AD-A261 042

RL-TR-92-249
Final Technical Report
October 1992

DTIC
ELECTE
FEB 2 3 1993
S C D

SOFTWARE LIFECYCLE SUPPORT ENVIRONMENT (SLCSE) KNOWLEDGE-BASED ENHANCEMENTS

Honeywell Systems and Research Center

Bhavesh Damania

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

93-03709

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

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

Although this report references *limited documents listed on page 4, no limited information has been extracted. Distribution for limited documents is:

USGO agencies and private individuals or enterprises eligible to obtain export-controlled technical data iaw regulations implementing 10 U.S.C. 104c; Aug 88. Other requests RL (C3CB) 525 Brooks Rd, Griffiss AFB NY 13441-4505.

RL-TR-92-249 has been reviewed and is approved for publication.

APPROVED: Deborah A. Cerino

DEBORAH A. CERINO
Project Engineer

FOR THE COMMANDER: John A. Graniero

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

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

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

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1992	Final Sep 89 - Jun 92

4. TITLE AND SUBTITLE
SOFTWARE LIFECYCLE SUPPORT ENVIRONMENT (SLCSE)
KNOWLEDGE-BASED ENHANCEMENTS

5. FUNDING NUMBERS
C - F30602-89-C-0199
PE - 63728F
PR - 2527
TA - 02
WU - 22

6. AUTHOR(S)
Bhavesh Damania

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Honeywell Systems and Research Center
3660 Technology Drive
Minneapolis MN 55418

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CB)
525 Brooks Rd
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-92-249

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Deborah A. Cerino/C3CB/(315) 330-2054

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

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This report summarizes the assessment of the Rome Laboratory Software Life Cycle Support Environment (SLCSE) for potential application of knowledge-based technology. The work performed under this contract included three major tasks: Task 1: Analyze SLCSE; Task 2: Specify Approaches for inserting knowledge-based techniques; Task 3: Develop a Software Requirements Specification and Software Development Plan for incorporating near-term knowledge-based enhancements into the SLCSE. This report provides an overview of these three tasks, and concludes with areas for further evaluation.

14. SUBJECT TERMS
Knowledge-Bases, Software Engineering Environments

15. NUMBER OF PAGES
48

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	UL

Accesion For
NTIS CRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

Contents

DTIC QUALITY INSPECTED 3

1 Scope **1**

1.1 Identification 1

1.2 Background 1

1.3 Overview 3

2 Applicable Documents **4**

2.1 Government Documents 4

2.2 *Non-Government Documents* 5

3 SLCSE Analyses **7**

4 Requirements and Alternate Approaches **13**

4.1 Infrastructure Enhancements 13

4.1.1 Infrastructure Capabilities and Requirements 14

4.1.1.1 Data Modeling and Process Modeling 14

4.1.1.1.1 Data Model 14

4.1.1.1.2 Execution Model 15

4.1.1.1.3 Data Modeling Language 16

4.1.1.1.4 Process Model 16

4.1.1.2 Data Management Services and Process Enactment Services 17

4.1.1.3 Database Schemas and Processes 19

4.2 Infrastructure Enhancement Alternate Approaches 20

4.2.1 Tasks Involved in Infrastructure Enhancement 20

4.2.2 Alternate Approaches to Infrastructure Enhancement 21

4.2.2.1 Data Model Selection 21
4.2.2.2 Process Model Selection 22
4.2.2.3 Infrastructure Enhancement Approaches 23
4.2.2.3.1 Custom Object/Logic Base 24
4.2.2.3.2 Pre-existing Rule Based System 24
4.2.2.3.3 Pre-existing Object Base 25
4.2.2.3.4 Evolve Existing SLCSE 25
4.3 Toolset Enhancements 26
4.3.1 General Toolset Enhancements 26
4.3.2 Specific Toolset Enhancements 27
4.4 User Interface Enhancements 31

5 Conclusions **33**
5.1 Areas for Further Evaluation 33
5.2 Lessons Learned 35

List of Figures

3.1 Capabilities of a KB Framework 11

4.1 Estimated Level of Effort for the Four Approaches in Person-Months 26

4.2 Tool Enhancements Relationships to Enabling Technologies 28

Chapter 1

Scope

1.1 Identification

This document is the Final Report of *Knowledge-Based Extensions to the Software Life Cycle Support Environment (KBE-SLCSE),* Rome Laboratory contract F30602-89-C-0199. This report describes the conclusions of the work performed under that contract.

1.2 Background

The work performed under this contract included three major tasks: Task 1, SLCSE Analyses; Task 2, Specify Alternate Approaches; Task 3, develop a Software Requirements Specification and Software Development Plan.

Task 1 (SLCSE Analyses [KBE1]) analyzed the SLCSE architecture and existing capabilities and identified knowledge-based technologies and necessary associated framework technologies that should be incorporated within SLCSE to better support the software engineering process and environment maintenance. Some insertions will improve the support that SLCSE currently offers; others will provide functionality (particularly knowledge-based functionality) not currently in the SLCSE. The accomplishment of this task is the identification of the key infrastructure services needed to support an enhanced SLCSE toolset and user interface.

Task 2 (Specify Alternate Approaches, [KBE2]) developed a plan for achieving the incorporation of knowledge-based technology into the SLCSE. It describes the required technology, identifies the targeted SLCSE components, describes the user familiarization requirements, and identifies the necessary hardware/software or equipment interfacing requirements for the Rome Software Engineering Laboratory. The report specified an incremental approach to implementing the identified capabilities, including identifying the level of effort and corresponding time span required. Each technology insertion was prioritized according to highest potential payoffs. The accomplishment of this task is the requirements and capabilities spec-

ification for KBE-SLCSE and the alternate approaches in accomplishing the enhancements.

Task 3 (Requirements) develops a detailed Software Requirements Specification [KBE4] and Software Development Plan [KBE3] for incorporating selected near-term knowledge-based enhancements. The requirements specification established the requirements for the knowledge-base (KBE) SLCSE computer software conponents (CSCs) identified as database, user interface, and toolset. The development plan specifies the plan for the KBE-SLCSE CSCs identified as database, Knowledge Definition Language (KDL), user interface, and toolset. The emphasis on these reports was to specify all possible requirements and capabilities and then incorporate only selected near-term enhancements in the Software Development Plan. Task 2 identified a selected set of requirements which can be accomplished in 1-3 years timeframe. The development plan assumed an evolutionary approach to inserting the enhancements.

The final report summarizes the results of the work performed under the contract. It describes at a high level

- The program objectives
- The accomplishments and the time spent on each task
- The analysis, studies and investigations which were performed on the framework components
- The conclusions which were drawn and the reasons for drawing those conclusions
- The areas for future investigations
- Issues for implementing the enhancements
- The lessons learned

The main purpose of KBE-SLCSE is to provide intelligent assistance within an integrated software development environment for using a comprehensive set of software development tools supporting DoD-STD-2167A development methodology, managing and supporting project specific software development data, supporting the management of a software development process; and providing a consistent and a uniform user interface to tools. KBE-SLCSE will support these functions in a way that ensures that KBE-SLCSE is:

- Adaptable to each installation;
- Distributed, thereby making it available on a network of hosts to maximize performance and throughput;
- Extensible in supporting new tools, new project-specific methodologies, and new hardware and software technologies;

- Usable by providing the performance and robustness needed to develop large-scale software development projects;
- Functional by providing the functions needed to support software developments.

1.3 Overview

The report is organized as follows:

- Chapter 2 lists applicable government and nongovernment documents.
- Chapter 3 summarizes the results of Task 1 which analyzed the SLCSE architecture and capabilities and identified the knowledge-based technologies to be inserted in SLCSE.
- Chapter 4 summarizes the results of Task 2 and Task 3 specifying the requirements and capabilities for KBE-SLCSE and the alternate approaches to enhancements.

Chapter 2

Applicable Documents

2.1 Government Documents

This technical report references or incorporates portions of the following:

* [SLCSE1] *SLCSE Software Requirements Specification*, SLCSE Technical Staff, July 1988, General Research Corporation CR-3-1537.

* [SLCSE2] *SLCSE Software Top Level Design Document*, SLCSE Technical Staff, August 1988, General Research Corporation CR-3-1537.

[IKB1] *Task 1: Identify High-Payoff Life Cycle Activities*, revised interim report, July 21, 1988, IIT Research Institute.

[IKB2] *Task 1: Identify High-Payoff Life Cycle Activities Appendix A – Annotated Bibliography*, revised interim report, April 29, 1988, IIT Research Institute.

[IKB3] *Task 2: Analysis of the Software Life Cycle Support Environment SLCSE*, interim report, January 17, 1989, IIT Research Institute.

[IKB4] *Task 3: Five Year Plan for Incorporating Knowledge-Based Technology into the SLCSE*, revised interim report, February 1, 1990, IIT Research Institute.

[IKB5] *System Specification for the Knowledge-Based Software Life Cycle Support Environment*, January 10, 1990, IIT Research Institute.

[IKB6] *Integration of Knowledge-Based and Conventional Software Tools Final Report*, May 9, 1990, IIT Research Institute.

[KBE1] *Task 1 Report: SLCSE Analyses*, May 1991, Honeywell Systems and Research Center.

[KBE2] *Task 2 Report: Specify Alternate Analyses*, August 1991, Honeywell Systems and Research Center.

[KBE3] *Task 3 Report: Development Plan*, December 1991, Honeywell Systems and Research Center.

[KBE4] *Task 4 Report: Requirements Specification*, December 1991, Honeywell Systems and Research Center.

2.2 Non-Government Documents

[BERN] P. A. Bernstein, "Database System Support for Software Engineering." *Proceedings of the 9th International Conference on Software Engineering*, 1987.

[CFI] CFI Architecture TSC Extension Language Working Group Minutes of August 23-0245 1990 meeting; prepared by A. Remy Lalan, Mentor Graphics Corp.

[DUH] J. Duhl, C. Damon. "A Performance Comparison of Object and Relational Databases Using the Sun Benchmark." *OOPSLA '88 Proceedings*, SigPlan Notices, 23-11, November 1988.

[Lar] J. Larson. "Interactive Software Tools for Building Interactive User Interface." *Yourdon Prentice,* Englewood Cliffs, New Jersey 07632, 1992.

[MAI] D. Maier. "Why Object-Oriented Databases Can Succeed Where Others Have Failed." *Proceedings of the 1986 International Workshop on Object-Oriented Database Systems*, 1986.

[MAY] D. Mayhew. "Principles and Guidelines in Software User Interface Design." *Prentice Hall,* Englewood Cliffs, New Jersey 07632, 1992.

[IRDS] Information Resource Dictionary System (IRDS), 5 Apr 89. FIPS PUB 153.

[ORA] Oracle Corporation, SQL Forms Designer's Reference, Version 3.0." 1991.

[PCTE] Portable Common Tool Environment (PCTE) Abstract Specification. Standard ECMA-149. ISO/IEC JTC1/SC7.

[PMDB] M. H. Penedo, C. Shu, "Acquiring Experiences with the Modeling and Implementation of the Project Life-cycle Process – The PMDB Work," Technical Report Arcadia-TRW-90-001, 1990.

[PRE] W. J. Premerlani, M. R. Blaha, J. E. Rumbaugh, T. A. Varwig. "An Object-Oriented Relational Database." *Communications of the ACM*, 33-11, November 1990.

[SMI] Smith, K. E., Zdonik, S. B. "Intermedia: A Case-Study of the Differences Between Relational and Object-Oriented Database Systems." *OOPSLA '87 Conference Proceedings*, SigPlan Notices, 22-12, December 1987.

Chapter 3

SLCSE Analyses

The objective of Task 1, *SLCSE Analyses*, was to analyze the SLCSE architecture and capabilities and identified knowledge-based (KB) technologies for incorporation within SLCSE in order to better support the software process and environment maintenance. Task 1 also looked at general approaches for inserting this technology. A total of 35% of the project effort was spent on Task 1. The investment was very significant since it resulted in an in-depth analysis of SLCSE and eased the successful completion of other tasks.

The Task 1 analyses examined:

- The SLCSE *toolset*, including both the developmental and the non-developmental tools which facilitate performing the life-cycle activities. The toolset analyzed includes: environment management; the elimination of tools superseded by enhanced framework functionality; the addition of tools to support maintaining the SLCSE knowledge-base; addition of KB tools to assist in evolving the SLCSE toolset and database/knowledge-base; the impact of framework changes on existing tools, and the approaches to minimize adverse impact.

- The SLCSE *project database*, including: database adequacy to support KB technology, constraint management capability, database enhancements to support DOD-STD-2167A document generation, necessary modifications to SLCSE's entity-relationship (E-R) data model, and its implementation: the schema definition language (SDL) compiler, and the entity-relationship interface (ERIF) package for permitting directly coupled tools access to the data, and the commands for permitting indirectly coupled tools access to the data. An integrated set of data management services including long duration transaction management support, fine-grained access control mechanisms, configuration and change management, audit trailing capability and other data administrative services.

- The SLCSE *user interface*, including: the applicability of KB technology to the existing user interface, including existing user help and training facilities; guidance/control of

what tools/capabilities are used, and at what times (focusing on testing, QA, and documentation tools); extension/revision of SLCSE's current windowing package.

- *general issues:* impact on resources and performance: CPU, memory, disk, response time; adequacy/completeness of SLCSE to support development of knowledge-based software for embedded or mission critical applications; evaluation of integrating COTS knowledge-based toolsets (expert system shells, etc.); benefits to SLCSE of KB insertions; adequacy of the hardware in the Rome Laboratory.

A requirements-driven analyses and evaluation was performed, based on SLCSE goals. Building on results from the *Integration of Knowledge-Based and Conventional Software Tools* task order, Rome Laboratory contract F30602-87-D-0094, the analyses was tailored to provide specific additional evaluations, including evaluations of impacts and benefits. The focus was particularly on knowledge-based technology to support environment management, tool integration, methodology support, and documentation development. The analyses was performed on the infrastructure, the toolset and the user interface.

Task 1 concluded the need for incorporating the following knowledge-based technology into SLCSE.

Domain-specific expert assistance. Based on knowledge about specific subdomains of software engineering, provide expert assistance targeted to particular life cycle activities and particular roles.

Capture/communication of project knowledge. Capture project knowledge during life cycle activities and communicate it to later life cycle activities. Most of the knowledge is captured in the background as a side effect of user actions; users can later access the knowledge by asking questions, requesting analyses, requesting evaluations, etc.

Error prevention and correction. Prevents actions (by users or processes) which would produce unrecoverable inconsistency; re-establish consistency after changes based on the constraints that should be maintained on objects in the project database. Provide remedial advice when errors or inconsistencies are detected.

Coordination of agents. Manage the sequencing and communication of user actions and tool processes as they operate on the project database, enforcing and facilitating the specified methodology; prevent actions (by users and processes) that would violate the methodology; weave tool invocations and user actions into higher-level operations that leverage the synergy among tools and activities.

User guidance. Present only legitimate choices (as defined by context and methodology) to users; track details for users; provide intelligent help sensitive to context and project state.

Opportunistic processing. If the results of a given operation are required (possibly transitively), and it is appropriate to perform the operation, then automatically initiate the operation; if the inputs of a given operation that would advance the project become available, and it is appropriate to perform the operation, then automatically initiate the operation.

Synthesizing objects from specifications. Synthesizes products from more abstract specifications of those products (e.g., relocatable object from source code, source code from program specification, etc.), where possible and appropriate. Synthesis or re-synthesis may occur automatically in response to changes in the project state.

To provide the above services, Task 1 arrived at the following conclusions:

- The strength of a framework lies in the integration and synergy it provides; this is even more true of KB frameworks. The highest payoff for insertion of KB technology is in framework-wide synergy among tools rather than through piecemeal insertion of KB capabilities into individual tools.
- A common representation (data model, knowledge representation scheme) of project knowledge and life cycle information is a necessity in the long term; this representation must support high level KB capabilities including operations, constraints, and rules. An approach that utilizes a common representation will allow a smoother integration of tools and be more easily extensible.
- The common representation must support multiple inference schemes for efficiency.
- Support for automated capture of system knowledge (system structure, design alternatives, system history, etc.) is a necessity in the long-term.
- The data model for a KB framework should be object-based. That is, it should be capable of defining the operations that describe the actions to be performed on the engineering objects. It should also encapsulate the state of those objects so that only the defined operations can access their state. These capabilities are necessary so that the framework can be active and can control the automated and manual actions of the engineering process.
- The data model should include a constraint language, that enables the intensional declarative description of engineering objects. This constraint language will allow the derivation of desired information from existing information, the automated maintenance of consistency in the face of change, and the opportunistic invocation of certain actions.
- The constraints placed on the operations should not be subvertible – all invocations of those operations should be required to obey the specified constraints.

- The constraint language should include special-case inferencing which makes it easy to express and efficient to perform commonly-occurring inferences.
- Expert tool support services should be provided – i.e., the ability to make inferential queries over the information in the project database/knowledge-base. These services may take the form of an inferential query language that serves as the data definition language/data manipulation language of the database/knowledge-base.
- The role of tools in such a framework will change. The environment becomes more "intelligent" and active as the the framework becomes more than a simple storage server that assumes that all knowledge is built into the tools. Rather, the objects within the framework take on a more "pro-active" role in the development process.

Task 1 arrived at the following six primary capabilities that aid in the creation of engineering environments:

- Object management services, provides the means for creating and managing a project knowledge-base to serve as a focal point for data management.
- User interface services, provides the means for tools to interact with the user in a uniform fashion, and support the user in effectively interacting with the environment.
- Tool and data integration services, provides the means for integrating the various tools and their data into a coherent, active environment. These capabilities would be provided by the Object/Logic Base and Knowledge Manager Subsystems.
- Knowledge capture and communication services, support the capture of abstract project knowledge in the knowledge-base, and its propagation from one activity to another.
- Methodology support services, provides the means for coordinating the interactions among tools and users to facilitate and enforce a consistent software methodology.
- Expert tool services, provide the means for inferencing and querying the knowledge base.

To adequately support KB tools, the primary framework capabilities require a group of secondary capabilities. Figure 3.1 illustrates how these secondary capabilities support the primary capabilities provided by a framework.

The major accomplishment of this task was the architecture driven analysis of SLCSE which resulted in the identification of the key infrastructure requirements and capabilities critical to supporting enhanced user interface and toolset.

The conclusions of this task is that the enhancements needed for the user interface and the toolset relied heavily on the infrastructure (data model and process model) capabilities. This conclusion was reached by identifying the services needed to implement the user interface and

Primary Capability	Supporting Capabilities
Object Management	Object Model (E-R, inheritance, operations) Object Constraint Management Operation Constraint Management Extensibility (Schema Evolution) CCM, Design Transactions, and Permanence Transparent Distribution Access Control
User Interface	Protocols and Toolkits Knowledge Communication (browsers, etc.) Intelligent Help/Tutoring Natural Language *User Modeling*
Knowledge Capture	Static Analysis Background Capture Intrusive Capture
Tool and Data Integration	Envelopes (including KB) Intelligent help for Data Shadowing, Intelligent help for Import/Export Adaptor Generators
Methodology Support	Operation Constraints Planning Process Modeling Assistance Process Enactment Assistance
Expert Tool Support	Inference and Query over Project KB

Figure 3.1: Capabilities of a KB Framework

the toolset enhancement and comparing them with the services provided by the enhanced infrastructure. A benefits analysis was performed to identify the insertion strategies which will result in the most significant pay-off. The infrastructure services and capabilities were identified as the major technical enhancement. The above conclusions set the technical direction of the work to be performed on the subsequent tasks. Since the infrastructure was deemed to be the most important architectural component, the technical emphasis of Task 2 was to specify the requirements and the capabilities for the infrastructure and evaluate alternate enhancement approaches.

There were three major areas which needed further investigations

- An in-depth analysis of the tools supporting the various phases of the software development life-cycle. The approach taken by this project was to discuss the general toolset enhancement technologies, the services of which can be used in constructing enhanced toolset.
- The integration of the life-cycle specific methodologies and notations. For example, in the design phase the framework needs to enforce and facilitate selected structured design methods including structure charts, data flow diagrams and/or Booch object diagrams. An in-depth analysis needed to be performed to study the integration of commercial-off-the-shelf tools and the infrastructure services that needs to be provided to support life-cycle specific representations and notations.
- The process model supporting the specification and enforcement of software development methodologies.

Chapter 4

Requirements and Alternate Approaches

Task 2 developed a plan for achieving the incorporation of knowledge-based technology into SLCSE. A total of 35% effort was spent on Task 2. The Task 2 report developed a comprehensive set of requirements which formed the basis for specifying Software Requirements which was performed under Task 3. Task 2 also estimated the level-of-effort involved in inserting an enhancement which formed the basis for specifying a Software Development Plan which was performed under Task 3. The technical approach taken was to first specify the capabilities and the requirements for KBE-SLCSE and then to evaluate alternate approaches and architectures for enhancing SLCSE to support these capabilities and requirements. The technical emphasis on Task 2 was to concentrate on the infrastructure enhancements which formed the basis for supporting the user interface and the toolset enhancements.

The SLCSE enhancements were studied in terms of

- Infrastructure Enhancements
- Toolset Enhancements
- User Interface Enhancements

4.1 Infrastructure Enhancements

The major effort undertaken in task 2 was to specify the capabilities and the requirements for the infrastructure and evaluating alternate approaches to enhancing the infrastructure to support these capabilities and the requirements.

The environment infrastructure or framework supplies the data and process modeling capabilities and the data and process management services. These capabilities collectively enable

the creation and management of a project database that act as a focal point for the capture and control of project information and for the representation and enactment of the software development methodology.

- Infrastructure Capabilities and Requirements.
- Infrastructure Enhancement Alternate Approaches.

4.1.1 Infrastructure Capabilities and Requirements

The infrastructure capabilities and requirements were specified in terms of

- Data Modeling and Process Modeling
- Data Management Services and Process Enactment Services
- Database Schemas and Processes

4.1.1.1 Data Modeling and Process Modeling

The data modeling capability provides the formal means for capturing and manipulating project and process knowledge, and it provides the basis for reasoning about the knowledge. The process modeling capability provides a rich and robust means for characterizing the software development process and supporting process variation.

The overall recommendation is to ensure that KBE-SLCSE minimally support the requirements identified by the Task 2 report for the data modeling capability. The enhancements to SLCSE to fulfill the requirements for the data model, the execution model and the data modeling language was concluded by Task 2 to be a high priority enhancement.

The data modeling capabilities and requirements were specified in terms of

- Data Model
- Execution Model
- Data Modeling Language

4.1.1.1.1 Data Model The data model which is the formal means for describing engineering objects is the basic building block in the framework.

The data model should include the following capabilities:

Expressive Data Model Supporting the representation of the structural, behavioral and dynamic aspects of the object.

Powerful and Adaptable Typing Support Supporting the ability to

- Handle a wide extensible range of data types,
- Represent one-to-one, one-to-many, many-to-many and inverse relationships among data objects,
- Specify operations applicable to various types,
- Specify integrity constraints and self-descriptive modeling.

Rules in Rulesets Supporting forward-chaining and backward-chaining over a "ruleset" (a collection of rules).

Rule-based Triggers Supporting specification and implementation of constraints with optimized implementation for derived-value, consistency-maintenance, opportunistic-processing and process-error prevention constraints.

4.1.1.1.2 Execution Model The execution model defines the semantics for the operations defined on the objects, the triggers defined on the objects and operations (including constraints), and the rule processing. The environment provides an execution engine (command interpreter) that implements this execution model.

The execution model should include the following capabilities:

Semantics for Invocation of Operations on Data Reads/Writes The execution model defines what the environment's execution engine (command executive) does when an operation is invoked or when data is accessed. It should be possible for an operation on an object to be a pre-existing tool, piece of code linked into the execution engine, or code interpreted by some interpreter.

Type-Based Resolution of Requests for Operations Supporting strong typing and polymorphic resolution on operation requests.

Semantics for Creation and Management of Objects Supporting services for assignment of unique identifier when objects are recreating and reclaiming storage space when object is deleted.

Control and Monitor Request Execution Support services for execution of function request in a distributed environment.

Event-Based and State-Based Trigger Evaluation and Execution Support services for evaluation and execution of triggers on object accesses and operation invocations.

Asynchronous/Synchronous Execution Support services for initiating, controlling and monitoring asynchronous/synchronous execution of operations.

Exception Handling Support flexible exception handling capability to cope with errors returned by tools.

Transaction Management Support robust and comprehensive database-level short-duration transaction.

Change Management Support services for cooperating with the change model to capture change and history and manage references to changing objects.

Extensibility/Adaptability/Cutover Support sophisticated and powerful incremental compilation/dynamic linking capability allowing process description to be customized, extended, adapted, and replaced dynamically.

Inferential Query Support engine(s) that may be invoked via reads (backward-chaining to acquire information) or writes (forward-chaining to derive consequences of new information).

Other Services Support distribution, access control, persistent processes, and process migration services.

4.1.1.1.3 Data Modeling Language The data modeling language provides the formal means for describing software engineering process in terms of classes and instances of agents, activities, and artifacts.

KDL provides the following capabilities

- A Data Definition Language (DDL) for defining application-specific data types.
- A Data Manipulation Language (DML) for locating, reading, and writing data.
- A host language for operating on data retrieved by the DML.
- An Activity Description Language (ADL) for defining activities ranging from tool invocations to life-cycle phases, and the constraints among them.
- An Activity Invocation Language (AIL) for initiating activities.

4.1.1.1.4 Process Model The process model provides a rich and robust means for characterizing the software development process and supporting process variations.

The process language will support the following expressible constructs.

- Support for standard flow constructs including conditional selection and repetition.
- Specification of access mechanisms for underlying database and system values.

- Specification of serial, parallel, preferred, and deferred activities.
- Specification of multi-user paradigms and interprocess communication.
- Support for embedded activity definition.
- Process definition extensibility and tailorability.
- Error handling capabilities.

Task 2 analyzed alternate approaches for defining and implementing a process.

- Graphical Representations. The graphical representations which were analyzed were data flow diagrams, finite state machines, petri nets, idef notations.
- Rule-Based Grammar. Rule-based grammar including inference-based production rules and simple production rules.
- Process Generator Generator. Process Generator Generator supports a meta language for defining process representation languages The various process generator generators which were analyzed included:
  - Parser generator generators (e.g., YACC, ICC);
  - Language rewriting systems (e.g., NEWYACC);
  - Graph rewriting systems (e.g., OPTRAN, TWIG);
  - User interface generators (e.g., X View);
  - Visual programming systems (e.g., KUIE, GARDEN);
  - Visual language system generators (e.g., SIL-ICON).

4.1.1.2 Data Management Services and Process Enactment Services

The data management services provide the means by which the knowledge represented via the modeling capability is implemented in a database, interpreted, and utilized. The data management services implement the chosen data model, execution model, and change model. The following capabilities and requirements will be satisfied by the KBE-SLCSE data management services.

Data Definition and Manipulation Language(s) A framework extension language which provides convenient and interactive access to engineering data.

Data Dictionary Services and Typing A self-descriptive schema management services enforcing consistency of data with type definitions and supporting storage of arbitrary data types.

Storage A data storage facility which efficiently stores large amounts of data with hidden internal structure and provides services for database creation, deletion, backup and recovery.

Multiple Levels of Databases Supporting hierarchy of databases (workspaces) for task decomposition and cooperative work.

Performance A performance which is acceptable for a sizable team of engineers and degrades gracefully as physical and virtual memory are used and which includes performance-enhancing capabilities such as buffering, caching and clustering.

Locking and Concurrency Control A uniform locking support at several levels of granularity and both optimistic and pessimistic concurrency control protocols.

Short-Term Transactions A two-phase commit transaction protocol supporting operations including starting, stopping, aborting, checkpointing and journaling.

Data-Driven Processing and Triggers Supporting specification of slot triggers causing forward-chaining on slot access.

Query A content-based retrieval supporting relational associative query, inferential query, and demand-driven processing.

Exception Handling A flexible exception handling to cope with error conditions signaled by tools or caused by human activities.

Views A database view facility supporting role-specific user views by leveraging the typing mechanisms.

Change and Configuration Management Advanced change and configuration management concepts, including, the concepts of multiple workspaces and engineering transactions.

Data Integration Supporting multiple data representations for interfacing to different languages, tools, and hardware.

Distribution A support for distributed schema management, automatic object migration and location transparency in distributed environment.

Schema Development Support Browsers and tools for application and schema development.

The process enactment services enforce the process described in terms of the process model. The mechanisms can play an active role in the software development process by enforcing process constraints, providing context-sensitive help, coordinating the development in a multi-user environment, automating tasks within the software process, and performing other useful functions. The requirements on the process enactment mechanisms include

- Binding mechanisms for database and system values
- Process program execution efficiency
- Process enactment portability
- Support for multi-user paradigms and interprocess communication
- Longevity and Permanence of process programs

4.1.1.3 Database Schemas and Processes

The database schemas formally captures the software engineering domain model. The following schemas will be enhanced to describe additional services and capabilities

Environment Management Subschema – Formally describing the services provided by the environment manager.

Schema Management Subschema – The schema management subschema will formally describe the services for defining and modifying schemas and types within the schema.

Meta Subschema – The meta subschema shall formally describe the data model and the services available to schema definer for describing application specific schemas. The schema will minimally define type, object, attribute, relationship, operation, function, function request, constraint, etc. This subschema will additionally define a range of extensible data types including recursive nested types, structure, unions, arrays, sets, lists, directed graph.

Tool/User Interface Registration Subschema – Describing the semantics for registering the tool's computational software and user interface within the framework.

Execution Engine Subschema – Describing the services of the execution engine.

Data Management Services Subschema – Describing the semantics for the data management services such as access control, audit trail, configuration and change management, short-term transaction management, engineering transaction management, concurrency control, etc.

User Interface Prototyper Subschema – Describing the semantics of the user interface prototyper tool and constructs that describe the user interface generated using the user interface prototyper.

A process is a program in terms of the process model. The applications which will be described using the process model includes:

Software Methodology Support Automated support for tailorable software methodologies.

Environment Management Support Specification and modification of system environment which may be described as a collection of activities with prespecified ordering.

Intelligent Context-Sensitive Help, User Guidance, and Constraint Management The process enactment mechanisms can support intelligent context-sensitive help, user guidance and constraint management.

Task 2 specified the tasks needed to enhance the infrastructure and analyzed alternate approaches to enhancing the infrastructure.

4.2 Infrastructure Enhancement Alternate Approaches

The infrastructure enhancement alternate approaches is described in terms of:

- The tasks involved in infrastructure enhancement
- Alternate approaches to infrastructure enhancement

4.2.1 Tasks Involved in Infrastructure Enhancement

Several approaches to enhancing the infrastructure were investigated. The set of tasks that must be accomplished for each approach are roughly the same. Differences between the approaches will cause the costs for each task to be different. Enhancing the infrastructure involves these six steps:

- **Task 1, Define the Enhanced Data Modeling and Process Modeling Capability** – Define the data definition/data manipulation and process definition capability necessary to describe the engineering process such that the description can drive an environment.
- **Task 2, Identify (and Possibly Acquire or Implement) the Infrastructure Platform** – The platform should be able to support the data modeling and data management capabilities required by SLCSE goals. Additionally, the platform should support process modeling and process management services.
- **Task 3, Define the Framework Extension Language (KDL) and the Process Definition Language (PrDL)** – The data description/data manipulation language KDL provides programmatic access to data modeling capabilities and to the capabilities of the framework and its tools. The PrDL provides the constructs for describing a

process program. Ideally, the implementation platform would include an extension language suitable for use as KDL and PrDL. The PrDL may be an extension of KDL, with additional constructs needed for process modeling.

- **Task 4, Develop the Compilation and Instantiation Facilities That Enable the Descriptions of Artifacts, Agents, and Activities to Be Instantiated and Executed on the Implementation Platform** – This task will depend on the capabilities that come with the identified implementation platform and the expressiveness capabilities of the KDL and the PrDL. This task may be nonexistent or minimal if the implementation platform includes an extension language suitable to be the KDL and provides support for process programming.

- **Task 5, Rehost the Existing Schema and Tools on the New Data Modeling and Management Capabilities Extending/Enhancing Them to Use the Power and Performance of the New Database** – The database schema, including contract subschema, system requirements subschema, and software requirements subschema, formally describe the project knowledge utilizing the SDL. The database schemas capture the static structural aspects of 2167A-based projects. The dynamic and behavioral characteristics of such projects are not modeled and are thus not defined. The schemas also suffer from the inadequate typing facilities provided by SDL. Additional schemas for modeling process artifacts, schema management services, extensible set of complex and primitive data types, etc., will have to be defined. (An SDL/ERIF emulation layer can allow current tools to access data in the new objectbase until they are rehosted. This task depends on the result of Task 3:

  - How the KDL is defined,
  - Compilation/instantiation facilities of the KDL defined.

  This step can partially overlap the previous step. The amount of work required for this step may be greatly reduced if the platform is an engineering framework with sufficient tool-integration services and a number of general-purpose tools are already integrated.

- **Task 6, Add Additional Capabilities** – This depends on the capabilities that come with the identified implementation platform. Examples are static analysis of descriptions (data type definitions), evaluation, and optimization of descriptions.

4.2.2 Alternate Approaches to Infrastructure Enhancement

Several approaches to enhancing the infrastructure were investigated.

4.2.2.1 Data Model Selection

A variety of data models as described in the Task 1 report were analyzed including:

- **Entity-Relationship (E-R) Model** – The E-R model can easily represent structural characteristics. It does not however, provide direct support for describing behavioral or dynamic characteristics, nor does it support intensional descriptions, because it cannot straightforwardly express subtyping, disjunction, negation, or quantification.
- **Semantic Nets** – Semantic nets are primarily declarative; that is, constructs for specifying behavior are typically not provided. They provide both extensional and intensional representation ability.
- **Frames** – Frames mix declarative and procedural representation styles. They also provide both intensional and extensional capabilities and excels at representing complex, highly structured collections of facts and relationships. Frames do not support the conventional general-inference mechanism, there is no provision for information hiding or the specification of operations to make frames be abstract data types.
- **Object-Oriented Model** – Easily captures complex structural and behavioral characteristics, but typically does not provide direct support for describing dynamic characteristics. It has powerful subtyping and inheritance, but lacks other intensional description capabilities, since it cannot express disjunction, negation, or quantification.
- ***Production Systems*** – *Primarily declarative* representation scheme with provisions for both extensional and intensional capabilities representation. Rules are an appropriate knowledge-representation scheme when control is data-directed or goal-directed rather than sequential. However, the disadvantages of production systems include difficulty understanding and maintaining large systems, difficulty following the flow of control, and difficulty representing complex relationships among multiple entries (objects, activities, events, etc.).
- **Logic Models** – First-order predicate calculus rules are highly expressive, supporting both extensional and intensional styles of description.

Thus, the Task 1 report concluded that a data modeling capability like KDL should be built on a mixed object/logic model – an object-oriented model enhanced with rule-based constraints. The constraint language should be tailored to efficiently support engineering applications. Task 3 recommended the use of object model which is based on emerging standards such as Portable Common Tool Environment (PCTE), Information Resource Dictionary Service (IRDS), Object Management Group (OMG), and A Tool Integration System (ATIS).

4.2.2.2 Process Model Selection

A variety of process models as described in the Task 2 report were analyzed to support the process model requirements including:

- **Graphical Representations** Supporting rapid process definition with limited expressibility to model complex software processes. Several graphical representations were analyzed including:
  - Data Flow Diagrams
  - Finite State Machines
  - Petri Nets
  - Idef Notations
- **Rule-Based Grammar** Supporting process description as a sequence of embedded activities. The different production-rule languages analyzed included:
  - **Inference-Based Production Rules** – Inference-based production rules which can be translated into *rule-based* languages. Examples are Prolog, CLIPS, and LogLisp.
  - **Simple Production Rules** – Simple production rules are translated into *procedural* languages. Examples of procedural languages are Ada, Pascal, and C.
- **Process Generator Generators** A meta language supporting definition of process representation languages (compared to process programs). The various application generator generators considered included:
  - Parser generator generators (e.g., YACC, ICC);
  - Language rewriting systems (e.g., NEWYACC);
  - Graph rewriting systems (e.g., OPTRAN, TWIG);
  - User interface generators (e.g., X View);
  - Visual programming systems (e.g., KUIE, GARDEN);
  - Visual language system generators (e.g., SIL-ICON).
- **Process Definition Language** Enhancements to the Knowledge Definition Language in supporting constructs for describing process programs.

Task 2 described how each of these alternate representation could be mapped onto the services provided by the data model. The overall conclusion was to use the Process Definition Language in the short-term and to provide support for graphical representation and process generator generators in the long-term.

4.2.2.3 Infrastructure Enhancement Approaches

Task 2 analyzed four alternate approaches to infrastructure enhancements.

4.2.2.3.1 Custom Object/Logic Base The first approach involved development of a custom object/logic base with triggering and rule-processing capabilities.

The pros of this approach were:

- The data modeling and data management capabilities could be optimized for SLCSE's goals.
- Full ownership and control of the design and implementation.

The cons of this approach were:

- Tremendously expensive.
- Large up-front effort resulting in delay to most other SLCSE enhancements that rely on the enhanced database.
- Duplication of functionality currently available through commercial packages.
- Recurring maintenance and enhancement costs.
- Difficult to match up with respect to reliability, performance, documentation quality, and other characteristics with commercial offerings.

4.2.2.3.2 Pre-existing Rule Based System This approach involved rehosting SLCSE on a pre-existing rule-based system. The feasibility of enhancing the C Language Integrated Production System (CLIPS) to meet the infrastructure requirements were analyzed. CLIPS is a forward-chaining rule-based system supporting OPS5 type capabilities. The overall analysis is also applicable to commercial products including KEE, ART and NexPert.

The pros for the approach include:

- Powerful and flexible data model that is object-oriented, with sophisticated rule processing.
- Easier integration of the process modeling capability.

The cons for this approach include:

- Many data management requirements would not be met.
- Performance will likely be a problem.
- Tool-integration facilities are also limited.

4.2.2.3.3 Pre-existing Object Base This approach involved rehosting SLCSE on a pre-existing object base, and adding triggering and rule-processing capabilities. An in-depth analysis on three object-based systems was performed: COHESION, ONTOS, and ITASCA. COHESION is DEC's CASE environment framework supporting ATIS data model. ONTOS is a full object database system developed, marketed and supported by Ontologic. ITASCA is a distributed object-oriented database management system developed, marketed and supported by ITASCA.

The pros for the approach include:

- Powerful and flexible data model.
- Satisfies most or all data management requirements.
- Satisfactory performance since underlying database geared towards engineering applications.

The cons for the approach include:

- Requires effort to integrate the missing rule-processing and process-modeling capabilities with the object base.
- Sufficiently-mature object bases are only starting to appear (e.g., Itasca, ODI ObjectStore, DEC Cohesion's CDD/Repository).

4.2.2.3.4 Evolve Existing SLCSE This approach involved incrementally enhancing the existing SLCSE, evolving its current data modeling and data management capabilities and inserting inserting pre-existing rule-based capabilities.

The pros for the approach include:

- Incremental and gradual.
- Least costly way of adding new capabilities while preserving existing ones in the short term.
- System owned by Rome Laboratory thereby minimizing licensing costs.

The cons for the approach include:

- It is difficult to get high performance when emulating an engineering database atop a relational database.
- The current SDL and ERIF are of prototype quality. The engineering database they provide must be maintained and enhanced by the SLCSE provider.

Figure 4.1 summarizes the effort involved in enhancing the infrastructure using the alternate approaches.

Task	Approach 1: Custom Objectbase	Approach 2: Commercial Rulebase	Approach 3: Commercial Objectbase	Approach 4: Evolve Existing SLCSE
1) Define Data Capability (Task 1)	6MM	6MM	6MM	6MM
2) Acquire/Implement Platform (Task 2)	150MM	6MM	3MM	50MM
3) Define KDL and PrDL (Task 3)	6MM	2-6MM	2-6MM	6MM
4) Develop Compilation (Task 4)	24MM	0-12MM	0-12MM	18-24MM
5) Rehost Schema and Tools(Task 5)	6MM	6MM	6MM	6MM
6) Add Capabilities (DB6)	3-9MM per tool	3-9MM per tool	3-9MM per tool	3-9MM per tool

Figure 4.1: Estimated Level of Effort for the Four Approaches in Person-Months

4.3 Toolset Enhancements

The task 2 identified the requirements and the capabilities of the SLCSE toolset and analyzed the strategies for implementing the enhancements to the current toolset. The toolset enhancements were categorized into general toolset and specific toolset. The capabilities and the services by providing general toolset can be leveraged by a wide range of SLCSE tools, while the enhancements to specific tools will benefit particular life cycle tools. Task 2 concluded that the general toolset can be accomplished in the mid-term and would rely primarily on the services provided by the data and the process model. The specific toolset enhancements were concluded to be more of importance in the near-term and the implementations of this enhancement can benefit from the data and the process model.

4.3.1 General Toolset Enhancements

Task 2 identified the following general toolset enhancements and described alternate enhancement strategies.

Domain-specific expert assistance In deep and narrow subdomains of software engineering expertise can be captured and formally represented in expert systems. Expert assistance can then be provided to everyday users as they work in a particular life-cycle activity or role. Such expert systems may answer questions, offer advice, make diagnoses, test hypotheses, provide critiques, or abstracts.

Knowledge capture and communication Advanced environments like SLCSE capture a considerable amount of information about a wide range of "large-grained" software products that can be communicated to later life-cycle activities. Knowledge capture can be done both in background and intrusively. Services can be provided for making queries and inferences over the captured knowledge.

User guidance Because a knowledge-based framework can reason about project state, it can offer considerable help in complexity management and can guide the user in several ways. It can provide legitimate choices, track details of the session state, maintain constraints amongst objects and the actions that are required.

Error prevention/control When the framework is knowledgeable about the dynamic aspects of the objects under its control - how these objects depend on each other - it can actively maintain consistency within and between products. This capability will prevent users from violating constraints and providing automatic consistency maintenance.

Coordination of agents Since, a knowledge-based framework can reason about the project state, it can coordinate the actions of multiple agents according to a specified methodology (i.e., process program). It can manage the sequencing of user actions and tool processes as they operate on the project database, require that certain communications occur, prevent actions that would violate the methodology, and weave tool invocations and user actions into higher-level operations that encapsulate complex activities.

Opportunistic Processing By reasoning about the project state the environment can function as an assistant, actively initiating operations in response to user actions and requests.

Task 2 described alternate approaches to supporting the general toolset capabilities. Figure 4.2 illustrates how the enabling technologies can be leveraged to accomplish general toolset enhancements.

4.3.2 Specific Toolset Enhancements

The specific toolset enhancements identify the characteristics and capabilities of SLCSE tools. The task 2 identified the requirements and the capabilities of the tool and discussed the strategies for implementing enhancements to the current toolset.

- **Prototyping Tools** – Prototyping tools supports the rapid construction of the interactive user interface based on X-Windows and the object base. Currently, SLCSE provides a VT-100 based user-interface prototyper tool supporting windows and menu definitions. The object base prototyper is supported via a textual language SDL.
  - Object editor prototyper – The requirements and the the alternate approaches for developing an object editor prototyper tool supporting the creation and modification of a X-Windows based user interface for entering, reviewing and updating information for interrelated objects in the database was described.

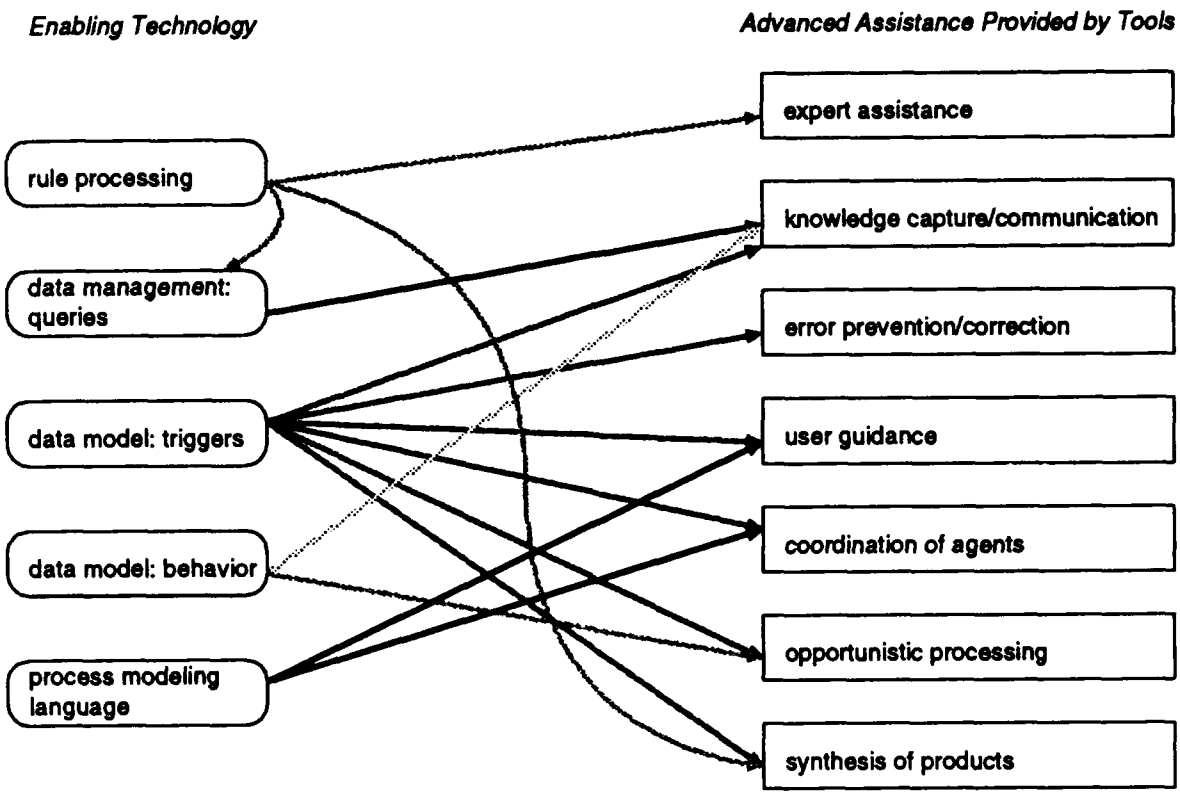

Figure 4.2: Tool Enhancements Relationships to Enabling Technologies

- Schema editor – The requirements and the alternate approaches for developing a schema editor supporting the means for entering, reviewing, and updating schema information was described.

- **General Support Tools** – The general support tools are a collection of tools that are applicable across all phases of the software development life-cycle. The requirement for the following general support tools is:

  - Schema browser – The requirements and the alternate approaches for developing a schema browser supporting the browsing of schema information was described. The current implementation of SLCSE supports life-cycle specific browsing tools.
  - Tool integrator – Enhancements to the tool integrator supporting the integration and removal of attached and integrated tools in SLCSE was described. Enhancements identified included support for developing data adaptors, support for automatic generation of consistent user interface for invocation and interaction with the tools, etc.
  - Form-based user interface for specifying tool invocation parameters – A collection of X-Windows user interface generated using the user interface prototyper providing the means for specifying tool invocation parameters. The current SLCSE

implementation utilizes VT-100 based user interface.

- **Requirements Tools** – Several near-term enhancements to the requirements tool were described including support for methodology-specific notations and diagramming capabilities, graphical portrayal of requirements, fine-grained integration of requirements thereby facilitating requirements traceability and impact analysis. Several long-term enhancements for the requirements tool were described including: support for reuse and specialization of requirements; verification, completeness and consistency checking, based on knowledge of specification language and application domain; configurable requirement support tool that provides the adaptive support for alternative representations and presentations of the requirements information.

- **Design Tools** – The design tools support the development and refinement of a software design or system design that satisfies the specified functional and performance requirements. Several near-term enhancements were described including:

  - A configurable design support tool provides adaptive support for the design of software utilizing alternative representations and presentations of design information. The configurable design tool facilitates and enforces selected structured design methods including data flow diagrams, structure charts and/or Booch object diagrams.
  - Support for methodology and domain specific verification, completeness, and consistency checking.
  - Support for refinement of high-level functional components into detailed design modules and establishment of traceability information between the requirements and the design tool

- **Coding and Debugging Tools** – The following long-term enhancements to the coding and design tools were described

  - intelligent editing, including consistency checking.
  - reuse assistance, for identifying, retrieving, and specializing modules and cliches.
  - intelligent assistance for debugging.
  - capability to automatically generate complete source code needed for the software application from design specifications.

- **Configuration and Change Management (CCM) Tools** – Configuration and change management is a set of abstractions, techniques, and tools that shall assist in managing the evolution of systems of interdependent design objects. The following near-to-mid term enhancements were described for the configuration and change management tool: storing of multiple and parallel versions; access to multiple configurations efficiently; versioning support at multiple levels of granularity; referring to changing objects consistently; providing dependency maintenance and change notifications.

- **Project Management Tools** – The project management tools supports the planning, controlling, and coordinating the life cycle activities of a software development project. The following near-term enhancements were identified
  - Integration of COTS Project Estimation Tool supporting alternate costing methods including COCOMO and Function Point Counting Practices Manual.
  - Support for project tracking, metrics collection and program management reporting.
- **Environment Manager** – The environment manager tool supports the installation and maintenance of the framework and the configuration, creation, evolution, and maintenance of project environments. The following long-term enhancements to the environment manager tool were described
  - expert advice for determining effective and efficient hardware configuration of hosts and workstations, appropriate for project.
  - expert advice for selecting a toolset appropriate for project.
  - consistency maintenance support for effectively and efficiently adding and deleting users, user roles, and tools.
- **Methodology Support Tools** – The current implementation of SLCSE does not support the description and the enforcement of software development methodology spanning across tools and phases of the development life cycle. Limited guidance is provided by individual tools which guides through the steps involved in interacting with the tool. The following mid-to-long term enhancements to the methodology tool, independent of the software development methodology, were described.
  - guide users through consistent, logical development methodology leading to increased software reliability;
  - guide and control tool use, particularly testing tools, quality metric collection and analysis tools, and document generation/revision tools.
  - coordinate activities, including opportunistically performing activities in background upon deducing that results will be needed later.
  - provide intelligent help and user training from user interface.
  - facilitate and enforce project policy, as specified by rules and plans.
- **Document Generation Tools** – The current implementation of SLCSE provides support for generating documents conforming to the DOD-STD-2167A. It provides a collection of templates for developing DOD-STD-2167A compliant documents and a language for defining a general purpose document. The following mid-to-long term enhancements were described for the document generation tools.

- maintain consistency within and between documents, and between documents and other information generated by life cycle activities.
- facilitate and enforce production of documents according to rules about organization and structure.
- generate some documents automatically in response to changes in stored information about requirements, design, *etc.*
- provide change and configuration management for documentation.
- provide database browsing capability which will reduce need for paper documentation, providing multiple views of captured knowledge, including a 2167A view and a 2168 view.

- **Application/Life-cycle-specific schema editor** – A collection of special-purpose schema editors that supports the editing and browsing of application- and/or life-cycle-specific schemas. The current implementation of SLCSE provides a collection of tools including Design, Requirement, PCRP which provides the support for browsing and manipulating life-cycle specific information. These tools support E-R based browsing and inspecting of life-cycle phase specific data using a VT-100 based user interface. A near-term enhancement is to have these tools support an X-Windows based user interface and support the browsing and inspecting of the behavioral and dynamic characteristics of the data based on the enhanced functionality supported by the enhanced data model.

4.4 User Interface Enhancements

The Task 2 Report identified the requirements and the capabilities of the SLCSE user interface and analyzed the strategies for implementing the enhancements to the current user interface. Task 2 concluded that the majority of the user interface enhancements can be accomplished in the mid-term by leveraging the services of the data model and the process model.

The KBE-SLCSE user interface for conformant tools will provide:

- X-Window compliant user interface – All integrated and attached tools will be based on X-Windows.
- User interface protocol and toolkit – The user interface protocol provides the guidelines for developing conformant tool's user interface, while, the toolkit provides the interfaces necessary to generate a conformant user interface.
- Enhanced context-sensitive help – Conformant tools will provide enhanced context-sensitive help facility that guides the user in selecting the action to be taken and the description as to how that action can be accomplished.

- Enhanced error-handling capability – The error-handling capabilities will provide the mechanisms necessary to get detailed error descriptions on demand and support the means to store errors on secondary devices for later use.
- Adaptable to multiple modes – The user interface supports a broad range of users from expert to novices including providing terse and verbose modes. The expert mode will be supported via keyboard equivalents and accelerators.
- User Modeling – KBE-SLCSE supports the tailoring of the user environments by maintaining user profiles, user preferences, etc.
- Mode for executing host operating system commands – The user interface provides the means for escaping to a command-line or an equivalent interface to execute host operating system commands.
- Transparency in underling heterogeneity – The user interface hides the heterogeneous nature of the system by providing the run-time mechanisms for implementing the client-server computation model.
- Browsers – KBE-SLCSE shall permit the browsing of the SLCSE database. The browser shall support two modes of browsing: inheritance and structural interdependency browsing.

Chapter 5

Conclusions

The major accomplishment of this task was the importance of incorporating the enhancements to the infrastructure. The most critical technical enhancement to be made to SLCSE is to migrate the current data model and the data management services to the ones which were identified in Task 2 and the Requirements Specification. The conclusions from investigating alternate approaches to infrastructure enhancements indicated the need for selecting a commercial-off-the-shelf infrastructure and to provide other services on top of it. The approach of evolving SLCSE gradually, maintaining its current capabilities and incorporating additional knowledge-based capabilities had several limitations.

5.1 Areas for Further Evaluation

The following areas needs further evaluation

- The Task 2 and Task 3 identified the requirements and the capabilities of the infrastructure and evaluated alternate enhancement approaches. At the time the analysis was performed several commercial proprietary systems provided a subset of the functionality and the capability of the infrastructure identified. Few of the commercial offerings were evaluated. Since, the time the analysis was performed several infrastructure standards have emerged and tool vendors are developing solutions around these standards. The standards which are being increasing accepted by tool vendors include PCTE, ATIS, IRDS and Object Broker Services from OMG. A task needs to be performed in selecting a suitable infrastructure standard and extending the functionality of the infrastructure to support the capabilities and the requirements identified in Task 2.

- Task 1 concluded that the enhancements needed to support a fully integrated knowledge-based software developed relied heavily on the infrastructure services. Therefore, Task

2 and Task 3 concentrated on identifying the infrastructure requirements and the alternate approaches to inserting the enhancement. As a consequence of which little resources were devoted towards identifying knowledge-based capabilities and requirements for tools. Instead, general toolset enhancements were identified. Currently, several commercial vendors are offering stable versions of life-cycle specific tools which incorporate knowledge-based technologies A more in-depth analysis needs to be performed in identifying tools which provide advanced capabilities and to seamlessly integrate within the environment such that the control and the data can be shared.

- The current SLCSE toolset provides limited support for life-cycle specific notations and methodologies. For example requirements have traditionally been specified using one or more of alternate notations such as finite state machines, structured analysis techniques, statecharts, etc. An in-depth analysis needs to be perform to define and implement services which support representation of alternate notations and methodologies. A task then needs to be performed in selecting commercial tools and integrating their control and data in terms of the defined services.

- Defining a process model which allows the description of alternate representation of process notations. The Task 2 described the requirements for the process model and the mapping of the process model services on the infrastructure services. However, the requirements for the process model were never validated with examples of representative methodologies. An analysis needs to be performed to describe the DOD-STD-2167A, Defense Information System Agency's Information Management Process Model, etc. in terms of the process model described in Task 2.

- Supporting the Process Model within the Framework. Task 2 described alternate strategies for defining and implementing the process model in terms of infrastructure services. However, further analysis needs to be performed for mapping the constructs of the process model on the underlying infrastructure services, extending them if necessary.

- An analysis needs to performed for defining and enacting alternate process definitions within the framework. Clearly, SEI Capability Maturity Model will have a major impact on the process which is followed in developing software products. A task needs to be performed on integrating the SEI process model within the framework and use the framework as the enactment mechanism.

- Domain specific analysis. The current SLCSE implementation is target towards mission critical software development domains. However, the services provided by KBE-SLCSE can be used to support other domains such as Management Information Systems, Real-Time Embedded Software Development, Manufacturing, etc. An analysis needs to be performed to extend the core set of KBE-SLCSE services and tools with domain specific notations, methodologies and tools. That will make available several growth opportunities for insertion of SLCSE technology.

- Integration of key technology areas which are emerging as critical technologies for supporting large-scaled software development projects including reuse and reverse engineering. KBE-SLCSE services need to be support seamless integration of reuse within the software development lifecycle. Reverse engineering supporting the migration of legacy monolithic systems to systems developed in an open systems environment must be further analyzed.

5.2 Lessons Learned

At the time SLCSE was developed initially the software development environments technology was in its early stages of research. At that time it was important to develop a software development environment to demonstrate the viability of the concepts of software development environments. The technology however has matured since then and it is at critical juncture where several commercial technologies have competitive edge both in terms of functionality and performance against one-of-a-kind developed systems. Clearly, the tool technologies have come a long way and there are numerous tools in the marketplace which support advanced functionality. The infrastructure technology in a similar fashion has emerged to a point where stable implementations based on standards are becoming increasingly available. The SLCSE implementation has served as a reference for several commercial implementations of software development environments. However, at the time this contract was being performed rapid technical advancements were occurring both in terms of commercial offerings such as DEC's Cohesion and HP's SoftBench and standards such as PCTE and Common Object Request Broker Architecture (CORBA) from Object Management Group. Thus, the KBE-SLCSE had to perform a delicate balance between specifying a development plan based on evolving SLCSE to KBE-SLCSE vs selecting and integrating commercial technologies providing the enhanced functionality. In order to make the research effort relevant to future directions and acquisitions, the team took the following approach. An in-depth analysis of the current system was performed to identify the limitations of the system in supporting knowledge-based functionality. It then performed an indepth specification of the requirements for a knowledge-based system and then creation of the development plan and task list specification which is independent of whether the knowledge-based technology is acquired or developed.

U.S. GOVERNMENT PRINTING OFFICE