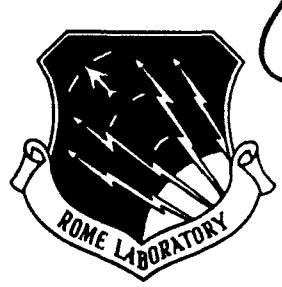
RL-TR-91-166
Final Technical Report
August 1991



TECHNOLOGY ASSESSMENT OF THE SOFTWARE LIFE CYCLE SUPPORT ENVIRONMENT (SLCSE)

The MITRE Corporation

R.W. Baldwin and D.E. Emery

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

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-91-166 has been reviewed and is approved for publication.

APPROVED: Deborah A. Cerino

DEBORAH A. CERINO
Project Engineer

FOR THE COMMANDER: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Technical Director
Command & Control 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 1991	Final Sep 89 - Sep 90

4. TITLE AND SUBTITLE
TECHNOLOGY ASSESSMENT OF THE SOFTWARE LIFE CYCLE SUPPORT ENVIRONMENT (SLCSE)

5. FUNDING NUMBERS
C - F19628-89-C-0001
PE - 63728F
PR - 2527
TA - 02
WU - 24

6. AUTHOR(S)
R. W. Baldwin, D. E. Emery

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The MITRE Corporation
Burlington Road
Bedford MA 01730-0208

8. PERFORMING ORGANIZATION REPORT NUMBER

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

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-91-166

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Deborah A. Cerino/C3CB/(315) 300-2054

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

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report summarizes the technology assessment of the Rome Laboratory Software Life Cycle Suport Environment (SLCSE) performed by the MITRE Corporation. MITRE assesses the SLCSE for applicability to Electronic Systems Division (ESD) System Program Offices (SPOs) and their contractors, and identifies changes required to tailor the SLCSE to ESD requirements. The report also includes a plan for transfer of the SLCSE from the Command Center Evaluation System (CCES) to various SPOs and changes that may be necessary to productize SLCSE, in particular, for use on large scale C3I software development efforts.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Engineering Environment	88
	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	U/L

EXECUTIVE SUMMARY

BACKGROUND

The Software Life Cycle Support Environment (SLCSE) is a computer-based environment framework consisting of software tools integrated with an entity-relationship-attribute database which is intended to support the acquisition, full-scale development and post-deployment support of Mission Critical Computer Software (MCCS).

PURPOSE

This report provides a technology assessment of the Software Life Cycle Support Environment (SLCSE) based on use of the version of SLCSE that was installed at the Electronic Systems Division (ESD) Command Center Evaluation System (CCES) and MITRE's knowledge of System Program Office (SPO) and contractor requirements for software engineering support capabilities and software engineering environment technology. It provides an assessment of the suitability of SLCSE for Air Force acquisition and software development support, recommendations about directions for future SLCSE-related research and development, recommendations for productization of the current capability to support large scale projects, and a recommended approach to the transfer of the productized SLCSE to ESD SPOs and their contractors.

ASSESSMENT

SLCSE is a modern software engineering framework substantially conforming to the requirements expressed in software engineering environment reference models. It provides many of the capabilities necessary to conduct a software acquisition and development effort and has the potential to substantially improve the productivity of software engineers. A centralized database captures all of the data required to automatically generate documentation in conformance with the DOD-STD-2167A software development standard. Tools are provided to populate and analyze the contents of the database for completeness, consistency, and to generate the required documentation.

The current implementation of SLCSE is not without shortcomings. It is currently available only for a VAX/VMS environment. It uses a centralized processor model whereby almost all tools must be resident on the host VAX/VMS cluster to be used, accessible only by VT-100® class terminals, or equivalent. Yet its most (potentially) powerful tool (ALICIA) requires a VAXstation, a graphics workstation with a high resolution bit-mapped display, to use. The SLCSE interface is modeled after a popular personal computer's graphical user interface, but does not provide all of the ease of operation and convenience implied by, and expected of, that interface.

SLCSE has just passed beyond the status of laboratory prototype, having gone into beta test at three USAF sites. Additional testing of SLCSE is necessary to determine its usability by

many concurrent users and to identify as yet undiscovered bugs in the software. The Productization Plan identifies the steps Rome Air Development Center (RADC) should next take to improve both the software and the SLCSE documentation. These include fixing known bugs, improving the user interface, providing the ability to dynamically modify the database schema, developing a Concept of Operations guide and a training manual illustrating how to design and build the database effectively.

The Technology Transition Plan identifies the ways in which RADC can transition SLCSE from RADC to USAF ESD, AFLC, and their associated contractors. A transfer agent should be established to work with RADC and the potential recipients to provide the support required to establish SLCSE as the preferred operating environment. A support organization must be established, trained, and staffed to provide the same kind and degree of technical support expected of commercial tool vendors.

ACKNOWLEDGEMENTS

This project was sponsored by and performed under the direction of the USAF Rome Laboratory (COEE), Griffiss AFB, Rome, N.Y. The project officer was Deborah Cerino.

The authors wish to thank D. Cerino and Frank Lamonica of Rome Laboratory (formerly Rome Air Development Center) for their careful review and suggestions, which greatly improved the organization and content of this report. Thanks are also extended to T.K. Backman, J. C. Fohlin, M. Hazle, R. Hilliard, S. D. Litvintchouk, and S. F. Stanten for their comments and suggestions, and to L. Gaudet and J. E. Lavery for editing this document.

TABLE OF CONTENTS

LIST OF FIGURES

SECTION 1

INTRODUCTION

1.1 BACKGROUND

The Software Life Cycle Support Environment (SLCSE) is a computer-based environment framework consisting of software tools integrated with an entity-relationship-attribute database which is intended to support the acquisition, full-scale development and post-deployment support of Mission Critical Computer Software (MCCS).

SLCSE was developed by General Research Corporation (GRC), Software Productivity Solutions, Inc. (SPS), and Intermetrics, Inc., under contract to the United States Air Force (USAF) Rome Air Development Center (RADC). The project began in 1985 as a research and development effort to create a modern framework that would capture all of the technical design, documentation, and administrative information generated by a software development project in one, comprehensive database. The database could be manipulated to automatically generate all of the appropriate technical and administrative documentation required by Department of Defense (DOD) contracts, as well as to provide automated support for configuration control and management, requirements analysis, and requirements traceability through design, coding, testing and integration.

The final prototype was delivered to RADC in August 1989. Currently, SLCSE has been installed at three USAF Logistics Centers and at Charles Stark Draper Laboratories for beta testing.

1.2 PURPOSE

The MITRE task on Project 5360 provided a technology assessment of the Software Life Cycle Support Environment (SLCSE). This assessment was based on use of the version of SLCSE that was installed at the Electronic Systems Division (ESD) Command Center Evaluation System (CCES) and MITRE's knowledge of System Program Office (SPO) and contractor requirements for software engineering support capabilities and software engineering environment technology. The results are an assessment of the suitability of SLCSE for Air Force acquisition and software development support, recommendations about directions for future SLCSE-related research and development, recommendations for productization of the current capability to support large scale projects, and a recommended approach to the transfer of the productized SLCSE to ESD SPOs and their contractors.

All of the specific tasks in the Technical Objectives and Plans (TO&P) on this project have been completed. Due to the breadth of scope of this investigation, some tasks were given a lesser degree of emphasis to complete this assessment on schedule. In particular, the hands-on assessment, and productization and technology transfer tasks were emphasized.

SECTION 2

STATUS OF TASK

2.1 SLCSE INSTALLATION

The Software Life Cycle Support Environment, version 3.7.2, has been installed at the Air Force Systems Command (AFSC) Electronic Systems Division Command Center Evaluation System, building 1704, Hanscom AFB, MA. Version 3.7.2 was the most current release available from GRC at the time of installation. Access to SLCSE from MITRE was arranged through the use of modems and dial-in telephone lines until the CCES local area network (LAN) could be connected to the main ESD LAN. Originally scheduled for completion by 1 June 1990, the network connection was completed in late July 1990.

Installation of SLCSE required the leasing of a ShareBase Server, model 300, the purchase of an Ethernet® processor board for the Server, and the purchase of a license for SmartStar, a software product that provides an SQL interface to the Server. The SLCSE software provides the environment framework, the user interface, the database schema definition, and specific tools to manipulate the database. SLCSE also provides access to VAX/VMS resident tools, both standard and layered products, that are acquired separately and independently of SLCSE.

The installation of the Server, however, was fraught with problems. Initial attempts to install the Server failed due to a faulty ShareBase cartridge tape. Certain system files could not be read causing installation to abort. A new tape was obtained which corrected that problem; we then discovered that a hardware fault had occurred. Inspection of the interior of the Server revealed a badly damaged cable. This, too, was replaced by ShareBase, but it still did not solve the hardware problem. After two days of repair efforts by the CCES System Manager, and two days of on-site repair efforts by a ShareBase technician, the Server was returned to the ShareBase factory for repairs. Upon return of the Server, after both the processor and hard disk drive had been replaced, and with the Server software installed at the factory, an attempt to boot the Server generated both similar and new error messages compared to the previous installation attempt. Repeated boot attempts and calls to ShareBase, did not reveal the solution to the problem, but ShareBase suggested we try reseating the processor board, as this had worked in the past at other sites. After six tries at reseating the board by two different people, the problem was resolved and the Server was able to boot. The SLCSE software installation was then completed and configured to work with the Server.

Once the installation of the Server was successfully completed, and after the training course was finished, an additional problem developed. The Smartstar software became unable to access the Server. Repeated attempts to reconfigure the software were unsuccessful, and it ultimately was necessary to reinstall the Smartstar software, and reconfigure SLCSE to correct the problem. In contrast, the installation of SLCSE with VAX/Rdb®, the configuration used during the training course was free of problems.

These problems have been reported in detail to highlight the fact that quality control problems and support issues will have a major impact on whether SLCSE is accepted at ESD. Requiring the use of components that are beyond the control of the SLCSE vendor to deliver and maintain will require the site administrator to act as systems integrator, vendor coordinator, and troubleshooter. When a problem occurs that the site administrator cannot handle, there is no single point of contact available to resolve the problem.

It should be noted that, while a sample of one installation is not statistically significant, these problems are not uncommon according to the information provided by the Sharebase technical support staff and GRC's own experiences. This makes continued reliance on the Server and the Smartstar software a risk. For SLCSE to become a widely used product at ESD, such installation and operational problems must be minimized or eliminated. Short-term enhancements to SLCSE should, therefore, be targeted toward reducing or eliminating dependence on both the Sharebase Server and the SmartStar software.

2.2 SLCSE TRAINING

A training course on the concepts and use of SLCSE was held at the CCES for MITRE Bedford personnel involved in the SLCSE assessment task, MITRE Houston personnel involved in an evaluation of SLCSE for the National Aeronautics and Space Administration (NASA), and for interested ESD personnel. General Research Corporation provided the instructor and the course materials, per their contract with MITRE.

The course lasted three and one-half days and covered the major functions of SLCSE, including the user interface, project management, software analysis, programming, configuration management, and SLCSE environment management. Also covered, but in much less detail, were the schema definition language, integration of tools into SLCSE, and the generation of documentation compliant with the data items associated with DOD-STD-2167A, the Defense System Software Development Standard.

The training course was given with SLCSE installed on a MicroVax II, but, due to the installation problems with the Server mentioned above, used VAX/Rdb as the underlying database management system. This proved not to be a problem as the training database was small and we were able to use two MicroVaxes clustered together, with two teams logged into each, to reduce system load and minimize performance concerns.

GRC's other responsibility under this task was to provide telephone support to MITRE on an as-needed basis. GRC was not required to perform any maintenance or make enhancements to SLCSE during this assessment.

2.3 TECHNOLOGY INVESTIGATIONS

Review of professional journals, trade journals, commercial product literature, and evaluations of existing government-sponsored and privately-developed software engineering environments have updated our knowledge of environment technology. Attendance at technology and product seminars and informal interviews with other technically qualified

staff with experience in software research and development have also been used to gain an understanding of the current work being done in this field, and to establish a technology baseline. For a technology baseline we defined what is considered state-of-the-art, and what is considered state-of-the-practice in software engineering environments. We have also looked at government projects that address other aspects related to software development on government contracts.

SECTION 3

ASSESSMENT OF SLCSE

3.1 DEFINITION OF TECHNOLOGY ASSESSMENT

Technology assessment, as we understood it for this task, required examining the technology from two different perspectives. First, it involved taking a broad, overall view of the technology in question, analyzing it, critiquing it, and evaluating its intrinsic value and its general potential contribution to a specific field of science. Second, it required analyzing and evaluating the technology as specific solutions to potential or existing problems.

SLCSE is a software technology that implements engineering framework and project database concepts that have evolved from both basic software engineering research and from practical industrial experience. We evaluated SLCSE to ascertain not only its modernity vis-a-vis the state of the art in software engineering environments, but also, and perhaps more importantly, its total functionality and applicability in providing a usable, practical, environment with which to develop and maintain software.

3.2 EVALUATION APPROACH

The evaluation of integrated software environments can occur at different levels. A traditional approach to an evaluation consists of detailed testing of each of the various components of the environment to determine its level of functionality, completeness, robustness, etc. The Ada Language System (ALS) and the WIS Software Development and Maintenance Environment (SDME) are two examples of environments that were evaluated in this manner by MITRE.

Our approach to evaluating SLCSE, because it is a technology assessment, took a different tack. Because SLCSE was developed as a framework, rather than a monolithic integrated environment, it was evaluated from a higher, conceptual viewpoint to determine if it does (or could in the future) provide the desired capabilities.

We drew on the latest work in this field to provide general background guidance and objective criteria with which to assess SLCSE. Specifically, the Ada Programming Support Environment (APSE) E&V Guidebook and APSE E&V Reference Manual, developed for the Ada Joint Program Office, and the CASEE (Computer Aided Software Engineering Environment) reference model work being performed at Hewlett Packard Laboratories, were used to identify the specific requirements and criteria against which modern software environments and frameworks are being evaluated.

The APSE E&V guidelines have been developed specifically to assist government and industry in gaining an overall understanding of APSE assessment and in evaluating Ada specific software environments. The E&V guidebook contains information to assess APSEs from a "whole APSE" perspective as well as from an individual components perspective.

Additionally, assistance in assessing certain non-technical issues, important to the successful use of an APSE, such as cost, maturity, and licensing issues are covered.

The CASEE Reference Model provides a definition of services that ought to be available in a modern software engineering framework. While a CASEE reference model is not itself a standard, it nevertheless provides a useful model with which to assess SLCSE against current and future trends in CASE tools and environments.

The assessment of SLCSE, however, was still primarily oriented toward evaluating its suitability for use by ESD SPOs and contractors. We also based our assessment of SLCSE on the hands-on experience obtained through the training course and by running an experimental project. This project was representative of software development efforts undertaken by ESD contractors. The project went through the requirements and design phases of the software life cycle, as defined in DOD-STD-2167A, developing the requisite information appropriate to each phase, and storing it in the SLCSE database. In this manner, we determined if the SLCSE database schema was logically complete, and could support software development and management as generally performed by an ESD contractor. We were then able to make recommendations regarding improvements in data collection and reporting capabilities.

3.3 FACTORS TO BE CONSIDERED

For SLCSE to gain acceptance and become a widely used tool at ESD and ESD contractor sites, it must provide the necessary functionality, as discussed above. It must meet certain other criteria, as well. The factors that we took into consideration during the assessment of SLCSE were the following:

Capabilities Provided - SLCSE must provide the necessary tools to perform project management, requirements analysis, software design, coding, testing, integration, document generation, configuration management, etc., in the manner that is consistent with the general way that contractors and DOD SPOs carry out their work. New tools, from third party sources that are in widespread use, must be able to be easily integrated into SLCSE and be able to populate the database as if they had been specifically designed to do so.

Ease of Learning - SLCSE must be easy to learn and use. While this seems like an obvious point, it cannot be over-stressed. Any large, complex tool suite, especially one with a complex underlying database, that is expected to be used by both technical and non-technical staff, must be simple to learn and intuitive to use.

Productivity Improvements - For SLCSE to be adopted, it must provide a measurable increase in the productivity of the staff who use it. For managers, that implies being able to more closely (but unobtrusively) monitor the status of ongoing work, and identify and report problems earlier than would otherwise be possible; for technical staff, it implies being able to analyze, design, review, implement and test software systems, either faster or with higher quality, or (preferably) both. These

gains must be sufficient to justify the overhead expenses involved in installing and maintaining SLCSE and associated hardware and software, and training project personnel in its use.

Robustness/Stability - Of prime importance to project personnel, at both ESD and contractor sites, is the robustness and stability of SLCSE. For SLCSE to be used on real projects, it must be available for use on an almost nonstop basis, subject only to the availability of its underlying hardware base, with only short scheduled downtimes interrupting service. Unscheduled downtime, e.g., software crashes, must be kept to a minimum. SLCSE must not permit internal system errors or user errors to cause it to abruptly crash, perhaps losing much work, but provide graceful degradation and recovery. Maintaining consistency of the database is particularly important.

Supportability - Its robustness notwithstanding, SLCSE must be a supported product. When the inevitable problems do arise, adequate technical support must be available to the user community. Serious bugs must be fixed and new versions released on a regular basis, and workarounds developed and provided in the interim. New capabilities and enhancements to existing functions must also be provided on a regular basis.

Capacity - SLCSE must be capable of supporting large as well as small projects. By large, we mean projects with upward of 300-500 people at any given time requiring concurrent access to SLCSE. Short response times for interactive work and reasonable turnaround times for batch-oriented activities must be supported. While the maximum capacity will also be dependent on the hardware base used, there should be no inherent limiting factors built into SLCSE.

3.4 SLCSE ASSESSMENT

3.4.1 Conformance to the CASEE Reference Model

The CASEE Reference Model is a conceptual framework describing and providing the high level requirements necessary or desirable in a Computer-Aided Software Engineering Environment (CASEE). Viewing a CASEE as a black box, figure 1, taken from the CASEE Reference Model, illustrates the types of information and resources input to a CASEE, and the type of information that is output.

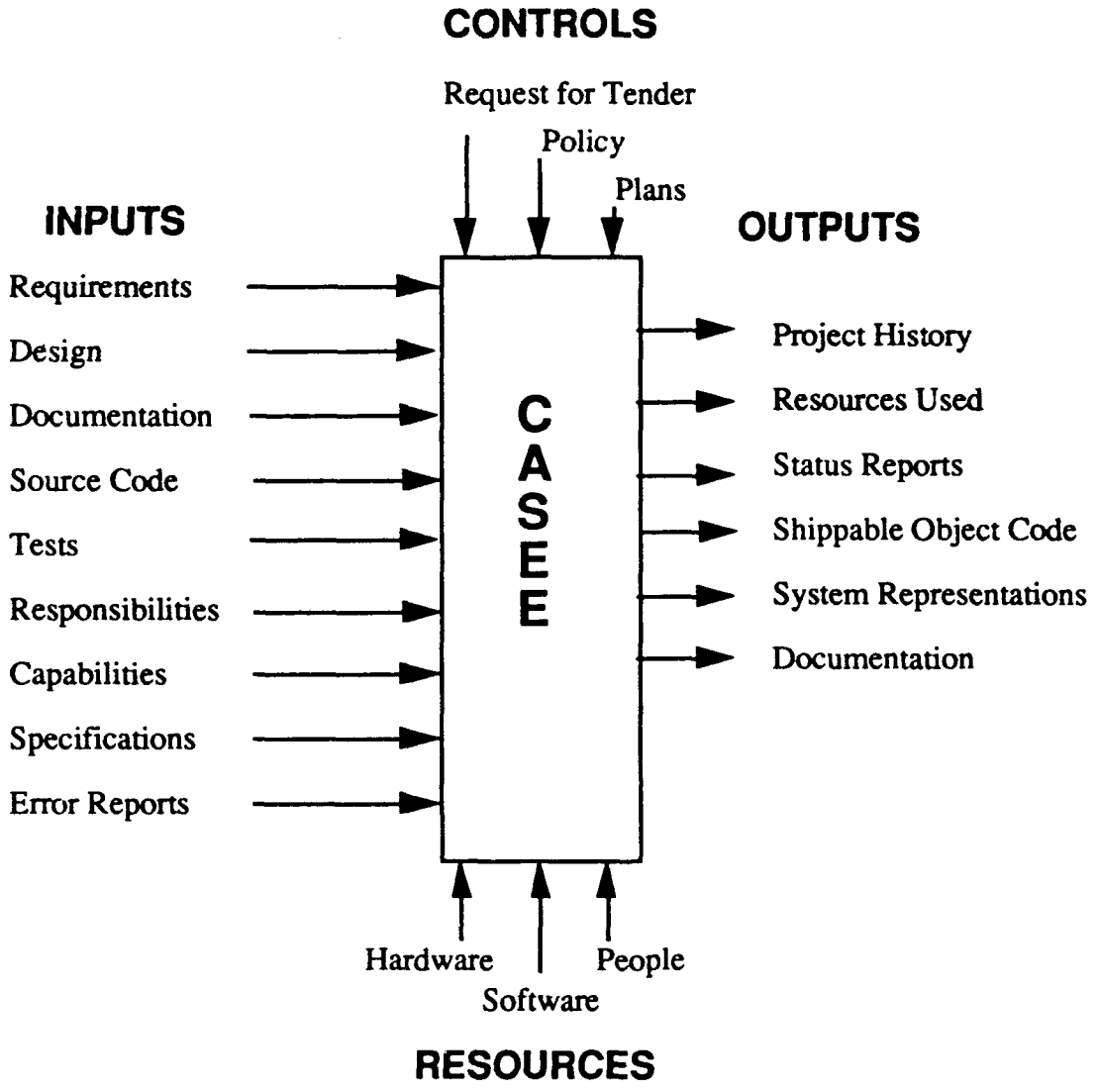


Figure 1. Example Inputs, Outputs, Controls, and Resources of a CASEE [Earl, A.]

From this perspective, we conclude that SLCSE conforms well, in principle, to this view of a CASEE. As we have ascertained by reviewing the SLCSE documentation, SLCSE accepts as inputs the types of information specified, and can produce outputs of the types listed. This view, however, does not deal with the internals or structure, of a CASEE.

The overall structure of the CASEE Reference Model is shown in figure 2. As indicated, a CASEE is described in terms of the services provided. We now evaluate the degree to which SLCSE provides the specified services.

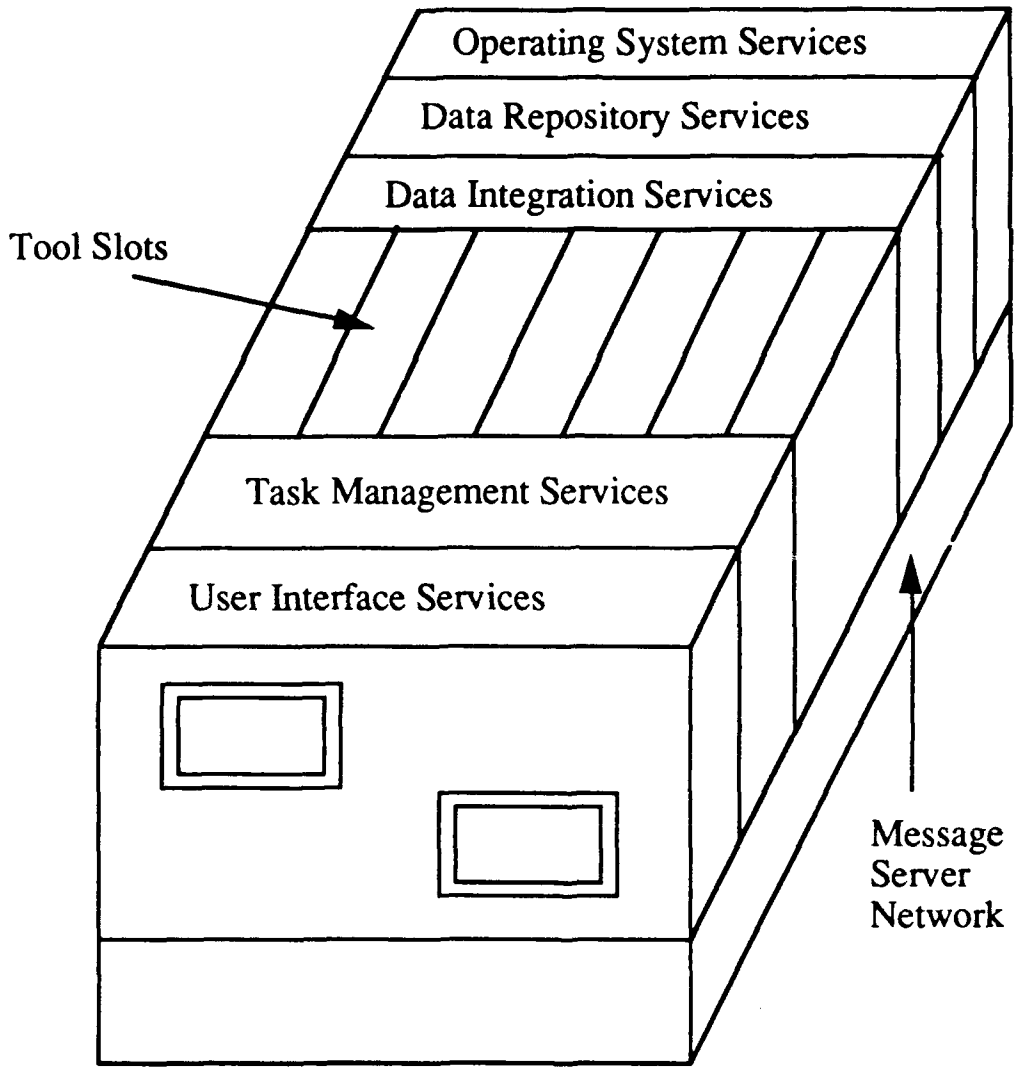


Figure 2. Overall Reference Model Structure [Earl, A.]

3.4.1.1 Operating System Services

The CASEE reference model provides no detail about Operating System Services, as they are presumably well-understood, except to make a distinction between operating system kernel services and networking services. As SLCSE uses and indirectly makes available the services of the underlying VMS operating system for both kernel and networking services,

SLCSE is consistent with the CASEE model, to the extent that VMS and DECnet provide the appropriate services. SLCSE currently is not operating system independent, being available only on VAX/VMS.

3.4.1.2 Data Repository Services

The management and maintenance of data entities and the relationships among them is the general purpose of the data repository. Data repository services are comprised of the following services:

- Data storage service - SLCSE provides the capabilities necessary to be consistent with the reference model. All primary services relating to entities are supported: create, read, update and delete. A complete data model is provided along with the capability to modify that model by changing the type of an entity or adding and deleting attributes.

- Relationship service - SLCSE provides the capabilities necessary to be consistent with the reference model. Relationships (or links) between entities are supported. Relationships are typed and have attributes and cardinality. In addition, the semantics are defined such that a relationship cannot exist unless both the domain and range entities exist. Currently, a relationship is not automatically deleted when both domain and range entities are deleted, but this will be corrected in a future release.

- Version service - SLCSE provides a versioning capability that is consistent with the reference model. Creation and management of different versions of entities is supported, as is the ability to baseline a set of entities per DOD requirements.

- Name service - SLCSE supports the use of "surrogates," i.e., unique system-generated identifiers that are never reused. This eliminates the possibility that inconsistencies will occur in the identification of entities and relationships. User naming of entities is supported as well to provide a more readable representation.

- Configuration service - SLCSE supports the concept of configuration service. Composite entities are supported wherein a composite entity is the union of a set of individual entities. An entity can be treated as a single item or as a structured collection of entities. Also supported is the ability to define a configuration for baselining purposes. This service is well integrated with the version service.

- Archive service - The reference model states there has to be a service which allows on-line information to be transferred to an off-line form and yet provide visibility to the user for retrieval of archived data. In a large project, it may not be feasible to keep all development data on-line for long periods of time. SLCSE does not provide such an archival service.

- Data transaction service - SLCSE supports this service to the extent that the underlying database management system provides a transaction processing capability, with commit and rollback operations defined in the traditional sense.

- Concurrency service - SLCSE supports the ability to allow users to lock and unlock entities to avoid the lost update scenario whereby the second update to an entity unwittingly overwrites the first update. There is, however, an outstanding problem trouble report at GRC concerning this which should be addressed in any future updates.

- Security service - SLCSE supports access controls at the tool and entity level that conform to the type suggested by the reference model. In particular, access controls can be set based on who the user is and what role he is playing. We did not find any information to suggest that violations are logged and notifications issued (perhaps to a system or program manager) as recommended in the reference model. Other security issues, such as covert channels, performance, and operating systems security are only mentioned in the reference model as topics for discussion at a future date. SLCSE could not currently support classified software development in compartmented mode because it lacks the capability to tag objects with a security level.

- Backup service - SLCSE supports a database load and unload capability that, combined with standard VMS backup and restore procedures, provides services consistent with the reference model. Backup of the SLCSE database is limited to full backup and restore. It would be advantageous to have an incremental backup capability.

3.4.1.3 Data Integration Services

Support should be provided for metadata and services for both metadata and data. Data is the specific information, or values, stored in the database. Metadata is data about data, and enables the user to change the structure of the database by modifying existing data types and relationships and adding new data types and relationships. The proper utilization of these services allows for the "plug compatibility" of tools as it relates to data. Data integration services are comprised of the following services:

- Query service - SLCSE provides limited support for users to define their own data interrogation commands. When in the EditER, a "query-by-example" function, where the example is user-defined, can be invoked to search the database for entities meeting a specific set of conditions. Support for querying metadata is not provided to the end user.

- Metadata - SLCSE supports the definition and use of metadata. New entity types, new relationships, and new attributes can be defined and added to the database schema. Existing definitions of entity types and relationship types can be changed by modifying the Metaschema. A limiting factor, however, is the restriction on the types of data that can be represented directly by the underlying relational database management system. Much data that is required to be captured in a software development effort is in the form of composite record types and various forms of graphic diagrams. Direct support for these would be desirable.

- State monitoring - SLCSE does not provide this service yet. Also called "triggering," this service enables the definition and specification of database states, and actions to be taken should these states occur. There is no capability for defining states of data or metadata, or for sending notification that such a state has been achieved. Currently, only a limited form of event monitoring of tools is provided.

- Sub-environment service - SLCSE provides this service by enabling the SLCSE Environment Manager to set user access rights to the database subschemas and environment tools. Users will then see just the data in their own workspaces and in assigned subschemas and have access to only the tools required for that particular role.

3.4.1.4 Tools

The CASEE Reference Model defines a tool as "an artifact which automates some function or activity, performing it on behalf of the user who operates within his task." Tools are envisioned as "plugging-in" to the environment between the task management and data integration services. A tool is permitted to store (almost) no data within its own code or, at the other extreme, store all of its data within its own private database. Tools may be added to an environment by encapsulating them with software to make them work in that environment without having to modify the tool. At a minimum, an environment should identify which tools exist and describe the purpose and facilities of each tool.

SLCSE provides full support for the description and integration of tools provided in the reference model. Tools are supplied with SLCSE which automate many functions that previously would have been performed manually. An encapsulation capability is provided, through the Entity-Relationship Interface (ERIF) package and the windowing tools, to integrate new tools into SLCSE and provide a conforming interface as well as enable the tools to utilize the database appropriately for that tool's purpose. A description of each tool's function is available in the documentation for SLCSE and on-line in the help facility for some of the tools. While the reference model does not categorize tools, e.g., "management" tools, or "requirements analysis" tools, tools in SLCSE are categorized according to the phases/activities of the life cycle they support or as general purpose. For convenience, a complete list of tools available in SLCSE is included in the appendix.

3.4.1.5 Task Management Services

Task (or activity) management is defined in the reference model as services that allow the CASEE to be task-oriented, i.e., to support process modelling. A layer of abstraction is provided which allows the user to deal with tasks as opposed to accomplishing a job by a series of invocations on individual tools. The following services comprise task management:

- Task Definition Service - SLCSE does not provide a task definition capability as defined by the reference model. All SLCSE activities are carried out by individual tools. There is currently no way of defining and providing a higher level abstract activity.

- Task Execution Service - SLCSE provides limited support for this service to the extent that executing tools can be considered primitive tasks executing in the SLCSE environment. Multiple tools cannot be executed concurrently in an interactive mode by a single SLCSE user.

- Task Transaction Service - This enables the user to execute a task which consists of many sub-tasks, each of which does useful work that is saved even though some sub-tasks fail. SLCSE does not provide this service. Individual tools may provide a form of this service in that a tool might provide a sequence of primitive operations, some of which may fail, but the results of those which are successful are not discarded.

- Event Monitoring Service - SLCSE provides a limited form of this service. Rules can be defined which act as triggers when specific tools are invoked (pre-condition) or complete operations (post-condition). Other types of events are not supported.

- Audit and Accounting Service - SLCSE does not directly provide this service. However, VMS-based auditing and accounting tools external to SLCSE are commonly used to provide cost and performance information. These tools could easily be made accessible in SLCSE should a project manager require it.

- Task History Service - SLCSE does not support this service since task definition is not supported.

- Role Management Service - SLCSE supports this service by defining user roles and requiring SLCSE users to play a specific role based on the work to be performed, e.g., software analysis, project management, configuration management. This service could be expanded in SLCSE to provide additional information such as what roles each SLCSE user is capable of playing (vs. what roles they are currently assigned), relationships between roles, e.g., "all software analysts can also carry out the programming role."

3.4.1.6 User Interface Services

The CASEE Reference Model notes that the subject of user interfaces is a complex issue which is more general than integration frameworks. It allows that a consistent user interface may be adopted for a complete framework. A layered reference model for user interfaces, based on the X Window System®, has been taken from the Open Software Foundation. From this model it can be inferred that an important characteristic of a user interface is the separation of the presentation of information from the application level functionality that generates the information.

SLCSE supports this service by implementing a user interface that provides a consistent "look and feel" when accessing file system objects and invoking tools. It does this by providing a separate window management package which implements a Macintosh®-like interface on a character-based terminal. Further, it standardizes on a single editing function

(EditER), which all tools that manipulate the database use. Creation, modification and deletion of database entities and relationships have the same "look and feel" regardless of the tool in use.

This support is limited in that the user interface only provides a shell around an installed tool. The SLCSE windowing interface expects a VT-100® compatible text stream as input. Tools to be integrated that will be accessed from the SLCSE menu and run from the standard SLCSE terminal must provide a VT-100 interface. The form of the interface is not separated from the function, as recommended in the reference model. This will become a serious limitation as more tools are becoming available that have graphic interfaces and generate graphic displays.

3.4.1.7 Message Server Network Services

The CASEE Reference Model is incomplete regarding these services. However, tool registration and standard communication services such as inter-tool and inter-service communications are identified as the primary services to be provided.

SLCSE supports the tool registration service in that a well-defined set of procedures and interfaces exist to incorporate new tools into the SLCSE environment.

SLCSE does not directly provide the communications services described in the reference model. Tool-to-tool communications is indirectly supported by using the database to store information common to many tools. Service-to-service and tool-to-service communications are not supported as SLCSE does not provide services in the manner assumed by the reference model. However, SLCSE uses VMS extensively to provide the underlying communications capabilities expressed in the reference model. Therefore, SLCSE substantially conforms to the reference model in this area.

3.4.1.8 Distribution

Support for distributed software development is firmly established as a requirement in the reference model. All services in the CASEE should be "available over a distributed collection of heterogeneous processors and storage devices." SLCSE does not fully conform to this view of software development. The current SLCSE model of computing is based on a central processing system being accessed by timesharing users with "dumb" terminals. All tools, with the exception of two that are hosted on a Macintosh, run in the VAX/VMS environment. Distributed development is supported only to the extent that multiple VAX/VMS systems, either networked or clustered, can be used concurrently.

3.4.1.9 Target Service

Target service provides a means of communication and control between the development environment and a target environment where the software under development will be executed. SLCSE does not support this service. It is currently a self-hosted and self-targeted environment. New tools which would support cross-compiling, downloading of software to a target environment, and execution control can be integrated into SLCSE. These tools would

then have to rely on the communication services of VMS to provide the linkage between the host and target. As an example, RADC has integrated the J73 toolset which supports the MIL-STD-1750A target processor.

3.4.1.10 Users of a CASEE

The reference model defines several roles that are associated with CASE development, management, maintenance, and customization. They are:

- Project managers - supervise definers, environment builders, developers, etc.
- Definers - define customized environments supporting a specific software engineering methodology.
- IPSE (or CASEE) managers - supervise tool writers and project managers employed by manufacturers.
- Manufacturers - vendors of CASE tools and utilities.
- Tool writers - develop CASE tools and utilities.
- Developers - build target systems.
- Environment builders - produce customized environments.
- Customers - provide the software requirements for the target systems.

It is clear from the brief descriptions given that the user role definition capability in SLCSE already supports this service and that specific roles mentioned above that may not be specifically defined could easily be added.

3.4.2 Conformance to Other Factors

In this section we provide an assessment of SLCSE, evaluating its overall capabilities, ease of use, potential productivity improvements, robustness, and capacity.

3.4.2.1 Capabilities of SLCSE

The current implementation of SLCSE provides database capabilities and tools, from various sources, that adequately cover the software development life cycle. The entity-relationship-attribute database captures the data required in documentation conforming to the current DOD software development standard, DOD-STD-2167A. The database input and analysis tools provide the necessary functionality to create, update, and delete entities and relationships. The reports generated by the database analysis and verification tools enable a user to list the entities and the current values of their attributes in the database and the relationships between them. Potentially missing relationships can also be reported providing

a capability for checking for completeness. Project documents, with formats conforming to DOD-STD-2167A data item descriptions, are easily created. Customized documents can also be created for project specific purposes.

Other capabilities provided by SLCSE are specific to tools installed within or accessible through SLCSE and will not be discussed individually.

3.4.2.2 Ease of Learning SLCSE

Overall, we found learning the mechanical aspects of using SLCSE to be quite easy. The user interface is easy to understand, and the keypad mappings, used to invoke various functions within a tool, are consistent across all tools developed specifically for SLCSE. Accessing tools and user-created objects is a simple procedure due to the pull-down menus invoked when choosing an item in the menu bar. Scrolling through a long list of tools can be tedious, however, so the capability to re-order the items on the tools list, to place the most often accessed tools first, is an advantage, as well as being able to scroll to the tool by typing the first few characters of the tool name.

Tools created for SLCSE that provide access to the database were easy to learn to use. Those tools, e.g., ModifyER and VerifyER, that provide a graphical view of the database subschemas utilize the same EditER function to create, update, and delete entities as those tools, e.g., Requirements and Design, whose interfaces are completely textually based. It is not possible however, to enter the attribute values of a relationship created between two entities at the time the relationship is created, thus requiring a two-step process which complicates the learning process. Time-consuming operations, e.g., database updates and document creation, are performed in batch mode, with a textual message displayed when the operation is complete.

There are, however, a number of areas in which SLCSE could be improved. In some cases, system responses are misleading, ambiguous or nonexistent; in other cases, presentation of information is confusing, awkward, or difficult to read. The following examples will illustrate what we mean:

- When using tools that provide a graphical view of a subschema, such as ModifyER, entities and relationships are chosen by using the arrow keys to navigate around the subschema graph. Due to the way the windowing system works, however, the direction traversed is not always along the relationship links. In many instances, pressing an arrow key caused the cursor to go in a direction orthogonal or opposite to what was expected.

- After creating relationships between entities, the user exits that function by pressing keypad-0, invoking the Done command. There is no feedback at this point to indicate how or whether the newly created relationships will be saved (they are).

- When an entity is highlighted by the cursor in a menu list, it is ambiguous to the user whether or not it has been selected.

- When a list of entities are highlighted in a menu showing the potential values the range entity of a relationship may be assigned, all entities that have current relationships with the domain entity are highlighted, not just the entities that are related by the specific relationship chosen.

- When a user tries to update an entity to which he does not have update privileges, the resulting messages indicate that the update was successful, even though the update did not occur.

- Long names in menu windows and in the Current Objects field often get obscured and become difficult to read. Sometimes this prevented us from being able to distinguish between two similarly named entities because the latter part of the name was cut off. With the windowing system providing the capability to keep open multiple, overlapping windows, it would be advantageous to be able to move the windows around on the screen so that one does not obscure the other.

- When multiple entities or relationships are marked for deletion, a save request will perform the first deletion. The next deletion terminates the save operation with the following message: "Database synchronization error; save stopped." As many save commands must be issued as there were entities marked for deletion to effect the complete operation, but after each, the above error message is generated.

- After entering a set of entities into the database, and creating the "CSCI_Capability Utilizes Required_Data_Element " relationship, keypad-0 is pressed to return to the previous menu. At this point, the screen shown in figure 3 is displayed. The message at the bottom states "press Gold PF3 to insert rel. attributes ..." But pressing Gold PF3 brings up a template to edit the attributes of the relationships just created. The FQT reference is misleading.

- When editing attributes of a relationship, a domain entity is first chosen. Then the relationship type is chosen. After editing the appropriate relationships for the specified entity, continued use of the forward key (keypad-4) retrieves all relationships for all entities. While this may be desirable for small databases, it could pose a problem in a large database when scrolling through a menu with many relationships listed, if one only wants to deal with a small, selected subset.

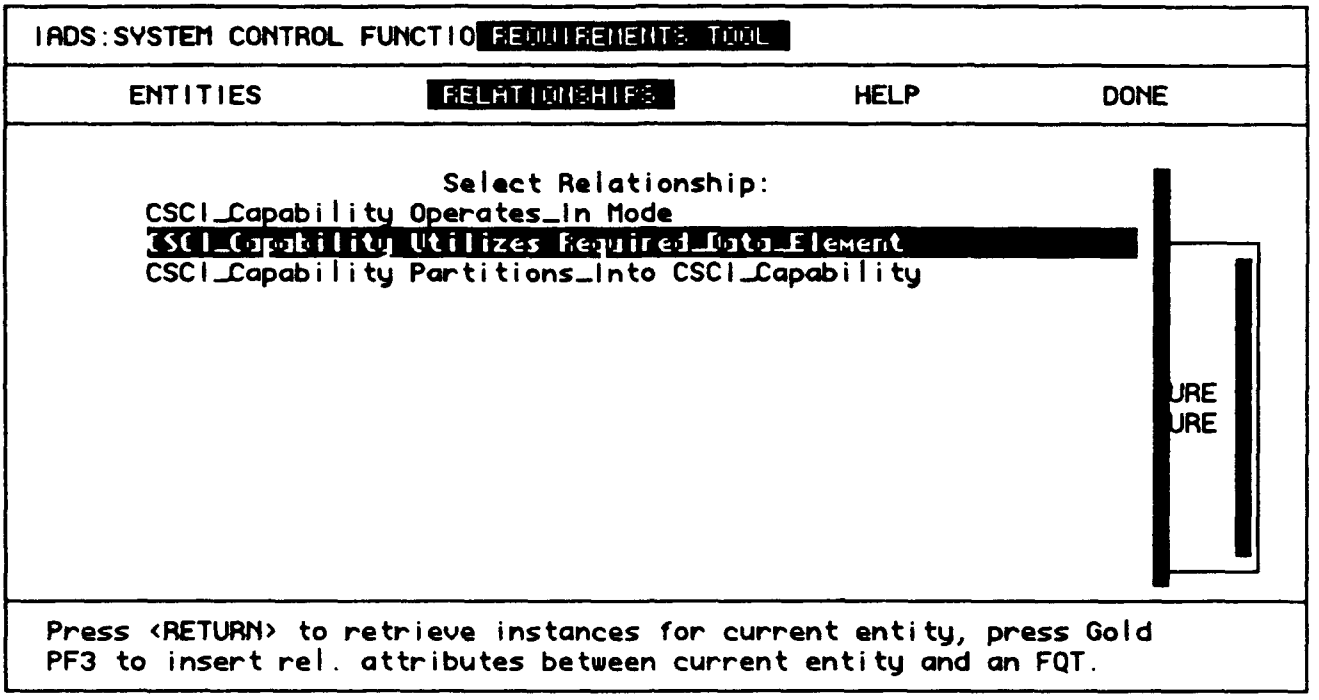


Figure 3. A Misleading Help Message

The most difficult aspect we discovered in using SLCSE was determining the specific type of data that should be entered in the attributes of both entities and relationships. No help is given, either on-line or in the SLCSE documentation, to explain the semantics of attributes that are listed in an EditER template, but are not part of the entity schema definition. For example, the type definition for CSCI_Capability is:

```
entity type CSCI_Capability is
    PUID : string(20);
    Purpose : text;
    Performance : text;
    Control_Flow_Diagram : text;
    Data_Flow_Diagram : text;
    end entity;
```

On the entry form for defining CSCI_Capabilities, however, "Access Name," "Desc Name," "Version," and "Key" are also displayed, as well as fields for storing information about its creation. Little help is available to explain why these fields exist or how to make best use of

them other than a short example illustrating the "Query by Example" function in the EditER User's Manual. A description and paradigm for use of these fields and a naming/numbering convention to follow on an actual project would be of great help.

Similarly, to understand exactly what data should be input in specific (entity or relationship) attributes, we had to continuously refer to the Document Generation Language (DGL) code for a given document, e.g., the SRS for software requirements data, to determine where the data in that attribute would appear, i.e., in what section, in a document produced by DOCGEN_2167A. Thus, a complete understanding of the defined schema; the access, descriptive and search keys; and the requirements of DOD-STD-2167A are necessary to correctly create a project database.

3.4.2.3 Productivity Improvements Gained Using SLCSE

We believe, based on this assessment and on our prior development experience in industry, that SLCSE has the potential to improve both the productivity of users and the quality of their work. Automating the generation of project documentation from its constituent parts in the database should reduce the amount of manual labor associated with producing the documentation. Performing traceability studies should reduce the number of inconsistencies and gaps often found in delivered documentation ensuring, for example, that no requirements are left uncovered by the design. Impact analyses could reduce the time necessary to determine the scope of changes required by a proposed modification. The latter two capabilities would be particularly useful by SPOs.

Because our hands-on evaluation was not intended to be a detailed beta test, and was hampered by limited access to SLCSE, we are unable to positively confirm that the productivity gains specified above would actually be realized by a project. In fact, our experience leads us to believe that productivity would initially be lower due to the education and training necessary to properly design and construct a project database and use the tools properly. Once that learning curve has been overcome, we believe gains will be realized. It is unknown at this time whether the gains that can be realized will be sufficient to justify the expense involved establishing a SLCSE-based development environment.

3.4.2.4 Robustness and Stability of SLCSE

Overall, we found SLCSE to be stable. It only crashed catastrophically once, and then it was as a result of a failure of the underlying hardware. We define a "catastrophic crash" as one where SLCSE becomes completely inaccessible and causes or results in a large loss of work.

The MicroVAX crashed unexpectedly prior to a database update being completed. We were never able to gain multiple user access to SLCSE so we are unable to determine how stability is affected, if at all, by multiple, concurrent use of the database. Concurrency control and deadlock issues should be tested further in a more detailed beta test of SLCSE. There were a number of instances when the Sharebase Server became inaccessible to SLCSE due to network or Smartstar problems. Network problems are beyond the scope of SLCSE to resolve, but problems with Smartstar should be addressed. Figures 4 and 5 are a pair of screen dumps which illustrates a failure to access the Server due to a problem with either the network or Smartstar (we were not able to determine which was at fault).

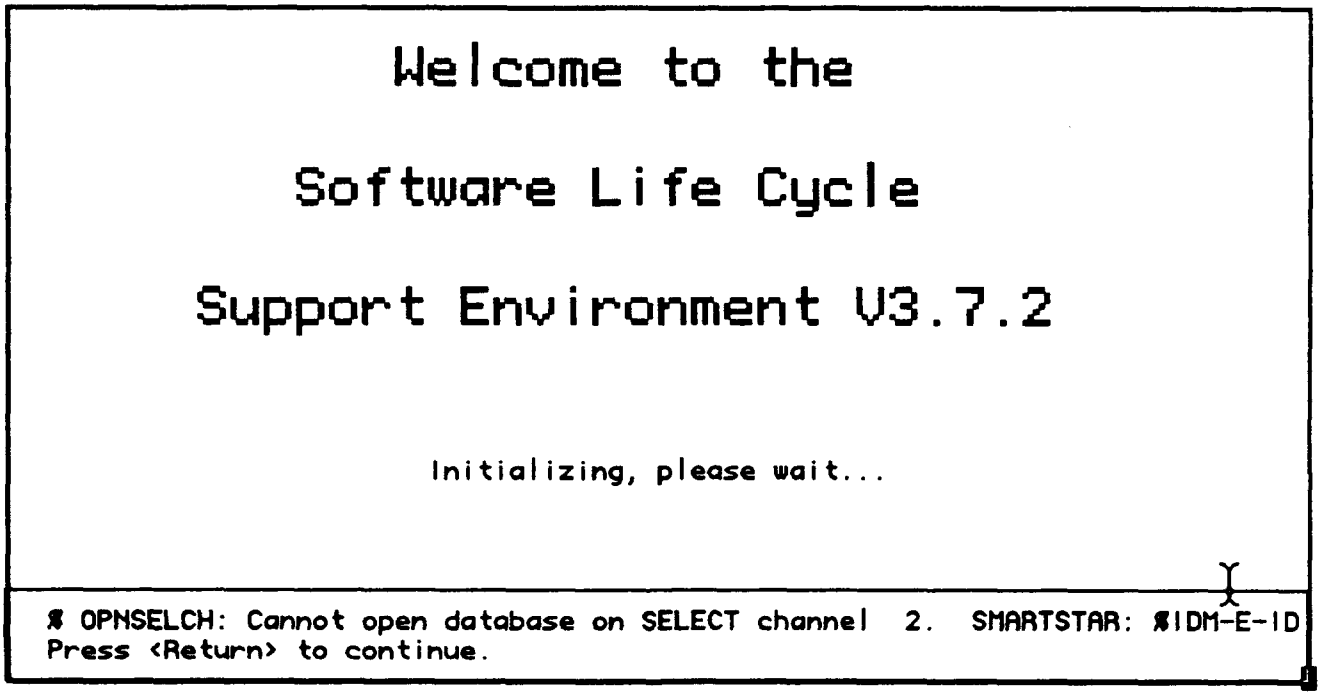


Figure 4. SLCSE Cannot Access the ShareBase Server

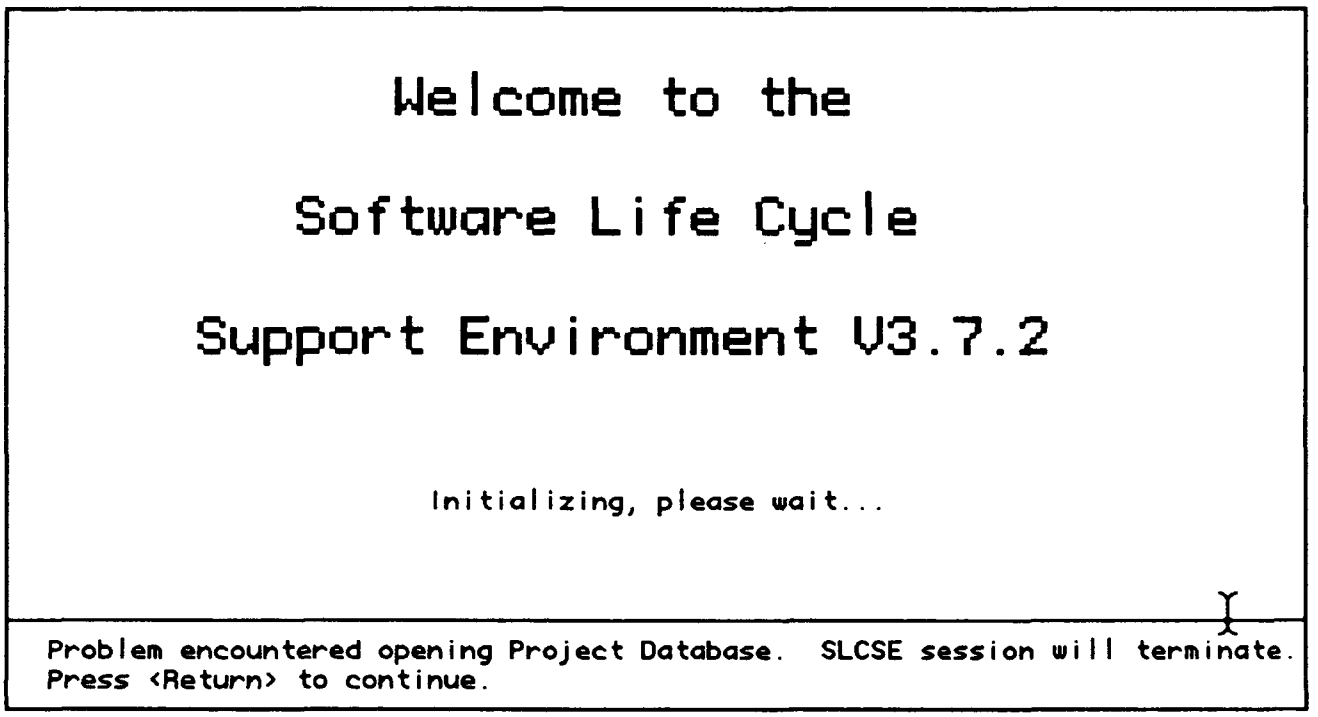


Figure 5. SLCSE Terminates Abnormally

This problem occurred repeatedly during our evaluation of SLCSE, severely hampering our ability to build the Icelandic Air Defense System (IADS) project database (see section 3.4.3).

In figure 6, the error message clearly indicates a problem with Smartstar. An incorrect SQL statement is the apparent cause, yet there should be no reason for this to occur at SLCSE start-up time if no system modifications have been made.

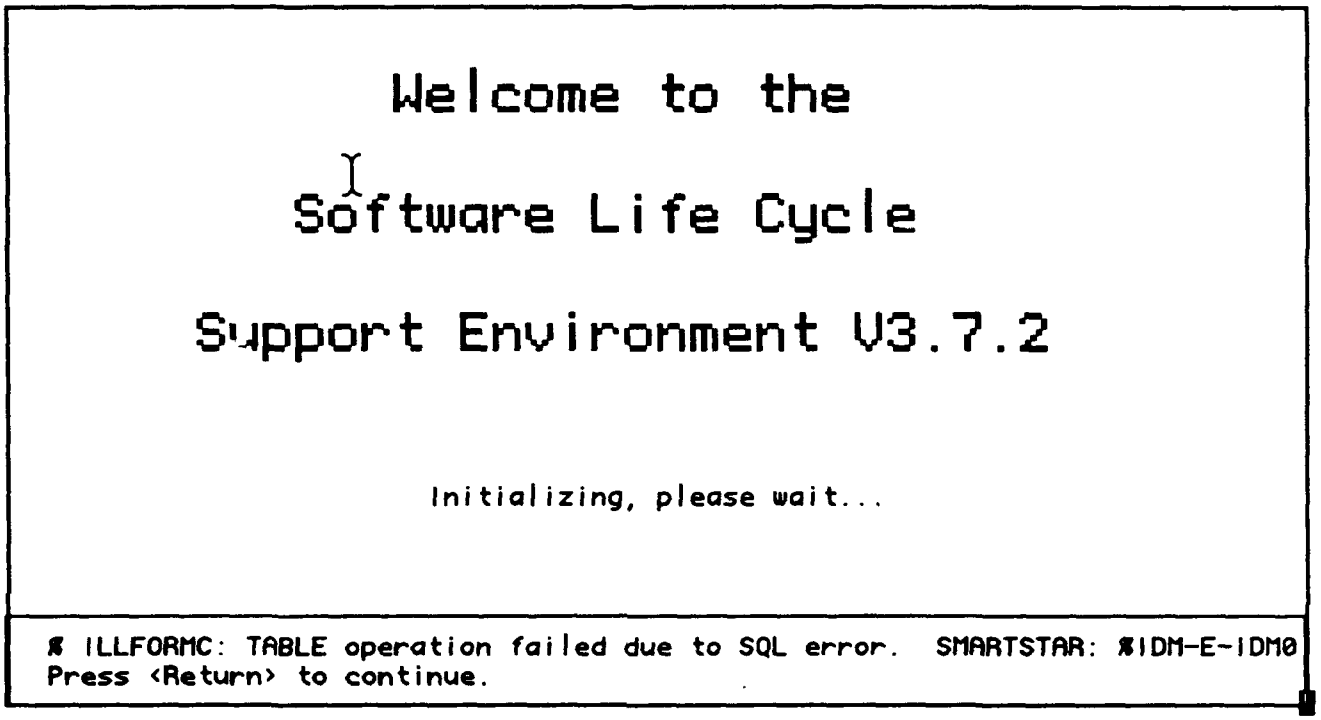


Figure 6. SLCSE Fails to Access the Database Due to a Smartstar Error

There were also a few instances when (sometimes) unknown errors caused a tool to abort, returning the user to the main SLCSE window or, in the extreme case, all the way back to a Digital Command Language (DCL) prompt. Figures 7 and 8 illustrate the former and figure 9 illustrates the latter problems.

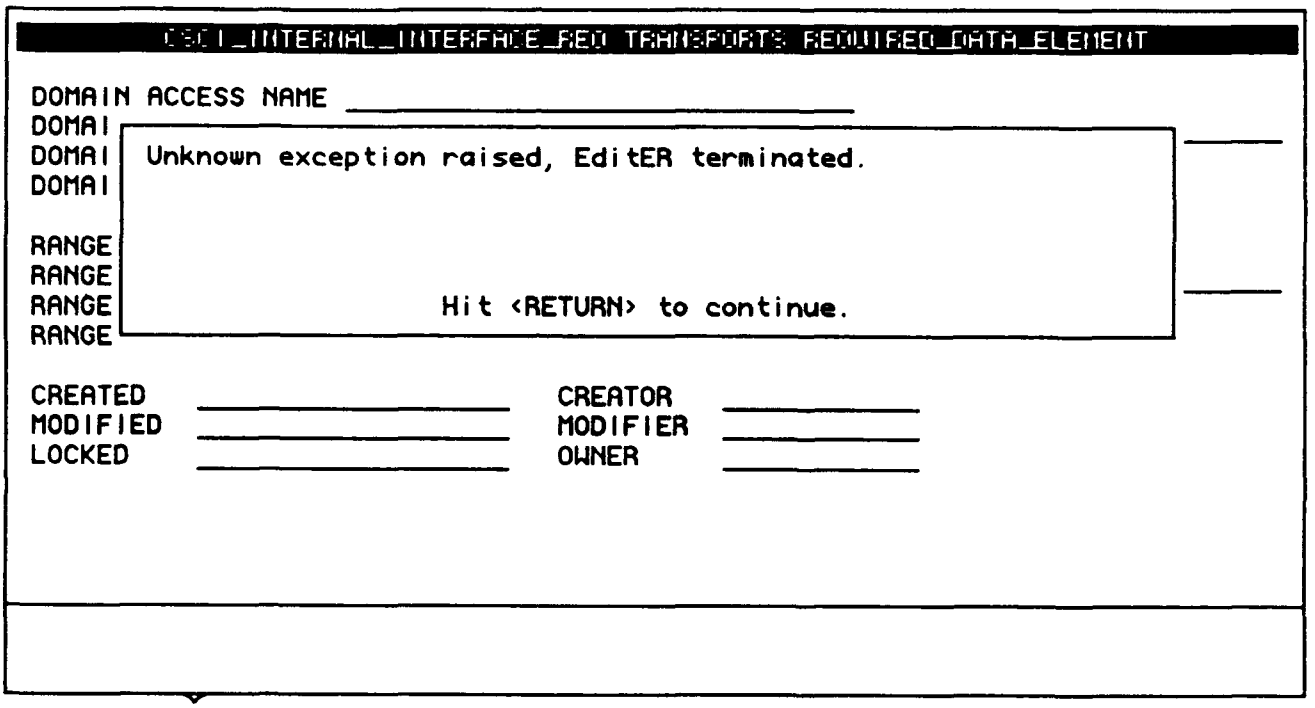


Figure 7. EditER Aborts

In figure 7, EditER abruptly terminated while trying to view the instances of the selected relationship type. Pressing the Return key returned the user to the main SLCSE window.

This operation had been performed prior to and subsequent to this failure, so it is not known what caused it as it was not consistently reproducible.

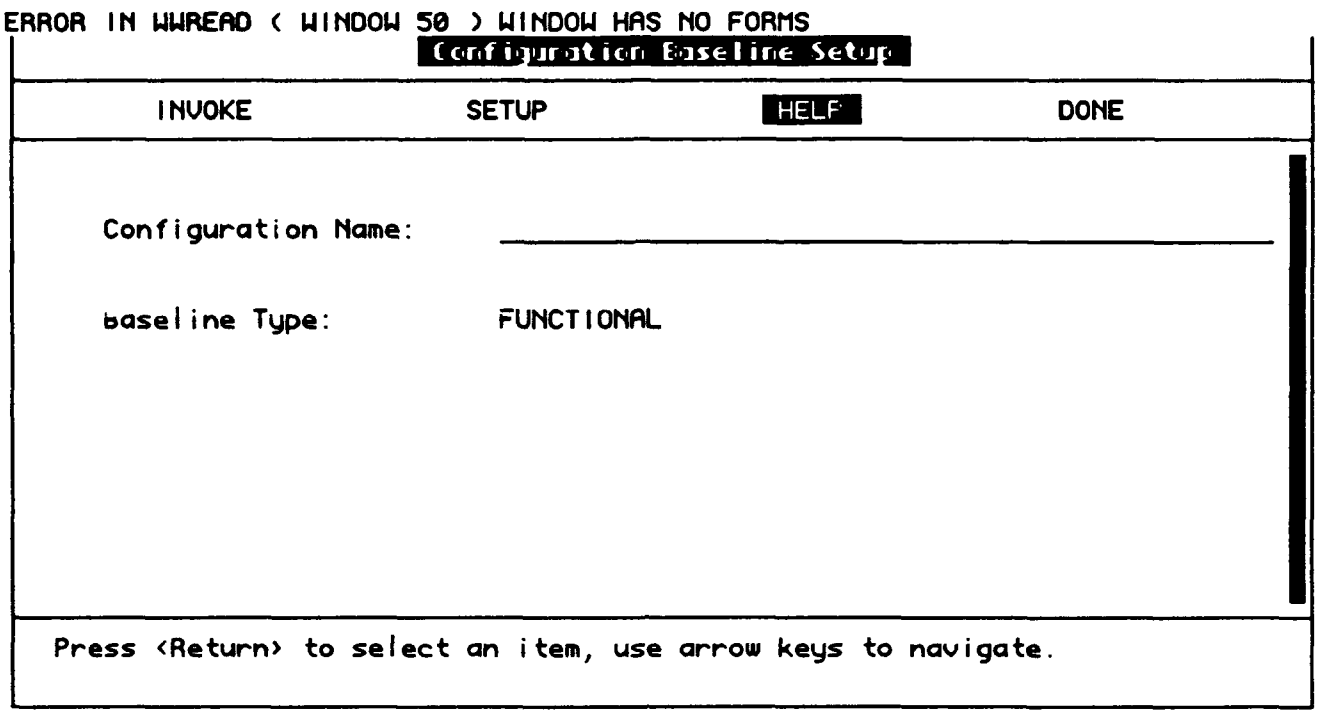


Figure 8. BaselinER Crashes

In figure 8, an error in the window setup caused BaselinER to completely hang up, forcing the use of the Control-Y command to abort the tool. This problem should be easily fixed.

In figure 9, an unknown error occurred while in EDT Setup mode which caused SLCSE to completely abort, throwing the user back into the VMS command shell. This error was also not consistently reproducible. We did note that EDT is not accessible from the DCL prompt once SLCSE is installed. Perhaps there is a link.

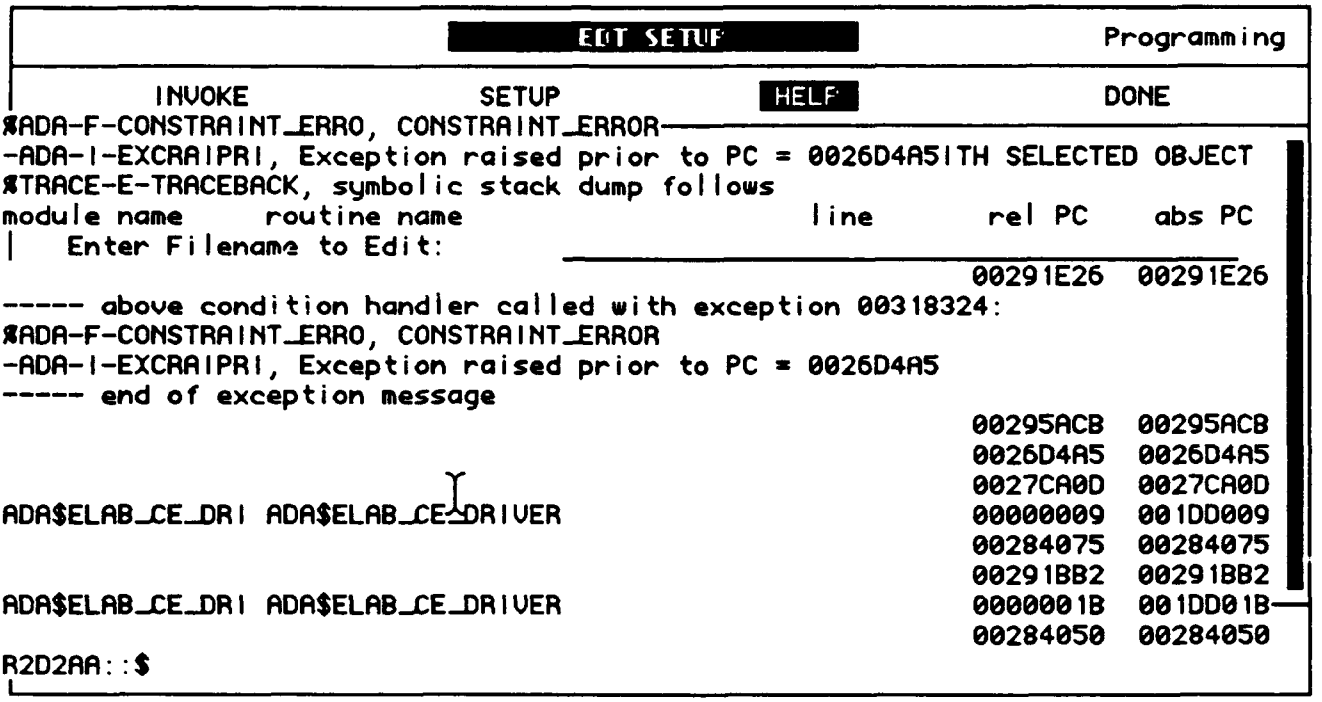


Figure 9. SLCSE Crashes

Additional testing for stability should be performed by a team of users accessing SLCSE concurrently to uncover additional situations in which problems similar to the above might occur and fixes should be made.

3.4.2.5 Capacity of SLCSE

In reviewing the documentation and in using SLCSE, we discovered no inherently limiting factors to supporting large projects. A large amount of disk space is required for the installation of SLCSE and tools such as ADL, AMS, Tex, and ATVS, in addition to the normal VAX-layered products, Ada, Fortran, EDT, CMS, MMS, etc., that would be used on a project. Due to the design of SLCSE–attribute information too large to be stored directly in a field in a Server table is stored in a VMS file under a user's account–a large amount of VAX disk space must be dedicated for each SLCSE user in addition to the disk space that is provided by the Server. Since almost all of the tools in SLCSE currently run only on VMS, a large amount of processing power must be provided by the VAX system.

The host of our SLCSE system was a MicroVax II. It supported the training course adequately, with three concurrent users, but only because steps were taken to eliminate concurrent access to the same parts of the database. A MicroVax II was also used during the hands-on assessment, with the ShareBase Server used as the database engine. Performance

was very slow, especially when doing database saves or searches involving many entities at one time. Based on this, we believe that, when sizing a configuration, the performance equivalent of a VAX 3100 workstation (a VAX 3100 provides approximately three times the performance of a MicroVAX II) be allocated for each SLCSE user. This may be done by clustering an appropriate number of large VAXes, such as the VAX 6000, as multi-user timesharing systems, or by using single user workstations networked to a single, powerful, central VAX acting as the database server. This VAX would also control access to the ShareBase Server, if used.

3.4.3 Results of SLCSE Usage

For the hands-on evaluation of SLCSE, we chose the IADS Software Engineering Prototype (SEP) program. We chose this project because it is a new project and has requirements and design documentation available in electronic form. It is using a tailored version of DOD-STD-2167 reflecting the object oriented design approach being taken. It might also use SLCSE if the results of the evaluation indicated the benefits derived would justify its installation at MITRE and its continued use on the project.

3.4.3.1 Production of Documentation

It was important to have the documentation on-line to minimize the purely mechanical aspects of inputting a large volume of existing data. By using a Macintosh as both a word processor to read the document (which was in Microsoft Word format), and as a terminal emulator (to access SLCSE) we were able to cut and paste text from the on-line document into the appropriate attributes of the entities as they were created. Even so, this took a considerable length of time. We were slowed down by the fact that we were never able to access SLCSE through the ESD local area network, relying solely on a 2400 baud modem link to access SLCSE. Due to an address resolution problem, the ESD gateway computer would not allow telnet access to the computers in the CCES. Although we had expected it to be, this issue was not resolved in time for us to enable multiple users to access SLCSE concurrently. Thus, we had the equivalent of one person working full-time for four months to build the database.

To properly populate the SLCSE database, we had to input the correct information from each section of the existing IADS documentation into instances of the appropriate entity types in the database. This primarily involved dissecting the IADS Software Requirements Specification (SRS) and mapping each section, subsection, or paragraph in the functional requirements chapter into the CSCI_Capability and Required_Data_Element entities and then constructing the appropriate relationships. Since the SRS was developed in accordance with DOD-STD-2167, each major function was documented in an input, processing, output format. This required us to analyze and recombine where necessary, the input and output data to put it in a form that conformed to the Required_Data_Element entity type definition. A Required_Data_Element entity was then created for each of the data elements described. CSCI_Capability entities were created for each identified major function.

The processing information for each major function was hierarchically decomposed in the SRS and we followed the same structure in creating CSCI_Capability entities that were partitions of the major functions. One function in particular, the Radar Data Processing

Function, required a partitioning depth of five levels to adequately capture all the requirements. Since the number of horizontal branches at the second partitioning level for each major function was generally between three and five, the entity structures rapidly grew. It became difficult to visualize the structure created in the database solely from viewing the relationships on-line because the SLCSE tools such as Requirements highlight all entities that have a relationship between the two entities in question. The reports provided by ReportER and VerifyER were somewhat helpful in checking the consistency of the database, but to do it correctly, requires a two-step process. First, we would generate a report listing all the entity pairs that have that relationship and examine it for correctness. We would then generate a report listing all the "missing" relationships for a given type, e.g., CSCI_Capability Partitions_Into CSCI_Capability, and determine which entities were really missing that relationship, and which are reported because that entity is at the bottom of the decomposition hierarchy and should not have that relationship. This process was very time-consuming because both types of reports are generated in batch mode only and, depending on system load and size of database, can take hours to complete. In addition, the on-line display within the Requirements tool was misleading. Before we realized the tool was displaying all range entities that have a relationship with the chosen domain entity, not just the specified relationship, we deleted and rebuilt the desired relationships several times. Further complicating this process was our lack of understanding of the purpose of some of the attributes presented when in the EditER, as mentioned in section 3.4.2.2. We found it necessary to revamp our naming convention to make the search and retrieval functions more efficient and to be able to read the names on the screen. This required re-creating those entities with new names and deleting the original entities because SLCSE prohibits modifying "Access_Name," Desc_Name," and PUID once a value is assigned.

Additional information was entered into the Contract subschema to provide the system name, contract number, etc., required for the title page and boiler plate text of a printed SRS. We then attempted to generate the SRS from the database. We discovered, however, that due to disk space limitations on the CCES VAX cluster, LaTeX was not installed.

To generate the document, we installed a copy of OzTeX (another public domain version of Tex for the Macintosh) on our Macintosh and downloaded the SRS.TEX file generated by Docgen_2167A, along with the necessary style files. We then printed the SRS locally. We had set the Print Location Tags option to yes so we would be able to identify and trace any missing information or information that was in the wrong place.

The document created contained many TBDs, of course, as we concentrated on CSCI requirements and required data, which are sections 3.2 and 3.4 of an SRS conforming to DOD-STD-2167A. The printed document, however, contained no requirements information in section 3.2. None of the requirements information from any of the CSCI_Capability entities was input into the section. A table of data elements internal to the CSCI was properly generated, but did not contain all of the expected data. We do not fully understand why the problem occurred although we believe it was, at least partially, due to the way the document generation code accesses the database by using particular keys such as the PUID. We were unable to fix the database and make another attempt due to time constraints.

3.4.3.2 Mapping to an Object Oriented Design

Our attempt to map the IADS object oriented design into the SLCSE design database was limited and only partially successful. The IADS software design document format was heavily tailored to capture both the CSCI design information and the definitions of 76 classes of objects. In that document, objects were mapped to computer software components (CSCs) and operations on the objects were mapped to computer software units (CSUs). Objects may also be further decomposed into smaller CSCs. We tried to follow that mapping in the database. Since each class is separately defined, each would be represented by a CSC entity. Projecting forward, however, there would be no way to create instances of each class in the database and represent them in the design subschema. Neither of the existing relationships, CSC Partitions_Into CSC and CSC Interfaces_with CSC, seemed satisfactory. We also found no satisfactory way to distinguish between classes if specific objects were represented as instances of the CSC entity type. The major problem here is that there is no way to represent the class of which objects are instances. In either case, there seemed no way to represent the concept of inheritance in the the current database schema. We were also hampered in this mapping because the IADS SDD was incomplete and did not contain sufficient information regarding their proposed mapping of CSCs and CSUs to classes and objects. Thus, we were unable to input sufficient data to attempt the generation of an SDD from the SLCSE database.

We believe further definitional work in extending the SLCSE schema to support object oriented design representations is warranted.

SECTION 4

TECHNOLOGY TRANSFER APPROACH

4.1 BACKGROUND

What is technology transfer? A common sense definition suggests that technology transfer is the process by which a specific, identifiable piece of technology is moved from a laboratory research project to practical government or commercial use, i.e., from being state-of-the-art to becoming state-of-the-practice. An example of successful technology transfer, in the domain of computer science, is the UNIX® operating system. Developed at AT&T® Bell Laboratories initially, as a single user, interactive multi-tasking operating system designed to provide a more productive environment (than batch processing) in which to do computer programming, UNIX has since become a de facto standard operating system, offered by every major computer vendor on every size platform, from Macintoshes (A/UX®) and PCs (XENIX®) to SUN®-class workstations and IBM®-class (UTS®) mainframes. Unix was able to do so because it was available from AT&T for only a nominal licensing fee, and because it was viewed as an open system, easily tailorable to new hardware platforms. Many universities acquired UNIX source code and used UNIX as a basis for further research and development. As a result, many computer science graduates had considerable experience with UNIX in their academic environment.

Each implementation of UNIX, however, is somewhat different, leading to incompatibilities in operating system calling conventions. Because of this and industry's desire to improve portability of tools and applications between UNIX systems, the POSIX effort was established to define a common set of UNIX-derived operating system interfaces. On the other hand, it must be noted that it has taken 20 years for UNIX to make the transition from initial concept to its current wide acceptance.

The technology to be transferred can take many forms. It may be a software or hardware product to be purchased outright as an end in itself; a manufacturing or management process to be installed in a factory or implemented in an office; it may be a software architecture; it may be in the form of knowledge itself, e.g., a formal design technique; or it may be a combination of the above.

As described in the above example, the transfer of UNIX technology is primarily in the form of a source code product (and license), but also in the knowledge of "what" UNIX is, what operating system services should be provided in general, and how they should be implemented on various computers.

4.2 TECHNOLOGY TRANSFER MODELS

To provide a basis for discussing the transfer of SLCSE, we are using two technology transfer models/approaches. The first approach was developed at the Microelectronics and Computer Technology Corporation (MCC) by Babcock, Belady and Gore, and is described in

[Babcock, James D., Laszlo A. Belady and Nancy C. Gore. "The Evolution of Technology Transfer at MCC's Software Technology Program: From Didactic to Dialectic," MCC TR STP-404-89, published in the proceedings of 12th International Conference on Software Engineering, Nice, France, March 1990]. The second was developed at the Software Engineering Institute by John Maher and is described in *[Maher, John H, Jr., "Planning for Technology Transition," Seventh WAdaS Tutorials, June 1990].* The MCC paper, in particular, documents some of the problems that the MCC has had in transitioning its prototypes to its industrial affiliates.

4.2.1 MCC Experience with Technology Transfer

The MCC paper describes the set of activities that MCC originally performed to transfer technology to its affiliates, and then describes where this model failed. The primary failure occurred when the MCC-developed prototype was "thrown over the fence" to the recipients, and failed to meet the user needs, both functional and in terms of reliability and usability.

According to MCC, "success" is defined by either of the two following events:

- A technology is fully integrated into a comprehensive design application or process used by affiliates to create profit-generating products. Domain-specific knowledge must be wrapped around the tool for this to be the case. In this case, success is measured by process improvement; the company has learned a new technique and is using it in producing products.

- The technology has been the basis for a "productization" that has generated profit for an affiliate company. In other words, the affiliate has turned the technology into a product in its own right.

The emphasis in the new MCC model, shown in figure 10, is on a collaborative approach to "fill the gap" and effect technology transfer between the developer and the recipient. In this approach, the developer and recipient work together on a series of evolving prototypes. Each prototype moves closer to meeting the recipient's needs, including functional needs, user interface requirements, and prototype stability and reliability. As a result of this collaboration, the direction taken by the developer often changes from the developer's original plans. This arrangement will continue as long as the recipient believes that there is still chance for "success" as defined above, and is willing to work with the developer to bring the technology closer to the recipient's needs. The recipient commits to spending the time to evaluate prototypes and provide the developer with feedback, and the developer commits to enhancing the prototype to better meet the recipient's needs.

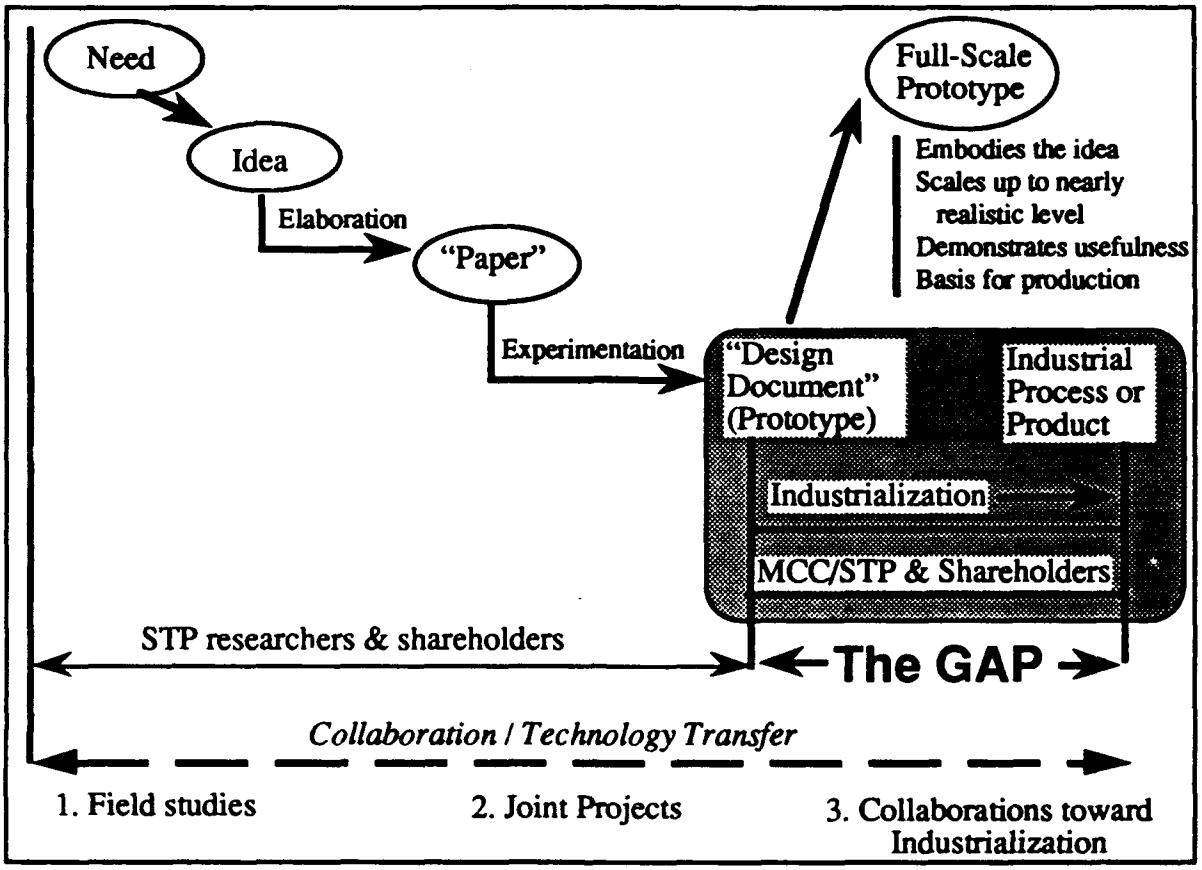


Figure 10. The Technology Transfer Gap and Three Major Types of Technology Transfer Activities [from MCC TR STP-404-89]

4.2.2 SEI View of Planning Technology Transfer

The Software Engineering Institute (SEI) approach actually consists of two models. The first model establishes the "product" view of the technology, answering the question "*What are you transferring?*" The second model shows the process, answering the question "*Who is doing what to make this happen?*" Figure 11 graphically illustrates the context in which technology transfer occurs. New technologies are produced and advocated or marketed by various organizations. Receptors are generally advocates and agents of the new technology in the using organizations who champion the use of the technology.

The SEI tutorial material contains a set of questions that must be answered by the various people involved in technology transfer [Maher, J.]. Answers to these questions can be viewed as the preconditions for successful transfer.

The set of questions that define the "product" include the following (paraphrased):

- What is the user's need that this product will satisfy?
- What is the current mode of operations performed by the intended user? How will this mode change with the new product? (Clearly the intended user must be willing to change if the transfer is to be effective.)
- What constraints (e.g., budget, personnel, time) does the user have?
- How will the user measure successful transition of the technology?

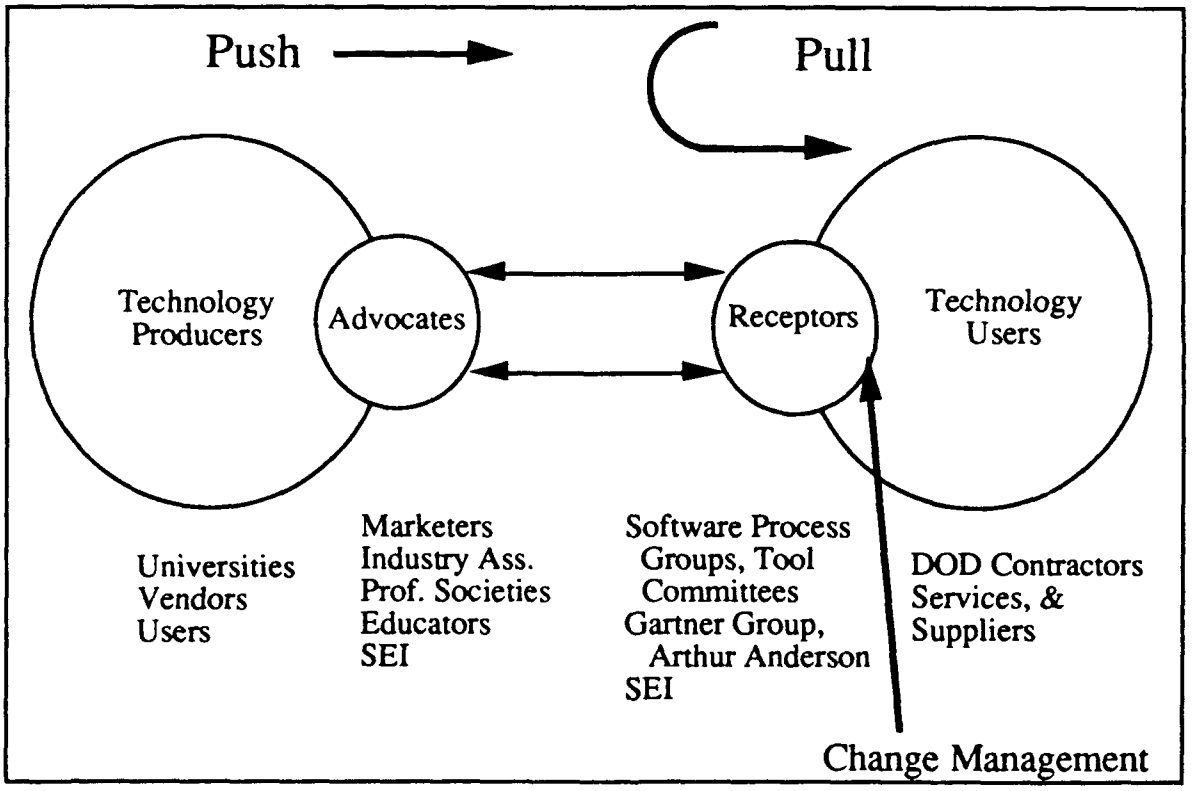


Figure 11. Technology Transition Context [Maher, J.]

The SEI process model for technology transfer identifies four roles in technology transfer:

- *Sponsors*: Individuals or groups who can legitimize the transition effort.

- *Champions*: Individuals or groups who want the new technology, but who have no power to legitimize the effort. They are often called product champions.

- *Agents*: Individuals or groups charged with implementing the effort.

- *Targets*: Individuals or groups who have to use the technology; those who must change the way they work.

The SEI tutorial emphasized the importance of obtaining the right sponsors. It is essential to obtain strong sponsors, and to maintain their commitment during the transition. Among the requirements for the sponsors are:

- The sponsors must have strong communication skills.

- The sponsor must have strong management skills and the ability to accept and tolerate uncertainty and change.

- The sponsor must demonstrate his support for the transition, and must make the transition a priority.

- The sponsor must understand the resource requirements for the transition, and be committed to providing necessary resources.

Targets are the other critical part of the SEI model. The main objective is to overcome the target's resistance to change and to demonstrate (in large part via support from sponsors) the positive results of making the change.

The people who help the targets overcome their resistance to change in the SEI model are the transition agents. Here are some of the characteristics of effective transition agents:

- Agents are considered extremely competent in their job duties and are knowledgeable of the targets.

- Agents have earned the trust of the sponsors.

- Agents have a high tolerance for stress and uncertainty, and are effective in working in unstructured environments.

- Agents understand the formal and informal power structures in an organization, and understand how people change and react to change.

Together, we believe that these two approaches to technology transfer provide the basis for planning successful technology transfer of SLCSE from RADC to ESD. The MCC model emphasizes the need for collaboration between RADC and the SLCSE developers and ESD and the SLCSE users. The SEI models provide a framework for identifying critical people in the technology transfer, and provide an outline for the tasks to be performed by each role during the transition process.

4.3 APPROACHES TO TRANSITIONING INTO ESD

4.3.1 The Government-Contractor Relationship

In general, the introduction of a new technology into an organization requires the organization to make changes. Changes in policies, procedures, support, personnel training, and the costs associated with these changes, create inherent resistance to adopting the new technology. At contractor organizations, financial constraints caused by the need to minimize overhead rates and bid costs also minimize the funds available to invest in capital improvements and training of personnel. To introduce a new technology, additional training costs must be allocated, a support system must be established, and experience and confidence must be gained before the technology will be applied on a real project. Thus, new tools are thoroughly scrutinized, evaluated, and a critical need for the technology is established, before any investment is made. Also heavily considered in the acquisition of a new technology is the viability of the vendor and the potential for long-term support.

When the government is the developer of the technology, the maturity and support issues become even more critical. The government is usually not prepared or willing to provide the long-term support necessary to successfully establish a product at a using organization. In general, when the government provides a product as GFE to a contractor, the government can be held liable for problems caused by the failure of that product. To avoid this, the contractor is usually required to assume all responsibility for the product. Yet, to ensure a successful transition, initial training must be provided, and a support organization that can provide a high degree of "handholding" to assist the end user in using the product properly, must be established. The end user must be guaranteed that deficiencies discovered during use of the product will be fixed, with workarounds provided in the interim. Support may also have to be provided to assist the using organization in integrating the use of the product into the organization's standard development methodology.

4.3.2 Transitioning to ESD Contractors

Because of the lack of support for government furnished equipment (GFE), contractors are reluctant to use GFE software on real projects that have to meet stringent cost and schedule requirements. They fear that if a serious deficiency develops, and the product fails to function properly, successful completion of that contract may be jeopardized. Generally, a company will only use GFE software if the company can be indemnified against the failure of the software to meet requirements. ESD, as a matter of policy, rarely provides as GFE software to contractors because ESD does not want to be held responsible for any problems that may occur in the use of the product, nor open the door to the possibility of the contractor demanding cost and schedule relief as a result of problems, real or perceived, caused by the government.

Even with all the impediments to transitioning technology into ESD and its contractor community, we still believe there are some approaches to explore that may be successful.

One approach to getting SLCSE used by ESD contractors would be to contract for an experimental, perhaps throwaway, product where SLCSE must be used as the development

environment. The contractor would be required to follow his normal policies and procedures and use his traditional techniques, but with SLCSE providing the automated support, instead of his in-place system. Feedback from the contractor, on how well SLCSE fits his way of development and what changes might be needed or recommended, would be gathered.

Another approach would be to require the use of SLCSE as the development environment during a planned research and development effort, using 6.2 or 6.3 funds, to continue development of one of the prototype tools in SLCSE. For example, we suggest in section 6 investing in further development in ALICIA. ESD contractors, needing such a tool, might be very interested in bidding on such a job. Using SLCSE as the environment for this work could potentially have a synergistic effect in determining host and target improvements for both SLCSE and ALICIA as both are being used. This approach could be implemented by GRC or SPS in enhancing ALICIA or to make improvements to SLCSE. MITRE could also be tasked to use SLCSE as the development environment in a separate ESD project involving prototyping an application in Ada or other supported language.

Another approach would be to modify an existing contract at ESD, which will be developing a real product, to require SLCSE as the development environment, but provide additional funds to the contractor through the SPO for the initial training of contractor personnel, and for ongoing technical support to the project.

Yet another alternative vehicle for supporting the transition of SLCSE to ESD contractors is the omnibus contract, formally known as the Command Center(s) Requirements Contracts.

To reduce the cost and time it usually takes to develop command center systems, ESD is planning to implement a process by which new command center systems can be developed through reuse and redevelopment of existing command center software. Following the STARS (Software Technology for Adaptable, Reliable Systems) model, three contracts will be competitively awarded to qualified bidders who will be tasked to develop generic, reusable components out of existing command center software. When a new command center is to be developed or an existing system is to be updated, a prototype system will be developed in the CCES using the generic components. Once agreement is reached with the user/requester, final specifications will be developed and a task issued to one of the three contractors to develop the full system. If extensive new development is required, then an open competitive source selection will be performed. The three contractors will also perform sustaining tasks to operate the Command Center Evaluation System, the Multi-Level Secure Laboratory and develop computer resources management technology. The omnibus contracts are expected to be awarded in the first quarter of fiscal year 1992 by ESD/AVS.

4.3.3 Transitioning to ESD SPOs

ESD SPOs are just beginning to use automation in their jobs. One of the reasons for this is that few systems support system acquisition, rather than system development. Much of the software currently used by SPOs was developed in-house, or by other government agencies. The Contract Data Management System, for example, was developed at Aeronautical Systems Division, and acquired by ESD. This software is often unsupported, non-commercial products, and the users do not have much time to devote to customizing or supporting the software.

One traditional weak point in the SPO shops is their ability to manage and examine the large quantities of documentation that they receive. DOD-STD-2167A defines the documentation set for software acquisition, but, despite the level of automation in industry, these document deliveries are most often made via trailers of paper copies. The task of evaluating the documentation is reduced to a proofreading drill, by poring over paper copies of the different deliverables. For instance, doing a requirements traceability audit requires the auditor to manually move from the system requirements specification to the software requirements specification to the software design specification.

We believe that SLCSE could have much to offer to an ESD SPO, as a repository for contractor-developed and SPO-developed products. Initiatives such as DOD-STD-2167A and the Computer-aided Acquisition Logistics Support (CALS) provide the framework for automating the SPO functions by defining the content and format of data to be delivered. The SPO users can use the information stored in the SLCSE database to do much of their consistency checking and auditing, and could also use tools such as ALICIA to study the impacts of requirements and design changes that are now performed without much automated support.

We think a pilot application in a SPO organization to support the SPO project management and technical review activities, performed as a shadow project, could demonstrate the advantages in using SLCSE. Such a project could occur subsequent to the SLCSE Technology Exploitation task and be performed by MITRE or SPO personnel applying SLCSE on a real acquisition. Separate funding from RADC could be used to eliminate funding issues with the SPO and provide further incentive. The shadow project could be done on a low or non-interfering basis; it would collect data, anecdotes, and testimony from project and contractor personnel, while performing the task, to determine the effectiveness of using SLCSE and the contributions it might have made. Training and on-call support, as needed, could again be provided by GRC to minimize support concerns.

4.4 ISSUES IN TRANSFERRING TO ESD SPOS

For the rest of this section, we will concentrate on planning the transition of SLCSE into an ESD SPO organization. The reason for this is that we think a transfer to a SPO, rather than to a contractor, is most feasible. Given the need for interaction and for sponsorship documented in the MCC and SEI technology transfer models, a SPO shop is much more likely to provide the necessary support, without the contractual impediments that would interfere with an initial transfer to an ESD contractor.

One of the primary needs of the SPO is to receive and process the large number of deliverables specified by DOD-STD-2167A. Typically, the SPO is responsible for reviewing the documents, and ensuring that they meet both the "syntactic" requirements of the DID, and also the "semantic" requirements of consistency and traceability with other project deliverables. As mentioned earlier, this has traditionally been a proofreading performed by the SPO or support organizations such as MITRE. By placing the contractor deliverables into the SLCSE database, the SPO can run the variety of automated tools provided by SLCSE (or developed separately and integrated with SLCSE) to analyze contractor products. Code

deliveries can be placed in the database and analyzed for complexity, structure, timing issues, etc. Traceability links from the code back to the design and requirements specifications are required to enable a SPO to determine the impact of a proposed change.

A very rough process description for the SPO is as follows:

Develop the System Specification
Develop the RFP and the SOW
Conduct Source Selection
Monitor Contractor Work
Conduct Reviews IAW DOD-STD-2167A and MIL-STD-1521B
Prepare and Conduct IV&V
Prepare and Conduct FQT
Transition Project to Using & Maintenance Organizations

For SLCSE to be successful, it must be useful to the SPO in several of these activities, and also must not adversely affect the SPO in other activities. Therefore, it is important to identify how SLCSE will support these SPO activities. Prior to the actual transition of SLCSE to a given SPO, it will be necessary to develop a much more comprehensive process model of the SPO's functions, and identify, for each step, how SLCSE will support the SPO's requirements, and what the SPO personnel must do at that point to work with SLCSE.

For instance, prior to PDR, the contractor should deliver a preliminary version of the SRS to the SPO. This version should be entered into the SLCSE database. SPO personnel can then use the SLCSE database to examine requirements traceability and to identify any requirements in the System Specification not covered in the SRS. Additionally, the SPO personnel can examine specifications such as the timing and sizing data, and comparing the contractor's estimates to estimates developed by the Government. Particularly for an acquisition where there are multiple PDR's on different parts of the system, it will be very useful for the SPO to see the new SRS as compared to the old SRS, and to compare the SRS for one CSCI to the SRS of another CSCI, and the IRS that specifies the interface between the two.

A significant problem for integrating SLCSE in an existing SPO shop is that effective use of SLCSE by the SPO will require some changes to how the contractor delivers documentation. SLCSE will be ineffective if the contractor's deliverables cannot be easily incorporated into the SLCSE database. This means that the contractor must deliver his documentation on tape in a format that can be quickly entered into the SLCSE database. Depending on the SLCSE requirements for loading an entire deliverable into the database, it may require substantial work on the part of the contractor (or SPO) to transform the documentation from the contractor's internal format to the format required for entry in SLCSE. At the very least, the Government must be willing to pay the contractor for any additional charges the contractor incurs to meet this new requirement. This is less of a requirement for a new project, where the delivery format can be worked into the contract. However, both the SPO and the contractor must understand the SLCSE format and how it will be used to ensure that the contractor's deliverables provide SLCSE with the right information in the correct format.

4.5 TASKS TO BE ACCOMPLISHED BY RADC

To ensure a successful transfer of SLCSE, there are several tasks that must be completed by RADC. These include "productizing" SLCSE, identifying the specific SPO (or contractor) to receive SLCSE, working with the recipient to define the recipient's transfer plan, establishing a joint improvement/support activity for SLCSE, and developing a training program for the recipient. From the RADC perspective, the activities comprise three phases:

- Selecting recipient(s) and preparing SLCSE for delivery. This phase will take about one year, with most of the time being used to "productize" SLCSE prior to delivery to the recipient.

- The initial delivery, installation and beta-testing of SLCSE. This should take about six months.

- Ongoing maintenance and enhancement of SLCSE. Given the average three year life-cycle of an ESD project, RADC should expect this phase to last another two and 1/2 years after the initial delivery to the SPO/Contractor.

Productization of SLCSE is covered in section 5.

RADC must work with ESD to identify a SPO or contractor (or both) to receive SLCSE. Initially, SLCSE will be in a beta-test mode, where the recipient and RADC are working together to resolve problems with SLCSE. This should last about six months. During this time, the SLCSE recipient must be willing to accept the fact that there will be bugs in the product, and that some rough edges remain to be resolved. To be successful, the recipient must be able to manage the risks posed during this initial period.

Once the recipient has been identified, one of the first actions that RADC must do with the recipient is establish a joint group to work out the details of the transfer. This group will be responsible for several activities, including:

- Writing an implementation plan that includes schedule, resources, and training.

- Establishing procedures for resolving questions, bugs, and improvements.

- Establishing funding for SLCSE hardware, software, and maintenance/support.

To be able to evaluate the effects of the transfer, RADC should work with the recipient as he develops a description of his current process. In particular, RADC should look for ways to measure improvements in the recipient's process brought about by using SLCSE.

Finally, RADC needs to develop a training program for the recipient. This training program should provide several levels of training, including:

- User Training (emphasis on using SLCSE to support software development).
- Management Training (emphasis on how SLCSE supports the existing process, and on using SLCSE to get management information).
- System Administration (including administration of COTS components of SLCSE).
- Database Administration/Tool Integration (emphasis on customizing SLCSE).

One of the requirements to productize SLCSE is to prepare generic training courses and materials. Here the emphasis is on taking those training materials and customizing them for the specific recipient. A fully developed sample problem in the user's domain illustrating all aspects of constructing the database, and a detailed sample Software Development Plan, if the user is a contractor, illustrating where and how SLCSE is used could be developed.

4.6 TASKS TO BE ACCOMPLISHED BY THE RECIPIENT

There are two critical tasks for the recipient of SLCSE if the transfer is to be successful. First, the recipient must understand his process and how SLCSE fits into this process. This includes an understanding of how to measure the benefits of SLCSE. Second, the recipient must have a champion--someone who is committed to the SLCSE technology transfer and who has sufficient organizational power and influence to overcome the hurdles that may occur.

Both technology transfer models that we have reviewed emphasize understanding the recipient's current process and how that process will change with the new technology. In some respects, this is obvious. If you don't know what you're doing now, you won't know how to improve. The requirement for understanding the recipient's process goes deeper than a simple overview of what the recipient does day-to-day.

There are a thousand impediments to successful technology transfer within an organization. Many of these are due to human nature, particularly reluctance to change. Also, any new technology presents a risk that must be accepted and managed. The recipient must have a champion who is willing to work within the organization to overcome user resistance, and who is willing to accept the risks involved with the new technology. This champion has to be able to implement the necessary changes to schedule, personnel, etc. that can occur as a result of problems learning or using SLCSE.

There must be overall support from the recipient's management team. The champion cannot single-handedly change the organization. In particular, the management team must be willing to work with the champion to motivate the users to use SLCSE effectively, and also must work with the champion to overcome the risks and problems that occur when using a new technology.

The recipient and RADC need to form a joint committee to plan and monitor the SLCSE insertion. From the recipient's side, this is the forum for identifying issues with SLCSE and working with RADC to resolve the issues. One of the first products of this joint committee is

a plan that shows both how SLCSE will be physically installed in the recipient's facility, and also the new process model that shows how the recipient will use SLCSE. The recipient must work with RADC to provide the resources for running SLCSE, and the funding sources for supporting SLCSE (hardware, software and training/consulting support).

4.7 OTHER TRANSFER ACTIVITIES

We believe SLCSE, or portions of SLCSE, could be put into the various government repositories, e.g., STARS repository, SIMTEL-20, a possible ESD RAPID repository, if one is established. The STARS and SIMTEL-20 repositories have been established to provide storage and access to government-owned software that may be of specific interest to government contractors. The STARS repository will hold the products of the STARS program being developed to improve the practice of software engineering in DOD contractors. SIMTEL-20 is a general repository, managed by the US Army and accessible to the DOD community at large, which contains much software developed at government expense and which may be acquired and reused at the user's own risk and expense. RAPID is an US Army project to develop a center dedicated to providing a full-service reuse library. Users will be able to classify, store, analyze and retrieve reusable Ada software components. A similar center at ESD would be capable of providing access to reusable components for command and control applications and also tools to develop those applications. With the emphasis in DOD to improve the software engineering capabilities of contractors, this would permit and encourage the general contractor community to obtain and experiment with SLCSE on an informal basis. IR&D funding could be used to support its evaluation and trial application on internal efforts. Licensing, copyright and data rights to SLCSE must be clarified, particularly with respect to proprietary software (WINNIE, MOO) developed by GRC and used in SLCSE.

Finally, we believe that other government centers need to become familiar with SLCSE. This could be accomplished by arranging briefings and demonstrations, and if warranted, installing SLCSE for test and evaluation. RADC might consider funding the installation at selected sites in the near term to minimize the possibility of refusal. The following places should be considered:

- STARS Technology Center - The STARS program has been established by DOD to develop and make available to industry, new or improved software technologies and products that may improve a contractor's software development process and resulting delivered products. Repositories have been established by each of the three STARS contractors. SLCSE should be included as a technology and as a product which can be provided to a contractor to improve his software engineering capabilities.

- USAF Computer Resources Acquisition Course (CRAC), Brookes AFB, TX - Students of the acquisition process should become aware of tools which may be of use to them in the performance of their jobs.

- USAF Institute of Technology, Wright Patterson AFB, OH - USAF officers studying software engineering and related technologies need to be exposed to the

latest technologies and become familiar with tools and technologies that might be potentially useful to them when assigned to acquisition or logistics commands.

- USAF Academy, Colorado Springs, CO - Cadets should be exposed to software technology work in the USAF as a general part of their education.

- Software Technology Support Center, Ogden AFB, UT - The USAF logistics centers are potentially the most important sites for SLCSE to be installed, as they will require a modern software engineering environment to maintain flight software.

- SDI Software Center, National Test Facility, Falcon AFB, CO - This has been established to evaluate products and technologies potentially useful to the SDI effort. As the SDI software development effort is estimated to be as large as 10,000,000 lines of code, an appropriate development environment will be required.

4.8 LESSONS LEARNED FOR FUTURE TECHNOLOGY TRANSFERS

We believe that SLCSE is not unique in its requirements for technology transfer from RADC to ESD. There are several lessons learned that RADC could use for future technology transition efforts.

We recommend the establishment of a single technology transfer agency at RADC, and a counterpart at ESD. The RADC agency should start the planning for technology transfer, and should have a source of funding to support efforts like the "productization" of SLCSE. For a software project like SLCSE (or the RADC Knowledge-Based Software Assistant (KBSA) work), the RADC transition agency should include people experienced in training requirements, analysis of computer systems, configuration management, computer system administration and software process modelling. The agency should report to senior-level management at RADC, to assist in overcoming the administrative hurdles that both the SEI tutorial and the MCC report document. The transfer agency at ESD should employ individuals experienced with acquisition and the contracting process for ESD systems, and individuals skilled in computer system acquisition and installation. Like the RADC transfer agency, the ESD agency should also have sufficient authority in ESD to overcome administrative hobbles. Funding is less important to the ESD agency than this authority; with sufficient authority it can obtain funding from the supported project.

As both technology transfer models point out, continued interaction with the technology recipient is essential. RADC and ESD should investigate ways for RADC to interact with ESD SPOs and ESD contractors on a continuing basis. One approach used by several research consortia, such as the SEI, MCC and SPC, is the notion of "resident affiliates." For instance, at the SEI, a company sponsors one of its employees at the SEI for a period of six months to a year, to work on a specific SEI project. This gives the SEI the benefit of the employee's experience, and the employee is exposed to the wide set of technologies (not just his individual project) available to his sponsor from the SEI. Another approach is to periodically give briefings by RADC personnel to ESD, MITRE, and ESD contractors. This is analogous to the "member meetings" at MCC and SPC.

Finally, as noted earlier, there is a tremendous amount of support that must be provided by the technology developer during the technology transition. Figure 12 illustrates the stages an organization goes through in adapting to a new technology. It may take many months for an organization to reach the Adoption/General Use stage for a specific technology. Yet, it is still possible that a technology transfer effort can fail at any point prior to the Institutionalization level of commitment.

In addition to the requirement to productize SLCSE, there are the requirements for consulting and troubleshooting, maintenance and enhancement, and training. As the SLCSE experience shows, RADC does not currently have the mission or the funding to provide these services on an ongoing basis. The Air Force needs to identify a productization/support agency for laboratory products, particularly if those products are to enter general, widespread use. RADC can establish a technology transfer group with ESD, to move RADC products out of the laboratory into ESD, but then the Air Force as a whole needs to decide how to support a product such as SLCSE once it gains widespread use at ESD. At some point in the technology transition life-cycle, the RADC involvement will wind down, and some other organization will need to take over the support of products such as SLCSE.

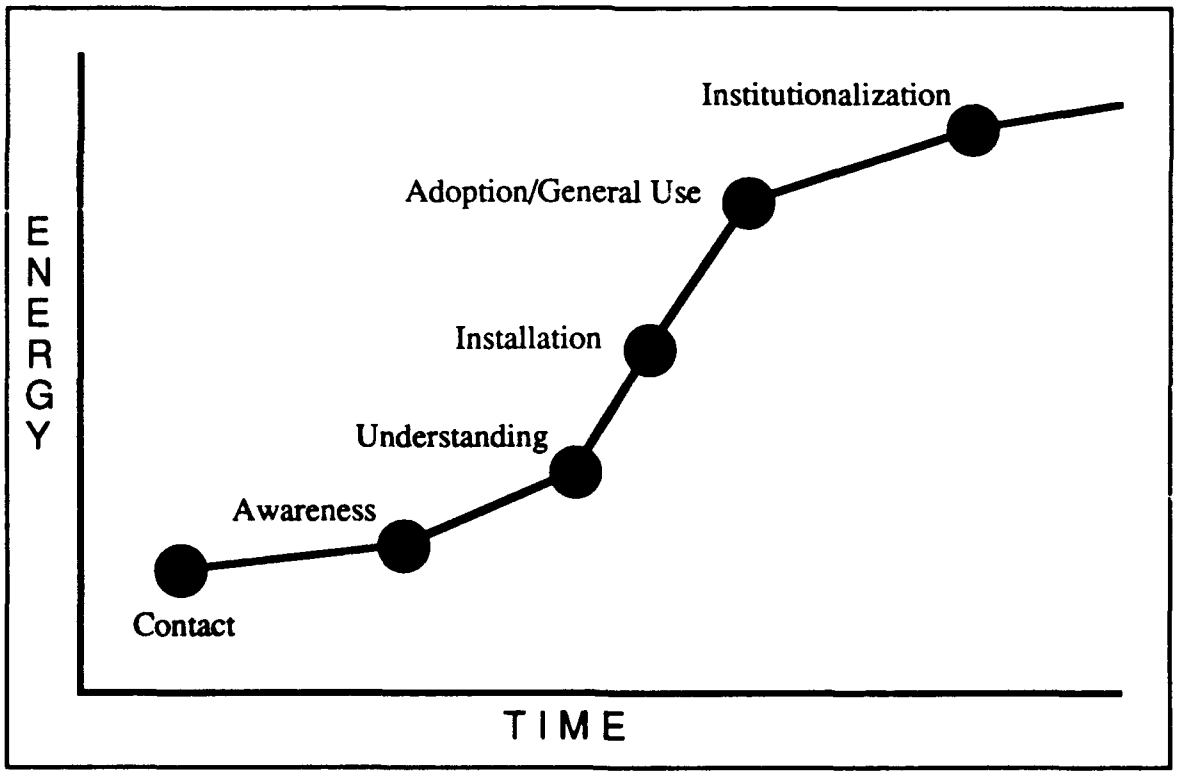


Figure 12. Stages of Adaptation to New Technology [Maher, J.]

SECTION 5

PRODUCTIZATION APPROACH

5.1 BACKGROUND

The Technical Objectives and Plans for this task state that the productization plan should "detail the changes that are necessary to 'productize' SLCSE for use on large scale C3I software development efforts, under the PE64740F or equivalent program." This implies detailing enhancements to SLCSE necessary to be used on large scale programs. Productization in the commercial market carries a somewhat different connotation, however. There, productization generally means doing all the things necessary to prepare a product for sale and subsequent support, most of which do not involve functional improvements to the software. Non-technical preparations include writing installation, users, and system administration manuals; preparing training manuals; and organizing, staffing, and training a support organization. Technical preparations may include "bullet-proofing" the software to make it more robust, improving the way operations are accessed or performed without changing the functionality of the operation, and providing minor performance improvements.

5.2 INDUSTRIAL EXPERIENCE IN PRODUCTIZATION

To gain the benefits of industrial experience with developing and supporting products, MITRE interviewed executives at several companies who have been involved in developing products or converting prototypes into products within the Ada community. Stowe Boyd, Vice President of R&D, Meridian Software; Steve Deller, Vice President of Product Support, Verdix; Jim Bladen, Vice President, Telesoft; and Karl Nyberg, President of Grebyn Corp. (and formerly of Verdix) provided much valuable information in this area.

5.3 PRODUCTIZATION PROCESS

The experts that MITRE interviewed all stated that there are four phases to the product life-cycle (after the initial prototype is developed). These four stages are Initial Productization, Beta Testing, Reproductization, and Ongoing Support.

5.3.1 Initial Productization

Initial productization starts when the prototype is complete and a decision has been made by the company to market the product. For SLCSE, this is the current state of the project as the prototype SLCSE system has been delivered to the Air Force and the Air Force has (for the purposes of this report) decided to "market" SLCSE to ESD. Activities during this phase include fixing known bugs, Quality Assurance Testing, improving the user interface, preparing documentation and training materials, and developing a support organization. Section 5.5 of this report covers some specific recommendations for work to be accomplished

during the initial productization. Several of the people MITRE interviewed stressed that it is important to determine if the market will justify the costs of productization before committing the resources to productize a prototype.

The length of this phase is mostly dependent on the condition of the prototype. One of the experts interviewed by MITRE stated that in one instance where his company productized a prototype received from a third party, they ended up completely redesigning and implementing the system from scratch. In another project, they had to do substantial redesign and reworking of the system, to make the system more reliable and to fix problems with the user interface.

The user interface is particularly important. One expert observed, "The best tool in the world won't be used if the user interface gets in the way." Since many prototypes have been incrementally developed, often the user interface has grown and changed as the prototype has developed. In other instances, where there are several different independently developed components, the problem is specifying a common interface covering the various components. SLCSE exhibits both tendencies in places, but overall the user interface for GRC-developed parts of the system is generally consistent. Instead, SLCSE has the problem that it includes several components not developed by GRC, such as the commercial DBMS, DEC® VMS® compilers and language-sensitive editors.

An issue related to the user interface is documentation. Like the machine presentation, the documentation presentation must also be consistent. Where the system consists of independently developed parts, the documentation for each part must be welded into a consistent manual set. In particular, it is important that the documentation for a component part be understandable and support the documentation for the whole system. For instance, in SLCSE, the database documentation should be presented so that the user of SLCSE understands how the database features work in SLCSE, and not in general. Where the system uses only part of the underlying components, then the best approach is for the system documentation to completely replace the component documentation. In SLCSE terms, this means that the SLCSE manuals must cover those parts of the DBMS (and only those parts) used by SLCSE. The end user should not need to go to SLCSE manuals and to DBMS manuals to use SLCSE.

It is also important that the system accept responsibility for its components not directly implemented by the developer. In the case of SLCSE, the underlying DBMS has a major impact on the entire system. Whoever productizes SLCSE must take responsibility for the reliability and usability of the DBMS. When a customer has a problem, he expects to get it fixed by the SLCSE vendor, and not have to take his problems to a third party.

Besides the "productized" system and initial documentation, an important product of this phase is a set of tests to be used later on for regression testing. Failure to develop a system for regression testing is a common mistake made by companies when they develop a commercial product the first time. Regression testing goes hand-in-hand with adequate configuration management.

5.3.2 Beta Testing

Every expert interviewed by MITRE stressed the importance of beta testing. When asked if one could skip beta testing, one expert replied, "Well, you might be lucky, but I wouldn't try it." The goals for beta testing are to identify needed improvements in the tool, and to provide stress testing that cannot be adequately duplicated by the developer's Quality Assurance program.

In many respects, beta testing is the single greatest risk during productization. The "worst case" scenario mentioned by one of the experts is when known problems are fixed going into beta testing, and one get 500 bug reports and improvement requests. These are fixed, the system is rereleased, and 1,000 more bug reports and improvement requests are generated. This reflects a situation where the product has been too narrowly focused, or is not flexible enough to support the beta user's needs. Often what happens is that the assumptions about how the product would be used are incorrect, and it becomes a major effort to redo the product with the beta user's assumptions.

The beta testing period evaluates both the product and its support. One of the critical parts of beta test is feedback on the documentation. Quite often the documentation is not well done in the rush to get a product to beta test. Another common problem is that the documentation assumes knowledge of the product (or how it will be used) on the part of the user that the user does not have. Usually, the beta test identifies gaps in the documentation.

5.3.3 Reproductization

Beta Testing and Reproductization are partially overlapping activities. The typical cycle is to send out a version for beta, receive bug reports, fix the problems, and send out a new version. This is where the developer can get caught in the loop of increasing the numbers of bug reports. The goal of the Beta Test/Reproductization cycle is to achieve convergence on zero outstanding problem reports, but this is rarely achievable in practice.

In general, Reproductization is oriented towards customer-generated changes. A large part of the work done in Reproductization is regression testing, to ensure that a change to one part of the system does not break other parts of the system.

Configuration Management is very important during Reproductization. With multiple beta sites, it is easy to get a large number of incompatible systems, as bugs are fixed and new versions of the system are released to the beta test sites. A related problem can occur when a customer reports a bug in one version of the system. The bug is fixed in a new version of the system, but in the meantime the user has developed a workaround. The problem occurs when the customer's workaround is incompatible with the fix in the system, and the customer has grown to depend on his workaround. When a customer uses beta test versions of systems for product development, he is often unwilling to test a subsequent release because of the impact on his own work.

5.3.4 Ongoing Support

The final phase of productization occurs when the product has completed beta testing and is released to the general user community. Preparing for and conducting ongoing support is a critical part of productization, and planning for support starts when the decision has been made to turn a prototype into a product. As mentioned before, Quality Assurance and regression testing must start with the initial productization of the system and be ongoing through the beta testing period.

One of the challenges for the supporting organization is to know when to fix bugs and issue new versions of the system. Each user wants his bugs fixed immediately, but does not want to go through the struggle of upgrading to a version that does not fix his bugs. Industry experience seems to be that semi-annual releases work best, with a provision for sending a special release to a customer to fix critical bugs. As bugs are fixed, the regression test must grow to include the customer bugs. Customers lose faith in a product when today's bug fix reintroduces a bug that was fixed in yesterday's release. Conversely, they will lose faith when their bugs are not fixed at all.

Ongoing support includes more than simple bug fixes. It also includes maintaining and improving the documentation and providing special services for customers, such as training and customization. Another mistake made by companies on their first product is not developing and providing training in the product to the customer base. With SLCSE, this training should include courses for SLCSE users and database administrators, and the supporting organization should be ready to provide consulting services for customizing the SLCSE database, modifying the SLCSE schema, and integrating new tools into SLCSE.

5.3.5 Costs and Schedule Experience

Although each expert had his own "rules of thumb" for productizing an existing prototype, each person also had different assumptions about the state of the system when productization starts and included different tasks in the estimate.

The rough estimate of how much it costs to productize an existing system runs from the cost of developing the prototype to 10 times the cost of the prototype. For SLCSE, this would yield a cost in the range of $2.5M to $25M. This is one reason why it is important to determine the actual market for a product before spending the additional money to productize it. These estimates generally include the "technical" costs, but do not include the additional costs of marketing the product. An additional proactive marketing effort, coordinated with the ongoing support effort, would also be recommended to publicize the advantages of SLCSE and to garner feedback from users for future enhancements and potential users to see if they are impressed enough to consider using it.

The time required to productize a prototype also varies. However, there was general agreement among the experts MITRE talked to that the minimum time was close to a year, regardless of the costs and personnel applied to the product. It takes this much time to do the initial productization and conduct a thorough beta test/reproductization. Usually beta test periods last three months, with at least one month for reproductization before full release.

For SLCSE beta testing, we believe a longer period should be allowed due to its nature. Six months to a year should be allowed to be able to thoroughly exercise all tools and all subschemas in the database.

The size of the support organization also varied. One expert said he plans on one support person per 200 users, while a second expert said that his goal would be one support person per 10 users, at least during beta test. One organization has an installed base of about 7,000 users for their product running on 2,400 machines. They have eight full-time customer support personnel (who mostly answer phones and investigate bug reports), three people doing bug fixing and customization, and another four people who do Quality Assurance and preparation for delivery. They also have one person who maintains the customer database, showing what sites have what versions of the software.

5.4 SPECIFIC PRODUCTIZATION RECOMMENDATIONS

5.4.1 Technical

The following technical improvements should be made to productize SLCSE:

- A bulk load capability should be provided. There should be a mechanism provided that will enable an existing project to format and enter existing project data into a SLCSE database in a batch-oriented, automated way. This would permit existing projects to retrofit their data into SLCSE without requiring large amounts of manual labor.

- All known problems with SLCSE should be fixed. GRC has a list of open problems, some of which adversely affect the integrity of the database. Other problems, which we have reported in section 3, should be further investigated and fixed, and any problems resulting from the beta tests should also be corrected.

- Eliminate reliance on SmartStar software as the interface between SLCSE and the underlying database management system. The acquisition and maintenance costs of SmartStar software can be very expensive when licensing such software for multiple, large VAX computers, as would be required for large projects. Also, our experience thus far has indicated the SmartStar software can be easily corrupted and lose access to the database.

- Improve the navigation capabilities of tools that provide a graphic representation of a subschema, such as ModifyER,and VerifyER. Currently, when using the arrow keys to move around the screen, the entity the cursor moves to is not always intuitively obvious. The cursor keys should allow navigation along relationships to simplify selection of objects on the screen.

- Add an incremental or dynamic schema modification capability. It should be possible to make additions to the schema, e.g., adding new attributes to an entity,

adding new types of relationships between entities without having to unload the database, recompile the entire schema, and reload the entire database if these additions do not affect existing data.

- Simplify the process of adding new tools to SLCSE. A dynamic window numbering allocation scheme and dynamic linking in addition to the schema modifications mentioned previously should be considered.

- Correct misleading or incorrect prompts that were discussed in section 3.4.

- In database access tools (Requirements, Design, etc.), when the relationships function is chosen, modify the pop-up window displays to highlight only those entities that have the specified relationship with the domain entity.

- Provide a way to easily move or resize windows that overlap on the screen. Long object names are often partially obliterated by an overlapping window. When there are multiple versions listed, the distinguishing aspects of the name, i.e., the version number, is often covered by an overlapping window.

- Provide an alternate means of highlighting a menu item. When the cursor is on an item, it looks exactly the same as a menu item that is chosen. Colorizing menu items when appropriate terminals, personal computers, or workstations are supported would reduce confusion.

- Provide an option to remain in SLCSE and choose another project. Currently, when a user wants to change from one project to another, the user has to completely exit SLCSE and reinitialize. If multiple projects share the same logical database, it should be possible to switch projects without exiting SLCSE. Also, if SLCSE cannot access a particular project, it quits.

- In the AMS_Analyze tool, a component type is referred to as a TLCSC. This nomenclature was used in DOD-STD-2167, but not in DOD-STD-2167A. The documentation should reflect the change to CSCI.

- Provide additional prompt information to state that when finished making the selections, pressing Keypad-0 will save the choices made as well as returning to previous screen. New users found it disconcerting that after making selections, there was no explicit way mentioned to save and act on the chosen information.

- Increase the length of the DESCR_NAME attribute. The specification title of an External_Document is stored in this attribute and it is too short for many of the long titles used on current systems.

- The DGL code for the 17 documents should be thoroughly exercised. We discovered a section formatting code in the SRS DGL was missing, causing incorrect section numbers to be generated. The sample database provided with the

training course was not complete enough to exercise all sections of the code for the SRS. Based on this and scanning the other sample documents, we believe this will hold true in the DGL for the other documents as well.

- Make the tool list object sensitive. When an object is selected, consider highlighting in bold the tools in the menu which operate on the selected object. Alternatively, gray out those tools which would not be chosen at that point.

5.4.2 Other

The following non-technical improvements should be made to productize SLCSE:

- A Concept of Operations document should be developed detailing the general way in which SLCSE can be, or should be used on a large project. Specific examples should be provided to highlight the benefits gained from using SLCSE.

- Additional training material should be developed to provide in-depth training in the concept and use of the database schema. Specific examples should be provided to explain what type of information each entity type is designed to capture, what goes into each of the attribute fields, what format the data should be in, how to create customized reports using the document generation language, etc. For example, SLCSE requires that data and control flow diagrams be stored in Postscript® format, but that fact is not stated anywhere in the documentation. The Access_Name and Descr_Name fields of entity types are defined as string data types, modifiable and unprotected. This is misleading as once a value is entered into one of these fields, that value is not modifiable.

- A support organization should be identified and established that will provide the on-going support to users of SLCSE.

- Help screens should be provided in tools that currently do not have them. Database_Insert, ATVS_Source_Reader, and TexPrint were three we discovered.

SECTION 6

FUTURE DIRECTIONS

Based on our investigations to date, our study of the SLCSE documentation, and the hands-on experience, we believe the technologies described below warrant further investigation for possible incorporation into SLCSE. In some instances, we make a firm recommendation for a technology insertion that we believe is necessary to increase the probability of SLCSE being adopted by ESD or its contractors. In other instances, we describe technologies that look promising, but need further investigation.

6.1 FRAMEWORK TECHNOLOGIES

Our initial assessment of the SLCSE framework is that the entity-relationship-attribute model incorporated into SLCSE is a good start toward capturing and organizing the large quantity of data generated by a project. The database schema, which models the data requirements of DOD-STD-2167A, is sufficient to support a full-scale development effort. The ability to generate the requisite documents directly from the database is a significant feature. A major limiting factor, however, appears to be the small number of legal data types that can be supported by the underlying relational database management system. For example, dataflow diagrams generated by a CASE tool are stored as text strings, either directly in the table field or, if too long for the table, stored in a text file within VMS, with a pointer to the file stored in the attribute field. In either case, the database does not reflect the true nature of that attribute.

A fruitful area for future research then, is in the area of Object-Oriented data modelling and the use of an object oriented database management system to implement it. Direct management of the data objects would eliminate the necessity of maintaining the two different modes of storage described above. Direct object management would also improve access controls to the objects within SLCSE. Currently, it is possible to directly access a SLCSE object stored in a VMS file, through VMS, without going through the SLCSE interface and controls. SLCSE utilizes VMS files to store objects created by SLCSE tools. This VMS file is stored in a subdirectory in the users account, and is accessible to the user by using the OBJECTS menu or by using VMS commands to access that directory directly. The user can then use a VMS tool such as EDT to edit that object file, and create a new version. This new version will not be known to SLCSE and not show up on listings of the object. This could potentially compromise the integrity of the data stored within SLCSE by permitting direct modification of objects without going through a controlled procedure. Object-oriented DBMS have strong data typing capabilities, are capable of managing arbitrarily typed objects, and provide complete protection by storing the object in a directory controlled by the database and not accessible to the user. Also, it would then be possible to directly map object-oriented designs into the SLCSE database.

The SLCSE database is capable of automatically generating, in hardcopy form, the 17 documents required by DOD-STD-2167A. Currently, DOD-STD-2167A does not address electronic distribution of the documentation it mandates. There is, however, another DOD-

sponsored effort, the Computer-aided Acquisition and Logistic Support (CALS) effort, which is developing common standards for transferability, storage, and accessibility of information in digital format. We believe that tools should be developed for SLCSE to automatically generate CALS compliant documentation, on electronic media specified in MIL-STD-1840A, Automated Interchange of Technical Information. This would entail the development of document type definitions (DTD), and translators to generate CALS compliant Standard Generalized Markup Language (SGML) documents, which also conform to DOD-STD-2167A requirements, directly from a SLCSE database. Projects would then be able to transfer project deliverables (CDRLs) electronically without necessarily generating the paper copies. Additionally, it would be useful to be able to preview documents on-line, in final printable format, prior to printing the hardcopy. This would reduce turnaround time, paper usage, and speed up internal reviews of documents prior to delivery. It would also permit incremental data transfer from a contractor site using SLCSE to a SPO site using SLCSE.

Another area for future development is the expansion and generalization of the import and export capabilities in SLCSE to simplify the integration of CASE and other tools that run on non-VAX/VMS computers, e.g., SUN workstations, Apple Macintoshes, and IBM PCs. For example, ICONIX PowerTools® and Excelerator/RTS® are two popular personal computer-based CASE tools in use at various ESD contractor sites. Other products, hosted on various computers, are also in use. Currently, only the VMS-based MentorCase® tools, Analyst/RT® and Designer® from Mentor Graphics Corporation, are integrated into SLCSE. Tight integration of a CASE tool will likely require the modification of the SLCSE database schema, holding open the possibility that different users of SLCSE, who have standardized on different CASE tools, will need to modify the schema in different, possibly mutually incompatible ways. This would reduce the transportability of data between sites, such as from contractor to Government.

SLCSE is currently designed around a centralized processor model, with user access provided via DEC VT-100 terminals or equivalent. SLCSE provides various levels of support for the integration of tools which reside on the VAX/VMS® configuration on which SLCSE is hosted, but only minimally supports the integration of tools which reside on other computer platforms. Since personal computers have come into widespread use at ESD and at its associated contractor community, and engineering workstations are being used more frequently, we believe the trend in the industry is heavily oriented toward a truly distributed client server architecture. The current SLCSE model then, would not make it attractive to use in many contractor organizations, unless it can be adapted to their environments.

By implementing a client server model in SLCSE, with possibly an object-oriented database, and generalizing the import and export capabilities, tools running on personal computers and workstations could not only store (import) their data in the SLCSE database, but also have the capability to retrieve (export) data in the appropriate formats, while still retaining a logically centralized, integrated database. This would also facilitate porting SLCSE database functionality to another host operating system/DBMS combination. As Unix becomes the predominant open operating system, many organizations have adopted it as the corporate

standard for supporting software development efforts, particularly on workstations. Also, many of the same database products are available on Unix systems as well as VMS. Making SLCSE available on Unix would broaden its potential customer base to many companies with a different infrastructure in place than VAX/VMS.

The database schema is the single most important feature of SLCSE. It makes possible the capture and analysis of the myriad information generated by a software project. We believe that to require the use of either the ShareBase Server or VAX/Rdb will be a limiting factor in acceptance of SLCSE. The Sharebase Server is designed as a special purpose box, containing a uni-processor accelerator and disk drives, which requires access to the host computer via Ethernet. Performance characteristics of Ethernet networks – heavily loaded networks often reach only 20% of theoretical capacity – and of the single processor may severely limit the maximum performance the Server can achieve on a large project. If one adds to that the cost of acquiring and maintaining a suitably large configuration, and it becomes a less viable alternative to acquiring and dedicating a large, fast VAX as a database server. The Server currently holds a very small share of the database market, and it is unlikely to improve its market position given the advances being made in both software-only DBMS implementations and the research being done on parallel processor architectures for database machines.

VAX/Rdb currently holds a 20% share of the VAX/VMS based DBMS market, while Oracle® holds another 20%. Other relational and non-relational DBMS systems comprise the remainder. Companies that have already invested in a relational DBMS will be unlikely to want to switch to VAX/Rdb. Even though VAX/Rdb is now distributed free with VMS, SLCSE still requires the SmartStar software to interface to the database. SmartStar licenses are scaled to the size of the VAX to which it is being licensed, which can be a considerable extra expense. So to enhance the prospects of SLCSE being adopted by a site that already has a different database system in place, consideration should be given to porting SLCSE to other relational database systems which are in use by contractors, e.g., Oracle, Sybase®, etc. We believe that making the schema available on other DBMSs, with improved import/export capability, will permit organizations with development environments based around another DBMS to install a customized version of SLCSE.

Another basic improvement that should be considered is the addition of color output to the screen displays, which can be used effectively to highlight important or exceptional information. For example, when using project management software, PERT charts, showing CDRLs delivered, delivery dates, task start and end dates, and schedule slips can be color coded to enhance readability and highlight problem spots. Trend data from management metrics tools can be displayed in green for example, with abnormal statistics displayed in a contrasting yellow or red, depending upon severity. During software development, tools such as Ada Test and Verification System (ATVS), AnalyzER, and ALICIA could provide color coded displays to highlight hot spots in analyzed code, inconsistencies in the database, traceability threads through the database. With low-cost color terminals becoming readily available as replacements for VT-100 terminals, and color capable personal computers already in widespread use, we believe further enhancement in this area could be very fruitful.

The rule-based process support should be extended and generalized to operate on entities, activities and time-critical events as well as tools. Trigger mechanisms can be provided to track schedules of deliverable documents, code, technical and management reviews to issue notifications a set period of time prior to the event, and to alert project management that a delivery has slipped. Entities and source code units could be tracked to provide notification to other users when updates are made. Further work in the area of the Knowledge-Based Software Assistant (KBSA) program should be done only as a continuing research effort and kept separate and distinct from the productization efforts discussed in section 5. Many of the recommended improvements to SLCSE made in this report will establish a better platform to incorporate knowledge-based enhancements. A dynamically modifiable object-oriented database schema, simplified tool integration procedures, and better graphical user interface (see next section) are all required to be able to effectively integrate knowledge-based development tools. If SLCSE is to be successfully transitioned into ESD and contractor shops, the near term focus should be on the conventional improvements discussed above and establishing an effective technology transition program as discussed in section 4.

6.2 USER INTERFACE CAPABILITIES

As already discussed, the primary access to SLCSE is through a DEC VT-100 compatible terminal or terminal emulator. The user interface presented, however, is not quite the standard, character-based, command line oriented interface that a VT-100 terminal implies. SLCSE does provide a menu-driven, window-oriented interface that looks very much like a standard Macintosh interface. We found it to be intuitive and easy to use. For contractor and government sites with VT 100/220s as the standard desktop terminal, this interface should be quite acceptable. We did notice a certain awkwardness in using arrow keys and keypad and return keys in various combinations to navigate and select items on the screen that a pointing device, such as a mouse or trackball, would alleviate.

We have found, however, that the trend in industry is moving away from the character-based class of terminals represented by the VT-100. We have observed a concerted push by the major computer manufacturers to incorporate windowing standards requiring bit-mapped graphics terminals and pointing devices as standard equipment. The X Window System, in fact, has rapidly been adopted as the de facto standard. Because of this trend, we recommend that an X Window System interface be developed for SLCSE. This would permit the use of X terminals, workstations, and personal computers, with bit-mapped screens and pointing devices to remotely access SLCSE, and still take advantage of the advanced built-in windowing and graphics capabilities.

Another user interface, one that SLCSE already resembles, is the proprietary Macintosh windowing system. This is accomplished by using the GRC-developed windowing software, WINNIE. In using a Macintosh to access SLCSE, terminal emulation software, such as Macterminal, must be used to make the Macintosh respond as a VT-100 to SLCSE. Thus, a Macintosh look and feel is provided, but at the cost of running emulation software on both the VAX and the Macintosh, and sacrificing the use of the mouse. This raises the issue of a government sponsored product using the "look and feel" of a proprietary interface, perhaps violating Apple Computers' copyright.

SLCSE also enables data generated by selected Macintosh applications to be integrated into the SLCSE database. Project management functions, using Micro Planner®, outlining and presentation functions, using More II®, are already available. We believe further development expanding this integration is warranted, given the capabilities and tools available on Macintosh II class computers. RADC, under the SLCSE Project Management System (SPMS) contract is currently integrating MacProject® II, Microsoft® Excel, and Microsoft Word into the SLCSE project management system. The Macintosh is utilized as a DECnet® client, and directly accesses the built-in graphics and mouse via the toolbox. The user interface is run directly on the Macintosh and tools are invoked on either computer, as appropriate, with all the data being stored in the SLCSE database. With the client server architecture, data can be retrieved from the database and manipulated by the Macintosh-based tools while maintaining the data under configuration control.

6.3 SOFTWARE DEVELOPMENT TOOLS

The standard set of software development tools currently available in SLCSE mainly support the detailed design, coding, and testing phases of the life cycle. Tools in these phases are numerous, well developed, and readily available from many vendors for the VAX/VMS environment. The tools available for the initial phases of the life cycle, concept formulation, and requirements analysis, appear to be fewer in number and also less well developed. Based upon our examination of the documentation for ALICIA, and the brief demonstration of it at the course, we believe it is the kind of tool that is necessary, even vital, on large software developments, and even more important for maintenance. We would recommend that further research and development be pursued in this area. We feel, however, that requiring a VAXstation II/GPX, with VAX Workstation Software (VWS)®, and the Graphical Kernel System (GKS) providing the graphics support, to host ALICIA will severely limit its availability. This is due partly to the different hardware requirements, and partly due to DEC transitioning to DECwindows®, a DEC implementation of the X-Window System as the primary means of providing a graphical user interface on its workstations. Thus, a site with SLCSE installed on a VAX system would also have to obtain an appropriately configured VAXstation® to run ALICIA. A large project might have to acquire a number of VAXstations to support the number of project and task managers needing access to ALICIA. This hardware would be an additional cost burden for organizations on limited budgets which are paying for the large central VAX computer and any personal computers being acquired. For sites at USAF Logistics commands, which may only have VAXes and VT-100 class terminals for software development, it would prevent them from using ALICIA at all. Yet, an impact analysis tool would be of major benefit to a maintenance organization. Requiring only an X interface to SLCSE would enable personal computers, running an X package, to be used to run ALICIA. Also, X Window System terminals, which are now available from DEC (VT-1200), as well as other vendors, will most likely become the de facto terminal type in the future and should be easier, and much less costly, to acquire than VAXstations.

We believe enhancing ALICIA to run with an X-Window System interface would substantially improve its availability, by permitting it to be run on any VAX using either an

X terminal or a personal computer with an X interface. Either of these alternatives is much less expensive than a properly configured VAXstation.

On-line visualization of the entity structures created would be very useful when populating a database. When a requirement is partitioned into multiple functions, each of which is further partitioned, being able to see the tree structure graphically on-line would simplify the completeness and verification checking. Similarly, graphically displaying other entity relationships, e.g., CSCI_Capability Utilizes Required_Data_Element, will simplify the verification checking of non-hierarchical relationships among entities. Being able to generate hard copy output of these graphs would also be desirable. The DGL for the SRS and SDD could be modified to automatically include such graphs in the hard-copy documentation generated by SLCSE, providing a model for tailoring DOD-STD-2167A data item descriptions. These types of graphs would also be useful at the code level to provide structure charts generated automatically from the database.

A great deal of government attention is being given to software quality metrics. The data collected in the SLCSE database appears to be sufficient, and of a fine enough granularity, to be used as input to tools that can generate software quality metrics and software management metrics, and create the appropriate reports through the documentation generation capability. We believe further tool development in this area is warranted. One approach might be to develop the export capability further to search the database and download the appropriate data into spreadsheet and presentation tools residing on either the VAX or on personal computers. Microsoft Excel, for example, is widely used by managers to plan and track budgeted and actual costs and labor hours expended for individual tasks within a project. The built-in graphing functions can then be used to create charts to illustrate progress, do trend analyses, and highlight problem areas. This is partially accomplished via the SLCSE Project Management System (SPMS) contract.

Another aspect of program management that we believe needs to be considered is the use, by many organizations, of stand-alone project management tools such as Artemus. In the general SLCSE project management tool, the SPMS, consideration should be given to ensuring that the data requirements and schema modifications that may need to be made are compatible with data available from these other tools. By doing so, it permits the possibility of integrating these tools into SLCSE without further schema modification.

Cost estimation tools are often used by SPOs to determine a cost estimate before Request for Proposals are issued to industry. Such tools, particularly if they are government-owned, could be incorporated into SLCSE and provided as part of the basic system.

Documentation and source code analysis comprise much of the work a SPO does during a contract. Gnu Emacs, an advanced editor which is available as "free software" and is widely used in the industrial and research community, should be included in SLCSE as a basic element of the tool suite. The ability to open multiple buffers simultaneously and its extensive search capabilities would provide users a better code and documentation analysis tool than currently exists in SLCSE.

Source code analysis by SPOs also includes "reverse engineering" the code to extract the code structure and measure code complexity and performance. The government owned

version of ATVS should be included as a basic part of the SLCSE tool suite. This would give users the opportunity to use the tool and see whether it provided the specific functionality needed for that project. They then could upgrade, at additional expense, to the commercial version from GRC as project needs dictate.

In large software development efforts, much of the effort expended is related to travel, meetings ,and document preparation and production. Providing support in SLCSE for secure teleconferencing capabilities and multimedia electronic mail could substantially reduce the need for travel and face-to-face meetings. Adapting current tools, or developing new tools, to support on-line collaborative work on requirements analysis, design, and document preparation could substantially reduce labor hours expended in travel, meetings, reviews, etc.

SECTION 7

SUMMARY

The Software Life Cycle Support Environment is a modern software engineering framework incorporating many of the tools and technologies necessary to provide a powerful and robust software development environment. It utilizes a significant number of commercially available products to minimize the amount of custom development required to provide the capabilities expected in a modern environment.

SLCSE development has reached the point where, with the additional capabilities specified in sections 3 and 6, a focused productization effort described in section 5, and a concerted effort at technology transfer, described in section 4, it could be transitioned to ESD and used initially on small to medium-scale software acquisition and development programs. As more experience is gained with SLCSE, further tailoring and enhancements will be identified and incorporated. Additional tools will be integrated into SLCSE. The database schema will be expanded to support additional design representations.

Because SLCSE is a framework, a longer period of transition should be expected. Unlike an individual CASE tool, which could be acquired, tested, and used by only one person and, if found deficient, replaced with only a minimal impact on the development, SLCSE must be adopted by an entire team or project to be beneficial. This will naturally cause a SPO or contractor's project manager to proceed cautiously before fully committing to its use.

SLCSE has the potential to improve the way ESD and its contractors manage and develop software. To realize that potential, future SLCSE development and support must be managed in the same manner as a commercial product.

APPENDIX

LIST OF TOOLS AVAILABLE FOR SLCSE

Tools Developed Specifically for SLCSE:

Requirements Tool	Creates, modifies, and deletes entities, relationships, and attributes in the System Requirements and Software Requirements subschemas.
Design Tool	Creates, modifies, and deletes entities, relationships, and attributes in the Software Design subschema.
Test_Manager	Creates, modifies, and deletes entities, relationships, and attributes in the Test subschema.
AnalyzER	Entity/Relationship analysis and reporting tool.
ModifyER	A general purpose database tool used to access subschemas, entities, relationships, and attributes.
ReportER	A general purpose database report generator.
VerifyER	A general purpose database consistency checker.
BaselinER	Creates and modifies configurations, baselines configurations, and generates reports describing the contents of configurations.
EditER	A form-based database access utility program used by several tools to create, modify, and delete entities, relationships, and attributes.
Docgen_2167A	Creates DOD-STD-2167A compliant documents from the database.
Docgen_Report	Creates custom reports from the database.
SDF_Create	Creates a Software Development Folder (SDF) and associates it with a specific CSCI, CSC, or CSU entity instance.
SDF_Delete	Deletes a Software Development Folder (SDF) and all of its contained files associated with a specific CSCI, CSC, or CSU entity instance.

SDL Compiler	Processes schema specifications written in Schema Definition Language (SDL) and produces the SQL necessary to create the relational database tables which implement the SLCSE database model.
SDL_Convert	Translates a subschema specification written in SDL into a form acceptable to AnalyzER.
Get_File	Provides controlled access to source and PDL files under SLCSE configuration management control (baselined).
Put_File	Returns files acquired via Get_File to SLCSE database and placed, once again, under database control.
Add_File	Inserts new files into the SLCSE database and places them under configuration control.
Import	Copies files from the external VMS system into SLCSE.
Export	Copies files from the SLCSE Objects window to the external VMS system or to a Software Development Folder.
Micro_Import	Populates Project Management subschema with information extracted from the Macintosh-resident tools, MicroPlanner Plus, and More II.

Problem Change Report Processor (PCRP) Populates and modifies entities and relationships in subschemas relevant to problem change reporting and tracking.

SLCSE Environment Manager (SEM) Provides SLCSE system administration functions; defines projects, tools available, and legal users and assigns user roles; defines computer resources available; defines rule-based methodology to be followed.

TexPrint	A DCL command file used in conjunction with LaTeX and TeX. It converts the device independent output file generated by LaTeX into a device dependent file and outputs that file to a user-specified device.

GRC Proprietary Products Incorporated into SLCSE:

WINNIE	Interactive tool for constructing window-oriented interactive user interfaces for VT-100 type terminals.
MOO	Menu Operations Organizer used in conjunction with WINNIE.

SmartStar Proprietary Products Incorporated into SLCSE:

SDesign	Designs and creates applications for a relational database.
SQL	An interpreter for the Structured Query Language (SQL).
SQuery	Executes database applications created by SDesign.
SReport	Creates reports from the database applications created by SDesign.

Public Domain Products Incorporated into SLCSE:

Kermit	A public domain file transfer program.
LaTeX	A document formatting system built on top of the public domain TeX document formatting system.

Macintosh-Based COTS Products Usable with SLCSE:

MacProject II	A general purpose project management tool from Claris® Corp.
MicroPlanner +	A general purpose project management tool from MicroPlanning International, Inc.
Microsoft Excel	A general purpose spreadsheet tool from Microsoft Corp.
Microsoft Word	A general purpose word processing tool from Microsoft Corp.
MORE II	An outlining and presentation tool from Symantec Corp.

SLCSE-Independent Products Usable in SLCSE:

ATVS	Ada Test and Verification System (from GRC).
AMS	Automated Measurement System (from Harris Corp.).
ALICIA	Automated Life Cycle Impact Analysis (from SPS).
ADL	Ada-Based Design Language (from SPS).
SDDL	Software Design and Documentation Language (from K2 Associates).
CAVS	COBOL Automated Verification System (from GRC).
Jovial J73	A compiler for Jovial J73 (from Proprietary Software Systems, Inc.).
J73AVS	A Jovial J73 Automated Verification System (from GRC).
PIGMY	An Interactive Graphics Utility (from GRC).
MentorCase	Computer-Aided Software Engineering tools (from Mentor Graphics).
RXVP80	A Fortran Automated Verification System (from GRC).

VAX/VMS Products from DEC Usable in SLCSE:

VAX/Ada Compiler
VAX/Ada Compilation System (ACS)
VAX-11 COBOL Compiler
VAX FORTRAN Compiler
Digital Standard Runoff. A document formatting system
VAX/Language Sensitive Editor (LSE)
VAX/EDT Editor
VAX/EVE Editor
VAX MACRO Assembler

VAX/VMS DCL Utilities Usable in SLCSE:

Copy
CPU
Create
Delete
Directory
Mail
Print
Purge
Rename
Run
Type

GLOSSARY

ALICIA	Automated Life Cycle Impact Analysis
ALS	Ada Language System
AFSC	Air Force Systems Command
APSE	Ada Programming Support Environment
ATVS	Ada Test and Verification System
AWDS	Automated Weather Distribution System
CALS	Computer-Aided Acquisition and Logistic Support
CASE	Computer Aided Software Engineering
CCES	Command Center Evaluation System
CCPDS-R	Command Center Processing & Display System-Replacement
CDRL	Contract Data Requirements List
COTS	Commercial-Off-The-Shelf
C3I	Command, Control, Communications and Intelligence
DBMS	Database Management System
DCL	Digital Command Language
DEC	Digital Equipment Corporation
DGL	Document Generation Language
DOD	Department of Defense
DTD	Document Type Definitions
EDT	VAX/VMS - hosted text editor
ESD	Electronic Systems Division
GFE	Government Furnished Equipment
GKS	Graphical Kernel System
GRC	The General Research Corporation
IADS	Icelandic Air Defense System
IBM	International Business Machines, Inc.
IR&D	Independent Research and Development
JSTARS	Joint Surveillance Target Attack Radar System
LAN	Local Area Network
MAC/IPS	Military Airlift Command/Information Processing System
MCC	Microelectronics and Computer Technology Corporation
MCCS	Mission Critical Computer Software
NASA	National Aeronautics and Space Administration
P3I	Preplanned Product Improvement
PERT	Program Evaluation and Review Technique
POSIX	Portable Operating System Interface
RADC	Rome Air Development Center
RAPID	Reusable Ada Packages for Information System Development
RSIP	Radar Software Improvement Program
SCIS	Survivable Communications Integration System
SDD	Software Design Document
SDME	Software Development and Maintenance Environment
SEI	Software Engineering Institute
SEP	Software Engineering Prototype
SLCSE	Software Life Cycle Support Environment

SGML	Standard Generalized Markup Language
SPO	System Program Office
SPS	Software Productivity Solutions, Inc.
SRS	Software Requirements Specification
STARS	Software Technology for Adaptable Reliable Systems
STP	Software Technology Program
SQL	Structured Query Language
TO&P	Technical Objectives and Plans
USAF	United States Air Force
VAX	A computer manufactured by DEC
VMS	Virtual Memory System
VAX/Rdb	VAX/VMS-hosted relational database from DEC
VWS	VAX Workstation Software
WIS	WWMCCS Information System
WWMCCS	Worldwide Military Command and Control System

TRADEMARKS

UNIX and AT&T are registered trademarks of American Telephone and Telegraph.

Apple, Macintosh, and A/UX are registered trademarks licensed to Apple Computers, Inc.

IBM and IBM PC are registered trademarks of International Business Machines Corporation.

XENIX is a registered trademark of Microsoft Corporation.

Postscript is a registered trademark of Adobe Systems, Inc.

ICONIX and Power Tools are registered trademarks of ICONIX Software Engineering, Inc.

Excelerator/RTS is a registered trademark of Index Technology Corporation.

Mentor Case, Analyst/RT and Designer are registered trademarks of Mentor Graphics Corporation.

DEC, VAX, VMS, VT-100, Rdb, VWS, Vaxstation, DEC windows, and DECnet are registered trademarks of Digital Equipment Corporation.

Oracle is a registered trademark of ORACLE Corporation.

X Window System is a registered trademark of Massachusetts Institute of Technology.

UTS is a registered trademark of Amdahl Computers, Inc.

SUN is a registered trademark of SUN Microsystems, Inc.

SYBASE is a registered trademark of SYBASE Corporation.

Micro Planner is a registered trademark of Micro Planning, Inc.

MORE II is a registered trademark of Symantec Corporation.

Ethernet is a registered trademark of XEROX Corporation.

Claris and MacProject are registered trademarks of Claris Corporation.

LIST OF REFERENCES

E&V Guidebook, Version 2.0, 30 September 1989, Technical Report. US Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB.

E&V Reference Manual, 30 September 1989, Version 2.0, Technical Report. US Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB.

Earl, A., January 30-31, 1990, *A Reference Model for Computer Aided Software Engineering Environments: A Conceptual Viewpoint.* International CASE Architectures and Standards Course, National Technological University.

Bryan, M., 1988, *SGML: An Author's Guide to the Standard Generalized Markup Language,* Addison-Wesley Publishing Company.

Grau, J. K., Software Productivity Solutions Inc., *SLCSE's Relationship to CALS,* Briefing presented at CALS Expo, '89, Orlando, FL.

Jones, T. C., August 1990, Seminar on Productivity, Boston Computer Society.

Software Life Cycle Support Environment, August 1989, *Software Programmer's Manual,* General Research Corp.

Software Life Cycle Support Environment, August 1989, *Software User's Manual, Vol. I,* General Research Corp.

Software Life Cycle Support Environment, August 1989, *Software User's Manual, Vol. II: SLCSE Environment Manager,* General Research Corp.

Software Life Cycle Support Environment, October 1989, *DOD-STD-2167A Schema,* General Research Corp.

Software Life Cycle Support Environment, October 1989, *Document Generation Language (DGL) for the Software Design Document (SDD),* Software Productivity Solutions, Inc.

Software Life Cycle Support Environment, February 1990, *On-the-Job Training Course, Volume I: Management Overview,* General Research Corp.

Software Life Cycle Support Environment, February 1990, *On-the-Job Training Course, Volume II: User Orientation,* General Research Corp.

Software Life Cycle Support Environment, February 1990, *On-the-Job Training Course, Volume III: Support Orientation,* General Research Corp.

Software Life Cycle Support Environment, February 1990, *On-the-Job Training Course, Appendices I - VIII,* General Research Corp.

Automated Interchange of Technical Information, 22 December 1987, Department of Defense, MIL-STD-1840A.

Defense System Software Development Standard, 29 February 1988, Department of Defense, DOD-STD-2167A.

Evaluation of Ada Environments, March 1987, SEI Technical Report, CMU/SEI-87-TR-1, ESD-TR-87-101.

Babcock, J., D. Laszlo, A. Belady, and N. C. Gore, March 1990, *"The Evolution of Technology Transfer at MCC's Software Technology Program: From Didactic to Dialectic"* MCC TR STP-404–89, Proceedings of 12th International Conference on Software Engineering, Nice, France.

Maher, J. H, Jr., June 1990,*"Planning for Technology Transition"* Seventh WAdaS Tutorials.

Schaefer, A. L., June 1989, *CALS - The Computer-Aided Acquisition and Logistic Support Initiative and its Implications for Software Acquisition,* MTR-10584, The MITRE Corporation, Bedford, MA.

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.