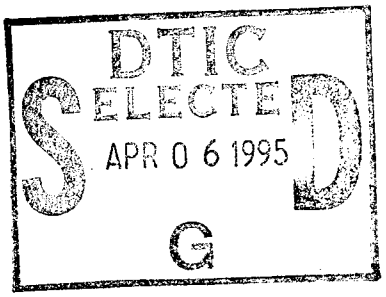
Task/Subtask IV02.1
CDRL Sequence A014–006
17 January 1995

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

Technical Papers: Integrating a SEE for Megaprogramming: Lessons Learned

Contract No. F19628–93–C–0129
Task IV02 – Megaprogramming Transition Support



Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731–2116

Prepared by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

Task/Subtask IV02.1
CDRL Sequence A014–006
17 January 1995

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

Technical Papers: Integrating a SEE for Megaprogramming: Lessons Learned

Contract No. F19628–93–C–0129
Task IV02 – Megaprogramming Transition Support

Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731–2116

Prepared by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

REPORT DOCUMENTION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1/17/95	Informal Technical Report

4. TITLE AND SUBTITLE

Integrating a SEE for Megaprogramming: Lessons Learned

5. FUNDING NUMBERS

F19628-93-C-0129

6. AUTHOR(S)

Dr. Richard L. Randall, Robert K. Ekman, and Gary S. Turner of Loral Federal Systems - Gaithersburg; and
Captain Scott Kent, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

8. PERFORMING ORGANIZATION REPORT NUMBER

A014-006

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Electronic Systems Center/ENS
Air Force Materiel Command
5 Eglin Street, Building 1704
Hanscom Air Force Base, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

N/A

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Cleared for Public Release, Distribution is Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The objective of megaprogramming technology is to make possible a software product-line organization - one that can produce related systems cheaper, better, and faster by using a methodical process based on a common architectural approach. To apply such state-of-the-art technology, a product-line organization needs a common Software Engineering Environment (SEE) that is itself a result of product-line engineering. This paper describes how megaprogramming principles are being applied to assemble and integrate the SEE for one of the three STARS Demonstration Projects. It provides practicle lesson learned - some hard-won - that should be useful to organizations planning their own SEE integration efforts, particularly for organizations considering a transition to megaprogramming.

14. SUBJECT TERMS

software engineering environments (SEE), megaprogramming, STARS, integration, process

15. NUMBER OF PAGES

60

16. PRICE CODE

N/A

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

Preface

This document was developed by the Loral Federal Systems - Gaithersburg, located at 700 North Frederick Avenue, Gaithersburg, MD 20879. Questions or comments should be directed to Dr. Richard L. Randall at 719-554-6597 (Internet: randallr@lfs.loral.com).

Presenters: Dr. Richard Randall, Robert Ekman, Capt. Scott Kent, Gary Turner
Title: Integrating a SEE for Megaprogramming: Lessons Learned
Track: Track 11 - Software Engineering Environments
Day: Tuesday, April 11
Keywords: Software Engineering Environments, SEEs, Megaprogramming, STARS, Integration, Process

INTEGRATING A SEE FOR MEGAPROGRAMMING: LESSONS LEARNED

1. INTRODUCTION

The objective of megaprogramming is to make possible a software product-line organization - one that can produce related systems cheaper, better, and faster by using a methodical process based on a common architectural approach. The STARS program has been building up a technological basis for megaprogramming for many years, and is now transitioning this technology to practice. To effectively apply such state-of-the-art technology, a product-line organization needs a common Software Engineering Environment (SEE) that is itself a result of product-line engineering. To use STARS megaprogramming terminology, the SEE must emphasize domain-specific reuse and support a systematic SEE process with extensive automation.

This paper describes how megaprogramming principles are being applied to assemble and integrate the SEE for one of the three STARS Demonstration Projects. It provides practical lessons learned - some hard-won - that should be useful to organizations planning their own SEE integration efforts, particularly for organizations considering a transition to megaprogramming.

The paper is organized as follows:

- **Section 2, Context**, provides background on the STARS Program and the Air Force/STARS Demonstration Project, which provided the experience for this report.
- **Section 3, Demonstration Project SEE,** describes the Demonstration Project SEE and provides a brief history of SEE assembly and integration.
- **Section 4, Lessons Learned,** summarizes the lessons learned to date on the Demonstration Project, and elaborates selected lessons in more depth.
- **Section 5, Megaprogramming and SEE Integration**, reflects upon the above experience, tying together the topics of Megaprogramming and SEE Integration.
- **Section 6, Conclusion,** summarizes the key points of this paper.

2. CONTEXT

2.1 THE STARS PROGRAM

The ARPA STARS program is a technology development, integration and transition program to demonstrate a process-driven, domain specific reuse-based approach to software engineering that is supported by appropriate tool and environment technology - an approach referred to as "megaprogramming".

2.1.1 Megaprogramming

Megaprogramming is a product-line (family of systems) approach to the creation and maintenance of software intensive systems. It is characterized by the reuse of software life-cycle assets within a product-line including common architecture, models and components. Megaprogramming also includes the definition and enactment of disciplined processes for the development of applications and the evolution of the product-line as a whole. Finally, megaprogramming calls for automated support for the process via advanced software engineering environment (SEE) tools and integration among those tools.

STARS has three technology thrusts that it views as essential for megaprogramming:

- **Domain-Specific Reuse** - STARS contends that high-payoff reuse is best achieved on software "product-lines", where a coherent architectural approach can be used for all of the applications. The product-line will typically be managed by a single individual who will assure that adequate engineering is performed at the domain level.
- **Systematic Process** - In addition to a common architectural approach, a common process framework is needed so that the engineering of each application in the product-line is performed with proper attention to commonality and quality.
- **Automated Support** - To support the above two thrusts, STARS seeks to bring the best available SEE tools and integration technology to bear.

For a more in-depth introduction to the STARS program and the notions of megaprogramming, please refer to [Trimble94].

2.1.2 Demonstration Projects

The STARS program mission is to accelerate the transition to the megaprogramming paradigm. Key vehicles for bringing this about are the three STARS Demonstration Projects - one with each of the three services (Air Force, Army, and Navy) - which are applying the principles of megaprogramming to the development of DoD systems. Each STARS Demonstration Project is documenting qualitative and quantitative experience about the benefits and costs of megaprogramming, as well as the effectiveness of specific tools and techniques. This experience is to be used to initiate the transition of megaprogramming practices to each Demonstration Project's parent organization.

2.2 THE AIR FORCE/STARS DEMONSTRATION PROJECT

In 1992 Air Force Space Command's (AFSPC) Space and Warning System Center (SWSC)[1] won the bid for the Air Force's STARS Demonstration Project. The SWSC is responsible for the maintenance and evolution of software for the C^2 centers at the Cheyenne Mountain Air Force Station (CMAS) - which have the mission for national attack warning/assessment and space surveillance/defense/control. A large number of mission-critical systems are involved, with a high annual maintenance cost.

The SWSC, determined to apply new software technologies to reduce maintenance costs, had already been working to build up a megaprogramming capability; and the partnership with STARS was a natural way to accelerate the transition.

The Demonstration Project application being developed is the Space Command and Control Architectural Infrastructure (SCAI) System, which will be a mobile space control capability.

[1] Effective Februrary, 1995, the SWSC is transfering to the Air Force Materiel Command (AFMC) and will be known as the Space and Warning Systems Directorate (SWSD).

Table 1 provides a summary of the progress to date on the project - in terms of the three STARS megaprogramming technology thrusts.

	Original SWSC Posture	Accomplishments since Establishing STARS Partnership	Activities in Progress (as of 1/95)
Domain-Specific Reuse	• Strong architecture, based on Reusable Integrated Command Center (RICC) architectural infrastructure • Strong emphasis on Open Systems, commercial tools • Domain models underway • Commitment to Ada	• Demonstrated viability of architectural approach • Defined tailored specification standard based on Cleanroom, MIL-STD 498, and others • Completed object-based application models; specified SCAI system and first two SCAI releases • Developed and tested SCAI Release 1	• Developing SCAI Releases 2; specifying Release 3 • Refining product-line architectural framework • Continuing to develop domain models
Systematic Process	• Understanding of importance of process • SWSC Software Engineering Process Group (SEPG) established • Semi-formal process definition in selected areas • Corporate Information Management (CIM) IDEF model underway	• Instituted formal approach to process definition, based on STARS/SEI collaboration • Created a product-line process architecture • Integrated OO, Cleanroom, and the Ada Process Model methods • Formally defined processes for Application Engineering (AE) • Launched a major metrics initiative • Automated staff hour and defect metrics collection	• Nearing completion of formal definition of CM process • Beginning formal definition of Domain Engineering process • Working on second round of AE specification process • Using automated process modeling and enactment support for SCAI Release 2 and 3
Automated Support	• Commitment to Rational Ada support product-line • Commitment to Universal Network Architecture Services (UNAS), and RICC for Architectural Infrastructure	• Integrated a state-of-the-art open systems SEE: IBM and Sun platforms, Rational and TRW toolsets • Installed advanced process support toolset; encoded and began automated enactment of SCAI Release 2 and 3 processes • Instituted automated tracking capability for problems, action items, etc. • Began use of Rational SoDA for automated document production	• Enhancing the functionality and integration of the process tools • Applying Amadeus to automate collection of SEE usage metrics • Extending process automation across geographical locations

Table 1. Demonstration Project - Megaprogramming Progress and Status

3. Demonstration Project SEE

3.1 SEE Composition

The Demonstration Project SEE is composed of approximately 50 workstations connected to 3 server-class machines. It is about evenly divided between Sun and IBM Unix-based platforms. The network is distributed across two geographical sites and operates with restricted, classified access.

The SEE is populated with an integrated set of tools to support end-user functionality, as depicted in Figure 1. In this diagram, we cite only end-user tools, but there are many "infrastructure" facilities, such as relational database management systems (Sybase and Oracle are both used on the SCAI SEE).

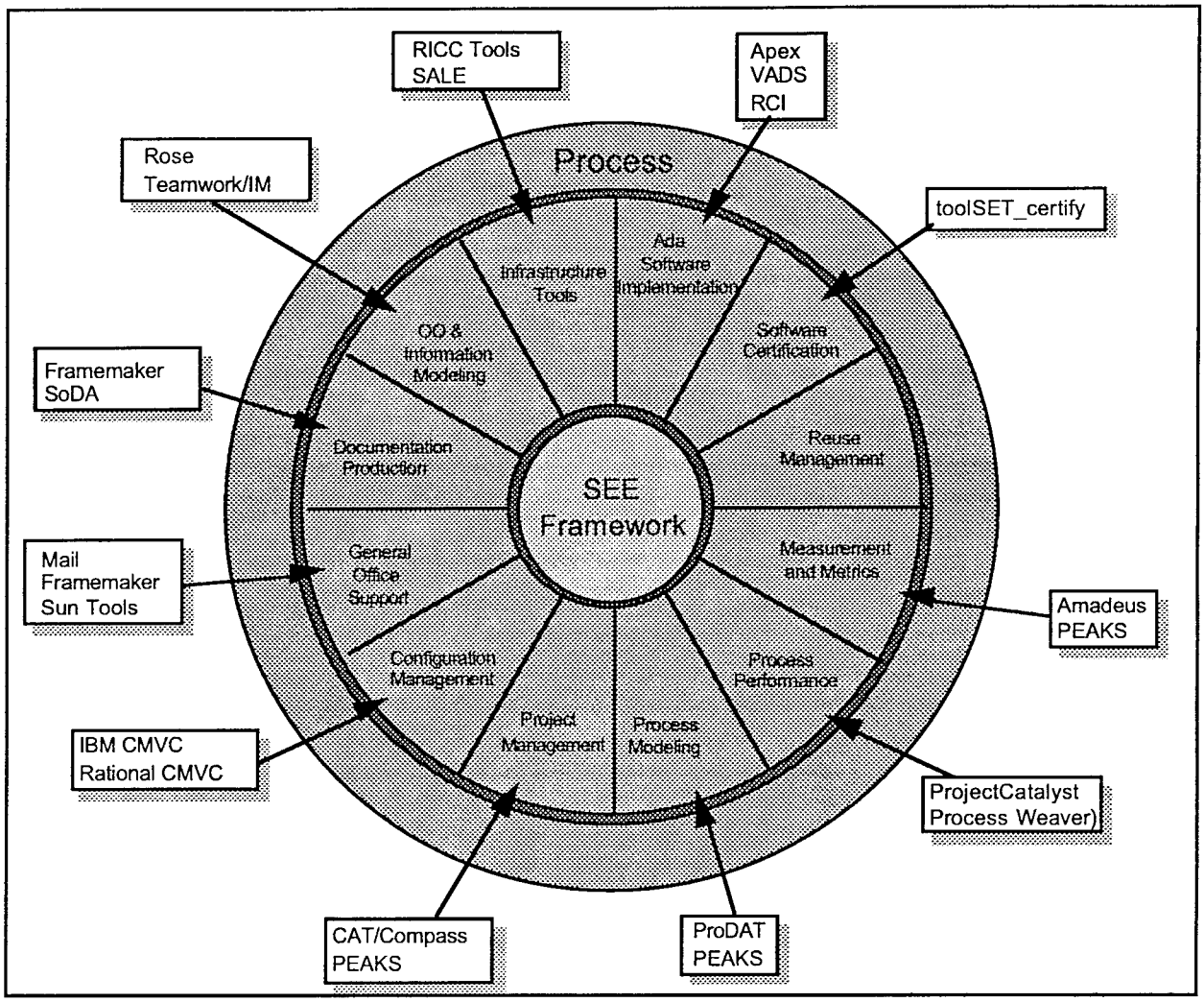


Figure 1. Demonstration Project SEE: Functionality Groups

Table 2 identifies the supplier of each tool shown in Figure 1. As shown, the SWSC has selected Rational and TRW as major toolset providers for the Demonstration Project. Also shown in the table are four tools that have been developed with STARS support.

Type	Tool Name	Vendor/Developer	Purpose
Rational Ada Development Toolset			
	Apex	Rational	Code creation and testing
	RCI	Rational	Interfaces Apex with other Ada compilers
	Rose	Rational	Object oriented analysis and design
	SoDA	Rational	Automated document generation
	VADS	Rational	Verdix compiler
TRW Architectural Infrastructure Support Toolset			
	RICC Tools	TRW	Application display, message, and database definition
	SALE	TRW	Application network definition (used with UNAS)
	UNAS	TRW	Application network manager
STARS-Supported Tools			
	Amadeus	Amadeus Software Research, Inc.	Metrics repository and analysis
	PEAKS	Cedar Creek Process Engineering (ccPE)	Process modeling, planning, and plan simulation
	ProjectCatalyst	Software Engineering Technology (SET)	Low-level process definition and process execution
	toolSET_certify	SET	Cleanroom certification testing support
Other Tools			
	CAT/Compass	Robbins-Gioia	Project management
	CMVC	IBM	Configuration management and tracking system
	FrameMaker	Frame Technology, Inc.	Documentation and publication
	ProDAT	Embedded Computer Resource Support Improvement Program (ESIP), managed by Sacramento/ALC	IDEF-oriented process definition
	Process Weaver	Cap Gemini America	Process workflow manager
	SunTools	Sun	General office support
	Teamwork/IM	Cadre Technologies	Information modeling

Table 2. Demonstration Project SEE Tool Suppliers

The integration of the SEE tools is accomplished through a combination of techniques and mechanisms. Control integration (tool invocation and communication) is provided by

- IBM AIX SDE WorkBench/6000 broadcast messaging service, the Process Weaver message service, operating system process services, and TCP/IP sockets. The WorkBench and operating system process control provided local service within a user's machine session. Process Weaver and TCP/IP provided service between users and between machines.

- Data integration is provided by the Oracle Relational Database Management System (RDBMS) and the operating system file system.
- Presentation and user interface integration is provided by the X Window System and the Motif window manager.
- Process integration is provided by the STARS-sponsored Process Support Environment toolset: PEAKS, ProjectCatalyst (in conjunction with Process Weaver).

3.2 HISTORY

The Demonstration Project began in 1992 and is scheduled to complete in 1995. The first year, the Preparation Phase, was intended to lay the groundwork for the final two years, the Performance Phase. This paper was written just after the first year of the Performance Phase.

3.2.1 Preparation Phase (11/92-10/93)

Early in 1993, a joint team of Loral STARS and AF engineers was established to handle the SCAI SEE. During the Preparation Phase, the team developed an initial overall SEE integration strategy, documented in Version 2.0 of the SCAI Demonstration Project Management Plan (SDPMP).

The SCAI project planned to simultaneously develop several key aspects of its overall approach. Early decisions had to be made about SEE composition prior to detailed understanding of the process the SEE was to support. The key early decisions were to use IBM workstations, Rational Ada development software, and TRW domain-specific support software. The SWSC - committed to the use of Ada for the application product-line - selected the Rational Ada development toolset as a key part of the SEE. The application architectural strategy is based on a TRW-originated architectural infrastructure, and the SWSC has adopted the corresponding TRW toolset as another key part of the SEE. In addition, the Air Force decided to sponsor enhancements to the TRW toolset, and movement continued towards commercialization.

A major new focus for the SCAI project was process support technology. Both the SWSC and STARS organizations were committed to improving the processes used on the SCAI project - and jointly decided to explore the emerging technologies of process modeling, project process management, process execution, and automated metrics collection. It was decided to proceed with development of the Loral STARS Process Support Environment (PSE) toolset during the Preparation Phase and into the Performance Phase. The PSE was designed to provide advanced process modeling, planning and execution capability.

During the Preparation Phase, the SEE hardware and software was delivered in two major installments and integrated with the existing Sun workstations. The SEE was used primarily to develop system and software engineering models that captured domain understanding and laid the groundwork for the SCAI specification effort.

3.2.2 First Year of the Performance Phase (11/93-10/94)

During the first year of the Performance Phase, the team planned and ordered the third and final major SEE installment. Due to changes in STARS funding, the installation was delayed significantly, and though originally planned for April 1994, it was not installed until October. The impact of this delay was to reduce the team's productivity somewhat, although adjusted work patterns and interim SEE assets helped offset this impact. The STARS PSE toolset, also delivered later than originally anticipated, was used for the SCAI Release 2 work. During this period, experience with the SEE was used to define a major update to the PSE, which is now being used for SCAI Release 3 work.

At the start of the Performance Phase, several tools were still under evaluation, including IBM CMVC, Amadeus, the STARS PSE toolset, SoDA, and ProDAT. The primary means of evaluating these tools was to conduct pilots. Based on pilot usage, all of the tools were accepted for use. The pilots took unanticipated labor, especially the PSE.

SEE integration activity during this period resulted in improvements to the PSE (elaborated later in this paper), configuration management, documentation production, metrics, and overall engineering database analysis.

At the project level, there were substantial adjustments to the approach and to the process - necessitating adjustments to SEE strategy, design and implementation.

3.3 PRAGMATIC FACTORS

The following is a recap of some of the constraints, considerations and other practical factors that affected the establishment of the SEE.

- The AF and each of their contractors had an installed base of software with which they were familiar. The selection of the SCAI SEE products required significant consensus efforts to enhance this existing base.
- The SEE included diverse products from many vendors, requiring diverse integration strategies. Some tools were part of single-vendor toolsets, however, lessening the need for integration.
- The combined team (Air Force, Air Force contractors, and Loral) worked in different locations and different time zones. For the first two years of the project, these groups were forced to use segregated SEEs. More recently a link was established between the two primary project locations. The geographical separation also impeded the Loral team's ability to support SEE activities.
- The application is classified, which required the SEE to be classified as well - compounding the factors cited in the preceding point.
- Several products (e.g., process support tools) had release and development delays. The products were less mature than expected. In addi-

tion, because of the need for advanced tools, the Air Force conducted several Beta test and pilot evaluation activities.

- Products and prices changed during the selection and acquisition period. Budgets and plans required several adjustments. In addition, STARS funding delays required shifting some acquisitions into late 1994, and funding reductions required reduction in SEE scope and function.

3.4 ACCOMPLISHMENTS

- **SEE Engineering**: The team piloted and adopted several new tools (Amadeus, IBM CMVC, PSE, SoDA, ProDAT). The first release of the PSE was completed and delivered. The team gathered usage experience and scheduled a major PSE upgrade (completed in late 1994 and now in use). A prototype integration between IBM CMVC and Rational CMVC was built. An artifact database study was completed, laying the groundwork for improvements to overall SEE data integration. The team instituted the use of IBM CMVC for general purpose tracking (problems, action items, etc.).

- **Process Support**: The team applied the PSE to SCAI Release 2, which provided usage experience leading to a major PSE upgrade. The team began using the ESIP-sponsored ProDAT tool to support process definition, and they began using Amadeus for labor metrics collection.

- **Domain/Application Engineering Support**: The team began applying state-of-the art tools for SCAI engineering work, including Rational Rose (object-oriented domain/application modeling), RICC Tools (architectural infrastructure development), and Rational Apex (Ada design/development).

- **SEE Improvement Process**: Two user surveys were conducted which provided valuable input to the overall development of the SEE. IBM CMVC is now being used to support SEE improvement procedures once handled manually. This includes support for:
  - System management work order processing,
  - SEE product problem/enhancement tracking, and
  - Commercial tool vendor feedback (partially implemented process).

- **SEE Technology Transition**: To perform technology transition, the team conducted classes and pilots to introduce new tools. They conducted presentations and demonstrations of the SEE to the SWSC and to outside organizations.

- **Team Connectivity**: Progress has been slow in creating useful electronic team connectivity. This is largely due to logistical problems associated with adding new outside connections into the classified enclosure within the SWSC. Connectivity with one prime contractor, Kaman

Sciences, was achieved by physically relocating the personnel. Connectivity with another prime contractor, TRW, was achieved by implementing a secure link to their facility. The project has coped well with the limitations - e.g., software development productivity has exceeded expectations for the SCAI Release 1 effort - but it is clear that overall synergy could have been better if all project personnel had been linked on a common SEE from the outset.

4. LESSONS LEARNED

4.1 OVERVIEW

The lessons learned presented in this paper are based on a more complete set of SEE lessons documented in the Draft Version 2.0 of the ***Air Force/STARS Demonstration Project Experience Report*** [DemExp95]. Table 3 itemizes lessons from the Experience Report that are particularly relevant to the two focus areas addressed by this paper: megaprogramming and integration. Since space does not permit treatment of all of the lessons in Table 3, we have chosen a subset for more discussion in Section 4.2, Elaboration of Selected Lessons Learned - the lesson numbers are provided in the "Lesson #" column of the table.

The table also cross-references each lesson as to its relevance to the three megaprogramming technology thrusts: domain-specific reuse, systematic process, and automated support (see Section 2.1.1, on page 2, for a definition of these categories). These designations in this table refer to how reuse, process and automation apply to the SEE itself.

The lessons are grouped into the following life-cycle activities:

- **Domain Engineering** - developing a strategy for the family of SEEs that will support a product-line organization.
- **SEE Management** - managing the administrative and engineering activities for a SEE, with emphasis on aspects present when providing a SEE for a significantly new way of doing business - such as megaprogramming.
- **SEE Architecture** - engineering a common architecture for a SEE product-line.
- **Platform and Tool Selection** - selecting hardware, networking and software tools for a new SEE.
- **SEE Integration** - combining the functions of the SEE so that there is reduced end-user perception of the SEE's individual components.
- **SEE Improvement** - gathering information about the SEE's functions, performance, maintainability, etc., and pursuing needed improvements.
- **Technology Transition** - transitioning new SEE approaches and techniques to SEE engineers and end-users.

Lesson Category	Brief Lesson Statement	Lesson #	Megaprogramming Relevance		
			Reuse	Process	Automation
Domain Engineering	A product -line mentality, complete with domain engineering, is appropriate for the SEE product line	1	X	X	X
	Establish a focal point SEE organization for the entire SEE product line	2	X	X	
SEE Management	Managing change in the domain of the SEE is a key challenge for organizations transitioning to megaprogramming	3		X	
	Stringent security requirements represent a significant hurdle to SEE assembly and integration			X	
	A project should place early priority on intra-and inter-project communication and coordination			X	
	SEE engineering "bandwidth" will be impacted by a surprising amount of non-technical work	4		X	
	A SEE "czar" is required for effective management of the SEE activity	5		X	
SEE Architecture	SEE architecture must be developed incrementally	6	X	X	
Platform and Tool Selection	Heterogeneous open-systems platforms can provide an important degree of flexibility	7	X		X
	A project must be judicious about the number of new tools			X	
	A project should be cautious about committing to tools still under development			X	X
	Pilots are an important vehicle for both tool selection and tool usage	8		X	X
SEE Integration	Logistical factors, such as delays in shipment, nuances in licensing, etc., can significantly complicate SEE assembly and integration			X	
	Upgrades to an operating system (or other SEE infrastructure components) can disrupt integration	9		X	
	High-payoff integration can result from selecting a vendor whose toolset evolution objectives align with your organization's goals	10		X	
	Acquiring a single vendor toolset is usually more cost-effective than integrating separate tools	11		X	
	Maintaining process information in multiple forms poses a significant maintenance challenge	12		X	X
	Artifact management (notably configuration management) is a central integration issue	13	X	X	
	Use of BMS messaging services required unanticipated engineering effort to provide guaranteed message delivery	14			X
	A Process Support Environment can provide an effective SEE integration layer	15	X	X	X
SEE Improvement	Medium-to-large projects need a SEE-supported problem tracking system			X	X
	Management of the SEE can be enhanced via an automated tracking tool			X	X
	A project can use an automated tracking tool as the basis for systematic feedback to vendors about their products.			X	X
	Process-driven databases are a good basis for metrics	16		X	X
	Periodic SEE surveys are essential to the organization's SEE improvement cycle	17		X	
Tech Transition	The wide variety of new approaches and tools introduced on a new SEE necessitates a substantial investment in technology transition			X	

Table 3. Summary of Lessons Learned

4.2 Elaboration of Selected Lessons Learned

4.2.1 Domain Engineering Lessons

"Domain engineering" refers to the identification and disciplined exploitation of commonality across a family of related systems - and is a major emphasis in a megaprogramming organization. A key objective of the Demonstration Project is to establish a domain engineering baseline for the SWSC product-line - models, common architectural framework, similar processes, strategic tools, etc., that will allow future applications in the product-line to be built using the same methodology.

We are now of the belief that the SEE which supports the development of the product-line applications actually constitutes a product-line of its own. Accordingly, this subsection supplies two observations about the applicability of domain engineering principles to the SEE.

Lesson 1. A product-line mentality, complete with domain engineering, is appropriate for the SEE product line.

The development of an architecture for a product-line SEE requires a product-line mentality for the SEE itself - complete with SEE domain engineering.

This objective - to support the future SWSC product-line requirements - has led to SEE thought processes that are analogous to SCAI application domain thought processes. Upon reflection, this is entirely appropriate. In the future application product-line organization, there will be multiple logical SEEs - one for each application being developed. These SEEs will share many common hardware and software assets, but each project will have a unique view of these assets - which will guide instantiation and augmentation activities to produce its own tailored realization of the SEE.

The requirements for these SEEs are dictated by the organization's common process (the same process framework, but tailored realizations) and the common application construction methodology (same architectural approach, but some tailoring in each application instantiation). Thus, to deliver end-user functionality "cheaper, better, faster" to the product-line organization, SEE engineers must perform domain engineering of the SEE domain.

Looking back over the experience to-date with the SCAI SEE (and its forerunners at the SWSC prior to the Demonstration Project), it is clear that domain engineering thought processes have influenced the acquisitions and integrations that have led to our current SEE.

This lesson provides the main theme for this paper, and it is discussed in some detail in Section 5, Megaprogramming and SEE Integration, starting on page 24.

Lesson 2. Establish a focal point SEE organization for the entire SEE product line.

A product-line organization should establish a focal point SEE organization responsible for both SEE engineering and SEE management for the entire product-line. Having an organization responsible for the SEE product-line ensures that the SEEs are managed as a product line and that product line activities (such as domain analysis) are done. The SWSC Software Engineering Process Group established an Integrated Process Team (IPT) to address the proliferation of SEE tools across the SWSC directorates, and in doing so acknowledged the problem of "stovepipes"[2] within the SEE domain. As the SWSC moves toward a product-line way of doing business for its end-user applications, it will need to move to a SEE product-line to support it.

4.2.2 SEE Management Lessons

Lesson 3. Managing change in the domain of the SEE is a key challenge for organizations transitioning to megaprogramming.

For most organizations, transitioning to megaprogramming represents significant change - with corresponding pressure on the SEE to change to support the new ways of doing business. Well-intended visionaries will seek the latest tools and technologies - which is justifiable, since there are many gaps in proven tooling for megaprogramming (support for domain engineering, integrated process definition and enactment, architecture-directed program generation, etc.).

However, such changes bring management headaches as well - such as the following, all of which have been encountered on the Demonstration Project:

- Planning uncertainty - due to such factors as slipped delivery dates and integration mismatches;
- SEE instability - due to the newness of the tools, including the possibility that a tool must ultimately be thrown out;
- SEE technology transition issues - due to such factors as paradigm mismatches, extensive need to train users, and the possible need to adapt current processes to take advantage of new tool capabilities.

There is no formula for judging where to draw the line, but here are some weapons in the manager's arsenal:

- A SEE engineering staff that includes someone with extensive experience with assembling and integrating SEEs - preferably with some battle scars with respect to the pitfalls of new tools;
- Piloting (see also lesson 8); and

[2] "Stovepipe" systems are systems that are developed independently, with no intent to share architecture or information except at external interfaces.

- Common sense.

The main point is to be aware that change itself is a key SEE challenge.

Lesson 4. SEE Engineering "bandwidth" will be impacted by a surprising amount of non-technical work.

An organization changing to a significantly new way of doing business (such as megaprogramming) must address a challenging mix of technical and non-technical SEE issues, with a surprisingly large percentage of this effort going to non-technical issues. Further, because the non-technical issues constrain the engineering solution, proportionately less engineering bandwidth is available to address technical issues.

In our case, SEE engineering was complicated by such non-technical factors as:

- The degree of change (cited in the prior lesson);
- The continuing evolution of the processes to be supported by the SEE;
- Significant changes in budget and timing;
- The geographical separation of personnel (initially, three major contractors were located at different locations - although this is becoming less of a problem, since the team is being consolidated in one location); and
- The classified nature of the SCAI application (this necessitated the SEE to be classified - which in turn degraded external electronic communication and diminished the ability of off-site personnel to support the project).

Lesson 5. A SEE "czar" is required for effective management of the SEE activity.

In the face of the inevitable uncertainty in requirements, and the need for long lead times for acquisition and meaningful integration, appointing a qualified SEE "czar" early in the program is necessary. This lead person will be a single point of responsibility for the project's SEE and will coordinate both SEE engineering and planning, making it easier to force certain strategic decisions early - despite the inevitable lack of complete information.

Some key SEE issues that should be addressed early are

- Artifact definition and management, especially configuration management;
- Project-wide tracking support (problems, issues, action items, etc.); and
- Metrics instrumentation strategy.

It is also essential to have an experienced lead engineer devoted to such engineering functions as SEE architecture, tool selection guidance, integration, etc. Due to the criticality of the SEE to achievement of megaprogramming objectives, management should attempt to insulate this engineer from the host of non-technical problems that are sure to occur (discussed above). The conventional approach is to treat the SEE as an acquisition and maintenance problem; thus, the engineering aspects of the SEE tend to lag.

4.2.3 SEE Architecture Lessons

Lesson 6. SEE architecture must be developed incrementally.

An outside consultant cannot present an organization with a pre-integrated SEE to satisfy product-line objectives. There are too many variations; and too much needs to be customized.

At the start of the Preparation Phase (10/92), it was thought that Loral, on behalf of STARS, would provide an off-the-shelf SEE that would be fully integrated by the start of the Performance Phase (10/93). In retrospect, this was naive.

The design of the SEE is predicated on a clear statement of the requirements, namely the automation requirements to support the application product-line. These requirements, in turn, are dependent on the organization's understanding of its process. In the case of the Demonstration Project, some aspects of the basic approach were still being worked out at the beginning of the Performance Phase - let alone the process.

Thus, by the end of the Preparation Phase, Loral and the Air Force were able to cite one of the major lessons learned thus far on the Demonstration Project - one that transcends the SEE: ***All aspects of the product-line approach - domain models, process, application architecture, and in particular the SEE - must be built-up incrementally***. An incremental, iterative approach is necessary since the organization will be simultaneously grappling with fundamental approach issues - delaying the availability of firm SEE requirements.

This principle is recognized by [Brown92], who argues for "evolutionary approaches, rather then revolutionary".

4.2.4 Platform and Tool Selection Lessons

Lesson 7. Heterogeneous open-systems platforms can provide an important degree of flexibility.

The SCAI SEE includes both Sun and IBM RISC System/6000 platforms. The Suns were part of the existing SEE before the Demonstration Project, but the project decided to complete the SEE with IBM platforms and to network the Suns and IBMs together. Although the two platforms cannot run each others' binaries, it is commonplace to network their file systems together and to remotely execute applications via XWindow/Motif.

Although having the two platform types meant that maintenance and system management was more complicated, the project has realized some important benefits. New tools and new versions of tools are often released on one platform (usually the Sun) and then ported to others - and there can be quite a bit of time between ports. The project has thus been able to take early advantage of new releases of several tools during this period, including Rational ROSE and Rational SoDA. It has also been able to take advantage of tools that run only on one platform, such as the STARS PSE (IBM platform), and ProDAT (Sun).

Thus, the additional flexibility provided by heterogeneous platforms can at least partially offset the additional maintenance overhead.

Lesson 8. Pilots are an important vehicle for both tool selection and tool usage.

If there is no past experience with a tool, the best method of in-depth evaluation is to conduct a pilot. The purposes of the pilot are twofold: not only to determine whether to accept the tool, but also to learn how best to use the tool and how to best integrate it into the SEE. The pilot must be based on realistic usage scenarios. The best pilots are essentially precursor production usages on non-critical path activities. On the SCAI project, we piloted Amadeus, IBM CMVC, the STARS Process Support Environment (PSE) toolset, and ProDAT.

4.2.5 SEE Integration Lessons

Lesson 9. Upgrades to an operating system (or other SEE infrastructural components) can disrupt integration.

In an integrated SEE, many of the tools will depend on particular versions of other tools (or parts of the SEE), so that upgrades cannot take place in isolation. Rather, upgrades have to be coordinated with one another. Infrastructural tools (such as operating systems or data bases) have many dependencies, so special care must be taken when planning an upgrade to them. Some upgrades may have to be delayed until dependent tools can be upgraded. Code may even have to be written to temporarily mitigate the effect of upgrades. We recommend building a tool dependency matrix, to better understand the impact of proposed upgrades.

Lesson 10. High-payoff integration can result from selecting a vendor whose toolset evolution objectives align with your organization's goals.

Two examples of how this strategy has paid off for the Demonstration Project are the Rational toolset (Apex, ROSE, SoDA), and the TRW toolset (UNAS, SALE, and the RICC tools[3]). In both cases, the Demonstration Project organization had already established confidence in the vendors, and their toolsets were already on a path that matched the needs of the SWSC product-line objectives. In the case of Rational, the project is committed to Ada (supported by Apex), OO modeling (supported by ROSE) and documentation automation (supported by SoDA). In the case of TRW, the project is committed to the underlying architectural infrastructure supported by the TRW toolset. The Air Force choice to use these vendors' solutions for the Demonstration Project has worked out well, since both have since pursued a vigorous product improvement cycle, including progressively higher degrees of integration.

Lesson 11. Acquiring a single vendor toolset is usually more cost-effective than integrating separate tools.

The foregoing discussion of the Rational and TRW toolsets illustrates that off-the-shelf integration has worked quite well for this organization. Although there has been relatively little home-grown integration to date, experience suggests that the following advantages and drawbacks should be weighed:

- **Advantages of off-the-shelf integration:**
  - Your organization will require less maintenance labor/expertise in-house.
  - The fewer vendors to deal with, the less logistics hassles; the fewer vendors, the less glue. Integration complexity increases as the number of tools/vendors increases.
  - The vendor takes care of assuring the toolset remains integrated as components change; whereas if the integration is the responsibility of the local organization, there are considerable headaches when one of the integrated components changes.
  - With more concentrated investment in a single vendor's product (as opposed to scattered investment in numerous vendors), you have more leverage in getting the single vendor to accommodate your specific requirements.
- **Drawbacks of off-the-shelf integration:**
  - You have to accept the vendor's integration, and you still have to solve the integration problems with other tools.
  - All of the component tools in the integrated toolset have to be updated at once.

[3] The TRW Reusable Integration Command Center (RICC) tools are Display Builder and RHMI (both available commercially), Query Builder and Query Processor (GOTS), and Message Builder and Generic Message Parser (GOTS).

- Sometimes the vendor's plans do not match well with your own schedule. For example, if an underlying OS or DBMS has to change, you will have to wait, as we had to, for the vendor's total package update, instead of a single component's update. If you are doing your own integration, you can swap out a piece (reasons for swapping: licensing issues, functionality, etc.).
- It may be hard - or even impossible - to integrate a component of a monolithic integrated toolset with an outside tool. A monolithic toolset may have missing functionality that must be provided by another toolset that has overlapping functionality with the first - resulting in wasted redundant functionality in the SEE. (An example of this currently is Rational Apex, which supports version control but not problem tracking; IBM CMVC, which also supports version control and problem tracking but does not provide Apex's semantic understanding of Ada.)

Lesson 12. Maintaining process information in multiple forms poses a significant maintenance challenge.

On the SCAI project, process modeling and enactment is split among three tools. ProDAT supports activity-based modeling, using IDEF notation as a basis; PEAKS supports workflow modeling and process-driven planning, using ETVX notation as a basis; and ProjectCatalyst supports workflow execution and coordination, using task lists and agendas as a basis. Each tool provides its own independent method for entering and storing its data, and each supports a different set of process modeling notions. Using all three tools allows the project to take advantage of the unique capabilities of each but has the drawback that process information is split and must be managed carefully to keep it synchronized.

The SCAI is grappling with this problem now, and some remedial action is in progress. First, PEAKS and ProjectCatalyst were already partially integrated and share a common relational database management system for their data. In addition, both have recently been upgraded to provide better integration: the ProjectCatalyst view of the process can be instantiated from the PEAKS view. Second, the potentially voluminous combination of IDEF diagrams and supporting text is being consolidated into a single database via ProDAT. Hopefully, these two measures will greatly reduce the maintenance of SCAI process information, which will in turn increase the likelihood that the processes will actually be enacted in accordance with their definitions - thereby enabling a genuine process improvement cycle.

Lesson 13. Artifact management (notably configuration management) is a central integration issue.

Since the SEE's primary role is to provide the means to create, maintain and access project artifacts, configuration management is a pre-eminent SEE integration issue. At the time of this report, CM support on the SEE is still fragmented. For example, IBM CMVC is in use for many aspects of project work (e.g., most problem tracking), and Rational Apex CMVC is in use for Ada code development (code management only, no problem tracking), but as of yet, there is no SEE integration between the two - although some prototyping has been conducted to pave the way for future integration.

There are several reasons why integration lagged in this key area:

- The CM process is only partially defined. There is a high-level $IDEF_0$ description, but the project is still grappling with nuts-and-bolts definition of artifacts, how they will be constructed and managed, how they will be delivered from point to point within the process, and how the directories should be set up on the SEE to support this work. (A complicating factor is that developing the SCAI's new megaprogramming approach has involved coming to grips with new types of artifacts, combined in new ways.)
- Several candidate CM tools must be considered, three of which are already in use on the SCAI (IBM CMVC, Rational CMVC, the State Data Repository part of ProjectCatalyst), and many more of which are tools currently in use elsewhere within the SWSC.
- CM-implications of defining a product-line process are not well understood. This is currently an active research topic.

Lesson 14. Use of Broadcast Message Server (BMS) messaging services required unanticipated engineering effort to provide guaranteed message delivery.

The integration mechanism within IBM AIX SDE WorkBench - the Broadcast Message Server (BMS) - is intended for dialog communications, so relatively sophisticated programming is needed on both sides. The way we programmed the use of WorkBench messaging service did not force the startup of the receiver of the message. As a consequence, some messages were undelivered, and integrity of the integration between the tools was lost. We now understand the requirement for more handshaking and error checking within the message event loops.

There were other limitations in WorkBench that surfaced during the SEE integration effort. Communications between tools were supported only within a single user session. The IBM WorkBench product is an implementation of the Hewlett-Packard Broadcast Message Server (BMS) technology. We believe many of the WorkBench limitations were the result of the lagging implementation of BMS improvements.

The emerging standard for SEE integration messaging is found in Sun's ToolTalk [ToolTalk94]. This technology is part of the emerging industry standard referred to as the Common Desktop Environment (CDE). All the major environment platform vendors (Sun, Hewlett-Packard, IBM, Digital, and others) have agreed to provide CDE compliant products with their platforms. We believe that future releases of AIX will provide an implementation of BMS (through ToolTalk) that will ease most of the limitations that we encountered.

Lesson 15. A Process Support Environment can provide an effective SEE integration layer.

[Brown92, p.304] discusses the fundamental role an organization's process plays in establishing requirements for a SEE: "Understanding the process must precede introduction of solutions." Prior to its affiliation with the Demonstration Project, Loral conducted extensive R&D in the area of process automation. This work led to the notion of a "process support environment" (PSE).

The PSE is a set of tools which supports the organization in defining its process and then using the process as the basis for SEE-assisted planning and enactment. Experience in developing and using PSEs - including the recent experience on the Demonstration Project - has led to the view that it is useful to think of the PSE as an abstract integration layer, which encapsulates the organization's process and exports the SEE's functionality to the end-user in the context of the process. This layer is only partially realized to date, but experience has already shown its potential for implementing process-driven SEE integration.

Please refer to a more extensive discussion of the PSE in Section 5.3, "A Key Integration Layer: The Process Support Environment," on page 27.

Also refer to related lessons 12, 13, and 16.

4.2.6 SEE Improvement Lessons

Lesson 16. Process-driven databases are a good basis for metrics.

Process-driven tools can provide a credible source of measurements to support metrics, since the measurements are driven by the work itself.

There are two notable examples where this principle is being applied:

- Process-driven planning and enactment

  ProjectCatalyst tasks are tied to the PEAKS-generated plans - which in turn are based on the project's defined process. As tasks are executed by practitioners, the start and completion dates are automatically posted to both the ProjectCatalyst and the PEAKS databases. This means that task leads and managers are provided accurate visibility of work progress - since milestones are reached only in accordance with the defined process, complete with prescribed validations.

In addition, as tasks are completed, the SEE gathers task wall-clock time (automatically) and expended labor statistics (via prompts to the user). These are also posted to the ProjectCatalyst and PEAKS databases and are available for metrics analysis.

In both cases, the accuracy, timeliness and completeness of the data is greatly improved by the fact that it is directly tied to the work itself.

- Problem and change tracking

  The project is using an automated tracking system to track not only problems but also suggested improvements and work orders - which are entered into the system as they are conceived. The system tracks their progress as problems are resolved and work orders implemented. We can quickly compute metrics on the progress and easily identify problem areas.

  Here again, people don't have to do anything special: the accuracy, timeliness and completeness of the measurement data is derived from the fact that it is tied directly to work processes.

Lesson 17. Periodic SEE surveys are essential to the organization's SEE improvement cycle.

Active process improvement and SEE improvement programs are central to megaprogramming. To identify needed improvements, the SEE's end-users must be regarded as customers. We have conducted two SEE surveys to date. Feedback was sought on individual tools, SEE integration among the tools, and system management support responsiveness. [DemExp95] contains an extensive appendix discussing our survey techniques, which included innovations on assessing tools as well as on determining high-payoff opportunities for improving integration.

The following are some of the survey findings:

- Users felt the SEE's functionality was adequate for their work and "headed in the right direction".
- The areas of tool functionality that were deemed "strategic" to the project's long term objectives were:
  - Ada software production (currently supported by Rational Apex and related tools)
  - Architectural infrastructure support (currently addressed by the RICC tools)
  - Object-oriented and information modeling (currently addressed by ROSE and Teamwork/IM)
  - Metrics
  - Process definition (currently addressed by ProDAT and PEAKS).
  - Documentation and automated documentation production (currently addressed by FrameMaker and SoDA)

- The areas of SEE integration that were deemed "strategic" were:
  - Configuration management - with most other SEE areas
  - Project management - with most other SEE areas
  - Metrics - with most other SEE areas
  - Object-oriented and information modeling - with document production and Ada software production
  - Ada software production - with architectural infrastructure tools and software certification
  - Process modeling - with project management and process execution

Based on the analysis of the survey results, the following survey improvements will be taken into account for the next survey:

- A larger cross-section of the end-user population will be surveyed (40% vs 25%).
- Additional judgments will be requested to enhance the survey's utility in deriving metrics. For example, people were asked to assess the importance of functionality groupings to their own jobs; they should also be asked to estimated the importance to the future product-line objectives.
- Users will be asked to rate the value of the survey itself - as well as to provide suggestions for improvement.
- Outside consulting will be sought to review survey techniques and recommend improvements.

5. MEGAPROGRAMMING AND SEE INTEGRATION

5.1 VIEWING THE SEE AS A MEGAPROGRAMMING PRODUCT LINE

Although this paper is primarily an experience report, the authors' reflection has led us to an interesting extension of the megaprogramming approach - an extension that has helped us to better understand our own Demonstration Project experience, and one that may be of use to others in plotting the course for their SEE. The realization is this: ***the SEE itself is a product-line and is as much subject to megaprogramming principles as the applications it supports.***

Consider that the fundamental intent of megaprogramming is to enable systematic reuse across a product-line of applications - applications that are under the control of a single domain manager, and whose natures are similar enough to merit a common engineering approach. As shown in Figure 2, it is natural to think of the organization's SEE assets as several ***logical*** SEEs - even if they share many of the same ***physical*** assets (workstations, software licenses, network connections, etc.). In effect, each application has its own supporting SEE.

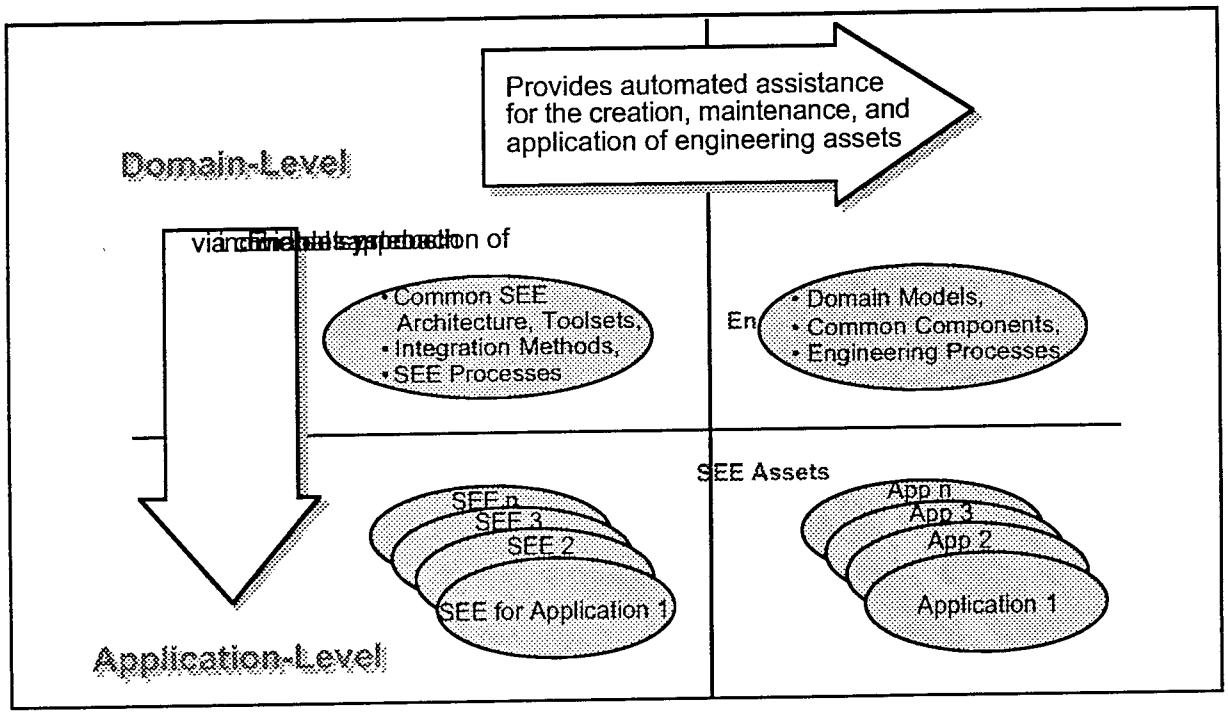


Figure 2. A Product-Line of SEEs in Support of a Product-Line of Applications

Although we want the maximum commonality across this group of SEEs - just as we do with a product-line of applications - we also acknowledge that there are unique aspects of each. For example, for a SEE supporting a space catalog maintenance application, we may need capabilities which are not needed for a missile

warning application - such as a special orbital mechanics analysis support package, or a Prolog engine to support rule-based artificial intelligence research. In addition, although the megaprogramming processes for each application follow a common process architecture, each application has its own nuances - thus each SEE must support a tailored rendition of the common process.

5.2 APPLYING MEGAPROGRAMMING DISCIPLINE TO THE SEE

Since there does appear to be a product-line of SEEs, the natural question is: do megaprogramming principles apply to developing, maintaining, and evolving the product-line? Although space does not permit a detailed analysis here, the answer appears to be yes, as illustrated in Table 4.

Megaprogramming Principle	Applicability to SEE Product-Line
• Domain-Specific Reuse	• Domain requirements established by ***Application Product-line's*** common process framework and need for automated support; domain engineering is appropriate to establish domain models, etc., for the ***SEE Product-Line***. • Common SEE architecture framework (see Table 5) to maximize potential for reuse • Common artifact management strategy, CM strategy (including reuse library) • Common toolset, minimizing license costs, minimizing training costs, enabling maximum use of common integration • Common toolset tailoring and integration approach, with process for product-line adaptation of common scripts and data
• Systematic Process	• View application product-line's processes as integral to the SEE • Use formal processes for SEE procurement, SEE engineering, SEE assembly and installation, SEE management (problem and work item tracking) - and SEE improvement (including survey techniques)
• Automated Support	• Automated tracking tool to assist with SEE administration • Commercial GUI builder for locally-written applications • Commercial integration framework services

Table 4. Applicability of Megaprogramming Principles to SEE Product-Line

To reiterate, although this discussion shows that a product-line mentality is highly appropriate to the SEE product-line, it is only on reflection that our group has come to this realization. For example, we did not start the Demonstration Project with a concerted effort to plan out our SEE using domain engineering. As it turns out, however, we now can see that we have been using product-line thinking as we decided what hardware to acquire and which tools were strategic.

For example, Table 5 illustrates how our current SEE exhibits architectural characteristics that provide a good foundation for domain-specific reuse for the SEE product-line.

Architectural Characteristics	Example Sub-Categories	SCAI SEE Domain Implementation Examples
• Layered architecture	• Common underlying operating system environment	• Unix, TCP/IP, NFS, X
	• Process support environment	• STARS PSE
	• Major functionality encapsulations	• Rational's integration family of Ada support tools (centered around Apex)
		• TRW's integration family of code generation tools supporting the architectural infrastructure (UNAS, RICC)
• Common user interface	• Common window-handling	• Motif
	• Common window behavior look and feel	• Open Interface (from Neuron Data), Display Builder (from TRW)
• Common program interface mechanisms	• Low-level API services	• Broadcast Message System (part of HP's SoftBench)
		• TCP/IP Sockets (for guaranteed message delivery)
	• High-level data repository services	• COTS DBMSs (Oracle, Sybase)
• Component reuse	• COTS tools	• Various

Table 5. Architecture Characteristics for the SEE Product-Line

Our intent here is not to say we have done a remarkable job of megaprogramming for the SEE (after all, we are only supporting a single application so far), but rather to say we believe megaprogramming applies to this product line.

As we learn more about how to do megaprogramming, we look forward to applying the improved techniques and practices to the SEE product-line as well.

5.3 A KEY INTEGRATION LAYER: THE PROCESS SUPPORT ENVIRONMENT

Figure 3 illustrates how each of the three megaprogramming technology thrusts is brought to bear to engineer the product-line SEE. The SEE is depicted in the center of the diagram as two layers - a key architectural viewpoint that is elabo-

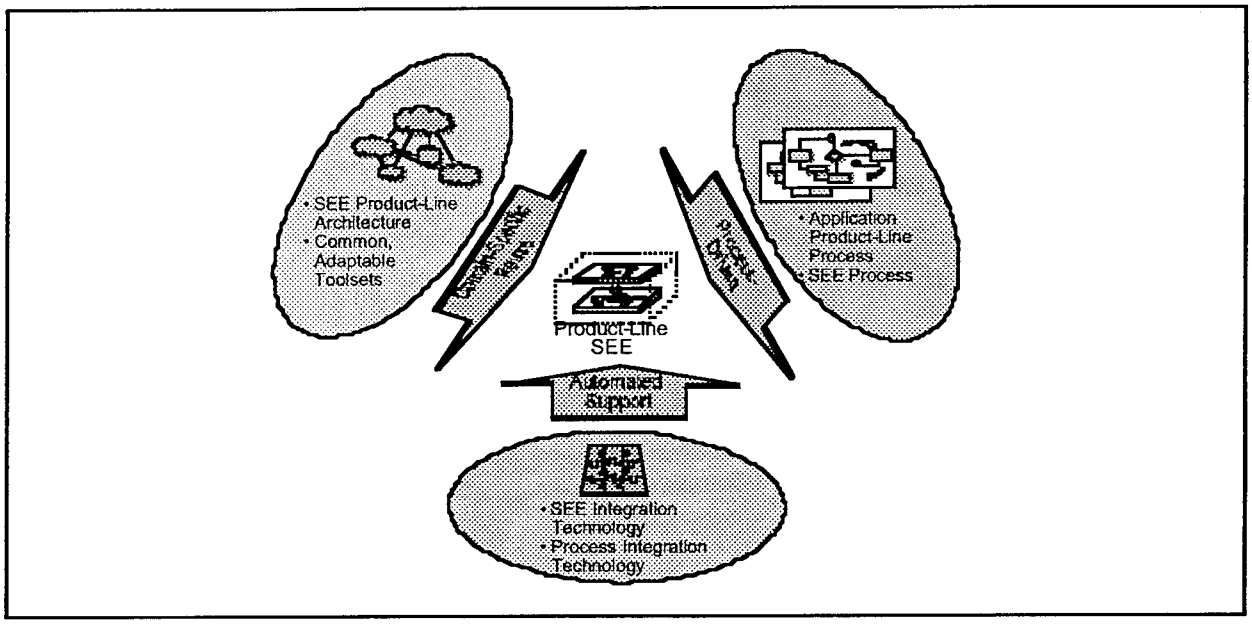


Figure 3. The Motivation for a PSE Encapsulation Layer

rated below.

A key consequence of this line of reasoning is the identification of an "encapsulation layer" that can be instrumental in SEE integration: the Process Support Environment (PSE). As depicted in Figure 4, the PSE is a set of tools and associated data that allows an organization to define, carry out, and improve its process. It also serves as an integration vehicle for orchestrating the rest of the SEE.

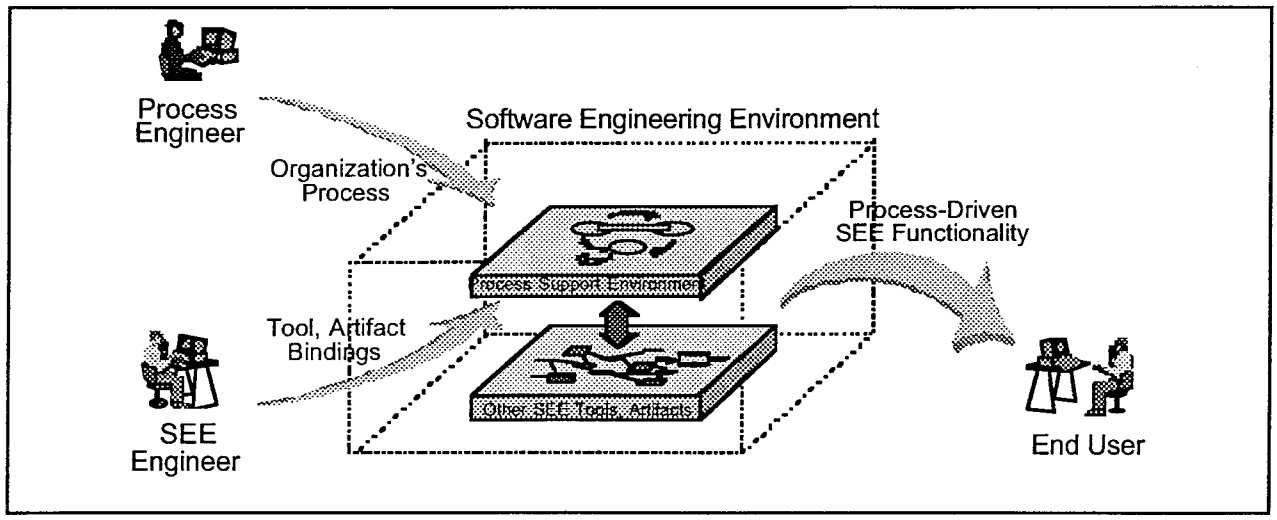


Figure 4. Conceptual Model for a Process Support Environment Encapsulation Layer

The result is a unique form of integration: process is integrated with the SEE, ***and*** the encoded process is then integrated with the rest of the SEE's tools and artifacts. Thus, ***two types of integration are occurring simultaneously***.

Figure 5 shows how the PSE layer fits in context with an abstract architecture for the SEE - depicted in terms of several other possible encapsulation layers that can be called upon to carry out process functions.

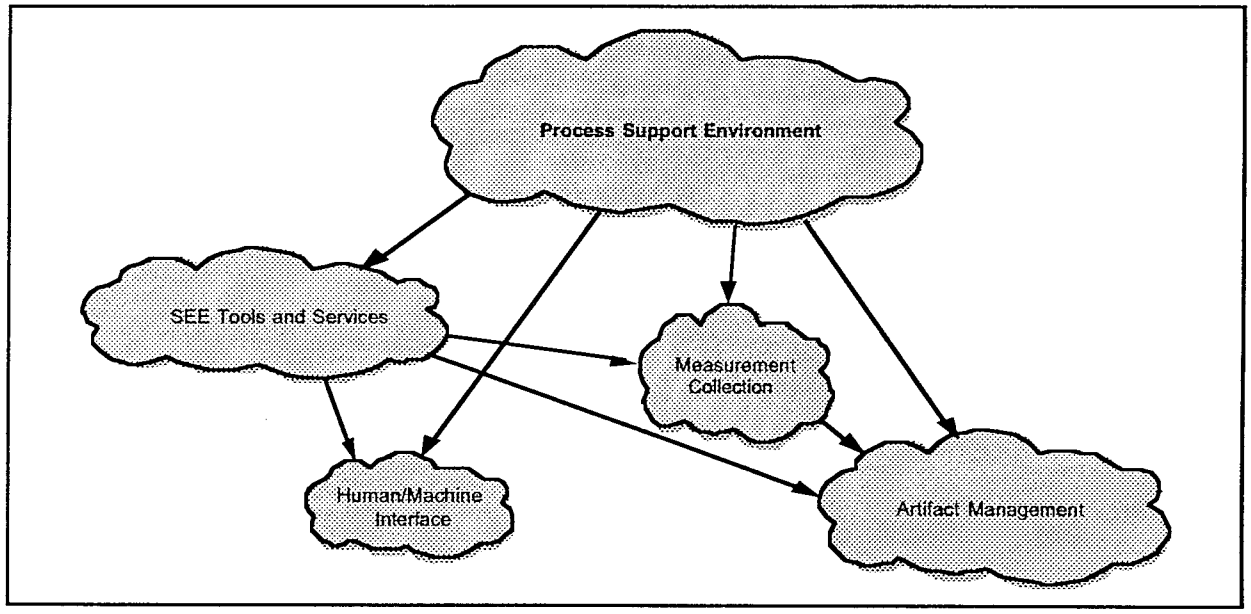


Figure 5. SEE Encapsulation Layers

Figure 6 illustrates the PSE currently in use on the Demonstration Project.

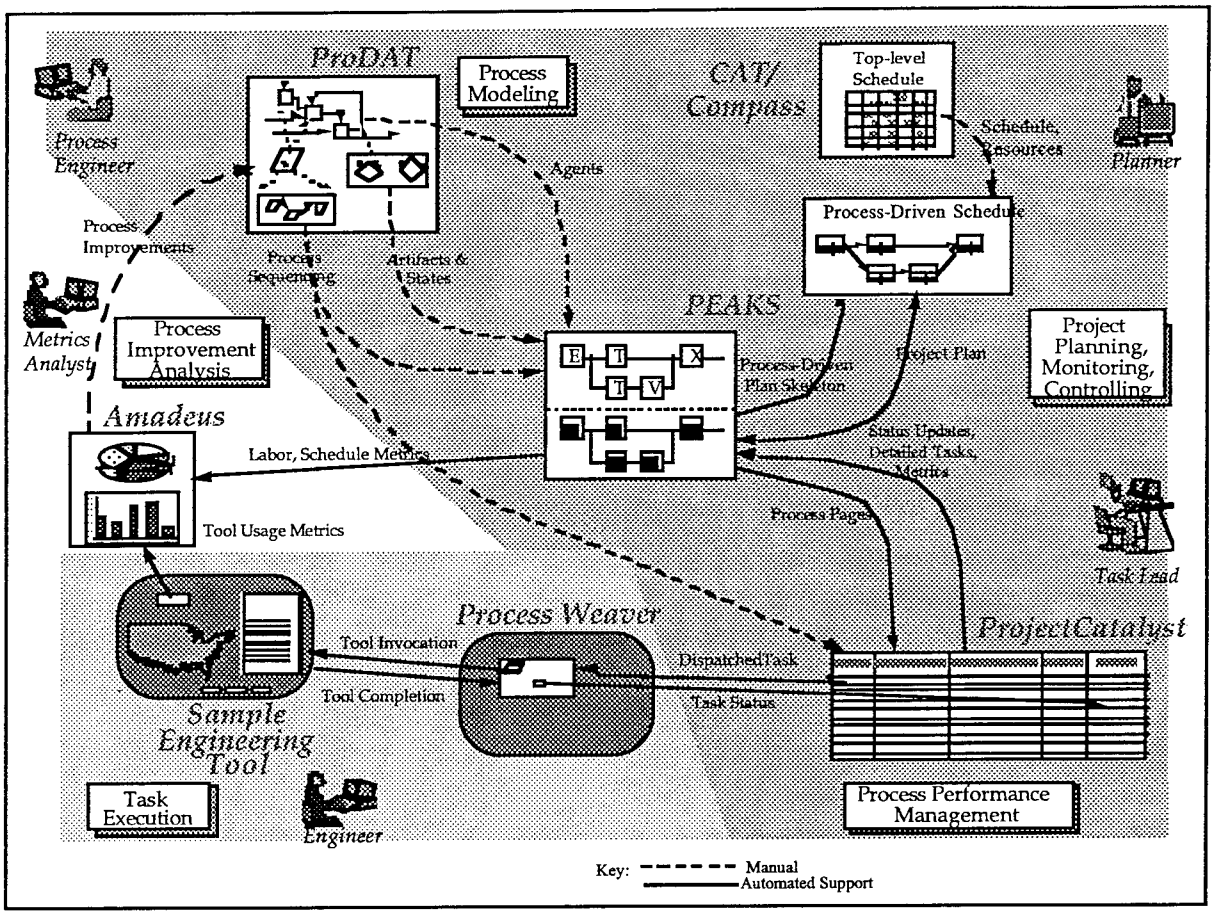


Figure 6. The Demonstration Project Process Support Environment

The PSE layer includes the STARS-sponsored tools PEAKS, ProjectCatalyst, and Amadeus. PEAKS is used to model the process using a graphical ETVX[4] notation and to construct a process-driven plan for the project. ProjectCatalyst [ProjCat94] is used to support the coordination and execution of the process-driven plan. To provide this support, ProjectCatalyst imports the PEAKS plan and presents it to task leads as a hierarchy of process component "pages". The pages list the work tasks, inputs, outputs, verification points, and lower-level process components called for by the process-driven plan. The task leads use these pages (which look much like spreadsheets) to dispatch tasks to engineers and to monitor status. They can also refine the pages to add additional steps not part of the formal process. As each task is dispatched to the designated engineer, ProjectCatalyst supplies a "work context" to Process Weaver, providing the engineer with convenient access to the artifacts (files, etc.) and the SEE tools to be used for the task. As tasks are completed and milestones are reached, status is reported dynamically to the PEAKS planning database -

[4] ETVX - Entry, Task, Validation, eXit - a notation for depicting process sequencing.

providing managers with up to the minute status of project activities. This closed-loop SEE mechanism assures the project that its work is taking place in accordance with the process and that status information is based on completion of prescribed validations. Further, as key points in the process execution are reached, measurement data (such as labor hours, elapsed clock time, automated quality measures, etc.) is captured in the Amadeus database, assuring that resulting metrics are similarly based on actual work processes - and allowing analysts to derive credible information to assist with process improvement.

The basic usage paradigm supported by the STARS PSE toolset seems promising to the end-users. Based on survey feedback, task leads seem at home with ProjectCatalyst's concise spreadsheet-like way of viewing their delegated work in progress and appreciate the automatic reporting of status based on the engineering work starts and stops. Engineers have indicated that they appreciate the structured way their tasks are maintained in Process Weaver agendas, as well as the ability to quickly navigate to their work product files from their work contexts.

As discussed in the Lessons Learned section, however, there are still implementation difficulties with the current tools that would have to be overcome before they could be considered for production use. As of this writing, while PEAKS is nearing commercialization, it appears that ProjectCatalyst will remain a prototype implementation. Current plans are to implement some of the ProjectCatalyst functionality in PEAKS and to continue to assess other avenues for supporting the automated enactment aspects of the PSE.

Although much work remains to fully realize a Process Support Environment, experience to date demonstrates the viability of the ideas. Interested readers should refer to the paper "Using Process to Integrate SEEs" [Randall95], also published in these proceedings - which further elaborates the motivation for the PSE layer, provides a model for its structure and interfaces, and discusses a number of lessons learned specific to integrating process and SEE.

To dramatize the relevance and timeliness of this work, the April, 1994 STSC Report on SEE Technology [STSC94] identified seven key SEE technology areas that have received insufficient attention to date and in which the industry now seems postured to make significant progress. Of these seven areas, the top three were process modeling, process definition, and process enactment and enforcement. The PSE encapsulation layer addresses all three of these areas.

6. Conclusion

We have used the three megaprogramming technology thrusts - domain-specific reuse, systematic process, and automated support - as the basis for organizing this paper. The Air Force/STARS Demonstration Project is in the midst of transitioning megaprogramming into practice, with the intent to parley the SCAI application into a product-line of similarly constructed systems. Although the team has been grappling for two years on how to build product-line applications, it was not until recently that it occurred to us to apply megaprogramming to the Software Engineering Environments supporting those applications.

In retrospect, this is no great revelation. In a product-line organization, each unique application will be built using a unique SEE. In both cases, however, our objective is to seek out - and exploit - commonality, to allow the systems (applications and supporting SEEs) to be constructed, maintained and evolved with maximum efficiency. Thus, to effectively support an application product-line, we believe a megaprogramming organization must engineer a SEE product-line.

We have used this thought pattern to re-examine the lessons learned in the SEE section of the most recent Demonstration Project Experience Report [DemExp95]. This paper cites 26 SEE lessons that we believe are relevant to organizations transitioning to megaprogramming - and elaborates 17 of them in some detail. It should be noted that many of the lessons are not unique to transitioning to megaprogramming - they apply equally well to organizations undergoing significant change to their business paradigms and wishing to transform their SEEs as well as their processes.

We closed the paper with a reflective section on megaprogramming and SEE integration, which we hope lays some of the theoretical groundwork for a more formal development of how to apply megaprogramming principles to the SEE product-line.

We intend to pursue the following during the remainder of the Demonstration Project and beyond:

- Further definition of theory and practice for a SEE product-line - notably domain engineering of the SEE domain;
- Further refinement of process and SEE integration - as introduced in Section 5.3, on page 27, and as discussed in detail in [Randall95], also published in these proceedings;
- Additional integration initiatives for the SCAI SEE - notably in the areas of process-driven project management, configuration management, metrics, and engineering artifact interrelationships;
- Evaluation of emerging SEE integration technology - such as Sun's ToolTalk and the Common Desktop Environment - with the intent of setting long-range SEE evolution priorities; and
- Participation with other SWSC groups - to communicate our approach, understand their approaches, and develop a unified SEE strategy across the organization.

REFERENCES

[Bristow95] Bristow D.J., Bulat B.G., Burton R. ***Product Line Process Development***, *Proceedings Seventh Annual Software Technology Conference*, Salt Lake City, UT, April 1995. (Published simultaneously with the present paper)

[Brown92] Brown, A. W., Earl A. N., and J. A. McDermid, ***Software Engineering Environments: Automated Support for Software Engineering***. McGraw-Hill Book Co., UK, 1992.

[Bulat95] Bulat B.G. ***SWSC Domain Engineering Experiences***, *Proceedings Seventh Annual Software Technology Conference*, Salt Lake City, UT, April 1995. (Published simultaneously with the present paper)

[DemExp95] ***AF/STARS Demonstration Project Experience Report, Version 2.0 (Draft)***, CDRL Sequence A011-002D, Electronic Systems Center, AFMC, USAF, December 1994 (currently under review for comment by the Government).

[NGCR93] ***Reference Model for Project Support Environments, Version 2.0 (Draft)***, Next Generation Computer Resources, 2 September 1993.

[NIST93] ***NIST Special Publication 500-211, Reference Model for Frameworks of Software Engineering Environments*** (Technical Report ECMA TR/55, 3rd ed.), National Institute of Standards and Technology, August 1993.

[ProjCat94] ***ProjectCatalyst Users Manual***, Software Engineering Technology, Inc., September 1994 (available upon request from SET, 2770 Indian River Blvd., Vero Beach, FL 32960).

[STSC94] Hanrahan R., Daud C., Meiser, K., Peterson J. ***Software Engineering Environment Technology Report***, Software Technology Support Center, OO-ALC/TISE, Hill AFB, Utah, April 1994.

[SchMel92] Shlaer S., Mellor S.J. ***Object Lifecycles, Modeling the World in States***, Yourdon Press, 1992.

[ToolTalk94] ***Common Desktop Environment: Getting Started Using ToolTalk Messaging***, Sun Microsystems Inc., Mountain View, CA, 1994.

[Trimble94] Trimble J. (ed.), ***STARS Program History 1983-1993 Version 1.1***, available from the STARS Program Office.

AUTHOR BIOGRAPHIES

Dr. Richard L. Randall

Dr. Randall is currently working on the ARPA STARS program as the on-site lead for the Air Force/STARS Demonstration Project at Peterson AFB, Colorado. He provides consultation to the Air Force on megaprogramming technology transition issues and on long-range strategies for enabling a future product-line for the Space and Warning Systems Center. His area of technical focus on the Demo Project is the integration of the Demo Project Software Engineering Environment - notably the aspects dealing with the organization's process and artifact management.

Dr. Randall's research interests include software engineering methods and integrated project support environments (IPSEs). He has over 25 years of experience in all aspects of software engineering for large real-time systems on projects such as Gemini, Apollo, Safeguard, Sea Nymph, and B2. During this time, he has focused on the integration of methods and tools into the software process, and he has played lead role in various division-level software and systems engineering steering groups.

Dr. Randall received a BS in Mathematics from MIT and a PhD in Computer Science from UCSD. He is a member of the IEEE Computer Society and the ACM.

Robert W. Ekman

Robert Ekman is a Senior Programmer in the Loral Federal Systems, Systems Technology and Products organization at Gaithersburg Maryland. He is currently lead engineer for the internal System Development Environments project. Previously he was a deputy architect on the IBM/Loral STARS program. Mr. Ekman has worked much of his thirty year career in software development environments and tools. He has been continuously involved in emerging software technologies. Recently, he has focused on development environment integration standards and techniques, and their relations to development processes.

Mr. Ekman received a BA in Physics from Gettysburg College. He is a member of the IEEE Computer Society and the ACM.

Captain Scott P. Kent, USAF

Captain Kent has over five years experience working for the U.S. Air Force on several command and control software development projects, including two years as a programmer. During this time he has devoted much of his attention to various aspects of software engineering environments - including selection, acquisition, integration, and management. Until April 1, 1995, Captain Kent was the Software Engineering Environment Team Lead on the AF/STARS Demonstration Project, which is demonstrating ARPA STARS megaprogramming technologies. He recently moved to Hanscom AFB working in the AWACS program office.

Captain Kent received a BS in Computer Science from the University of New Hampshire and is currently pursuing a Masters in Software Engineering from Colorado Technical College.

Gary S. Turner

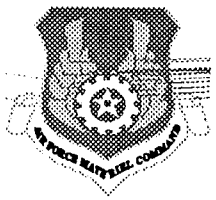
Gary S. Turner has 14 years of experience in all aspects of software engineering, including real-time systems, simulation, and software engineering environments. Mr. Turner is currently working on the ARPA STARS program and is an onsite representative supporting the demonstration of STARS megaprogramming technologies on a real DoD (Air Force) project. He has been working on the STARS project since its inception, and has contributed in the areas of software reuse, software engineering environments, and process-driven development. Currently his work focuses on the development and integration of process support tools.

Mr. Turner received both a BS and an MS degree in Information and Computer Science from Georgia Tech, with a masters thesis on the subject of rapid prototyping.

Integrating a SEE for Megaprogramming: Lessons Learned

7th Annual Software Technology Conference
Track 11 - Software Engineering Environments
Tuesday, 11 April 1995

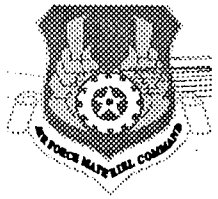
Dr. Richard Randall
Robert Ekman
Capt. Scott Kent, USAF
Gary Turner



LORAL
Federal Systems

Air Force/STARS Demonstration Project

LORAL
Federal Systems



Outline

AF/STARS SCAI

- Context
- Demonstration Project SEE
- Lessons Learned
- Megaprogramming and SEE Integration
- Conclusion

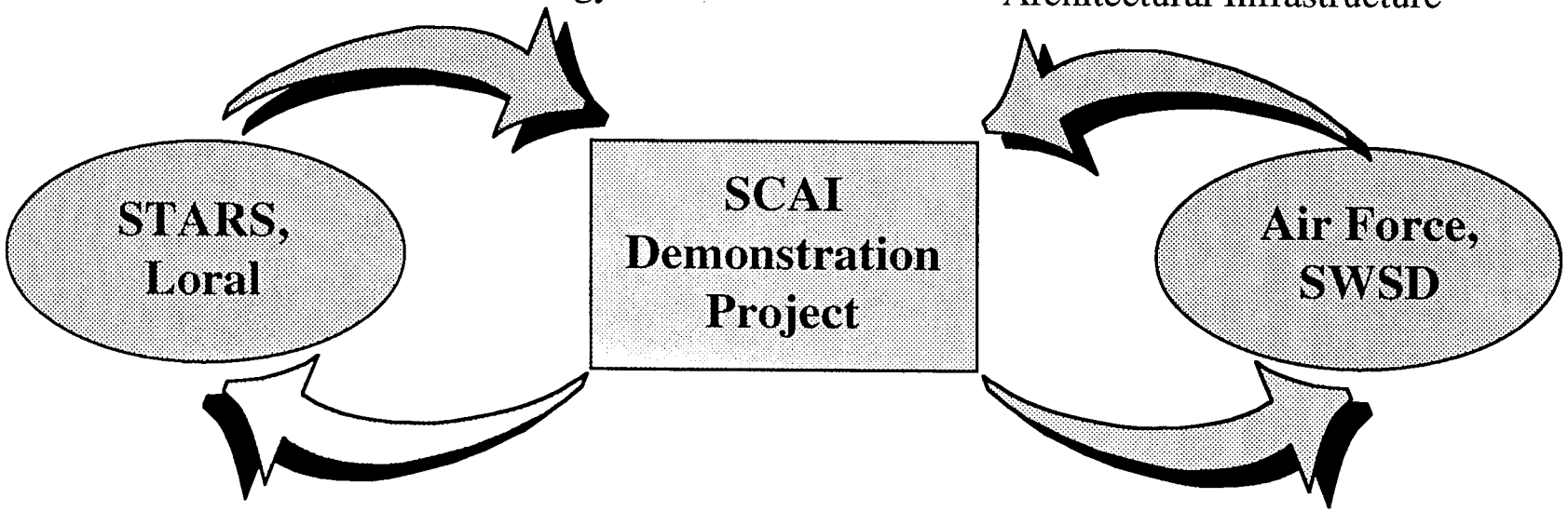
LORAL
Federal Systems

AF/STARS Demo Partnership

AF/STARS SCAI

- STARS Technology
- RISC/AIX Technology

- SWSD Technology
- Domain Expertise
- Architectural Infrastructure



- Enhanced Technology
- Real Project Experience

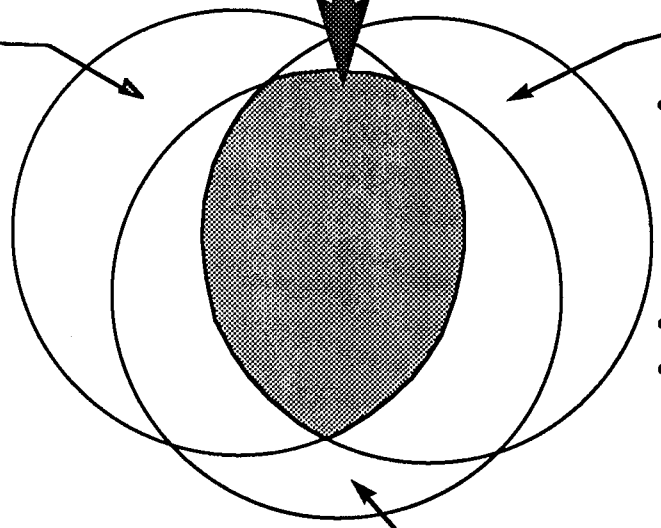
- Working Application
- Product-Line Foundation

LORAL
Federal Systems

STARS & Megaprogramming

AF/STARS SCAI

VISION: Efficient Product-Lines of Applications
MISSION: Accelerate Transition
Goals: Demonstrate Viability
Gain Experience
Establish Product-Line "Beachheads"



Domain-Specific Reuse-Based

- Guided by reuse process.
- Based on application domain architecture
- Systems composed from reusable assets
- Assets include all life-cycle artifacts
- Supports continuous improvement in reuse processes and products

Process-Driven

- Guided by a defined process
  - Developed from reusable process components
  - Adaptable to meet project goals
  - Promotes teamwork
- Supported by tools
- Supports continuous improvement in process and product

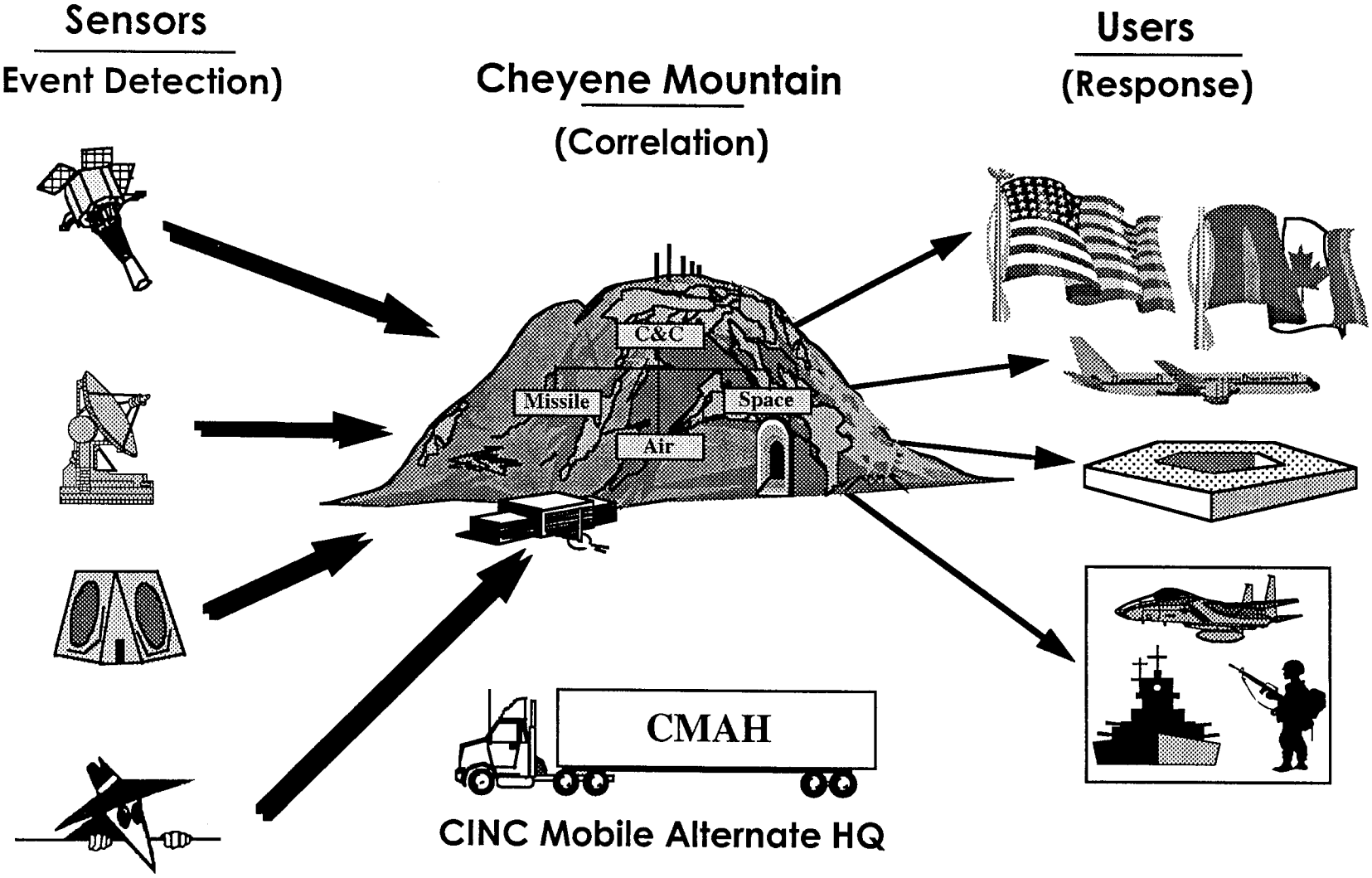
Technology Supported

- Process/Reuse Packaged in a SEE
- Based upon open architecture framework
- Adaptable approach for incorporating new technologies
- Continuous improvement in portability, adaptability reliability, and scalability

LORAL
Federal Systems

SWSD Mission: Support CMAS Major Systems

AF/STARS SCAI

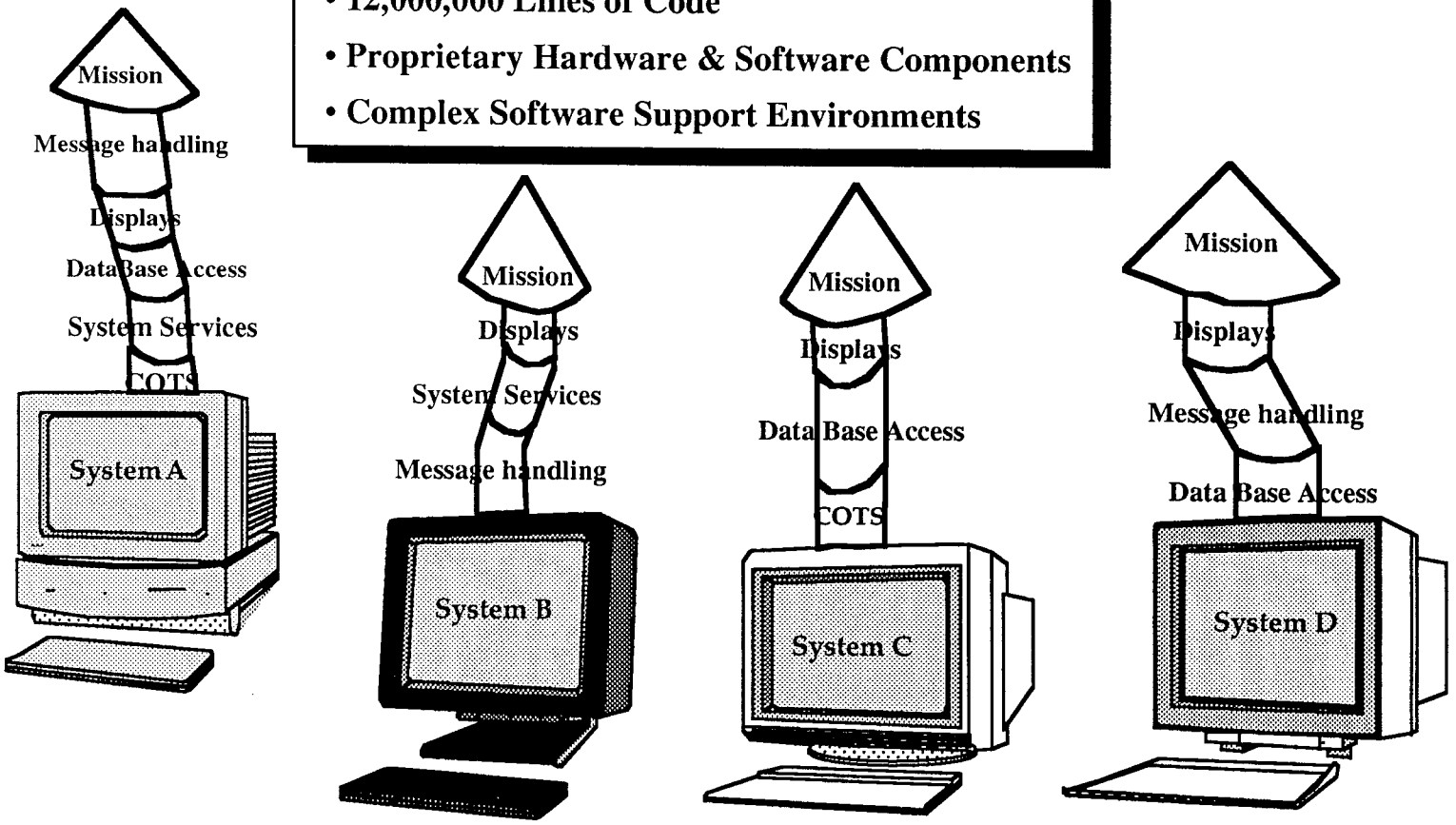


LORAL
Federal Systems

Problem Statement: Eliminating The Stovepipes

AF/STARS SCAI

- 34 Separate Operational Systems
- 27 Languages
- 12,000,000 Lines of Code
- Proprietary Hardware & Software Components
- Complex Software Support Environments

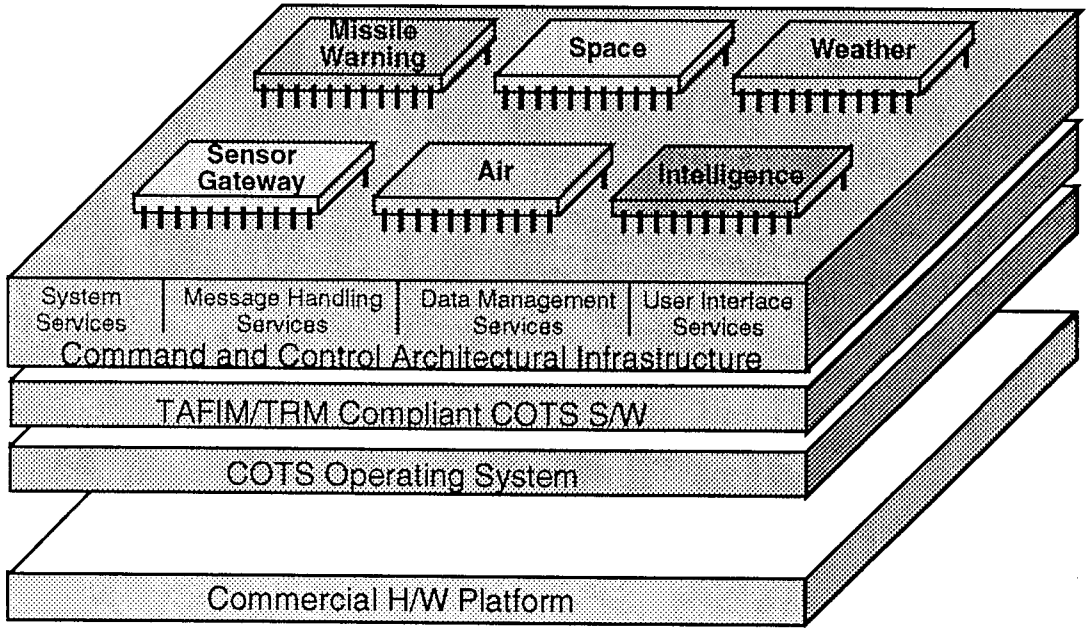


Affects Warfighters and Supporters

LORAL
Federal Systems

Product-Line Architecture

AF/STARS SCAI



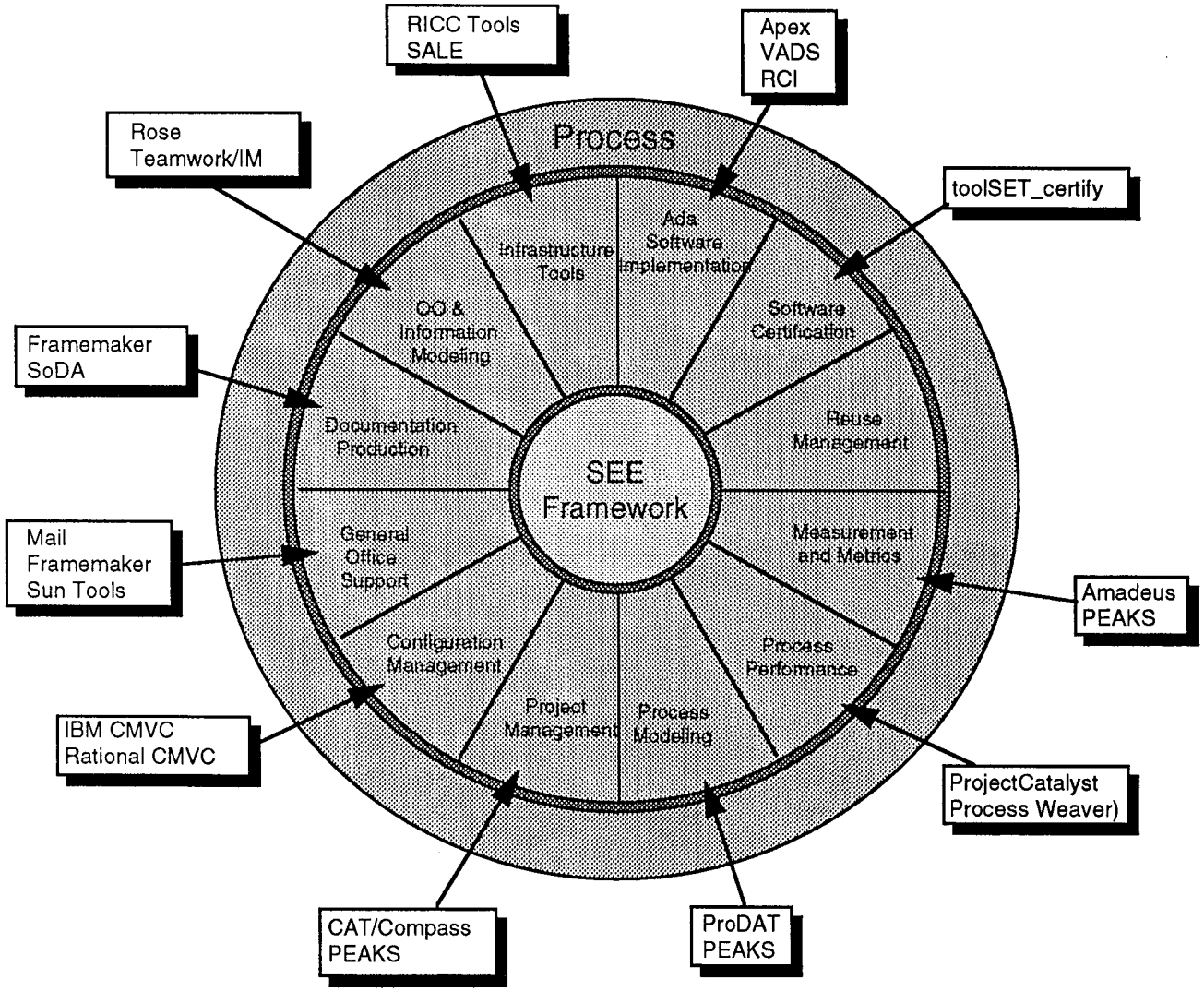
Basis for:

- Domain-Specific Reuse
- Process
- Automated Support

LORAL
Federal Systems

Demo Project SEE

AF/STARS SCAI



LORAL
Federal Systems

Roadblocks and Accomplishments

AF/STARS SCAI

- Roadblocks

  - Technology Transition: a two-way street

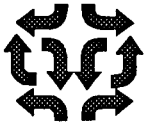
  - Rapidly evolving approach

  - Geographical separation

  - Classified SEE
  - Unrealistic SEE objectives

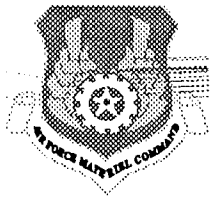
- Accomplishments

  - State-of-the-practice toolsets
  - Inroads in process automation



  - SEE "Adequate Functionality; Good Direction"
  - Enhanced expertise

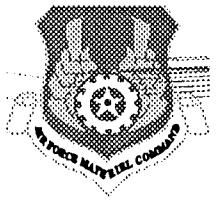
LORAL
Federal Systems



Lesson Categories

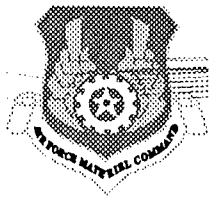
AF/STARS SCAI

- Domain Engineering
- SEE Management
- SEE Architecture
- Platform and Tool Selection
- SEE Integration
- SEE Improvement
- Technology Transition



SEE Management

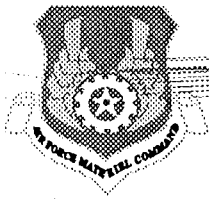
- Fluctuating SEE requirements
- Unexpected level of non-technical activities
  - Degree of change
  - Number of leading edge tools
- SEE "czar" needed
- Devoted SEE engineer needed
- Incremental approach is mandatory



Platform and Tool Selection

- The hidden value of heterogeneous platforms
- The necessity - and cost - of piloting
- The "bleeding edge"

LORAL
Federal Systems

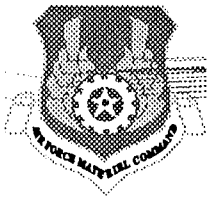


SEE Integration

AF/STARS
SCAI

- Impact of logistics
- Impact of infrastructure changes
- Value of "aligned toolset vendors"
- Pivotal technical issues:
  - Artifact management
  - Process
  - Metrics
- Multiple databases - a pervasive problem
- Pitfalls in control integration via messaging

LORAL
Federal Systems

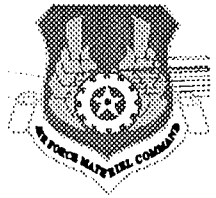


SEE Improvement

AF/STARS SCAI

- Project-wide tracking system
  - Action items, issues, problems, experiences
  - Managing SEE work orders
  - Feedback to vendors
- Process-driven databases: vital for metrics
- Surveys: individual capabilities & integration

LORAL
Federal Systems

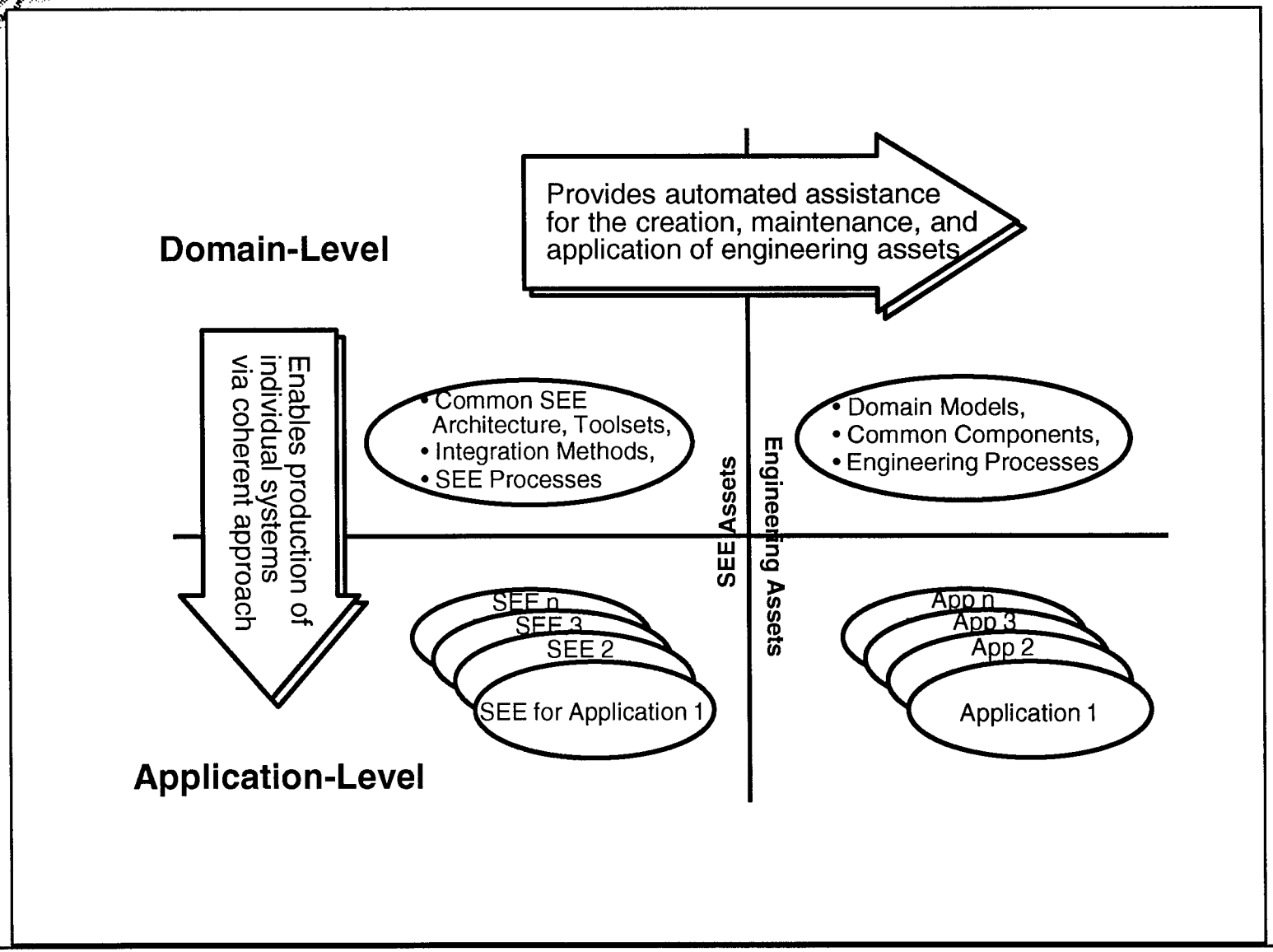


Megaprogramming and SEE Integration

AF/STARS SCAI

- The "SEE Product-Line"
- Applying Megaprogramming to the SEE

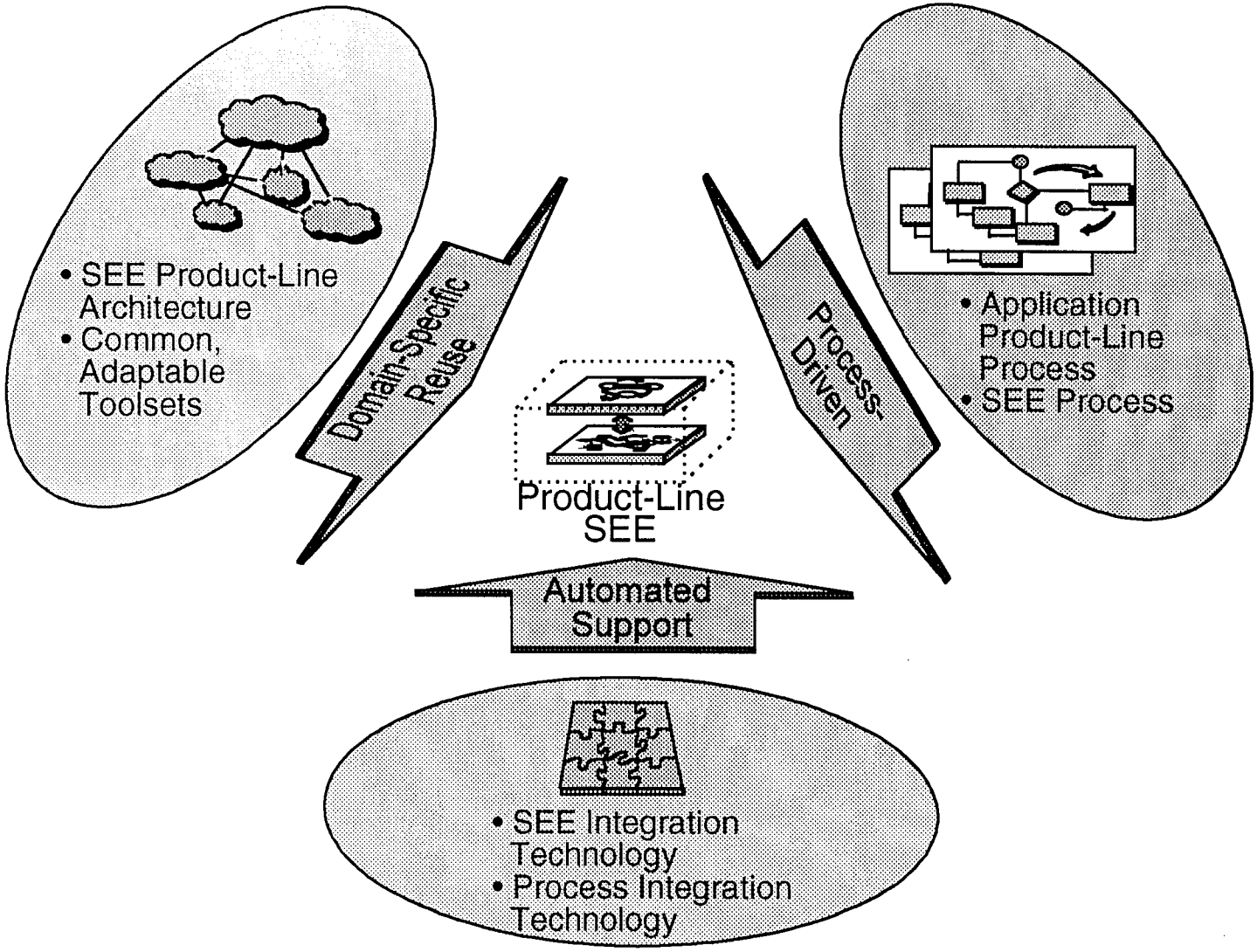
The SEE Product-Line



LORAL
Federal Systems

AF/STARS SCAI

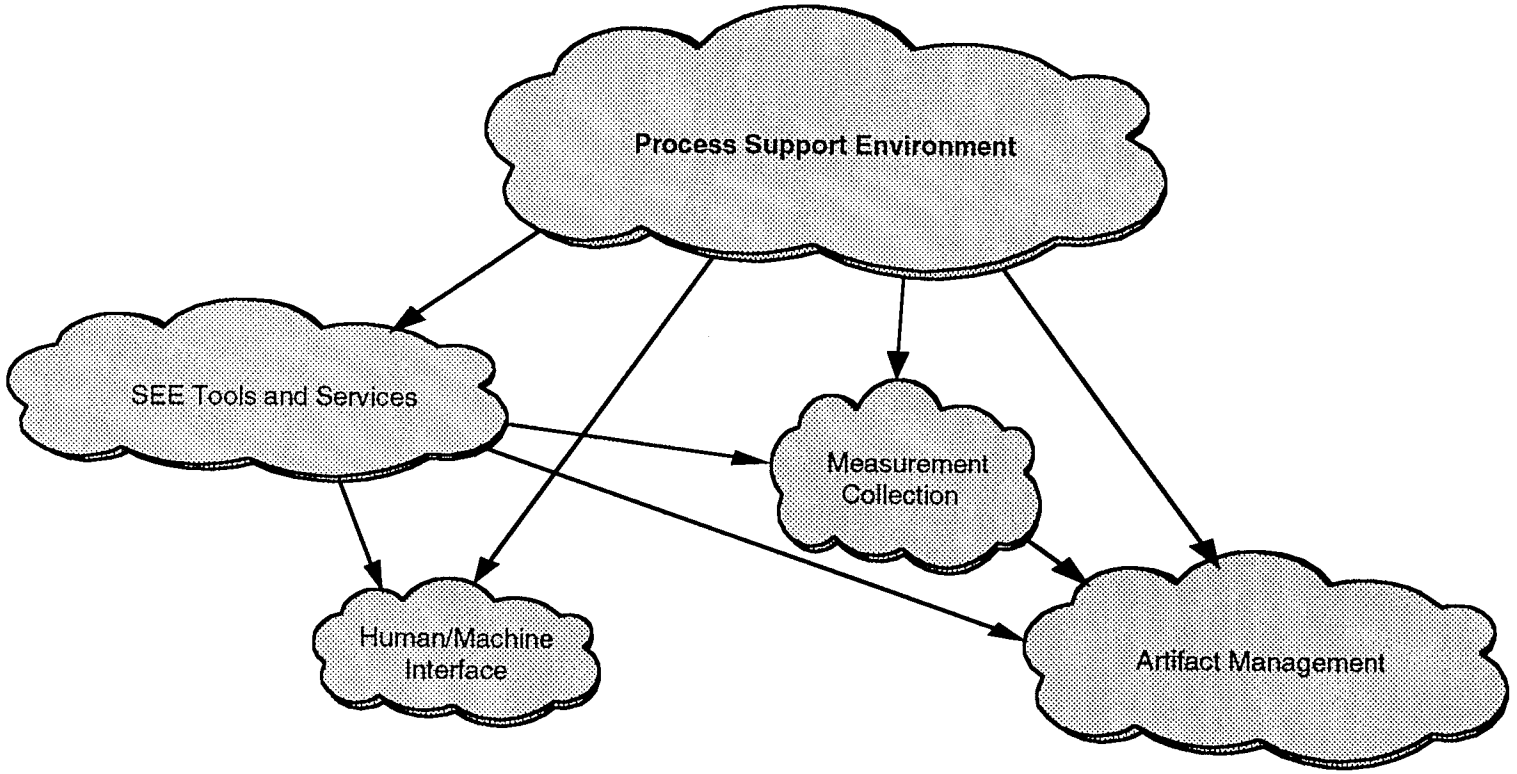
Megaprogramming the SEE



LORAL
Federal Systems

SEE Architectural Layers

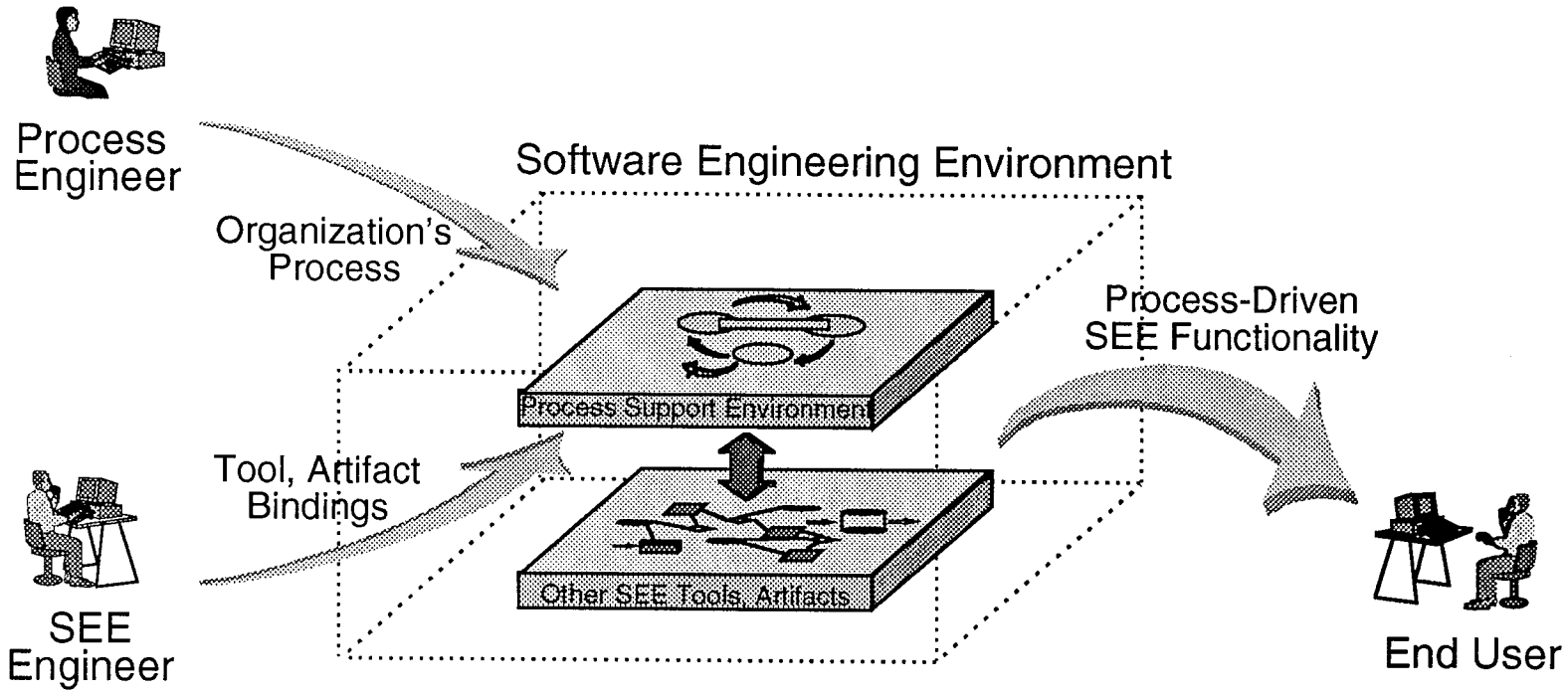
AF/STARS SCAI



LORAL
Federal Systems

Process/SEE Integration

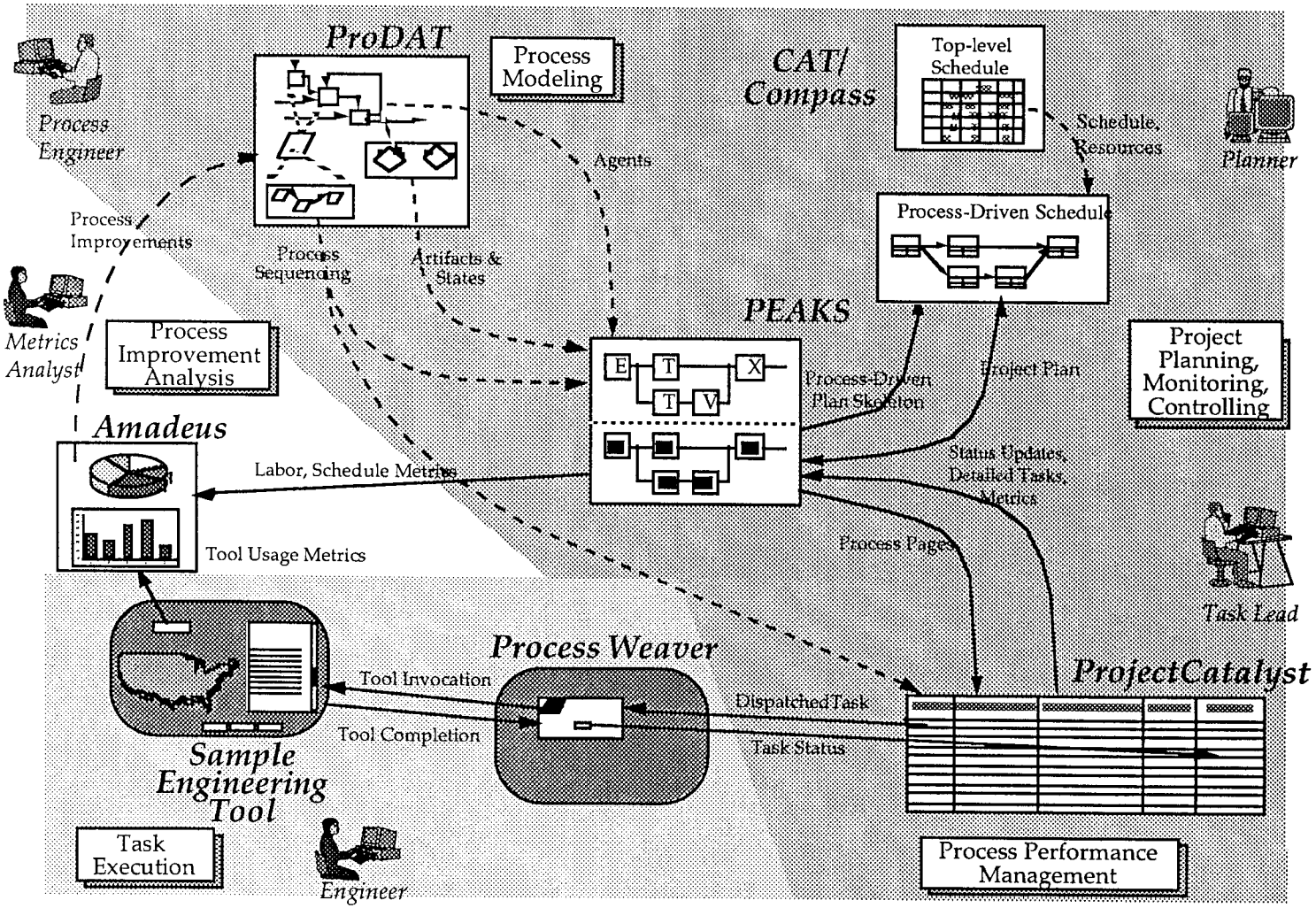
AF/STARS SCAI



LORAL
Federal Systems

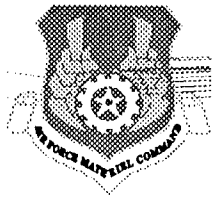
The SCAI PSE

AF/STARS SCAI



Key: - - - - - - Manual
——— Automated Support

LORAL
Federal Systems



Conclusion

AF/STARS SCAI

- Results
  - Experience in building SEEs in the midst of change
  - Viable SEE functionality, direction
  - Inroads in process automation
  - Recognition of "SEE product-line"
- Plans
  - Make selective improvements to SCAI SEE integration
  - Follow - and guide - emerging SEE technology
  - Participate in formulating organization-wide SEE strategy
  - Continue refining process automation methodology
  - Continue developing theory of SEE megaprogramming