operations to associate particular versions of

file objects (e.g., PDL and source code files) with a particular configuration defined in the Project Database.

3.3.1.17.1.1 Example

The entity type SYSTEM may have an instance called, "B2_SC:B-2_SUPPORT_COMPUTER:1", which is the first version of an entity identifying the actual system being developed within an environment instantiation. At some point, the system configuration, consisting of the "B2_SC:B-2_SUPPORT_COMPUTER:1" instance and many other related entities in the Project Database, would be baselined (e.g., a Functional baseline is typically done when the requirements of the system are known). The developmental configuration of the system could then be identified by a SYSTEM entity type instance called, "B2_SC:B-2_SUPPORT_COMPUTER:2", the second version. This configuration could then be baselined, and so on and so forth.

3.3.1.17.2 ROD - Operations

PROJECT DATABASE: In SLCSE, the Version Service operations are provided by the ERIF, and DCL Interface Utilities. Both the ERIF and DCL Interface Utilities provide: "insert" operations for the creation of first-version entity instances, "duplicate" operations for the creation of variants of an entity type instance, "update" operations, "reserve" and "release" operations, and "delete" operations. The DCL Interface also provides a "purge" operation for the deletion of a certain number of old versions of entities, and the ERIF provides a "table" operation for querying about entities and relationships. For more information, refer to [13].

PROJECT FILES HIERARCHY: The operations to associate versions of files with a configuration stored and maintained in the Project Database are provided by the SLCSE tools BaselinER, Addfile, Putfile, Getfile, SDF_Create, SDF_Delete, Import, and Export, which utilize both ERIF and Operating System operations. These tools are described in the comments section for the Reference Model service called "Data Integration" (9.3.1), and in [15].

3.3.1.17.2.1 Example

The following is an example of the DCL Database Interface Utility operation for the creation of new versions of one or more existing entities listed in a special format text file named ENTITIES.LIS:

```
$DATABASE DUPLICATE /IN=ENTITIES.LIS
```

3.3.1.17.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_17_version FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_17_version (FORWARD) [object_management]
 1- 7_17_version depends_on 7_2_data_storage
 -->2- 7_2_data_storage depends_on 11_1_message_delivery
 -->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->2- 7_2_data_storage depends_on 7_16_global_schema
 -->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
 -->3- 7_16_global_schema depends_on 7_1_data_model
```

This service is required by the following service that is provided by SLCSE:

```
TRACE ON ENTITY 7_17_version BACKWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_17_version (BACKWARD) [object_management]
 1- 7_18_configuration depends_on 7_17_version
```

3.3.1.18 Configuration Service (7.18)

3.3.1.18.1 ICE - Conceptual

PROJECT DATABASE: The Configuration Service of SLCSE is provided by the Project Database associated with an environment instantiation, and the tools integrated with that environment. In SLCSE, a "configuration" consists of an identifiable set of entities and relationships associated with a system being developed within an environment instantiation. As the system evolves, it may be "baselined" several times, forming a number of configurations (e.g., DOD-STD-2167A identifies Functional, Allocated, Product, Developmental, and Product Version baselining). These configurations can include multiple combinations of the entities and relationships stored in an environment's Project Database (e.g., requirements, design, source code, documentation, etc.).

PROJECT FILES HIERARCHY: SLCSE tools provide the necessary operations to associate groups of file objects (e.g., PDL and source code files) with a particular configuration defined in the Project Database.

3.3.1.18.1.1 Example

A group of Project Database entities and relationships, and Project Files Hierarchy file objects associated with an instance of the entity type SYSTEM can be included in a configuration that is identified by an instance of the CONFIGURATION_ID entity type that is linked to those objects by a number of CONFIGURATION_ID INCLUDES PRODUCT relationships.

3.3.1.18.2 ROD - Operations

In SLCSE, the Configuration Service operations are provided by the ERIF at the application interface level. Compound operations called "include", "baseline", "generate report", and "edit" are provided by the SLCSE BaselinER tool:

"include" - entity query and selection for incorporation into a configuration.
"baseline" - freeze a configuration.
"generate report" - query the objects forming a configuration.
"edit" - edit an instance of the CONFIGURATION_ID entity type.

BaselinER is founded on the DOD-STD-2167A development methodology and the Configuration Management Subschema of the DOD-STD-2167A ER Data Model developed for SLCSE. Refer to [15].

3.3.1.18.2.1 Example

The following is an example of the Configuration Service of SLCSE using the BaselinER tool:

1. Query on instances of the DELIVERABLE_DOCUMENT entity type, and include the appropriate ones into a configuration identified by a CONFIGURATION_ID entity type instance.

2. Generate a report listing the configuration components.

3. If satisfied with the entities included in the configuration, then baseline the configuration.

3.3.1.18.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_18_configuration FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_18_configuration (FORWARD) [object_management]
1- 7_18_configuration depends_on 7_17_version
-->2- 7_17_version depends_on 7_2_data_storage
-->3- 7_2_data_storage depends_on 11_1_message_delivery
-->3- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->3- 7_2_data_storage depends_on 7_16_global_schema
-->4- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->4- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.1.19 Query Service (7.19)

3.3.1.19.1 ICE - Conceptual

PROJECT DATABASE OBJECTS: The Query Service of SLCSE for Project Database objects is provided at the user level (by EditER), at the application level (by the ERIF), and at the relational level (by the RDBMS). A query on entities and relationships may be specified in the form of a "qualified search", where partial entity/relationship identification specifications are provided, and the values returned are of those of the set of entities/relationships that match the partial specification. In addition, the SLCSE data model provides the capability to define an entity type (and relationship type) "union", which permits queries on the set of instances that are of the different entity types (and relationship types) included in the union definition.

INFRASTRUCTURE DATABASE OBJECTS: The Query Service of SLCSE for Infrastructure Database objects is provided at the user level (by the SLCSE Command Executive and "infra-coupled" SLCSE tools) and at the application level (by DSA routines). A query on local files, available subschemas, available tools, assigned roles, etc., results in the retrieval and presentation of lists of the objects stored in an Infrastructure Database.

PROJECT FILES HIERARCHY OBJECTS: The Query Service of SLCSE for file objects in the Project Files Hierarchy are provided at the user and application levels by the facilities provided by the underlying Operating System. In addition, for text attribute files in the Database Files Hierarchy (part of the Project Files Hierarchy), the Query Service of SLCSE for the Project Database also applies.

3.3.1.19.1.1 Example

PROJECT DATABASE OBJECTS:

A SLCSE user (within the EditER Entity-Relationship (ER) on-line query form) presses the proper sequence of keys for a qualified search, and specifies that all of the entities of the CSCI entity type that have a Descriptive Name beginning with the letter "S" be retrieved from the Project Database. EditER calls on ERIF routines to direct the necessary RDBMS operations to retrieve the requested information. The information is provided at the relational level by the RDBMS to the ERIF, to EditER by ERIF at the ER/application level, and to the user by EditER through the presentation of information in query form displays.

INFRASTRUCTURE DATABASE:

A SLCSE user presses the return key with the cursor on the TOOLS button presented by the User Interface. The Command Executive translates the request for tool information, and displays the TOOLS menu (loaded by the Command Executive with information retrieved from the Infrastructure Database using DSA routines at the beginning of the SLCSE user session). The TOOLS menu shows the tools available for the user's current role.

PROJECT FILES HIERARCHY:

Using the DIRECTORY tool of the VMS Operating System within the SLCSE environment, the user queries about all of the local files with a file extension of "ADA", and the DIRECTORY tool, in turn, calls on low-level Operating System routines to obtain the requested information. All of the requested information about the files matching the partial file name specification is then displayed to the user.

3.3.1.19.2 ROD - Operations

PROJECT DATABASE OBJECTS: The operations for the Query Service of SLCSE on the Project Database are provided by EditER at the user level. EditER operations are described below:

BACKWARD: Goes back to the previous entity or relationship in the buffer list. If the current record is the first in the buffer list then a message to this effect is displayed.

CLEAR ALL: Deletes all buffers from the list. Any changes to entities or relationships contained in the buffers are lost. They will not be made in the database.

CLEAR FIELD: Sets the current field to blanks. This has the following effects: Causes string, enumeration, boolean, user, file, and time attributes to become all blanks. Causes entity attribute key values to become zero. Causes integer and float attributes to become zero. Causes text attributes to contain no characters.

CLEAR FORM: Sets all modifiable fields of a form to blanks, with the exception of ACCESS_NAME, DESCR_NAME, KEY, DOMAIN_KEY, and RANGE_KEY fields. The effect on the fields which are changed is given in the description of CLEAR FIELD.

COPY ENTITY: Makes a copy of the field values of the current Entity, and inserts it into the buffer list. Text attributes contents are copied copied to the new Entity. This operation should be used to create a new version of an existing entity. It can not be used to copy relationships (relationships do not have versions).

DELETE: For entities or relationships which were retrieved from the database, this causes the current entity or relationship to be marked as deleted. When changes are saved, the entity or relationship will be removed from the database. The UNDELETE key can be used (before changes are saved) to undelete an entity or relationship which was marked as deleted. For entities and relationships which were inserted during the current session rather than being retrieved, this removes the entity or relationship from the buffer list immediately, since there is no need to wait until changes are saved. There is no way to undelete such an entity or relationship.

DONE: Exits the current form.

EDIT ENTITY ATTRIBUTE: If the cursor is on an entity attribute field, then this operation brings up a window with a list of the entity instances which the field may reference. By using the up and down arrow keys, the desired entity instance can be selected. If <RETURN> is pressed then the entity attribute will reference the selected entity instance. If the PREVIOUS key is pressed, then the entity attribute value is not changed. If the cursor is not on an entity attribute field, then an error message is displayed.

EDIT TEXT ATTRIBUTE: If the cursor is on a text attribute field, then this operation brings up the TPU editor with the file which represents the text attribute. The file may be edited as desired. If TPU is exited using the file will not be saved. If TPU is exited normally, then changes will be saved. If the cursor is not on a text attribute field, then an error message is displayed.

FIRST: Displays the first entity or relationship in the buffer list.

FORWARD: Goes to the next entity or relationship in the buffer list. If there are no more entities or relationships in the buffer then the next is read from the database. If the there are no more in the database then a message to this effect is displayed.

HELP: Displays a description of each of the EditER operations in a scrollable window.

INSERT: For entity forms, this inserts a blank entity form into the buffer list. This operation should be used to create new entities. The COPY ENTITY key should be used to create new versions of existing entities. For relationship forms, this causes a new relationship to be created. The user is prompted to specify the domain and range entities at the time the new relationship is inserted.

KEYPAD: Displays a picture of the EditER keypad layout.

LAST: Displays the last entity or relationship in the buffer list. This operation will not cause additional entities or relationships to be read from the database, if there are any. The FORWARD key must be used for that purpose. This key merely takes one to the last entity or relationship in the buffer list.

QBE: This operation performs qualified searches using the query by example method.

QUALIFY: Used to specify a restriction on the values of the current field which will be retrieved using QBE.

REPAINT: Redraws the entire screen.

RETRIEVE: Discards the current buffer list, and performs an unqualified search of the database for the entities or relationships of the same type as the form. If any are found, then the first is retrieved into the buffer. If none are found then a message to this effect is displayed.

RETURN: Goes to the next field of the form.

SAVE: Traverses the buffer list and updates the database entry for any entities or relationships which were modified, inserted, or deleted. The save process involves repeating the original query, if there was one, and synchronizing the buffer list to the records which are retrieved. This is necessary since the database only allows the current record to be modified or deleted. From a user perspective this involves automatically stepping through each buffer in the list, and pausing to make the update on those which have been changed.

SCROLL DOWN: Causes the form to scroll down to the next screen, if there is one.

SCROLL UP: Causes the form to scroll up to the previous screen, if there is one.

SHOW ERROR: Brings up a window containing the last error message.

SHOW FIELD: Brings up a window which displays information about the field where the cursor is located. The following information about the field is listed: The name of the database attribute, the field datatype, the field length in characters, whether the field is modifiable by the user, whether the field is WINNIE protected, whether the database is indexed on the field, whether the database must be unique on the field, whether null values are permitted for the field, the field's default value, the legal range of values for the field, the enumeration choices (if the field is an enumeration datatype), and the list of entity type names whose instances the field may reference (if the field is an entity datatype).

TOGGLE: If the cursor is on a boolean or enumeration field then this operation causes the field to change to its next legal value. All blanks represents a null and legal value. Repeated toggling will cycle through the various legal choices for the field. If the cursor in not on a boolean or enumeration field, then an error message is displayed.

UNDELETE: Causes an entity or relationship which was retrieved from the database, and then marked as deleted, to be marked as unmodified. This will prevent the entity or relationship from being deleted when changes are saved. This key can not be used to bring back an entity or relationship which was inserted and then deleted.

UNMODIFY: Causes an entity or relationship which was retrieved from the database, and then edited so that its status became modified, to be marked as unmodified. This will prevent the changes made to the entity or relationship from being made against the database when changes are saved. This key does not bring back the old values of the entity or relationship.

EditER uses the operations provided by the ERIF (see Data Storage Service/Persistence (7.2) ROD - Operations factor service description). The ERIF, in turn, uses the query services provided by the underlying RDBMS of the Project Database.

INFRASTRUCTURE DATABASE OBJECTS: The operations for the Query Service of SLCSE on the Infrastructure Database are provided by the Command Executive (CE) and "infra-coupled" SLCSE tools at the user level. The operations of the CE correspond with "command buttons" of the SLCSE User Interface, and are:

OBJECTS: When the Objects command button is selected, a pull-down menu appears with a display of all the objects available to the user for their current role. Objects consist of local files and "folders" (implemented as directories) and certain information about them is stored in the Infrastructure Database.

TOOLS: When the TOOLS command button is selected, a pull-down menu appears with a display of all the tools available to the user for their current role. Information about tools is stored in the Infrastructure Database (e.g., setting execution command options, input files and output files specifications, etc.), and is presented to the user through tool "set-up" windows that are loaded by the CE with the information retrieved from the Infrastructure Database.

SETTINGS: When the SETTINGS command button is selected, a pull-down menu appears that displays all of the settings tailored by the user (or established by default) for SLCSE operation (e.g., current role, scroll bar, prompt display, message search interval, automatic purge, etc.). This information is stored in the Infrastructure Database and retrieved from it by the CE.

The Command Executive and infra-coupled tools use DSA operations at the application level to query about Infrastructure Database information (see Data Storage Service/Persistence (7.2) ROD - Operations factor service description). An example of an infra-coupled tool is ModifyER, which presents a menu of the Project Database subschema names to which the user is permitted access for his/her current role. ModifyER uses DSA operations to query on Infrastructure Database subschema information.

PROJECT FILES HIERARCHY OBJECTS: Operations for the Query Service of SLCSE on Project Files Hierarchy objects are provided by utilities of the VMS Operating System at the user level, and are primarily the following:

ANALYZE/<Qualifier_List> <File_Specification>
DIFFERENCES/<Qualifier_List> <File_Specification>
SHOW ACL/<Qualifier_List> <File_Specification>
TYPE/<Qualifier_List> <File_Specification>

The ability to query about a group of files with common attributes is provided by both the qualifier list (e.g., about files created after a certain date) and the file specification, where "wild card" characters can be used to query with partial specification matching (e.g., DIRECTORY *.ADA). These VMS Operating System utilities are described in detail in [27]. These utilities and

other applications are provided with low-level Operating System operations that are to numerous to list here.

The OS "Run-Time Library" provides many VAX/VMS system level routines. Examples are "LIB$RENAME_FILE", and "LIB$DELETE_FILE". Pre-defined Ada interfaces for many system level routines are provided in the "STARLET", "SYSTEM", and "CALENDAR" packages. Ada interfaces for some commonly used Run-Time Library routines which were not included in the pre-defined packages have been written and included in a package called "MORE_STARLET".

The Run-Time Library routines are documented in a series of reference manuals, and descriptions of the individual Run-Time facilities, along with reference sections describing the individual routines in detail, and can be found in [20], [21], [22], [23], [24], [25], and [26].

3.3.1.19.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_19_query FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_19_query (FORWARD) [object_management]
1- 7_19_query depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.1.20 Metadata Service (7.20)

3.3.1.20.1 ICE - Conceptual

PROJECT DATABASE METADATA: The data (i.e., entity/relationship/attribute types) about the data (i.e., entity/relationship/attribute type instances) stored in a SLCSE Project Database is formally specified in the Schema Definition Language (SDL).

The schema of SLCSE also includes a "metadata" subschema, which is a metadata definition that is stored on-line in a Project Database created from that schema.

INFRASTRUCTURE DATABASE METADATA: The data (i.e., list, record, and component types) about the data (i.e., list, record, and component instances) stored in a SLCSE Infrastructure Database is formally specified in Ada.

3.3.1.20.1.1 Example

PROJECT DATABASE: The entity type CSCI in the DOD-STD-2167A schema is defined to have a number of attributes of specific data types, and defines the attributes that an instance of that type will have.

INFRASTRUCTURE DATABASE: In the metadata description of a SLCSE Infrastructure Database, the list TOOLS_LIST is defined to have TOOLS_RECORD records with components of specific data types, and defines the components that an instance of a record, of an instance on that list, will have.

3.3.1.20.2 ROD - Operations

In SLCSE, the operations involved with Project Database and Infrastructure Database metadata are essentially the same as those involved with the instances (i.e., create, query, update, and delete), with the exception that create, update, and delete operations are only possible before an environment is instantiated from the SLCSE framework. It should be noted that the query operations involved with Infrastructure Database metadata after an environment is instantiated are extremely limited compared to those for Project Database metadata.

3.3.1.20.2.1 Example

PROJECT DATABASE:

Examples of query operations on metadata for Project Database metadata are the ERIF routines "Relationship_Type_Of", "Domain_Type_Of", "Range_Type_Of", "Relationship_Types_Of", "Attribute_Types_Of", "Cardinality_Of", and "Datatype_Of".

INFRASTRUCTURE DATABASE:

An example of a query operation on Infrastructure Database metadata is the DSA routine "Display_DSA_Database".

3.3.1.20.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_20_metadata FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_20_metadata (FORWARD) [object_management]
1- 7_20_metadata depends_on 7_16_global_schema
-->2- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->2- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.1.21 State Monitoring Service/Triggering (7.21)

3.3.1.21.1 ICE - Conceptual

SLCSE provides a "Database Monitor", which periodically examines the Project Database for events that have occurred. The entities and relationships, and the attributes of them, that are to be monitored (and what they are to be monitored for) are specified using the Document Generation Language (DGL), along with a schedule for the frequency of monitoring using the "Database Monitor Manager". The "Database Monitor" initiates the monitoring process according to the schedule specified using the Database Monitor Manager. The actions that can occur as a result of database events take two different forms: (1) event notification, and (2) event report generation and distribution. Typically, both of these actions occur as the result of a single event. Event notification results in an electronic mail announcement to project personnel on a notification list of an event's occurance. Event report generation and distribution results in the generation of a detailed report about the event, and the distribution of the report to project personnel on a distribution list. The notification and distribution lists are also established using the Database Monitor Manager.

3.3.1.21.1.1 Example

If the COMPLETED boolean type attribute of an instance of the entity type MILESTONE has the value of FALSE, and the current date is within a week of the scheduled date for the milestone's completion, then a notification to the all project personnel in the form of a warning could be initiated, and a report describing the milestone could be distributed to the project manager.

3.3.1.21.2 ROD - Operations

The basic set of operations (create, query, update, and delete) apply to this service, and in SLCSE are:

Create:
"Add Monitor Action"
"Add 'Meta' Message"
"Add 'Meta' Report"
"Generate Message"
"Generate Report"

Query:
"Read Monitor Action"
"Read 'Meta' Message"
"Read 'Meta' Report"

"Read Monitor Schedule"
"Read Message"
"Read Report"

Update:
"Modify Monitor Action"
"Modify 'Meta' Message"
"Modify 'Meta' Report"
"Modify Monitor Schedule"

Delete:
"Delete Monitor Action"
"Delete Monitor Schedule"

By the term 'Meta', what is meant is data about the message to be sent or the report to be generated as the result of a database event. A 'Meta' Message and a 'Meta' Report are part of a Monitor Action, and are stored in an entity of the Project Database.

Event Monitor Actions are created and updated using the Database Monitor Manager. Event notification and report distribution are performed via a combination of the Database Monitor, the VAX/VMS Batch Queue, and Electronic Mail. Report Generation is performed by the SLCSE tool DOCGEN which extracts the pertinent information concerning an event from the Project Database.

3.3.1.21.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 7_21_state_monitoring FORWARD
9 LEVEL RELATIONSHIP

-- AS A MEMBER OF [all] SUBSETS.

7_21_state_monitoring (FORWARD) [object_management]
1- 7_21_state_monitoring depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is required by the following service that is provided by SLCSE:

TRACE ON ENTITY 7_21_state_monitoring BACKWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

7_21_state_monitoring (BACKWARD) [object_management]
1- 10_5_event_monitoring depends_on 7_21_state_monitoring

3.3.1.22 Sub-Environment (Views) Service (7.22)

Unless the environment of a single user (with roles, tools, and data access constraints specific to the user) on a project (consisting of multiple users) can be considered as a sub-environment of an environment consisting of the entire set of user environments for a project, then this service is not provided in SLCSE.

3.3.1.23 Data Interchange Service (7.23)

3.3.1.23.1 ICE - Conceptual

For SLCSE Project Database information, the translation of that data to some a text file format, and from that format to a Project Database, is provided in SLCSE by the Digital Command Language (DCL) Interface Utilities. The format of the data is readily transferrable to portable storage media (e.g., magnetic tape), and, given a format translation utility, could be used to transfer data to and from the repositories of non-SLCSE environments.

3.3.1.23.1.1 Example

An example of the DCL Interface Utilities text file format (produced by the DATABASE RETRIEVE Utility) is provided below:

1021 | ALICIA | AUTOMATED LIFE CYCLE IMPACT ANALYSIS SYSTEM
1020 | AMS | AUTOMATED MEASUREMENT SYSTEM
1022 | ANALYZER | ANALYZER VERSION 4.0
1019 | ATVS | ADA TEST AND VERIFICATION SYSTEM
1018 | SLCSE | SOFTWARE LIFE CYCLE SUPPORT ENVIRONMENT
1103 | SPMS | SLCSE PROJECT MANAGEMENT SYSTEM

3.3.1.23.2 ROD - Operations

The operations provided by the DCL Interface Utilities are:

```
DATABASE RETRIEVE <entity-type>
        [/ATTRIBUTES=". . . "]
        [/NAME]
        [/TABCHAR=( . . . )]
        [/TABS=( . . . )]
        [/OUT= . . . ]
        [/APPEND]
        [/QUERY=" . . . "]
        [/ORDER="<attr-name> [ASCENDING | DESCENDING][,... ]"]

DATABASE INSERT
        [/IN= . . . ]
        [/TABCHAR=" . . . "]
        [/TABS=( . . . )]
        [/OUT= . . . ]
        [/APPEND]

DATABASE UPDATE
        [/IN= . . . ]
        [/TABCHAR=" . . . "]
        [/TABS= ( . . . ) ]

DATABASE DELETE
        [/IN= . . . ]
        [/TABCHAR=" . . . " ]
        [/TABS= ( . . . ) ]

DATABASE DUPLICATE
        [/IN= . . . ]
        [/TABCHAR=" . . . " ]
        [/TABS= ( . . . ) ]
```

[/OUT=. . .]
[/APPEND]

DATABASE [RESERVE | RELEASE]
[/IN=. . .]
[/TABCHAR=". . . "]
[/TABS=". . . "]

DATABASE PURGE
[/SCHEMA=. . .]
[/ENTITY=. . .]
[/KEEP=n]
[/LOG=. . .]

ENTITY RETRIEVE <entity-type>
[/QUERY=". . . "]
[/NAME=". . . "]
[/ATTRIBUTE= (<attr-name>,<attr-name>)]
[/OUTVALUE=(<symbol>,<symbol>)]
[/OUTNAME= <symbol>]
[/OUTENTITY= <symbol>]

RELATIONSHIP RETRIEVE <relation-type>
[/QUERY=". . . "]
[/DOMAIN=". . . "]
[/RANGE=". . . "]
[/ATTRIBUTE(<attr-name>, <attr-name>)]
[OUTVALUE=(<symbol>,<symbol>)]
[OUTDOMAIN=<symbol>]
[OUTRANGE=<symbol>]
[OUTRELATIONSHIP=<symbol>]

ENTITY INSERT <entity-type>
[/NAME=". . . "]
[/ATTRIBUTE= (<attr-name>,<attr-name>)]
[/VALUE=("...","...")]
[/OUTNAME=<symbol>]

RELATIONSHIP INSERT <relation-type>
[/DOMAIN=". . . "]
[/RANGE=". . . "]
[/ATTRIBUTE= (<attr-name>,<attr-name>)]
[/VALUE=(<symbol>,<symbol>)]

```
        [/OUTDOMAIN=<symbol>]
        [/OUTRANGE=<symbol>]
        [/OUTRELATIONSHIP=<symbol>]
ENTITY UPDATE <entity-type>
        [/NAME=" . . . "]
        [/ATTRIBUTE=(<attr-name>,<attr-name>)]
        [/VALUE=(" . . . "," . . . ")]

RELATIONSHIP UPDATE <relation-type>
        [/DOMAIN=" . . . "]
        [/RANGE=" . . . "]
        [/ATTRIBUTE=(<attr-name>,<attr-name>)]
        [/VALUE=(<symbol>,<symbol>)]

ENTITY DELETE <entity-type>
        [/NAME=" . . . "]

RELATIONSHIP DELETE <relation-type>
        [/DOMAIN=" . . . "]
        [/RANGE=" . . . "]
ENTITY [RESERVE | RELEASE] <entity-type>
        [/NAME=" . . . "]

RELATIONSHIP [RESERVE | RELEASE] <relation-type>
        [/DOMAIN="..."]

DATABASE UNLOAD
        [/SCHEMA= . . . ]
        [/ENTITY= . . . ]
        [/RELATIONSHIP= . . . ]
        [/OUT= . . . ]
        [/LOG= . . . ]

DATABASE LOAD
        [/IN= . . . ]
        [/LOG= . . . ]
```

3.3.1.23.2.1 Example

An example of a DCL Database Utilities command (which produced the example given in the "ICE - Conceptual" Service Dimension Form for this service) is shown below:

```
$ database retrieve csci -
        /attributes="key,access_name,descr_name" -
        /out=csci.txt
```

3.3.1.23.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_23_data_interchange FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_23_data_interchange (FORWARD) [object_management]
 1- 7_23_data_interchange depends_on 7_2_data_storage
 -->2- 7_2_data_storage depends_on 11_1_message_delivery
 -->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->2- 7_2_data_storage depends_on 7_16_global_schema
 -->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
 -->3- 7_16_global_schema depends_on 7_1_data_model
```

This service is required by the following services which are provided by SLCSE:

```
TRACE ON ENTITY 7_23_data_interchange BACKWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

7_23_data_interchange (BACKWARD) [object_management]
 1- 7_10_backup depends_on 7_23_data_interchange
 <--2- 7_9_archive depends_on 7_10_backup
```

3.3.1.24 Tool Registration (7.24)

Although the data used/produced by tools is registered in the SLCSE Project Database (by defining the necessary entity types, etc.), the object manager is not aware of which tools are integrated in SLCSE. Tool registration, similarly, is not provided for the Infrastructure Database or the Project Files Hierarchy (i.e., DSA does not know which tools are recorded in the Infrastructure Database, and the VAX/VMS Operating System is not aware of the changes made to the Maintenance Files Hierarchy of the Project Files Hierarchy in support of tool integration).

3.3.2 Tools (9)

3.3.2.1 Tool Integration (9.3)

3.3.2.1.1 Data Integration (9.3.1)

3.3.2.1.1.1 ICE - Conceptual

There are fully-data-integrated, partially-data-integrated, and data-interoperable tools currently in SLCSE.

SLCSE tools are of several types in terms of their integration within an instantiation of the environment framework, and are categorized as described in the Service Mapping Form for this service. Those that apply to data integration are:

DIRECTLY-COUPLED
Tool uses the SLCSE Project Database directly through the ERIF/HLERIF.

INDIRECTLY-COUPLED
Tool uses the SLCSE Project Database via the DCL Interface Utilities or some similar interface that does not by-pass the ERIF/HLERIF.

LOOSELY-COUPLED
Tool does not use the SLCSE Project Database.

INFRA-COUPLED
Tool uses the SLCSE Infrastructure Database via DSA.

NON-INFRA-COUPLED
Tool does not use the SLCSE Infrastructure Database.

DEPENDENTLY-COUPLED

Tool is totally dependent on the SLCSE Project Database and/or the SLCSE Infrastructure Database to maintain its data, and has no persistent database of its own.

These categories describe the integration of tools with SLCSE, and do not describe the direct integration of one tool with another, since that is an unpredictable feature that is specific to the tools, and beyond the control of the SLCSE framework which houses them. For example, a tool, T1, which is directly-coupled, would not necessarily be fully-data-integrated with another directly-coupled tool, T2, if either tool, T1 or T2, is independently-coupled.

3.3.2.1.1.1 Example

Directly-coupled - DOCGEN
Loosely-coupled - ATVS
Infra-coupled - SPMS

3.3.2.1.1.2 ROD - Operations

Refer to the operations described for the services listed in the "Relationships Between Services" section for this service.

3.3.2.1.1.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 9_3_1_data_integration FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

9_3_1_data_integration (FORWARD) [tools]

1- 9_3_1_data_integration depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.2.1.2 Control Integration (9.3.2)

3.3.2.1.2.1 ICE - Conceptual

There are fully-control-integrated (e.g., ModifyER and EditER), partially-control-integrated (e.g., BaselinER and DOCGEN), and control-interoperable (e.g., BaselinER and its submitted batch jobs)tools currently in SLCSE.

SLCSE tools are of several types in terms of their integration within an instantiation of the environment framework, and are categorized as described below in relation to this service.

CONTROL-COUPLED
Tool is invokable by SLCSE as a sub-process that must return control to SLCSE upon the completion of its execution.

NON-CONTROL-COUPLED
Tool not invokable by SLCSE, but is integrated with the SLCSE Project Database and/or the SLCSE Infrastructure Database.

As with data integration, the focus in SLCSE is tool-to-environment integration, as opposed to direct tool-to-tool integration (i.e., tool-to-tool integration is supported via tool-to-environment-to-tool integration).

3.3.2.1.2.1.1 Example

Control-Coupled - Nearly all VAX tools integrated with SLCSE are control-coupled.

3.3.2.1.2.2 ROD - Operations

Refer to the operations described for the services listed in the "Relationships Between Services" section for this service.

3.3.2.1.2.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 9_3_2_control_integration FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

9_3_2_control_integration (FORWARD) [tools]
1- 9_3_2_control_integration depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.2.1.3 User Interface Integration (9.3.3)

3.3.2.1.3.1 ICE - Conceptual

All SLCSE tools are UI-interoperable except for those that do not run on the VAX/VMS Operating System (that is, for example, Macintosh-based SPMS tools, although similar in appearance, are not UI-interoperable with VAX/VMS-based SLCSE tools, according to the definition provided in the Reference Model). Currently, there are no partially-UI-integrated SLCSE tools (i.e., all SLCSE "conformant" tools are fully-UI-integrated, but this *is not* related to their level of control integration).

SLCSE tools are of several types in terms of their integration within an instantiation of the environment framework, and are categorized as follows in relation to this service.

CONFORMANT
Tool user interface style conforms to that of the SLCSE User Interface.

NON-CONFORMANT
Tool user interface style does not conform to that of the SLCSE User Interface.

As with data and control integration, the focus in SLCSE is tool-to-environment integration, as opposed to direct tool-to-tool integration (i.e., tool-to-tool integration is supported via tool-to-environment-to-tool integration).

3.3.2.1.3.1.1 Example

All conformant tools in SLCSE are totally-UI-integrated.

3.3.2.1.3.2 ROD - Operations

Refer to the operations described for the services listed in the "Relationships Between Services" section for this service.

3.3.2.1.3.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 9_3_3_ui_integration FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

9_3_3_ui_integration (FORWARD) [tools]
1- 9_3_3_ui_integration depends_on 7_2_data_storage
-->2- 7_2_data_storage depends_on 11_1_message_delivery
-->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->2- 7_2_data_storage depends_on 7_16_global_schema
-->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->3- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.3 Task Management Services (10)

3.3.3.1 Task Definition Service (10.1)

3.3.3.1.1 ICE - Conceptual

During environment definition and modification using the SLCSE Environment Manager (SEM), it is possible to define pre-invocation (rules that are checked prior to a tool's invocation) and post-execution rules (rules that are checked after a tool's execution) that are specific to a particular tool of the environment. Both pre-Invocation and post-execution rules can be defined for any tool according to a pre-defined syntax consisting of the following keywords:

/=
<
<=
=
>
>=
ABORT
AND
CURRENT_TIME
DAY
DAY_OF_WEEK
DISPLAY
ELSE
FALSE
FRIDAY
HOUR
IF
LAST_STATUS()

MAIL
MONDAY
NOT
NUMBER_OF_INVOCATIONS()
OR
SATURDAY
SUCCESS()
SUNDAY
THEN
THURSDAY
TIME_OF_LAST_INVOCATION()
TO
TRUE
TUESDAY
WEDNESDAY
WEEK

3.3.3.1.1.1 Example

Some examples of rules that can be defined are:

```
Rule Number: 1 Tool: DOCGEN_2167A Rule Type: PRE-INVOCATION

IF NUMBER_OF_INVOCATIONS(DOCGEN_2167A) > 2 THEN
        DISPLAY
                "YOU'RE GOING TO STRESS THE MICROVAX -
                DON'T RUN DOCGEN AGAIN DURING THIS SESSION"

Rule Number: 2 Tool: DOCGEN_2167A Rule Type: PRE-INVOCATION

IF NUMBER_OF_INVOCATIONS(DOCGEN_2167A) > 3 THEN
        DISPLAY
                "YOU WERE WARNED!"
        AND MAIL
                "I'VE IGNORED SLCSE'S WARNING ABOUT DOCGEN -
                TAKE APPROPRIATE MEASURES AGAINST ME."
        TO SYSTEM
        AND ABORT

Rule Number: 3 Tool: LOGIN Rule Type: PRE-INVOCATION

IF DAY = SATURDAY OR DAY = SUNDAY THEN
        MAIL "UNAUTHORIZED WEEKEND ACCESS TO SLCSE"
```

```
        TO SYSTEM
        AND ABORT

Rule Number: 4 Tool: ADA Rule Type: PRE-INVOCATION

IF SUCCESS(LAST_STATUS(ADA)) THEN
        DISPLAY
                "BE SURE TO RUN THE ATVS"

Rule Number: 5 Tool: EXIT Rule Type: POST-EXECUTION

IF NUMBER_OF_INVOCATIONS(ADA) > 0 AND
NUMBER_OF_INVOCATIONS(ATVS) = 0 THEN
        MAIL
                "I'VE BEEN COMPILING WITHOUT RUNNING ATVS -
                YOU HAD BETTER SEE WHAT I'M UP TO."
        TO BOSS
```

3.3.3.1.2 ROD - Operations

The operations involved with rule definition in SEM are create, query, update, and delete.

3.3.3.1.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 10_1_task_definition FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

10_1_task_definition (FORWARD) [task_management]
```

1- 10_1_task_definition depends_on 14_1_tool_registration
-->2- 14_1_tool_registration depends_on 7_2_data_storage
-->3- 7_2_data_storage depends_on 11_1_message_delivery
-->3- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->3- 7_2_data_storage depends_on 7_16_global_schema
-->4- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->4- 7_16_global_schema depends_on 7_1_data_model

This service is required by the following services which are provided by SLCSE:

TRACE ON ENTITY 10_1_task_definition BACKWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

10_1_task_definition (BACKWARD) [task_management]
1- 10_2_task_execution depends_on 10_1_task_definition
<--2- 10_4_task_history depends_on 10_2_task_execution
<--3- 10_2_task_execution@ depends_on 10_4_task_history
<--2- 10_5_event_monitoring depends_on 10_2_task_execution
<--2- 10_6_audit_accounting depends_on 10_2_task_execution

3.3.3.2 Task Execution Service (10.2)

As rule definition in SLCSE relates to the Task Definition Service (10.1), so does rule enforcement in SLCSE relate to this service, and is described as part of the Event Monitoring Service (10.5) of the Task Management Services.

3.3.3.3 Task History Service (10.4)

This service is implemented as part of the Task Management Audit and Accounting Service (10.4).

3.3.3.4 Event Monitoring Service (10.5)

3.3.3.4.1 ICE - Conceptual

As part of the SLCSE Command Executive (CE), there exists what is called the Rule Base. The Rule Base is that portion of the CE which ensures that before a tool is invoked, and after the tool completes its execution, if a rule exists, then the actions specified by the rule are carried out. In this case, the event is tool invocation or tool execution termination, and the actions

that can be taken (depending on the conditions of the rule) are display a message to the user, mail a message from the user to another user, or abort the execution of the tool. Rule information is stored in the SLCSE Infrastructure Database.

3.3.3.4.1.1 Example

If the following post-execution rule were defined for the EXIT tool:

```
IF NUMBER_OF_INVOCATIONS(ADA) > 0 AND
NUMBER_OF_INVOCATIONS(ATVS) = 0 THEN
    MAIL
        "I'VE BEEN COMPILING WITHOUT RUNNING ATVS -
        YOU HAD BETTER SEE WHAT I'M UP TO"
    TO BOSS
```

and the rule was found to be violated by the Rule Base for the user, then another user having the BOSS account would receive an electronic mail message from the user (sent by the Rule Base) saying, "I'VE BEEN COMPILING WITHOUT RUNNING ATVS - YOU HAD BETTER SEE WHAT I'M UP TO."

3.3.3.4.2 ROD - Operations

The operations performed by the CE Rule Base in monitoring for the execution of tools and the resultant triggering of actions are:

1. Check for rule before tool invocation.

2. (If pre-invocation rule exists) Parse the rule for interpretation of:
   a. Truth value of antecedent clause, and
   b. (If antecedent clause is true) Action of consequent clause.

3. (If antecedent clause is true) Perform the action prescribed by the rule:
   a. Display,
   b. Mail a message, or
   c. Abort the tool's execution.

4. (If tool executes) Check for rule after tool execution.

5. (If post-execution rule exists) Parse the rule for interpretation of:
   a. Truth value of antecedent clause, and
   b. (If antecedent clause is true) Action of consequent clause.

6. (If antecedent clause is true) Perform the action prescribed by the rule:
   a. Display, or
   b. Mail a message.

3.3.3.4.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 10_5_event_monitoring FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

10_5_event_monitoring (FORWARD) [task_management]
1- 10_5_event_monitoring depends_on 10_2_task_execution
-->2- 10_2_task_execution depends_on 10_1_task_definition
-->3- 10_1_task_definition depends_on 14_1_tool_registration
-->4- 14_1_tool_registration depends_on 7_2_data_storage@
-->2- 10_2_task_execution depends_on 10_4_task_history
-->3- 10_4_task_history depends_on 10_2_task_execution@
1- 10_5_event_monitoring depends_on 7_21_state_monitoring
-->2- 7_21_state_monitoring depends_on 7_2_data_storage
-->3- 7_2_data_storage depends_on 11_1_message_delivery
-->3- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->3- 7_2_data_storage depends_on 7_16_global_schema
-->4- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->4- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.3.5 Audit and Accounting Service (10.6)

3.3.3.5.1 ICE - Conceptual

In SLCSE, data is maintained for the enforcement of rules that are defined for tools. In the Infrastructure Database, a user has a PERSONNEL_RECORD on a list in his/her PROJECT_ROOT_RECORD. On the PERSONNEL_RECORD is an ASSIGNED_ROLES list of ROLE_RECORDs. Each ROLE_RECORD has a TOOLS_LIST of TOOL_RECORDs. Each TOOL_RECORD has the following information that is maintained by SLCSE as a record of what has been done with a tool within the development environment for the enforcement of rules that are defined in records on the TOOL_RECORD's PRE_RULE_LIST and POST_RULE_LIST:

```
TIME_LAST_INVOCATION : CALENDAR.TIME;
NUMBER_OF_INVOCATIONS : NATURAL;
LAST_STATUS_OF_TOOL :
        CONDITION_HANDLING.COND_VALUE_TYPE;
```

The VAX/VMS Operating System of SLCSE also provides an Audit and Accounting Service. VAX/VMS provides the ACCOUNTING utility (refer to [18]), which can be used to query about how the system has been used, how it has performed, and in some cases, how particular individuals have used the system. While the ACCOUNTING utility is not an actual part of the SLCSE, it is available to users of the system upon which SLCSE is implemented, and provides the following kinds of accounting data:

1. Batch job termination.
2. Detached process termination.
3. Image activation.
4. Interactive job termination.
5. Login failures.
6. User messages.
7. Network job termination.
8. Print jobs.
9. Process termination.
10. Subprocess termination.

3.3.3.5.1.1 Example

Every time a tool is executed by a SLCSE user, the following TOOL_RECORD fields are updated for that tool by the Rule Base of the Command Executive: TIME_LAST_INVOCATION, NUMBER_OF_INVOCATIONS, and LAST_STATUS_OF_TOOL. In

addition, the VAX/VMS operating system records various ACCOUNTING information about the activity of the user and system processes.

3.3.3.5.2 ROD - Operations

The operations of the Audit and Accounting Service of SLCSE in regards to pre-invocation and post-execution rule data recording are provided by DSA and used by the Command Executive Rule Base to create, update, and query the information prior to a tool's invocation and after its execution.

The operations of the Audit and Accounting Service of VAX/VMS in regards to the ACCOUNTING utility are also create, update, and query. Accounting information is created in an ACCOUNTING.DAT file by VAX/VMS whenever an event that is to be accounted for (estabished by the SET ACCOUNTING command) occurs, but information that was previously recorded is never updated as it is a permanent record of past events. However, it possible to update what information that controls what is to be captured for ACCOUNTING (using the SET ACCOUNTING command), and to query about this information (using the SHOW ACCOUNTING command). The query operations are provided by the ACCOUNTING command itself, and are characterized by its command qualifiers:

/ACCOUNT
/ADDRESS
/BEFORE
/BINARY
/ENTRY
/FULL
/IDENTIFICATION
/IMAGE
/JOB
/LOG
/NODE
/OUTPUT
/OWNER
/PRIORITY
/PROCESS
/QUEUE
/REJECTED
/REMOTE_ID
/REPORT
/SINCE
/SORT

/STATUS
/SUMMARY
/TERM
/TITLE
/TYPE
/UIC
/USER

3.3.3.5.2.1 Example

The following is actual code taken from the SLCSE Command Executive, and is an update operation provided by DSA on the Infrastructure Database tool record information with regard to rules (i.e., the number of invocations of the tool):

```
DSA_DATA_STRUCTURES.TOOL_POINTER :=
        DATASET_AT ( DSA_LOCKER ( 3 ) );
TOOL_POINTER.NUMBER_OF_INVOCATIONS :=
        TOOL_POINTER.NUMBER_OF_INVOCATIONS + 1;;
```

An example of a query operation and the data obtained using the ACCOUNTING utility of the VAX/VMS Operating system is:

```
$ ACCOUNTING/SINCE=TODAY/ACCOUNT=JONES

Date         Time     Type   Subtype    User    ID       Source   Status

1-MAY-1991 13:39:20 IMAGE ADA        JONES   22E00142 TWA30: 10000001
1-MAY-1991 13:40:11 IMAGE MODIFYER JONES   22E00142 TWA30: 00000001
```

3.3.3.5.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 10_6_audit_accounting FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

10_6_audit_accounting (FORWARD) [task_management]
1- 10_6_audit_accounting depends_on 10_2_task_execution
-->2- 10_2_task_execution depends_on 10_1_task_definition
-->3- 10_1_task_definition depends_on 14_1_tool_registration
-->4- 14_1_tool_registration depends_on 7_2_data_storage
-->5- 7_2_data_storage depends_on 11_1_message_delivery
-->5- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->5- 7_2_data_storage depends_on 7_16_global_schema
-->6- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->6- 7_16_global_schema depends_on 7_1_data_model
-->2- 10_2_task_execution depends_on 10_4_task_history
-->3- 10_4_task_history depends_on 10_2_task_execution@

This service is not required by any of the mapped services.

3.3.3.6 Role Management Service (10.7)

The Role Management Service of SLCSE is a part of the two SLCSE Framework Administration services called Framework Definition/Modification and Environment Definition/Modification, which are services not specifically found in the Reference Model.

3.3.3.7 Tool Registration (10.8)

In SLCSE, the Tool Registration Service is provided only in terms of rules, and is implemented as part of the Framework Administration and Configuration Service called the Environment Definition/Modification, which is a service that is not specifically described in the Reference Model.

3.3.4 Message Services (11)

3.3.4.1 Message Delivery Service (11.1)

3.3.4.1.1 ICE - Conceptual

SLCSE, in general, does not provide framework-to-framework two-way communication.

Tool-to-tool communication in SLCSE is specific to the design of the tools that are integrated with the SLCSE framework. Tool-to-tool communication in SLCSE is usually not two-way, however, but is accomplished via the services of the environment (e.g., tool-to-OM-to-tool).

Service-to-service two-way communication is often internal to the SLCSE Command Executive (since the Command Executive provides most of the services for the user environment). The exceptions (refer to Figure 4 of the General Questions Form) are the communication that occurs between: (1) the SLCSE Command Executive (CE) and the SLCSE Tooler, (2) the CE and the SLCSE Message Handler, (3) the Message Handler and the VMS Batch Queue, and (4) the CE and the various services of the Object Manager (i.e., DSA, ERIF, HLERIF, SMARTSTAR, VMS, RDBMS, Database Monitor, and the Database Client/Servers). In all cases, the Message Delivery Service is provided, at the lowest level, by the VAX/VMS Operating System whether it be mail-box message passing, process-to-process termination status passing, DECnet data packet passing, or message passing internal to an executing program.

Tool-to-service two-way communication is accomplished via mail-box message passing (e.g., between tools and the SLCSE Tooler), process-to-process termination status passing (e.g., between tools and the Database Monitor), and message passing internal to an executing program (e.g., between tools and the ERIF).

All message passing in SLCSE is point-to-point, with the exception of messages captured by SLCSE or SLCSE tools from the Operating System (i.e., one-way broadcast). For broadcast messages from the Operating System, a point-to-point message delivery from the SLCSE CE to the user (service-to-user), or from a tool to the user (tool-to-user), subsequently follows via the SLCSE, or tool, User Interface.

3.3.4.1.2 ROD - Operations

The operations that apply to this service in SLCSE are create message, send message, and read message.

3.3.4.1.2.1 Example

The SLCSE Command Executive builds a command and sends a command to the SLCSE Tooler. The SLCSE Tooler reads the command,

executes a tool in accordance with the message, and sends a tool status message back to the Command Executive when the tool finishes its execution.

3.3.4.1.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires none of the mapped services.

This service is required by the following services which are provided by SLCSE:

TRACE ON ENTITY 11_1_message_delivery BACKWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

11_1_message_delivery (BACKWARD) [message]
1- 7_2_data_storage depends_on 11_1_message_delivery
<--2- 14_1_tool_registration depends_on 7_2_data_storage
<--3- 10_1_task_definition depends_on 14_1_tool_registration
<--4- 10_2_task_execution depends_on 10_1_task_definition
<--5- 10_4_task_history depends_on 10_2_task_execution
<--6- 10_2_task_execution@ depends_on 10_4_task_history
<--5- 10_5_event_monitoring depends_on 10_2_task_execution
<--5- 10_6_audit_accounting depends_on 10_2_task_execution
<--3- 10_7_role_mgt depends_on 14_1_tool_registration
<--3- 10_8_tool_registration depends_on 14_1_tool_registration
<--3- 15_1_4_tool_to_om depends_on 14_1_tool_registration
<--4- 15_1_5_om_to_om depends_on 15_1_4_tool_to_om
<--5- 15_1_1_data_integration depends_on 15_1_5_om_to_om
<--2- 7_11_derivation depends_on 7_2_data_storage
<--2- 7_12_replication depends_on 7_2_data_storage
<--3- 15_1_6_consistency_mgt depends_on 7_12_replication
<--2- 7_13_access_control depends_on 7_2_data_storage
<--3- 13_2_security_control depends_on 7_13_access_control

<--2- 7_17_version depends_on 7_2_data_storage
<--3- 7_18_configuration depends_on 7_17_version
<--2- 7_19_query depends_on 7_2_data_storage
<--2- 7_21_state_monitoring depends_on 7_2_data_storage
<--3- 10_5_event_monitoring depends_on 7_21_state_monitoring
<--2- 7_23_data_interchange depends_on 7_2_data_storage
<--3- 7_10_backup depends_on 7_23_data_interchange
<--4- 7_9_archive depends_on 7_10_backup
<--2- 7_3_relationship depends_on 7_2_data_storage
<--2- 7_4_name depends_on 7_2_data_storage
<--2- 7_6_data_transaction depends_on 7_2_data_storage
<--2- 7_7_concurrency depends_on 7_2_data_storage
<--3- 15_1_6_consistency_mgt depends_on 7_7_concurrency
<--2- 9_3_1_data_integration depends_on 7_2_data_storage
<--2- 9_3_2_control_integration depends_on 7_2_data_storage
<--2- 9_3_3_ui_integration depends_on 7_2_data_storage

3.3.4.2 Tool Registration Service (11.2)

SLCSE does not explicitly provide support for Tool Registration of this kind.

3.3.5 Security (13)

3.3.5.1 Security Information Class (13.1)

SLCSE provides neither the Authentication Service, Attribute Service, nor the Interdomain Service. The VAX/VMS Operating System may provide some degree of these services, but it is beyond the scope of the current implementation of SLCSE.

3.3.5.2 Security Control Services (13.2)

3.3.5.2.1 ICE - Conceptual

SLCSE does not provide the Secure Association Service, but does provide a minimum degree of the Authorization Service, as described below.

AUTHORIZATION SERVICE: The authorization service is provided in part by SLCSE itself, in part by the VAX/VMS Operating System, and in part by the underlying RDBMS of the SLCSE Project Database.

Before a user can operate within an environment instantiation of the SLCSE framework, the SLCSE Command Executive checks the identification of the user as provided by the Operating System and determines, from an examination of the MASTER_PERSONNEL_LIST of the Infrastructure Database, if the user is assigned to the project specified by the user. If a match is not found, then the user is totally denied access to the environment.

Once within a session of SLCSE, access control on the Project Files Hierarchy is enforced by the Operating System according to the Access Control Lists (ACLs) and file protections established during framework administration. Access control on the Project Database of an environment is enforced both by the Command Executive and infra-coupled SLCSE applications at the subschema level according to the permissions specified using the SLCSE Environment Manager (SEM) during framework administration. In addition, access control on the Project Database of an environment is enforced by the underlying RDBMS at the entity and relationship level according to the permissions specified using the SEM during framework administration.

3.3.5.2.1.1 Example

If a user, identified by the Operating System as "Jones", attempts to use SLCSE for a project named "Foxfire", and the user is not registered on the MASTER_PERSONNEL_LIST of the Infrastructure Database, then the user is informed that he/she does not have access to the specified project, and the session with SLCSE ends.

3.3.5.2.2 ROD - Operations

SLCSE does not provide the Secure Association Service, but does provide a minimum degree of the Authorization Service, as described below.

AUTHORIZATION SERVICE: The authorization service is provided in part by SLCSE itself, in part by the VAX/VMS Operating System, and in part by the underlying RDBMS of the SLCSE Project Database.

The operations for the Authorization Service of SLCSE are authorize and deny.

3.3.5.2.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between

services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 13_2_security_control FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

13_2_security_control (FORWARD) [security]
1- 13_2_security_control depends_on 7_13_access_control
-->2- 7_13_access_control depends_on 7_2_data_storage
-->3- 7_2_data_storage depends_on 11_1_message_delivery
-->3- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->3- 7_2_data_storage depends_on 7_16_global_schema
-->4- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->4- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.5.3 Security Monitor Services (13.3)

SLCSE does not provide the Security Audit Information Collection Service. However, the VAX/VMS Operating System may provide a certain level of this service, but which is considered beyond the scope of SLCSE.

3.3.5.4 Related Services (13.4)

This service, although listed as a service in [2], is not actually a service, but rather an editorial error in [2]. "Related Services" actually refers to the dimension factor called "Relationships Between Services".

3.3.6 Framework Administration and Configuration (14)

3.3.6.1 Tool Registration (14.1)

3.3.6.1.1 ICE - Conceptual

The SLCSE Environment Manager (SEM) provides the Tool Registration Service, and allows tools to be registered with the framework for use within environments instantiated from the framework. This includes the identification of the tool's name and information particular to its integration with the framework.

3.3.6.1.1.1 Example

The following is a depiction of the information provided to SEM for the registration of the tool named ACS_LINK:

Tool invocation data form for: ACS_LINK

Setup window ID:	227
Ada procedure to invoke? . .	YES
TOOLER subprocess invocation? .	YES
Available in KEYWORD mode only? .	NO
Clear screen before invocation? . .	NO
Repaint screen after invocation? .	NO
Invocation mode:	BATCH_OR_INTERACTIVE
Direct tool output to: . . .	OUTPUT_FILE
Display completion status message? .	YES
Number of files required: . . .	1

3.3.6.1.2 ROD - Operations

The operations involved with Tool Registration in SLCSE are register and unregister. The SLCSE Environment Manager (SEM) provides these operations, which involves writing to the Infrastructure Database.

3.3.6.1.2.1 Example

To register a tool (e.g., the ACS linker) a name is assigned to the tool (e.g., ACS_LINK) and values are entered into to a Tool Invocation Data Form display of the SEM. SEM then creates a TOOL_RECORD for this information in the Infrastructure Database.

3.3.6.1.3 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on"

relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

It should be noted that the Entity-Relationship model established for this part of the mapping exercise considers this Tool Registration service to include all of the additional services provided by SLCSE in terms of Framework Administration, as described in the Service Mapping form for this service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 14_1_tool_registration FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

14_1_tool_registration (FORWARD) [framework_admin]
 1- 14_1_tool_registration depends_on 7_2_data_storage
 -->2- 7_2_data_storage depends_on 11_1_message_delivery
 -->2- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->2- 7_2_data_storage depends_on 7_16_global_schema
 -->3- 7_16_global_schema depends_on 15_1_2_common_data_descr
 -->3- 7_16_global_schema depends_on 7_1_data_model
```

This service is required by the following services which are provided by SLCSE:

```
TRACE ON ENTITY 14_1_tool_registration BACKWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

14_1_tool_registration (BACKWARD) [framework_admin]
 1- 10_1_task_definition depends_on 14_1_tool_registration
 <--2- 10_2_task_execution depends_on 10_1_task_definition
 <--3- 10_4_task_history depends_on 10_2_task_execution
 <--4- 10_2_task_execution@ depends_on 10_4_task_history
 <--3- 10_5_event_monitoring depends_on 10_2_task_execution
 <--3- 10_6_audit_accounting depends_on 10_2_task_execution
 1- 10_7_role_mgt depends_on 14_1_tool_registration
```

1- 10_8_tool_registration depends_on 14_1_tool_registration
1- 15_1_4_tool_to_om depends_on 14_1_tool_registration
<--2- 15_1_5_om_to_om depends_on 15_1_4_tool_to_om
<--3- 15_1_1_data_integration depends_on 15_1_5_om_to_om

3.3.7 Integration (15)

3.3.7.1 Data Integration (15.1)

3.3.7.1.1 Object Management as a Data Integration Mechanism (15.1.1)

3.3.7.1.1.1 ICE - Conceptual

The role of the Object Manager (OM) for the SLCSE Database Subsystem (in particular, the Project Database and the ERIF, the Infrastructure Database and DSA, and the Project Files Hierarchy and the VAX/VMS Operating System, in descending order of importance) in an environment instantiated from the SLCSE framework is to provide a manageable and sharable (by project personnel and environment tools within and between the life cycle phases of a software development project) data repository for all project information.

It is not assumed that all tools will use the same Object Manager in SLCSE, as desirable as this might be. In the SLCSE concept of operations, it is realized that an environment framework must be flexible to allow the integration of any tool, including one that has its own Object Manager. Two reasons for this are:

1. Productivity - People who do a job get used to doing it well with a set of tools that they adopt, and may not be willing to use tools that operate with the OM of an environment if those tools then behave differently than the ones that people are accustomed to.

2. Capitalization - It costs next to nothing and takes very little time to integrate a tool into the SLCSE framework if it is not required to change it (or, worse, required to develop from scratch an environment OM-coupled tool to replace it) to work with the Object Manager of an environment. It usually costs only a little more to build a shell of mechanisms (e.g., via DCL Interface Utilities) about the tool to transform and transfer useful data between the tool and the OM of an environment.

The highest expectation in SLCSE would be to have every tool

directly-coupled with a Project Database whose schema is especially devised according to the data produced by the tools, used by the tools, and required for the project. The major deficit in SLCSE for the DOD environment is the lack of enough tools to automatically populate the entire DOD-STD-2167A Project Database in a way that is transparent to the user, i.e., as a by-product of the software development process.

3.3.7.1.1.1.1 Example

The SLCSE Project Management System (SPMS) is an example of where the data from Commercial Off-The-Shelf (COTS) tools is integrated into the SLCSE Project Database via the OM of SLCSE (i.e., the HLERIF, Database Client/Servers, the ERIF, and RDBMS) and used by other directly-coupled SLCSE tools via the OM of SLCSE.

3.3.7.1.1.2 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

```
TRACE ON ENTITY 15_1_1_data_integration FORWARD
 9 LEVEL RELATIONSHIP
 -- AS A MEMBER OF [all] SUBSETS.

15_1_1_data_integration (FORWARD) [integration]
  1- 15_1_1_data_integration depends_on 15_1_5_om_to_om
 -->2- 15_1_5_om_to_om depends_on 15_1_4_tool_to_om
 -->3- 15_1_4_tool_to_om depends_on 14_1_tool_registration
 -->4- 14_1_tool_registration depends_on 7_2_data_storage
 -->5- 7_2_data_storage depends_on 11_1_message_delivery
 -->5- 7_2_data_storage depends_on 15_1_2_common_data_descr
 -->5- 7_2_data_storage depends_on 7_16_global_schema
 -->6- 7_16_global_schema depends_on 15_1_2_common_data_descr
```

-->6- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any of the mapped services.

3.3.7.1.2 Common Data Descriptions (15.1.2)

3.3.7.1.2.1 ICE - Conceptual

The following common data descriptions are defined or used by tools in the SLCSE environment:

Program Internal Forms - The Ada Test and Verification System (ATVS) uses DIANA trees to represent the structure of Ada programs for static and dynamic analyses.

Document Internal Forms - The Document Generation Language (DGL) of SLCSE is used by the DOCGEN tools, and, when processed, can produce a variety of text processor formats (e.g., LaTeX, DSR, LQP, etc.).

Pre-defined Schemas - The DOD-STD-2167A Entity-Relationship Model schema for a SLCSE Project Database defines (in the form of the Schema Definition Language (SDL)) all of the data objects and their relationships to such a fine granularity as to permit the automatic generation of all seventeen (17) of the data items (i.e., documents and specifications) required by the Military Standard for Mission Critical Computer System Software Development.

Data Interchange Formats - The DCL Interface Utilities of SLCSE have an ASCII text file format that can be used to facilitate the interchange of data between tools and environments. DECnet is another data interchange format used in the client-server architecture, distributed SLCSE environment.

3.3.7.1.2.1.1 Example

The following is a sample DGL DESCRIPTION text attribute possessed by an instance of the REPORT entity type that is used by the DOCGEN Report tool to produce a Digital Standard Runoff (DSR) file that, when it is processed by DSR, produces a report based on information stored in a SLCSE Project Database.

```
-- ================================================================
--
-- Name: CSCI_REQUIREMENTS_REPORT
```

```
--
-- Version: 1
--
-- Abstract: THIS REPORT TELLS WHAT THE CSCI ENGINEERING
-- REQUIREMENTS ARE FOR EACH CSCI.  THE OUTPUT
-- IS IN THE FORM OF DIGITAL STANDARD RUNOFF.
--
-- System:  VAX/VMS
--
-- Author:  JAMES R.  MILLIGAN
--
-- Date:  22-SEP-1990
--
-- ----------------------------------------

document CSCI_REQ_RPT is

translate off ;

text_line ".KEEP" ;
text_line ".NO FLAGS ALL" ;

-- Get CSCI information.  For each CSCI ...

variable CSCI_KEY ;
variable CSCI_ABBR ;
variable CSCI_NAME ;
variable CSCI_PURPOSE ;

query GET_CSCI_INFO is
        from "csci"
                "key" into CSCI_KEY
                "access_name" into CSCI_ABBR
                "descr_name" into CSCI_NAME
                "purpose"into CSCI_PURPOSE
        where " " ;

iterate over GET_CSCI_INFO

text_line ".CENTER" ;
text_line  "<**********************************************>"  ;
text_line ".SKIP" ;
text_line ".PARAGRAPH" ;
```

```
text_line "The CSCI " & CSCI_ABBR & ":" & CSCI_NAME & "," ;
text_line "the purpose of which is:" ;
text_line ".SKIP" ;
text_line ".LM +3" ;
text_line """ & CSCI_PURPOSE & """ ;
text_line ".SKIP" ;
text_line ".LM -3" ;
text_line "has the following CSCI engineering requirements:" ;
text_line ".NOFILL" ;
text_line ".SKIP" ;

table "CSCI Engineering Requirements" is
        format ASCII
                column "Requirement Type" width 16
                column "Requirement Name" width 47

-- ... get the relationship information between the CSCI and CSCI
engineering
-- requirements. For each relationship found ...

variable RANGE_KEY;

query GET_RELA_INFO is
        from "csci has csci_engineering_req"
                "range_key" into RANGE_KEY
        where "domain_key = " & CSCI_KEY ;

iterate over GET_RELA_INFO

-- ... get the requirement information.

variable REQ_ABBR ;
variable REQ_NAME ;
variable ENT_TYPE ;

query GET_REQ_INFO is
        from "csci_engineering_req"
                "access_name" into REQ_ABBR
                "descr_name" into REQ_NAME
                "entity_type" into ENT_TYPE
        where "key = " & RANGE_KEY ;

iterate over GET_REQ_INFO
```

```
text ENT_TYPE ;
text REQ_ABBR & ":" & REQ_NAME ;

end iterate ;
end iterate ;
end table ;

text_line ".FILL" ;
text_line ".PAGE" ;

end iterate ;

end document ;
```

3.3.7.1.2.2 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service does not require any of the mapped services.

This service is required by the following services which are provided by SLCSE:

```
TRACE ON ENTITY 15_1_2_common_data_descr BACKWARD
 9 LEVEL RELATIONSHIP
 – AS A MEMBER OF [all] SUBSETS.

15_1_2_common_data_descr (BACKWARD) [integration]
 1- 7_16_global_schema depends_on 15_1_2_common_data_descr
 <--2- 7_14_constraint_mgt depends_on 7_16_global_schema
 <--2- 7_20_metadata depends_on 7_16_global_schema
 <--2- 7_2_data_storage@ depends_on 7_16_global_schema
 1- 7_2_data_storage depends_on 15_1_2_common_data_descr
 <--2- 14_1_tool_registration depends_on 7_2_data_storage
```

<--3- 10_1_task_definition depends_on 14_1_tool_registration
<--4- 10_2_task_execution depends_on 10_1_task_definition
<--5- 10_4_task_history depends_on 10_2_task_execution
<--6- 10_2_task_execution@ depends_on 10_4_task_history
<--5- 10_5_event_monitoring depends_on 10_2_task_execution
<--5- 10_6_audit_accounting depends_on 10_2_task_execution
<--3- 10_7_role_mgt depends_on 14_1_tool_registration
<--3- 10_8_tool_registration depends_on 14_1_tool_registration
<--3- 15_1_4_tool_to_om depends_on 14_1_tool_registration
<--4- 15_1_5_om_to_om depends_on 15_1_4_tool_to_om
<--5- 15_1_1_data_integration depends_on 15_1_5_om_to_om
<--2- 7_11_derivation depends_on 7_2_data_storage
<--2- 7_12_replication depends_on 7_2_data_storage
<--3- 15_1_6_consistency_mgt depends_on 7_12_replication
<--2- 7_13_access_control depends_on 7_2_data_storage
<--3- 13_2_security_control depends_on 7_13_access_control
<--2- 7_17_version depends_on 7_2_data_storage
<--3- 7_18_configuration depends_on 7_17_version
<--2- 7_19_query depends_on 7_2_data_storage
<--2- 7_21_state_monitoring depends_on 7_2_data_storage
<--3- 10_5_event_monitoring depends_on 7_21_state_monitoring
<--2- 7_23_data_interchange depends_on 7_2_data_storage
<--3- 7_10_backup depends_on 7_23_data_interchange
<--4- 7_9_archive depends_on 7_10_backup
<--2- 7_3_relationship depends_on 7_2_data_storage
<--2- 7_4_name depends_on 7_2_data_storage
<--2- 7_6_data_transaction depends_on 7_2_data_storage
<--2- 7_7_concurrency depends_on 7_2_data_storage
<--3- 15_1_6_consistency_mgt depends_on 7_7_concurrency
<--2- 9_3_1_data_integration depends_on 7_2_data_storage
<--2- 9_3_2_control_integration depends_on 7_2_data_storage
<--2- 9_3_3_ui_integration depends_on 7_2_data_storage

3.3.7.1.3 Tool-to-Tool Data Translators (15.1.3)

SLCSE supports an alternate concept, and that is tool-to-tool data translation via the SLCSE Database Subsystem, i.e., tool-to-OM-to-tool data translation. OM-to-tool data translation provided by SLCSE (and also tool-to-OM data translation) is described in the service description for Tool-to-OM Translators (15.1.4).

3.3.7.1.4 Tool-to-OM Translators (15.1.4)

3.3.7.1.4.1 ICE - Conceptual

SLCSE provides both tool-to-OM data translation and OM-to-tool data translation services. It is these services which provide the tool integration framework of SLCSE.

The Digital Command Language (DCL) Interface Utilities of SLCSE can be used to develop "shells" about a tool for data translation to and from the SLCSE Project Database. A tool that exports data (preferrably in a documented ASCII format) can have a data translator written to transform its data into the format of a DCL Database Utility text file format. Likewise, a tool that imports data (again, preferrably in a documented ASCII format) can have a data translator written to transform a DCL Database Utility text file format into the format of the tool. The DCL Database Utilities provide the capability to import and export DCL Database Utility text file information to and from the SLCSE Project Database via the Enitity-Relationship InterFace (ERIF).

In a level of complexity slightly higher than the concept of the DCL Interface Utilities, data translators can be written that directly use the ERIF.

For tools that are resident on network nodes that are remotely connected to the host platform for the SLCSE Project Database RDBMS, data translators can be written for these tools utilizing the Higher-Level ERIF (HLERIF) and the Database Client/Servers.

3.3.7.1.4.1.1 Example

The Ada Test and Verification System (ATVS) has a data translator that exports its data into the form of the DCL Interface Utility Database Insert text file format.

The Automated Life Cycle Impact Analysis (ALICIA) System has a data translator that directly uses the ERIF, and imports its data from the SLCSE Project Database into text files used by the ALICIA System.

The SLCSE Project Management System (SPMS) contains Macintosh-based Commercial Off-The-Shelf (COTS) tools that have data translators (called Companion Database Interface (CDI) tools) that utilize the HLERIF and the Database Client/Servers.

3.3.7.1.4.2 Relationships Between Services

An Entity-Relationship model of the dependencies between each of the SLCSE services was developed to determine the relationships between services. Each service was modeled as an entity with various "depends_on" relationships to other services. An analysis on the model using the SLCSE analyzER tool resulted in the generation of forward and backward "trace" reports that were optimized to eliminate redundant relationship information. Forward trace reports on a service show the services that are required by the service. Backward trace reports on a service show the services that require the service.

This service requires the following services which are provided by SLCSE:

TRACE ON ENTITY 15_1_4_tool_to_om FORWARD
9 LEVEL RELATIONSHIP
-- AS A MEMBER OF [all] SUBSETS.

15_1_4_tool_to_om (FORWARD) [integration]
1- 15_1_4_tool_to_om depends_on 14_1_tool_registration
-->2- 14_1_tool_registration depends_on 7_2_data_storage
-->3- 7_2_data_storage depends_on 11_1_message_delivery
-->3- 7_2_data_storage depends_on 15_1_2_common_data_descr
-->3- 7_2_data_storage depends_on 7_16_global_schema
-->4- 7_16_global_schema depends_on 15_1_2_common_data_descr
-->4- 7_16_global_schema depends_on 7_1_data_model

This service is not required by any other mapped services.

3.3.7.1.5 OM-to-OM Exchange (15.1.5)

The OM-to-OM Exchange service of SLCSE is a concept identical to that of the Tool-to-OM Translators (15.1.4) service of SLCSE. The ability to apply this service depends on how open of an architecture a tool or an OM is in terms of importing and exporting its internal data.

3.3.7.1.6 Consistency Management (15.1.6)

This service is implemented as part of the Concurrency Service (7.7) and as part of the Replication/Synchronization Service (7.12).

3.4 Comments on the Reference Model

3.4.1 Applicability of Dimension Factors for SLCSE

Comments, particularly with regard to SLCSE, on the applicability of the various factors for each dimension defined in the NIST ISEE Reference Model are provided in the following sections.

3.4.1.1 ICE - Internal

Due to time constraints, the mapping exercise did not utilize this factor to describe any of the services in SLCSE. However, this factor could be used for almost every SLCSE service to describe the implementation details involved with a service. This, on the other hand, would be very laborious and perhaps of minimal value (i.e., who would take the time to read it?).

3.4.1.2 ICE - Conceptual

This is the most important factor of the ICE dimension in the description of SLCSE services, and perhaps the most important factor of all. It allows an overview description of a service, and can be used to describe every service in SLCSE.

3.4.1.3 ICE - External

Due to time constraints, the mapping exercise did not utilize this factor to describe any of the services in SLCSE. However, the external factor would apply most to services that have interfaces with other services and tools in the SLCSE environment.

3.4.1.4 ROD - Rules

Due to time constraints, the mapping exercise did not utilize this factor to describe any of the services in SLCSE. However, it is likely that the rule factor would be beneficial when describing services that are associated with constraints on their application by other services, tools, or the user.

3.4.1.5 ROD - Operations

This is the most important factor of the ROD dimension in the description of SLCSE services. It is very useful to describe how a service works in a functional kind of way. The operations factor would seem highly related to the external factor of the ICE dimension.

3.4.1.6 ROD - Data

Due to time constraints, the mapping exercise did not utilize this factor to describe any of the services in SLCSE. However, for those services to which the operations factor apply, the data factor usually would also apply in the description of a SLCSE service.

3.4.1.7 TIM - Types

Due to time constraints, the mapping exercise did not utilize this factor to describe any of the services in SLCSE. However, few services other than those that deal with the semantic properties of the Project Database, Infrastructure Database, and Project Files Hierarchy of SLCSE (in particular, the Data Model, Global/Canonical Schema, and Data Storage Service/Persistence services) would apply to this factor.

3.4.1.8 TIM - Instances

Due to time constraints, the mapping exercise did not utilize this factor to describe any of the services in SLCSE. However, few services other than those that deal with the semantic properties of the Project Database, Infrastructure Database, and Project Files Hierarchy of SLCSE (in particular, the Data Model, Global/Canonical Schema, and Data Storage Service/Persistence services) would apply to this factor.

3.4.1.9 TIM - Metadata

Due to time constraints, the mapping exercise did not utilize this factor to describe any of the services in SLCSE. However, few services other than those that deal with the semantic properties of the Project Database, Infrastructure Database, and Project Files Hierarchy of SLCSE (in particular, the Data Model, Global/Canonical Schema, and Data Storage Service/Persistence services) would apply to this factor.

3.4.2 Comments on Existing Service Descriptions

The following sections discuss the service descriptions of the Reference Model in terms of what was discovered when SLCSE services were mapped to them.

3.4.2.1 Object Management Services (7)

3.4.2.1.1 Data Model (7.1)

The data models in SLCSE fully implement the service described in the reference model. However, none of the data models are truly object-oriented, and, therefore, do not support the concept of methods associated with objects nor multiple inheritance.

3.4.2.1.2 Relationship Service (7.3)

The Relationship Service in SLCSE is fully implemented for Project Database Objects, implemented with limitations for Infrastructure Database Objects, and not implemented for Project Files Hierarchy Objects.

3.4.2.1.3 Distribution/Location Service (7.5)

A SLCSE Project Database is both logically and physically centralized, and therefore, there is no concept of a logically centralized database mapping to a physically distributed database in SLCSE. However, there are facilities that allow for the management of Project Database Objects on distributed computing platforms that are not the actual platform hosting a Project Database. This alternate concept is described more fully in the mapping to the Replication/Synchronization Service of the Reference Model (paragraph 7.12 of the Reference Model).

3.4.2.1.4 Data Transaction Service (7.6)

The Data Transaction Service is provided by SLCSE for transactions involving Project Database objects, but is not provided for Infrastructure Database or Project Files Hierarchy objects.

3.4.2.1.5 Concurrency Service (7.7)

This service is provided only for Project Database objects.

This service is not provided for Infrastructure Database objects, but, if it was, would be advantageous when framework administration occurs simultaneously while SLCSE user sessions are in progress.

For User Files Hierarchy objects, concurrent operations on them do not typically occur since these files are local to a SLCSE user. Database Files Hierarchy objects are under the jurisdiction of the Project Database which covers the Concurrency Service. Concurrent operations on Maintenance Files Hierarchy objects are not of consequence because they involve only read-only and sharable executable files used by the SLCSE Command Executive and SLCSE tools.

3.4.2.1.6 Process Support Service (7.8)

SLCSE does not provide the Process Support Service as described in the Reference Model. However, the VMS operating system upon which SLCSE is currently implemented does. The VMS operating system provides for prioritized process scheduling queues, process control blocks (for process state information), process state transition control (e.g., make wait, execute, prepare for execution, swap out, swap in, etc.), input/output process requests handling, and a number of utilities for querying about process information (e.g., the accounting utility, the show process utility, and the monitor process utility) and for changing processes (e.g., the set utility). SLCSE provides tool integration services for the incorporation of such utilities as tools in an environment instantiation (e.g., a tool that was integrated into SLCSE was the CPU tool, which, at the VMS level, translates to the "monitor process/topcpu" utility).

3.4.2.1.7 Archive Service (7.9)

The VMS Operating System (OS), upon which SLCSE is currently implemented, provides a complete Archive Service for files stored on disk. Through a combination of the VMS backup (to tape from disk) utility (reference [19]), and the Backup Service of SLCSE (reference service mapping to Reference Model section 7.10, Backup Service), all data can be moved off-line, and restored.

3.4.2.1.8 Derivation Service (7.11)

It is assumed that SLCSE does not provide a full spectrum Derivation Service, but, on the other hand, it is not clear what a full spectrum Derivation Service would provide based on the current version of the Reference Model.

3.4.2.1.9 Replication/Synchronization Service (7.12)

This service is not provided for Infrastructure Database or Project Files Hierarchy Objects, since they are never replicated within a distributed environment for any constructive purpose in the context of SLCSE.

3.4.2.1.10 Constraint/Inconsistency Management (7.14)

For a SLCSE Project Database, while it is possible to define, maintain and enforce constraints on entity and relationship attributes and the bounds on their values, the ability to dynamically enable and disable these constraints

does not exist in SLCSE (with the exception of access control constraints). Similarly, constraints on the values and properties of Infrastructure Database objects are maintained and enforced, but cannot be dynamically enabled and disabled (i.e., an item in a list must conform to the type specified for the items that the list is intended to hold). Constraints on file objects are maintained, enforced, enabled, and disabled by the services of the Operating System, and will not be described in the scope of this mapping exercise.

3.4.2.1.11 Function Attachment (7.15)

SLCSE does not provide for Function Attachment to objects, as none of its data models are truly object-oriented. In SLCSE, Function Attachment to objects would be done, alternatively, through tool integration into an environment instantiated from the framework. The tools would then provide the functions associated with objects.

3.4.2.1.12 Global/Canonical Schema (7.16)

SLCSE does not provide a Global/Canonical Schema of process definitions, but does provide a Global/Canonical Schema of data in support of the life cycle processes that occur within an environment instantiation.

3.4.2.1.13 Version Service (7.17)

SLCSE does not provide this service for the Infrastructure Database.

3.4.2.1.14 Configuration Service (7.18)

SLCSE does not provide this service for the Infrastructure Database.

3.4.2.1.15 Metadata Service (7.20)

Metadata about an Project Files Hierarchy data is not provided in SLCSE to any great extent.

3.4.2.1.16 State Monitoring Service/Triggering (7.21)

Although SLCSE provides such a service for the Project Database, it is not one which involves the instantaneous triggering of events when the state of the database changes, but, rather, involves the triggering of events when database state changes are periodically examined for and detected. The limitations of the actions that can be triggered as the result of database events are described with the "ICE - Conceptual" factor for this service.

SLCSE does not provide this service in relation to the Infrastructure Database and Project Files Hierarchy.

3.4.2.1.17 Sub-Environment (Views) Service (7.22)

Unless the environment of a single user (with roles, tools, and data access constraints specific to the user) on a project (consisting of multiple users) can be considered as a sub-environment of an environment consisting of the entire set of user environments for a project, then this service is not provided in SLCSE.

3.4.2.1.18 Data Interchange Service (7.23)

This service is provided explicitly by SLCSE only for the Project Database, although the Import and Export tools of SLCSE may augment the Operating System Data Interchange Services provided by VMS (e.g., the Backup Utility) which are not addressed here.

3.4.2.1.19 Tool Registration (7.24)

Although the data used/produced by tools is registered in the SLCSE Project Database (by defining the necessary entity types, etc.), the object manager is not aware of which tools are integrated in SLCSE. Tool registration, similarly, is not provided for the Infrastructure Database or the Project Files Hierarchy (i.e., DSA does not know which tools are recorded in the Infrastructure Database, and the VAX/VMS Operating System is not aware of the changes made to the Maintenance Files Hierarchy of the Project Files Hierarchy in support of tool integration).

3.4.2.2 Tools (9)

3.4.2.2.1 Tool Integration (9.3)

It is proposed to merge Reference Model section 9.3, "Tool Integration", with section 15, "Integration".

3.4.2.2.2.1 Data Integration (9.3.1)

All degrees of data integration described in the Reference Model exist within the current toolset of SLCSE. However, not every tool is data integrated in the same fashion, as described with the "ICE - Conceptual" factor for this service.

In general, the Reference Model does not provide a service for describing either the tools offered and currently integrated in an environment, the functions and forms of integration for each tool, or the taxonomy used in an environment for tool classification. The current tools of the SLCSE environment, for example, are classified to fall into the following categories:

GENERAL SUPPORT

Tools which support activities that span all user roles, e.g., text editors, electronic mail, document formatters, etc.

REQUIREMENTS DEFINITION

Tools which support system/software requirements definition and modification, display and reporting, and consistency checking.

DESIGN

Tools which support allocation of requirements to design objects (e.g., computer software components and units) and description of interfaces, data, and processing algorithms.

CODING

Tools which support creation, modification, compilation, and debugging of source code, linking of object code and execution of programs implementing design.

PROJECT MANAGEMENT

Tools which support project planning, tracking, monitoring, reporting, assessment, adjustment.

PROTOTYPING

Tools which support the rapid construction of interactive user interfaces to provide developers and end-users of a system the "look and feel" of a system before the commitment to an undesirable design, and possibly to serve as the actual user interface of the completed product.

TESTING/VERIFICATION AND VALIDATION/QUALITY ASSURANCE

Tools which support test case definition, organization, evaluation, and problem reporting as well as the collection and evaluation of software quality metrics, programming standards, consistency, and traceability.

CONFIGURATION MANAGEMENT

Tools which support the definition, modification, control, and analysis (e.g., change impact analysis) of software configurations.

ENVIRONMENT MANAGEMENT

Tools which support SLCSE environment definition, modification, and tailoring (e.g., for projects, databases, users, roles, rules, and tools as well as tool and data integration services).

For each category, a representative set of tools was integrated into SLCSE. Some of these tools were specifically developed and integrated for use within SLCSE (Developmental Software (DS) tools), while others required no development, and were simply integrated as Non-Developmental Software (NDS) tools.

The forms of tool integration with SLCSE are as follows:

CONFORMANT

Tool user interface style conforms to that of the SLCSE User Interface.

NON-CONFORMANT

Tool user interface style does not conform to that of the SLCSE User Interface.

DIRECTLY-COUPLED

Tool uses the SLCSE Project Database directly through the ERIF/HLERIF.

INDIRECTLY-COUPLED

Tool uses the SLCSE Project Database via the DCL Interface Utilities or some similar interface that does not by-pass the ERIF/HLERIF.

LOOSELY-COUPLED

Tool does not use the SLCSE Project Database.

INFRA-COUPLED

Tool uses the SLCSE Infrastructure Database via DSA.

NON-INFRA-COUPLED

Tool does not use the SLCSE Infrastructure Database.

DEPENDENTLY-COUPLED

Tool is totally dependent on the SLCSE Project Database and/or the SLCSE Infrastructure Database to maintain its data, and has no persistent database of its own.

CONTROL-COUPLED

Tool is invokable by SLCSE as a sub-process that must return control to SLCSE upon the completion of its execution.

NON-CONTROL-COUPLED

Tool not invokable by SLCSE, but is integrated with the SLCSE Project Database and/or the SLCSE Infrastructure Database.

The NDS tools that were integrated with the SLCSE user interface and/or database subsystems include:

ADA COMPILATION SYSTEM (ACS) - allows users to perform a variety of operations (e.g., link, compile, check, recompile, etc.) on Ada objects (e.g., libraries, units, etc.). ACS is integrated into SLCSE with its own non-conformant line-oriented command user interface.

ACS LINKER - allows users to link compiled Ada source code objects to create executable images. SLCSE provides the DEC ACS linker with a setup window for user specification of command parameters.

ACTA - a Macintosh-based outline manager that provides the user with the capability to manage project organization hierarchies. ACTA is part of the MacSLCSE/Micro Import toolset.

ADA COMPILER - allows users to compile Ada source code. SLCSE provides the DEC Ada compiler with a setup window.

ADA DESIGN LANGUAGE (ADL) PROCESSOR - allows users to process ADL specifications. SLCSE provides the ADL processor with a setup window.

ANALYZER - The analyzER provides automated support for E-R model analysis, verification, and documentation. It produces a variety of textual reports that fall into four general categories. Graphical displays and hard-copies are also generated by the analyzER for two of these categories. In addition, the analyzER performs both syntactic and semantic analyses with diagnostics reporting for ER models defined using Prolog constructs. The analyzER is integrated with a SLCSE-conformant user interface.

AUTOMATED LIFE CYCLE IMPACT ANALYSIS (ALICIA) SYSTEM - allows users to navigate an ER model of a Project Database and to identify instances of entity and relationship types which are impacted by a particular change, and provides a variety of algorithms for the reporting of potential impacts throughout the database. The ALICIA system was implemented on a VAXstation II workstation with a user interface developed using the DEC Graphical Kernel System (GKS), and interfaces with a SLCSE Project Database via the ERIF.

AUTOMATED MEASUREMENT SYSTEM (AMS) - provides automated quality measurement of Ada and Fortran source code as well as quality information about the entire software life cycle for MCCS development. The AMS is the precursor tool of the QUality Evaluation System (QUES), and is integrated with SLCSE having its own user interface style (based heavily on ReGiS graphics) and data repository.

AMS ANALYZE - allows users to process AMS files. AMS Analyze is integrated with SLCSE having a conformant user interface.

CLEAR - allows users to clear the field for the name of the current file object upon which tools, by default, perform operations.

COBOL AUTOMATED VERIFICATION SYSTEM (CAVS) - allows users to perform a variety of static and dynamic tests on computer programs written in the COBOL programming language in support of verification activities associated with the coding, integration and maintenance phases of the software life cycle. CAVS is integrated into SLCSE with its own non-conformant user interface.

COBOL COMPILER - allows users to compile COBOL source code. SLCSE provides the DEC COBOL compiler with a setup window.

COPY - allows users to make copies of files in their local workspace. SLCSE provides the Copy utility with a setup window.

CPU - is one of the MONITOR PROCESS/TOPCPU utilities of the VAX/VMS Operating System. CPU is integrated with its own user interface style.

DIGITAL STANDARD RUNOFF (DSR) - allows users to process DSR input files to produce formatted documents. SLCSE provides the DSR tool with a pop-up window for an input file name specification.

DIRECTORY - allows users to show a list of the files in their local workspace. SLCSE provides the Directory utility with a setup window.

EDT and EVE - allow users to edit files in their local workspace. SLCSE provides the DEC Edt and Eve text editors with setup windows.

EXCEL - a Macintosh-based spreadsheet application useful for Organizational Breakdown Structure (OBS) and Work Breakdown Structure (WBS) construction and analysis. Excel is part of the MacSLCSE/Micro Import toolset, and also is directly-coupled to the SLCSE Project Database as part of the SLCSE Project Management System (SPMS).

FORTRAN COMPILER - allows users to compile Fortran source code. SLCSE provides the DEC Fortran compiler with a setup window.

JOVIAL J73 AUTOMATED VERIFICATION SYSTEM (J73AVS) - allows users to perform static and dynamic analysis testing of JOVIAL J73 source code. SLCSE provides the J73AVS tool with a setup window.

JOVIAL J73 COMPILER - allows users to compile JOVIAL J73 source code. SLCSE provides the JOVIAL compiler with a setup window.

KERMIT - Kermit is a file transfer protocol. It allows the transfer of files over terminal lines from a remote Kermit program to the local Kermit program. Kermit is integrated into SLCSE with its own user interface style.

LANGUAGE QUALITY PRINTER (LQP) - allows users to format and print documents. SLCSE provides the LQP tool with a setup window.

LANGUAGE SENSITIVE EDITOR (LSE) - provides users programming language sensitive editing functions. SLCSE provides the DEC LSE tool with a setup window.

LATEX - allows users to produce high-quality typesetting of documents. SLCSE provides the LaTeX document formatting system with a setup window.

LINK - allows users to link object code produced by DEC compilers to produce executable images. SLCSE provides the DEC Link utility with a setup window.

LOTUS 1-2-3 - an integrated commercial software package supporting spreadsheet analysis, database management, and business graphics. LOTUS

1-2-3 is integrated into SLCSE with its own proprietary user interface, and is used as a building block for the SPMS WBS/OBS Editors.

MACPROJECT II - a commercial Macintosh-based project management software package providing a variety of functions for project planning, assessment, adjustment, and reporting. MacProject II has its own proprietary user interface, and is used as a building block in the SPMS Macintosh toolset.

MACRO - allows users to assemble VAX MACRO assembly code to produce object code. SLCSE provides the DEC VAX MACRO assembler with a setup window.

MAIL - a VAX/VMS Personal Mail Utility (MAIL), which is used to send messages electronically between users of the system. MAIL is integrated with SLCSE having its own proprietary user interface.

MICRO PLANNER PLUS - a Macintosh-based commercial project management software package that forms a part of the MacSLCSE/Micro Import toolset, and is indirectly-coupled with the SLCSE Project Database.

OBJECTS - allows users to view the files in their local workspace while operating the SLCSE in keyword-driven mode.

PIGMY - an interactive graphics utility that is integrated into SLCSE with its own user interface style.

PRINT - allows users to print files. SLCSE provides the DEC Print utility with a setup window.

PROCEDURE - allows users to execute Digital Command Language (DCL) command procedure files. SLCSE provides the Procedure utility with a setup window.

PROJECT EXCHANGE - a Macintosh-based commercial software application used to translate Micro Planner Plus project management information into an ASCII format that is transferrable to the VAX, and translatable by the VAX-resident Micro Import tool for loading data into the SLCSE Project Database via DCL Interface Utilities. It forms a part of the MacSLCSE toolset.

PURGE - allows users to purge (delete all but the latest version file) all files (or only those files matching a particular name specification) in their

local SLCSE workspace. SLCSE provides the Purge utility with a pop-up window for a file name specification.

RENAME - allows users to change the name of a file in their local workspace. SLCSE provides the Rename utility with a setup window.

RUN - allows users to execute images (compiled and linked programs) residing in their local workspace. SLCSE provides the Run utility with a setup window.

RXVP80 - allows users to perform a variety of static and dynamic tests on Fortran computer programs in support of verification activities associated with the coding, integration and maintenance phases of the software life cycle. RXVP80 is integrated into SLCSE with its own non-conformant user interface.

SDESIGN - allows users to design and create applications for a relational database. SLCSE provides the SmartStar SDESIGN tool with a pop-up window the user specification of an application file name.

SELECT - allows users to select a file as the current object upon which a tool will operate while working in the SLCSE keyword-driven mode.

STRUCTURED QUERY LANGUAGE (SQL) INTERPRETER - allows users to perform relational database operations via SmartStar SQL commands and queries. SLCSE provides the SQL interpreter with a pop-up window the user specification of a hardware or software target database implementation.

SQUERY - allows users to execute database applications created by SDESIGN. SLCSE provides the SmartStar SQUERY tool with a pop-up window the user specification of an application file name.

SREPORT - allows users to create reports from the database applications created by SDESIGN. SLCSE provides the SmartStar SREPORT tool with a pop-up window the user specification of an application file name.

SOFTWARE DESIGN AND DOCUMENT LANGUAGE (SDDL) PROCESSOR - allows users to process SDDL specifications. SLCSE provides the SDDL processor with a setup window.

TEXPRINT - allows users to process device-independent LaTeX files into device-dependent output files which are then printed on a specified device. SLCSE provides the TeXPrint utility with a setup window.

TYPE - allows users to display the contents of a file in their local workspace on the terminal screen. SLCSE provides the Type utility with a setup window.

The DS tools that were integrated with the SLCSE user interface and/or database subsystems include:

ADDFILE - allows users to add source code and Program Design Language (PDL) files in their local workspace to the Project Database where they will be associated with a Computer Software Configuration Item (CSCI), Computer Software Component (CSC), or Computer Software Unit (CSU), and placed under SLCSE configuration management control (i.e., baselined). Addfile interfaces with the Project Database via the ERIF.

BASELINER - allows users to create/modify configurations, include entity type instances in configurations, baseline configurations, and generate reports describing the contents of a configuration. BaselinER interfaces with the Project Database via the ERIF.

CREATE - allows users to create files (or folders to contain files) in their local workspace. SLCSE provides the Create tool with a setup window.

DELETE - allows users to delete files or folders in their local workspace. SLCSE provides the Delete tool with a setup window.

DIGITAL COMMAND LANGUAGE (DCL) INTERFACE TOOLS - a set of tools for manipulating the Project Database from the VMS DCL level (i.e., Database Load, Unload, Retrieve, Insert, Update, Delete, Duplicate, Reserve, Release, and Purge; and Entity/Relationship Retrieve, Insert, Update, Delete, Reserve, Release). All DCL Interface tools access the SLCSE Project Database via the ERIF.

DESIGN - allows users to populate and modify Project Database subschemas relevant to design. Entity and relationship type instances can be created, modified, and deleted. Attribute values for entity type instances can be defined and modified. The Design tool interfaces with the Project Database via the ERIF.

DOCGEN 2167A - allows users to generate documents conformant with DOD-STD-2167A Data Item Descriptions (DIDs) from the contents of the Project Database. DOCGEN 2167A interfaces with a DOD-STD-2167A Project Database via the ERIF.

DOCGEN REPORT - allows users to generate customized reports based on the contents of the Project Database. DOCGEN Report interfaces with a Project Database via the ERIF.

EDITER - is used by several SLCSE applications to edit the contents of a Project Database. Those applications include: BaselinER, Design, DOCGEN 2167A, DOCGEN Report, ModifyER, PCRP, Requirements, Test Manager. EditER can also be used independently within SLCSE. EditER interfaces with a Project Database via the ERIF.

EXIT - a special SLCSE tool executed only at SLCSE session termination, and exists primarily to allow the establishment of pre-invocation and post-execution rule definitions at SLCSE wrap-up time.

EXPORT - allows users to export files from their local workspace to the external VMS file system or to a Software Development Folder (SDF) associated with a CSCI, CSC, or CSU. Export interfaces with the Project Database via the ERIF.

GETFILE - allows users to copy source code and PDL files from the Project Database that are under SLCSE configuration management control for CSCI, CSC, or CSU entity type instances to the user's local workspace. Getfile interfaces with the Project Database via the ERIF.

IMPORT - allows users to import files from the external VMS file system or a SDF associated with a CSCI, CSC, or CSU to their local SLCSE workspace. Import interfaces with the Project Database via the ERIF.

LOGIN - a special SLCSE tool executed only at SLCSE initialization time, and exists primarily to allow the establishment of pre-invocation and post-execution rule definitions at SLCSE start-up time.

MACSLCSE - a Macintosh-based project management software integration program implemented as a Hypercard stack that automates the transfer of Micro Planner Plus/Project Exchange, Acta, More, and Excel files to the VAX for manipulation and transfer to the SLCSE Project Database via Micro Import.

MENU - a special SLCSE tool used to return to the default menu-driven mode of SLCSE operation from the keyword-driven mode of SLCSE operation.

MICRO_IMPORT - allows users to upload data from Macintosh-based project management related commercial software tools (i.e., MORE and Acta, outline managers for project organization hierarchies; Micro Planner Plus and Project Exchange, a project management package; and Excel, a spreadsheet) to the Project Database.

MODIFYER - allows users to edit and browse the Project Database. ModifyER interfaces with the Project Database via the ERIF.

MORE - a Macintosh-based outline manager that provides the user with the capability to manage project organization hierarchies. MORE is part of the MacSLCSE/Micro Import toolset.

PROBLEM CHANGE REPORT PROCESSOR (PCRP) - allows users to populate and modify Project Database subschemas relevant to problem and change reporting and tracking. Entity and the relationship type instances can be created, modified, and deleted. Attribute values for entity type instances can be defined and modified. PCRP interfaces with the Project Database via the ERIF.

PUTFILE - allows users to return source code and PDL files that are associated with CSCI, CSC, and CSU entity type instances (files that were previously extracted from the Project Database to a user's local workspace via GETFILE) to the Project Database where they will be placed back under SLCSE configuration management control. Putfile interfaces with a the Project Database via the ERIF.

REPORTER - allows users to generate reports about the instances of selected entity and relationship types in the Project Database. ReportER interfaces with a the Project Database via the ERIF.

REQUIREMENTS - allows users to populate and modify Project Database subschemas relevant to system and software requirements. Entity and the relationship type instances can be created, modified, and deleted. Attribute values for entity type instances can be defined and modified. The Requirements tool interfaces with a the Project Database via the ERIF.

SDF CREATE - allows users to create a Software Development Folder (SDF) for a CSCI, CSC, or CSU entity type instance. SDF Create interfaces with a the Project Database via the ERIF.

SDF DELETE - allows users to delete a SDF for a CSCI, CSC, or CSU entity type instance. SDF Delete interfaces with a the Project Database via the ERIF.

SDL COMPILER - allows users to compile ER model Schema Definition Language (SDL) specifications to create SmartStar SQL code for the creation of a Project Database as well as run-time support files for applications that use the ERIF.

SDL CONVERT - permits the conversion of SDL into a Prolog database of facts about entities, relationships, and subschemas. SDL Convert output is used as input by the AnalyzER tool, and is integrated into SLCSE with a setup window.

SLCSE ENVIRONMENT MANAGER (SEM) - supports environment definition, creation, modification, and tailoring (e.g., defining users, roles, tools, Project Database access privileges, etc., as well as creating supporting files and directory structures) for specific software development projects within the context of the SLCSE framework.

TEST MANAGER - allows users to populate and modify Project Database subschemas relevant to system and software testing. Entity and the relationship type instances can be created, modified, and deleted. Attribute values for entity type instances can be defined and modified. The Test Manager interfaces with the Project Database via the ERIF.

TOOLS - allows users to view the list of tools for their current role while operating SLCSE in its keyword-driven mode.

VERIFYER - allows users to check the consistency of a Project Database through the verification of the existence of selected entity and relationship type instances. VerifyER interfaces with the Project Database via the ERIF.

WINNIE - a prototyper that allows users to develop windows for data driven application user interfaces. WINNIE is has a SLCSE-conformant user interface, and was, in fact, a tool used in the development of the SLCSE user interface.

In addition to DS tools, several other tools have been developed/integrated to operate within the SLCSE environment:

ADA TEST AND VERIFICATION SYSTEM (ATVS) - allows users to perform a variety of static and dynamic tests on computer programs written

in the Ada programming language in support of verification activities associated with the coding, integration and maintenance phases of the software life cycle. The ATVS is integrated with a SLCSE-conformant user interface, and exports data which is transferrable to a Project Database that contains an ATVS subschema via the Database Insert tool.

ATVS LIBRARY READER - an Ada library manager of the ATVS with a conformant user interface.

ATVS STANDARDS DEFINER - an Ada coding standards definer of the ATVS with a conformant user interface. This tool allows standards violations established by a project manager to be flagged during the static analysis of Ada source code by the ATVS.

C COMPILER - allows users to compile C programming language source code. SLCSE provides the C compiler with a setup window.

DATABASE ACCESS MANAGER - a Macintosh-based SPMS application used to manage data imported and exported to and from the SLCSE Project Database via HLERIF.

DATABASE INSERT - one of the DCL Interface Utilities. SLCSE provides a setup window for Database Insert to import information (from tools like ATVS) that is in a DCL Interface Tool file format to the Project Database.

DATABASE MONITOR MANAGER - a VAX-based SPMS application with a SLCSE-conformant user interface that is used to specify Project Database events to be monitored, the schedule for the monitoring, and the actions to be taken in the occurrence of an event.

DATABASE MONITOR SCHEDULER - a VAX-based SPMS application that initiates Project Database Monitoring as specified by the Database Monitor Manager. This tool is indirectly-coupled to the Project Database via the DOCGEN Report tool.

MENTOR CASE - a Computer-Aided Software Engineering (CASE) VAX/VMS project management software package. Mentor CASE is integrated with SLCSE having a pop-up window for command entry, and is indirectly-coupled with the SLCSE Project Database via the Mentor Import tool.

MENTOR IMPORT - a tool used to translate and import Mentor CASE output file data to the SLCSE Project Database. Mentor Import uses the DCL Interface Utilities, and has a SLCSE-conformant user interface.

MICROSOFT WORD - a Macintosh-based commercial word processing tool that is a directly-coupled tool of the SPMS Macintosh Toolset.

OBS EDITOR - a VAX-based SPMS tool based on Lotus 1-2-3 macro capabilities. The OBS Editor provides project managers with the capability to develop and analyze Organizational Breakdown Structures (OBSs), and is directly coupled via HLERIF to the SLCSE Project Database.

SLCSE PROJECT MANAGEMENT SYSTEM (SPMS) - provides advanced project management tools supporting the planning, monitoring, assessment, and plan adjustment of software development projects. The SPMS is being implemented using Apple Macintosh-based and VAX-based COTS/DS tools, and is being integrated with the SLCSE user interface and database subsystems. Macintosh COTS tools include MacProject II, Microsoft Word, and Excel. VAX-based COTS tools include Lotus 1-2-3. DS includes Companion Database Interface (CDI) tools, a Database Access Manager, and a Client-Server architecture for SLCSE Project Database access by Macintosh-based tools. VAX-based DS tools include an Organizational Breakdown Structure (OBS) Editor and a Work Breakdown Structure (WBS) Editor built upon Lotus 1-2-3 and the SPMS Client-Server architecture, as well as a Database Monitor Manager and a Database Monitor Scheduler for the monitoring of Project Database events (or the lack thereof). Also included in the SPMS VAX DS is a Problem/Change Report Processor (PCRP) tool that provides more functionality than the original SLCSE PCRP tool (e.g., various report generation capabilities).

QUALITY EVALUATION SYSTEM (QUES) - provides automated quality measurement of Ada and Fortran source code as well as quality information about the entire software life cycle for MCCS development based on 13 quality factors derived from the Rome Laboratory Software Quality Framework, AFSC Pamphlet 800-14 Software Quality Indicators, and the AFSC Pamphlet 800-43 Software Management Indicators. The QUES is being implemented on VAX and SUN workstations with an X-windows user interface and an interface to a SLCSE Project Database via the ERIF.

WBS EDITOR - a VAX-based SPMS tool based on Lotus 1-2-3 macro capabilities. The WBS Editor provides project managers with the capability to develop and analyze Work Breakdown Structures (WBS), and is directly coupled via HLERIF to the SLCSE Project Database.

The following tables summarize the SLCSE toolset that exists, and information about the tools:

TOOL INTEGRATION TYPE CODES

CO - Conformant (User Interface Style)
NC - Non-Conformant (User Interface Style)
DI - Direct (Project Database Coupling)
ID - Indirect(Project Database Coupling)
LC - Loosely-Coupled (Project Database Coupling)
DE - Dependent (Project Database Dependency)
IN - Independent (Project Database Dependency)
IF - Infra-Coupled (Infrastructure Database Coupling)
NF - Non-Infra-Coupled (Infrastructure Database Coupling)
CC - Control-Coupled (SLCSE Control Coupling)
AC - Non-Control-Coupled (SLCSE Control Coupling)

TOOL CATEGORY CODES

GST - General Support Tool
PMT - Project Management Tool
RQT - Requirements Tool
DST - Design Tool
PRT - Prototyping Tool
COT - Coding Tool
TQT - Testing/V&V/QA Tool
CMT - Configuration Management Tool
ENT - Environment Management Tool

DEVELOPMENTAL SOFTWARE
OR
NON-DEVELOPMENTAL SOFTWARE
CODES

DSA - Developmental Software, ATVS
DSO - Developmental Software, Other
DSP - Developmental Software, SPMS
DSQ - Developmental Software, QUES
DSS - Developmental Software, SLCSE
NDS - Non-Developmental Software

HOST PLATFORM CODES

VAX - VAX
MAC - Macintosh Workstation
SUN - SUN Workstation

	TOOLS: INTEGRATION TYPE, CATEGORY, DS/NDS, and HOST PLATFORM
ACS	NC,LC,IN,NF,CC; COT; NDS; VAX
ACS LINKER	CO,LC,IN,NF,CC; COT; NDS; VAX
ACTA	NC,ID,IN,NF,AC; PMT; NDS; MAC
ADA COMPILER	CO,LC,IN,NF,CC; COT; NDS; VAX
ADDFILE	CO,DI,DE,NF,CC; CMT; DSS; VAX
ADL PROCESSOR	CO,LC,IN,NF,CC; DST; NDS; VAX
ALICIA	NC,DI,DE,NF,CC; CMT; NDS; VAX
AMS	NC,LC,IN,NF,CC; TQT; NDS; VAX
AMS ANALYZE	CO,LC,IN,NF,CC; TQT; NDS; VAX
ANALYZER.	CO,LC,IN,NF,CC; ENT; NDS; VAX
ATVS	CO,ID,IN,NF,CC; TQT; DSA; VAX
ATVS LIBRARY MANAGER	CO,LC,IN,NF,CC; TQT; DSA; VAX
ATVS STANDARDS DEFINER	CO,LC,IN,NF,CC; PMT; DSA; VAX
BASELINER.	CO,DI,DE,NF,CC; CMT; DSS; VAX
C	CO,LC,IN,NF,CC; COT; NDS; VAX
CAVS	NC,LC,IN,NF,CC; TQT; NDS; VAX
CLEAR	CO,LC,IN,NF,CC; GST; DSS; VAX
COBOL COMPILER.	CO,LC,IN,NF,CC; COT; NDS; VAX
COPY	CO,LC,IN,NF,CC; GST; NDS; VAX
CPU	NC,LC,IN,NF,CC; GST; NDS; VAX
CREATE	CO,LC,DE,IF,CC; GST; DSS; VAX
DATABASE INSERT	CO,DI,DE,NF,CC; ENT; DSS; VAX
DATABASE ACCESS MANAGER	NC,DI,DE,NF,AC; PMT; DSP; MAC
DATABASE MONITOR MANAGER	CO,DI,DE,IF,CC; PMT; DSP; VAX
DATABASE MONITOR SCHEDULER	NC,ID,DE,NF,AC; PMT; DSP; VAX
DCL INTERFACE UTILITIES	NC,DI,DE,NF,CC; ENT; DSS; VAX
DELETE	CO,LC,DE,IF,CC; GST; DSS; VAX
DESIGN	CO,DI,DE,IF,CC; DST; DSS; VAX
DIRECTORY.	CO,LC,IN,NF,CC; GST; NDS; VAX
DOCGEN 2167A	CO,DI,DE,NF,CC; CMT; DSS; VAX
DOCGEN REPORT	CO,DI,DE,NF,CC; GST; DSS; VAX
DSR	CO,LC,IN,NF,CC; GST; NDS; VAX

EDITER	CO,DI,DE,NF,CC; GST; DSS; VAX
EDT	CO,LC,IN,NF,CC; GST; NDS; VAX
EVE	CO,LC,IN,NF,CC; GST; NDS; VAX
EXCEL	NC,DI,IN,NF,CC; PMT; NDS; MAC
EXIT	CO,LC,DE,IF,CC; GST; DSS; VAX
EXPORT	CO,DI,DE,NF,CC; GST; DSS; VAX
FORTRAN COMPILER	CO,LC,IN,NF,CC; COT; NDS; VAX
GETFILE	CO,DI,DE,NF,CC; CMT; DSS; VAX
IMPORT	CO,DI,DE,NF,CC; GST; DSS; VAX
J73AVS	CO,LC,IN,NF,CC; TQT; NDS; VAX
JOVIAL J73 COMPILER	CO,LC,IN,NF,CC; COT; NDS; VAX
KERMIT	NC,LC,IN,NF,CC; GST; NDS; VAX
LATEX	CO,LC,IN,NF,CC; GST; NDS; VAX
LINK	CO,LC,IN,NF,CC; COT; NDS; VAX
LOGIN	CO,LC,DE,IF,CC; GST; DSS; VAX
LOTUS 1-2-3	NC,LC,IN,NF,CC; PMT; NDS; VAX
LQP	CO,LC,IN,NF,CC; GST; NDS; VAX
LSE	CO,LC,IN,NF,CC; COT; NDS; VAX
MACPROJECT II	NC,DI,IN,NF,AC; PMT; NDS; MAC
MACRO	CO,LC,IN,NF,CC; COT; NDS; VAX
MACSLCSE	NC,ID,IN,NF,AC; PMT; DSS; MAC
MAIL.	NC,LC,IN,NF,CC; GST; NDS; VAX
MENTOR CASE	CO,ID,IN,NF,CC; PMT; NDS; VAX
MENTOR IMPORT.	CO,ID,DE,NF,CC; PMT; DSO; VAX
MENU.	CO,LC,IN,NF,CC; GST; DSS; VAX
MICRO PLANNER PLUS.	NC,ID,IN,NF,AC; PMT; NDS; MAC
MICROSOFT WORD	NC,ID,IN,NF,AC; PMT; NDS; MAC
MICRO IMPORT	CO,ID,DE,NF,CC; PMT; DSS; VAX
MODIFYER	CO,DI,DE,NF,CC; GST; DSS; VAX
MORE.	NC,ID,IN,NF,AC; PMT; NDS; MAC
OBJECTS	CO,LC,IN,NF,CC; GST; DSS; VAX
OBS EDITOR	NC,DI,IN,NF,CC; PMT; DSP; VAX
PCRP	CO,DI,DE,IF,CC; PMT; DSP; VAX
PIGMY	NC,LC,IN,NF,CC; GST; NDS; VAX
PRINT	CO,LC,IN,NF,CC; GST; NDS; VAX
PROCEDURE	CO,LC,IN,NF,CC; GST; NDS; VAX
PROJECT EXCHANGE	NC,ID,IN,NF,AC; PMT; NDS; MAC
PURGE	CO,LC,IN,NF,CC; GST; NDS; VAX
PUTFILE	CO,DI,DE,NF,CC; CMT; DSS; VAX
QUES	NC,DI,IN,NF,AC; TQT; DSQ; VAX,SUN
RENAME	CO,LC,IN,NF,CC; GST; NDS; VAX
REPORTER	CO,DI,DE,IF,CC; GST; DSS; VAX
REQUIREMENTS	CO,DI,DE,IF,CC; RQT; DSS; VAX

RXVP80	NC,LC,IN,NF,CC; TQT; NDS; VAX
RUN	CO,LC,IN,NF,CC; COT; NDS; VAX
SDDL PROCESSOR	CO,LC,IN,NF,CC; DST; NDS; VAX
SDESIGN	NC,LC,IN,NF,CC; PRT; NDS; VAX
SDF CREATE	CO,DI,DE,NF,CC; CMT; DSS; VAX
SDF DELETE	CO,DI,DE,NF,CC; CMT; DSS; VAX
SDL COMPILER	CO,DI,DE,NF,CC; ENT; DSS; VAX
SDL CONVERT	CO,LC,IN,NF,CC; ENT; DSS; VAX
SELECT	CO,LC,IN,NF,CC; GST; DSS; VAX
SEM	CO,DI,DE,IF,CC; ENT; DSS; VAX
SPMS	NC,DI,IN,IF,AC; PMT; DSP; VAX,MAC
SQL INTERPRETER	NC,LC,IN,NF,CC; ENT; NDS; VAX
SQUERY	NC,LC,IN,NF,CC; PRT; NDS; VAX
SREPORT	NC,LC,IN,NF,CC; PRT; NDS; VAX
TEST MANAGER	CO,DI,DE,IF,CC; TQT; DSS; VAX
TEXPRINT	CO,LC,IN,NF,CC; GST; NDS; VAX
TOOLS	CO,LC,IN,NF,CC; GST; DSS; VAX
TYPE	CO,LC,IN,NF,CC; GST; NDS; VAX
VERIFYER	CO,DI,DE,IF,CC; CMT; DSS; VAX
WBS EDITOR	NC,DI,IN,NF,CC; PMT; DSP; VAX
WINNIE	CO,LC,IN,NF,CC; PRT; NDS; VAX

3.4.2.2.2.2 Control Integration (9.3.2)

All degrees of control integration described in the Reference Model exist within the current toolset of SLCSE. However, not every tool is control integrated in the same fashion, as described with the "ICE - Conceptual" factor for this service.

3.4.2.2.2.3 User Interface Integration (9.3.3)

All degrees of user interface integration described in the Reference Model exist within the current toolset of SLCSE. However, not every tool is user interface integrated in the same fashion, as described with the "ICE - Conceptual" factor.

3.4.2.3 Task Management Services (10)

3.4.2.3.1 Task Definition Service (10.1)

Although task definition, as described by the Reference Model, is not completely supported by SLCSE, there is the concept of rule definition which is a rudimentary form of task definition.

3.4.2.3.2 Task Execution Service (10.2)

As rule definition in SLCSE relates to the Task Definition Service (10.1), so does rule enforcement in SLCSE relate to this service, and is described as part of the Event Monitoring Service (10.5) of the Task Management Services.

3.4.2.3.3 Task History Service (10.4)

This service is implemented as part of the Task Management Audit and Accounting Service (10.4).

3.4.2.3.4 Event Monitoring Service (10.5)

As rule definition in SLCSE relates to the Task Definition Service (10.1), so does rule enforcement in SLCSE relate to this service. Also refer to the State Monitoring Service (7.21) of SLCSE, since the monitoring of state changes to the Project Database are related to this service.

3.4.2.3.5 Audit and Accounting Service (10.6)

As rule definition in SLCSE relates to the Task Definition Service (10.1), so does rule data recording in SLCSE relate to this service. In addition, the underlying VAX/VMS Operating System of SLCSE also provides a certain degree of the Audit and Accounting Service.

3.4.2.3.6 Role Management Service (10.7)

The Role Management Service of SLCSE is a part of the two SLCSE Framework Administration Services called Framework Definition/Modification and Environment Definition/Modification, which are services not specifically found in the Reference Model.

3.4.2.3.7 Tool Registration (10.8)

In SLCSE, the Tool Registration Service is provided only in terms of rules, and is implemented as part of the Framework Administration and Configuration Service called the Environment Definition/Modification, which is a service that is not specifically described in the Reference Model.

3.4.2.4 Message Services (11)

3.4.2.4.1 Message Delivery Service (11.1)

SLCSE, in general, does not provide framework-to-framework two-way communication. Other limitations with regard to this service are described with the "ICE - Conceptual" factor for this service.

It is proposed that the following additional kinds of two-way communication be considered for inclusion in the Reference Model description of the Message Delivery Service:

1. service-to-user,
2. tool-to-user,
3. environment-to-environment, and
4. framework-to-environment.

3.4.2.4.2 Tool Registration Service (11.2)

SLCSE does not explicitly provide support for Tool Registration of this kind.

3.4.2.5 Security (13)

3.4.2.5.1 Security Information Class (13.1)

SLCSE provides neither the Authentication Service, Attribute Service, nor the Interdomain Service. The VAX/VMS Operating System may provide some degree of these services, but it is beyond the scope of the current implementation of SLCSE.

It is recommended that each of the services identified in this section of the Reference Model have its own section with descriptions according to applicable dimensions.

3.4.2.5.2 Security Control Services (13.2)

SLCSE does not provide the Secure Association Service, but does provide a minimum of the Authorization Service.

It is recommended that each of the services identified in this section of the Reference Model have its own section with descriptions according to applicable dimensions.

3.4.2.5.3 Security Monitor Services (13.3)

SLCSE does not provide the Security Audit Information Collection Service. However, the VAX/VMS Operating System may provide a certain level of this service, but which is considered beyond the scope of SLCSE.

It is recommended that the Security Audit Information Collection Service in this section of the Reference Model have its own sub-section with descriptions according to applicable dimensions.

3.4.2.5.4 Related Services (13.4)

This service, although listed as a service in [2], is not actually a service, but rather, is listed as the result of an editorial error in [2]. "Related Services" actually refers to the dimension factor called "Relationships Between Services".

3.4.2.6 Framework Administration and Configuration (14)

3.4.2.6.1 Tool Registration (14.1)

SLCSE provides fully implements the "Tool Registration" service, as described in the Reference Model, but also provides other Framework Administration and Configuration Services, as discussed in section 3.4.3, Additional Service Descriptions Recommended, of this report.

3.4.2.7 Integration (15)

It is proposed to merge Reference Model section 9.3, "Tool Integration", with section 15, "Integration".

3.4.2.7.1 Data Integration (15.1)

3.4.2.7.1.1 Object Management as a Data Integration Mechanism (15.1.1)

SLCSE provides Object Management as a Data Integration Mechanism, and also as a *Tool Integration Mechanism.*

3.4.2.7.1.2 Common Data Descriptions (15.1.2)

3.4.2.7.1.3 Tool-to-Tool Data Translators (15.1.3)

SLCSE supports an alternate concept, and that is tool-to-tool data translation via the SLCSE Database Subsystem, i.e., tool-to-OM-to-tool data translation. OM-to-tool data translation provided by SLCSE (and also tool-to-OM data translation) is described in the service description for Tool-to-OM Translators (15.1.4).

3.4.2.7.1.4 Tool-to-OM Translators (15.1.4)

SLCSE also supports OM-to-tool data translation (i.e., two-way data translation).

3.4.2.7.1.5 OM-to-OM Exchange (15.1.5)

The OM-to-OM Exchange service of SLCSE is a concept identical to that of the Tool-to-OM Translators (15.1.4) service of SLCSE. The ability to apply this service depends on how open of an architecture a tool or an OM is in terms of importing and exporting its internal data.

3.4.2.7.1.6 Consistency Management (15.1.6)

This service is implemented as part of the Concurrency Service (7.7) and as part of the Replication/Synchronization Service (7.12). The control of SLCSE over the consistency of data in different data pools is limited, and decreases in the following order when considering:

1. Different data pools used by the same OMS (SLCSE Database Subsystem OMSs):
    a. SLCSE Project Database OMS,
    b. SLCSE Infrastructure Database OMS, and
    c. SLCSE Project Files Hierarchy OMS,

2. Different implementations of the same OMS:
    a. SLCSE Project Database OMS,
    b. SLCSE Infrastructure Database OMS, and
    c. SLCSE Project Files Hierarchy OMS,

3. Different OMSs utilizing different data pools:
    a. SLCSE Project Database OMS,
    b. SLCSE Infrastructure Database OMS, and
    c. SLCSE Project Files Hierarchy OMS,

It is indisputable that SLCSE (or any existing environment framework) does not provide a full Consistency Management Service along these lines.

3.4.3 Additional Service Descriptions Recommended

With respect to the services of the Reference Model, the following sections describe additional service descriptions that were desired during the SLCSE mapping exercise.

3.4.3.1 Tools (9)

Section 9, "Tools", is not organized as the prior "Object Management Services" section was, which makes the Reference Model inconsistent in its presentation style.

More importantly, there is some redundancy in the Reference Model because section 9.3, "Tool Integration" describes various forms of tool-to-tool integration, which would reasonably be merged with section 15, "Integration".

In addition, Section 9 of the Reference Model does not provide a service for describing: (1) the taxonomy used in an environment for tool classification, (2) the tools offered by an environment framework and their classification/functions, and (3) the forms of integration for each of those tools with the framework and/or with each other.

Below, the row on the left describes the current organization of section 9, and the row on the right describes a suggested organization that is more consistent with the previous portion of the Reference Model. Also, the "Tool Integration" section is relocated to section 15, and additional services are suggested to allow for the description of tools that are integral to a particular environment framework.

9 Tools	9 Tools
9.1 Conceptual	9.1 Tool Taxonomy
9.2 Types and Instances	9.1.1 Internal
9.3 Tool Integration	9.1.2 Conceptual
9.3.1 Data Integration	9.1.3 External
9.3.2 Control Integration	9.1.4 Rules
9.3.3 User Interface Integration	9.1.5 Operations
	9.1.6 Data
	9.1.7 Types
	9.1.8 Instances
	9.1.9 Metadata
	9.1.10 Degree of Understanding

9.1.11 Relationships Between Services
9.1.12 Justification For Including Services
9.1.13 Examples
9.2 Classification/Functions of Tools
9.2.1 Internal
9.2.2 Conceptual
9.2.x etc.
9.3 Integration of Tools
9.3.1 Internal
9.3.2 Conceptual
9.3.x etc.

It is proposed to merge Reference Model section 9.3, "Tool Integration", with section 15, "Integration".

3.4.3.2 Framework Administration and Configuration (14)

The following additional Framework Administration and Configuration Services are provided by SLCSE, and are described at the Conceptual level:

1. Framework Definition/Modification Service - This includes establishing/modifying the name for the framework, defining/modifying the set of tools (and the aspects of their integration with the framework) that are available to environment instantiations of the framework (see the Tool Registration (14.1) description), defining/modifying the default role definitions (in terms of the tools available to a user that is assigned a particular role) for an environment instantiation, and a master list of personnel that can be assigned to an environment instantiation.

2. Tool Integration Service - This includes providing mechanisms for the most convenient as possible coupling of tools with the framework, i.e., common user interface "wrappers" for tools, "plug and play" slots for tools, and tool-to-environment and environment-to-tool object management data transfer utilities.

3. Environment Data Repository Creation/Modification Service - This includes the definition/modification of the schema (using the data model of the framework) to be used in an environment for a particular software development project, the creation/modification of the data repository for the environment from that schema, and the loading/unloading of data into and out of the repository that is necessary to support the project. These operations

are typically dependent upon the tools that are to be integrated into the framework and used within the environment.

4. Environment Definition/Modification Service - This includes definition/modification of the characteristics of an environment for a particular software development project. In particular, the characteristics specific to the Object Manager for an environment (i.e., the location of the data repository created for the environment), the personnel to be assigned/deassigned to a project, the roles assigned to those personnel, the modification of the default role definitions for the framework in terms of tool availability to each person assigned to a project, the data access privileges of personnel assigned to a project, adjusting an environment's definition to incorporate the modifications made to the framework, and the definition/modification of pre-invocation and post-execution rules to be applied to the tools in the environment.

5. Environment Deletion Service - This involves the deletion of existing environment instantiations of the framework with options to delete/save information pertaining to the environment to be deleted.

4. CONCLUSION

The mapping exercise revealed that the NIST ISEE Reference Model can provide an excellent means of neutrally describing and comparing CASE environment framework standards and products. The mapping exercise, on the other hand, also revealed a certain lack of maturity in the Reference Model that will, in time, improve as a function of its use on actual ISEE initiatives. As ISEE technology matures, so too will the Reference Model when it is applied to that technology. This mapping exercise was the first of many potential opportunities to validate and improve upon the correctness and usefulness of the Reference Model, and it was successful in doing that.

The mapping exercise was not only beneficial to the development of the NIST Reference Model, but also aided Rome Laboratory in assessing areas where SLCSE can be enhanced (under the FY 91 program entitled, "SLCSE Enhancements and Demonstration Program") to result in a product that offers as rich a set of ISEE services as possible.

5. REFERENCES

[1] A Reference Model for Computer Assisted Software Engineering Frameworks, ECMA TR/55, European Computer Manufacturers Association, December, 1990.

[2] A Reference Model for Computer Assisted Software Engineering Frameworks, NIST Version 1.0, NIST Working Draft, National Institute of Standards and Technology, March 1991.

[3] ANSI/MIL-STD-1815A, Ada Programming Language, January 1983.

[4] Ada Test and Verification System (ATVS), Software User's Manual, General Research Corporation, January 1990.

[5] DOD-STD-2167A, Military Standard for Mission Critical Computer System Software Development, February 1988.

[6] Database Administrator's Manual, ShareBase Corporation, October 1989.

[7] F. LaMonica, Software Life Cycle Support Environment (SLCSE), Product Overview, SLCSE Enhancements & Demonstration Program, Rome Laboratory, July 1991.

[8] Guide to VMS System Security, Digital Equipment Corporation, June 1989.

[9] J. Milligan, Software Life Cycle Support Environment (SLCSE) Project Management System (SPMS): Not Just Another Project Management Tool, Rome Laboratory, July 1991.

[10] J. Milligan, Software Tools Via Logic Programming, RADC-TR-90-261, Rome Laboratory, August 1990.

[11] J. Milligan, Software User's Manual for AnalyzER 3.0, RADC-TR-89-83, Rome Laboratory, July 1989.

[12] SMARTSTAR Administrator's Guide, Signal Technology Incorporated, 1990.

[13] Software Life Cycle Support Environment, Software Detailed Design Document, General Research Corporation, July 1989.

[14] Software Life Cycle Support Environment, Software Programmer's Manual, General Research Corporation, August 1989.

[15] Software Life Cycle Support Environment, Software User's Manual, General Research Corporation, August 1989.

[16] Software Life Cycle Support Environment Project Management System, Software Design Document, General Research Corporation, September 1991.

[17] T. Strelich, Software Life Cycle Support Environment, RADC-TR-89-385, Rome Laboratory, February 1989.

[18] VMS Accounting Utility Manual, Digital Equipment Corporation, April 1988.

[19] VMS Backup Utility Manual, Digital Equipment Corporation, April 1988.

[20] VMS RTL DECtalk (DTK$) Manual, Digital Equipment Corporation, April 1988.

[21] VMS RTL General Purpose (OTS$) Manual, Digital Equipment Corporation, April 1988.

[22] VMS RTL Library (LIB$) Manual, Digital Equipment Corporation, April 1988.

[23] VMS RTL Mathematics (MTH$) Manual, Digital Equipment Corporation, April 1988.

[24] VMS RTL Parallel Processing (PPL$) Manual, Digital Equipment Corporation, April 1988.

[25] VMS RTL Screen Management (SMG$) Manual, Digital Equipment Corporation, April 1988.

[26] VMS RTL String Manipulation (STR$) Manual, Digital Equipment Corporation, April 1988.

[27] VMS User's Guide, Digital Equipment Corporation, June 1989.

☆U.S. GOVERNMENT PRINTING OFFICE: 1993-716-093-60091

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C^3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.