Special Report
CMU/SEI-94-SR-008
November 1994

# Proceedings of the First Annual Software Engineering Techniques Workshop, May 1994: Software Reengineering

**Leonard Green**
John Bergey
Walter Lamia
Dennis Smith

Reengineering Center

**Software Engineering Institute**
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/ENS
5 Eglin Street
Hanscom AFB, MA 01731-2116

**Review and Approval**

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

(signature on file)

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

# Table of Contents

## List of Figures

# List of Tables

# Proceedings of the May 1994 Reengineering Workshop

**Abstract:** This is a report of the proceedings of the May 1994 Software Engineering Institute's Software Engineering Techniques Workshop on Software Reengineering. The report includes brief biographies of the workshop speakers and authors, and detailed accounts of nine working group session discussions on the following topics: reengineering architecture, decision analysis, system evolution and design records, reengineering economics analysis, understanding legacy systems, using lessons learned from other software reengineering projects, reengineering process models, software reuse, and reengineering technology and tools. The workshop established a foundation for capturing the best practices within reengineering and resulted in a detailed outline for a reengineering best practices handbook.

# 1 Introduction

## 1.1 History

The Reengineering Center was created at the Software Engineering Institute (SEI) to address the growing needs of the community in dealing with legacy software systems. Since the cost of maintaining legacy software systems is increasing along with the need to maximize the return on investment due to declining budgets, alternate solutions to replacing legacy software systems with newly developed software systems must be devised. Reengineering legacy software systems is one alternative that could decrease the normal costs incurred by developing new software systems.

The mission of the Reengineering Center is to ascertain the community's issues and needs, and then develop strategic plan(s) to address them accordingly. The Reengineering Workshop held in October 1993 was the center's first endeavor in surveying the issues and needs of the community. As a direct result of the October workshop, a preliminary set of strategic plans (short and long term) have been established. These plans will be focused in the general areas of:

- Foundation issues
- Technology issues
- Management issues
- Transition issues

The strategic plans of the center encompass such (leveraged/non-leveraged) activities as:

- Workshops
- Best practices

- Lessons learned
- Roadmap

## 1.2 Purpose

The purpose of the May 1994 Reengineering Workshop was to continue the momentum and efforts of the October 1993 workshop by addressing the needs and issues of the community (government, industry, and academe) and fostering an awareness of the supporting activities happening within the community.[1] The workshop's primary goal was to focus on specific areas or aspects of reengineering legacy software systems and to establish a foundation for capturing the best practices within reengineering. This was accomplished by requiring the various working groups to develop a detailed chapter outline for their respective area to possibly be included into a best practices handbook. A "merged draft outline," which represents an outline template for the handbook from the results of all the working group efforts, can be obtained from the Software Engineering Institute.[2]

The information captured in these minutes is organized chronologically, in the sequence of the workshop's events.

## 1.3 Future Plans

The Reengineering Center has drafted a preliminary set of long-range goals that may involve other SEI projects and/or external organizations. One such goal is the publication of a best practices handbook in reengineering. The Reengineering Center plans to accomplish this goal primarily by two mechanisms:

- Use data collected from reengineering case studies.
- Establish self-sustaining work groups to address specific areas/topics in reengineering.

Supplemental to collecting best practices from case studies, the center plans to develop a lessons learned interview framework that may identify a set of essential data that may be required to use the data effectively. In addition, there may be plans to develop a roadmap to help guide perspective users through predictable situations/scenarios that may arise in reengineering legacy software systems.

1. Workshop presentation materials can be found in *Software Engineering Techniques Workshop on Software Reengineering*. For a copy of the book, contact Customer Relations at the Software Engineering Institute at **customer-relations@sei.cmu.edu**.

2. Contact Customer Relations at **customer-relations@sei.cmu.edu**.

# 2 Plenary Sessions

## 2.1 Overview

The May 1994 Reengineering Workshop began with three plenary sessions given by distinguished guest speakers from the reengineering community. A list of the speakers and abstracts from their presentations given on the first day of the workshop is presented below.[1]

## 2.2 First Session

**Speaker:** William Ulrich, Tactical Strategy Group Inc.

**Abstract:** Software reengineering of MIS systems has matured in recent years. Once viewed only as a tactic for rewriting existing packages, software reengineering is now seen as a strategic, enabling technology that supports a wide range of projects. Factors contributing to this include evolving techniques, maturing technology, and an overall re-positioning of software reengineering in the IS marketplace. This (keynote) address considered:

- MIS reengineering requirements and industry background.
- Evolution of techniques, frameworks, and tools.
- Project scenarios benefiting from software reengineering.
- Industry directions: where will we be in 3-5 years?

## 2.3 Second Session

**Speaker:** William Carlson, Intermetrics Inc.

**Abstract:** This keynote address was on software renovation, architecture, and city planning of mission-critical computer resources. Mr. Carlson asserted that "all software is a subsystem of an organizational process and/or hardware system," and "software engineering is not very different from other engineering disciplines." The other forms of engineering disciplines have defined classes of structure or product line architectures. Thus, software should be developed or reengineered into model-based or generic architectures, which go through continuous improvement, to be used and reused over time, taking into account a more evolutionary strategy/plan.

## 2.4 Third Session

**Speaker:** Dennis Smith, Software Engineering Institute

1. A list of the presentation materials can be found in *Software Engineering Techniques Workshop on Software Reengineering*. For a copy of the book, contact Customer Relations at the Software Engineering Institute at **customer-relations@sei.cmu.edu**.

**Abstract:** The keynote address gave an introduction to the Reengineering Center, the center's current status, and the objective of the Reengineering Workshop. In addition to talking about the ongoing activities within the community, the keynote speaker expounded on the center's mission and its perspective in supporting the reengineering community, not by inventing but by leveraging off the past. The workshops (in October 1993 and May 1994) were some of the mechanisms by which this would occur. The ultimate goal of the May workshop was to leverage off the experiences of organizations within the reengineering community in developing a best practices guide.

# 3 Parallel Sessions

## 3.1 Overview

The Reengineering Center sponsored morning and afternoon parallel sessions on the first day of the workshop. These parallel sessions were given by various members of the reengineering community. A listing of the authors and abstracts from their proposals/papers presented at the workshop is given below.[1]

## 3.2 Morning Sessions

### 3.2.1 Session 1a, Application of Integrated Process Definition to Reverse Engineering

**Author(s):** Frank Svoboda and Carol Klinger (Unisys Government Systems)

**Abstract:** Clear, comprehensive definition is essential in establishing an enactable, measurable, and repeatable reverse engineering process. Process definition that is represented using complementary, textual, and graphic notations (integrated process definition) enhances clarity and helps to insure coverage of significant process issues. The benefits of integrated process definition apply to reengineering and its component processes, reverse engineering and forward engineering. Reverse engineering is generally performed in an ad-hoc manner, often lagging behind forward engineering in incorporating new technology such as integrated process definition. The Army/STARS/Unisys Demonstration Project has successfully applied integrated process definition to promote understanding of its reverse engineering process and to communicate this understanding among the project team and to external stakeholders. This paper highlights the project's reverse engineering process and the impact of a integrated approach to the definition of that process.

### 3.2.2 Session 1b, A Cost Modeling Formalism for Reengineering Decisions

**Author(s):** Benjamin Keller (Virginia Polytechnic Institute & State University)

**Abstract:** The role of cost models in software reengineering (indeed all of software engineering) is to support decision making. One useful application of cost models is the selection of reengineering strategies for a particular task. A decision model for strategy selection requires different cost models for each strategy in order to differentiate between them. A cost modeling formalism designed to support definition of such cost models is presented.

1. A list of the presentation materials can be found in *Software Engineering Techniques Workshop on Software Reengineering*. For a copy of the book, contact Customer Relations at the Software Engineering Institute at **customer-relations@sei.cmu.edu**.

### 3.2.3 Session 2a, Reengineering the Software Life Cycle

**Author(s):** Noah Prywes and J. Ahrens (Computer Command and Control Company)

**Abstract:** Current software engineering environments (SEEs) must be extensively reengineered because they do not support today's entire software life cycle. In particular, they do not adequately support software reengineering and software maintenance, precisely those areas that account for the great majority of today's software labor. A corollary to this is the need to support the evolving software life cycle by integrating software reengineering environments with software engineering environments (Ahrens et al., 1994). This paper explores this topic and the corresponding issues involved.

### 3.2.4 Session 2b, Software Systems Reengineering Process Model

**Author(s):** Tamra Moore (Defense Information Systems Agency)

**Abstract:** The Center for Information Management (CIM) Software Systems Reengineering Process Model provides guidance for applying software reengineering technology for the development and modernization of automated information systems (AISs) within the Department of Defense (DoD). This presentation describes the CIM software reengineering process as defined in the model, which is composed of activities for creating AISs to support current business needs. The CIM software reengineering process is represented using the IDEF0 Activity Modeling technique, the DoD standard for process modeling. The intended audience for the model is any organization within the DoD that must reengineer AISs.

### 3.2.5 Session 2c, Software Reengineering Risks Taxonomy

**Author(s):** Joan Smith (Defense Information Systems Agency)

**Abstract:** Software reengineering is increasingly becoming an important information system function as systems development and maintenance organization attempt to contain maintenance costs, preserve investments in their software assets, and modernize their information systems. Enormous investments in DoD AISs has created formidable maintenance and evolution obligations. Even with shrinking software budgets, the DoD AISs are in more demand than ever. Yet current AISs are aging and becoming outdated. Costs of maintaining these systems escalate to the point of inhibiting their evolution. It is also recognized that many of these systems replicate functions in other AISs in the DoD. Consolidation of these systems appear to have potential for reducing costs.

The Software Reengineering Risks Taxonomy Report addresses software reengineering risks identification within the context of DoD AISs. It provides a taxonomy of five reengineering risk categories:

- Planning
- Process

- Personnel
- Product
- Technology

Although this software reengineering taxonomy is relatively comprehensive, it is not exhaustive and does not cover all risks. However, taxonomy does offer a program or project manager facing software reengineering decisions a good starting point from which to identify critical software reengineering risks.

### 3.2.6 Session 3a, Maintenance Measurement and Reengineering ROI

**Author(s):** Robert Arnold (Software Evolution Technology Inc.)

**Abstract:** Software maintenance measurement is the collection and analysis of data about software maintenance activities and products to improve maintenance performance relative to business objectives. Return on investment (ROI) for reengineering is the calculation of payoffs points for reengineering activities and is to be used to decide if reengineering efforts should be undertaken, and to monitor reengineering actual activities with cost estimates. Software maintenance measurement is important to make maintenance activities and effects more visible. This is a prerequisite for monitoring and for maintenance process improvement activities. It is also integral to moving an organization to higher SEI capability maturity levels. Reengineering ROI calculation is important for encouraging economics and concrete planning to play key roles in reengineering activities. The theme of this presentation focuses on two major points. One is on understanding how cost models can help you predict and measure a return on investment for reengineering. The second is on getting estimates that can help you decide on reengineering options.

### 3.2.7 Session 3b, Overview of an Enabling Technology for Software Reengineering

**Author(s):** Lawrence Markosian (Reasoning Systems)

**Abstract:** Over the past 10 years there has been a proliferation of CASE tools for developing new software. In contrast, computer support for reengineering existing systems, particularly large legacy systems, has been minimal. Software tools have been developed for reverse engineering, that is, for analyzing code to produce structure charts and other reports. However, these tools usually don’t quite meet a project’s requirements. They may not work with the particular language dialect, or they don’t perform the required analysis. Worse yet, they can’t be modified--the vendors don’t supply an interface for building customizations and extensions.

This problem is exacerbated when we look for automated support for reengineering, that is, transforming existing code to meet new requirements. Here, we are confronted with the cross product of the range of dialects and subsystems (databases, screen generators, etc.) in the existing system, and the range of possible dialects and subsystems for the reengineered system.

Even when no porting is required, the requirements that the reengineered code must meet - and hence the modifications that are required - are often dictated by the characteristics of a particular project. This paper is a proposal for a presentation introducing a new technology for software reengineering.

## 3.3 Afternoon Sessions

### 3.3.1 Session 4a, System Diagnostic Approach to Reengineering Legacy Systems

**Author(s):** Lewis Hayes, Anthony Pease, and Paul Oliver (Booz Allen & Hamilton Inc.)

**Abstract:** This paper presents an approach to maximizing the future value of legacy systems through a software reengineering diagnostic process and approach. The Systems Diagnostic assists the information systems (IS) manager in planning and executing a successful modernization program. It describes how the Systems Diagnostic process has been conceived from a repository-based information model, and how it has been implemented with a combination of off-the-shelf tools, custom interfaces, and an internally developed expert system, while prior experience with software reengineering has provided the foundation for our expert system. The results of the Systems Diagnostic can be used as the basis for successful modernization of a wide variety of legacy systems.

### 3.3.2 Session 4b, Use of Cleanroom Technology to a Software Support Activity

**Author(s):** Wayne Sherer (Software Technology for Adaptable, Reliable Systems (STARS) DPM, Army), Paul Arnold (Loral Federal Systems), and Ara Kouchakdjian (Software Engineering Technology)

**Abstract:** This presentation presents the results and lessons learned from a STARs sponsored process technology transfer demonstration. The Armament Munitions and Chemical Command (AMCCOM) Life Cycle Software Engineering at Picatinny Arsenal was selected to demonstrate that Cleanroom Software Engineering (CSE) could be successfully applied to a typical DoD Software Support Activity (SSA) Software Reengineering Effort. Results indicate that:

- CSE practices and process-guided project management can be successfully transferred to a DoD SSA.
- Engineering staff productivity and quality was increased while simultaneously increasing job satisfaction.
- Return on investment of a least 9:1 has been realized on the first project to which CSE reengineering techniques were applied.

### 3.3.3 Session 4c, Experiences Using a Library of Reengineering Process Models

**Author(s):** Robert Arnold (Software Evolution Technology Inc.)

**Abstract:** The purpose of this talk is to explain the differences between creating a set of process models and applying them in practice. In particular, one has to decide on how they will be specified, who will use them, and how and when to use them. For example, often the models are more educational than prescriptive for end users. Having an expert in reengineering to help users understand and apply the models is very useful. It is useful to view the process models as covering a baseline set of reengineering scenarios, which provide real-world problem entry points for end users. The relevant process model is then tailored for the user's specific situation. Another point is that the models need to be connected to selections of the tools, so that the process can be readily 'tooled up' at a moment's notice. Planning for this cannot be haphazard. Getting the model 'right' (i.e., represented/described appropriately to satisfy any user's level of experience in applying the model) is an evolving process. An issue is how detailed to make the steps described in the models. Experts can get by with high-level instructions like "institute configuration management." Some inexperienced users need more detailed explanations of the steps until their experience allows them to work with less prescriptive descriptions of steps.

### 3.3.4 Session 5a, Reengineering Legacy Systems: Case Study of Software Redevelopment and Business Process Improvement

**Author(s):** Reginald Hobbs and Glenn Racine (Army Research Laboratory--Software Technology Branch)

**Abstract:** This paper describes a software reengineering project whose purpose was to reengineer a legacy COBOL systems application for implementation into Ada. As a result of the reengineered system, the material status reporting process has been greatly improved at over 60 Army installations. This business process improvement was realized by creating a system that is structured, easier to maintain, downsized from mainframe-to-desktop, and easier to install and maintain. The paper describes and evaluates the reengineering strategy employed, discusses the use of automation in this type of transition effort, and assesses the organizational impact of the project in terms of cost/benefits analysis.

### 3.3.5 Session 5b, Software Reengineering with Legacy Systems

**Author(s):** Grace Hammonds (AGCS), Randall Lichota (Hughes), and Robert Vesprini (Raytheon)

**Abstract:** The upgrade of legacy systems is a recurrent problem in the DoD. A typical legacy system represents a considerable amount of time, money, and effort and therefore one could expect it to form the basis for any upgrade to that system. Unfortunately, the reusability of legacy software varies significantly, depending on the quality of documentation, the viability of existing support tools, and the nature of the upgrade itself. In general, the latter may involve rehosting the software on a new machine, adding additional features to the software, integration with another system, or some combination of the three. An upgrade may also require interpretability with other DoD systems, architectural migration to a distributed computing environment, adherence to established and emerging standards, and integration with off-the-shelf products. Finally, security considerations may dictate that the software execute under stringent access controls, such as occurs in multilevel security situations.

The Portable, Reusable, Integrated, Software Modules (PRISM) Program is an Air Force Electronic Systems Center (ESC) Software Center (ESC/ENS) initiative to eliminate long procurement cycles and large expenses involved in the custom development of one-of-a-kind command centers. As an alternative to traditional requirement analysis, PRISM uses a process-driven approach to command and control software development based on user prototypes. On the one hand, this process provides users with an alternative to a complete replacement of the legacy system or, on the other hand, to a difficult modification of the current system. PRISM employs a domain-specific architectural framework within which a prototype may be rapidly created using mega-programming techniques. This prototype, while presenting interfaces and functionality equivalent to a given legacy system, is composed of modular software building blocks that correspond to components in the architectural framework. The functionality of these building blocks may be provided by code drawn. This paper discusses two examples in which the PRISM team is addressing the issue of legacy system upgrades.

### 3.3.6 Session 5c, Domain Analysis and Reverse Engineering

**Author(s):** Spencer Rugaber (Georgia Institute of Technology)

**Abstract:** Reverse engineering takes a program and constructs a high level representation useful for documentation, maintenance, or reuse. To accomplish this, most current reverse engineering techniques begin by analyzing a program’s structure. The structure is determined by lexical, syntactic, and semantic rules for legal program constructs. Because we know how to do these kinds of analyses quite well, it is natural to try to apply them in understanding a program.

But knowledge of program structures alone is insufficient to achieve understanding, just as knowing the rules of grammar for English are not sufficient to understand essays or articles or stories. Imagine trying to understand a program in which all identifiers have been systematically replaced by random names and in which all indentation and comments have been removed. The task would be difficult if not impossible.

The problem is that programs have a purpose; their job is to compute something. And for the computation to be of value, the program must model or approximate some aspect of the real world. To the extent that the model is accurate, the program will succeed in accomplishing its purpose. To the extent that the model is comprehended by the reverse engineer, the process of understanding the program will be eased. In order to understand a program,

therefore, it makes sense to try to understand its context: that part of the world it is modeling.

Given that the source code by itself is not sufficient to understand the program and given that traditional forms of documentation are not likely to provide the information needed, the question arises whether there is an alternate approach better suited to the needs of reverse engineering. This essay argues that application domain modeling provides such an approach.

### 3.3.7 Session 6a, Interactive Reengineering Tool for Constructive Language Translation

**Author(s):** Daniel Wilkening and Joseph Loyall (The Analytical Sciences Corporation), Marc Pitarys and Kenneth Littlejohn (USAF Wright Laboratory)

**Abstract:** The United States Air Force’s Wright Laboratory and The Analytical Sciences Corporation (TASC) are developing an environment for reengineering of software from one language to another. Our approach engineers a program in the new language by reusing portions of the original implementation and design. We use reverse engineering and computer-assisted restructuring to help the engineer construct and refine a program using design and implementation information from the original system. We use automatic translation of low-level program statements to free the engineer from the tedium associated with syntactic differences between languages. This paper outlines the architecture of our reengineering tool, describes the process by which an engineer uses the tool, and explains some problems that we have discovered in converting a sample application.

### 3.3.8 Session 6b, Software Migration: Software Performance Engineering for Legacy Systems

**Author(s):** Debra Carr (Electronic Data Systems)

**Abstract:** The development world is much more complex than it was just a few years ago. In the past, monolithic application development and maintenance were possible with relatively few tools. Today, fast changing business requirements and advances in technology are causing companies to transform their business processes, organizations, and compute and communications platforms, as well as applications functionality, application development environments, and system data. ...Only by addressing all of these areas can businesses successfully transform their legacy systems. Software Performance Engineering for Legacy Systems (SPELS) is a process for improving and/or migrating legacy systems. This process includes activities, tasks, and deliverables and is supported by recommended tools where appropriate. The SPELS process has been used successfully to improve legacy systems by redirecting up to 50 percent of traditional maintenance investments toward future development projects. This overview describes how SPELS can maximize the investment embedded in legacy systems by transitioning application functionality, application development environments, and system data into new technologies and open systems.

### 3.3.9 Session 6c, A Testbed to Study the Reengineering Process

**Author(s):** Angelika Zobel (Siemens Corp.) and Bernd Bruegge (Carnegie Mellon University)

**Abstract:** In the past, software reengineering has been associated with legacy systems. Some properties of legacy systems make it difficult to study the reengineering process: they are usually large, and therefore any reengineering project is bound to take a long time. Consequently, the turnaround time of one entire reengineering cycle is too large to be useful to study the reengineering process in practice.

To date, the reengineering process is ill-defined, and there is not even clarity about the phases that contribute to the entire process. We feel that to obtain a realistic process model, we must study real reengineering projects. To obtain a reasonable project turnaround time, a controlled setting is needed. By that we mean three things. First, the project must be time-boxed. Second, the discussions that lead to decision making in the project must be well documented. Third, and most importantly, the software system to be reengineered must be small enough to allow a timely reengineering turnaround, yet it must have legacy system properties such that a realistic assessment of the reengineering process is possible.

The main problem is to find software that is not too complex, so that the reengineering turnaround time is reasonable, yet it must exhibit legacy system properties. What are the main characteristics of a legacy system? One essential legacy system property is that the original design is lost and must be recovered. Our position is that we can find this characteristic in a specific class of rapid prototypes.

In this paper, we introduce a software reengineering testbed, which we have established in the framework of a software engineering undergraduate course at Carnegie Mellon University. We describe the setting of our first reengineering process assessment experiment conducted by developing an accident management system from a model-based rapid prototype.

# 4 Reengineering Community Progress Report

## 4.1 Overview

A progress report of selected activities within the reengineering community was given by the Software Technology Support Center (STSC), the Defense Manage Review (DMR) Group, and IEEE. A summary of their presentations is given below.[1]

## 4.2 STSC Perspective

Chris Sittenauer and Mike Olsem gave a 15-minute presentation on the current status of the following STSC activities:

- Software Reengineering Assessment Handbook
- Reengineering Tools Data Base
- JOVIAL Reverse Engineering ToolSet
- Reengineering Tutorial
- STSC's 9 Steps When Preparing to Reengineer

## 4.3 DMR Group/IEEE Perspective

Elliot Chikofsky gave a 15-minute presentation on the ongoing and upcoming activities (conferences, newsletters, etc.) of the DMR Group and the IEEE Society.

1. A list of the presentation materials can be found in *Software Engineering Techniques Workshop on Software Reengineering*. For a copy of the book, contact Customer Relations at the Software Engineering Institute at **customer-relations@sei.cmu.edu**.

# 5 Panel Discussion

## 5.1 Panelists

1. Evan Lock, Computer Communication and Control Center (CCCC) / Lead
2. Grady Campbell, Software Productivity Consortium (SPC)
3. Elliot Chikofsky, DMR Group
4. Harry Crisp, Naval Surface Warfare Center (NSWC)
5. Julia McCreary, Internal Revenue Service (IRS)
6. Bernard Zaranski, James Martin & Co.

## 5.2 Scope

The focus of the panel was on characterizing reengineering in the management information systems (MIS) and mission-critical computer resources (MCCR) communities and promoting understanding of each other's needs. This included the identification of their unique aspects, areas of greatest concern, unique and common opportunities and challenges, and areas where they can cooperatively work together to their mutual benefit.

## 5.3 Summary

The panel started off with a 15-minute presentation by Julia McCreary representing the characteristics and issues from an MIS perspective. Afterward, Harry Crisp gave a 15-minute presentation that represented the characteristics and issues from an MCCR perspective.[1] Their presentations were followed by a brief perspective from the remaining panel members. Finally, the panel opened the floor for questions and comments from the audience.

1. A list of the presentation slides can be found in *Software Engineering Techniques Workshop on Software Reengineering*. For a copy of the book, contact Customer Relations at the Software Engineering Institute at **customer-relations@sei.cmu.edu**.

# 6 Working Group Sessions

## 6.1 Mission

The mission of the working groups was to generate a detailed outline for each of their respective areas that would ultimately be developed into a chapter that could be included in a reengineering best practices handbook. Information would not be limited to DoD-related issues, but to the private sector and academia as well. It was also hoped that each working group would continue its efforts independently of the Software Engineering Institute to instantiate the detailed outline with the appropriate information in creating a proposed chapter for inclusion within the handbook.

The subject areas (see below) that were chosen for the working groups were conceived from the combined efforts of members of the reengineering community and the Software Engineering Institute's Reengineering Center. It should be noted that these "chosen" subject areas do not reflect the complete list of topics to be covered by the reengineering best practices handbook.

### 6.1.1 Reengineering Working Group Descriptions

The following is a listing of each of nine working groups convened concurrently at the reengineering workshop and their proposed scope of operation:

- **Architecture**. This group will focus on the aspects of architecture and its involvement with reengineering (e.g., recapturing the architecture from a legacy software system). They may also address the roles of domain analysis/model-based engineering, evolutionary systems, and open systems with respect to legacy systems.
- **Decision Analysis**. With greater frequency, reengineering is being considered as a strategy for system evolution. There are many decisions that must be confronted to resolve issue such as "which system" and "how" to reengineer. Once the basic decision to reengineer has been made, a series of more detailed decisions must be addressed. For example, there are choices with regard to methods, tools/technology, and transition plans. The primary focus of this working group will be to elaborate on a key set of non-economic decisions to be made and to suggest a set of guidelines for improving the decision-making process. The group will review current practices as a basis for identifying critical success factors and needed improvements.
- **Design Record**. The motivation for the creation of a design record has primarily been to capture the "design decisions" that led to a particular design instantiation. The primary focus for this working group will be to identify and discuss the issues and criteria associated with defining the contents of a design record to support reengineering. The group will consider the

effectiveness and value-added benefits of design records, the information boundaries of design records, criteria for selecting design record information, and cost-effective means of overcoming barriers such as collecting and maintaining up-to-date design record data.

- **Economic Analysis**. This group will focus on the economics of reengineering legacy software. Specifically, the group will investigate the economics of reengineering decision strategies and process models and synthesize this information into a decision process for use by the appropriate software practitioners and decision makers. Pertinent case studies and existing/ongoing work in software reengineering assessment will be evaluated. Issues to be discussed and commented upon will include:

  What is the framework for discussing reengineering costs?

  What is the process for performing an economic assessment of reengineering projects and strategies?

  What is the process for determining the economic indicators (return on investment, etc.)?

  What is the process for selecting from among multiple projects and strategies?

  What tools are needed for evaluating costs for reengineering versus continued maintenance of the legacy system or development of a new system? (This issue will include reviews of existing cost models and identification of general metrics and guidelines for using the models.)

  Can a model be formulated to predict reengineering costs?

- **Understanding Legacy**. This group will examine the issues involved in the comprehension of legacy software systems. They will investigate the required aspects or artifacts for understanding legacy, supportive research trends or methodologies, capturing requirements, etc.

- **Using Lessons Learned**. This working group will consider how to effectively use information about the prior experiences of other projects which have undertaken software reengineering efforts of one sort or another. Some of the topics to be discussed are:

  What types of data are important to know about referenced projects? (What data are irrelevant?)

  What decisions would you make differently if you had access to this information?

  What level of detail about the data is necessary?

  How can confidentiality of sources be maintained?

  How (i.e., what mechanisms) are needed to ask additional questions to get more detail?

What forms and formats would be most effective for communicating and distributing lessons learned information?

What access methods are most interesting/useful/effective?

- **Process Models**. This group will focus on the models which defined the process steps for reengineering legacy software systems. The group will examine the issues and the needs of a process model, fundamental steps of a model, their relationship to/impact on common life cycle models, the impact of cleanroom technology, and the Capability Maturity Model (CMM) relationship.
- **Reuse**. This working group will address the relationship between reuse and reengineering. They will discuss common issues, concerns, and benefits for reusable artifacts (e.g., code, design, requirements), reuse-driven reengineering, developing product families, etc.
- **Technology and Tools**. This working group will examine the state of technology to support the reengineering process, including support for reengineering methods, capture and analysis of information, automated transformation, and the evolution of target architectures. Topics will include tool capabilities and their assessment, the reengineering environment versus the existing and target environments, repositories and storage/analysis requirements, CASE support, and migration to new architectures such as client/server and open systems.

## 6.2 Architecture Working Group Session

### 6.2.1 Participants

1. Paul Clements, Software Engineering Institute (technical lead)
2. Dennis Smith, Software Engineering Institute (facilitator)
3. David Carney, Software Engineering Institute (minutes taker)
4. Phillip Bonesteele, FileNet Corp.
5. Alan Kocka, Grumman
6. Randall Lichota, Hanscom

   He is a senior systems engineer with Hughes Technical Services Company and is currently a member of the Portable Reusable Integrated Software Modules (PRISM) project. As a member of PRISM, he has been involved in software process definition, software architecture and performance modeling, and systems engineering. Dr. Lichota is the author of the PRISM Qualification Methodology report and has contributed to the definition of the PRISM reusability process. He is currently involved in an evaluation of the Open Software Foundation's (OSF's) Distributed Processing Environment (DCE) and Distributed Management Environment (DME) for application to the command center domain. Prior to his assignment on PRISM, Dr. Lichota was Hughes' resident affiliate at the Software Engineering Institute, where he contributed to the development of the Durra language and tools. Previously, he conducted research in real-time parallel processing systems, distributed Ada-based systems, and specification techniques for real-time systems. He provided consulting services for Hughes' parent company, General Motors.

7. Mike Rice, Defense Information Systems Agency / Joint Logistics Command (DISA/JLC)
8. Spencer Rugaber, Georgia Institute of Technology

   Dr. Rugaber received his Ph.D. from Yale University in 1978 where he worked with Dr. Alan Perlis. His thesis was titled "A Model of the Understandability of Computer Program." Since receiving his Ph.D., Dr. Rugaber has worked in industry: for Bell Laboratories, where he was responsible for the development of programming language tools; for Interactive Systems Corp., where he was one of the developers of TEN/PLUS, an integrated environment for interacting with the UNIX operating system; with Sperry Corp. where he was responsible for the development of environments for improving software productivity; and with National Advanced Systems, where he was a technical product planner. Dr. Rugaber has been at Georgia Tech since January, 1988. During that time, he has been a member of the Software Research Center. His research has concentrated on software maintenance and reverse engineering. He is currently Vice Chair of the Reverse Engineering Subcommittee of the IEEE Technical Committee on Software Engineering, and program co-chair for the Third Workshop on Program Comprehension.

9. Robert Vesprini, Hanscom

   He is a Principal Engineer with the Raytheon Company and is currently a member of the PRISM project. As a member of PRISM, he has been involved in the development of the Generic Command Center Prototype, the incorporation of new technology into that prototype, performance analysis using discrete event simulation, and product assessment. Prior to this he has worked on radar systems, command and control systems, Ada technology, Kalman filtering, simulation, and numerical analysis.

10. Ron Weller, Science Applications International Corporation (SAIC)

### 6.2.2 Introduction

Preliminary discussion focussed on high-level and general issues, i.e., the overall purpose for this working group and the scope of its activity. The first issue is a direct result of the previous workshop, when many participants indicated the need for discussion about the relationships between architecture and reengineering. We noted that there has been little work to date, so much of the discussion would be essentially speculative.

We began with the notion that in reengineering systems, there is a need to separate some common areas of those systems. We realized also the need to distinguish between reengineering a single system as opposed to reengineering a family of systems. The goal is to address some large percentage of the system requirements. This led to an agreement that we need a list of the interesting features of a system: how do we recognize an “architecture”?

We considered four key aspects of reengineering:

1. Elicit information from the existing system.
2. Analyze it.
3. Capture the results of the analysis.
4. Validate that the captured information is correct.

At this point, it was necessary to attempt a definition of the term “architecture,” both because it was necessary in order to continue, but also to determine the degree of consistency in the working group. For example, architecture, depending on how it is defined, could refer to something that provides a framework for reengineering or, viewed differently, could be something that reengineering seeks to discover.

The definition was achieved through a series of assertions about architecture:

- Typically, one would move some system toward a new architecture, possibly one that exists through some family of systems, because the old system has degraded in some way.
- The goals of the reengineering effort have an impact: is it simply providing documentation? a little tweaking? optimizing? total overhaul?

- The applicability of “architectural” notions is tied to the many different reengineering strategies.
- The key question that must be answered is: is there some intrinsic property of a system that is not discernible from its components, standards, and interfaces? (An example of this can be found in the F14 executive, where much of the knowledge of the system was buried in the data tables used by the application; reengineering through code translation would not capture such knowledge.)
- An “architecture” includes a philosophy that a design imposes on a system. This notion can be seen in “real” architecture, where a portal or doorway typically has an area in front of it that reflects assumptions the designer makes about how the doorway will be used. Notions such as these are often not written down, yet are major factors in the design of the original system.

The working group then discussed a proposed definition by Ron Weller. After some revision, there was general agreement on the following definition:

> **Architecture.** A description of a system in terms of a collection of abstractions that characterize its structure and other attributes necessary to specify, design, and construct or implement that system.

Further discussion centered on the general agreement that although “architecture” is exceedingly difficult to define, it [architecture] forms a part of the solution set, as opposed to the problem statement. For reengineering, this implies that the backward motion from the existing system eventually leaves the realm of the solution (i.e., the existing system) and enters the realm of the original problem. We firmly agreed that the notion of “architecture” lies on the solution side (although it may conceptually be very close to the dividing line).

### 6.2.3 Draft Outline for Best Practices

The questions were ordered into a list (see below), which forms the general outline of the book chapter. Each of these is expanded by considering how the issue was addressed in the past, how it is addressed in the present, and some possible ways the issue may be considered in the future.

## I. When is architecture a concern for reengineering?

A. **Past:** Even though systems were redesigned, there was little attempt made to salvage (e.g., save) their architecture. The effort was essentially redevelopment, and was centered on salvaging the functional behavior of the old system.

B. **Present:** This may depend on different circumstances. A present system’s architecture may be an issue when:

1. The hardware architecture changes, through addition or replacement.
2. Moving to a GUI design, or moving from a batch to a client-server or an O-O paradigm.
3. The system needs an improvement in performance.

4. The needs of the system’s resources change.
5. There is need for expanded functionality of the system.
6. There is need for fuller understanding of the system (e.g., through domain analysis).
7. Re-documentation is necessary.

**C.** **Future:** We envision the notion of a family of products that share certain architectural aspects.

## II. How can we determine (or bound) the problem domain?

**A.** **Past:** This was done through informal means. We relied on the experience of technical “gurus.” The degree of formality related to the specific problem at hand.

**B.** **Present:** We now attempt to deal with problems through reuse libraries. There is also sporadic informal use of artifacts that exhibit cross-domain characteristics (i.e., multiple inheritance) and of various generators.

**C.** **Future:** We need to begin to exploit domain information; to use “generator generators”; and to use domain models to understand systems and help in extracting, analyzing, representing, and validating architectures (see the following four sections).

## III. How do we extract architecture from existing systems?

**A.** **Past:** We extracted such items as:

1. Call graphs
2. Data flow diagrams
3. Cross reference lists
4. Flowcharts
5. Decision tables
6. State machines

We also relied on conversations with the system’s builders and on running the existing code.

**B.** **Present:** Many of the above extractions are automated (e.g., software refinery); the use of query languages (though knowledge of queries tends to be ad hoc) is also emerging. Other techniques include slicing, and plan/cliche recognition.

**C.** **Future:**

1. We need domain-specific query systems.
2. We need to learn to make inferences based on the above artifacts, and inferences that are derived from the interaction of those structures.

3. We need a transformation capability (in the sense of transforming these pieces of existing architectures to new ones).
4. We need to learn to extract the important parts of the existing architecture.

## IV. How do we represent architectures?

**A.** **Past:** We generally used the same set of artifacts as in Section III above, e.g., call graphs.

**B.** **Present:** O-O frameworks such as Directed Graphs, or Logical Nets. The capabilities of tools such as the reusability library framework (RLF) developer through the STARS program is also useful in this regard. We are also seeing the growth of architectural representation languages and of state charts and mode charts.

**C.** **Future:** We need a notion of a "standard set" of views or abstractions of a system. These would provide the minimal set of abstractions necessary to build the system. This can be expressed as a function:

fn: Goals of reengineering ------> The set of views/the information needed

We also need the notion of "visualization," i.e., that of an "integrated" set of n relevant views.

## V. How do we analyze architectures?

**A.** **Past:** The principal means was a set of metrics about the extracted information (i.e., the list of artifacts in Sections III and IV).

**B.** **Present:** We rely on the following:

1. Metrics
2. Simulation
3. Maintenance estimates
4. Portability checkers
5. Reliability analysis
6. Performance monitors
7. Conformance to standards
8. Runtime assertion monitors

**C.** **Future:** There is need for verification and theorem-proving methods. In general, we would like to analyze the architecture for satisfaction of requirements (functional, modifiability, performance, security, reliability) and for conformance to standards. We would also like to do run-time load swapping and detection of intermodule constraints.

**Note:** We note that the notion of “analyze for {n} characteristic” may in fact have some correlation with the “set of abstractions” from our definition of architecture; thus, the set of abstractions may be defined for that set of characteristics that we wish to analyze.

## VI. How do we validate the “captured” architecture?

**A.** **Past:** We used such techniques as:

1. Coupling and cohesion
2. Design reviews
3. Walkthroughs

We also relied on face-to-face discussion with the original engineers.

**B.** **Present:** We rely on such techniques as portability checking or standards checking. We also rely on the analysis features noted in the previous section.

**C.** **Future:** We need to develop techniques to find and document the “philosophy” of the original architecture. We also need to find ways to determine the conformity of the new architecture to the original creators’ style. We also need greater technology for tracing requirements.

**Note:** During discussion of this item, we noted a very probable link between these issues and those that were being discussed in the Design Record and the Legacy Systems working groups.

## VII. How do we “re-architect” a system? And validate the new architecture?

**Note:** This notion both embraces a simple “tweaking” of the architecture and can extend to throwing the entire thing out and starting over again.

**A.** **Past:** This was accomplished through ad hoc means, generally through all of the representation and analysis techniques described in the previous sections.

**B.** **Present:** We often rely on a “firewall” technique: for instance, if a system is moving toward an O-O structure, to use “wrappers” around pieces of the system. For the issue of validation, which is separable from that of “re-architecting,” we use black-box testing, regression testing, and test case generators.

**C.** **Future:** We have need of application generators (both automatic and semi-automatic). We also need transformation technologies and repositories of domain-specific software architectures.

**Note:** At this point, we considered the question of distinguishing "re-architecting" from "reengineering" and also from "reuse." There was only one agreement: it is not a binary issue. The line between any of these activities is fuzzy at best, and it is probably not of value to attempt to place hard distinctions between them.

## VIII. What is the "rearchitecting" process?

**Note:** This question and the following one were not broken into "Past-Present-Future."

We considered that the "re-architecting" process consisted of the steps outlined above in Sections III through VII. We also noted the following needs:

- Enfranchise the mountaineers of systems.
- Revise DISA's reengineering manual for architecture-specific issues.

We also noted during this discussion a very probable link to the discussions of the Process working group.

## IX. What tools are useful for this process?

We noted the need for:

- Executable specifications
- Domain-specific tools
- Many needs in the area of tools integration

### 6.2.4 Key Issues Raised by the Working Group

**Note:** These issues were collected and organized by Spencer Rugaber.

#### 6.2.4.1 Definitional Questions

- What is an appropriate definition for the term "architecture"?
- How do reengineering, maintenance, and reuse differ in their relationship with architecture?
- Where does enhancement leave off and re-architecture begin?
- Is it possible to relate architectural variants? That is, is there a concept of architectural purity?

#### 6.2.4.2 The Relationship of Architectures to Domains

- What is the relationship of a domain model to an architecture?
- How should domain information be used in reengineering?

- Under what circumstances does it make sense to obtain and use domain knowledge to support a reengineering effort?
- What differences result from reengineering a family of applications versus re-engineering a single application?
- How should domain information from various sources be integrated?
- How should knowledge from multiple domains be combined in re-engineering an application?
- How should organizational information be integrated into a comprehensive data architecture for the organization?

#### 6.2.4.3 Process Issues

- How do architectural differences effect a reengineering effort? Are there specific aspects of an architecture that play a dominant role in a re-engineering effort?
- And under what circumstances should an architecture be used in the re-engineering process?
- When is re-architecting appropriate versus continued enhancement? When is architectural “tweaking” appropriate versus throwing out the code and starting over?
- How much detail is appropriate in an architectural description that will be used to support reengineering?
- How can we detect and extract architectural information from an existing system? What should be done when an existing architecture has degraded so much it is hard to extract or modify? Can we detect such degradation?
- Our architectural reengineering process includes steps for extraction, analysis, representation, and validation. Under what circumstances should these steps be applied?
- What is the relationship of reengineering process to tools? Can we use tools in the absence of process? Must we build custom tools to support our process?
- How can we validate our understanding of the architectural aspects of an existing system?

#### 6.2.4.4 Representational Issues

- How should architectures be represented and documented?
- During initial design and implementation, what information should be left behind for the developers of the next revision?
- What is a necessary/sufficient representation of an architecture to support re-engineering?
- When we extract architectural information from an existing system, how will the information be accessed during reengineering?

## 6.3 Decision Analysis Working Group Session

### 6.3.1 Participants

1. Evan Lock, Computer Communication and Control Center (CCCC) (technical lead)

2. Anne Quinn, Defense Logistics Agency / Software Engineering Institute (minutes taker)

3. Harry Crisp, Naval Surface Warfare Center (NSWC)

4. Joan Smith, Defense Information Systems Agency / Center for Information Management (DISA/CIM)

   Ms. Smith is a computer specialist in the Software Systems Engineering Directorate, Reengineering Division, (TXER) of DISA, Joint Interoperability and Engineering Organization, CIM. Ms. Smith worked on software development projects in Ada and ORACLE in support of the Joint Staff at DISA. She was a team member and project manager of the in-house initial development of the Worldwide Command, Control and Communications (WWMCSS) Automated Configuration Management System. Ms. Smith was in the Department of the Army Intern Program where she worked on the Army Consolidated Personnel System (ACPERS). Previously, Ms. Smith was the Program Coordinator for the Military District of Washington Family Member Education and Employment Resource Center where she formulated the implementation plan for this program.

5. Mike Olsem, Software Technology Support Center (STSC)

   He is a principal investigator with the Science Applications International Corporation (SAIC). As a technical staff member with the STSC reengineering research group, he maintains a reengineering tools data base and is developing a repository of reengineering project data to assist in the development of the Software Reengineering Assessment Handbook (SRAH). Mr. Olsem has made assessments of DoD organizations' software maintenance processes and recommended reengineering strategies. He has co-authored STSC reports, and he has written about reengineering and knowledge-based technologies for IEEE's Reverse Engineering Newsletter and DoD's CrossTalk. He has had 17 years of experience in data processing where he has worked at Hercules Aerospace as technical support project leader for computer graphics and expert systems development and capacity planner for the computer systems' strategic planning group. He managed computer technical support for Norton-Christensen, and he served as a programming supervisor for the software reengineering efforts within the Department of Social Services of Iowa.

6. Gibbie Hart, Software Engineering Institute

7. Bernard Zaranski, Managing Consultant

8. Samuel Hoke, Nations Banc Services

9. Carl Tegfeldt, Planning Research Corporation Inc. / Joint Logistics Command (PRC Inc. / JLC)

10. David Breuker, Martin Marietta Astronautics

11. Glenn Racine, Army Research Laboratory (ARL)

    He is the team leader for the Software Engineering Team at ARL-STB. His current areas of responsibility are applied research activities in Software Engineering and Very Large Data Bases.

12. Cris Sittenauer, Software Technology Support Center (STSC)

    He is currently a leader in the completion of the Air Force Software Reengineering Assessment Handbook (SRAH) along with the Air Force Cost Analysis Agency and SAF/AQK. Mr. Sittenauer has been with the STSC for over three years as software engineer and lead engineer of the reengineering team. He has been an author for the STSC Reengineering Technology Report for the last three years and several technical articles for CrossTalk and IEEE reverse engineering newsletter. He has 11 years of software and electrical/avionics design experience on commercial and military programs as a contractor at Boeing Military Airplane Company and as an Air Force civilian.

### 6.3.2 Introduction

To set the tone for the working group, Evan Lock started with the proposition that with reengineering you change the rules. He then suggested that we try to elaborate a kernel set of non-economic decisions to be made as part of the reengineering decision model and that we recommend a set of guidelines for improving decision making. To initiate discussions he posed the following questions to the group:

1. **Scope?** We discussed an organization level scope versus an instance of reengineering and the idea of Business Process Reengineering (BPR). There was general consensus that BPR was outside the immediate scope of this working group and would be dealt with at a higher level. Initially, there was a working group designated to address BPR. Still, BPR encompasses this decision analysis working group, the economic analysis working group, and others. Finally we decided to focus now on an instance of reengineering and then to expand the model to the organization level.
2. **Breadth vs. Depth?** We decided to cover the breadth now in these short two days and cover the depth in our continuing efforts.
3. **Specific vs. General?** We tried to cover all of the general areas and add specifics as time would allow. We plan to add details as we continue with this project in a distributed collaborative environment (DCE).
4. **Dependence on Process?** It was noted several times that this working group is dependent on many of the other working groups and vice versa. As we worked, we tried to note specific interfacing points with specific working groups. The folks responsible for consolidating all of the resultant best practices will need to consider these interactions, interfaces, and overlaps.
5. **Standing on the shoulders of giants?** We all agreed that we did not want to re-invent the wheel. Instead we wanted to take the best of what is available and expand upon it. In fact, Mike Olsem volunteered to provide the retrospective view section of the resulting report. Some of the sources cited

during the meeting were: SRSWG (Harry Crisp), Software Reengineering Assessment Handbook (STSC), Defense Information Systems Agency / Center for Information Management (DISA/CIM), Joint Logistics Command (JLC), Arnold, Byrne, Keller, Ulrich, DFAS (Reengineering Decision Model - Gini Calchera).

6. **Follow-up activities?** Needless to say, we only scratched the surface during the two days of the working group. But most members agreed to continue working as a group in a distributed collaborative environment. Specific future tasks are described later.

The next step was to begin defining the questions we needed to ask. Our first cut produced:

- When to reengineer
- How to reengineer
- What to reengineer
- Why reengineer

We retained these four high-level questions throughout our two days of discussion. Our second level of questions started with:

- Do you have the staff, the tools, the training, and the money? Contract out? What metrics? How do you prepare?
- Who makes decision to reengineer?
- What is the motivation for reengineering?
- Planning for reengineering?
- Reengineering is a series of tasks - what type of reengineering should be used? Need a spectrum of strategies.

At this point we realized we needed to define the boundaries of our discussion and goals. We decided to brainstorm the key concepts we thought needed to be in the decision model, keeping in mind that ours was a bounded task. The results of the brainstorming session are contained in 6.3.7.1, Defining the Decision Analysis Working Group Bounds. As a result of the brainstorming session, we decided that there are two levels of decisions:

- Identify a reengineering project.
- Identify how to implement a reengineering project.

### 6.3.3 Draft Outline for Best Practices

We decided to focus on “How to implement a reengineering project.” But at the top of the chapter we will note that there are two levels of decision making and that we are concentrating on the lower level, i.e. deciding how and what to implement on a selected system.

**Note:** There is a relationship between these levels.

The next item on our agenda was to establish the outline of our report. The report outline is as follows:

## I. Introduction

### A. Purpose and . . .

1. Motivation
2. Objectives and goals, including impact analysis and identifying show stoppers. Work within a set of constraints.
3. Use of Decision Model in terms of time, money, tailoring, etc.

### B. Target audience

### C. Scope

1. Levels of decisions
2. Exclusions
3. Constraints vs. decisions

**D. Caveat -** model is unproven; it is intended to be a flexible decision model not a single pass/path solution, i.e., the order of decisions is flexible.

### E. Participants

## II. Retrospective View

### A. Ad hoc - pitfalls, war/horror stories

### B. Existing work (Kanko)

## III. The Decision Model

### A. Description

1. Precedence - List some scenarios. If strategy already made, do kernel decisions 1, 2, 3, 4.
2. Decision template

**B. Current Practice** e.g., follow Software Reengineering Assessment Handbook

### C. Gaps and improvement opportunity

1. Gaps between existing view and this ideal view
2. Improvement opportunities for this model and the project

D. **Transition -** how to transition to this model (mention caveats)

## IV. Future Plans

## V. Summary

### 6.3.4 Elaborating on the Decision Model

Having agreed to a report outline, we began brainstorming the basic questions in the decision model. The results of this Decision Model brainstorming session are contained in *Software Engineering Techniques Workshop on Software Reengineering*.[1] Our team leader suggested that we take these questions and identify the kernel decisions (KD). Our kernel decision set is:

**Kernel Decisions** (KD):

1. Goal/objectives
2. Resources
3. System analysis - current state (see the Software Reengineering Assessment Handbook for analysis)
4. Organization’s current state (see the Software Reengineering Assessment Handbook for analysis)
5. Strategy/plan
6. Technical approach

==============================================

7. Technology - tool driven vs. strategy driven - is in 3, 4, 5, 6,
8. Risk is everywhere

The line indicates that the first six are distinct kernel decisions. We decided that the last two decisions are pervasive in the first six KDs. Currently we are focusing on the first six KDs while considering Technology and Risk with each KD. The questions resulting from the brainstorming session are each tied to at least one of the Kernel Decisions. This mapping is contained in 6.3.7.2, Constraints or Pre-Conditional Decisions.

For each kernel decision we decided that we would use a decision template. This template would allow us to address the issues and interdependencies at each level. The decision template is:

1. Contact Customer Relations at the Software Engineering Institute at **customer-relations@sei.cmu.edu**.

**Decision template (interdependence here with the Design Record)**

1. Goal
2. Objective
3. Measure success/effectiveness
4. Risks
5. Support why decisions are made
6. Dependencies - previous decisions and their impact - cost
7. Needed analysis number - helpful techniques that can be used
8. Action items
9. Constraints - mandates, project decisions
10. Trade-offs
11. Resource requirements (money, time, people), consequences

### 6.3.5 Applying the Decision Template

With the final few hours in our working group, we decided to apply the decision template to the two most difficult and interesting kernel decisions: 5, What Strategy to Use, and 6, What Technical Approach to Apply. First, we tried to identify the finite set of strategies. We listed three: innovative, incremental, and tactical or opportunistic. We realized that these terms may not be consistently applied throughout the software industry and we will provide a glossary. This is our first attempt at defining the three strategy alternatives and related issues:

**KD 5, Strategy - tactical techniques, planning**
**Decision: determine scope of strategy**
What strategy to use? (Note: This is related to the Process working group.) See Figure 1.

- Innovative

    multiple changes: languages

    platform

    new requirements

    “Big Bang”

- Incremental

    subset

    phased

- Tactical/opportunistic

    pareto analysis of system

use a pilot - will it scale-up?

focused

reuse library

**Other strategy issues**

- Consider viability
- How do I determine my subsets?
- Complexity
- Will I contribute to a reuse library
- Risk analysis - consider various strategies

**Note:** Political, legal, social, economic, technical requirements, constraints, and mandates may dictate strategy.

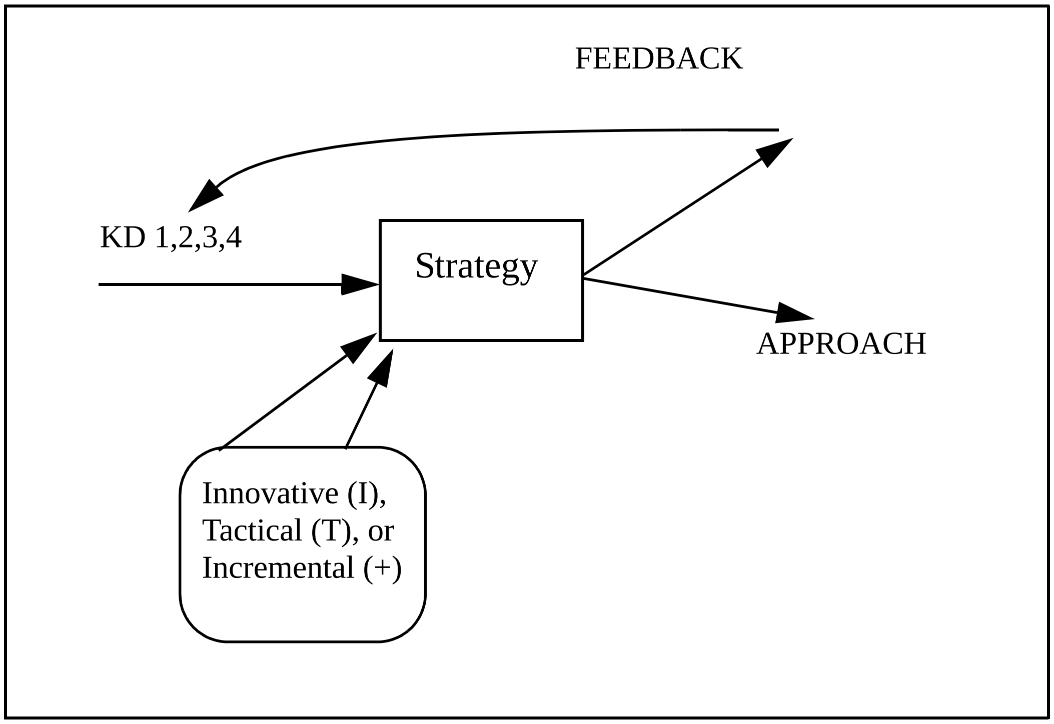


**Figure 1: Relationship of Key Decisions to Reengineering Strategy and Approach**

Within each strategy, there is a set of decisions. We identified the following:

SUB Decisions to Innovative Strategy (See Figure 2.)

- Determine “impact” on system parts.

- Determine “impact” on external entities: e.g., other systems, interfaces, users, hardware.
- Determine “impact” on platforms (develop, test, target, all makes and vendors)
- Determine “impact” on requirements input, e.g., real-time system performance, function.
- Decide on implementation strategy, e.g., partial, evolution.
- No-brainer.
- What happens to new changes?
- The “freeze” frees up staff.

SUB Decisions to Incremental Strategy that are different from Innovative:

- Emphasis is different, e.g., on the interface (with external entities).
- Different type of platform issues, e.g., CASE.
- Time considerations - layout parts.

SUB Decisions to Tactical Strategy that are different from Innovative:

- “Bandaid” approach.
- Less concern about interfaces.
- Opportunities are influenced by miscellaneous factors, e.g., some other system changed.
- Be sure you know KDs 1 and 2 (Goals and Resources).

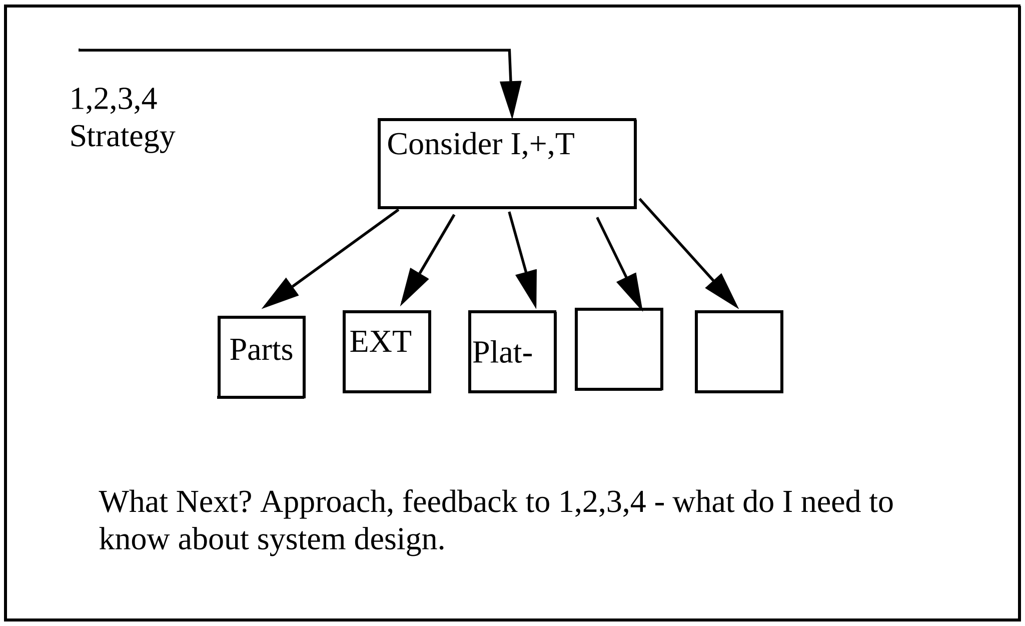

1,2,3,4
Strategy
Consider I,+,T
Parts
EXT
Plat-
What Next? Approach, feedback to 1,2,3,4 - what do I need to know about system design.

**Figure 2: Relationship of Type of Strategy to SUB Decisions**

This is our first attempt at defining the issues related to the Technical Approach decision.

### KD 6, Technical Approach

What reengineering approach to use?

- Alternative/options

1. Reengineer

   Tools/techniques/scheme/methods?

   Translation - is a translator available?

2. Use COTS
3. Retire
4. Redevelop

**Note:** At each level you can select combinations of choices. Also type of system (KD #3) may dictate approach: MIS, embedded real-time, transaction, PC-based, mainframe, Language, functional, etc.

### Questions and Issues

- Which approaches to follow?
- Which reengineering activity?

- Taxonomy is unclear - see IEEE.
- Tag approach to architectures.
- Data driven?
- Can I use a reuse library?
- Risk analysis - consider various strategies.

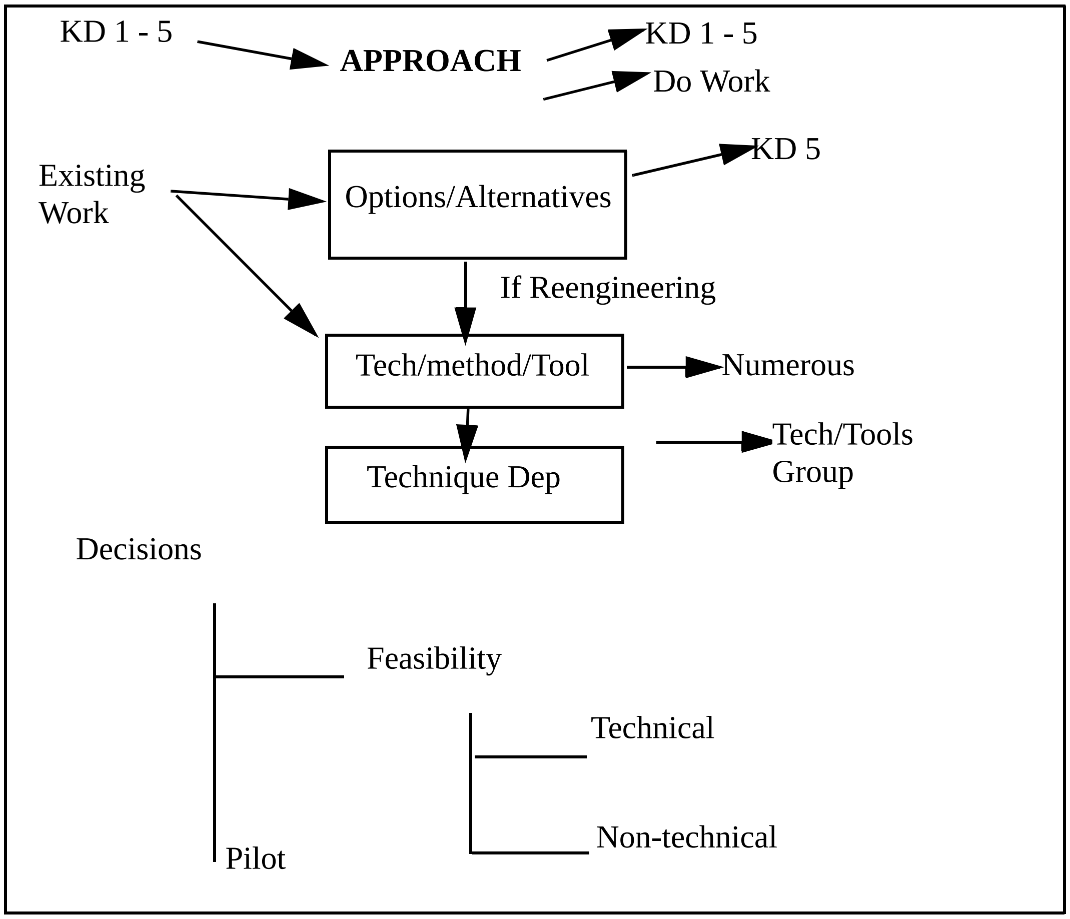


**Figure 3: Issues Related to the Technical Approach Decision**

### 6.3.6 Action Items

The following action items were noted during our two-day session:

- Which “giant” work covers which level? How well? Gaps?
- Overlap with other groups - interfaces.
- Validate the model.
- Reconcile strategy definitions and names. Wait to hear from Joan Smith, DISA. Coordinate with other groups - especially process.
- Determine decisions within technical reference model (Martin).

- Later item - what is the class of the system? System attributes.
- Look into decision analysis tools and techniques.
- Explore probabilistic decision tree idea.
- This group needs to coordinate the decision template with the Design Record Group.
- Coordinate with Understanding Legacy Systems group for kernel decision 3.
- Tools group needs justification from us.
- Review the process model group’s stuff and/or existing process models for reengineering and categorize for these strategies.
- Approach taxonomy is unclear - see IEEE.
- Evaluate practicality/usefulness and coordinate with process group.
- Review process models from other working groups and categorize them to approach (innovative, incremental, tactical).

### 6.3.7 Added Notes

Prior to departing, Glenn Racine offered these thoughts for future work.

When determining organization state, consider:

1. Software Engineering Assessment Handbook analysis
2. Personnel: training, expertise
3. SEI capability maturity model (CMM) rating
4. Reuse libraries
5. Technology
6. Corporate culture
7. Line staff vs. matrix
8. ISO-9000 Rating
9. Contracting ability
10. Political climate
11. Top management support
12. Requirements
13. Staff roles
14. Present process

Some references for approach include:

1. Santa Barbara I: Approaches for COBOL MIS
2. “Database Programming Design,” March 1994, pages 23-25, for Legacy to Client Server.

#### 6.3.7.1 Defining the Decision Analysis Working Groups Bounds

Strategic - to reengineer or not

Tactical - what, how, when, implementation, technical transfer, operational

Business Process Reengineering (BPR) - how do we include this?

Motivation - initiating event

Preparation - staff, technical transfer, managing technical change

Target audience - management, customer, maintenance, government, corporate, academe, decision maker, use the process, use the output of the process

Purpose of this chapter

Target area - system, program, functional area, application

Format of decision model, representation

Technology, tools, platforms

Reuse

Mechanism to relate to other groups

Lists things deliberately ignored

Recognition of giants work

Use of giants work - strengths, weaknesses, gaps

Citations

Staffing

Decision metrics

Methodology

Formal methods

Organizational behavior

Motivation - initiating event

Selection criteria

Roles - where and how they play in the decision making

Risk

Goals

Elaborating the decision model. DECISIONS surrounding the reengineering project

#### 6.3.7.2 Constraints or Pre-Conditional Decisions

**Note:** The numbers indicate the kernel decision (KD) the item is associated with.

What is my budget? 2

What is my goal? 1

Do I have time? 2

What is target application or system? 1

What reengineering approach to use? (methods, how?) 6

- Re - X
- Use COTS, retire, redevelop

Data driven

What Strategy to use? (tactics, techniques, process, planning.) 5

(See DISA Risk Taxonomy pp 22,23)

- Innovative - multiple changes: languages, platform
- Incremental - subset

Use a pilot - will it scale-up

- Tactical/opportunistic - pareto analysis of system, focused

Consider viability

Technology 7

- Availability
- Contractor source
- Applicability
- Integration

Interface Issues

What standards will be used? 1

Or adopt 3, 4

How do I determine my subsets? 5

Complexity

How well prepared? 4

Roles of staff? - how much internal vs. external? 2, 4

Will present process support reengineering? 4

Can I use a reuse library? 6

Or reuse parts of existing system? 3

Will I contribute to a reuse library? 1, 5

Determine health of system 3

Cost/benefit analysis 3

Risk analysis - consider various strategies - all

What are the independent and dependent variables 3

Determine metrics 3

Future organizational oversight - maintenance 1

What are the testing requirements? 3

What are the training requirements? 4

Organization requirements? 4

Data requirements? 3

## 6.4 Design Record Working Group Session

### 6.4.1 Participants

1. John Bergey, Software Engineering Institute (technical lead / minutes taker)
2. John Leary, Software Engineering Institute (facilitator)
3. Robert Arnold, Software Evolution Technology (SEVTEC), Herndon, VA

   Founder and president of SEVTEC. He is the author of Software Reengineering (IEEE Computer Society Press, 1993) and Software Restructuring (IEEE Computer Society Press, 1986). He was Program Chair, General Chair, and Steering Committee Chair for the Conference on Software Maintenance. At SEVTEC he now provides educational, information, consulting, and research services in software maintenance, impact analysis, reengineering, and redevelopment.
4. Jean-Marie Favre, University Joseph Fourier, France

   He is currently a Ph.D. student at the University Joseph Fourier working on his dissertation. Mr. Favre works on the ADELE Team at the university.
5. Judy Hardin, Loral Space Information Systems (SIS), Houston, TX
6. Butch Hayes, Booz Allen Hamilton, McLean VA

   Mr. Hayes is a Senior Associate at Booz, Allen & Hamilton. He has close to twenty-five years of professional experience, with a career highlighted by both technical and business achievements. He has received recognition in such diverse fields as CASE technology, software engineering and reengineering, operations research, and psycho-motor skills measurement. As a manager, he has held both staff and line positions and managed his own company for more than two and a half years. At Booz Allen, Mr. Hayes provides senior support for Systems Modernization services offered to a variety of the company's clients. This support covers issues such as Information Systems Planning, Downsizing/Rightsizing, Reengineering, CASE Implementation and Integration, Client/Server Deployment, and Technology Enhancement.
7. Arun Lakhotia, University of Southwestern Louisiana, Lafayette, LA

   He is an Assistant Professor with the Center for Advanced Computer Studies and Director of the Software Research Laboratory. Mr. Lakhotia's research interests are centered on issues related to improving the productivity of programmers and reliability of their programs. He is currently directing a project to recover dataflow oriented designs from source code of legacy systems.
8. Frank Svoboda, UNISYS Government Systems, Reston, VA

   He provides methodology, tool, and process support as a member of the STARS Demo Project Application Engineering Team. Frank has extensive experience in real-time software engineering and the Ada programming language. His professional interests include reengineering, reverse engineering, domain engineering, reuse, and CASE.

### 6.4.2 Introduction

The primary thrust behind the creation of a Design Record has been to capture the "design decisions" that led to the evolution of a particular architecture and design instantiation. Accordingly, the charter for the Design Record Working Group was to identify and discuss the issues and criteria associated with defining the contents of a Design Record to support reengineering. Topics included the effectiveness and value-added benefits of Design Records, the information boundaries of Design Records, criteria for selecting Design Record information, and cost effective means of overcoming barriers such as collecting and maintaining up to date Design Record data.

#### 6.4.2.1 Goals and Objectives

The specific goals and objectives of the Design Record Working Group were several:

1. Explore the issues, implications, and technical aspects of the Design Record concept.
2. Define and bound the Design Record concept as it relates to reengineering.
3. Draft Outline for Design Records Chapter for SEI Guide to Best Practice in Reengineering.
4. Explore the possibility of forming a follow-on working group to research and write the Chapter on Design Records for the Guide to Best Practice in Reengineering.

#### 6.4.2.2 Preliminary Discussion and Reference Materials

Preliminary discussion centered on the participants' backgrounds and their particular interest in the Design Record concept. Most of the participants shared their viewpoints and concerns and offered reference material to the Design Record effort, including:

- Arnold: several published papers, and consulting results on the topic.
- Hardin: (possible) insight into LORAL reengineering for the Space Station.
- Bergey: several applicable papers on process life cycle.
- Leary: papers on Information Architecture, e.g., Zachman.
- Hayes: a paper on Information Modeling.
- Svoboda: documentation from an Army reengineering / Design Record effort.
- Lakhotia: papers on reverse engineering technology.

Others asked for reference material on Design Records, especially Robert Arnold's work, the Santa Barbara reports, and John Salasin's work, which they were largely unfamiliar with.

#### 6.4.2.3 Design Record Issues

The working group's initial brainstorming session focused on the following strawman Design Record issues that were used as a point of reference and catalyst for discussion.

**Design Record Issues:**

1. Purpose/objective
2. Scope/system boundaries
3. Information content
4. Customer or user
5. Need / usefulness of data / benefit
6. Data collection means
7. Product form
8. Means for viewing/browsing data
9. Maintenance
10. Cost effectiveness
11. Criteria/policies/standards
12. Other issues

The group generally held the belief that Design Records were an attempt to capture important supplemental information related to design decisions and design history rationale that typically are not captured during development but “fall through the cracks” under the pressures of project milestones and cost containment. The Design Record was viewed as the mechanism, or enabling technology, for capturing key architectural and design information that is only considered supplemental in the sense that it is not used for the actual generation of the software code.

#### 6.4.2.4 Design Record Questions

During the first day’s brainstorming session on the Design Record issues, many basic questions were raised concerning the efficacy and application of Design Records. There was not adequate time to discuss them all, but they are included below for completeness and consideration by the Design Record team in post-conference working sessions. The questions reflect the fact that Design Records are more of a concept than an actual practice and additional research needs to be undertaken and empirical evidence gathered to prove the concept, refine it, and transition it into actual practice.

- What is a Design Record and what is its purpose? What are some of its uses?
- Who is the customer or user of the Design Record?
- Who has created and is using Design Records? How do they benefit from them?
- What are some of the claims for value (cf., Frank Svoboda’s work)?
- What are some of the sources of information for Design Records?

- What is the nature of the information content of a Design Record? What is in it? What information is pointed to by Design Records?
- What is the desired end-product? Is the Design Record just another document? A repository? A collection of things?
- What type of information should be captured in a Design Record? How would the Design Record data be used to support maintenance? Reengineering? Development?
- What is the developer's response to Design Records? What is the value of Design Records relative to:

  Cost of maintenance

  Cost of reengineering

  Technical feasibility

  Efficiency

  Productivity

- What is the process flow for collecting, using, updating, maintaining Design Records? What is the relationship of the Design Record to:

  Architecture

  Tools

  Process

  Methods

  Configuration management

  Lessons learned

  Maintenance

  Reengineering

  Application profile

- What is the information model for the Design Record?
- What are the envisioned usage scenarios for Design Records?
- What will it cost to collect, maintain, access, and use the Design Record data?
- What functions do Design Records serve as an enabling technology?
- What is the evidence for usefulness of Design Records?
- What guidelines are appropriate/practical for addressing different kinds of applications?
- What should Design Records look like for legacy systems?

- How much information should be captured?
- How can one avoid the common approach to Design Records of capturing any and all data that may be considered relevant?
- What kinds of limits and restrictions should be imposed to put reasonable boundaries on the scope and content of a Design Record? What criteria would be appropriate to address such factors as:

  The kinds of information to capture.

  The amount of information to be collected.

  The data collection, access, and display means.

  The usability of the data.

  The cost effectiveness of the data.

  The currency and maintenance of the data.

- How should these factors be weighted and/or prioritized?
- Can the information content of the Design Record be justified on a cost-effectiveness basis?
- How can the cost effectiveness of the Design Record concept be demonstrated?
- Would the existence of a Design Record “document,” or its equivalent, supercede the need for one or more of the traditional system-level or software-level design documents?
- Would the Design Record replace the traditional Design Document or a portion of it? What would be its relationship to the DoD-Standard 2167A and SDD documents?
- How does the information content of a Design Record differ from the data that is traditionally specified via the standard Data Item Descriptions (DIDs) associated with these standards?
- Would the Design Record be a contractual deliverable? Would DoD need a Data Item Description for a Design Record?
- Will the Design Record have to be treated as proprietary data?
- Could access to the Design Record data be construed as compromising the developer's competitive edge or providing insight into company proprietary processes?
- Would the existence of a comprehensive Design Record preclude the need to reverse engineer a system?
- What are the essential differences between a Design Record and a Unit Development Folder or an engineering notebook? In what areas is the data similar, different, or altogether new?

- Does the Design Record have to reflect the totality of a system and all its subsystems (and their major components) or should an individual Design Record be associated with each unique computer system?
- Would it be useful in formulating the Design Record concept to distinguish between a System Design Record and a Computer System Design Record?
- Should the Design Record data be organized around an informational taxonomy? (e.g., design data, design history, design alternatives/tradeoffs/ decisions, etc.)
- What specific kinds of data should be included in a Design Record? Design decisions? Design history? To what level of detail? How can they best be represented?
- What are the most useful representations of information for inclusion in the Design Record?
- Is the Design Record information primarily for the purpose of recording the design decision process? What are the boundaries of the information content?
- Where does the Design Record make its primary contribution? Is it in the area of non-functional quality, behavioral performance for maintenance personnel, a repository of engineering parameters, rationale for program structure, system architecture, or a semantic/conceptual mapping for understanding?
- Does the Design Record itself contain information on the baseline design itself (i.e., the design instantiation which resulted from the design decisions captured in the Design Record)? If not, where and how is the baseline design documented? What will have to be assumed?
- How and to what degree will the Design Records themselves be evolvable?
- Are tools a prerequisite to the creation or use of Design Records? What tools, techniques, or technology developments could promote the Design Record concept?
- How would the Design Record data be used in a reengineering effort?
- How can Design Record data be stored, accessed, represented, and displayed?
- What are the major impediments to the concept of a Design Record? (Amount of data, variety of data types/form, cost of collecting and maintaining data, data collection means, currency of data, demonstrated usefulness of data, identification/tagging of data, accessibility of data, no empirical evidence for their worth, etc.)
- What would motivate an organization to adopt the concept of Design Records?
- What experiments could be conducted to demonstrate the value of a Design Record?

- Does a Design Record have to be generated during development to support follow-on maintenance? Or is it still feasible/beneficial/profitable to create a Design Record during the maintenance phase or as part of a reengineering effort?

#### 6.4.2.5 Working Group Discussion

There was strong consensus that Design Record “technology” is largely immature today, and that few explicit Design Record practices are in use or will be located; this led to the conclusion that future work should focus on eliciting the pivotal concepts and not simply upon an attempt to disseminate isolated instances of relevant Design Record forms and practices. It also appeared to be a central theme from all participants that a Design Record was primarily aimed at supporting a software engineering practitioner in maintenance, reengineering, or evolution. There was some agreement that a Design Record's schema and update/use/tool support would tend to vary according to typifying or characteristic profiles of reference applications.

Our discussion surfaced concerns with three areas in particular. These were usefulness of the Design Record, cost aspects, and its practicability.

1. **Usefulness**. We reached consensus that the initial beneficiaries would be maintenance activities and maintenance personnel as a precursor to reengineering. All more or less accepted the notions that a Design Record can enhance technical feasibility and streamline both reengineering and evolution. We discussed potential content of a Design Record in these contexts. With regard to the nature of Design Record content, there was little debate that it should capture design decisions and rationale, and history. There was some discussion as to how best to capture history, i.e., as a series of design/decision snapshots, or as a series of change ‘deltas’ and analysis results. The degree to which a Design Record presents and references various aspects of design detail was discussed but not resolved. One view was that only significant aspects of design “which would otherwise be lost” should be included. Another view was that the former “gap filling information” should be retained within a schema which also contains at least references to all other relevant aspects of design information needed to reconstitute a system. Concerns which bounded this discussion were that the Design Record content be cost-effective to capture, be automatically maintainable, be intellectually apprehensible, have practical application, and have real, demonstrated value to at least one set of users (e.g., maintenance personnel).
2. **Cost**. We discussed establishing the cost-effectiveness of Design Records as being the pivotal aspect of justifying the creation and maintenance of Design Records. Some cost considerations included elimination of some level of reverse engineering, reduction in delays needed to effect required maintenance, speedier results from reengineering effort, and technical inputs for activities such as selecting reusable components and understanding of legacy systems. We did not discuss in great detail the components or the cost of creating a Design Record, although several work group members were concerned with this cost.

3. **Practicability**. We discussed the need to collect empirical evidence of successful implementation of a Design Record as an effective means of supporting software evolution. There was general consensus regarding the premise that the software elements and artifacts that comprise the Design Record content must be automatically collected and updated to insure the practicality of the Design Record concept. Concern was also expressed about the need to assure that the content of a Design Record does not become overwhelmingly huge.

Various models were advanced to provide a context for Design Records. One of these diagrammed software activities to reflect how certain activities contribute design information. Another referenced the National Institute of Standards and Technology (NIST) maintenance model (i.e., correction, perfection, and adaptation) to suggest that the Design Record concept embraces maintenance, since adaptation often involves changes to function, changes to platform (hardware and software), and additions to capability based upon legacy. Dr. Arnold's context for reengineering emphasized that reengineering enables maintenance and development, and the use of Design Records to support reengineering must inherently be based on tool artifacts to facilitate automation. Dr. Lakhotia's position was that reengineering use of a Design Record should be seen as largely involving design rationale and decisions rather than elemental design detail.

The group generally accepted the notion that a question and answer (Q&A) section should be included in a chapter on a Design Record to help stimulate understanding of several of the many important perspectives. Leary offered the following sample questions to stimulate both discussion and understanding of the potential value added to a guidebook by a Q&A section:

1. On **utility**:
    a. How is a Design Record useful for reducing perfective and adaptive maintenance effort?
    b. How is a Design Record useful for developing conceptual models for new capabilities that necessitate reengineering?
    c. How is a Design Record useful for limiting integration and testing support in maintenance or in reengineering?
2. On **relationships**:
    a. How does Design Record content relate to deciding the feasibility of a proposed maintenance effort?
    b. How does Design Record content relate to identifying design alternatives and design trade-offs for selecting a design or reengineering strategy and approach?
    c. How does a Design Record's content preserve the user's/owner's options for systems evolution?

3. On **cost**:

    a. How will a Design Record reduce reengineering or maintenance cost?

    b. How will a Design Record enhance estimating the cost of maintenance efforts?

    c. How is investment in a Design Record balanced against alternative investment in reverse engineering or in maintenance?

#### 6.4.2.6 Summary

The presentations given on Wednesday and Thursday to the general session of the workshop were a good top-level reflection of the deliberations of our work group. We produced a general outline for a chapter in a reengineering handbook. It will certainly need more elaboration and subsequent reorganization, but all of the topics we identified ought to be covered in the planned Guide to Best Practice in Reengineering.

### 6.4.3 Draft Outline for Best Practices

The working group spent the last day organizing the information recorded in the earlier brainstorming sessions into a draft outline for a Chapter on Design Records. It was apparent that the generic outline that was proposed for the chapters in the Reengineering Best Practice Guide was not appropriate for a chapter on Design Records, since we are trying to advance a concept as opposed to describing best practice. The following is the draft outline for the Chapter on Design Records to be included in the Software Engineering Institute's Guide to Best Practice in Reengineering.

**I. Executive summary**

**II. Introduction**

- **A. Purpose**
- **B. Design Record definition**
- **C. Scope**
- **D. Goals and objectives**
- **E. Contributions**

**III. Concept of a Design Record**

- **A. Assumptions**
- **B. Ground rules**
- **C. Expected contributions to maintenance and reengineering**
- **D. Information sources**
- **E. Criteria for content**

**F. Information model**

**G. Process flow model**

**H. Relationship to other best practices and technology developments**

## IV. Background

**A. Origins of concept**

**B. Nomenclature**

**C. Design Record literature**

**D. Current Design Record practitioners and users**

**E. Evidence for Design Record**

**F. Design Record Points of view (themes)**

**G. Commercial practice for the Design Record paradigm**

**H. Open market support**

## V. Design Record issues

**A. Answers to tough questions (Q&A) on Design Records**

**B. Value, utility, and cost effectiveness**

**C. Support for legacy systems, maintenance, and reengineering**

**D. Design Record content**

1. Design information
2. Design history
3. Design decisions
4. Design rationale

**E. Practicality and feasibility**

## VI. Approach to transition from concept to practice

**A. Guidelines**

**B. Profile of targeted users**

**C. Usage scenarios**

**D. Criteria**

**E. Identification of high payoff items**

## VII. Strawman Design Record

**A. Enumeration of artifacts**

**B. Description of artifacts**

**C. Information model**

**D. Guidelines**

**E. Collection process**

**F. Example of a maintenance-oriented Design Record**

**G. Example of a reengineering-oriented Design Record**

**VIII. Future work**

**A. Experimentation**

**B. Validation**

**C. Refinement of Design Record model**

**IX. Summary and conclusions**

**A. Findings**

**B. Recommendations**

**X. Glossary**

**XI. Bibliography**

#### 6.4.4 Future Plans

The members of the Design Record Working Group who indicated they were willing to carry on with further work are John Bergey, Arun Lakhotia, John Leary, and Frank Svoboda. Our visitor from France is not able to commit to future effort. Our three commercial contributors (Arnold, Hardin, and Hayes) *may* be able to contribute, especially in an e-mail mode, but cannot make promises to commit substantial periods of their effort. All three, however, are interested in some kind of continuing involvement, and all but Judy Hardin have e-mail access. What is envisioned for the post-conference working group is defining a modus operandi for refinement of the outline, selected writing assignments, document exchange and review, and e-mail communication.

## 6.5 Reengineering Economics Analysis Working Group Session

### 6.5.1 Participants

1. John Clark, Comptek Federal Systems (technical lead)

   Mr. Clark is a program manager with Comptek Federal Systems, Inc. He has over twenty seven years of software development, maintenance, and cost management experience. Mr. Clark is experienced in all aspects of the software life cycle for several DoD programs developed for the US Air Force, Navy, Marines, and Coast Guard. He was the industry co-chair of the Reengineering Economics Panel at the First Software Reengineering Workshop (Santa Barbara - I) sponsored by the Joint Logistics Command, Joint Policy Coordinating Group, Computer Resource Management (JLC-JPCG-CRM). He was a principal architect and author of the draft Reengineering Economics Handbook (MIL-HDBK-REH).

2. Bob Park, Software Engineering Institute (minutes taker)
3. Lisa Brownsword, Software Engineering Institute
4. Susan Crosswait, Marine Corps, Marine Corps Tactical Systems Support Agency /Joint Logistics Command (MCTSSA/JLC)
5. Brad Donald, Air Force Cost Analysis Agency / JLC
6. Paul Huntwork, Digital Equipment Corporation
7. Ben Keller, Systems Research Center / Virginia Tech

   He is a Ph.D. student in computer science at Virginia Tech specializing in software reverse engineering and reengineering. His research interests include methodologies for software and systems evolution, specification and programming languages, formal methods, and applications of algebra to software engineering.

8. Dave Struble, Texas Instruments
9. Dan Sullivan, James Martin & Co.
10. Bob Webster, Air Force, Electronic Systems Center (ESC/JLC)

### 6.5.2 Introduction

Context setting of the working group activities during the workshop.

### 6.5.3 Draft Outline for Best Practices

## I. Introduction

**A. Purpose**

**B. Scope and objectives**

1. Guidance for CFO/PMO, etc., concerning what they should know about reengineering economics

2. Guidance to key issues for decision makers
3. Guidance to CFO/PMO etc. on reengineer/not reengineer decision
4. Guidance in how to conduct an economic assessment of reengineering strategies
5. Practical guide to analyst and senior executive/decision maker
6. List of deliverables from economic analysis
7. Generic tutorial
8. Bibliography of estimation methods (pointers)
9. Worked example
10. Decision matrix (parametric description or guide to process)
11. Savings for the investment
12. Spreadsheet
13. Improvement opportunities, transitional issues challenges
14. Common context setting
15. Provide pointers to “cookbooks” rather than include them
16. Provide pointers to other practical sources of information
    a) keyed to topics
    b) annotated bibliography
17. Survey, description, and analysis of state-of-practice, state-of-art
18. Pointers to sources (methods)
19. Characterization rather than analysis
20. Descriptive evaluation

**C. Overview**

**D. Key Issues**

1. Government vs. commercial
2. Business case
3. Portfolio analysis
4. Product obsolescence
5. Profit motive
6. Terminology and indicators
7. Interdependency of processes/chapters
8. Audience/users
    a) software manager
    b) cost analyst/engineer
    c) software engineer
    d) decision maker
    e) IEEE member
    f) standards user

g) program executive officers/government executives

h) acquisition agent

i) program management officer

j) chief financial officer

9. Consider inclusion of all costs (include wash costs)

### E. Things addressed elsewhere

## II. Retrospective view

How did we get where we are?

Examples of commercial/government reengineering economics.

Statement of current reengineering economics.

Motivations: maintenance costs, aging systems, desire to exploit new technologies, changing/expanding mission requirements.

### A. Anecdotal Evidence

### B. Lack of proof/credibility

1. Technology credibility (tools, terminology)
2. Cost estimating credibility
   a) process definition
   b) technical assessment
   c) work breakdown structure
   d) cost element structure
   e) cost models
   f) other methods (analogy, engineering judgment, consensus, best guess, SWAG)
3. Metrics credibility
   a) new development
   b) maintenance
   c) reengineering
   d) reuse

### C. Success stories

1. Lack of examples
2. Lack of Publicity

### D. Failures

1. Lack of examples
2. Lack of lessons learned

## III. State of the art/practice

Examples of economic analysis or reengineering economics

Table of comparison of practices, their characteristics, and areas that each addresses.

Methods of quantifying key economic issues.

Classification of techniques by economic indicators, typed approach (“grass roots,” analogy, parametric), and problem/issue/decision.

**A. Survey results (process)**

1. Software Reengineering Assessment Handbook
2. Other handbooks/processes (if any)
3. New development cost models/methods
4. Maintenance cost models/methods
5. Reengineering cost models/methods
6. Reuse cost models/methods

**B. Survey results (projects ongoing/completed)**

**C. Recommended best practices**

1. Software Reengineering Assessment Handbook
2. Other handbooks/processes (if any)

**D. Gaps**

**E. Foundational methods**

**F. Cautions in the use of the techniques**

1. Garbage in, garbage out

## IV. Challenges

**A. Government acquisition, budgeting, accounting method**

**B. Re-engineering process improvement (technical, economic and management decision models)**

**C. Reuse process improvement (technical, economic and management decision models)**

**D. Success stories/publicity**

**E. Community feedback/communication**

**F. Identifying benefits**

**G. Integrating benefits analysis with cost?**

**H. Measurement needs and metric techniques**

1. How do we quantify savings in maintenance after reengineering?
2. How do we quantify benefits before and after?

**I. Forecasting credible costs and benefits (development, maintenance, reengineering, reuse)**

## V. Improvement opportunities (see IV. Challenges)

**Table 1: Opportunities For Improvement in Components of Financial Estimation**

| | Create Glossary | Collect Stories | Collect/ Characterize Financial Models | Collect/ Characterize Estimating Models | Collect Metric Data |
|---|---|---|---|---|---|
| Reconcile Government/ Commercial | Strong | Weak | Strong | | |
| Estimating Per Technique | Weak | | | Strong | Strong |
| Financial Models | Weak | | Strong | | Strong |
| Risk Assessment | | | Weak | Weak | Weak |

## VI. Transitional issues

### A. Dissemination of best practice

1. Standards
    a) terminology
    b) estimating/accounting
2. Training
3. Tools
4. Finding paradigm pioneers

### B. Feedback of results

### C. Revision of best practice

### D. Continuous evolution (go to A. Dissemination of best practice)

## VII. Prospective view

### A. Describe business cases/benefits

### B. Incorporate domain analysis reuse and product line strategies

### C. Reengineering economics vs. other approaches to meeting future system needs (COTS, GOTS, retire/end of life cycle)

1. Always consider COTS/GOTS strategy (if COTS/GOTS solution is available).

2. Always consider scrapping the program.

## VIII. Definitions and bibliography

### A. Definition of terms

1. Mapping of government to commercial

### B. Bibliography

1. Foundational references
2. Pointers to basic references
3. Review source of definitions of Net Present Value

## 6.5.4 Future plans

Continue/expand working group participation.

Get some business analysts involved.

Leadership/organization.

Sponsorship/funding.

## 6.6 Understanding Legacy Working Group Session

### 6.6.1 Participants

1. Doug Waugh (technical lead)
   Member of the technical staff
   Software Engineering Institute

2. Bob Krut (minutes taker)
   Member of the technical Staff
   Software Engineering Institute

3. Maj. Ken Mullins
   Software Control and Evaluation
   Strategic Command (STRATCOM)
   War Planning Systems

4. Julia McCreary
   Reengineering Technical Manager
   Internal Revenue Service

5. Charles Stump
   Senior Software Engineer
   Leverage Technologists Inc.

   Mr. Stump has participated in the development, integration, and application of CASE tools with particular emphasis on reverse engineering and software analysis. Mr. Stump’s experience includes software development, process definition, software quality assurance, and standards definition.

6. Edward Coyne
   Group Director
   SETA Corporation

7. William Hasling
   Group Leader, Design Methodology and Tool (CASE)
   Siemens Corporate Research, Inc.

8. Scott Tilley
   Department of Computer Science
   University of Victoria

### 6.6.2 Introduction

The session began with Mr. Waugh presenting a few slides as “food for thought.” These slides presented topic areas such as:

- The issues involved in understanding legacy systems.
- Obtaining information about the current work in the field of reengineering.
- The “state of the art” and “state of the practices.”
- The technology base.
- The possibility of generating a “guide to best practices.”

These slides also addressed:

- How much understanding is really necessary? Example sources of information about legacy systems would be system artifacts (documentation, architecture, design, drawings, etc.), operational instance (what does it do), stimulus and response, developer/users, gain human understanding of the system, and development/operational/maintenance history.
- What are the issues given this understanding? Such issues would be the information lacking in the legacy systems; the difficulty in manipulating artifacts; being more intelligent about how we look at the system; the "man-in-the-loop" may be cumbersome; how do you ask intelligent questions of the designer; incorrect or inconsistent information, understanding the levels of granularity; the need to have a more effective way to capture the information (e.g., it was in someone's mind at one time but it has probably been lost); and a stronger pattern recognition.
- Process issues? This includes the fact that development of understanding is a one-time process. It is a labor intensive process, generally unable to carry over experience to new situations. Once understanding is derived, it is seldom maintained. This puts you into the same cycle that you just overcame.
- What are the advantages of understanding legacy? Are the advantages a rich set of information, operational model, well defined semantics at code level, human generated identifiers, cliches, style?
- Miscellaneous issues such as how do you know that you understand the system, how do you capture the understanding, how do you navigate through the system, and would the use of domain models help?

Other issues that the group felt should be included were:

- There is no attempt to follow the thread of information and what goes in and out (i.e., the dynamic paths through the system, a butterfly traversal through the system).
- There are no tools in real-time to express these dynamic paths. The current best is state transition diagrams.
- In addition to hardware and software, how the user uses the system should be included.
- The capturing of information in design decisions and audit trials.
- Information overload and the development of a usable, organized method of storage and retrieval of the information.

### 6.6.3 Draft Outline for Best Practices

This working group felt that Understanding Legacy Systems was essentially the name of the conference (i.e., what is it that needs to be understood to do your reengineering task?). The group felt that the discussions held during this working group session have helped order the sequence of what needs to be done to enhance the understanding of the effort.

The members of the Understanding Legacy System Working Group concluded by establishing an outline for the Guide to Best “Understanding Legacy Systems” Practices Repository. The outline is:

- **I. Introduction**
- **II. Aspects of legacy system understanding**
  - **A. Objectives**
  - **B. Domains**
  - **C. Artifacts**
- **III. Techniques for gathering information**
  - **A. Static**
  - **B. Dynamic**
  - **C. Program history**
- **IV. How to understand/reengineer your system**
- **V. Pointers to case studies**
- **VI. How to update the case study repository**
  - **A. Formats**
  - **B. Templates**

### 6.6.4 Detailed Aspects of the Outline

Portions of this outline have been decomposed into more detailed information by the working group to clarify their points. This information is elaborated in the following sections.

#### 6.6.4.1 Artifacts and Objectives

To get a better grip on these issues, the group decided to create a list of “artifacts” and “objectives” for a legacy system. Artifacts capture the state of the legacy systems, whereas the reengineering objectives are the objectives that determine the depth of understanding required. The generated artifacts are:

> source code; executable (or the running system); data (files, schema, models); JCL (scripts), build/make files; configuration management history (version control and change history); system requirements; design document; architecture; documentation (user guide, programmers manuals); design record; analogous systems; informal information; and screens.

It was noted that all of this is good if you are lucky enough to find this stuff sitting around. The generated objectives are:

modify "as is"; platform/operating system changes; database migration; porting to a new language; gather documentation (enhancing maintainability); restructuring of the system (not necessarily changing the functionality); reuse and salvage; quality improvement; interoperability and integration; improve performance; enhance functionality; training; and whether to reengineer or not.

An interesting point was brought up that discussed why most systems are considered maintenance efforts rather than reengineering efforts. Maintenance covers a broad category because all systems can change through maintenance, but anything else is considered the creation of a new system and this becomes hard to do and hard to get funded. For example, a maintenance effort could be done in the current language vs. the DoD Ada mandate for new systems.

It was felt that we should step back and look at the title (Understanding Legacy Systems) and define the activities that one would perform to activate this: use objectives to determine necessary artifacts, are the artifacts available (derive those needed if not available), organize the artifacts (how are they gathered and how do I store this information in a meaningful way), and what are some of the consequences of the suggested changes (impact analysis, domain modeling, etc.)?

At this point, the group generated an initial outline, "Best practices for understanding legacy systems depends on --". The outline consisted of the following three points:

- Reengineering objectives that determine depth of understanding required.
- Artifacts available - state of legacy system.
- Characteristics of application domain.

Domain characteristics are used to determine what artifacts to look for. Some of these are: interactive/driver, mathematical/algorithm, transaction/atomic process, real-time/ (scheduling, constraints, process communication, interrupts, how interfaced), language, platform, and function/end-user (application domain).

#### 6.6.4.2 Techniques

The group decided to discuss some known techniques for gathering information. The initial list covers what was defined as static analysis techniques. These include simple design methods (representations), data item cross reference, static structures (hierarchical design), CALL trees, data or program slicing, partitioning (breaking the program into logical smaller modules), generic restructuring (not really changing the function of the code but rather reclustering the modules for understanding, including repackaging and data localization), pattern matching (all levels of granularity), control flow graphs, comment capture and analysis (comments from the code), comment generation/validation from the source code, dead code analysis (identifying it), metrics collection, subgraph identification (complexity comparisons to look for areas to simplify the complexity), formal methods, path analysis, and walk throughs.

The dynamic analysis techniques discussed included profilers, debuggers, simulations, assertions, and stimulus-response tests.

In addition to the static and dynamic analysis techniques for information collection, the group felt that there existed an additional category for human techniques that captures the informal information. This category was referred to as the program history analysis and includes: interviews with coders, managers, users, architect/designers, as well as reviewing reports such as system notes, maintenance notes, historical records, change history, error logs (including a known problem log), work orders, design notes, and any additional documentation that would complete the paper trail. This category would also include the knowledge that the code was generated during a certain time era (the software archeology).

For the best possible understanding of the legacy system, several of these analysis techniques should be used in conjunction.

#### 6.6.4.3 Information Dissemination

Now that the artifacts, objectives, and techniques have been discussed, where do we go from here? While maintaining a focus on the generation of the outline for the guide to best practices report, the group discussed how this information was to get out and who was going to be responsible for the maintenance of the guide. It was suggested that the Software Engineering Institute should serve as the repository of information to be disseminated out to whomever desires it. A sort of clearing house for reengineering information.

The repository should include information from as many perspectives as possible. For example, the repository should include knowledge of what organizations or vendors are doing what and how their products impact reengineering or reverse engineering.

The group decided that this effort would benefit from case studies to understand the full complement of the issues surrounding reengineering. This should include the process and techniques used and the experience of the effort. This should not include tools, because of the ever changing field of tool support.

The group also decided that the repository being discussed may provide too much information to be useful. Therefore, some sort of search mechanism should be provided to better search through the possible reams of information.

A possible solution for the repository was the creation of a mosaic repository. The SEI would be the caretakers of the “mosaic files for reengineering” with multiple authors contributing to the repository. The repository searches would be by keywords or summaries of the material.

All members of the working group agreed that if the SEI created this “best practices” repository, the group would be the focal point for review. In addition, Mr. Tilley and Mr. Stump agreed to provide written materials for the repository. For example, Mr. Tilley stated that he is part of a group that will publish a document dealing with the state of the art for reengineering. This document is expected by mid-summer and will be available for inclusion into the repository.

Mr. Tilley also stated that he has access to slides discussing the state of the practice which would also be available. The group agreed that e-mail would be the medium by which the information would be passed out among each other.

## 6.7 Using Lessons Learned Working Group Session

### 6.7.1 Participants

1. Walter Lamia, Software Engineering Institute (technical lead)
2. Mike Webb,Texas Instruments / Software Engineering Institute Resident Affiliate (minutes taker)
3. Karen Kalil Brown, Raytheon
4. Donn DiNunno, Computer Sciences Corp
5. Reginald Hobbs, Army Research Labs

   He is a computer scientist on the Software Engineering Team at ARL-STB. Mr. Hobbs has research interests in the areas of systems reengineering, CASE tools, and hypermedia applications.
6. Lisa Levine, Booz Allen Hamilton
7. Willie Pickens, Loral Federal Systems
8. Fran Pobletts, National Security Agency (NSA)
9. Stephanie White, Grumman Aerospace

   Ms. White is principal software engineer who leads an inter-divisional team solving system/software engineering interface issues and serves on the Integrated Methods Technical Advisory Group at the Software Productivity Consortium. Ms. White's primary research interests are system and software requirements modeling, analysis and traceability of system behavior, and reengineering. She serves as vice-chair of the IEEE Computer Society Task Force on the Engineering of Computer-Based Systems (ECBS) and previously chaired its State of Practice Working Group.
10. Wayne Sherer, Army Software Technology for Adaptable, Reliable Systems (STARS) program

    Over twenty years working on Army and Air Force Mission Critical Computer Resources (MCCR) or Battle Automated Systems (BAS). Work has included acquisition review, coordination, tailoring and policy/compliance activities for BAS; contract monitoring for acceptable performance; development and Post Deployment Software Support of BAS software; and management of Software Process Improvement (SPI), configuration management (CM) and reuse efforts at Armament Munitions and Chemical Command (AMCCOM), Life Cycle Software Engineering Center (LCSE).
11. Bill Hefley, Software Engineering Institute

    Mr. Helfley is a member of the technical staff, Software Engineering Institute at Carnegie Mellon University, Pittsburgh, Pennsylvania. His efforts at the SEI have focused on technology transition, most recently in applying networked hypermedia services, such as the World Wide Web, to the delivery of software engineering information. Previously with Lockheed Missiles and Space Co., his experience is in Systems Engineering, Human Factors

Engineering and Software Development functions. He has contributed in the areas of requirements and software development for numerous spacecraft command and control systems, both ground- and space-based and has been a technical lead in both Software Engineering and Human Factors/Mission Operations Analysis positions. He has also contributed to the development and delivery of financial and manufacturing systems for manufacturing firms in both heavy manufacturing industries and in the semiconductor industry. Mr. Hefley has 19 years' experience in these roles in both commercial and DoD software development environments.

### 6.7.2 Introduction

The "Use of Lessons Learned" working group will consider how to effectively use information about the prior experiences of other projects which have undertaken software reengineering efforts of one sort or another.

Some of the individual topics to be discussed are:

- What types of data are important to know about referenced projects? What data are irrelevant?
- What decisions would you make differently if you had access to this information?
- What level of detail about the data is necessary?
- How can confidentiality of sources be maintained?
- How (i.e., what mechanisms) are needed to ask additional questions to get more detail?
- What forms and formats would be most effective for communicating and distributing lessons learned information?
- What access methods are most interesting/useful/effective?

It has been observed that many more 'lessons learned' are written than are read and used. Why is this? What is the 'value added' of having information about other projects. What are the impediments that mitigate against the use of this information? Why would you NOT want to use information about experiences of other projects? Is it information quality, quantity, accessibility, or something else?

Are quantitative data more important than qualitative and evaluative data? Would the interpretations of others be accepted without having access to, or being able to reproduce, the numerical data analyses?

What would motivate your organization to contribute to a 'lessons' data collection? What would it take to justify the expenditure of resources to do it? How much time/effort/money could you realistically allocate to such an effort? What tools/techniques or other aids would make it easier to participate?

### 6.7.3 Draft Outline for Best Practices

The working group did not develop a formal outline for the best practices handbook. Instead, they elaborated on various issues and concerns with respect to a central repository or "infobase" of lessons learned information. The results of the working group discussions are detailed in the following sections.

### 6.7.4 Principal Users of Accumulated Lessons Learned Infobase

The following potential users of lessons learned about reengineering projects were identified, and prioritized with a Normal Group Technique-style multi-vote. The user roles selected for further consideration are labeled with (*).

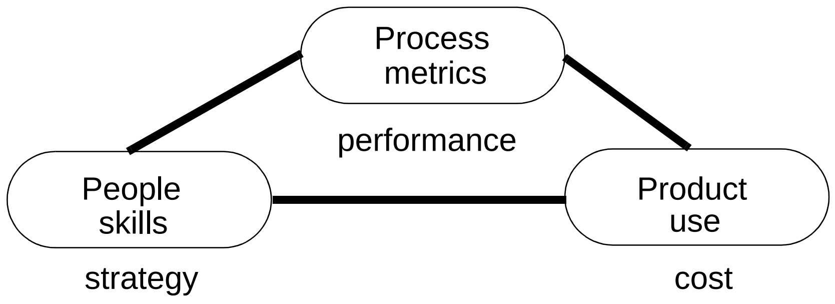
1. Process engineers (e.g. Software Engineering Process Group members) *
2. Reengineering project leaders *
3. Strategy decision makers *
4. Bid proposal team
5. Domain analyst
6. Systems engineers
7. MIS director / chief information officer (CIO) / program executive officer (PEO)
8. Systems architect
9. Cost estimators
10. QA staff
11. Customer/sponsor
12. Users
13. Tool evaluators
14. Facilitators
15. Test and evaluation staff
16. Technical marketing
17. Risk assessment team
18. Program managers
19. Domain architects
20. Individual engineers
21. Technical trainers

### 6.7.5 Questions Users Might Wish to Answer with Access to Infobase

#### 6.7.5.1 Strategy Decision Makers

- What is it?
- Why do it?
- How much will it cost?
- Return on investment (ROI)?
- Impact to staff/scheduling
- Training?
- How long will it take?
- How quickly can it be done?
- What are the pros/cons?
- Market concerns? (competitiveness)
- Compatibility (inter-, intra-operability)
- Politics?

#### 6.7.5.2 Process Engineers



- Process models

  What is our process? - techniques, methods, tasks
  How did we tailor others' models?
  What reengineering process models exist?
  How can they be tailored?

- Process activities

  What tasks are most important?
  What tasks take longest?
  What tasks have most risk?
  What tools/methods are appropriate?
  Who is the stakeholder?

- Process methods

How have others integrated methods?
- method names
- integration mechanisms

Problems in use? - strengths?
Coverage - widely used? - % success - why?

- Process tools

  What does it do?
  Where is it applicable?
  What information does it provide?

- Product quality

  When is reengineering appropriate?
  Case studies
  Return on investment (ROI)
  Risks
  Non-functional issues
  What are the goals that drive this effort?
  What is the cost of poor quality?

- Product measurement

  Software complexity, etc.
  Software size
  Software cost
  Business/strategy
  Costs/schedule
  Project metrics

- People-training

  What activities are to be performed?
  What skills are needed/desirable/optimal (technical and non-technical)
  What is our current skill level?
  What training has been used by others? -success measures? -lessons learned?
  What training is available (cost, schedule)?

- People-cultural change

  What are the impacts to our organization?
   - are we ready for reengineering?
   - standards for measurement

  What are the critical success factors?
  Team composition?

- People-staffing

  How many people were needed?
  What were the team characteristics?

#### 6.7.5.3 Reengineering Project Leader

- What processes have been used successfully/failed?
- Budget/schedule impact of reengineering?
- What methods?
- Impact on maintenance?
- Skills for the staff?
- Is there training/tools to support the process?
- Did the customer get involved?
- Impact on labor requirements
- Who has expertise? Who can help?
- How did it impact the operational system?
- How effective was it?
  - original vs. actual estimates
  - comparisons with comparable projects (with and without reengineering)
- Process identified - where is it available?
- What does it cost? (acquisition)
- Were there any modifications to process, and rationale for them?
- Would you use it again? Why or why not?
- What would you change?
- How were maintenance costs (distinguish hardware and software costs) affected?
- What are the maintenance effort/defect rate differences?
- Mean Time Between Failure (MTBF) data? Mean Time to Repair (MTTR) data?

### 6.7.6 Data Items Required to Answer the Questions

- Glossary of terms and components
- Risks
  - alternatives
  - time to implement
  - selection criteria
- Benefits
  - ROI
  - shorter life cycle
  - vendor support
- Metrics (old vs. new)

- LOC/Man Month
- productivity gains
- cost
- maintenance
- upgrade
- training
- investment
- development
- operational costs
- externally driven costs

- Payoffs
  - benefit/cost ratios
- Interfaces
- Specifications
- New work guidelines/procedures
- Skill mix data
  - tasks/roles, years of experience on legacy system
- New language?

**Table 2: Staff Skill Levels**

| Skills/experience with: | Before starting REE | Needed to do REE | Need post-REE |
|---|---|---|---|
| Total experience level | | | |
| Tools | | | |
| Methods | | | |
| System | | | |
| Process | | | |
| Tailoring | | | |

- What was the training plan?
- How much people loading, by time and task?
- Impact on operational system.

- Did you have to take down the operational system?
- What were the performance metrics and their values, before and after?
- What were the customer satisfaction metrics, before and after?
- How much has the system lifetime been extended?

### 6.7.7 Decisions About Reengineering Projects that Would be Influenced by Infobase

- How many people to assign.
- What kinds of people to assign (skill mix).
- How long it should take.
- How much it should cost.
- What (tools, process, methods, platforms) to use.
- What benefits should be expected from reengineering.
- What metrics should be used.
- What training should the staff receive.
- Should we reengineer (a product/application).

### 6.7.8 Logistics of Constructing, Maintaining, and Providing Access to a Useful Infobase of Lessons Learned about Reengineering

#### 6.7.8.1 Incentives and Barriers to Organizations to *Contribute* Lessons Learned

**Table 3: Incentives and Barriers**

| Incentives to contribute to lessons learned | Barriers to contribute |
|---|---|
| visibility | unwillingness to “airing dirty laundry” |
| good publicity | liability/slander |
| organizes thoughts | information is power |
| get more out than put in | copyright and royalties |
| altruism | takes time |
| anonymity | classified data |
|  | no record keeping |
|  | validation of data |
|  | resistance to paperwork |
|  | competitive advantage |
|  | proprietary information |
|  | humility |
|  | discounting value of information/experience |
|  | embarrassment |
|  | inability to write well |
|  | technical barriers to access |
|  | multiple repositories |
|  | common goals of contributors |

#### 6.7.8.2 Barriers to the *Use* of Lessons Learned

- Vocabulary ambiguities
- Value added hard to quantify
- Network access restrictions, technical and administrative

- Finding the right information
- Time available to do research
- Quality of information - opinions vs. objective data
- Relevance of lessons learned to project
- Timeliness
- Organizational culture
- Novelty
- Write-up of inappropriate expertise level of lessons learned
- Formatting
- Communications technologies
- Barriers to using new access technologies - it’s new, it’s different

#### 6.7.8.3 Design Alternatives for Structuring Lessons Learned Infobase

- Tree structures
- Q&A forums (such as CompuServe forums, Internet newsgroups)
- Free text with full text searching facilities
- Forms and templates for information collection
- Relational data bases
- Scenario-based cases
- Hypertext threaded documents

### 6.7.9 Future Plans

- Collect catalog of existing infobases.
- Collect samples of meta-data, data collecting instruments.
- Consider **comp.reengineering** newsgroup.
- Utilize reengineering mailing list - **reengineering@sei.cmu.edu.**
- Create a standard data collection instrument.
- Establish an Internet repository of reengineering lessons learned.
- Investigate utilizing existing or proposed repositories:

  Best Manufacturing Practices in the “Program Managers Workstation System” - Navy

  “ALLCARS” system - Air Force

  National Software Data and Information Repository prototype - Unisys GSG

## 6.8 Reengineering Process Models Working Group Session

### 6.8.1 Participants

Members of the working group

1. Bill Ulrich, Technical Strategy Group Inc. (technical leader)

   He is president and founder of Tactical Strategy Group, Inc. (TSG), a consulting firm specializing in strategic software reengineering planning support. Ulrich's Redevelopment Framework $^{TM}$ has served as the basis for information migration strategies as I/T organizations world-wide. Fortune 1000 companies, governmental agencies, software engineering vendors and consulting firms have sought his advice in the field of software reengineering. He is the developer of the first and only commercially available software reengineering methodology, The Systems Redevelopment Methodology (TSRM). Ulrich's specialty is his ability to examine large I/T environments, assess future information needs and assist organizations in developing cohesive information redevelopment strategies. Before TSG, Ulrich was Director of Reengineering Products at XA Systems where he had international responsibility for the reengineering product set. Ulrich also served in a senior management capacity at KPMG Peat Marwick as Director of Reengineering Product Strategies and managed numerous software reengineering consulting efforts.

2. Alice Muntz, Hughes Information Technology Corp (facilitator)

3. Alan Christie, Software Engineering Institute (minutes taker)

4. Marc Kellner, Software Engineering Institute

5. Karl Lebsanft, Siemens

6. Bob Moore, Vitro Corp

7. Tamra Moore, Defense Information Systems Agency / Center for Information Management (DISA/CIM)

   Ms. Moore is the author of the CIM Software Systems Reengineering Process Model. She is employed at the CIM in the Software Systems Engineering Directorate, Reengineering Division. She is also working on the CIM Software Reengineering Project Planning Guide. Previous assignments include the Information Systems Criteria for Applying Software Reengineering and other related software reengineering activities within CIM. Ms. Moore was previously employed at the Naval Surface Warfare Center, Dahlgren Division, where she led a Systems Reengineering Technology Project in the Engineering of Complex Systems Program. Ms Moore has been employed by DoD for over 7 years.

8. Larry Wear, Tandem Computers

   He is program manager in the Software Engineering and Services division. He is currently managing a program to improve the design process at Tandem. He has also developed and implemented a post-project evaluation process. Prior to joining Tandem, he was a professor of Computer Science and Engineering at California State University Chico.

9. Dave Workman, Science Applications International Corporation (SAIC)
10. Angelika Zobel, Siemens / Carnegie Mellon University

### 6.8.2 Introduction

#### 6.8.2.1 Initial Discussions

The meeting started out with each member introducing him/herself, after which Alice Muntz proposed an agenda for the day's activities. This agenda included defining the objectives of the group and a plan of attack. Initially the agenda included defining a process model, but the agenda was modified as the session progressed and a sense of direction was established.

Early on, there was considerable discussion about the direction the group should take, and what our relationship to the other reengineering working groups should be, about what reengineering methodology is, and what the deliverables from our group should be. With regard to the latter point, there was general agreement that the main deliverable should be an outline to the guidebook's chapter dealing with process issues. An initial cut at this content was:

1. Describe the relationship between software engineering process and reengineering process.
2. Provide guidance to management decision making, project planning, scoping.
3. Identify and describe activities and their inter-relationships in a model, and
4. Describe various uses for the model based on strategic and tactical objectives and scope.

This content was later modified as will be described below.

In order to focus our attention, Marc Kellner suggested that we address the following issues:

- Who is the audience?
- What is the purpose?
- What does the audience need to know?

With respect to the first point, two audience groups were identified:

1. Government and commercial
2. Management (e.g., redevelopment planners, implementers) and technical (e.g., maintainers, developers).

It was felt that both government and commercial interests could be covered but that the main (but not exclusive) focus should be a management rather than a technical audience. While the second point (what is the purpose?) was not covered in any depth, the third point (what does the audience need to know?) was covered in some detail and provided the mechanism through which the chapter outline was actually organized. With respect to needs, a variety of points were brought up. These are listed below:

- Need to identify categories of activities that could be performed by the different individuals involved in reengineering (Bill Ulrich).
- The process model should support all reengineering objectives/scenarios (Alice Muntz).
- Reengineering scope can be supported by a matrix correlating reengineering objectives against scenarios (Marc Kellner).

Also identified were issues related to the organization and content of the guidebook:

- We need to assure consistency between chapters in the book (Tamra Moore).
- Each chapter should stand on its own (Angelika Zobel).
- The guidebook should describe the relationship between sections, since there are ties between the content of, for example, the process chapter and the decision analysis and economics analysis chapters (Alice Muntz).
- We need to include information on the process of discovery in reengineering resource planning, manpower, etc.

Prior to the afternoon session Marc Kellner listed a set of 10 questions which he thought the potential audience for this section would need to have answered. To a great extent these questions were distilled from morning discussions.

#### 6.8.2.2 The Ten (Supportive) Questions from Potential Readers

1. How do reengineering processes relate to software engineering processes, lifecycles, etc.?
2. What are the key activities in reengineering and how are they inter-related?
3. What - to some degree of detail - is the content of each reengineering activity?
4. What are the major strategic/tactical objectives and tactical scenarios for reengineering? Which activities support each of the strategic/tactical objectives and typical scenarios?
5. How do/should reengineering processes relate to software process improvement (e.g., CMM, SPA, ISO9000)?
6. What are the available process models for reengineering? Also, what experience with them, evaluations of them, and lessons learned would help me in deciding to use them?
7. How do reengineering processes contribute to planning, controlling, managing?

8. How do we select methods, methodologies, techniques, approaches, and tools to support reengineering processes and activities?

9. Can my organization perform this process? What skill set do I need to perform the process and what skill set do I need for each activity?

10. What are the various uses for the model?

Based on these questions, an outline for the chapter was developed. This was done by grouping the questions into related clusters and identifying common themes.

### 6.8.3 Draft Outline for Best Practices

**I. Introduction: the ten questions and a roadmap to the rest of the chapter.**

**II. Why do you need a reengineering model?**

**III. Relationship to the Software Engineering Process**
Supportive questions: 1, 5

**IV. Reengineering process model breakdown**
Supportive questions: 2, 3

**V. How to use the reengineering process**
Supportive questions: 7, 9, 4

**VI. Bridge to technology and tools**
Supportive question: 8

**VII. Models in practice**

- **A. Model #1 uses, experiences**
- **B. Model #2 uses, experiences**

:
:

Supportive question: 6

### 6.8.4 Miscellaneous Issues

Some discussion was also given to why we need a reengineering process model. We need a reengineering process model to:

- Define resource requirements.
- Define staffing requirements.
- Delineate process to assign costs and benefits of project and its activities.

- Identify how reengineering augments/supports development and maintenance efforts.
- Assist in identifying specific deliverables for a reengineering-supported project.
- Define reengineering activities and lifecycle interface.
- Avoid reinventing the wheel.
- Assist with buy-in at all levels.
- Define tool requirements.
- Assist in defining strategic objective.
- Assist in implementing tactical requirements.
- Assist in defining scope of applications to be reengineered.
- Define scope of reengineering activities to be performed.
- Define and control the costs of reengineering.
- Delineate reengineering activities from standard maintenance activities.
- Establish a baseline model of activities that can be combined based on a given scenario.

In addition, roadblocks to the implementation of a successful reengineering project were discussed. These roadblocks were:

- All management wants is to adapt the code.
- Resource and funding limitations.
- Management doesn't know what reengineering will buy them.
- A lack of strategic objectives
- Management doesn’t understand the need for reengineering technical support.
- Lack of buy-in at all levels.
- Security considerations associated with non-employees.
- Existing production staff don't want to be bothered.

There was some debate about the differences between reengineering and the enhancement side of maintenance. A suggested major difference is that reengineering is tied to strategic objectives and is proactive rather than being a patch to the system (Alice Muntz). Reengineering often as to deal with architecture issues, redocumentation, and design recovery (Karl Lebsanft).

### 6.8.5 Process Models Identification

The group finished the first day by identifying reengineering process models that have been used in practice. This list included:

- NSA standard prototype process.
- HITC (Hughes Information Tech. Corp) data reengineering process..
- TSRM (The Systems Redevelopment Methodology).
- CIM (Center for Information Management) software systems reengineering process model.
- VITRO Corp.
- DSAC (at Indianapolis) software reengineering process model.

The criteria for a model's inclusion in the Process section are:

- Model must have been used multiple times.
- Model must be of sufficient detail to be useful.
- Models included must provide insights for discussion in the rest of the chapter and to a lesser degree provide information on what vendors are selling.

At the end of the first day, the following information was presented to all workshop members:

## Overheads: day 1

Overhead #1: Objectives of the process track

- Develop outline of the chapter
- Activities breakdown
- Follow-on activities
- Issues/action Items

Overheads #2, #3: Outline of the chapter (see chapter outline above, 6.8.3, Draft Outline for Best Practices)

Overhead #4: Issues and action Items:

- Guidebook needs an Abstract, Executive Summary or Preface
- Guidebook needs Introduction to define:
  - audience
  - relationships between chapters
  - clarification between reengineering and software engineering
  - why reengineering?
- Organizational Impacts
  - team composition
  - retrain (technology insertion, physical evidence)

- Should Decision Analysis and Economic Analysis be one group?
- Decision analysis, economic analysis, and reengineering process model are tightly related.

### 6.8.6 Review of Reengineering Process Models

In day two, the process group focussed in generating an activity breakdown. Four members of the process group (Bob Moore, Tamra Moore, Alice Muntz, Bill Ulrich) identified the highest level activities in their process models as a starting point. As will be shown later, these activities were used as a basis for defining a generic process sequence, as will be shown later. The four breakdowns are as follows:

Hughes' model (Alice Muntz):

- Audit phase
- Analysis phase
- Requirements phase
- Migration phase
- Redocumentation phase

Tactical Strategy Group model (Bill Ulrich):

- Enterprise redevelopment planning
- Inventory analysis
- Positioning
- Transformation

Vitro Corp. model (Bob Moore):

- Prepare to reengineer
- Reverse engineer
- Develop requirements and tests
- Engineer
- Test and deliver system

DISA/CIM model (Tamra Moore):

- Define project
- Reverse Engineer
- Forward Engineer

Some discussion then took place to understand the terms which the various reengineering processes were using. The following is a brief review of that discussion. For example, in the Tactical Strategy Group model, Bill Ulrich made the following clarifications:

Enterprise redevelopment planning

- This occurs above the project level.
- Involves high-level data architecture.
- Addresses conflicts within the existing system.

- Results in a transitional plan and identifies the reengineering project.

Inventory analysis

- Identifies inventory.
- High-level E-R of application developed.
- Cross-mapping to target model established.
- Output is a detailed reengineering plan.

Positioning

- Code improvements (only locally restructures code).
- Primes system prior to transformation.

Transformation

- Transformation of positioned to reengineered system.

### 6.8.7 Generic Activity Set

There was general agreement that these four processes had much in common and that a "generic" sequence of activities could be synthesized from them. Seven major activities were identified. These are:

1. Plan redevelopment strategy
    - Identify problems.
    - Assess situation.
    - Define strategic requirements.
    - Develop overall redevelopment recommendations.
    - Determine redevelopment priorities.
    - Commit to redevelopment projects.
2. Develop detailed project plan
    - Develop assessment plan.
    - Inventory assessment - programming environment, hardware, database.
    - Assess system and operational environment.
    - Develop Business Process decomposition.
    - Determine feasibility and alternatives.
    - Identify technology and tools.
    - Define team infrastructure (skills etc.).
    - Develop detailed reengineering plan.
3. Reverse engineer
    - Recover test data, test plans.
    - Produce documentation.
    - Recover dynamic characteristics (timing, control flow, event triggers).
    - Recover design.

- Recover requirements.
- Recover architecture (data, application, infrastructure architecture).

4. Analysis and redesign
    - Develop to-be requirements.
    - Identify discrepancies between as-is and to-be requirements.
    - Develop test environment (to-be and as-is).
    - Develop test plan and data.
    - Develop QA plan
    - Develop migration path.
    - Develop traceability matrix.
    - Develop to-be design.
    - Normalize process (of code) and data.
    - Develop integrated design.
    - Perform quality assurance.
5. Implementing the system
    - Restructure the code.
    - Identify reusable, duplicate, or dead code.
    - Select reusable code
    - Implement new code.
6. Deliver redeveloped system
    - Test.
    - Integrate.
    - Migrate data.
    - Deploy application.
7. Synchronize operational system
    - Retrain personnel.
    - Baseline existing system to be reengineered.
    - Deploy usage of reverse engineered products.
    - Participate in reverse engineering and implementation of system.

The above activities are not necessarily sequential. In particular, 'Reverse engineer' and 'Analysis and redesign' may be iterative. Also, elements of activity 7, Synchronize operational system, can occur throughout the whole reengineering process.

The above activities formed the basis of the presentation given by Bill Ulrich of the process group to the workshop members on day two.

### 6.8.8 Future Plans

Several items for future work were identified and discussed. They include breaking down the identified process activities further, describing the relationship between the working groups, and answering the questions identified in Section 6.8.2.2.

## 6.9 Software Reuse Working Group Session

### 6.9.1 Participants

1. Grady Campbell, Software Productivity Consortium (technical lead)

   Worked on Parnas's A-7 project at the Naval Research Laboratory (NRL), concerned about how to make systems more maintainable and reuse driven processes. Believes objectives of an organization determine the processes an organization should use. Reuse is a central idea that should be part of processes. Leads to the idea that organizations should take product-line approach to development. Recognizing variability is central to having a product-line. Beginning to look at reengineering, feels that reuse is central to reengineering, and that we should reengineer in context of product-line employing reuse.

2. Dan Burton, Software Engineering Institute (minutes taker)

   Works on Software Technology for Adaptable, Reliable Systems (STARS) program that is currently sponsoring demonstration projects in the Army, Air Force, and Navy that are applying domain-specific reuse approaches to the reengineering of legacy systems and development of new systems.

3. Sholom Cohen, Software Engineering Institute

   Works on Application of Software Models project at the SEI. Original area was Domain Analysis with the intent to define application areas that could be used on multiple systems. Completed domain analysis of Army movement control domain, including the software assets. The Army hasn't used this yet because of other problems. Now enhancing the movement control system with geographic and database capability. Working with the Navy group on a set of trainers (in procurement now, so can't say much). Trying to take into account changes that will take place over the system life and considering how to redesign the system to be more maintainable and anticipate issues that will come up a few years down the pike. How do you account for variability, what process do I follow to get capabilities needed today, three years out, or even later? Don't believe you can design in, but must be able to incorporate when needed, looking at how to encapsulate. Wants to achieve three things: method, a way to represent variability, and a design method for building. In general, wants to see what kind of standard products can be created that can be used to reengineer systems and make them more maintainable.

4. Bob Johnson, Army/Joint Logistics Command (JLC)

   Director of the Army Reuse Office and works on JLC reuse/reengineering activities, reuse/reengineering handbook, technical and economic aspects, trying to get in a state that supports services. Doesn't think economics of reuse/reengineering supports what program manager faces. Going to as many forums as possible to review/test documents to improve it (same handbook discussed by Chris/Mike yesterday). Gets information to improve document. The Army is institutionalizing reuse, sent plan to Congress in 1993. Has 412 tasks but no real funding. Difficulty trying to implement, scoped down some, and maintaining data base of tasks. How to support program executive officers (PEOs): find technology and tools that can be used, find targets of opportunity that can be done with minimal resources. In general, is looking for products (like reuse policy that got through legal, now going through staff), done PEO handbook, being reviewed now, trying to follow CARDS as most closely matches philosophy. Big mission, little budget to support, been in business for a year, policy, handbook, etc.

5. Dale Sampson, Air Force Audit Agency

   Accountant background. Does reviews of systems to see that accounting systems do what they're supposed to do. Traditionally goes in and audits after the fact and tell folks they did it wrong. Really want to let folks know before it's over that it's not right. The purpose here is to find out about tools/ methods that could help in the review process. Wants to be pro-active as consultants at the front end rather than auditors at the end telling people that they didn't do the right thing.

6. Farnam Jahanian, University of Michigan (former IBM)

   Interested in distributed & fault tolerant systems, creating building blocks, incremental reengineering by taking small, well-defined pieces of system and reengineering them.

7. Fred Hathorn, former DoD employee that worked with Paul Straussman

   Gains to be realized from reengineering, example of 193 KSLOC of Cobol reengineered to 43 KSLOC of reusable code. Results like this influence the decision makers to fund reengineering efforts.

8. Wayne Sherer, Software Technology for Adaptable Reliable Systems (STARS) Army Deputy Program Manager

## 6.9.2 Introduction

The purpose of the panel is to investigate the relationship between reuse and reengineering and to recommend a chapter outline on this topic for the reengineering handbook.

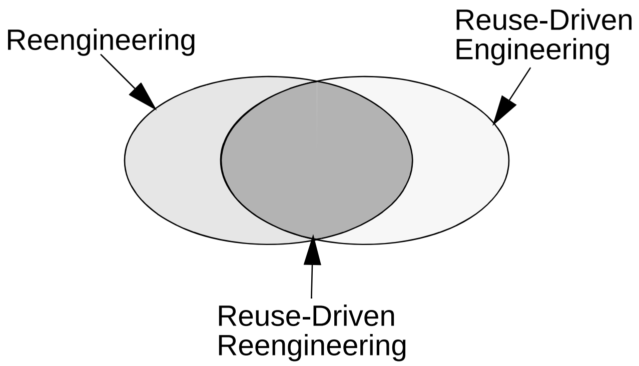
### 6.9.2.1 Relationship between Reuse and Reengineering

The group initially discussed some aspects of reuse:

- People moving away from library of components approach to some type of framework (or product line) based reuse.
- Predicting variation is key to reuse strategy, i.e., how can you predict where technology is going so you can predict the type of system you want to build in the future. (Example of FAA system cited as one where technology has changed so much since requirements were identified.)
- Requirements that are too stringent can hinder reuse (example of FAX requirements for president's aircraft).
- Need to define process for creating reusable assets so that others can be taught how to identify reusable components when developing a system.

For an in-depth discussion of reuse and the associated concepts, see the STARS Conceptual Framework for Reuse Processes (CFRP), Unisys STARS Technical Report, STARS-VC-A018/001/00, October 1993.

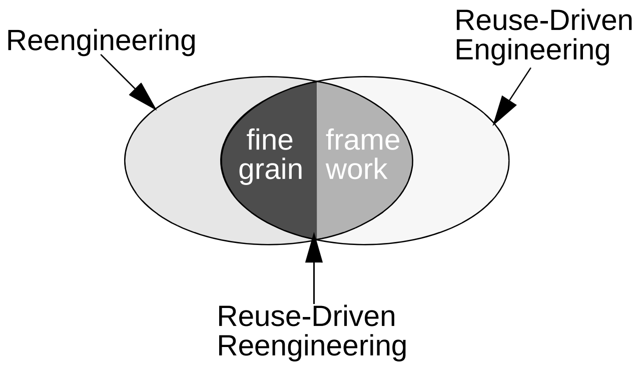
.



After significant discussion, we came up with a model, which is shown above. The oval on the left represents the universe of reengineering, the oval on the right represents the universe of reuse-driven engineering, and the intersection represents reuse-driven reengineering.

The non-intersecting part of the oval on the left represents those instances where a single system is reengineered with no consideration for reuse, i.e., no effort is made to create or utilize reusable assets. We acknowledge that in reengineering, reuse of some (or all) of the assets of the system being reengineered is occurring, but that really isn't what we call reuse-driven engineering, i.e., creating and/or utilizing reusable assets.

The non-intersecting part of the oval on the right represents those instances where reuse is applied on the development of new systems and no reengineering of existing systems is involved. The intersection of reengineering and reuse-driven engineering (reuse-driven reengineering) breaks down into two major sub-categories: reuse-driven reengineering, using fine-grained reuse assets; and reuse-driven reengineering, using framework (or product line) assets.

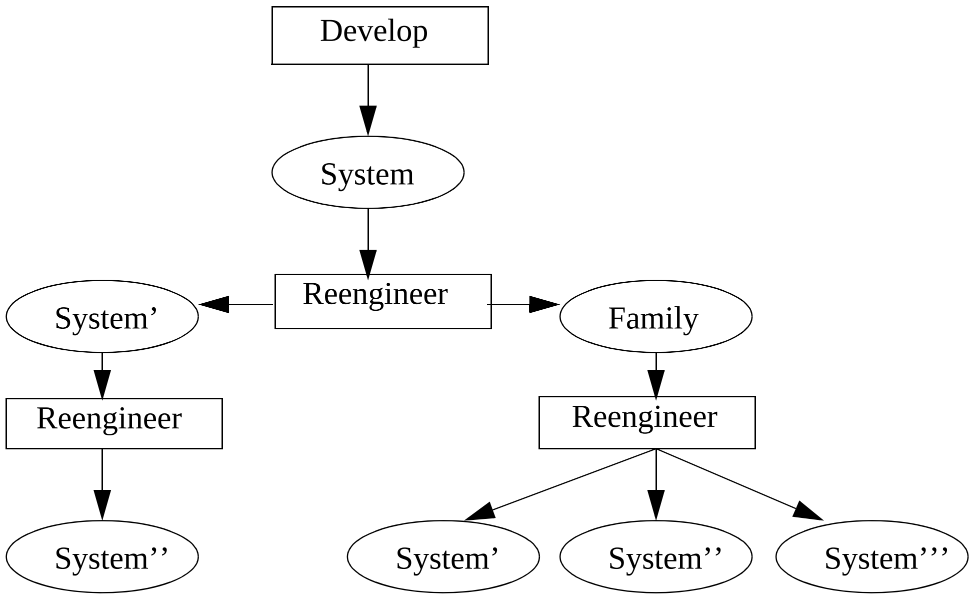


A further refinement has to do with the creation and utilization of reuse assets. The reuse-driven reengineering may utilize assets created elsewhere and create no new ones; it may not utilize any existing assets but instead may create reuse assets for other efforts, or it may be a combination of utilization and creation.

Combining the creation/utilization with the fine-grain and frameworks/family product line gives the following characterization.

1. Reengineering for/with fine-grain assets
    a. identifying opportunities for reuse within an application under reengineering
    b. producing small-grain components for follow-on efforts
    c. continuum small -> large grain
2. Reengineering for/with product family assets
    a. domain engineering of a product family
    b. using a product family

The following diagram offers another view based on a product line or family of systems. The left side of the diagram represents the reengineering of a single system with no reuse, the right side of the diagram represents the concept of reengineering a family of systems (a product line) in which reuse-driven reengineering is employed to create and utilize a product line architecture and set of reusable assets.



In a rapidly developing market, a company may decide that reuse is the answer to meeting that demand, but in most cases, justifying reuse is hard. Reengineering existing systems to reduce maintenance costs is easier to justify and is being done today.

### 6.9.3 Draft Outline for Best Practices

## I. Retrospective view

**A. Reengineering**

1. Transformation of a legacy system to a new system.
2. Focus on meeting changing requirements.
3. Focus not on generating reusable assets (or components).

**B. Reuse-driven engineering**

1. Approaches
   a) generating reusable assets within a system (small-grained reuse) e.g., class libraries
   b) generating reusable assets for a family of systems (large-grained reuse) e.g., frameworks

## II. State of the art/practice

**A. Characterize alternative hybrid approaches (purpose: select among/ tailor)**

**B. Best practice**

(Should have sources of information/experience as well as recommendation for best practice.)

1. What is domain engineering?
2. Use of tools?
3. Current approaches
   a) method
   b) benefits - cost avoidance example of what this means to the General, i.e., how many tanks will this buy
   c) prerequisites, inputs
   d) results, products
   e) supporting tools
   f) limitations, scope
   g) dependence of method on legacy domain
   h) provides or inhibits solution to attain open systems standards
   i) special system properties addressed, e.g., security
   j) dependence on operating environment
   k) cost avoidance evidence
   l) effect on maintainability
   m) approach to integration of mixed assets
   n) validation and verification approach

## III. Challenges

### A. Technological

1. What does domain engineering contribute to reengineering?
2. What are categories of reusable assets?
3. How do we reuse assets to assist in reengineering?
    a) system understanding
    b) reverse engineering
    c) forward engineering
4. Where does domain architecture fit? Are architecture and frameworks the same for reengineering?
5. Understanding assets that support the reengineering effort. What levels of assets (small-grain => large-grain, product family) are appropriate?

### B. Making a business case

1. Education and training
    a) identify/focus on internal concerns and issues
    b) identify/focus on cause of resistance
2. Success stories and return on investment - reference multiple environments/domains
3. Identify relationship of their effects/accomplishments both vertically and horizontally across domains

## IV. Improvement opportunities

### A. See Challenges

## V. Transitional Issues

### A. Conventional

## VI. Prospective view

### A. Model-based

### B. Prototyping oriented

## 6.9.4 Recommendations

1. Make unified treatment of reuse and reengineering a theme of reengineering best practices.
2. Recognize that business objectives determine level of reuse emphasis (and character of reengineering process).
3. Documented results should be used to substantiate best practice claims to support adoption of (reuse-driven) reengineering.

## 6.10 Reengineering Technology & Tools Working Group Session

### 6.10.1 Participants

1. Elliot Chikofsky, Defense Management Review (DMR) Group and Northeastern University (technical lead)

   CASE developer since the mid-70's. Consultant on IT technology transfer. Researcher on reverse engineering and reengineering. Chair of the IEEE technical committee on software engineering.

2. Len Green, Software Engineering Institute Resident Affiliate / National Security Agency (NSA) (facilitator)

   Member of the Reengineering Center. Also a member of the CASE Environment Projects, which is currently running an integration-focused reengineering experiment. Previously the software development manager from the research and development organization, has had ten+ years experience in developing real-time software systems that utilized various types of COTS CASE tools.

3. Ed Morris, SEI CASE Environments Project (minutes taker)

   Experience in software engineering tools and environments, including compiler and tool development. Currently co-editor of IEEE P1348 - Recommended Practice for CASE Adoption. Co-author of book, "Principles of CASE Tool Integration," to be published by Oxford University Press, Summer 1994.

4. Bill Ball, FileNet Corp.

   Interested in improving cycle times for product development efforts, automating documentation, facilitating intergroup communications and cross-development efforts, and tracking process and product metrics.

5. Mark Koltz, Decision Systems Technologies, Inc.

   Focus on Legacy systems and reengineering within the MIS/Business domain.

6. Bill Selfridge, SETA Corporation

   Reengineering consultant for government MIS systems. Strong interest in reengineering tools and processes.

7. Ravi Koka, SEEC Inc. - Founder and President of SEEC Inc. SEEC markets a PC-Windows based Cobol maintenance and reengineering tool (COBOL ANALYST). Conducted R&D in the area of software reengineering for last 7 years, including work in the areas of static analysis, language translation, and transformation technologies. Currently interested in tools and technology for migrating legacy systems to object-oriented development environments.

8. Howard Reubenstein, Mitre Corp.

Concern for analysis of older legacy systems (cms2, jovial, assembler). Interested in the language independent analysis of programs, integration of reverse engineering tools into forward engineering environments and processes, and adaptation of reverse engineering tools to DoD-specific needs.

9. Connie Clayton, Science Applications International Corporation (SAIC)

   Contractor to Army. Working on the Life Cycle Software Engineering Directorate. Working on defining reengineering process for the organization.

10. Karen White, The Analytical Sciences Corporation (TASC)

    Seventeen years in MIS (mostly mission critical). Software project leader for large mainframe system. Faced with reengineering that system into distributed client/server application. Formerly reengineered and re-hosted large corporate system from centralized to distributed processing.

11. Lawrence Markosian, Reasoning Systems

    VP for applications development for Reasoning Systems (customized tools from reengineering). Author of various papers on reengineering technology.

12. Carmen Castells-Schofield, Leverage Technologies

    Fifteen years' applied experience with software environments and technologies. Currently applying CASE and operations research techniques to reengineering.

13. Colonel Dan Litynski, US Army

    Professor and Department Head, Department of Electrical Engineering and Computer Science. US Military Academy, West Point, NY. SEI affiliate. Background in physics, optical signal processing, electrical engineering. Broadening knowledge into software engineering and reengineering.

14. Debbie Carr, Electronic Data Systems Inc. (EDS)

    Ten+ years experience. Developed EDS Reengineering Methodology as well as a number of CASE products. Current focus on technology for the transformation of applications.

## 6.10.2 Introduction

### 6.10.2.1 Goal of the Session

The goal of the Reengineering Technology and Tools Working Group Session was to examine the state of technology support for the reengineering process. Potential topics included:

- Reengineering methods.
- Reengineering tools.
- The reengineering environment.
- Repositories and the storage/analysis of requirements.
- Migration to new architectures such as client server, open systems.

#### 6.10.2.2 Presentations

The session was initiated with a short presentation by Elliot Chikofsky identifying the goals and potential topics for discussion. This was followed by short presentations from Ravi Koka and Lawrence Markosian. Debbie Carr presented information about the EDS approach later in the session.

Ravi cited statistics noting that there are about 80 billion lines of legacy code representing a $2 trillion investment. This represents one of the largest investments in the US (after oil). Among the problems that Ravi noted with legacy code were inconsistent documentation, spaghetti code, and limited use of data abstraction, leading to widespread impact and high expense when making changes to the code.

Ravi also identified a number of options for maintenance and some problems associated with these options:

- Automatic translation to modern languages is technically complex, and often cannot exploit the advantages of new technology. Automatically translated code can be hard to maintain.
- Redesigning and rewriting code using CASE tools involves high up-front costs and high risk (since there are few experts), and CASE methodologies and platforms are a moving target.
- Often the best option is to reengineer existing code. This approach is often practical, economical, and leverages past investment.

Ravi also contrasted two general approaches to reengineering: the big bang approach and the incremental approach. Ravi's experience suggests that the big bang approach does not work well. The approach involves reverse engineering the entire existing system; building a new system, incorporating enhancements; and changing the development language and platform. Incremental reengineering, on the other hand, involves reengineering parts of the system to ensure that the system continues to operate at all times. A frequent problem with this approach is that you end up with multiple languages and platforms that must be supported. In order for incremental engineering to be successful, the organization must be prepared to address multi-language environments, data type transformations, and data structure conflicts, as well as existence of numerous interfaces (or wrappers) between new and old system components.

In addition to the multiple approaches that can be used, Ravi highlighted the many different goals that organizations must meet in reengineering. End users want ease of change, ease of use, and ease of learning. Developers want readability, maintainability, and extensibility of the system code. Managers often desire multiple platform support, down-sizing, and rapid development of the new system.

The second presentation was made by Lawrence Markosian of Reasoning Systems. Lawrence suggested that large-scale reengineering projects are dominated by their unique requirements (specific platform, company-specific protocols, database interfaces). Therefore appropriate reengineering tools require both shrink-wrapped capabilities as well as extensive customizability. The cost of tool customization cannot be high; Lawrence suggested that customization costs must be returned on the first use of the tool, or it is probably not worth the expense to customize.

Reasoning Systems uses abstract syntax trees to capture program structure, with high level languages and libraries to query and update the resulting database. The approach also uses highly customizable rule-based program transformations.

The third presentation was made by Debbie Carr of EDS. Debbie suggested that EDS places its emphasis on a comprehensive systems engineering environment, including resource planning, business planning, work flow analysis, office automation, software configuration management, and reengineering.

Debbie discussed the Software Performance Engineering for Legacy Systems (SPELS) approach developed at EDS. SPELS is a three-step process involving:

- Assessment of existing software (both business-related and technical aspects).
- Improvement of existing software, involving reformatting, restructuring, modularization, data name standardization, data and language conversion, and distributed presentation.
- Transformation of the software, involving migration, reverse engineering, and redevelopment.

#### 6.10.2.3 Identification of Dimensions for Tool/Technology Evaluation

The group identified a number of dimensions that are important for evaluating tools and technologies. These dimensions are indicated below:

- Platform
- Language handled
- Seminar last attended (note - this is a frequently used but inappropriate dimension)
- Scale of the subject system
- Degree/kind of human involvement necessary
- Ease of use
- Techniques in use by the tool/technology
- The type of analysis (static or dynamic)
- Whether a guru is available
- Availability (including licensing)
- Support
- Price
- Cost to use/support in your environment
- Database supported
- Integration with forward engineering tools
- Level of operation (full system/application/program/statement)
- Openness

- Customizability
- Transaction Processing Monitor (?)
- Learning Curve
- Support available
- Single/multi-user
- Vendor stability
- Other tools from that vendor in house
- Closeness with the vendor
- Leverage with the vendor

#### 6.10.2.4 Components of a Guide to Best Practice

Using a brainstorming approach, the group identified what it would like to see and would not like to see in a guide to best practices. Only a few items were specifically rejected for a guide to best practice. These include extensive tool evaluations and listings of the capabilities of specific tools. The list of items that we wished to see in a guide to best practice is extensive and without doubt includes information that is better placed in other sections of the handbook. However, the complete list is provided here:

- Classifications of types of tools/tool capabilities.
- List of tough questions to ask vendors.
- Discussion of planning for reengineering project.
- Discussion of methodologies, tasks, deliverable, a template or reference model.
- Taxonomy of reengineering projects, classes of projects, examples of projects.
- Set of definitions of technical terms.
- Limitations of state of the art/tool technology.
- Information concerning what can be gained by a specific reengineering activity.
- Cost models.
- What are the metrics to accumulate before, during and after the reengineering effort.
- Things that should be in place before you use reengineering tools.
- Information concerning judging whether a tool was/is good for you
- Discussion of the skills necessary to learn a tool and perform reengineering activities.
- Discussion of how an organization should train in reengineering tool technology, and what happens if you don’t train.
- Case studies of diverse reengineering projects with different tools and toolsets.
- Critical success factors for reengineering projects.
- Success factors for use of technology or tools.
- Collected recommendations for organizations with experience in reengineering.
- Information concerning configuration management of reengineering work in process.

- Information concerning QA of reengineered work.
- Information concerning acquiring proof of transformation
- Guidance or a grid that discusses type of project/system and relates them to different types of (or individual) tools.
- A matrix mapping reengineering needs (artifacts) to capabilities of tools.
- Pointers to additional information about tools, techniques, methods, consulting
- Tools and techniques for managing differences between databases.
- Information concerning how to convince management that you are using the appropriate technology.
- Information concerning the politics of reengineering (personnel management).
- Information on how to modularize reengineering projects, how to come up with short-term objectives, and what steps to take.
- Information about expectations of developers, end users, and managers (management issues).
- A discussion of cultural issues.
- A discussion of ways in which the tools can drive the process (often leading to failure of the reengineering effort).
- Information about barriers to success for managers, end users, engineers.
- Mechanisms for motivating personnel and removing impediments to change.
- A discussion of the linkages between business process reengineering and software reengineering.
- A discussion of who does the reengineering (the same team of maintainers, different people, etc.).
- A discussion of team composition and roles for reengineering.
- A discussion of reengineering as one part of the maintenance life cycle.
- A discussion of reengineering project organization.
- References and suggested readings.
- Information about identifying when a system should be reengineered.
- Information concerning capability assessment for reengineering.
- A discussion of reengineering and changes to the organizations process
- A discussion of when to admit defeat (you bit off more than you could chew) and when to claim success (knowing when you are done).
- Information concerning monitoring the scope and process of reengineering.
- Information about recovering from faulty planning.
- Information about what to do if you selected the wrong tools?
- Information concerning resume writing.
- A discussion about appropriate pilot efforts.
- Guidelines for project planning.
- A discussion about interpreting the claims of a vendor/consultant.
- A discussion about whether to contract out for reengineering services.
- A discussion about facilitating technology transfer from consultants.

### 6.10.3 Draft Outline for Best Practices

The rest of the Tool and Technology Workshop Session was devoted to organizing the information obtained through earlier brainstorming activities and developing an initial outline for the Tool and Technology chapter for a guide to best practice in reengineering.

#### 6.10.3.1 Referred Topics

We determined that some of our brainstorming topics did not fit cleanly into the proposed chapter. These topics included:

Justification of Reengineering Activities

- decision analysis
- economic analysis

Reengineering Perspective

- process models
- understanding legacy

Reengineering Planning

- process models

Impact on the Enterprise

- lessons learned

Technology/Tool Case Studies Needed

- lessons learned

We refer these topics to other groups.

#### 6.10.3.2 First Draft Chapter Outline

## I. Infrastructure for reengineering

**A. Outline of requirements for use of technology**

1. Specific
    a) hardware
    b) software
    c) staffing
    d) training
    e) feasibility/evaluation Group
    f) establish metrics collection

## II. How technology/tools affect the process

**A. Technology impact on staffing**

1. Domain expertise
2. Technology expertise

3. Staffing levels
4. Expertise mixtures

**B. Technical impact on process**

1. Tool customization tasks
2. Effect on decision model
3. Problem recovery
4. Prototyping
5. Pilot project

## III. Planning for using reengineering technology

**A. Critical success factors for use of technology and tools**

1. How to upfront plan?
2. When is reengineering done?
    a) how to modularize for success
3. Tools/metrics which are reengineering candidates (when to start)

## IV. Roles and responsibilities in the use of reengineering technology

**A. System user**

**B. System owner**

1. Maintenance staff/QA/test
2. Administration/CM

**C. Reengineer**

1. Application engineers
2. Tools group
3. Trainers (how much tool transfer from consultants)
4. Consultants/contractors (which roles do they occupy)
5. Software engineers (who performs reengineering)

## V. Available technology

**A. Definition and background**

**B. Current state**

1. Classes of Technology
    a) Hardware/platforms
        (1) State of the art (issues)
            (a) parallel vs. sequential processing, distributed processing
        (2) Limits

(3) Challenges

(4) Opportunities

(5) Future

b) Static Analysis

(1) Definition

(2) State of the art

(a) manual and automatic

(b) interactive vs. batch

(c) program and system level

(d) limited languages (procedural only)

(e) impact analysis, logic analysis, data analysis

(f) metrics

(g) code slicing

(3) Limits

(a) machine size

(b) very large systems (millions of lines) are possible

(c) some performance issues as size of system analyzed

(d) human comprehension of too much information

(e) signal to noise ratio (not focusing on what we need)

(f) event-driven applications

(4) Challenges

(a) no standard metrics

(b) manual approaches are inconsistent

(c) uniformity of approaches between tools (example, tools disagree on slices)

(d) standardization of terms

(5) Improvement opportunities

(a) language availability for all tools

(6) Future

c) Dynamic Analysis

(1) Definition

(a) checking for memory leaks

(b) program optimization

(c) tracing

(d) frequency of execution

(e) timing

(f) code coverage

        (g) data-driven conditions
        (h) event driven
    (2) State of the art
        (a) instrumentation of code
        (b) GUI and event driven systems
        (c) essentially program level (some exceptions)
    (3) Limits
        (a) instrumentation of code
    (4) Challenges
        (a) language dependencies
        (b) integration between static and dynamic analysis
    (5) Opportunities
    (6) Future
        (a) expert system support

d) Visualization
    (1) Definition
        (a) flow charters
        (b) structure charts
        (c) inter-program control flow
        (d) data modeling
        (e) screens
        (f) data flow
        (g) job flow
        (h) hypertext
        (i) 2 vs. 3 dimension
    (2) State of the art
        (a) batch and interactive
    (3) Challenges
        (a) comprehension and ergonomics
        (b) lack of integration with other information
    (4) Limits
        (a) 7 +- 2 for understanding
        (b) density of items displayed
        (c) understanding what you are looking at

e) Procedure Transformation
    (1) Definitions
        (a) restructuring
        (b) translation

(c) reformatting
(d) modularization
(e) procedural to OO conversion
(f) procedural to event driven
(g) code to CASE (abstraction)
(h) code to documentation
(i) code to test case
(j) dead code, code redundancy elimination
(k) improving readability

(2) State of the art
(a) full semantic capture
(b) code generation

(3) Limits
(a) language supported
(b) some tools allow customization, others don't
(c) manual vs. automation

(4) Challenges
(a) some customization often required
(b) incorporating/capturing domain knowledge
(c) integration with analysis/modeling tools
(d) verifying/validation of transformation/correctness

(5) Opportunities

(6) Future
(a) expert systems
(b) proof of correctness

f) Data Transformation

(1) Definition
(a) flat file to dbms
(b) hierarchical to relational
(c) conversion to ER model
(d) centralized to distributed
(e) conversion to OO

(2) state of the art
(a) converting of model
(b) converting of data
(c) redundancy identification
(d) normalization
(e) manual vs. automatic

        (f) data validation
        (g) middleware
        (h) wrappers
    (3) Limits
        (a) proprietary databases
    (4) Challenges
        (a) performance
        (b) synchronization
        (c) impact on processing
        (d) domain analysis
    (5) Opportunities
    (6) Future

g) Abstraction
    (1) Definition - going from a physical model to a logical model
        (a) design capture
        (b) requirements capture
        (c) document generation
        (d) schema abstraction tools
        (e) aid to software understanding
    (2) State of the art
        (a) playing detective/archeologist
        (b) design improvement
        (c) normalized data model
        (d) manual/analyst directed/automatic
    (3) Limits
        (a) abstraction is still an art form
        (b) it depends on the reader
        (c) labor intensive
    (4) Challenges
        (a) understanding varying cognitive models of designers
        (b) maintaining links to other artifacts
        (c) poor understanding of how to deal with abstractions in general
    (5) Opportunities
    (6) Future
        (a) self-documenting
        (b) additional techniques

h) Enactment, enforcement, metrics

(1) For each class
   (a) applicability
   (b) characteristics
   (c) limitations
   (d) requirements for use
(2) Evolution
   (a) desired enhancements
   (b) trends

## VI. Technology/tool selection - evaluation - justification

**A. Tool classification/types**

1. Reformatting
2. Restructuring
3. Re-modularization
4. Reverse Engineering
5. Repositories

**B. Guidelines and rules of thumb**

1. List of things I wish I was told beforehand
   a) an index of either reverse engineering tasks, or system goals to specific tools, technology and tasks and processes.

**C. Guidelines for project planning (pilots)**

1. An index of project size, scope, and risk tolerance to crucial planning steps

## VII. Problems and pitfalls

**A. Need for training**

**B. Management/user involvement and commitment**

**C. Scoping**

**D. Tools driving process**

**E. Scoping**

**F. Tool inadequacy**

**G. Politics of reengineering**

1. Fear of change
2. Fear of job loss

**H. Integration of tools and techniques**

## VIII. Management of technology use in reengineering

**A. Managing Technology Expectations**

1. No silver bullets
2. Prototyping the technology insertion to establish expectations

**B. Problems introduced by technology**

1. Technology resistance
2. Technology enthusiasts

**C. Using technology to support management**

1. Measuring via technology
2. Training via technology
3. Enforcement via technology

## IX. Metrics and measurement of technology/tool use

**A. Before (SE process, product)**

1. Cost
2. Time
3. Bugs

**B. During (RE process)**

1. Cost
2. Time
3. Bugs

**C. After (SE process, products)**

1. Cost
2. Time
3. Bugs
4. Results

**D. Process metrics**

**E. Product metrics**

## X. Training

### A. Current technology

### B. Current business functionality

### C. Target technology

### D. Reengineering tool training

1. Learning curves

### E. Assimilation training

1. Management
2. Users
3. Maintenance staff
4. Developers

## XI. Case studies/ experiences of tools in reengineering

### A. Information to be provided in each

1. Scope (SLOC, #SW units)
2. Platforms (source, target, reengineering)
3. Languages
4. Project team composition and roles
5. Tools and technology used
6. Options/alternatives explored
7. Planned cost and schedule
8. Actual cost and schedule
9. Rationale for approach
10. Metrics used and captured
11. Critical success factors/expectations for tools
12. Informal critique of tool, including productivity impact

### B. Taxonomy of case studies

1. Size factors
2. System type
3. Goals
    a) in-language reengineering
    b) re-hosting
    c) translation
    d) subsystem change
4. Organization type
    a) academic

        b) commercial
        c) government
        d) R&D
    5. Comparative analysis with controls
    6. Technical characteristics
    7. Lessons learned
    8. Points of contact

## XII. Related subjects

**A. Project management**

**B. QA**

**C. CM**

**D. Organization**

**E. BPR/technology plan impact**

**F. Testing**

**G. Life-cycle approach**

**H. Integration with new development**

## XIII. References and resources

**A. Literature (Organizational publications, classic references, textbooks)**

**B. Organizations (STSC, SEI, SPC, IEEE)**

**C. Conferences (STC, IEEE, REF, SEI)**

**D. Electronic resources**

1. Electronic conferencing (Internet)
2. Bulletin boards
3. Newsgroup
4. FTP sites
5. WWW (Mosaic)

**E. Courses and workshops**

1. University offerings
2. Government offerings
3. Commercial offerings
4. Vendor offerings
5. R&D organization offerings

**F. Tool collection/evaluation resources**

1. Software Technology Support Center (STSC)
2. Ada Information Clearinghouse (AIC)

### 6.10.4 Future Plans

The final activity of the Tool and Technologies working group was to identify working group chairs and to define a schedule for those interested in continuing work on a guide to best practice. Len Green was chosen to be the Software Engineering Institute co-chair of the group, with Carmen Castells-Schofield acting as the industry co-chair. An initial schedule for the working group was defined as follows:

May

- A combined outline of all of the book chapters will be distributed to working group members.
- The post-conference working group composition will be identified.
- References and resource lists information will be forwarded to co-chairs.

June

- Working group mechanics will be defined.
- The combined outline will be revised to remove overlap and ensure the covering of issues.
- The combined reference and resource list to working groups.
- Delegation of writing assignments.
- Generation of an expanded outline.
- Conference call to finalize outline, deal with other pressing issues.

July

- More writing assignments.

August

- Draft of chapter (expanded outline format).

September

- Review of chapter (expanded outline format).
- Schedule Victoria meeting.
- Review of other chapters.

# 7 Acknowledgments

I would like to express my sincere gratitude for the excellent job done by the individuals who provided me with their notes from the workshop. This includes, but it is not limited to, the following individuals: Robert Arnold, John Bergey, Daniel Burton, David Carney, Alan Christie, Robert Krut, Edwin Morris, Robert Park, Anne Quinn, James Webb, and Michael Webb. Without their valiant efforts, this document would not be a comprehensive representation of the information discussed at the workshop. In addition, I would like to thank the authors of the submitted papers and workshop speakers for their contributions to this document. Furthermore, I would like to thank Lorraine Nemeth and Walter Lamia for their assistance in the development of this document. And finally, I would like to thank all the reviewers of this document for providing constructive feedback.

# Appendix List of Workshop Attendees

Julia H. Allen
Deputy Director
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-6942 F:(412) 268-5758
jha@sei.cmu.edu

Archie D. Andrews
Manager
Dept. of Defense Sector
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA  15213-3890
P:(412) 268-7387 F:(412) 268-5758
ada@sei.cmu.edu

Robert Arnold
Software Evolution Technology, Inc.
12613 Rock Ridge Road
Herndon VA  22070
P:(703) 450-6791 F:(703) 450-6791
r.arnold@compmail.com

William R. Ball, Jr.
Director
FileNet Corporation
3565 Harbor Boulevard
Costa Mesa CA 92626
P:(714) 966-3436 F:(714) 966-3290
ball@filenet.com

John K. Bergey
Visiting Scientist
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
jkb@sei.cmu.edu

Joseph Blackburn
James A. Speyca Prof. of Mgmt.
Vanderbilt University
401 21st Avenue South
Nashville TN 37203
P:(615) 322-2995 F:(615) 343-7177
blackbj@ctrvax.vanderbilt.edu

Phillip W. Bonesteele
Manager, FileNet Corporation
3565 Harbor Boulevard
Costa Mesa CA 92626-1420
P:(714) 966-3608 F:(714) 966-3290
pwb@filenet.com

David J. Breuker
Manager
Martin Marietta Astronautics
2025 Research Parkway
Colorado Springs CO 80920
P:(719) 528-3853 F:(719) 528-3822

Lisa Brownsword
Manager
OSD Programs
Software Engineering Institute
801 N. Randolph St., Ste. 405
Arlington VA 22203
P:(703) 908-8203 F:(703) 908-9317
llb@sei.cmu.edu

C. Daniel Burton
Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-6833 F:(412) 268-5758
dburton@sei.cmu.edu

Grady Campbell
Software Productivity Consortium
SPC Building, 2214 Rock Hill Road
Herndon VA 22070
P:(703) 742-7104 F:(703) 742-7200
campbell@software.org

William Carlson
Vice President & Chief Technology Officer
Intermetrics, Inc.
733 Concord Ave.
Cambridge MA 02138-1002
P:(617) 661-1840
carlson@inmet.com

David J. Carney
Member of Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-6525 F:(412) 268-5758
djc@sei.cmu.edu

Debra M. Carr
Advanced Systems Engineer
Electronic Data Systems
5400 Legacy Drive , H3-4D-27
Plano TX 75024
P:(214) 605-4112 F:(214) 605-4071

Carmen L. Castells-Schofield
President
Leverage Technologists
P.O. Box 4638
Rockville MD 20849-4638
P:(301) 309-8783
carmencs@levtech.com

Elliot Chikofsky
Principal
DMR Group, Inc.
404 Wyman Street, Suite 450
Waltham MA 02154
P:(617) 487-9026 F:(617) 487-5752
e.chikofsky@computer.org

Alan M. Christie
Member of the Techical Staff
CASE Environments
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-6324 F:(412) 268-5758
amc@sei.cmu.edu

John O. Clark
Program Manager
Comptek Federal Systems, Inc.
2877 Guardian Lane
Virginia Beach VA 23452
P:(804) 463-8500 F:(804) 463-5675
clark@comptek.mhs.compuserve.com

Connie L. Clayton
Sr. Engineer
Science Applications Intl. Corp.
500 N.W. Plaza, Ste. 719
St. Ann MO 63074
P:(314) 298-1665 F:(314) 298-1668

Paul C. Clements
Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
P:(412) 268-8243 F:(412) 268-5758
clements@sei.cmu.edu

Sholom Cohen
Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-5872 F:(412) 268-5758
sgc@sei.cmu.edu

Edward J. Coyne
Principal Scientist
SETA Corporation
6858 Old Dominion Drive
McLean VA 22101-3832
P:(703) 821-5678 F:(703) 821-8274
ecoyne@seta.com

Harry E. Crisp
ECS Program Manager
Naval Surface Warfare Center
Code B05
17320 Dahlgren Road
Dahlgren VA 22448-5100
P:(703) 663-8901 F:(703) 663-8223
hcrisp@relay.nswc.navy.mil

Susan N. Crosswait
U. S. Marine Corps
MCTSSA, OPS
Box 555171
Camp Pendleton CA 92055
P:(619) 725-9543 F:(619) 725-9515
wwhq03=cpptssa9@mqg1.usmc.mil

Donn J. DiNunno
Computer Scientist
Computer Sciences Corporation
3160 Fairview Park Drive
Falls Church VA 22042
(703) 876-1558 F:(703) 573-1286
ddinunno@csc.com

John B. Donald
Software Cost Analyst
U. S. Air Force Cost Analysis Agency
1111 Jefferson Davis Hwy., #403
Arlington VA 22202
P:(703) 604-0409 F:(703) 604-6646
donald@afcost.af.mil

Larry Druffel
Director
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7740 F:(412) 268-5758
ld@sei.cmu.edu

Jean-Marie Favre
University of Grenoble
Grenoble, FRANCE
P:+33-7-6514964 F:+33-7-6446575
jmfavre@imag.fr

Francis B. Goldbach
Information Systems Consultant
Blue Cross of Western Pennsylvania
Fifth Ave. Place
Pittsburgh PA 15222
P:(412) 255-8402 F:(412) 255-8550

Barbara R. Gratz
Program Assistant
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-2310 F:(412) 268-5758
brg@sei.cmu.edu

Leonard S. Green
Resident Affiliate
National Security Agency
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-6310 F:(412) 268-5857
lsg@sei.cmu.edu

Judy M. Hardin
Senior Programmer
Loral Space Information Systems
1816 Spark Park Drive
Houston TX 77258
P:(713) 335-2642 F:(713) 335-2630

Gibbie Lu Hart
Member of Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7780 F:(412) 268-5758
gibbie@sei.cmu.edu

William M. Hasling
Siemens Corporate Research, Inc.
755 College Road East
Princeton NJ 08540
P:(609) 734-6523 F:(609) 734-6565
bhasling@scr.siemens.com

Fred Hathorn
U. S. Department of Defense
7732 Jewelweed Court
Springfield VA 22152
P:(703) 569-1189

Lewis B. Hayes
Senior Associate
Booz, Allen & Hamilton Inc.
8283 Greensboro Drive
McLean VA 22102
P:(703) 902-4843 F:(703) 902-3025
hayes_butch@bah.com

William E. Hefley
Sr. Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7793 F:(412) 268-5758
weh@sei.cmu.edu

Reginald L. Hobbs
Computer Scientist
U. S. Army Research Laboratory
115 O'Keefe Building
Georgia Tech
Atlanta GA 30332-0800
P:(404) 894-1807 F:(404) 894-3142
hobbs@airmics.gatech.edu

Samuel E. Hoke
Vice President
Nations Banc Services
3 Commercial Place
Norfolk VA 23452
P:(804) 441-4527 F:(804) 624-2876

David W. Hsiao
Director, Process Leadership
General Research Corporation
1900 Gallows Road
Vienna VA 22182-3865
P:(703) 506-5528 F:(703) 506-9241
dhsiao@grci.com

Paul K. Huntwork
Digital Equipment Corporation
110 Spit Brook Road
Nashua NH 03062
P:(603) 881-0131 F:(603) 881-0120
huntwork@setc.enet.dec.com

Farnam Jahanian
University of Michigan
Dept. of EECS
Ann Arbor MI 48109
P:(313) 936-2974 F:(313) 747-9482
farnam@eecs.umich.edu

Robert E. Johnson
Director, Army SW Reuse Mgmt.
ODISC4 (ASRMO)
The Pentagon
Room 1C670
Washington DC 20310
P:(703) 285-6951 F:(703) 697-2177
robert.johnson@pentagon-
1DMS2.army.mil

Karen Kalil Brown
Lead Engineer
Raytheon Company
Missile Systems Division
50 Apple Hill Drive
Tewksbury MA 01876
P:(508) 858-5956 F:(508) 858-9380
kmk@swl.msd.ray.com

Benjamin J. Keller
Virginia Polytechnic Institute & State U
320 Femoyer Hall
VPI & SU
Blacksburgh VA 24061-0251
P:(703) 231-6144 F:(703) 231-4330
keller@csgrad.cs.vt.edu

Marc I. Kellner
Senior Scientist
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7721 F:(412) 268-5758
mik@sei.cmu.edu

Mark Klein
Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7615 F:(412) 268-5758
mk@sei.cmu.edu

Alan C. Kocka
Principle Engineer, Software
Grumman Aerospace & Electronics
MS B38-35
Bethpage NY 11714
P:(516) 575-9628 F:(516) 575-7528

Ravindra Koka
President
Software Engineering & Enhancement Ctr.
5001 Baum Blvd.
Pittsburgh PA 15213
P:(412) 682-4991 F:(412) 682-4958
rkoka@a.gp.cs.cmu.edu

Mark A. Kolz
Project Manager
Decision Systems Technologies, Inc.
6301 Ivy Lane, Ste. 600
Greenbelt MD 20770
P:(301) 441-3377 F:(301) 441-4571

Frank Konieczny
Vice President & Chief Scientist
General Research Corporation
1900 Gallows Road
Vienna VA 22182
P:(703) 506-5137 F:(703) 356-3027

Robert Krut
Member of Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-8505 F:(412) 268-5758
rk@sei.cmu.edu

Arun Lakhotia
Assistant Professor
University of Southwestern Louisiana
2 Rex Street, P.O. Box 44330
Lafayette LA 70504
P:(318) 231-6766 F:(318) 231-5791
arun@cacs.usl.edu

Walter M. Lamia
Member of Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-3443 F:(412) 268-5758
wml@sei.cmu.edu

John R. Leary
Member Technical Staff
Software Engineering Institute
801 North Randolph St., Ste. 405
Arlington VA 22203
P:(703) 908-8206 F:(703) 908-9317
jrl@sei.cmu.edu

Karl Lebsanft
System Analyst
Siemens
Otto-Hahn-Ring 6
81739  Munich GERMANY
P:+(49) 89-636-3365
F:+(49) 89-636-44424
karl.lebsanft@zfe.siemens.de

Lisa Levine
Associate
Booz, Allen & Hamilton Inc.
8283 Greensboro Drive
McLean VA 22102
P:(703) 902-5592 F:(703) 902-3025

LichotaRandall
Sr. Systems Engineer
Hughes Technical Services
Bldg. 1704, Room 107
Hanscom AFB MA 01731-2116
P:(617) 377-2520 F:(617) 377-5123
lichotar@gw1.hanscom.af.mil

Daniel M. Litynski
Professor and Head
U. S. Military Academy
D/EECS, USMA
West Point NY 10996-1787
P:(914) 938-2200 F:(914) 938-5956
dd1408@eecs1.eecs.usma.edu

Evan Lock
President & CEO
Computer Command and Control Company
2300 Chestnut Street, Suite 230
Philadelphia PA 19103
P:(215) 854-0555 F:(215) 854-0665
lock@cccc.com

Joseph P. Loyall
The Analytic Sciences Corporation
55 Walkers Brook Drive
Reading MA 01867
(617) 942-2000
jployall@tasc.com

Peter Malpass
LTQ Manager
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-5779 F:(412) 268-5758
pmalpass@sei.cmu.edu

Lawrence M. Markosian
V. P. for Applications Devel.
Reasoning Systems
3260 Hillview Avenue
Palo Alto CA 94304
P:(415) 494-6201 F:(415) 494-8053
zaven@reasoning.com

Julia McCreary
Reengineering Tech. Mgr.
Internal Revenue Service
1525 Wilson Boulevard, Suite 400
Arlington VA 22209
P:(703) 235-2755 F:(703) 235-3021

Robert L. Moore
Software Engineer
Vitro Corporation
45 West Gude Drive
Rockville MD 20850
P:(301) 231-2038 F:(410) 712-7212
moorer@vitro.com

Tamra Moore
Computer Scientist
Defense Information Systems Agency
JIEO/CIM/TXER-Fairview Park
701 South Courthouse Road
Arlington VA 22204-2199
(703) 285-6589
mooret@cc.ims.disa.mil

Edwin Morris
Member of the Technical Staff, CASE Environments
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-5754 F:(412) 268-5758
ejm@sei.cmu.edu

Kenneth D. Mullins
Chief, Quality Assurance
USSTRATCOM
901 SAC Boulevard
Offutt AFB NE 68113-5280
P:(402) 294-2068 F:(402) 294-6190

Alice H. Muntz
Principal Scientist
Hughes Information Technology Company
MS SC/S64/C410
2000 East El Segundo Boulevard
El Segundo CA 90245
P:(310) 364-6210 F:(310) 364-6699
ahmuntz@hac2arpa.hac.com

Lorraine J. Nemeth
Program Specialist
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7777 F:(412) 268-7779
ljn@sei.cmu.edu

Michael R. Olsem
Principal Investigator
Science Applications Intl. Corp.
00-ALC/TISEC
7278 4th Street
Hill AFB UT 84056-5205
P:(801) 825-2655 F:(801) 777-8069
olsemm@hillwpos.hill.af.mil

Lorrie Orndoff
Event Coordinator
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-3490 F:(412) 268-7401
lao@sei.cmu.edu

Robert E. Park
Sr. Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-5785 F:(412) 268-5758
rep@sei.cmu.edu

A. Spencer Peterson
Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7608 F:(412) 268-5758
asp@sei.cmu.edu

Willie B. Pickens
Software Engineer
Loral Federal Systems
9500 Godwin Drive
Manassas VA 22110
P:(703) 367-6937 F:(703) 367-5597

Frances K. Pobletts
Senior Computer Scientist
National Security Agency
9800 Savage Road
Ft. Meade MD 20755
P:(301) 688-9440 F:(301) 688-9436

Noah S. Prywes
President
Computer Command and Control Company
2300 Chestnut Street, Ste. 230
Philadelphia PA 19103
P:(215) 854-0555 F:(215) 854-0665
nsp@central.eis.upenn.edu

Anne M. Quinn
Defense Logistics Agency
P.O. Box 16895
Philadelphia PA 19142-0895
(610) 591-8571
aquinn@dcmdma2.dcmdm.dla.mil

Glenn E. Racine
U. S. Army Research Laboratory
115 O'Keefe Building
Georgia Institute of Technology
Atlanta GA 30332-0800
P:(404) 894-3110 F:(404) 894-3142
racine%airmics@gatech.edu

L. Scott Reed
Manager, Government Operations
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-8468 F:(412) 268-5758
lsr@sei.cmu.edu

Howard B. Reubenstein
Member of Technical Staff
The MITRE Corporation
202 Burlington Road
Bedford MA 01730
P:(617) 271-2658 F:(617) 271-2352
hbr@mitre.org

Michael L. Rice
Computer Engineer
Naval Sea Systems Command
2531 Jefferson Davis Hwy.
Arlington VA 22242-5160
P:(703) 602-7191 F:(703) 602-8123
rice_mike_l@hq.navsea

Jon Spencer Rugaber
Senior Research Scientist
Georgia Institute of Technology
801 Atlantic Drive
Atlanta GA 30332-0280
P:(404) 894-8450 F:(404) 894-9442
spencer@cc.gatech.edu

Dale W. Sampson
ADP Audit Manager
U. S. Air Force Audit Agency
4170 Hebble Creek Rd., Suite 1
Wright-Patterson AFB OH 45433-5644
P:(513) 257-5381 F:(513) 257-2769

William P. Selfridge
Director, Info. Sys.
SETA Corporation
6862 Elm Street
McLean VA 22101
P:(703) 821-8178
bselfrid@seta.com

S. Wayne Sherer
STARS Technology Center
801 N. Randolph St., Ste. 400
Arlington VA 22203
P:(703) 351-5316 F:(703) 528-2627
sherer@stars.ballston.paramax.com

Chris Sittenauer
U. S. Air Force
7278 4th Street
Hill AFB UT 84056-5205
P:(801) 777-9730 F:(801) 777-8069
sittenau@oodis01.hill.af.mil

Dennis Smith
MTS/PRoject Leader
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-6850 F:(412) 268-5758
dbs@sei.cmu.edu

Joan G. Smith
Computer Specialist
Defense Information Systems Agency
JIEO/CIM/TXER-Fairview Park
701 South Courthouse Road
Arlington VA 22204-2199
P:(703) 285-6589 F:(703) 285-6579
smith9j@cc.ims.disa.mil

David G. Struble
Manager, Systems Software
Texas Instruments
4556 Early Morn Drive
Plano TX 75093
P:(214) 575-5346 F:(214) 575-6200
struble@vax1.dseg.ti.com

Charles W. Stump II
Sr. Software Engineer
Leverage Technologists
P.O. Box 4638
Rockville MD 20849-4638
P:(301) 309-8783
cstump@levtech.com

Daniel J. Sullivan
James Martin & Co.
15 Hampshire Drive
Derry NH 03038
P:(603) 898-8940

Frank J. Svoboda
Unisys Government Systems Group
12010 Suntise Valley Drive
Reston VA 22091
P:(703) 620-7916
svoboda@stars.reston.paramax.com

Carl G. Tegfeldt
PRC, Inc.
1283 Stuart Road
Arlington VA 22202
P:(703) 414-0900

Scott R. Tilley
University of Victoria
P.O. Box 3055
Victoria BC V8W 3P6 CANADA
P:(604) 721-7294 F:(604) 721-7292
stilley@csr.uvic.ca

James E. Tomayko
Project Leader
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-6806 F:(412) 268-5758
jet@sei.cmu.edu

Joel Trimble
Member of the Technical Staff/STARS
Software Engineering Institute
801 North Randolph St., Ste. 400
Arlington VA 22203
P:(703) 351-5310 F:(703) 528-2627
jt@sei.cmu.edu

William M. Ulrich
President
Tactical Strategy Group, Inc.
1716 Cheryl Way
Aptos CA 95003
P:(408) 662-3165 F:(408) 662-9083
71732.3400@compuserve.com

Robert L. Vesprini
Principal Engineer
Raytheon Company
ESC/ENS Bldg. 1704
Hanscom AFB MA 01730
P:(617) 377-7828 F:(617) 377-5123
vesprinir@gw1.hanscom.af.mil

Douglas W. Waugh
Member Technical Staff
Software Engineering Institute
801 North Randolph St., Ste. 405
Arlington VA 22203
P:(703) 908-8207 F:(703) 908-9317
dww@sei.cmu.edu

Larry L. Wear
Program Manager
Tandem Computers, Inc.
10600 Ridgeview Court
Loc. 229-31
Cupertino CA 95014
P:(408) 285-0443 F:(408) 285-0020
wear_larry_l@tandem.com

J. Todd Webb
Software Engineering Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-8738 F:(412) 268-5758
jtw@sei.cmu.edu

S. Michael Webb
Resident Affiliate
Texas Instruments
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
P:(412) 268-7157 F:(412) 268-5758
smw@sei.cmu.edu

Robert A. Webster
Director of Technology Programs
U. S. Air Force
ESC/ENS
Bldg. 1704
Hanscom AFB MA 01741
P:(617) 377-4915 F:(617) 377-8325
webster@gw2.hanscom.af.mil

Ronald E. Weller
Systems Engineer
Science Applications Intl. Corp.
1213 Jeff Davis Hwy., Ste. 1500
Arlington VA 22202
P:(703) 414-3922 F:(703) 414-8267
rweller@cg4.saic.com

Karen R. White
Project Leader
The Analytic Sciences Corporation
55 Walkers Brook Drive
Reading MA 01867
P:(617) 942-2000 F:(617) 942-7100
krwhite@tasc.com

Stephanie M. White
Principal Engineer, Software
Grumman Corporation
MS B38-35
Bethpage NY 11714
P:(516) 575-2201 F:(516) 575-7528
steph@gdstech.grumman.com

Mark L. Wilson
Computer Engineer
Naval Surface Warfare Center
10901 New Hampshire Avenue
NSWCDD/WO, Code B40, Rm. 4-133
Silver Spring MD 20903-5640
P:(301) 394-5099 F:(301) 394-3179
mlwilso@relay.nswc.navy.mil

David A. Workman
Professor of Computer Science
Science Applications Intl. Corp.
3045 Technology Parkway
Orlando FL 32826-3299
P:(407) 282-6700 F:(407) 282-6260
workmand@orlva.saic.com

Bernard J. Zaranski
Managing Consultant
James Martin & Co.
2100 Reston Pkwy., Ste. 300
Reston VA 22091
P:(703) 715-4285 F:(703) 476-1335

Angelika Zobel
Siemens
5615 Darlington Road
Pittsburgh PA 15217
P:(412) 268-1449 F:(412) 521-6191
zobel@cs.cmu.edu

UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b. RESTRICTIVE MARKINGS<br>None |
| 2a. SECURITY CLASSIFICATION AUTHORITY<br>N/A | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for Public Release<br>Distribution Unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>CMU/SEI-94-SR-008 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |

| | | |
|---|---|---|
| 6a. NAME OF PERFORMING ORGANIZATION<br>Software Engineering Institute | 6b. OFFICE SYMBOL (if applicable)<br>SEI | 7a. NAME OF MONITORING ORGANIZATION<br>SEI Joint Program Office |
| 6c. ADDRESS (city, state, and zip code)<br>Carnegie Mellon University<br>Pittsburgh PA 15213 | | 7b. ADDRESS (city, state, and zip code)<br>HQ ESC/ENS<br>5 Eglin Street<br>Hanscom AFB, MA 01731-2116 |
| 8a. NAME OFFUNDING/SPONSORING ORGANIZATION<br>SEI Joint Program Office | 8b. OFFICE SYMBOL (if applicable)<br>ESC/ENS | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>F19628-90-C-0003 |
| 8c. ADDRESS (city, state, and zip code))<br>Carnegie Mellon University<br>Pittsburgh PA 15213 | | 10. SOURCE OF FUNDING NOS. |

10. SOURCE OF FUNDING NOS.

| PROGRAM ELEMENT NO | PROJECT NO. | TASK NO | WORK UNIT NO. |
|---|---|---|---|
| 63756E | N/A | N/A | N/A |

11. TITLE (Include Security Classification)
Proceedings of the First Annual Software Engineering Techniques Workshop, May 1994: Software Reengineering

12. PERSONAL AUTHOR(S)
Leonard Green, John Bergey, Walter Lamia, and Dennis Smith

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (year, month, day) | 15. PAGE COUNT |
|---|---|---|---|
| Final | FROM TO | November 1994 | 121 pp. |

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

| FIELD | GROUP | SUB. GR. |
|---|---|---|
| | | |
| | | |
| | | |

18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
software engineering
software reengineering techniques
reengineering decision making
reengineering process

19. ABSTRACT (continue on reverse if necessary and identify by block number)

This is a report of the proceedings of the May 1994 Software Engineering Institute’s Software Engineering Techniques Workshop on Software Reengineering. The report includes brief biographies of the workshop speakers and authors, and detailed accounts of nine working group session discussions on the following topics: reengineering architecture, decision, analysis, system evolution and design records, reengineering economics analysis, understanding legacy systems, using lessons learned from other software reengineering projects, reengineering process models, software reuse, and reengineering technology and tools. The workshop established a foundation for capturing the best practices within reengineering and resulted in a detailed outline for a reengineering best practices handbook.

(please turn over)

| | | |
|---|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■ | 21. ABSTRACT SECURITY CLASSIFICATION<br>Unclassified, Unlimited Distribution | |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Thomas R. Miller, Lt Col, USAF | 22b. TELEPHONE NUMBER (include area code)<br>(412) 268-7631 | 22c. OFFICE SYMBOL<br>ESC/ENS (SEI) |

ABSTRACT — continued from page one, block 19