AD-A228 771

# THE
# TECHNICAL
# COOPERATION
# PROGRAM

TTCP

## SUBCOMMITTEE ON NON-ATOMIC MILITARY RESEARCH AND DEVELOPMENT

# REQUIREMENTS ENGINEERING
# AND RAPID PROTOTYPING WORKSHOP

# PROCEEDINGS

Sheraton Hotel and Conference Center
Eatontown, NJ

November 14-16, 1989

DTIC
ELECTE
OCT 17 1990
S
B
D

*Hosted by*

U.S. Army Communications-Electronics Command
Center for Software Engineering

CECOM
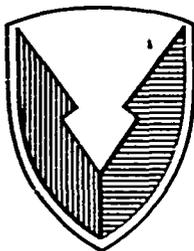
90 10 17 003

## Disclaimer

*The citation of trade names and names of manufacturers in this report is not to be construed as official Government endorsement or approval of commercial products or services referenced herein.*

This page is intentionally left blank.

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

| 1a. REPORT SECURITY CLASSIFICATION | 1b. RESTRICTIVE MARKINGS |
|---|---|
| UNCLASSIFIED | |

| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION / AVAILABILITY OF REPORT |
|---|---|
| | Approved for Public Release |
| 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE | Distribution Unlimited |

| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
|---|---|
| C-0103400000100 | |

| 6a. NAME OF PERFORMING ORGANIZATION | 6b. OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION |
|---|---|---|
| CECOM Center for Software Engineering | See 6c | |

| 6c. ADDRESS (City, State, and ZIP Code) | 7b. ADDRESS (City, State, and ZIP Code) |
|---|---|
| Commander, US Army Communications-Electronics Command, ATTN: AMSEL-RD-SE-AST-SE Fort Monmouth, NJ 07703 | |

| 8a. NAME OF FUNDING / SPONSORING ORGANIZATION | 8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
|---|---|---|
| SAME | | |

| 8c. ADDRESS (City, State, and ZIP Code) | 10. SOURCE OF FUNDING NUMBERS | | | |
|---|---|---|---|---|
| | PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO. |
| | | | | |

11. TITLE (Include Security Classification)

TTCP Requirements Engineering and Rapid Prototyping Workshop Proceedings

12. PERSONAL AUTHOR(S) Harlan Black (Editor), Alan Davis, Raymond Yeh, Winston Royce, George Sumrall, William Gilmore, Robert Poston

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Technical Report | FROM 14 Nov 89 TO 16 Nov 89 | 90-05-01 | 256 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Software Requirements, System Requirements, Software Methodology Requirements Engineering, Rapid Prototyping |
| 12 | 05 | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
On 14-16 November 1989, the US Army CECOM Center for Software Engineering hosted the TTCP Workshop on Requirements Engineering and Rapid Prototyping. This event was sponsored by the Technical Cooperation Program (TTCP). The workshop's forty-nine international participants met to share current information on the field, to identify and clarify the most pressing issues, and to provide recommendations to DoD for management, development, and research relating to Requirements Engineering. The workshop provided a forum for thirteen technical presentation. Participants divided into three working groups for small-group interaction. These preceedings document the presentations and findings of this workshop and its working groups.

| 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | UNCLASSIFIED |

| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE (Include Area Code) | 22c. OFFICE SYMBOL |
|---|---|---|
| HARLAN H. BLACK | | AMSEL-RD-SE-AST-SE |

DD Form 1473, JUN 86          *Previous editions are obsolete.*          SECURITY CLASSIFICATION OF THIS PAGE

This page is intentionally left blank.

# FOREWORD

The TTCP Technical Panel on Software Engineering (XTP-2) is grateful to the U.S. Army Communications-Electronics Command (CECOM), especially the Center for Software Engineering, for providing the resources and dedicated efforts which made this Workshop possible. It was quite evident from the excitement of the participants, the dynamics that occurred, and the smoothness by which the sessions proceeded that extensive planning and preparation went into the efforts of hosting the Workshop. In addition, the Panel extends its gratitude to the General Chairperson, the Workshop Coordinator, Working Group Chairpersons, and all the participants who worked long and late to make the outcome successful in every way. We are hopeful that the effort was as beneficial for all the participants as it was for the Panel Members.

Fulfillment of the Workshop objectives (to survey, evaluate and promote the use of requirements engineering and rapid prototyping for improving the quality of requirements for mission-critical defense systems) led to development of issues and recommendations, for the member TTCP Governments, in both management and technical areas. These are under review and in some areas appropriate actions are already underway.

Recognizing the importance and the potential of achieving major improvements in requirements engineering and rapid prototyping, the participants strongly suggested a follow-up workshop within the next few years. The TTCP Panel will closely monitor future developments in this area, and will fully consider this suggestion.
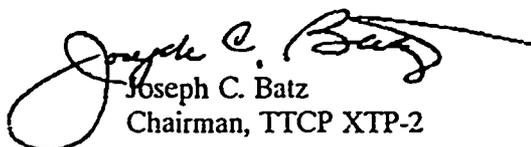
Joseph C. Batz
Chairman, TTCP XTP-2
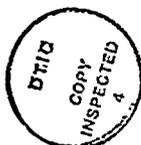
| Accession For | |
|---|---|
| NTIS GRA&I | ☑ |
| DTIC TAB | ☐ |
| Unannounced | ☐ |
| Justification | |
| By | |
| Distribution/ | |
| Availability Codes | |

| Dist | Avail and/or Special |
|---|---|
| A-1 | |

DTIC COPY INSPECTED 4

v

This page is intentionally left blank.

# TTCP Workshop on Requirements Engineering and Rapid Prototyping

November 14 -16, 1989

Sheraton Eatontown Hotel and Conference Center
Route 35 and Industrial Way East
Eatontown, NJ 07724

## Workshop Leaders

| | |
|---|---|
| Mr. George Sumrall | Workshop Chairperson |
| Mr. Harlan Black | Workshop Coordinator |
| Dr. Alan M. Davis | Working Group I Chairperson<br>"Requirements Development Process" |
| Dr. Raymond T. Yeh | Working Group II Chairperson<br>"Requirements Engineering Methodology, Languages, and Tools" |
| Dr. Winston W. Royce | Working Group III Chairperson<br>"Rapid Prototyping and Knowledge Based Techniques" |

## The Technical Cooperation Program
## Technical Panel Number 2 on Software Engineering (XTP-2)

Mr. Joseph Batz, United States National Leader and Chairperson

Mr. Jean-Claude Labbe, Canadian National Leader

Mr. Michael J. Looney, United Kingdom National Leader

Mr. Steven Landherr, Australian National Leader

Dr. Martin I. Wolfe, United States Army, CECOM
Mr. Larry Tubbs, United States Army, SDC
Mr. Phillip Andrews, United States Navy, SNWSC
Mr. Thomas P. Conrad, United States Navy, NUSC
Mr. Samuel DeNitto, United States Air Force, RADC
Mr. Charles Krueger, United States Air Force, AFWAL

This page is intentionally left blank.

# TABLE OF CONTENTS

# LIST OF APPENDICIES

# LIST OF FIGURES

# LIST OF TABLES

This page is intentionally left blank.

# 1

## EXECUTIVE SUMMARY

This page is intentionally left blank.

# 1 EXECUTIVE SUMMARY

## 1.1 Introduction

For both commercial and military computer-based systems, it is rare that the true needs of all stakeholders are fully stated and understood from the outset, nor are the requirements that are understood always agreed upon by all parties. In addition, requirements that have been documented are sometimes subject to interpretation by both users and developers. Even when requirements have been baselined, developers have difficulty in anticipating, controlling, and managing changes to the baseline.

These problems are a result of the lack of a well-defined Requirements Engineering (RE) discipline which, in turn, results in cost overruns, schedule slippages, poor quality, and systems that fail to satisfy mission needs.

The US Army CECOM Center for Software Engineering hosted the Requirements Engineering and Rapid Prototyping Workshop in Eatontown, NJ on November 14-16 1989. This event was sponsored by The Technical Cooperation Program's (TTCP) Panel on Software Engineering.

Many of the workshop's forty-nine participants are leading experts in Requirements and Software Engineering. They met to share current information on the field, to identify and clarify the most pressing issues, and to provide recommendations to Department of Defense (DoD) for management, development, and research relating to Requirements Engineering.

These Proceedings document the presentations and findings of the workshop and its three working groups.

## 1.2 The Requirements Engineering Process

Chairperson: Dr. Alan M. Davis

The group identified the following issues as having the highest priority: coping with requirements uncertainty and change; validating requirements; achieving consensus among multiple stakeholders; and measuring/tracking progress in requirements development.

The group members recommended the following for management: use an evolutionary acquisition approach; make personnel and stakeholders aware of acquisition alternatives and related technologies such as prototyping; involve all stakeholders in requirements determination and validation; orient acquisition and incentives around requirements "progress"; introduce risk-based requirements related decision making (multi-attribute utility, cost-benefit, Pareto optimization, etc.); and reduce barriers to developer-user interaction.

3

For development, they recommended that requirements be frozen in small incremental builds and that more testbeds be developed to validate interoperability earlier in the requirements process.

Finally, for research they recommended developing the following technologies and disciplines: requirements partitioning; change management; formal specification; multi-stakeholder process support; requirements normalization; process models; measurement techniques for requirements progress; tools and techniques to capture merits/trade-offs among requirements; and the selection of the appropriate acquisition and requirements technique for a given project.

## 1.3 Requirements Engineering Methodology, Tools, and Languages

Chairperson: Dr. Raymond T. Yeh

This group identified the following policy and management related issues: a lack of management awareness of the significance and importance of Requirements Engineering; and a lack of recognition that this discipline must be supported throughout a system's life cycle.

For development and research, they focussed on the following issues: the capture of requirements related information; non-functional requirements (the "ilities"); tool and technology integration; technology insertion for existing systems; and the measurement of key requirements process parameters.

The working group recommended the following for policy and management: adopt and support a requirements-centered development life cycle model; educate and train personnel in Requirements Engineering; establish a Requirements Engineering information/consultation center; and reallocate currently available research funds to support Requirements Engineering, spending less resources on downstream software activities (i.e., concentrate more resources on identifying and confirming what is to be built, rather than on how to build it).

For development and research, they recommended developing the following: a wide spectrum language which supports acquisition, representation, and reuse of requirements information; methods to capture, integrate, and measure non-functional requirements; an integrated environment of Requirements Engineering tools; methods and tools which support reverse engineering of current system's requirements documentation; requirements validation techniques; new approaches for requirements trade-off analysis; and metrics which support modern Requirements Engineering practices.

## 1.4 Knowledge-Based Techniques and Rapid Prototyping

Chairperson: Dr. Winston W. Royce

This group analyzed two specific aspects of Requirements Engineering: knowledge-based techniques and rapid prototyping.

4

The group identified the following issues which relate to knowledge-based techniques: the use of Knowledge Based Approaches (KBA) and their application to real systems; the risks and benefits of using KBA's for Requirements Engineering; the nature of a KBA specific software development process model; and the identification of existing knowledge-based technology.

The following were the group's management and policy recommendations: adopt policy and models that allow for incremental, evolutionary development and which accommodate KBA; invest in knowledge base development early in the acquisition phase; and reuse knowledge bases in related projects, to amortize investments across many projects.

For KBA development, they recommended learning from past KBA experience and trying KBA in a large, real project.

Research recommendations were: experiment using KBA for verification and validation (V & V); research KBA knowledge acquisition and management, especially in light of existing methodologies and tools; and research knowledge base models with advanced degrees of expressiveness.

Rapid prototyping issues that were identified were: participants and products in the prototyping process; standards and current practices; and uses, properties, and examples of prototyping systems and tools.

Management, policy and development recommendations for rapid prototyping were as follows: train personnel in the prototyping approach; modify the development stages and time frames to be supportive of prototyping; define the objectives of requirements/design reviews which use prototyping products; support competitive prototyping efforts; and consider acquisition models that include prototyping.

Finally, recommendations for research programs were proposed for the following: requirements traceability; validation of non-functional requirements; automatic prototype-to-documentation generation; stakeholder communication; legal issues; and lessons learned from prior prototyping efforts.


## 1.5    Recommendations and Conclusions

The workshop produced many valuable insights and recommendations. These insights and recommendations are fully documented in these Proceedings. It is important to note that although the three groups worked independently, a number of recommendations were common to the three groups. Every group saw the need for the DoD to change policy to accommodate evolutionary acquisition. The groups also saw the need for increased training for Government acquisition personnel to make them more aware of Requirements Engineering issues and techniques. Every group saw the need for additional emphasis and research in requirements validation. Most of the participants recognized the need for additional research in defining and using methods of measuring attributes and progress in the

Requirements Engineering process. Most identified the need for further work in specifying non-functional requirements. It was recommended that tools and techniques be developed which aid in identifying merits and trade-offs among requirements. Additional research in requirements traceability was also suggested. It was also recommended that continued special emphasis be given to multiple stakeholder issues as the Requirements Engineering process evolves. Finally, and most obviously, it was concluded that it is not enough to merely develop technologies. We must apply them as well.

# 2

## WORKSHOP CHARGE

This page is intentionally left blank.

# 2 WORKSHOP CHARGE

By: George E. Sumrall, Workshop Chairperson

Computer technology as we know it today is barely forty years old. We have made tremendous strides, in both hardware and software. Back in the early days, computers were the size of a wall and often filled a room. Now, you can hold one in your hand. With products like dBase or Lotus, you can store, manage, and exploit a wealth of data on a common home computer.

With the great strides that the commercial world has made in these technologies, the public, ourselves included, has great expectations for our software-intensive defense systems. There have been some successes; and there have been some problems. Many of these problems are identified in a report by the House Subcommittee on Investigations and Oversight of the House Committee on Science, Space, and Technology, entitled "Bugs in the Program", September, 1989. All too often, software is delivered late, and/or with cost overrun, and/or does not work the way it is supposed to, and/or doesn't do what the user wanted. According to the report, we end up paying twice for the software - once to develop it and again to make it work the way it was supposed to.

On the surface, it looks like those who are developing software for Mission Critical Defense Systems (MCDSs) are falling short, compared to those who develop commercial software products. But, there is a big difference:

- Software for Defense Systems is usually developed to meet "a user's needs", which are stated in the requirements specification,

- whereas, the primary requirements of a commercial product are usually that it offer a general capability, and that it be marketable. The concept of developing software to "meet the requirement" usually does not exist.

The Department of Defense is probably our country's largest buyer/developer of customer software. Our software is evaluated on the battlefield, not the marketplace. Our requirements are completely driven by the user. In manner that is timely for a given program we must capture and translate our customer's needs into a system that helps him do his job better, faster, safer.

Many times, our users do not know how to express what it is that they want or they are not able to know what they really need, during the time that we allocate for recording their requirements. It is not their fault. A typical user's job is to do his job, not to describe it, and not to describe it in a language that is understandable to a software developer. Because of the complexity and newness of the systems that we deal with, the user may be overwhelmed. After acquisition commitments, he often comes back with latent insights on how the proposed automated system can better help him. These new requirements are sometimes derived from subsequent experience with home computer technology. Sometimes, new requirements are

driven by changing battlefield realities. Let's stop blaming the user for changing requirements and find a way of developing systems and software despite an incomplete and changing set of requirements.

In reality, not a lot of attention has been given to the requirements problem. (I believe that the last workshop of this nature took place in Columbia, Maryland, in 1982.).

That is why we are here. In this room, we have a group of people who recognize that there is a problem, who have thought about it, and have even done something about it.

My hope for us is to bring our individual efforts into focus and try to chart our course for the future.

If you have any solutions now, let us know. If we are marching in the wrong direction, let us know. Let us know where we should concentrate our efforts over the next 2-3 years. That is our job over the next 2 days.

# 3

## *WORKSHOP PROCEEDINGS*

This page is intentionally left blank.

# 3   WORKSHOP PROCEEDINGS

## 3.1   Introduction

By the year 2000, it is projected that the total United States (US) software production costs, which have been growing exponentially, will reach $400 billion. By that time, the Department of Defense's (DoD's) annual investment will be $63 billion.

Software procurement, development, and maintenance are critical. Software is frequently cited as the reason for many systems being late, over budget, and not fully functional.

As much as fifty-five (55) percent of system errors are introduced during the requirements definition phase. This is when the needs of those who will ultimately be affected by the system are captured and re-written in a condensed form for solicitation and then later translated into a form that is best understood by those who develop the software.

Research has demonstrated that the cost of solving requirements-related problems increases drastically with the time it takes to detect an error. In a typical sample project, the estimated cost to fix a software problem (in the requirements phase) increased from a factor of two (2) to a factor of two-hundred (200), when a requirements-related problem was not noticed until the system was completed and installed.

For commercial and military computer-based systems alike, experience has shown that, especially for large and complex system developments, it is rare that the true needs of all stakeholders are fully stated and understood from the outset. Furthermore, even the requirements that are understood are not always agreed upon by all parties. To complicate matters more, requirements that have been documented are sometimes subject to interpretation by both users and developers. In addition to these problems, once requirements have been baselined, there are difficulties associated with anticipating, controlling, and managing changes to the baseline.

The above is a result of the lack of a well-defined Requirements Engineering (RE) discipline which, in turn, results in cost overruns, schedule slippages, poor quality, and systems that fail to satisfy mission needs.

Requirements-related problems are industry wide, not unique to the military. Requirements must not be merely addressed. They must be engineered. Accurate and timely requirements formulation and management is a skill, yet to be perfected.

On November 14-16 1989, the US Army Communications Electronics Command (CECOM) Center for Software Engineering (CSE) hosted the Requirements Engineering and Rapid Prototyping Workshop in Eatontown, NJ. This event was sponsored by The Technical Cooperation Program's (TTCP's) XTP-2 Panel on Software Engineering.

The CECOM Center for Software Engineering is the Center of Excellence for software engineering support to designated Army Mission Critical Defense Systems (MCDSs). It provides software engineering and support for communication and electronics systems, from initial system concept through development, deployment, and field sustainment. The CECOM CSE is committed to worldwide US Army readiness.

The TTCP is a formal arrangement for mutual sharing of research and development resources/tasks established by member country foreign and defense ministries. Member countries include Australia, Canada, New Zealand, the United Kingdom, and the United States. Within the structure of the TTCP, there are eleven (11) subgroups made up of forty-four (44) working panels and twenty-two (22) action groups. The TTCP/XTP-2 Panel is concerned with the creation and life cycle support of software for defense-related applications.

Many of the workshop's forty-nine (49) international participants are leading experts in Requirements and Software Engineering. They met to share current information on the field, to identify and clarify the most pressing issues, and to provide recommendations to DoD for management, development, and research relating to Requirements Engineering.

The workshop provided a forum for thirteen (13) technical presentations by leaders in the field. The workshop participants divided into three (3) working groups for small-group interaction on central issues. One working group addressed the Requirements Engineering process and was chaired by Dr. Alan Davis. Another dealt with requirements engineering methodologies, languages, and tools, chaired by Dr. Raymond Yeh. The third, chaired by Dr. Winston Royce, focussed on two (2) specific aspects of Requirements Engineering, knowledge-based approaches and rapid prototyping.

The workshop was chaired by Mr. George Sumrall and was coordinated by Mr. Harlan Black, both from the CECOM Center for Software Engineering. Mr. Black is responsible for the Center's efforts in Requirements Engineering.

These Proceedings document the presentations and findings of this workshop and its working groups.

## 3.2 Working Group 1:

## Requirements Engineering Process

Edited by: Dr. Alan M. Davis, Working Group Chair

### 3.2.1 General Information

#### 3.2.1.1 Working Group Participants

| NAME | EMPLOYER | COUNTRY |
|---|---|---|
| Andriole, Stephen J. | George Mason University | USA |
| Batz, Joseph | DoD Software and Computer Technology | USA |
| Black, Harlan | CECOM Center for Software Engineering | USA |
| Charette, Robert N. | ITABHI Corporation | USA |
| Davis, Alan M. | George Mason University | USA |
| Deutsch, Michael | Carnegie-Mellon University SEI | USA |
| Fink, Robert C. | Performance Resources, Inc. | USA |
| Fountain, Harrison | Naval Postgraduate School | USA |
| Harris Jr., Donald C. | US Army Air Defense Artillery School | USA |
| Menell, Raymond | CECOM Center for Software Engineering | USA |
| Overmyer, Scott P. | Contel Technology Center | USA |
| Podracky, Mark A. | Digital Fantacies Limited | USA |
| Schlosser, Edward H. | Lockheed Software Technology Center | USA |
| Toher, James | SD-SCICON | England |
| White, Douglas A. | Rome Air Development Center | USA |

#### 3.2.1.2 Roadmap: A Guide to Working Group 1 Activities

This report on the activities of Working Group 1 is divided into four parts. The introduction identifies seven key issues concerning the requirements engineering process. This is followed by a section on four (4) of the most critical issues, containing for each issue an analysis, assumptions, impact, and recommendations. A conclusion summarizes the recommendations for management and training, development, and research. This is followed by a glossary of key terms.

## 3.2.1.3    Working Group Assignments

Three (3) subgroups were formed to address the foremost critical issues. Subgroup 1 addressed issue 1. Subgroup 2 addressed issues 2 and 4. Subgroup 3 addressed issue 3. Issues 5 through 7 were not further analyzed. The members of the working group and their subgroup assignments were the following distinguished individuals:

| Subgroup 1 | Subgroup 2 | Subgroup 3 |
|---|---|---|
| Batz, Joseph | Davis, Alan M. | Andriole, Stephen J. |
| Black, Harlan | Deutsch, Michael | Harris, Donald C. |
| **Charette, Robert N. *** | Fountain, Harrison | Menell, Raymond |
| Fink, Robert C. | **Overmyer, Scott P. *** | Podracky, Mark A. |
| White, Douglas A. | Toher, James | **Schlosser, Edward H. *** |

**\* Subgroup Chairperson**

## 3.2.2    Introduction

The first of the three working groups at the Workshop addressed issues relating to the *requirements engineering process*. A requirements engineering process defines:

•    Each of the individual steps to create and enhance requirements,

•    The partial ordering of those steps, and

•    The overall flow of information among those steps.

The entire process is independent of the methods and tools utilized in any of those steps.

Working Group 1 identified seven key issues about the requirements engineering process. In decreasing order of importance, they are:

1.    Uncertainty and change are difficult to cope with.

   The real user needs are rarely well understood prior to system deployment. They are certainly not well understood during the early development phases when we must "baseline" the requirements. The result is that our perception of the requirements constantly changes throughout the development process.

2.    Validation of requirements is critical to project success.

   The validation of a to-be-established baseline traditionally entails a detailed comparison of that to-be-established baseline with a previously established baseline. In practice, that previously established baseline is usually the requirements specification. Thus, for example, we verify the design documentation by comparing it with the requirements. Using this traditional definition of validation, we now

16

have a significant problem with respect to validating the requirements: To what do we compare the requirements? The current practice is to have a customer sign off on the requirements; this is contractually acceptable, but not sufficient in achieving true validation. The best available technique today might be the use of a prototype.

3. Multiple stakeholders make it difficult to reach closure.

   Many individuals with many diverse backgrounds have a stake in the success of a project. Most have opinions concerning what the requirements are. How can we accommodate all these diverse goals?

4. We do not know how to track progress in requirements development.

   We all know of the famous "99% syndrome" in software development (i.e., it takes 25% of the time to complete the first 99% of the work, and 75% of the time to complete the last 1%). How can we prevent this in the software requirements specification (SRS)? The industry norm today is that we simply declare the SRS complete when it looks like it's time to move on to design.

5. Different processes are needed for different problems.

   There does not exist a universal process model for requirements. Each class of problem requires a different model.

6. Systems/Software/Requirements/Design distinction is unclear.

   There is little uniformity in the industry concerning the use of the terms "system requirements," "software requirements," "system design," "software design," and "specifications." But it is more than a semantic problem. During each of the phases, developers regularly violate the bounds of their phase. This may or may not be detrimental, but it must be understood.

7. The existing inventory of systems needs to be retrofitted to new requirements engineering technology.

   There is a large active community of people studying and performing "reverse engineering" to the huge inventory of existing software systems. These people are primarily retrofitting code quality into systems built before good coding principles became well understood. As we learn more and more about proper requirements practices, does it make sense to retrofit existing systems with this quality?

### 3.2.3 Issues

The following four (4) subsections address the first four issues described above. Three (3) subgroups were formed to address them. Work on the last three was deferred, due to time constraints.

### 3.2.3.1 Uncertainty and Change are Difficult to Cope With

During the requirements engineering process, we are repeatedly faced with uncertainty. Are the requirements correct? Do they accurately reflect real needs? Can a system be built that satisfies these requirements? Is it possible to validate that a system meets these requirements? We are also constantly presented with changes. User needs change. Our perception of user needs changes. Designers discover unsatisfiable requirements. Both uncertainty and change introduce significant risk into the system development and acquisition process. One means of reducing the risks associated with uncertainty and change is evolutionary acquisition. In this approach, we acquire a system in increments. Each increment is an improved superset of the previous increment's requirements driven by changing needs. Determination of these additional needs can be accomplished through a variety of evolutionary requirements engineering approaches including rapid prototyping. Evolutionary requirements engineering runs counter to the defense system acquisition "culture". The current belief that all system requirements can be specified at one time is deeply embedded in DoD standards and acquisition policy.

Unfortunately, premature freezing of requirements specifications may lead to:

- An incomplete understanding of true system requirements (both functional and non-functional).

- An incomplete understanding of engineering and political tradeoffs.

- The addition of non-essential/unnecessary requirements.

- The inability to respond adequately to external changes which occur in the operational context.

The last item is of critical importance. DoD systems are expected to respond to a wide variety of changing circumstances, some within DoD's control, and most not. These circumstances create new system requirements unforeseen, indeed even unpredictable, at the outset of system acquisition. These requirements are driven by political circumstances (e.g., changes in the threat or in domestic funding), changes in military doctrine, increased user insight, and changing technology. The result is that:

- Systems are 3-5 generations behind currently available technology

- Systems cannot change quickly enough to meet new requirements dictated by new operational contexts.

- Many systems exhibit poor quality, are over budget, are late, and/or fail to support the required mission.

An evolutionary acquisition process will mitigate these problems considerably. The first phase of an evolutionary acquisition process defines the set of acceptable requirements which can be partitioned into an incremental build of the system. The acceptable set of requirements consists of all requirements which are perceived as being necessary (although

18

some requirements may be better understood than others). This acceptable set is called the evolutionary framework. Using Joint Application Development (JAD), rapid-prototyping, mock-ups, etc., a partitioned subset of well-understood requirements (i.e., generally the requirements with the minimal uncertainty) is constructed. Once this set of requirements are defined, the second phase of the evolutionary process occurs.

The requirements of an evolutionary framework are used to build an increment of the system. An appropriate process model is applied to further refine the requirements. Each increment is a superset of the previous increment. The evolutionary requirements activity continues through the life of the system, until the need for evolution diminishes to near zero. Along the way, rapid prototypes are used to validate prospective requirements prior to the next build. This helps to reduce uncertainty and change, and thus risk.

### 3.2.3.1.1  Sub-Issues

There are several management and technical sub-issues that affect the feasibility of evolutionary acquisition. The management sub-issues are as follows:

- Current acquisition regulations and system and software engineering standards such as MIL-STD-490A and DOD-STD-2167A, encourage the early binding of requirements.

- Who manages the evolutionary requirements activity? There needs to be significant cooperation here between contractor and Government personnel. Only the Government can adequately represent the needs of the user community. Only the contractor can understand the design implications of requirements evolution.

- The acquisition agency must be aware that the evolutionary requirements engineering activity is on-going, and as such, will require funding and deliverable schedules which are subject to change. Government personnel may perceive this approach as open-ended and counter to effective cost control, schedule control, and other resource controls.

The technical sub-issues are as follows:

- How can we partition requirements into builds that make technical sense?

- The initial partition of requirements must be "correct enough" to serve as a proper foundation for later builds. It (and the initial few partitions) also must be of a sufficient breath and depth to gain support by the sponsoring activity. A partition which is "too small" for example, may not show "progress" in the eyes of the acquisition agency.

- We must use methodologies and tools which will support incremental acquisition. Methods such as defined within the U.S. Navy Research Laboratory's Software Cost Reduction Project is an example. This issue is related to the sub-issue concerning DOD-STD-2167A.

19

### 3.2.3.1.2  Assumptions

The following are the assumptions made:

- The evolutionary acquisition approach is assumed to be a more effective and lower risk approach than other current approaches, although no real proof is available to support this assumption.

- Partitions are subsets of the entire set of requirements.  Increments are the portions of the prototype that implement corresponding requirements partitions.  Partitions and their resultant increments must occur within a short time-frame to minimize changes to the next partition and increment.

- Initially, and at each subsequent stage, a stable set of requirements can be established and partitioned.

- All stakeholders will be involved in the partition of requirements into increments.

### 3.2.3.1.3  Impacts

If the evolutionary acquisition approach is implemented, we believe:

- Uncertainty concerning requirements will be reduced because uncertainty is addressed incrementally.

- Expectations will be more realistic.

- The final system will more closely meet expectations.

- Risk will be sharply reduced.

### 3.2.3.1.4  Recommendations

- Management and Training.

  - Make changes to acquisition policies, acquisition regulations, and DoD standards to facilitate evolutionary acquisition.

  - Educate contracting officers and their technical representatives on this evolutionary acquisition approach.  Emphasize that system requirements cannot be fully defined a priori, and that requirements engineering is continuous throughout the life of the system.

- Development

    - For each incremental build of a given software process or (in DoD terms) Computer Software Configuration Item (CSCI), the corresponding defined partition must remain frozen during the implementation of that build.

- Research

    - Research is required on techniques for defining acceptable partitions of requirements.

    - Research is required to determine if the evolutionary acquisition approach is more effective than others.

    - Research is required to determine how to define partitions in such a way that they can tolerate the inevitable changes that will occur.

### 3.2.3.1.5 Validation of Requirements is Critical to Project Success

The ability to determine whether documented requirements are an accurate reflection of actual requirements (i.e., the real user needs) is crucial to the success of any software development effort. Often requirements content is heuristic and judgmental. Many of the system issues addressed by requirements have no apparent right answers. In most cases, it is impossible to understand the real requirements without the presence of a working system in the users' hands. Since most acquisitions do not include up-front prototypes, most requirements are not validated in any way until after system deployment. An acquisition strategy involving prototyping provides an early system on which multiple stakeholders can base a decision concerning system suitability.

The validation process involves identifying the guarantors and developing validation statements. For any single system there can be many guarantors and validation statements of varying rigor and credence.

### 3.2.3.1.6 Sub-Issues

The goal of requirements validation is to reconcile documented requirements against a referent or set of referents. Realization of this goal substantially reduces the risk of later breakage of the software or hardware architectures caused by inaccurate or incomplete requirements. This goal is often complicated by the absence of a referent. The sub-issues are:

- What can be done to validate requirements when no referent exists?

- How can we validate the requirements against an existent referent?

### 3.2.3.1.7 Assumptions

The following are the assumptions made:

- Requirements validation is possible.

- The end user is the principal stakeholder. The relative importance of any stakeholders is contingent upon project constraints and the point at which the stakeholder enters the lifecycle process.

- In practice, systems are often, if not always, accepted without validated requirements.

- Validation is a dynamic process which, in concept, may never end.

### 3.2.3.1.8 Impacts

The impacts of requirements validation are:

- decreased likelihood of cost overruns

- elimination or reduction of rework and schedule slips

- lower risk of development (management, schedule, cost, etc.)

- more effective systems.

### 3.2.3.1.9 Recommendations

- Management and Training

    - Remove excessive DoD barriers to contractor contact with users.

    - Update acquisition policies to support evolutionary life cycles.

    - Increase awareness of prototyping methodologies.

- Development

    - Develop standardized models for interdisciplinary user/customer/contractor approach to requirements validation.

    - Construct widespread test beds (e.g., Army Interoperability Network -- AIN) and associated data bases in more applications areas.

- Research

  - Perform research into automating the synthesis of design from requirements.

  - Develop practical formal requirements methods.

### 3.2.3.2 Multiple Stakeholders Make it Difficult to Reach Closure

A software-intensive military system typically is employed by many users in a variety of situations and contexts. These users, situations, and contexts all provide different viewpoints for determining system requirements. Many other players also have important stakes in the success of the system: testers, developers, managers, acquisition personnel, configuration management personnel, quality assurance personnel, maintenance personnel, etc. The current DoD requirements process often fails to include some of these viewpoints. Conflicts among the different viewpoints and among the requirements based on them is often unrecognized or inadequately resolved. All of this leads to requirements that are incomplete, inconsistent, unrealistic, or misunderstood, resulting in poor quality systems delivered late and over budget.

#### 3.2.3.2.1 Sub-Issues

System stakeholders can be classified as those who:

a. Affect the system

b. Are affected by the system

c. Both affect and are affected by the system

Potential stakeholders include end-users, proponents, funders, program managers, builders, testers, and system maintainers. Viewpoints of military end-users are a function of their level or echelon, the unit mission or function, and their experience with automation/computerization. Proponents for military systems are charged with developing mission requirements, representing the end-user's viewpoint throughout the development process, and defining system requirements. Organizations which approve/control funding clearly are stakeholders in the system requirements. Program managers, their support staffs, and their contractors who build systems must interpret and modify requirements which are often vague, inconsistent, and incomplete. Organizations which maintain and extend the system have a significant stake in the system during most of its lifetime.

Three (3) sub-issues relate to the multiple stakeholders, the multiple system contexts, and the development life cycle phases:

- How can we resolve the disparate, possibly conflicting, needs and views of the <u>multiplicity of stakeholders</u>?

- How can we resolve the disparate needs resulting from classes of users who must operate with the system in <u>multiple contexts</u>? The users of a system typically emanate from multiple organizations. These organizations have different missions and different battlefield environments.

- How can we resolve the needs and views considering that they are <u>changing constantly over time</u>? They change constantly because the membership of the stakeholder group changes, the individual people themselves change as they learn and grow, and the requirements specification is used in different ways.

### 3.2.3.2.2 Assumptions

None Identified.

### 3.2.3.2.3 Impacts

Reconciliation of stakeholders viewpoints would result in:

- Significantly decreased risk of user dissatisfaction

- Less cost overruns and schedule slippages

- Increased productivity (stakeholder satisfaction per dollar)

- Increased trust among stakeholders

- Decreased risk of project cancellation

### 3.2.3.2.4 Recommendations

Reconciling divergent requirements perspectives of multiple stakeholders is a difficult problem. It will require the cooperative efforts of individuals representing all significant viewpoints. We have proposed three (3) approaches. They are ordered from the easiest to implement to the most difficult to implement. Their order also corresponds to the order from the least positive impact to the most positive impact.

- Develop and document a procedure to evaluate and rank the importance of requirements based on who the supportive stakeholder is.

- Expand the above procedure to evaluate and rank the importance of requirements based on the motivations and purposes expressed by the supportive stakeholder as well as on who the stakeholder is.

- Develop and document a procedure that can be used to capture the complete set of requirements, as follows:

  - Identify and define all significant viewpoints and stakeholders

  - Determine and define requirements for each viewpoint

  - Communicate viewpoints and requirements to all stakeholders

  - Jointly evaluate requirements

  - Negotiate a reasonable requirements envelope

  - Test the requirements envelope continually

  - Iterate through all activities until system retirement

This process must include all stakeholders and their requirements. Effective communication of the viewpoints and requirements depends upon a combination of good documentation and face-to-face refinement. Requirements should be evaluated jointly with respect to priority, volatility, consistency, feasibility. The concept of a "requirements envelope" is key. We believe that a single, completely consistent requirements set may be unattainable in many cases. It may also result in overly constrained requirements, leading towards a less adaptable system architecture. The goal is to achieve a consensus requirements envelope that reduces, but does not eliminate, variety and inconsistency. A good requirements envelope will focus the requirements sufficiently to satisfy current requirement perceptions without overly constraining them. The requirements envelope should include measures of priority and volatility. The process should test the requirements envelope continually, by testing, simulation, prototyping, and partial system deliveries.

Further specific recommendations are:

- Management and Training

  - Acknowledge the importance of multiple requirements perspectives. Management should require formal recognition of multiple stakeholders requirements perspectives, and expand the requirements analysis and prototyping phases to include these.

  - Enhance life cycle models to accommodate deeper requirements analysis and modeling of the interrelationships among requirements.

- Development

    - Develop a set of software tools to support "multiple stakeholder requirements perspectives" analysis. The tools should consist of user taxonomies of organizations, and methods for conducting requirements trade-off analysis.

    - Apply the new methods and tools developed above to real applications.

- Research

    - Develop models to capture multiple stakeholder requirements.

    - Develop and apply new methods for trade-off among competing and conflicting requirements. Risk-based decision techniques such as multi-attribute utility, classic cost-benefit, and Pareto optimization techniques, among others, can be used in this arena.

### 3.2.3.3 We Do Not Know How to Track Progress in Requirements Development

Progress metrics for the requirements process differ markedly from production oriented process metrics because there is no clear end point. Requirements engineering is a continuing process based on exploration and discovery, often creating unexpected iterations. Nonetheless, some subjective oriented indicators of progress are possible.

### 3.2.3.3.1 Sub-Issues

The following sub-issues bear on the problem:

- A technical feasibility indicator for implementing a requirements set is a desirable measure.

- A cost/schedule feasibility indicator for a requirements set is a desirable measure.

- The contractual/political environment does not accept that exploratory processes have a built-in level of backtracking and iteration.

- We are dealing with a judgmental, discovery driven process with no clear end-point.

- Progress is not necessarily monotonic. Time/schedule is, therefore, often a poor metric.

### 3.2.3.3.2 Assumptions

The following assumptions are made:

- Progress can be observed, but not necessarily measured in an objective fashion.

- An agreeable metric of progress is possible.

- Progress is not necessarily a linear or well-behaved function.

- Risk (as to technological feasibility and cost/schedule) can be assessed periodically and thereafter monitored.

### 3.2.3.3.3 Impacts

The impacts of measuring requirements progress are:

- An appropriate definition of progress that can substantially reduce risk

- Measurable progress observations that aid/feed the requirements development and validation process

- Well-thought out, accurate requirements

- Reduction of arbitrary and/or autocratic decisions concerning the completion of the requirements baseline

- Decriminalization of early problem recognition and correction.

### 3.2.3.3.4 Recommendations

- Management and Training

  - Current contracts often encourage the early freezing of requirements and discourage subsequent changes to those requirements. Award fee structures on contracts should be modified to encourage the creation and timeliness of requirements specifications.

  - Develop a team approach to help reduce unrealistic expectations on the part of the user/customer.

  - Educate program managers and team members that "changing your mind" as a result of new information is acceptable.

  - Train Government program managers in the use of acquisition models that employ prototyping.

27

- Development

    - Apply the new metrics developed above on actual projects.

    - Develop an explicit requirements validation plan for every project.

- Research

    - Develop and use effective metrics to measure requirements progress and completion.

    - Develop more rigorous risk assessment and risk management techniques.

### 3.2.4 Conclusion

In this section, we summarize the recommendations of Working Group 1:

### 3.2.4.1 Management and Training

- Change acquisition policies to accommodate evolutionary acquisition.

- Educate all stakeholders on various acquisition alternatives such as the evolutionary acquisition model.

- Train all stakeholders on the value and role of prototyping in the system life cycle.

- Involve all stakeholders in requirements:

    - Determination

    - Validation

- Realign incentives/milestones to more easily capture requirements "progress".

- Introduce risk-based decision making.

- Reduce DoD barriers to developer-user interaction.

### 3.2.4.2 Development

- Freeze requirements in small incremental builds.

- Develop more testbeds like AIN to validate interoperability earlier in the development process.

### 3.2.4.3 Research

- Develop new techniques to isolate acceptable requirements partitions.

- Develop new techniques to accommodate change in requirements and designs.

- Develop and refine practical formal requirements techniques.

- Define a multi-stakeholder requirements process.

- Develop thorough understanding of requirements "normalization." Somewhat analogous to database normalization, this envisioned technique would enable two sets of requirements to be shown to be equivalent.

- Define and understand requirements process models.

- Define and understand models of requirements progress.

- Perform experiments to determine what conditions make evolutionary acquisition and prototyping practical.

- Develop tools/techniques to capture merits/tradeoffs among requirements.

### 3.2.5 Glossary

**Requirements Specification** - A *requirements specification* is a document containing all the requirements for a system.

- A requirements specification is *complete* if everything that all the eventual stakeholders (customers, users, etc.) want is specified.

- A requirements specification is *consistent* if no two subsets of requirements conflict.

- A requirements specification is *unambiguous* if every one of its requirements has only one possible interpretation.

**Guarantor** - The *guarantor* is the group of stakeholders who are the final authority on the sanctioning of the requirements and the validation statements.

**Prototype** - A *prototype* is a partial implementation of a system constructed primarily to enable customers, users, or developers to learn more about a problem or its solution.

**Referent** - A *referent* is a baseline (such as a requirements specification document) to which we compare the requirements for validation.

**Stakeholder** - A *stakeholder* is an individual, group, organization or system which can influence or be influenced by the computer system.

**Validation Principle** - A *validation principle* is the accepted warrant that is appealed to in order to justify the validation process.

**Validation Statements** - *Validation statements* constitute the rationale or proof that connects the requirements to their referent. Some participants maintained that a complete proof for a requirements set is impossible.

## 3.3 Working Group 2:

## Requirements Engineering Methodology, Tools, and Languages

Edited by: Dr. Raymond T. Yeh, Working Group Chair, with Dr. William Gilmore.

### 3.3.1 General Information

#### 3.3.1.1 Working Group Participants

| NAME | EMPLOYER | COUNTRY |
|------|----------|---------|
| Comer, Edward R. | Software Productivity Solutions, Inc. | USA |
| Fisher, Gary E. | National Institute of Standards and Technology | USA |
| Gilmore, William | International Software Systems, Inc. | USA |
| Hamilton, Margaret | Hamilton Technologies, Inc. | USA |
| Harris, Robert L. | Wright Patterson Air Force Base | USA |
| Hsia, Pei | University of Texas at Arlington | USA |
| Labbe, Jean-Claude | Defence Research Establishment (Valcartier) | Canada |
| Larson, Aaron | Honeywell/Systems and Research Center | USA |
| Looney, Michael J. | Admiralty Research Establishment | England |
| Manley, Gary | Naval Postgraduate School | USA |
| Marks, Walter | CECOM Center for Software Engineering | USA |
| Ng, Peter | New Jersey Institute of Technology | USA |
| Racine, Glenn E. | AIRMICS | USA |
| Rzepka, William | Rome Air Development Center | USA |
| Samson, Donaldine | Sonex Enterprises, Inc. | USA |
| Singer, Carl A. | Bellcore | USA |
| Tanik, Murat M. | Southern Methodist University | USA |
| Yeh, Raymond T. | International Software Systems, Inc. | USA |

#### 3.3.1.2 Roadmap: A Guide to Working Group 2 Activities

This report on the activities of Working Group 2 consists of four parts. The introduction presents the group's approach of dividing into four subgroups, one each for methodology, tools, languages, and integration. It summarizes the major issues the working group addressed as well as the major recommendations it proposed, covering policy and management, development, and research. Next follows a section on methods and tools, which addresses the six interdependent subprocesses that, according to the group, best describe the requirements engineering process. For each subprocess, discussion is provided on the activities, methods, and tools that apply to it; an analysis of the problems and issues that occur within it; and recommendations. Activities across all subprocesses are addressed at the end of this section. The language section follows, focussing on problems and issues, objectives, features of existing languages, and recommendations. The report concludes with a glossary of key terms.

## 3.3.1.3    Working Group Assignments

The distinguished participants of Working Group 2 are divided into the following subgroups:

| Methodology | Tools | Languages | Integration |
|---|---|---|---|
| Gilmore, William | Comer, Edward R. | Fisher, Gary E. | Comer, Edward R. |
| Harris, Robert L. | Looney, Michael J. | Hamilton, Margaret* | Gilmore, William |
| Hsia, Pei* | Manley, Gary | Labbe, Jean-Claude | Samson, Donaldine |
| Ng, Peter | Marks, Walter | Larson, Aaron | Yeh, Raymond T.* |
| Samson, Donaldine | Racine, Glenn E. | Tanik, Murat M. | |
| Singer, Carl A. | Rzepka, William* | | |

*Subgroup chairperson.

Acknowledgment: The whole group wishes to thank COMCON, Inc., especially Diane Alexander, for their extensive support and technical contributions.

## 3.3.2    Introduction

Requirements engineering is a new, vital frontier for software research. Several organizations are researching and developing requirements engineering processes. These processes are only practical and cost-effective when supported by the appropriate methodologies, language, and tools. Many software engineering tools and methodologies have been developed to solve parts of the software engineering problem. But the methodologies, languages, and tools for software requirements have not received adequate emphasis in an integrated sense for a complete requirements process.

Requirements engineering methodologies, languages, and tools are support mechanisms for any requirements engineering process. The objective of Working Group 2 was to investigate specific mechanisms relating to a full spectrum of activities within the requirements engineering process.

Working Group 2 initially assumed that the requirements process is extensive over time and in level of detail, i.e., it may include generations of systems and broad domain analysis, as well as detailed systems specifications concerning user needs. Furthermore, it was assumed that the process is intertwined with the overall system evolution and has the following six generic sub-processes:

1.    **Context Analysis** - analysis of problem space and application domain; deals with description of problems only, not solutions.

2.    **Objective Analysis** - analysis of the solution space, and system objectives for life time use.

3. **Requirements Determination** - specification of characteristics the system must meet to satisfy user needs.

4. **Requirements Analysis** - analysis of expressed requirements; includes related refinement, elaboration, and correction.

5. **Synthesis** - formation of a cohesive specification from the detailed analyses; involves integration of partitioned analyses occurring due to problem complexity and breadth.

6. **Validation** - ensuring that the expressed requirements match real user needs and constraints.

These six generic requirements sub-processes do not necessarily occur sequentially, and are interdependent. Furthermore, the support mechanisms, which are methodology, tools, and languages, are interdependent.

The Working Group 2 approach was to break into individual analysis groups, one each for methodology, tools, and languages, and a fourth specifically for integration. Intermittent synthesis occurred by collective meetings and was catalyzed by the integration subgroup.

In order to analyze the support mechanisms, the subgroups were tasked with identifying specific activities associated with each sub-process, and identifying specific support mechanisms for these activities and sub-processes. Some of the activities, such as prototyping, span more than one sub-process. Detailed analyses for each sub-process are presented in individual sub-sections in this report.

The language analysis is presented in a separate section because the Language subgroup felt that language support integrates with the other areas in a broad way. The Language subgroup analyzed requirements for a common requirements language schema.

## 3.3.2.1 Major Issues

The following major issues surfaced during subgroup analysis and synthesis:

- Policy and Management Issues:

    - There is lack of widespread awareness of the importance of requirements engineering, especially in management and acquisition offices.

    - There is a lack of emphasis for the requirements process throughout the life cycle, and for its related policy and funding support.

    - There is general unawareness that requirements engineering is vital to system success, and hence to national security and economic vitality.

33

- Development and Research Issue

  - Currently used languages and methods fail to capture requirements information to effectively enable system evolution.

  - Lack of understanding of the so-called "non-functional" requirements - performance, reliability, maintainability, security, safety, etc.

  - Tools are not integrated to support the widespread needs of the requirements process.

  - Lack of effective means to salvage large investment in current, large software systems.

  - Lack of understanding of what to measure and how to measure key requirements process parameters.

3.3.2.2    Major Recommendations

Major recommendations developed by Working Group 2 are as follows:

- Policy and Management Issues

  - Change acquisition policies and management practice to support a requirements - centered development life cycle model.

  - Increase training of management/acquisition personnel in requirements engineering.

  - Establish an information/consultation center on requirements engineering (process, methods, tools, and metrics).

  - Reallocate currently available funds supporting downstream software activities to requirements engineering activities, (i.e., concentrate more resources on identifying and confirming what is to be built, rather than on how to build it).

- Development and Research Recommendations:

  - Develop wide spectrum language to support acquisition, representation, and reuse of requirements information and its related knowledge.

  - Develop methods to capture, integrate, and measure the so-called non-functional requirements.

  - Develop an integrated environment of requirements engineering tools.

  - Develop methods and tools to support reverse engineering of current systems that are able to be modernized.

34

-     Determine and develop meaningful metrics supporting modern requirements engineering practice.

### 3.3.3    Methods and Tools Support for the Requirements Process

This section presents the detailed results of analyzing the six generic requirements sub-processes. Each sub-process was analyzed for the following:

- The detailed activities that are components of that subprocess, methods for performing the activities, software tools that assist in performing the activities (presented in a table and related discussion);

- Problems and issues concerning methods and tools; and

- Recommendations and research areas concerning the methods and tools.

Recall that the six generic requirements engineering subprocesses are:

Context Analysis
Objective Analysis
Requirements Determination
Requirements Analysis
Synthesis
Validation

### 3.3.3.1    Context Analysis

Context analysis involves analysis of the problem space and application domain of a potential system to be developed. It deals with description of problems only, not solutions. (See Table 1.)

### 3.3.3.1.1    Discussion

Context analysis is a general activity under which four major sub-activities were identified. Requirements are those defined and derived from the "setting" within which the system must operate.

Identification of the problem space boundaries is important for understanding the environmental constraints under which systems will be developed, operated, and evolved. Methods for performing this activity include document reviews (mission, scenerios, and higher-level requirement statements of existing systems), interviews with potential users, market analysis, and policy guidelines. People involved include decision-makers and potential users. System environment identification also includes the physical, functional, economic, social, and cultural parameters that will be associated with or that affect requirements.

35

**Table 1.** Activities, methods, and tools for context analysis

| | |
|---|---|
| **Activities** | Identify problem space boundaries:<br>    political<br>    cultural<br>    legal<br>    resources  (material, human, informational, financial)<br>    organizational policies and procedures<br>    technological scope<br><br>Needs identification:<br>    identify market needs and trends<br>    threat assessment<br>    problems with current systems<br>    identify the common needs of different organizations<br><br>Application modeling<br>    enterprise modeling<br>    mission modeling<br>    identify information resources<br><br>Postulating solutions |
| **Methods** | Interview<br>Document Reviews<br>Conceptual Modeling<br>Delphi<br>Group Decision Support<br>Analysis<br>Surveying Current Systems<br>Observation<br>Role-Playing<br>Walk-through<br>Gaming |
| **Tools** | Concept Modeling Tools, e.g.:<br>    P-Tech<br><br>Knowledge Engineering Tools, e.g.:<br>    Expert System Shells, Prolog<br><br>Enterprise Modeling, e.g.:<br>    Entity-Relationship Models, Activity Models, Behavior Models<br><br>Simulation Models |

A second major sub-activity involves needs identification. This includes interviews with users of existing systems, customer questionnaires, reviews of official needs documents and statements of needs from customers, and market surveys. Support methods also include "Delphi", modeling, and critiquing of existing systems.

A third major sub-activity identified is application modeling. This involves spelling out those effects governed by the surrounding user's community that will affect requirements.

It may involve modeling the general user requirements at a meta-system level, using enterprise modeling tools. Such models should project future changes. Personal interviews and review of materials concerning the user's environments provide information. Participants include the customer/user. How the system will be maintained is an important consideration that feeds this activity. The system should be considered from the viewpoint of the business and its procedures and structure/organization. This leads to consideration of the "business" working methods and related ramifications in terms of need/change.

A fourth major activity is postulating solutions. It is performed not so much to identify solutions as to help clarify the problem. This activity may involve surveying technology, conceptual modeling, and gaming. Concept modeling and simulation tools support this activity.

### 3.3.3.1.2 Problems and Issues

The context analysis phase requires the management of many pieces of informal information. This information is dynamic and unstable and so it requires flexible tools.

The problems with these can be generally categorized as being too removed from those specifying the requirements and being too complex for them to make good use of the capabilities. The information being captured is in a large number of cases too general or informal. Most of the tools are static and require extensive resources both in terms of manpower and computers to simulate "world models" and provide meaningful outputs rather than the obvious.

### 3.3.3.1.3 Recommendations and Research Areas

This relatively infant sub-process needs extensive modeling in a number of areas to provide a base of support. Initially it should be supported by R&D. Modeling will involve knowledge acquisition and representation, and utilize common structured knowledge. Further research is needed regarding elicitation techniques.

Further support for multiple domain analyses is also needed, and these should model adaptation, change, what-if scenarios, and sensitivity analyses.

### 3.3.3.2 Objective Analysis

Objective analysis involves analysis of the solution space, and system objectives for lifetime use. (See Table 2.)

### 3.3.3.2.1 Discussion

This activity focuses on defining the "mission-level" requirements of a system. Definition as to how the system will satisfy user needs over the long-term is captured and refined. Therefore, the activities listed in Table 2 are intended to focus on defining (and later revising) the high-level, long-term objectives that the system, and all aspects related to its evolution, should satisfy.

37

**Table 2.** Activities, methods, and tools for objective analysis

| | |
|---|---|
| **Activities** | Define specific problem to be solved |
| | Define system/environment boundary and interface |
| | Define life cycle profile:<br>    length of use<br>    expected breadth of use<br>    desired Return On Investment |
| | Define user profile:<br>    frequency of use<br>    education/experience of user |
| | Identify non-functional requirements:<br>    necessary reliability, security, performance, etc. |
| | Identify critical success factors:<br>    prioritize major objectives |
| | Identify operational capabilities:<br>    basic needed functions of the target system<br>    determine wish lists of major objectives |
| | Conduct feasibility analysis<br>    physical/technical, financial, political, cultural |
| | Uncertainty and risk assessments for major objectives |
| | Perform trade-off analysis of major objectives |
| **Methods** | Interview |
| | Documention Review |
| | Trade-off Analysis<br>    modeling, role-playing |
| | Build scenarios of high level system usages, possibilities |
| | Delphi techniques |
| | Group decision support methods |
| **Tools** | Conceptual Modeling Tools |
| | Knowledge Engineering Tools |
| | Enterprise Modeling Tools |
| | Security Models |
| | Reliability Models |
| | Formal Verification Tools |

In addition to identifying the system/boundary interface, operational capabilities, and analyzing feasibility regarding technical, operational, and economic factors, there is other important information to gather. There is need to identify the expected breadth of use, and long-term time and economic scope of the new system. This includes developing a

38

long-term plan and an acquisition scheme, including a scenario of planned yearly goals, and a projection of the kinds of contracts to be used. It should also identify the anticipated evolution of the new system, i.e., is the system expected to support a static or dynamic environment. Toward this end it is particularly important to identify the critical success factors for primary decision-makers who will use the new system (this promotes estimates of Return on Investment (ROI) for the project, and trade-off analyses).

In addition, there is need to identify the resources for information contributing to requirements determination. This may involve creation of a plan for who, generically, should participate, and how to sustain continuity of expertise over the whole life cycle.

Activities also include identification of constraints, especially with respect to policy constraints levied by government by economic realities, current market conditions, or availability of resources. Schedule is also a constraint in terms of meeting a "window of opportunity".

Finally, we note that non-functional requirements concern reliability, security, maintainability, extensibility, etc. Allocation of priorities to objectives is also done. In order to prepare for work in deciding among alternatives, evaluation criteria called alternatives metrics must be considered.

People involved in the objective analysis process include experienced user and domain specialists (e.g., Training and Doctrine Command (TRADOC) people in the army), system architects (e.g., industry experts), operations research analysts, financial analysts, and policy makers.

Applicable tools during objective analysis include those tools which were used during context analysis. In addition, modeling tools which help with some non-functional requirements have been developed. For example, security models, formal verification systems, and reliability modeling tools now exist.

### 3.3.3.2.2  Problems and Issues

In general, the problems with modeling tools here concern their limited applicability, e.g., security modeling addresses a very big problem but in a very narrow domain of applicability. In addition, these modeling tools fail to scale up to realistically sized systems. In some cases, especially the reliability models, credibility of the results is an issue.

### 3.3.3.2.3  Recommendations and Research Areas

There needs to be R&D for how to specify non-functional requirements. In particular, we need methods and tools to

*   Support conflict resolution, e.g., maintainability vs. reliability,

*   Enable specifying "degree of", e.g., quantifying, such as levels of security,

- Help identify relationships among the "ilities",

- Model with wide applicability, e.g., scale up kinds of current modeling,

In addition, we need R&D to learn how to do more relevant workload modeling, analysis, and simulation.

### 3.3.3.3   Requirements Determination

Requirements Determination involves specification of characteristics the system must meet to satisfy user needs.  (See Table 3.)

### 3.3.3.3.1  Discussion

The requirements determination activity uses and analyzes the gathered information from context and objectives analyses (goals, objectives, and needs) to create a comprehensive list of requirements for the system to be developed.  Alternatives are identified and evaluated.  For each alternative under study, a feasibility study must be performed to assess the ability of the sponsoring organization to develop the alternative, technically and with respect to available resources.  This activity also involves on-going revision and evolution of such requirements.

In general, requirements can be classified as either functional or non-functional, although there is substantial interdependence.  Non-functional requirements traditionally refer to constraints, necessities in performance and security, and the "ilities" such as quality, reliability, availability, maintainability, etc.  The satisfaction of many non-functional requirements depends on whether and how certain functional requirements are met.

Methods focus on investigation through the building and examination of prototypes (functional/operational) to understand the requirements in-depth.  Generally, the combined set is not easily comprehended without some form of realistic viewing or testing.  Investigation is also supported by interviews with customer/user/management-personnel (who have been identified in the phase of context analysis), and by document review and feedback of information among the role players.

Specification methods, such as data flow and object-oriented, help thinking through the problem and characterizing the functional requirements for communication.  Templating supports the capture and description of non-functional requirements.  The techniques of $n^2$ charting and modeling, in association with prototyping, support trade-off analyses.

Among existing tools that deal with requirements determination are the range of currently available requirements modeling tools which support data flow diagrams, functional decomposition, state-transition diagrams, entity relationship diagrams, petri-nets, stimulus response networks, etc.  Other tools that are applicable here, especially for determining the feasibility of alternatives, include model development tools for analytical models, simulation models and cost models.  In the area of simulation models, some success has been gained by "tuning" or "tailoring" a model to a very narrow and specific application domain so that its results are produced with greater fidelity.

**Table 3.** Activities, methods, and tools for requirements determination

| | |
|---|---|
| **Activities** | Determine system requirements |
| |     Analyze identified needs |
| |     Examine different user viewpoints |
| |     Perform transaction analysis, create scenarios |
| |     Identify, analyze data requirements |
| |     Determine functional requirements |
| |     Determine non-functional requirements |
| | |
| | Identify alternatives, wish lists, potential enhancements or modifications |
| | |
| | Perform trade-off analyses |
| |     benefit for added cost |
| |     benefit for extra risk |
| |     expected lifetime, evolveability of solutions |
| |     uniqueness of solutions vs. common needs of different organizations |
| | |
| | Identify problems, issues, risks |
| | |
| | Do Planning |
| |     Workload characteristics expected for the future system |
| |     Developmental constraints |
| |     Schedule and resources needed |
| |     Allocation of people and resources to tasks to be performed |
| | |
| **Methods** | Prototyping |
| | |
| | Interviewing |
| | |
| | Specification |
| |     data flow, object oriented, state transition |
| | |
| | Templating |
| |     $n^2$ Chart |
| | |
| | Reviews with people, e.g: |
| |     discussion groups |
| | |
| | Study and observation: |
| |     current environments, existing systems, related documents |
| | |
| | Market the idea |
| | |
| **Tools** | Requirements modeling tools |
| |     DFD, Functional Decomposition, State Drawings, E-R Diagrams, Petri, CORE |
| | |
| | Models |
| |     Analytical, Performance, Simulation, Cost |
| | |
| | Mission Specific Simulations |

## 3.3.3.3.2 Problems and Issues

There is a need to develop improved process and methods to help identify true requirements. Problems concerning tools limitations were also identified. Specifically, cost models are usually driven by "old" data, or as in the case of Ada projects, by databases

which simply do not have sufficient information or enough existing Ada projects for baselining. Simulation models are limited in scope.

### 3.3.3.3.3 Recommendations and Research Areas

The group recommends that research be supported to develop improved process and methods, and to increase coupling between tools. To support coupling we should develop a CASE database objects standard. The integrated tools should include comprehensive multiple-view support with consistency checking, view generation, and support for generation of test cases. Future simulation tools should support multiple levels of abstraction and be able to handle changes in information easily (e.g., interactively).

### 3.3.3.4 Requirements Analysis

Requirements analysis involves analysis of expressed requirements; it includes related refinement, elaboration, and correction. (See Table 4.)

### 3.3.3.4.1 Discussion

This activity focuses on improving the consistency, completeness, correctness, and feasibility of the existing set of determined, expressed requirements for a given system. Consistency checking looks for requirements which are in contrast or direct conflict with others. Completeness checking looks for omissions in the expressed requirements that could significantly affect developers' ability to understand or build what is wanted. Correctness checking examines whether the set of expressed requirements, if followed, will result in a system which will satisfy the user and long-term needs and objectives. Feasibility analysis looks at whether the set of expressed requirements are feasible in terms of technology, operation, and economy.

In addition, this activity includes evaluation of usefulness, that is, to what degree will such a developed system satisfy the current and future needs of the organization. Significance, certainty, and interdependency are evaluated to help plan and prioritize work, especially in the face of uncertainty and future requirements revision, and for support of tradeoff analysis. Testability is evaluated both because it is needed as well as because it is a measure of the quality of expression and understandability of the requirements.

Finally, this activity includes identification of the linkage of requirements and review of their traceability in order to support thoroughness and consistency of future revisions of the expressed requirements, to support testing the requirements against the actual system, and to support maintenance of the developed system when needed changes or repairs are desired. Several methods and associated tools apply to these activities. Many, but not all, are listed in Table 4.

**Table 4. Activities, methods, and tools for requirements analysis**

| | |
|---|---|
| **Activities** | Consistency checking |
| | Completeness checking |
| | Correctness checking |
| |     compare specifications to major system objectives |
| | Analyze feasibility |
| |     physical, financial, cultural, political |
| | Review testability |
| | Review traceability and linkage |
| | Evaluate significance, certainty, and interdependencies |
| **Methods*** | Prototyping |
| | Structured Analysis (including modifications, real-time extensions), e.g.: |
| |     DeMarco, JSD, SADT, Yourdon, Ward-Mellor, Hatley-Pirbai |
| | Object-Oriented Analysis, e.g.: |
| |     OO1 AXES, OORA |
| | Finite State Machines |
| | Other specification methods, e.g: |
| |     E-R Models, Operational, Petri-Net, PSL/PSA, RLP, SREM, USE |
| | Quantitative analysis, mathematical modeling |
| | View Analysis |
| | Ranking, weighting, prioritizing |
| | Scenario Building |
| | Simulations |
| **Tools*** | Prototyping Tools |
| | Requirements Modeling Tools. e.g: |
| |     Cadre, IDS |
| | Analysis Tools/Models |
| |     consistency, completeness, performance |
| | Specification tools, e.g: |
| |     Statemate, Dream, PAISLey, PCSL, RTRL, SSL |
| | CORE |

\* Note: Most of these methods and tools are associated with languages.

For more thorough listings and descriptions of methods and tools, see (a) "Software Methodology Catalog" (U.S. Army CECOM Center for Software Engineering 1989 D Ft. Monmouth, NJ 07703-5000); (b) "Mapping the Design Information Representation Terrain" (Webster, D., 1988 D IEEE Computer, Vol 21, No. 12); (c) "Requirements Engineering: A Systematic Survey of the Literature" (King, K.N., 1987 D Software Engineering Research Center, Georgia Institute of Technology, Atlanta, GA 30332).

### 3.3.3.4.2  Problems and Issues

Several problems with tools for the requirements analysis activity were identified.  First and foremost, there are no multi-representational tools (ones which can accomplish all analytical aspects) currently available.  Another major shortfall identified was the inability of current tools to tailor, or fine tune, their representation.  Other tools suffered limitations as well.  Consistency tools should involve balancing various models to ensure that the processes and data identified in one model are consistent with another, e.g., Data Flow Diagram (DFD) vs. Entity-Relationship Diagram (ERD), State Transition Diagram (STD) vs. DFD, but such tools are limited.

Problems involving the extensibility and robustness of the tools were noted as well.  Current completeness tools are concerned with ensuring data identified is within ranges and values at identified points, but completeness involves much more than this.  Current performance tools are concerned with evaluating selection by performing "rough checks"; they lack detail and are not supported by models.  Such rough evaluations are insufficient for yielding the analysis results needed to specify systems more completely, feasibly, and to support satisfaction of the "ilities".

### 3.3.3.4.3  Recommendations and Research Areas

Recommendations from the requirements determination section apply to this activity.  In addition, further research into knowledge-based support tools is recommended for requirements analysis.  Prototyping tools need user interface definitions which are transparent to implementation hardware.  More robust modeling of function and performance of proposed specifications is needed, i.e., closer to actual real-world situations. Research is needed to learn how to capture non- functional requirements to the extent that the impact to proposed changes in a non-functional requirement can be predicted.  Finally, support for development of tools to help generate and capture operational scenarios is recommended.

### 3.3.3.5  Synthesis

Synthesis involves formation of a cohesive specification from the detailed analyses; it also involves integration of the partitioned analyses that  have occurred due to problem complexity and breadth.  (See Table 5.)

### 3.3.3.5.1  Discussion

The activities here are focused on synthesizing, integrating, revising, and polishing expressed requirements into a feasible, consistent, beneficial set.

Prototyping for synthesis involves constructing or using prototypes to check if the set of requirements can be synthesized into a system.  Similarly, simulations should mimic the entire system, not just specific parts, to examine how well the eventual system will do the job.  Sanity checks compare sets of requirements to check if they violate one another's basic assumptions.  Logical models are used to reveal any potential problems with the whole set of requirements.

44

**Table 5.** Activities, methods, and tools for synthesis

| | |
|---|---|
| **Activities** | Resolve conflicts |
| | Merge models and viewpoints |
| | Integrate concerns |
| | Integrate non-functional and functional requirements |
| | Collect feedback to correct objectives and specifications |
| **Methods** | Prototyping |
| | Simulation |
| | Sanity Check |
| | Logical Modeling |
| **Tools** | Requirements Modeling Tools |
| | Prototyping Tools |

The main emphasis of the tools should be to help the user observe the requirements at work (i.e., in action).

### 3.3.3.5.2 Problems and Issues

The main problems center on tool deficiencies. Prototyping is not rapid enough. There is not enough support for import/export between tools and/or models, both internal to this activity, and between this and other major activities. In addition, there is often a problematic issue of what to do when a user wants to keep the prototype as a part of the real system (not throw it away after completion). Most prototypes aren't built to be user-robust.

### 3.3.3.5.3 Recommendations and Research Areas

Recommended research should focus on synthesis of data schemas, and rapid prototyping via application domain reuse. More robust executable specifications are needed to examine the logic and function of proposed behaviors in more realistic, dynamic ways. Generally, research support for requirements synthesis tools is needed.

### 3.3.3.6 Validation

Validation involves ensuring that the expressed requirements match real user needs and constraints. (See Table 6.)

45

**Table 6.** Activities, methods, and tools for validation

---

| | |
|---|---|
| **Activities** | Collect stakeholders critiques, evaluations, reviews, and analyses.<br>Stakeholders are:<br>    users<br>    customers<br>    developers<br>    QA people<br>    V&V people |
| **Methods** | Walkthroughs<br><br>Reviews<br><br>Inspections<br><br>Evaluations of:<br>    Mock-ups<br>    Prototypes<br>    Simulations<br><br>Testing:<br>    testbeds<br>    trial use<br>    alpha, beta testing<br>    feedback during informal development tests and integration |
| **Tools** | Executable Specifications<br><br>Prototyping Tools<br><br>Simulation Models<br><br>Scenario Analysis<br><br>Testbeds<br><br>Theorem Provers |

---

### 3.3.3.6.1 Discussion

Validation is critical to the requirements process. It entails examining the appropriateness of expressed, synthesized requirements to judge and revise the system mission and objectives, and any or all system specifications.

Validation of requirements is not the culmination of the generic requirements process. Rather, it is an on-going activity.

Whereas traditionally communication with the user community has been thought to be a critical factor only for the validation of requirements, we take exception to this view on two counts. First, we believe that communication with the user community is a critical factor for all the generic activities. Second, we believe that validation comes not just from the user community, but from all the stakeholders, e.g., users, customers, developers, QA, and V&V.

Methods emphasize collecting evaluations and experiencing the ramifications of expressed requirements through testing, trial use, thought experiments, etc. Support of breadth of use and examination is encouraged. Collection and assimilation of feedback is essential. Tools that support this activity include those for prototyping, generation of executable specifications, simulations, scenarios and testbeds. Proofs of correctness are a desirable feature for validation with theorem provers as a potential tool.

### 3.3.3.6.2 Problems and Issues

The major problems with the tools being used are their limited applicability (they don't scale up to a system-wide version) and the fact that many (most) of the requirements models are not interoperable with the validation models. This relates to the problem of inadequate import/export capability in most tools.

### 3.3.3.6.3 Recommendations and Research Areas

Recommendations for research include the following:

- Coupling working models to real-world stimuli;

- Enabling dynamic analysis through animation of requirements statements, especially time based analysis;

- Greater focus on long-term research, such as for theorem provers.

### 3.3.3.7 Activities Across All Phases

Several activities, methods, and tools were identified for most of the generic activities of the requirements process. (See Table 7.)

### 3.3.3.7.1 Discussion

Obviously, a commonly identified activity across all activities in the requirements process is creation and revision of some type of dictionary and/or documentation facility. This activity is coupled with traceability to support more seamless flow between requirements expression, development, and revision of both requirements and product. Impact analysis closely relates to traceability, as does configuration management. These activities are embraced by some current CASE tools, but most are limited in their applicability.

The identification of activity commonality can be deceiving. We cannot emphasize strongly enough that while the activity, and even sometimes the method and tool identified are the same, the focus or application of the activity is different. This is part of the reason for identifying the generic activities - to encourage these multiple focuses. Prototyping, for example, is an activity of trial building to investigate alternatives. "What it is" that is being investigated varies, depending on the main generic activity. Hence, the use and purpose of prototyping will vary. Similarly, there is variation depending on context for recording rationale; creating and using executable specifications, simulations, and

47

**Table 7.** Activities, methods, and tools applicable to several generic requirements activities

| | |
|---|---|
| **Activities** | Creating and/or revising documentation |
| | Creating/revising dictionaries |
| | Recording and checking rationale |
| | Traceability |
| | Impact analysis |
| | Configuration management |
| **Methods** | Prototyping |
| | Interviewing |
| | Reviewing documents |
| | Modeling |
| **Tools** | Traceability Tools/Databases |
| | Impact Analysis Tools |
| | Document Production Tools |
| | Data Dictionaries |
| | Configuration Management Systems |

models; interviewing; and acquiring feedback. The fact that the same, or closely related, methods and tools can be used to support these activities is a great advantage and opportunity. In the previous discussions of problems and issues we have indicated that this opportunity is not being sufficiently seized upon. For example, limited applicability was a commonly cited problem, as was lack of tool integration, lack of multi-representation, and lack of extensibility and robustness.

### 3.3.3.7.2 Problems and Issues

The number one issue with regard to the requirements process in general concerns primacy of requirements and needed education. Although it comes as no surprise to requirements engineers, the centrality or primacy of requirements needs to be reinforced as both a policy and a practice within the systems development life cycle. For example, the life cycle should prohibit a systems developer from changing a few lines of code and updating the systems design without also updating the requirements data base or certifying that the current design or code change does not change the requirements. The way to maintain a system is via the requirements - propose changes in the requirements data base (see para 5.3.7.3, recommendation B.), then review them (impact analysis, engineering review, management review), and finally forward the approved changes into design and implementation.

Other specific problems and issues identified as applicable to all the six generic requirements engineering subprocesses - context analysis, objective analysis, requirements determination, requirements analysis, synthesis, validation - were as follows:

- How do you identify the entry and exit criteria for each activity, e.g., how do you know when you're done defining a requirement;

- Robust methods and tools for trade-off analysis is lacking.

- Insufficient consistency and completeness checking at multiple levels of abstraction;

- Lack of integration of requirements and development processes;

- Lack of clear delineation between prototyping and mock-up impairs selection of different approaches to system validation and requirements determination;

- Lack of traceability and requirements linkage; e.g., need to identify a model to depict the relationships and interactions among a set of requirements;

- Insufficient ability to handle rapid change and its impact on requirements;

- Impact analysis tools are limited in capability;

- Most data dictionaries are not object oriented;

- Configuration management tools are limited, control does not extend to manage changes of each individual requirement.

### 3.3.3.7.3 Recommendations and Research Areas

Seventeen research topics were identified. Each is listed below along with explanatory text.

1.  Groupware to formulate and clarify operation concepts and critical success factors. A number of consensus oriented, decision-support oriented, and knowledge-based approaches towards facilitating group efforts are now surfacing. The application of these techniques to the early activities of the requirements engineering domain should be most beneficial.

2.  A life cycle requirements database to capture and manage attributes of individual requirements and provide traceability. Given that

    - The requirements data base is the central repository of the system requirements,

    - All changes to requirements need to use this data base to perform impact analysis of candidate changes, and

49

- This data base must be kept current to reflect all approved changes, there is then a premium on traceability and linkage of requirements as well as the management of requirements attributes, such as level of importance, degree of certainty (in the statement), potential for change, expected requirements life span, etc. Tools and methods are crucial to facilitating, evaluating and possibly automating these requirements data base maintenance tasks.

  This recommendation can be made even stronger. To support tracing requirements to designs to code to tests to documentation, etc., a requirements database must be integrated with a database which spans all development and usage activities, not merely activities which cover the requirements aspects.

3. Identify a requirements model(s) to describe the interaction among requirements to provide understanding and synthesis support. Extensive requirements specifications are difficult to understand holistically! In addition to tracing and linkage, as well as proximity analysis methods (such as incidence, precedence, reachability and clustering matrices) there needs to be a better understanding of the higher meaning of several requirements interacting together. Such synthesis of requirements can be supported by requirements models.

4. Mechanism for trade-off analysis. Tools and techniques are needed to capture, organize, and help evaluate the many trade-offs that occur in requirements development. Intelligent impact analysis is an example.

5. More seamless integration between tools, and between requirements representations, to support propagation of change. As the requirements change - either as direct changes to an underlying data base or as changes in generated textual or diagrammatic derivatives - all representations of the requirements must be updated to reflect the change. Research into better linkage between representations is needed. Correspondingly, there is need for automated tools that link such methods as data flow, object oriented, state diagrams, text, etc., so changes in any such representation are reflected in all representations. Other related tools include those for automatically maintained consistency, configuration management, and automated documentation generation.

6. Methods for self-consistent, rapid modification of large systems. When emergency changes are made to mission-critical software, the requirements are often not updated (synchronized). Better methods and automated support for maintaining requirements data base consistency are needed to correct this problem.

7. Reverse engineering methods to derive requirements from existing systems. A number of existing systems are not accurately reflected in their requirements. This greatly limits the use and re-use of those systems. Failing to maintain synchronization between the requirements statement and the implemented system as the system evolves, there is a need to use reverse engineering to (re-)synchronize them.

8. Process modeling tool for active guidance; and integration of major activities. Managers and engineers like neatly formed boxes with clean arrows leading between them. Unfortunately, the real world of software requirements is not that well ordered. There is a need to determine criteria for leaving a major activity, returning to one, and transversing among the different activities of the requirements engineering process. Data on project statistics that correlates historical decisions with results would be of value here.

9. Mechanisms and metrics for aiding selection of methodologies. How do we choose which methodology (object oriented, data flow, state-transition, etc.) is best for any given requirements life cycle task? Collection and publication of data on project statistics would support this.

10. Hierarchical, multi-level Process Definition Language (HPDL) to facilitate expansion of requirements including localization, decomposition and information hiding. This tool would provide a method for several requirements engineers (different stakeholders, each with different levels of responsibility and domains of expertise) to identify and add detail in an orderly way in formulating a requirement. Information hiding and multiple levels of detail are beneficial characteristics for requirements browsing or other usage of requirements expression. For example, an initial HPDL statement might be:

```
Develop a payroll accounting system
    Pay hourly employee
    Pay weekly employees
```

But "outliner" capabilities may enable other detail to be present or be added, e.g.:

```
Develop a payroll accounting system
    Pay hourly employees
    {Determine regular pay              { -- decomposition }
        [Sum up regular hours worked    [ -- still further refined by an accountant ]
        Multiply by regular pay rate
        Add in bonuses.]
    Determine over-time pay
    Calculate Deductions.}
    Pay weekly employees.
```

11. Mechanisms for expressing ambiguity. There needs to be a method to purposely express ambiguity (such as response must be fast) as a temporary place holder. A requirements management system would then prompt a query, when it finally needs clarification, such as: "What do you mean by 'fast'? Please provide parameters."

12. Rigorous approach to consistency and completeness checking at different levels. Although rigorous mathematical techniques exist for consistency and completeness checking at the lowest level of requirements detail, e.g., data item/process, techniques do not exist at any aggregate levels. In general, there needs to be multiple levels of formalism.

13. Quantification of factors in needs identification and analysis. A number of requirements attributes need to be quantified; methods and metrics are needed.

14. "Ilities"-driven requirements engineering methods. "Ilities", expressed in non-functional requirements, are either (1) those which do not directly trace to basic operational concepts but rather to external constraints, or (2) those requirements which, unlike functional requirements, we have not yet learned to express formally. A number of "ilities", chief among them maintainability (or flexibility to change), need to be built-in to requirements as special items to be considered throughout the requirements engineering process.

15. Template and tools for identifying and describing "ilities". First a template to identify the non-functional requirements or "ilities" (see item #14 above) in order to keep them from falling through the cracks and then tools and/or languages to evaluate and express these non-functional requirements are vital to this ever important portion of the requirements data base.

16. Research into the impact of parallel processing on the requirements process. Determine what impact, if any, parallel processing capabilities in the target hardware has on the requirements engineering effort.

17. Develop system mock-up approaches and tools to aid requirements determination and system validation. Mock-ups (not to be confused with prototypes) have great utility in requirements determination and system validation. This technology needs to be exploited via better understanding and better tools.

### 3.3.4 Requirements Languages

Requirements engineering languages are mechanisms to express and control requirements information. A requirements engineering language can be proposed in two basic forms:

- A syntax for a specific language notation, or

- A schema for incorporating several language notations.

The approach taken by the language subgroup was to identify problems and issues with current requirements specification languages, develop a set of objectives, and make recommendations for developing an encompassing language schema to incorporate the strengths of the many specific requirements language notations. The group's activity was focused by constructing a set of tables which related the objectives to both present day languages and the six subprocesses in the requirements definition process. In order to create these tables the group progressed through an *actual* requirements definition process. Table 8 reflects a summary of the tables created during the group session.

### 3.3.4.1 Requirements Language Problems and Issues

The generic problems of system requirements are inherently due, in large degree, to the difficulty in specifying these requirements in a formal language. English is much too

ambiguous and context dependent to be used for any but the most mundane requirements. Design languages and programming languages available today are too limited to express the range of information types and relationships needed to fully define and document system requirements. The discovery of system failures due to errors in requirements is a continuing nightmare.

Current problems with requirements specification languages include the following:

- Currently used languages fail to capture requirements information effectively to support system evolution.

- Non-functional requirements cannot be adequately specified.

- The synchronization problem - Because requirements specifications are separate from the systems they represent, there is no automatic way to ensure that changes to systems are reflected in the requirements, or vice versa.

- Current languages are not expressive enough to represent diverse viewpoints.

- There are too many gaps in knowledge about requirements engineering. This leads to gaps in the formalisms that can represent system requirements. Functionally, requirements languages are discontinuous and incomplete across the spectrum from concept specification to executable code specification.

- Too many facts have to be known before any requirements specification language can be used.

### 3.3.4.2 Requirements Language Objectives

A comprehensive and integrated technology is needed for use in defining and automatically developing software systems. These needs point to a wide-spectrum requirements engineering language. Such a language should be usable to both define a system and also support its development. Specifically, such a language should include the ability to:

- Capture real-world definitions -- These include the definition of functions and objects in an object-oriented environment, and the mechanisms to hide information based on different views.

- Be inherently reliable -- Implementation-specific results are traceable to requirements objects and changes in objects, inconsistency and logical incompleteness are not allowed from the largest to the smallest system details (e.g., data flow, priority, and timing errors are eliminated).

- Maximize flexibility -- Requirements can be specified independent of platform; can be used in various modes (e.g., prototyping, production, documentation); may exist in multiple forms or syntaxes but have a single semantic meaning; are generally

53

declarative and non-procedural; are portable, based on an open architecture, modular, and can represent various levels of abstraction.

- Maximize the opportunity for parallelism -- Dependent, independent, and decision-making patterns are made explicit attributes of requirements. Finding out about parallelism issues would not have to wait until implementation.

- Maximize automation -- Automatic generation of executable and non-executable forms of the system are supported; multiple forms of the language can be generated; multiple forms of documentation can be generated (e.g., 2167A documents, FIPS 38, 7935); and automatic analysis for adherence to control structures rules is supported.

- Maximize reusability -- The language supports parameterized user extensions. Reusability would not have to wait until after development.

- Maximize productivity -- The combination of the above objectives contributes to orders of magnitude of productivity improvements; e.g., maintenance is minimized due to elimination of errors in requirements specification and the system can be made visible in a variety of automatically generated forms for analysis from orthogonal viewpoints.

### 3.3.4.3 Language Table

An analysis of the importance of specific existing requirements languages against the objectives of an ideal requirements language is shown in Table 8, Part A. Part B of Table 8 shows the impact of language-related objectives on the six subprocesses of the requirements definition process. The uncertainties in these estimates are reflected in the group's judgement that the actual usefulness of the languages, based on real life experiences, seemed closer than a comparison of the averages suggest.

### 3.3.4.4 Requirements Language Recommendations

The analysis of requirements for a wide-spectrum requirements specification language led to the following recommendations:

- The language should incorporate both non-procedural and procedural constructs. It should require the user to enter a minimum of control and data management information.

- It should provide multiple views of the system based on environmental contexts, i.e., graphical for conceptual views, textual for analysis, etc., but the semantic meaning should be constant for all views.

- The language should be executable for animation, simulation, and prototyping purposes.

**Table 8.** Ideal requirements language objectives

| PART A | LANGUAGE-RELATED OBJECTIVES | | | | | | |
|---|---|---|---|---|---|---|---|
| Current Requirements Languages | 1 | 2 | 3 | 4 | 5 | 6 | Average |
| Logic Based | 2 | 3 | 3 | 2 | 1 | 4 | 2.5 |
| Functional | 2 | 3 | 3 | 3 | 2 | 4 | 2.7 |
| Ada | 2 | 4 | 3 | 4 | 2 | 3 | 3.0 |
| Object Oriented (Smalltalk, C++, Simula) | 4 | 4 | 4 | 2 | 2 | 5 | 3.5 |
| Structured Analysis (SADT, SA/RT, etc.) | 4 | 2 | 3 | 2 | 2 | 3 | 2.7 |
| VDM | 4 | 4 | 3 | 2 | 1 | 4 | 2.7 |
| 001 AXES | 4 | 5 | 4 | 5 | 5 | 5 | 4.6 |
| 4GL | 3 | 3 | 3 | 1 | 2 | 4 | 2.7 |
| PROTO | 4 | 3 | 4 | 2 | 1 | 4 | 3.0 |

| PART B | LANGUAGE-RELATED OBJECTIVES | | | | | | |
|---|---|---|---|---|---|---|---|
| Requirements Sub-Processes | 1 | 2 | 3 | 4 | 5 | 6 | Average |
| Context Analysis | 5 | 1 | 5 | 5 | 2 | 5 | 3.8 |
| Objective Analysis | 5 | 1 | 5 | 5 | 2 | 5 | 3.8 |
| Requirements Definition | 3 | 2 | 5 | 3 | 4 | 5 | 3.7 |
| Requirements Analysis | 3 | 2 | 5 | 3 | 5 | 5 | 3.8 |
| Synthesis | 2 | 5 | 5 | 2 | 5 | 5 | 4.0 |
| Validation | 2 | 5 | 5 | 2 | 5 | 5 | 4.0 |

Weighting Factors

1 = minimum effect     5 = maximum effect

Language-Related Objectives Key

1 = Captures Real-World Definitions     4 = Maximizes Opportunity for Parallelism
2 = Concentrates on Reliability     5 = Maximizes Automation
3 = Maximizes Flexibility     6 = Maximizes Reusability

Average (1-6) = Maximizes Overall Productivity

- It should minimally provide the mechanisms for defining abstract high level concepts, intermediate architectures, logical and physical design information, and environmental constraints in both canonical and orthogonal forms.

Several recommendations surfaced as prescient and necessary for implementation of many of the other recommendations:

- Develop knowledge representations for requirements information.

- Solve the problem of defining the so-called "non-functional" requirements.

- Map project management and control structures to system views for automatic determination of static and dynamic resource allocation.

- Develop a wide-spectrum requirements engineering language that meets the objectives defined in this section.

### 3.3.5    Glossary

**001 AXES** - Object-oriented requirements language and methodology based upon a concept of control.

**Behavioral Prototype** - A prototype used to model what the system is supposed to do. It is black-box, and exhibits responses to stimuli. It is used for concept exploration and validation.

**CORE** - Controlled Requirements Engineering.

**Delphi Method** - In a Delphi method several people prepare estimates independently and are then told how their estimates compare to those of the others. Next, they are allowed to alter their estimates. This leads to an iterative technique in which many of the estimates finally converge to a narrower range from which a single value may be chosen.

**DREAM** - Design Realization, Evaluation, and Modeling system.

**E-R Models** - Entity Relationship models.

**Functional Requirements** - Requirements that express behaviors expected of a system, i.e., what the system should do.

**JSD** - Jackson Structured Design.

**Meta-system** - The set of systems that together support a given domain.

**Mock-up** - Material simulation of a system component used to help visualize that component's functionality.

**Non-functional Requirements** - Requirements that express constraints, attributes, or qualities that systems must possess or exhibit.

**OORA** - Object-Oriented Requirements Analysis.

**PAISLey** - Process-oriented, Applicative, and Interpretable Specification language.

**PCSL** - Process Control Software Specification language

**Petri Nets** - A directed graph representation language supporting parallel design.

**Prototype** - An initial implementation of a component or a system. It is generally deficient in one or more areas (e.g., performance, functionality, or robustness), but is able to demonstrate some features of interest. Prototypes are useful for investigating system behavior and structure. See also Behavioral Prototype and Structural Prototype.

**PSL/PSA** - Problem Statement Language / Problem Statement Analyzer.

**Requirements Centered Development Life Cycle Model** - The requirements process serves as the command and control center for system evolution. It steers other activities (e.g., prototyping, design, testing, validation), but requires information input from those activities to do so.

**Reverse Engineering** - Getting the documentation for existing systems "in sync" with the system's actual implementation. This especially includes the requirements documentation.

**RLP** - Requirements Language Processor.

**ROI** - Return on investment.

**RTRL** - Real-Time Requirements Language.

**SADT** - Structured Analysis and Design Technique

**Scenario** - A sequence of events which occur in the system/environment setting, or only within the system itself. A frequent use of scenarios is to depict the reaction of the system (also an event) to one or more prior events, i.e., stimulus/response group(s).

**Scheme** - A way of performing a set of activities.

**Simulation** - An executable model or mock-up of the system, or a significant part of it, which exhibits behavior or characteristics that aid analysis of issues. The inner mechanism of the simulation may have little in common with the final system solution.

**SREM** - Software Requirements Engineering Methodology.

**SSL** - System Specification Language.

**Stakeholders** - Persons or organizations, by category, who are participants in the process and who have particular needs, concerns, or responsibilities related to system definition, development, use, or acquisition.

**Structural Prototype** - A prototype used to model how the system will accomplish its black-box behavior. Thus, a structural prototype is a clear-box model. It is used to determine feasibility, explore design alternatives, and estimate implementation and execution costs.

**USE** - User Software Engineering Methodology.

**Verification and Validation (V&V)** - Analysis to judge whether requirements artifacts adequately express user needs and meet other quality attributes; to judge whether the actual needs appear to have been perceived sufficiently; and/or to judge and evaluate the system in terms of progress toward satisfying the requirements.

## 3.4 Working Group 3:

## Rapid Prototyping and Knowledge-Based Techniques

Edited by: Dr. Winston W. Royce, Working Group Chair, with Mr. Robert M. Poston.

### 3.4.1 General Information

#### 3.4.1.1 Working Group Participants

| NAME | EMPLOYER | COUNTRY |
|------|----------|---------|
| Bagley, David | CECOM Center for Software Engineering | USA |
| Casey, Philip | US Army Training & Doctrine Command | USA |
| Conrad, Thomas P. | Naval Underwater System Center | USA |
| Greene, Cordell | Kestrel Institute | USA |
| Harris, David R. | Sanders Associates, Inc. | USA |
| Huskins, James | Naval Post Graduate School | USA |
| Johnson, W. Lewis | University of Southern California | USA |
| Little, Reed | Carnegie-Mellon University (SEI) | USA |
| Morel, Martin | Le Groupe CGI | Canada |
| Poston, Robert M. | Programming Environments Inc. | USA |
| Royce, Winston W. | SoftwareFirst | USA |
| Sobolewski, Victor C. | Australian Embassy | Australia |
| Stachowitz, Rolf | Lockheed | USA |
| Watgen, David | Advanced Technology Inc. | USA |

#### 3.4.1.2 Roadmap: A Guide to Working Group 3 Activities

This report on the activities of Working Group 3 is divided into four parts. The introduction defines the problem domain of the two subtopics and the working group's approach. This is followed by an issues section, then recommendations, and finally a glossary. The issues and recommendations sections treat the two subtopics separately. Each issues subsection begins by posing a series of questions that the group deemed central to the subtopic. The rest of the subsection analyzes each of the questions in turn. The recommendations are divided into recommendations for management and policy, development, and research.

### 3.4.2 Introduction

#### 3.4.2.1 Definitions and Problem Domain

The task of Working Group 3 was to analyze two specific aspects of requirements engineering: Knowledge-Based Approaches (KBA) and rapid prototyping.

Knowledge bases are repositories of formalized knowledge about a domain or area of expertise. A knowledge-based approach is a technique that actively employs knowledge bases and knowledge-based tools. KBAs may be used to facilitate and enhance requirements engineering.

A prototype is an executable model of a proposed system. It may include only a partial functionality of the final system. It is generally not optimized for performance and may be written in a fourth-generation language (4GL). Uses of prototypes include demonstration of the user interface of the system and testing of various aspects of the future system. Rapid prototyping refers to the incremental process of building prototypes in a relatively short amount of time.

Requirements engineering is currently a complex, error-prone, manual task. Often it is difficult to stipulate the requirements and specifications for a system at the beginning of a project. Yet, a thorough requirements engineering process can greatly improve product quality as well as increase the productivity of the design and test phases and reduce the amount of time spent on maintenance of the system. Knowledge-based approaches and rapid prototyping can be used to strengthen and improve requirements engineering. The task of Working Group 3 was to explore the issues involved in employing rapid prototyping and knowledge-based approaches for requirements engineering and to develop a set of recommendations aimed at incorporating these techniques into the software development process.

#### 3.4.2.2 Working Group Approach

Working Group 3 met in three sessions. Unlike the other working groups, it did not break up into individual subgroups. During the first session, the group considered a set of ten questions (five knowledge base questions and five rapid prototyping questions) which had been prepared in advance by Chairperson Dr. Winston Royce. Members of the group added their own questions to this list (for a total of twenty-one questions) and then voted to determine which questions were most urgent. Eleven of these were rejected as being of a lower priority, and the remaining ten questions were combined into a set of seven questions (four knowledge base questions and three rapid prototyping questions). For each question, one person was assigned to lead the discussion of the question and a second person was assigned to record the responses to the question (the scribe). The remainder of the first session was a brainstorming session in which answers to and pertinent issues of the seven selected questions were suggested and recorded. If the proposed ideas were in conflict, no attempt was made to reconcile the conflicts at this time.

The second session focussed on the seven questions for a second time, with question leaders and scribes in reversed roles. The issues and answers were elaborated on and conflicting views were resolved. Based on these issues and answers, a set of recommendations was developed and recorded.

The third session cleaned up any final concerns and made some minor changes to the issues, answers and recommendations. Question leaders and scribes then met to draw up the final issues and recommendations for each question in preparation for the 16 November presentation.

### 3.4.3    Issues

The following sections address the issues that arose while analyzing knowledge-based techniques and rapid prototyping.

### 3.4.3.1    Knowledge-Based Techniques

The following questions pertaining to knowledge-based approaches to requirements engineering were examined:

- Are knowledge-based approaches to requirements engineering useful for real systems? What kinds of requirements engineering problems are best solved by KBAs?

- What are the special risks of using KBAs for requirements engineering? What are the benefits of using KBAs for requirements engineering?

- What changes are needed in the software development process -- and what features are needed in our models of the software development process -- to exploit knowledge-based approaches?

- What are the existing knowledge-based systems and tools?

The following sections grapple with each of these questions in turn.

### 3.4.3.1.1    The Use of KBAs and Their Application To Real Systems

In determining the usefulness of KBAs for requirements engineering, the following observations were made:

- Size of application. The feasibility of KBAs for requirements engineering has been established for applications ranging in size from 1000 to 30,000 requirements. Extensions to higher ranges remain uncertain.

- Availability of expertise to establish knowledge base. The availability of expertise to establish the required knowledge base varies significantly with the application domain. Obviously, the more available the knowledge is (the human experts used

61

to build the knowledge base), the more potentially useful a KBA approach will be to the system.

- Maturity of KBA tools. Although several automated tools are available to support KBAs, there have been relatively few experiences involving large applications. Consequently, there is some question of tool robustness.

- Skill base for using KBAs. In contrast with the expertise required for building knowledge bases, it is unknown whether there will be need for long learning periods prior to effective use of KBA tools. It seems to depend on the particular tool.

- Quality of KBA-generated requirements. There is a definite need to develop data on the quality of KBA-generated requirements. It has yet to be established whether or not KBA-generated requirements are of a higher quality than "normal" requirements. The lack of such data is an obstacle to the extended use of KBA's.

- Quantification of costs/benefits. There is a definite need to develop data on costs and benefits deriving from the use of KBAs for requirements engineering. The lack of such data is a serious obstacle to the expanded use of KBAs in the near term.

- Functional vs. non-functional requirements. While there was intuitive agreement that KBAs are potentially useful for both functional and non-functional requirements engineering, concern was expressed about a more fundamental problem. There are no (known) KBAs that address non-functional requirements, and there is a serious need for research in the realm of knowledge acquisition regarding requirements.

- Context of the knowledge. The context must be sufficiently bounded for KBAs to be useful.

### 3.4.3.1.2   Risks and Benefits Of Using KBAs For Requirements Engineering

The following were identified as special risks of using KBAs for requirements engineering:

- High cost per user ratio. If an organization is going to build a knowledge-based system, substantial resources will be invested in the requirements analysis phase. The resulting knowledge base is typically narrow and application dependent, with a low probability for reusability.

- Lack of skill base. There is a lack of a skill base in doing requirements engineering and creating knowledge-based systems. This impacts system quality and cost.

- Lack of suitable methodology. Knowledge-based systems are new and very complex. Without a formal methodology, the system may be misused.

- Lack of productivity metrics. There is a lack of standardized, accepted productivity metrics which would demonstrate why it is better to use a KBA over another approach.

- Overdependence. Systems are not envisioned to replace human creativity and critical thinking.

- Liability. There are potential legal liability issues as to who would be accountable for any errors in the knowledge base and the harm they may cause.

The following were identified as special benefits of using KBAs for requirements engineering:

- Reuse. The use of a knowledge base would provide corporate memory as well as project memory. Better tracking of intra-project dependencies are facilitated by knowledge-bases. Later, products that reuse the knowledge bases will require fewer up-front cost and time investments. Existing knowledge bases could also be marketable.

- Better Management of Changing Requirements throughout the software life cycle. Knowledge-based approaches provide the means to improve the integration of requirements with other life cycle phases. They support "live" requirements, ie. requirements that are continually being changed and upgraded throughout the software life cycle. Knowledge-based approaches would also provide the ability to compute the impact of changing requirements on the system and to generate documentation from requirements.

- Improved accuracy. When used properly, it was felt that knowledge-based approaches can provide a better facility for consistency and completeness testing. They can increase the analyzability (of performance, security, etc.) and testability of a system. They can also provide the capability for rapid prototyping and requirements validation.

### 3.4.3.1.3   A Software Development Process To Exploit KBAs

In order to fully exploit knowledge-based approaches, the software development process should allow for the following:

- Evolving requirements knowledge bases. In procurements it is often necessary for requirements to evolve; therefore, the requirements knowledge base must also evolve. In this case the process model should include an incremental knowledge acquisition activity.

- Validation and consensus of the requirements knowledge base. Validation of the requirements KB by software builders, buyers, and users must be part of the process model.

- Development resources planning and allocations. Knowledge engineering requires a high up-front investment to develop and analyze the knowledge base. If the knowledge base can be reused for another system, cost and schedule will be significantly reduced for the next system's initial phases.

63

### 3.4.3.1.4 Existing Knowledge-Based Technology

The group recommended that the following be considered as examples of knowledge-based approaches to requirements engineering:

- ARIES

  - Integrates formal/informal requirements
  - Concepts as objects in knowledge base
  - Formal transformation of requirements

- REFINE

  - Commercial VHLL (Very High Level Language) specification language
  - Specification transformation

- GIST

  - Used in software factory project, ARIES
  - Operational high level specification language

- EXPRESS

  - VHLL specification language
  - Automatic programming

- EVA (Expert System Validation Associate)

  - Logic-based
  - Meta knowledge-based

- Programmer's Apprentice

  - Basis for Requirements Apprentice
  - Basis for KBEmacs

- T

  - Commercial VHLL specification language
  - Specification transformation
  - Automatic verification

- KATE

  - Interactive requirements analysis
  - Requirements specification

- KIDS

  - Algorithm design
  - Interactive

- DRACO

  - Domain level specification
  - Reuse of domain knowledge

- Domain-specific application development systems:

  - WATSON (TELEPHONY)
  - Phinix (Oil Exploration)
  - VLSI router

- Foreign efforts:

  - ALVEY
  - ESPRIT
  - 5th generation efforts

### 3.4.3.2 Rapid Prototyping

The following questions pertaining to rapid prototyping were examined by Working Group 3:

- Who should do prototyping? What are the products of prototyping?

- Do current regulations and standards discourage prototyping? What changes to the acquisition process are needed to accommodate prototyping?

- How are prototypes used? What properties should a prototype have? What are some examples of current prototyping tools? What properties do/should they have?

Some general insights that arose during the discussion of prototyping were:

- Prototyping yields a competitive edge. Contractors tend to treat prototypes as proprietary items because the prototypes can sometimes provide an edge in further contract competition.

- The software development schedule must be rearranged to allow prototyping to affect the final product. Prototyping requires that more time be allotted to the requirements phase of development.

- Can we afford to prototype? Can we afford NOT to prototype? Prototyping adds to the start-up costs of projects. However, the group feels that this development cost is more than justified because prototyping can reduce risks during system

development. Prototyping helps to determine the "correct" requirements from the start, increasing the percentage of system functionality that is "built right the first time." Also, it is far more expensive to change a requirement during advanced development stages, compared with the cost of making the change in an earlier phase.

### 3.4.3.2.1  Participants and Products in the Prototyping Process

- Representatives from every stakeholder group which perceives a risk in the outcome of the final system should be involved in the prototyping process. Figure 1 portrays the interrelationships between stakeholders in the development of a typical military system.

- Possible products of rapid prototyping include:

    - Formal specifications

    - Operational prototypes

    - Representation (model) of requirements

    - Validation of experimental hypotheses

### 3.4.3.2.2  Standards, Current Development Practices, and Prototyping

The DoD currently has the Software Development Standard, DoD-STD-2167A, to guide the software development and documentation process.

- Prototyping products are not recognized. The products of prototyping have not been recognized as standard contract deliverables. This makes it difficult for an acquisition agency to require prototyping and specify what is to be delivered.

- Regulations and Standards inhibit innovations such as prototyping. Since individuals prefer the security that compliance with a standard provides, they are reluctant to accept deviation or change.

- Design review process is not amenable to prototyping. Design reviews currently have a well-established structure and schedule which are not compatible with the evolutionary requirements development process.

- Development times preclude effective prototyping. Time lines for current acquisition projects do not include sufficient time in the requirements process for effective prototyping.

### 3.4.3.2.3  Uses, Properties and Examples of Prototyping Systems and Tools

The group determined that prototypes may have one or more of the following uses and properties, depending on the purpose of the prototype:
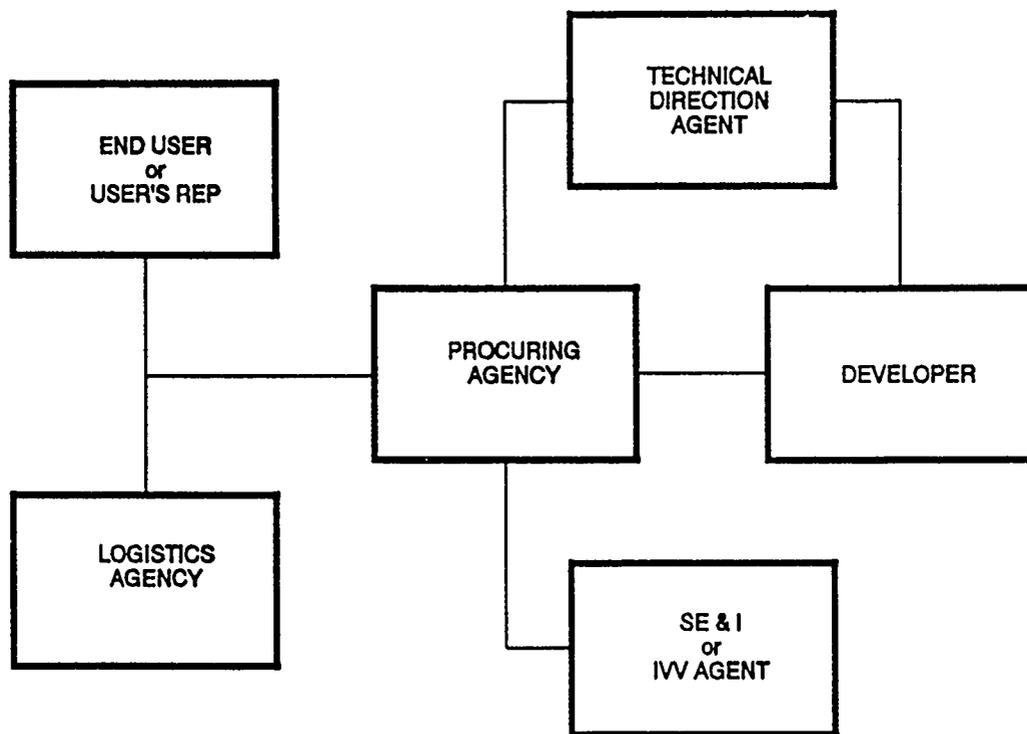
**Figure 1.** Interrelationships between stakeholders in the development of a typical military system

- Definition of an application's data domain (model), functional decomposition, and user interface. The prototype defines a model of the system to be built. It depicts the components of the system: the data and the functions that comprise it, and the user interface.

- Implementation of a subset of an application. A prototype may implement only some of the functionality of the proposed system in order to provide a rough model of how it will work.

- Provision of a tangible "running" system for the stakeholders. For an end user, the prototype can provide a hands-on, interactive representation of the final system. This type of prototype is mainly geared towards modeling the user interface. The prototype can also aid in the refinement of requirements. It can provide a clear demonstration of what a requirement is under at least one interpretation. This can bring out inconsistencies between stakeholders' requirements, providing a basis for discussion and reconciliation. For the developer of the system, the prototype can provide a tangible model of the system's behavior.

- Allow performance bottleneck prediction within the operational system and the development project process. A prototype can be constructed to provide a means of predicting the likely bottlenecks in a system, using alternate designs. It is not necessary for a prototype to be performance optimized. In fact, this may not be cost effective.

- Reduce risk. By implementing a prototype, users, developers, and managers, all players pictured in Figure 1, will have a clearer understanding of what functions require greater effort to implement than expected. The risk of unforeseen delays and uncontrolled costs will be reduced.

- Serve as a transition towards the implementation of an operational system. There was some disagreement as to whether or not systems should be built through successive refinement of the prototype. That is, whether systems should be constructed with evolutionary prototyping. It was agreed that this should be decided on a project-by-project basis.

The group recommended the following examples of prototyping tools:

- Data domain and functional decomposition tools:

  - 4GL RDBMS (Fourth Generation Language Relational Database Management Systems):

  - ORACLE, UNIFY.

  - Integrated CASE tools.

  - Software through Pictures.

- User interface definition tools:

  - Dan Bricklin's DEMO, Skylights GX, Videoworks, Supercard, Prototyper.

  - TAE Plus, Serpent, PROTO.

- Executable specification tools:

  - REFINE, APS, MICROSTEP, Statemate.

### 3.4.4 Recommendations

Recommendations are divided into those for knowledge-based techniques and those for rapid prototyping. Each section of recommendations addresses the areas of management and policy, development, and research.

### 3.4.4.1 Recommendations for Knowledge-Based Techniques

### 3.4.4.1.1 KBA Management and Policy Recommendations

- Formulate a new DoD software acquisition policy in order to:

    - Allow for an incremental, evolutionary process. A KBA development is typically incremental and evolutionary. Policy must sanction this methodology.

    - Accommodate KBAs in the requirements engineering phase. KBAs are resource intensive on personnel in the early phases of development.

- Develop and apply new software process model. We must gain practical experience in developing and applying this new, evolutionary model.

- Invest in KB development early in the process. Like changes in system requirements themselves, KBAs are less costly the earlier in a project they are introduced.

- Reuse knowledge bases in related projects. The knowledge base developed for one project should be useful for future projects.

- Amortize investments across many projects. Ideally this would be done in proportion to the expected payback of each individual project.

### 3.4.4.1.2 KBA Development Recommendations

- Initiate KBA for RE on a large, real project. It is important that we gain practical experience on a <u>real</u> project in order to determine where further development effort should be directed.

    - Minimize risk to the real project through use of a shadow project. This will provide the means to collect the necessary data without negatively impacting the main project. The use of a shadow project makes it possible to collect enough information to evaluate errors in the system in terms of the requirement specifications as well as the knowledge base and the tools that use the knowledge base.

    - Use the shadow project to:

        - Develop and apply quality metrics.

        - Develop and apply productivity metrics.

        - Perform cost and benefit analyses.

    - Consider change impact analysis as a candidate for a shadow project.

69

- Use previous experience with KBAs as input to future DoD standardization efforts. Past history will enable projection and minimization of the most probable errors for subsequent KBA efforts and provide a basis for standardization.

### 3.4.4.1.3 KBA Research Recommendations

- Extend research in verification and validation (V & V) techniques by using KBAs to test for:

  - Completeness -- All the requirements are satisfied.

  - Consistency -- There is no conflict among the requirements.

  - Correctness -- Every requirement satisfies the intended user need.

- Expand research in knowledge acquisition and management for RE. We need to know how to get the knowledge, and once we have it, figure out what to do with it. We also need to research configuration management of knowledge across products and projects. The KBs themselves become resources and can in fact be treated as commercial items.

- Expand research in knowledge acquisition and management in light of existing methodologies and tools.

- Extend research in more powerful models with greater expressiveness. There is a need to explore formalisms to encourage completeness checking in many different areas, such as:

  - Meta-models with self-knowledge. The knowledge base would have the ability to recursively explore itself.

  - Non-functional requirements. These include the so-called "ilities," such as maintainability, reliability, security, and performance.

  - Non-standard logics. For an adequate description of the possibilities, some situations require more than two truth values.

  - Non-monotonic logics. Many sets of requirements cannot admit certain new requirements without contradicting previously valid requirements.

  - Models with tolerance for inconsistency, uncertainty, etc. Projects often begin with insufficient or contradictory information; knowledge bases have to be able to handle these situations.

70

### 3.4.4.2 Recommendations for Rapid Prototyping

#### 3.4.4.2.1 Rapid Prototyping Management and Policy Recommendations

- Modify the development stages and time frames to be supportive of prototyping. The development stages need to be redefined and the amount of time required to complete each stage needs to be estimated. There may be a need for a separate requirements development phase.

- Define the objectives of requirements/design reviews for systems which use prototyping products. The use of a prototype creates the need to clarify the purpose of a design review.

- Define the products and the uses of those products for prototyping during the software development cycle. There is a need to specify the forms which the products should take and the manner in which they should be used. This information should be included within the appropriate military standard documents and guidelines for contract deliverables.

- Support competitive prototyping efforts. Contractors can, and should be able to, compete their prototypes against each other.

- Investigate alternative acquisition models. Consider, for instance, different contractors for the prototyping effort and the objective system development.

#### 3.4.4.2.2 Rapid Prototyping Development Recommendations

- Develop training strategies. Develop training programs for users, user representatives, and acquisition personnel to make them better aware of the prototyping approach.

#### 3.4.4.2.3 Rapid Prototyping Research Recommendations

- Conduct research on the traceability of requirements. Requirements should be traceable through the prototype and back into the development of the objective system.

- Conduct research on the validation of "non-functional" requirements. Prototyping should support the validation of non-functional requirements such as reliability (criticality, vulnerability and tolerance), maintainability, accuracy (precision), performance, timing, speed, and reusability.

- Conduct research on model documentation. Explore tools and process mechanisms which generate prototype model documentation. These tools should automatically document the user requirements, as demonstrated by the prototype.

- Conduct research on the communication of results. There needs to be a formalized method for communicating prototyping results between the various stakeholder groups.

- Conduct research on the legal issues of delivery. Contractual vehicles and responsibilities must be clear on the delivery of prototypes. Different parties may have different expectations of what the prototype should be, if prototypes are to be deliverable. There is a potential for the user who does not understand the purpose of a prototype to reject it as being a deficient system. Conversely, the user may want to field the prototype instead of the originally proposed system.

- Conduct research on the insertion of prototyping technology. Rapid prototyping has already caught on. We must learn from our experience in prototyping to better answer such questions as where in the life cycle prototyping should be used and what types of systems it is appropriate for.

### 3.4.5 Glossary

**Knowledge Base (KB)** - A repository of formalized knowledge about some domains and areas of expertise.

**Knowledge-Based Approach (KBA)** - A technique that actively employs knowledge bases and knowledge-based tools, and various programming techniques such as frames or rules.

**Meta-model** - As distinct from a model of a particular application, a model that, through knowledge of itself, describes the properties of, and the relations between, any and all the requirements statements of a system.

**Monotonic logics** - Logics in which the addition of new axioms does not invalidate previously proved theorems.

**Non-functional requirements** - Requirements that are not directly related to a particular function. Some examples include: reliability, availability, maintainability, security, ease of use, ease of learning, and performance.

**Non-monotonic logics** - Logics that are not monotonic.

**Non-standard logics** - Logics with more than two truth values.

**Requirements Engineering (RE)** - A systematic method for developing quantifiable and testable requirements.

**Shadow Project** - A separate, funded, research-like project that runs in parallel with, but does not impact upon, the main project.

### 3.4.6 Referenced Documents

Barr, A. and Cohen, P., <u>The Handbook of Artificial Intelligence, Vol. IV</u>, New York: Addison-Wesley, 1989.

## 3.5 Recommendations and Conclusions

The workshop produced many valuable insights and recommendations. These insights and recommendations are fully documented in these Proceedings. It is especially important to note the recommendations that were common to the three groups, which worked independently.

**3.5.1**  **DoD Policy Changes.**  Every group saw the need for DoD to change policy to accommodate evolutionary acquisition.

- Working Group One recommended DoD "make changes to acquisition policies and DoD standards to facilitate evolutionary acquisition."

- Working Group Two proposed DoD "change acquisition policies and management practice to support a requirements-centered development life cycle model."

- The third working group recommended DoD "formulate a new DoD software acquisition policy in order to allow for an incremental, evolutionary process ..." Further DoD needs to

  > ...modify the development stages and time frames to be supportive of prototyping. The development stages need to be redefined and the amount of time required to complete each stage needs to be estimated. There may be a need for a separate requirements development phase.

  In sum, we should "consider alternative acquisition models."

**3.5.2**  **Government Acquisition Personnel Training.**  All groups saw the need for increased training for Government acquisition personnel to make them more aware of Requirements Engineering issues and techniques.

- Working Group One recommended DoD "educate contracting officers and their technical representatives on the evolutionary acquisition approach; emphasize that system requirements can not be fully defined *a priori*; and that requirements engineering is continuous throughout the life cycle of the system." DoD must "educate program managers and team members that 'changing your mind' as a result of new information is acceptable." DoD must "train Government program managers in the use of acquisition models that employ prototyping."

- Working Group Two proposed DoD "increase training of management/acquisition personnel in Requirements Engineering." DoD should also "establish an information/consultation center on requirements engineering (process, methods, tools, and metrics.)"

74

- The third group recommended DoD "develop training programs for users, user representatives, and acquisition personnel to make them better aware of the prototyping approach."

**3.5.3   Requirements Validation.**   Every group saw the need for additional emphasis and exploration in requirements validation.

- Working Group One recommended DoD "develop an explicit requirements validation plan for every project."

- Working Group Two recommended research for "coupling working models to real-world stimuli; enabling dynamic analysis through animation of requirements statements, especially time based analysis; and greater focus on long-term research, such as for theorem provers."

- The third working group proposed research into how prototyping can "support the validation of non-functional requirements such as reliability (criticality, vulnerability, and tolerance), maintainability, accuracy (precision), performance, timing, speed, and reusability."

**3.5.4   Measuring Requirements Related Attributes and Progress.**   Most of the participants recognized the need for additional research in defining and using methods of measuring requirements related attributes and progress in the Requirements Engineering process.

- Working Group One recommended "DoD develop and use effective metrics to measure requirements progress and completion."

- Working Group Two saw the need for DoD to "determine and develop meaningful metrics supporting modern requirements engineering practice.   ...A number of requirements attributes need to be quantified; methods and metrics are needed."

**3.5.5   Non-Functional Requirements.**   Most identified the need for further work in specifying non-functional requirements.

- Working Group Two emphasized in several places the need to better address non-functional requirements.  They stated DoD must

> develop methods to capture, integrate, and measure the so-called non-functional requirements. There needs to be R&D for how to specify non-functional requirements.  In particular, we need methods and tools to: support conflict resolution, e.g., maintainability vs. reliability; enable specifying 'degree of', e.g., quantifying, such as levels of security; help identify relationships among the 'ilities'; model with wide applicability, e.g., scale up kinds of current modeling. ...Research is needed to learn how to capture non-functional requirements to the extent that the impact to proposed changes in a non-functional requirement can be predicted.

Methods for 'ilities'- driven engineering methods need to be developed. "A number of 'ilities', chief among them maintainability (or flexibility to change), need to be built-in to requirements as special items to be considered throughout the requirements engineering process." DoD must develop

> ...first a template to identify the non-functional requirements ... in order to keep them from falling through the cracks ... Tools and/or languages to evaluate and express these non-functional requirements are vital to this ever important requirements data base.

In their language section, Working Group Two proposed the need for research to "solve the problem of defining the so-called 'non-functional' requirements."

- Working Group Three proposed for DoD to

> explore formalisms to encourage completeness checking in many different areas such as ... non-functional requirements. These include the so-called 'ilities', such as maintainability, reliability, security, and performance. ... research how prototyping can support the validation of non-functional requirements ...

**3.5.6** **Requirements Trade-off Analysis.** Two working groups saw the need for additional work in requirements trade-off analysis.

- Working Group One recommended "DoD develop tools/techniques to capture merits/trade-offs among requirements."

- Working Group Two stated "tools and techniques are needed to capture, organize, and help evaluate the many trade-offs that occur in requirements development. Intelligent impact analysis is an example."

**3.5.7** **Requirements Traceability.** Additional research in requirements traceability was also suggested.

- Working Group Two proposed a "life cycle requirements database to capture and manage attributes of individual requirements and provide traceability".

- Working Group Three emphasized the need for research, stating, "requirements should be traceable through the prototype and back into the development of the objective system".

**3.5.8** **Multiple Stakeholder Issues.** Special emphasis was given to multiple stakeholder issues.

- Working Group One devoted an entire section of its report on the need to reach closure among multiple stakeholders.

- Working Group Three recommended the development of "formalized methods for communicating prototype results between the various stakeholder groups."

**3.5.9**    **Technology Application.**  Finally, and most obviously, the workshop concluded that it is not enough to merely research and develop technologies. DoD must constantly seek ways to apply those technical gains in the real world.

This page is intentionally left blank.

# 4

# BIBLIOGRAPHY

This page is intentionally left blank.

# 4 *BIBLIOGRAPHY*

To pursue your investigations into Requirements Engineering and Rapid Prototyping, we recommend that you consult the following works:

Alford, M.W. "A Requirements Engineering Methodology for Real-Time Processing Requirements," IEEE Transactions on Software Engineering 3,1 (January, 1977): 60-69.

Boehm, B. "A Spiral Model of Software Development and Enhancement," ACM Software Engineering Notes 11, 4:(August, 1986): 16-24. Reprinted in IEEE Computer 21,5 (May, 1988): 61-72.

Booch, G. "Object-Oriented Development," IEEE Transactions on Software Engineering 12,2 (February 1986): 211-221.

Charette, R. "Software Engineering Environments". McGraw-Hill, New York, 1986.

Davis, Alan M. "A Comparison of Techniques for the Specification of External Behavior of Systems," Communications of the ACM 31,9 (September, 1988): 1098-1115.

Davis, Alan M. "Software Requirements: Analysis and Specification", Prentice-Hall, Englewood Cliffs, NJ 1990. This book's comprehensive bibliography with thoughtful commentary is itself an invaluable resource.

Gehani, N. and McGettrick, A. eds. "Software Specification Techniques". Reading, Mass. 1986.

McMenamin, S. and Palmer J. "Essential Systems Analysis". Prentice-Hall, Englewood Cliffs, NJ, 1984.

Poston, R. "Preventing Software Requirements Specification Errors with IEEE 830", IEEE Software 2,1 (January 1985): 83-86.

Royce, Winston W. "Software Requirements Analysis: Sizing and Costing," in Practical Strategies for Developing Large-Scale Software, edited by E. Horowitz. Addison-Wesley. Reading, Mass. 1975, pp. 57-71.

Yeh, Raymond T. et al. "Software Requirements -- New Directions and Perspectives," in Handbook of Software Engineering, edited by C. Vick and C. Ramamoorthy. Van Nostrand Reinhold, New York, 1984, pp. 519-543.

Yeh, Raymond T. and Welch T. "Software Evolution: Forging a New Paradigm," in 1987 Proceedings of the ADM/IEEE Fall Joint Computer Conference, ACM Press of Association for Computing Machinery, 1987.

Yourdon, Ed. "Modern Structured Analysis". Yourdon Press. Englewood Cliffs, NJ. 1989.

Zave, P. "A Distributed Alternative to Finite State Machine Specifications," ACM Transactions on Programming Languages and Systems, 7,1 (January, 1985): 10-36.

# APPENDIX A

*Workshop Agenda*

This page is intentionally left blank.

# Workshop Agenda - Day One

**AM** 7:30 - 8:30    Registration/Executive Continental Breakfast

8:30 - 8:40    Administrative Remarks/Introduction of Speakers
Mr. George Sumrall
TTCP Workshop Chairperson
CECOM Center for Software Engineering, USA

8:40 - 9:10    Introduction
Mr. John H. Sintic
Acting Director
CECOM Center for Software Engineering, USA

CECOM Welcoming Remarks
Mr. Robert F. Giordano
Deputy PEO, Command and Control Systems, USA

TTCP Welcoming Remarks
Mr. Joseph Batz
United States National Leader and Chairperson, XTP-2

9:10 - 9:35    Technical Presentation 1
Mr. James Toher
Pembroke House, United Kingdom
"The Nature of Requirements"

9:35 - 10:00    Technical Presentation 2
Mr. Edward Schlosser
Lockheed Software Technology Center, USA
"The Role of Requirements in the System Development Process"

10:00 - 10:25    Technical Presentation 3
Dr. Scott P. Overmyer
Contel Technology Center, USA
"Overview of Rapid Prototyping Systems"

10:25 - 10:45    Break

10:45 - 11:10    Technical Presentation 4
Dr. Winston W. Royce
SoftwareFirst, USA
"The Requirements Development Process - Present and Future"

| | | |
|---|---|---|
| 11:10 - 11:35 | Technical Presentation 5 |
| | Dr. Alan M. Davis |
| | George Mason University, USA |
| | "Multiple Views of Requirements" |

11:10 - 11:35    Technical Presentation 5
Dr. Alan M. Davis
George Mason University, USA
"Multiple Views of Requirements"

11:35 - 12:00    Technical Presentation 6
Dr. Raymond T. Yeh
International Software Systems, USA
"Framework for the Requirements Process"

PM    12:00 -  1:30    Luncheon Buffet

1:30 -  1:40    Workshop Charge
Mr. George Sumrall
TTCP Workshop Chairperson

1:40 -  1:50    Working Group 1 Overview
Dr. Alan M. Davis
Working Group 1 Chairperson

1:50 -  2:00    Working Group 2 Overview
Dr. Raymond T. Yeh
Working Group 2 Chairperson

2:00 -  2:10    Working Group 3 Overview
Dr. Winston W. Royce
Working Group 3 Chairperson

2:10 -  5:30    Working Group Activities

5:30 -  6:00    Meeting - Working Group Chairpersons

7:00    Group Dinner

# Workshop Agenda - Day Two

AM   7:30 -  8:30     Executive Continental Breakfast

     8:30 -  8:55     Technical Presentation 7
                      Mr. Douglas A. White
                      Rome Air Development Center, USA
                      "Knowledge-Based Requirements Assistant"

     8:55 -  9:20     Technical Presentation 8
                      Mr. Michael Deutsch
                      Software Engineering Institute, USA
                      "Insights Into the Influence of Shared
                      User/Customer/Contractor Objectives on Project Success"

     9:20 -  9:45     Technical Presentation 9
                      Mr. Reed Little
                      Carnegie-Mellon University, USA
                      "The Serpent User Interface Management System"

     9:45 - 10:10     Technical Presentation 10
                      Dr. Robert C. Fink
                      Performance Resources Inc., USA
                      "Using Joint Application Design (JAD) Techniques to Accelerate
                      the Requirements Definition Process"

    10:10 - 10:30     Break

    10:30 - 10:55     Technical Presentation 11
                      Mr. Edward C. Comer
                      Software Productivity Solutions, USA
                      "Ada Box Structures for Object-Oriented Software Development"

    10:55 - 11:20     Technical Presentation 12
                      Mr. Martin Morel
                      Le Groupe CGI, Canada
                      "A Prototyping Methodology Applied to Tactical C2 Systems"

    11:20 - 11:45     Technical Presentation 13
                      Mr. William E. Rzepka
                      Rome Air Development Center, USA
                      "Requirements Engineering Testbed"

**PM**  11:45 - 1:15    Luncheon Buffet

1:15 - 5:30    Working Group Activities

5:30 - 6:00    Meeting - Working Group Chairpersons

5:30 - 9:00    Optional Working Group Activities

# Workshop Agenda - Day Three

<u>Thursday</u>        <u>November 16, 1989</u>

**AM**  7:30 -  8:30    Executive Continental Breakfast

     8:30 -  9:30    Working Group 1 Report

     9:30 - 10:30    Working Group 2 Report

    10:30 - 11:00    Break

    11:00 - 12:00    Working Group 3 Report

    12:00           Closing Remarks
                    Mr. George Sumrall
                    TCCP Workshop Chairperson

This page is intentionally left blank.

# APPENDIX B

*Attendee Directory*

This page is intentionally left blank.

# Attendee Directory Information

Dr. Stephen J. Andriole
George Mason University
Department of Information Systems and Systems Engineering
4400 University Drive
Fairfax, Virginia 22030
(703) 764-6751

Mr. David Bagley
CECOM Center for Software Engineering
ATTN: AMSEL-RD-SE-AST-SE
Fort Monmouth, New Jersey 07703-5000
(201) 532-2081

Mr. Joseph Batz
United States National Leader and Chairperson, XTP-2
Software Engineering / DOD - DDRE (R&AT) SCT
Software and Computer Technology
3E114 The Pentagon / OUSDA
Washington, District of Columbia 20301-3081    JBATZ @ AJPO.SEI.CMU.EDU
(202) 694-0212                                 AUTOVON #: 224-0201

Mr. Harlan Black
CECOM Center for Software Engineering
US Army HQ CECOM
ATTN: AMSEL-RD-SE-AST-SE                        FACSIMILE #: (201) 532-4129
Fort Monmouth, New Jersey 07703-5000           AMSEL-RD-SE-AST@CECOM-2.ARPA
(201) 544-2238                                  AUTOVON #: 992-2238

Mr. Philip Casey
US Army HQ TRADOC
ATTN: ATCD-CB (Casey)
Fort Monroe, Virginia 23651
(804) 727-3271

Mr. Robert N. Charette
ITABHI Corporation
9840 Main Street / Suite 201                    FACSIMILE #: (703) 352-1592
Fairfax, Virginia 22031
(703) 352-1566

Mr. Edward R. Comer
Software Productivity Solutions, Inc.
P.O. Box 361697
Melbourne, Florida 32936-1697          ECOMER @ AJPO.SEI.CMU.EDU
(407) 984-3370

Mr. Thomas P. Conrad
Naval Underwater System Center
Code 2211 / Building 1171              FACSIMILE #: (401) 841-4749
Newport, Rhode Island 02841-5047      TCONRAD @ NUSC.ADA.ARPA
(401) 841-3353                         AUTOVON #: 948-3353

Dr. Alan M. Davis, Working Group Chairperson
George Mason University
ISSE Department
4400 University Drive                  FACSIMILE #: (703) 323-2630
Fairfax, Virginia 22030-4444          ADAVIS @ GMUVAX.GMU.EDU
(703) 323-2792

Mr. Michael S. Deutsch
Hughes Aircraft Co.
c/o Software Engineering Institute
Carnegie-Mellon University            FACSIMILE #: (412) 268-5758
Pittsburg,Pennsylvania 15213          MSD@ SEI.CMU.EDU
(412) 268-7047

Mr. Robert C. Fink
Performance Resources, Inc.
5111 Leesburg Pike / Suite 301        FACSIMILE #: (703) 671-7522*
Falls Church, Virginia 22041
(703) 845-9600

Mr. Gary E. Fisher
NIST / NCSL
Technology Building Room B266
Gaithersburg, Maryland 20878          FISHER@SWE.NCSL.NIST.GOV

Mr. Harrison D. Fountain
US Naval Post Graduate School
Computer Science Department
Monterey, California 93943
(408) 646-2461

Dr. William Gilmore
International Software Systems, Inc.
9420 Research Blvd. / Suite 200       FACSIMILE #: (512) 338-5757
Austin, Texas 78759                   GILMORE @ISS.UUCP
(512) 338-5711

Dr. Cordell Green
Kestrel Institute
3260 Hillview Avenue
Palo Alto, California 94304          GREEN@KESTREL.EDU
(415) 493-6871

Ms. Margaret Hamilton
Hamilton Technologies, Inc.
17 Inman Street
Cambridge, Massachusetts 02139
(617) 492-0058

Mr. David R. Harris
Sanders Associates, Inc.
95 Canal Street, NCA1-2232
Nashua, New Hampshire 03061
(603) 885-9182

Mr. Donald C. Harris Jr.
Directorate of Combat Development
Tactical Software Division
Comdt, USAADASCH
ATTN: ATSA-CDS (Harris)
Ft. Bliss, Texas 79916-7050
(915) 568-2810/4431                  AUTOVON #: 978-2444

Mr. Robert L. Harris
US Air Force
WRDC/AAAF-3 Attn: R. Harris
Wright Patterson AirForce Base, Ohio 45433-6543    AUTOVON #: 785-3947
(513) 255-3947

Dr. Pei Hsia
University of Texas Arlington
Computer Science Engineering Department
P.O. Box 19015                       FACSIMILE #: (817) 273-2548
Arlington, Texas 76019               HSIA @ EVAX.ARL.UTEXAS.EDU
(817) 273-3735

Mr. James Huskins
US Naval Post Graduate School
Computer Science Department
Monterey, California 93943
(408) 646-2461

Dr. W. Lewis Johnson
University of Southern California / ISI
4676 Admiralty Way / Suite 937W
Marina del Rey, California 90291          JOHNSON @ ISI.EDU
(213) 822-1511

Mr. Jean-Claude Labbe
Defence Research Establishment, Valcartier
P.O. Box 8800
Courcelette, Quebec                       FACSIMILE #: (418) 844-4538
Canada, GOA IRO                           LABBE @ JUPITER.DREV.DND.CA
(418) 844-4346

Mr. Aaron Larson
Honeywell, Inc.
Systems and Research Center MN65-2100
3600 Technology Drive
Minneapolis, Minnesota 55418
(612) 782-7340

Mr. Reed Little
Software Engineering Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213            LITTLE @ SEI.CMU.EDU
(412) 268-5792

Mr. Michael J. Looney
AXC4, BLK3, ARE(PN)
Portsmouth.
P06 4AA, UK
0705 219999 Ext. 2330

Cpt Gary W. Manley
US Naval Post Graduate School
Computer Science Department
Spanagel Hall MS:52LQ
Monterey, California 93943                MANLEY @ CS.NPS.NAVY.MIL
(408) 544-2670

Mr. Walter Marks
CECOM Center for Software Engineering
ATTN: AMSEL-RD-SE-AST-SE                  AUTOVON #: 992-2146
Fort Monmouth, New Jersey 07703-5000
(201) 532-2146

Mr. Raymond Menell
CECOM Center for Software Engineering
ATTN: AMSEL-RD-SE-AST-SE
Fort Monmouth, New Jersey 07703-5000          AUTOVON#: 992-2343
(201) 532-2343                                FACSIMILE#: (201) 532-4129

Mr. Martin Morel
Le Groupe C.G.I
5300 Boulevard Des Galeries
Quebec, P.Q.                                  FACSIMILE #: (418) 623-4114
Canada G2K 2A2
(418) 623-0101

Dr. Peter Ng
New Jersey Institute of Technology
Department of Computer & Information Science
323 Dr. Martin Luther King Blvd.             FACSIMILE #: (201) 596-5777
Newark, New Jersey  07102                     NG-P@VIENNA.NJST.EDU
(201) 596-3387 / 3366

Mr. Scott P. Overmyer
Contel Technology Center
1500 Conference Center Drive
P.O. Box 10814                                OVERMYER @ CTC.CONTEL.CON
Chantilly, Virginia  22021-3808
(703) 818-4480

Mr. Mark A. Podracky
Digital Fantacies, Ltd
2230 Gallows Road / Suite 240
Dunn Loring, Virginia  22027
(703) 698-9455

Mr. Robert M. Poston
Programming Environments Inc.
4043 State Hwy 33                             FACSIMILE #: (201) 918-0113
Tinton Falls,  New Jersey  07753
(201) 918-0110

Mr. Glenn E. Racine
AIRMICS (US Army)
115 O'Keefe Building
Georgia Tech                                  FACSIMILE #: (404) 894-3142
Atlanta, Georgia  30332-0800                  RACINE @ AIRMICS.ARMY.MIL
(404) 894-3110

Dr. Winston W. Royce
SoftwareFirst
22534 Paul Revere Drive
Woodland Hills, CA 91364
(818) 887-1811

Mr. William Rzepka
Rome Air Development Center
RADC/COEE / Building 3
Griffiss Air Force Base, New York 13441-5700          RZEPKAW @ LONEX.RADC.AF.MIL
(315) 330-2762                                        AUTOVON #: 587-2762

Dr. Donaldine Samson
Weber State College
Ogden, Utah 84408-3804                                DSAMSON @ UTAH.CC.COM
(801) 626-7189

Mr. Edward H. Schlosser
Lockheed Software Technology Center
2100 East St. Elmo Road,
Bldg 30E, Organization 96-10
Austin, Texas 78744                                  SCHLOSSER @STC.LOCKHEED.COM
(512) 327-3672

Dr. Carl A. Singer
Bellcore
6 Corporate Place
Room PYA-1K282                                       BELLCORE @ PYUXE!SINGER
Piscataway, New Jersey 08854-4158                    FACSIMILE #: (201) 562-9305
(201) 699-8951

Dr. Victor Conrad Sobolewski
Government of Australia
1601 Massachusetts Avenue NW                         FACSIMILE #: (202) 797-3326
Washington, District of Columbia 20036               AUSDEF @ A.ISI.EDU
(202) 797-3378

Dr. Rolf Stachowitz
Lockheed
2100 E. St. Elmo Road
0/96-10, B/30E                                       FACSIMILE #: (512) 448-5728
Austin, Texas 78744                                  ROLF @ LOCKHEED.COM
(512) 448-5772

Mail: Dr. Rolf Stachowitz
c/o WW Royce
22534 Paul Revere Drive
Woodland Hills, Ca. 91364
(818) 887-1811

Mr. George Sumrall, Workshop Chairperson
CECOM Center for Software Engineering
ATTN: AMSEL-RD-SE-AST-SE (Sumrall)          FACSIMILE #: (201) 532-4129
Fort Monmouth, New Jersey 07703-5000          SUMRALL @ AJPO.SEI.CMU.EDU
(201) 532-2342                                AUTOVON #: 992-2342

Dr. Murat M. Tanik
Southern Methodist University
Computer Science Department
SMU, Dallas, Texas 75275
(214) 692-2854

Mr. James Toher
SD-SCICON
Pembroke House
Pembroke Broadway
Camberley, Surrey
GUI5 3XD, England

Mr. David H. Watjen
Advanced Technology, Inc.
2 Crystal Park
2121 Crystal Drive / Suite 200
Arlington, Virginia 22202
(703) 769-3000

Mr. Douglas A. White
US Air Force
Rome Air Development Center RADC/COES
Griffiss Air Force Base, New York 13438-5700      WHITE @ AIVAX.RADC.AF.MIL
(315) 330-3564                                     AUTOVON #: 587-3564

Dr. Martin I. Wolfe, Member XTP-2
CECOM Center for Software Engineeing
ATTN: AMSEL-RD-SE-AST (Wolfe)               FACSIMILE #: (201) 532-4129
Fort Monmouth, New Jersey 07703-5000          MWOLFE @ A.ISI.EDU
(201) 532-2423                                AUTOVON #: 992-2423

Dr. Raymond T. Yeh, Working Group Chairperson
International Software Systems, Inc.
9420 Research Blvd., Suite 200
Austin, Texas 78759
(512) 338-5700

FACSIMILE #: (512) 338-5757

# *APPENDIX C*

## *Letters from Chairpersons*

Note: The following letters were sent by each of the Working Group Chairpersons to the workshop participants in their respective Working Groups, prior to the workshop.

This page is intentionally left blank.

# For the Participants of Working Group 1
## From Dr. Alan M. Davis

## Statement of Goals

The term "process" in our title implies that we will be limiting our discussion to the activities, events and procedures that occur in the creation and evolution of system and software requirements. Given this immense charter and the vast combined experiences of the members of this working group, it is clear that we could probably attack any one requirements process-related topic and discuss it for seven (7) hours. However, our goals are to cover the full spectrum of the requirements process domain, not just to delve onto a set of specific topics. The *general* goal is easy: At the completion of the second day, every group member should have a clearer, more focused, view of all aspects of the requirements process. Here's a strawman set of *specific* technical goals that we want to achieve by the end of the second day of the workshop.

1.  Identify, clarify, and prioritize the issues relating to the requirements process. Note that this is a breadth-first analysis of the requirements process domain. We will be asking lots of questions, not necessarily answering them.

2.  What are the possible positions on each of the issues that we come up with? Note we need not agree to one position, but we do need to agree as to what the alternatives are.

3.  Enumerate efforts to date to resolve some of the issues. What have they shown? Are the results conclusive? What limitations do the results have?

4.  What additional work needs to be completed to resolve the issues?

5.  Debate and reach group consensus on one or more of the issues.

## Preliminary List of Issues

1.  *What are requirements and requirements engineering?* Do they include user needs analysis? Problem analysis? Description of the external behavior of the system to be built/procured? Definition of the system's constituent components? Do they end at the beginning of the design phase? How do they relate to the requirements changes that occur throughout the life cycle?

2.  *What are the relationships among system requirements, systems design, software requirements and the acquisition life-cycle?*

3.  *Is there such a thing as a "perfect" requirements process?* For all software? For any application area? For any particular effort? Must the process itself be flexible so that the process changes as new information is learned about the requirements themselves?

4.	*What are the constituent primitive elements that make up any requirements process?* Do such elements exist? If so, which are essential to any requirements process? Which are optional? What are the ways of combining them to form valid requirements process models? As an alternative, perhaps a better approach to defining all possible requirements process models is to first define all elements of the *product* of any requirements process.

5.	Recognizing that requirements engineering encompasses all aspects of the handling of requirements regardless of when they occur, *how does a requirements process interface with configuration management processes* that are designed to accommodate change (including requirements changes) during development? Are there other considerations to accommodate inevitable changes to requirements once the requirements are baselined? When should requirements be baselined?

6.	*What does it mean to validate requirements?* How can it be done? When should it be done?

## Suggested Reading Material

Yeh, Raymond, T., "Requirements Analysis - A Management Perspective," pp.410-416.

Davis, Alan, M., "A Taxonomy for the Early Stages of the Software Development Life Cycle," The Journal of Systems and Software, Vol. 8, No. 4, September, 1988, pp. 297-311.

Harel, David, "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming, Vol. 8, 1987, pp. 1-29.

# For the Participants of Working Group 2
## From Dr. Raymond T. Yeh

## A Brief Description of the Issues

Although much research work has been performed on requirements analysis, most published literature is concerned with tools, methods or notations, without asking to which extent they can be used in conjunction in order to support each other. I believe that an integrated perspective is necessary in order to attain the goal of this workshop. The following diagram provides major areas of concern in the requirement phase. The interrelationship between components forms the foundation for an integrated approach.
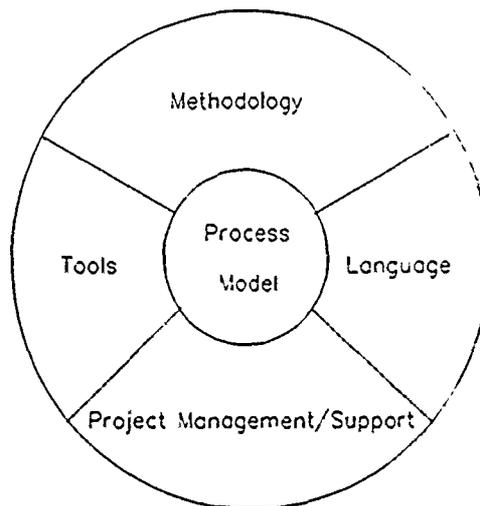


**Figure 1.** An Integrated View of Requirements Engineering.

The requirements analysis phase itself is split into a subphase concerned with studying the requirements of the complete system to be developed (hardware, software and organizational environment, functional and non-functional aspects), a subphase during which the boundary between hardware and software and organizational aspects of the new system is defined, and a set of potentially parallel subphases during which the particular hardware requirements, software requirements and organizational requirements are analyzed. Finally, requirements aspects to be best addressed during later phases of the life cycle need mentioning.

For each of the phases and subphases mentioned above, concrete objectives are set. Further, the following questions need to be answered:

- What are the particular steps taken and methods to be used during this subphase?

- What information is needed as input for this phase?

- What kind of analysis should be performed on this information to verify its truth?

- How should this information be documented?

- What are the exit criteria for each phase?

- What tools are needed to support this phase or what are the desirable properties of such tool?

I would like to see our group with three subgroups: methodology, language, and tools. The *methodology group* will be concerned with most of the questions raised above. For the *language group*, I suggest to look at the possibility of a common CORE for various requirements languages as shown in Figure 2. Is the CORE language a real language or simply a common schema, e.g., semantic net?
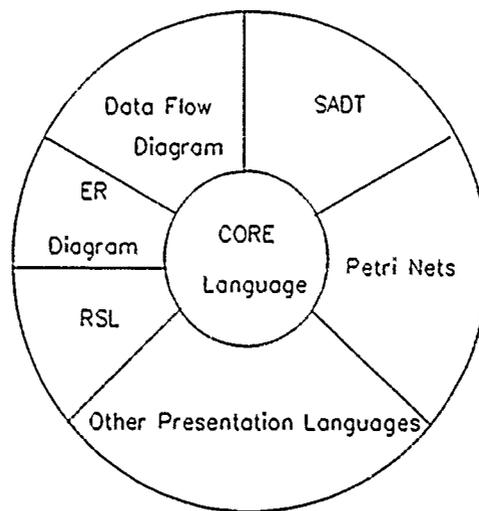


**Figure 2.** A System of Requirements Languages

For the *tools group*, I suggest looking at the integration issues. How can various tools be effectively integrated. Note that we have traditional tools as well as state-of-the-art tools. Clearly, this issue is very much linked with the language issue.

October 12, 1989

TO:      George Sumrall; All Working Group 3 Participants
FROM:    Win Royce
SUBJECT: Plan for Working Group 3, Technical Panel on
         Software Engineering, November 14 through November 16

Working Group 3 is assigned the task of analyzing prototyping and knowledge-based techniques as applied to requirements engineering.

We have. at most, two days to complete our task. To quickly focus on the issues and then resolve them. I am proposing the following approach. The working group will jointly construct eight to ten well-posed questions covering the most critical issues of our assigned subject. These questions. will be prioritized: and substantially more time will be allocated to analysis of the higher priority questions. Each question will be analyzed in two succeeding sessions. The first session will brainstorm the questions attempting to capture all ideas (even conflicting ones) that might be of value. The second session will aim at winnowing down these raw, possibly conflicting ideas to a shorter, consensus-achieving set with associated features. benefits. and actions. A third session is scheduled to complete our paperwork and a fourth session to report out our findings.

The four working group sessions are organized as follows:

<u>Session I:</u>     A 3-1/2 hour long planning and brainstorming session to answer 8 to 10 questions at an average rate of 15 minutes per question.

<u>Session II:</u>     A 4 hour long concentration on word-smithing sharply-honed answers to the questions based on benefits, features, and actions. The average time for each answer-creating response will be less than 20 minutes per question.

<u>Session III:</u>     A 2 hour long session to write up our findings. At least one-half of our written inputs will have already been done in realtime during sessions I and II.

<u>Session IV:</u>     A 1 hour long briefing on our findings by a working group spokesperson to the assembled participants from all working groups.

Succeeding sections of this plan include:
  -a listing of 14 potential questions
  -detailed instructions, agendas, and schedules for
     sessions I, II, and III
  -instructions for preparing the working group
     summary document

The 14 potential questions listed in the next sections are intended to stimulate the pre-workshop thinking of the Working Group III participants. Each participant ought to review the potential questions included here, reword them to be more sharply-put, or invent their own questions for consideration and bring them to Session I in a form ready to distribute to the other participants. The first item of business in Session I will be to select a set of questions and prioritize them. This selected set of prioritized questions will become the principal mechanism which organizes, focuses and otherwise guides all further deliberations of our working group. Selecting the right set of questions is important. (If no participant acts, the questions included in the following section will serve as the default set to guide us.)

Why are we devoting 66 hours of our professional lives to prototyping, and knowledge-based approaches to requirements engineering? Because it it important! The accompanying figure proposes one reason as to why our work is important: If there are other reasons they ought to be uncovered during the critical nine hours of our joint deliberations. The 8 to 10 questions which we will choose to concentrate on are best answered if we also understand why we are asking them.

Keep in mind that each participant will have no more than two minutes per question, per session, to make his point. We must all be prepared, focused, consensus-oriented and especially articulate (and fast writers-too) if we are going to complete our assigned task.

See you in November!


Win Royce

Q.    WHAT IS THE PRIMARY ISSUE DRIVING REQUIREMENTS ENGINEERING?

A.    SOFTWARE COMPETITIVENESS

- TACKLING HIGH TECHNOLOGY SYSTEMS WITH CONFIDENCE

- MOTIVATING PEOPLE TO BUILD HIGHER QUALITY SYSTEMS

- REDUCING COSTS; QUICKENING SCHEDULES

- BUILDING A "WE HAVE SOLVED THE SOFTWARE CRISIS" IMAGE
  FOR OUR INDUSTRY

Figure 1:    OUR TASK IS IMPORTANT!

SoftwareFirst - 1989

C-10

## Questions for Prototyping

-What qualities must a prototyping system have?  What problems must it solve? In the short term?  In the longer term?

-What are the best current examples of prototyping systems?

-How does the software development process have to be constructed to exploit prototyping ?

-Should major software acquisition agencies (e.g., governments) mandate prototyping ?

-How does the user and the acquisition agency interact with the prototyping system during development?

-Does the construction of prototyping systems have especially difficult development problems?  What are they?  Should the research community be stimulated to help?

-Are prototyping systems going to be easy to use?  Is special training required? Are there technology transfer problems?

# Knowledge Based Approaches (KBA's) to
# Requirements Engineering

-What kinds of requirements engineering problems are best solved by KBA's? In the short term? In the longer term?

-Are the underlying abstractions of KBA's too difficult for widespread usage? Is special training required?

-What kind of language syntax and semantics are needed? Can we get it into Ada and C?

-Can formal methods a la theorem proving be introduced into widespread practice?

-Can we achieve automatic document writing for producing acquisition agency deliverables?

-Can KBA's cause multi-skilled software development teams to work together more productively?

-How should the software development process change, particularly the up-front requirements engineering tasks, to exploit KBA's, theorem proving, and automatic document generation?

## Session 1: Tuesday 2:00-5:30

The first one-half hour will be concentrated on getting organized plus reviewing the schedule. The following selections and assignments will be made.

(1)    Eight to ten questions will be selected and prioritized from Q1 (highest) to Q10 (lowest).

(2)    A question-leader will be assigned to each question. The principal role of each question leader is to stand up and lead the discussion for their assigned question.

(3)    A back-up to the question-leader will be assigned for each question. The principal role of each back-up is to act as a scribe to capture the discussion content.

The schedule for Session 1 is as follows:

| Getting Organized | 30 minutes | 2:00-2:30 |
|---|---|---|
| -Agenda Discussion | | |
| -Question Selection | | |
| -Question leader, Back-up Assignment | | |

| Brainstorming | | |
|---|---|---|
| Q1 | 20 minutes | 2:30-2:50 |
| Q2 | 20 minutes | 2:50-3:10 |
| Q3 | 15 minutes | 3:10-3:25 |

| Break | 10 minutes | 3:25-3:35 |
|---|---|---|

| Brainstorming | | |
|---|---|---|
| Q4 | 10 minutes | 3:35-3:45 |
| Q5 | 10 minutes | 3:45-3:55 |
| Q6 | 20 minutes | 3:55-4:15 |
| Q7 | 20 minutes | 4:15-4:35 |

| Break | 5 minutes | 4:35-4:40 |
|---|---|---|
| Brainstorming | | |
| Q8 | 15 minutes | 4:40-4:55 |
| Q9 | 15 minutes | 4:55-5:10 |
| Q10 | 10 minutes | 5:10-5:20 |
| Reclama any Question | 10 minutes | 5:20-5:30 |

During Session I or immediately following Session I each question-leader and their back-up will prepare one or two vugraphs summarizing the content of each brainstorming response to the questions. These vugraphs will be needed for Session II and the final report.

## <u>Session II: Wednesday 1:30-5:30</u>

Each question-leader with the help of his back-up, will have prepared one
or two vugraphs summarizing the most interesting previous day's
brainstorms. The assigned question-leader and back-up for Session I will
exchange roles for Session II.

As in Session I each question will be addressed one at a time. The goal of
this second pass is to sharpen the focus on each question and to list
recommended actions as though we were omniscient and all-powerful.
Our answers to each question should take the form of:

|  |  |  |
|---|---|---|
| What? | i.e. | Features |
| Why? | i.e. | Benefits |
| How? | i.e. | Actions |

The schedule for Session II is as follows:

**Benefits, Features, Actions**

| | | |
|---|---|---|
| Q1 | 25 minutes | 1:30-1:55 |
| Q2 | 25 minutes | 1:55-2:20 |
| Q3 | 20 minutes | 2:20-2:40 |
| Q4 | 15 minutes | 2:40-2:55 |

| | | |
|---|---|---|
| Break | 10 minutes | 2:55-3:05 |

**Benefits, Features, Actions.**

| | | |
|---|---|---|
| Q5 | 10 minutes | 3:05-3:15 |
| Q6 | 25 minutes | 3:15-3:40 |
| Q7 | 25 minutes | 3:40-4:05 |
| Q8 | 20 minutes | 4:05-4:25 |

| | | |
|---|---|---|
| Break | 5 minutes | 4:25-4:30 |

**Benefits, Features, Actions**

| | | |
|---|---|---|
| Q9 | 15 minutes | 4:30-4:45 |
| Q10 | 15 minutes | 4:45-5:00 |

SoftwareFirst - 1988

| | | |
|---|---|---|
| Reclama any Qestion | 20 minutes | 5:00-5:20 |
| Writing assignments and redrafting document outline | 10 minutes | 5:20-5:30 |

During Session II or soon after the question-leader and their back-up will prepare one or two vugraphs summarizing the answers in a features; benefits, actions format.

## Session III: Wednesday, 7:00-9:00

A third session in the evening will be required to complete our write-ups. During the last ten minutes of Session II writing assignments will have been made.

The primary task will be for each queston-leader and back-up to write facing page text to the two to four vugraphs created in Sessions I and II. Each participant can expect to be involved with writing-up two questions plus writing-up one more brief section.

The tentative outline for our Working Group 3 document is as follows:

### Document Outline

1.      Working Group 3 Format
            -working group methodology
            -setting
            -participants

2.      Prototyping and knowlege-based approaches
            for requirements engineering:
            Problem Statement

3. _    Summary

       3.1      Short Term Technical Prospects
       3.2      Longer Term Technical Prospects
       3.3      Changes in the Software Development Process
                    Model
       3.4      Technical Transfer Prospects
       3.5      Supporting Research
       3.6      Special Problems

4.0    Question Summary

4.1    Q1 - vugraphs plus facing page text
4.2    Q2

.

.

.

4.10    Q10

5.    Appendix Material


This document outline will be redrafted, if necessary, at the end of Session II.

# *APPENDIX D*

## *Technical Presentation Vu-Graphs*

This page is intentionally left blank.

# TECHNICAL PRESENTATION VU-GRAPH REFERENCE

This page is intentionally left blank.

# The Technical Cooperation Program

## TTCP Welcoming Remarks

### Mr. Joseph Batz
### United States National Leader and Chairperson

# THE TECHNICAL COOPERATION PROGRAM



MEMBER NATIONS
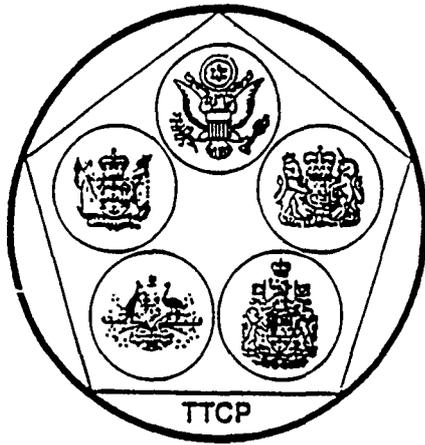
- AUSTRALIA
- CANADA
- NEW ZEALAND
- UNITED KINGDOM
- UNITED STATES

THE
TECHNICAL
COOPERATION
PROGRAM

# FUNCTION

PROVIDE MECHANISMS FOR:

- Science & Technology Information Exchange
- Collaborative Research & Development
- Scientific Personnel Exchange
- Science & Technology Materiel Exchange

QUID PRO QUO

GOVERNMENT TO GOVERNMENT

DEFENSE LIMITED

# JOINT DECLARATION
## U.S. President & British Prime Minister

### Oct. 25, 1957

"The arrangement which the nations of the free world have
made for collective defense and mutual help are based on
the recognition that the concept of national self-
sufficiency is now out of date.  The countries of the free
world are interdependent and only in genuine partnership,
by combining their resources and sharing tasks in many
fields, can progress and safety be found.  For our part we
have agreed that our two countries will henceforth act in
accordance with this principle."

- TRIPARTITE TECHNICAL COOPERATION PROGRAM
  Canada joined   U.S. & U.K.   immediately

- THE TECHNICAL COOPERATION PROGRAM
  Australia      -  July 1965
  New Zealand -  October 1969

# TTCP AIMS

- PROVIDE KNOWLEDGE & INFORMATION ON
  EACH OTHERS PROGRAMS

- AVOID UNNECESSARY DUPLICATION
  AMONG PARTICIPANTS

- PROMOTE CONCERTED JOINT EFFORTS
  TO CLOSE GAPS

ENCOMPASSING

- BASIC RESEARCH

- EXPLORATORY DEVELOPMENT

- DEMOS OF ADVANCED TECH DEVELOPMENT

## SUBGROUPS

- Chemical Defense
- Aeronautics Technology
- Radar Technology
- Electronic Warfare
- Behavioral Sciences

- Undersea Warfare
- Infrared & ElectroOptical Technology
- Materials
- Communications Technology & Information Systems
- Conventional Weapons Technology

- COMPUTING TECHNOLOGY

PANELS;   ACTION GROUPS;   TECH LIAISON GROUPS

---

THE
TECHNICAL
COOPERATION
PROGRAM

# COMPUTING TECHNOLOGIES
# SUBGROUP X (SGX)

## TECHNICAL PANELS

XTP1   -   Trustworthy Computing

XTP2   -   Software Engineering

XTP3   -   Architectures

XTP4   -   Machine Intelligence

## ACTION GROUPS

XAG2   -   Digital Design

XAG3   -   Image Information

PURPOSE: To improve the utilization of the collective resources and capacities of the member countries in the areas of software engineering and software technology.

SCOPE: The creation and life-cycle support of software for military applications.

Includes: PROCESSES, METHODS, TOOLS for

| | | |
|---|---|---|
| DEFINITION | SPECIFICATION | PROTOTYPING |
| DESIGN | INTEGRATION | TEST |
| EVALUATION | PORTING | REUSE |

DATABASE TECHNOLOGY

THE
TECHNICAL
COOPERATION
PROGRAM

# XTP2 - WORKSHOPS

- REAL TIME SYSTEMS AND ADA
  Conducted June 1988, at IDA, Washington DC.
  Approx. 40 participants.

- REQUIREMENTS ENGINEERING/RAPID PROTOTYPING
  Planned for November 1989, at Fort Monmouth (Eatontown), NJ.

- SOFTWARE METRICS
  Planned in 1990.

THE
TECHNICAL
COOPERATION
PROGRAM

TTCP
## XTP2 TASK

AIMS

- To examine the current state of methods and tools used for requirements engineering

- To identify their deficiencies

- To recommend new or improved methods and tools that need to be developed

# Technical Presentation 1

## "The Nature of Requirements"

**Mr. James Toher**
**Pembroke House, United Kingdom**

# *Overview*

○ **Assignments**

    *Consultancy, Training & R&D*
    *Large & Complex Systems*
    *Industry, Military & Govt*

○ *Issues*

    *Non-Functional Requirements*
    *Validation*
    *Politics*

# *Requirements*

○ *Functional + 'Non-Functional'*

○ *All Interact*

○ *NF Dominates F*

## Functional Requirements

- The rate of decelaration will be calculated, displayed to the driver who will compare it with the reference speed.

- On supply of retailer identification the authorisation number will be derived.

- ForAll e memberof(HCL)
    Exists r memberof(Verf-Rec)
    and r = PF(e)

- Automatic dialling of previously stored numbers by pressing a single key

## Non-Functional

| | | |
|---|---|---|
| Reliabilty | Materiality | Criticality |
| Safety | Security | Vulnerability |
| Performance | Risk | Repairability |
| Timing | Accuracy | Timeliness |
| Survivability | Confusion | Confidentiality |
| Maintainability | Cost | Tolerance |
| Transportability | Precision | Capacity |
| Speed | Ownership | Manning |
| Quality of Service | Interoperability | Traceability |
| Size | Usability | Latency |
| Media | Compatibility | Currency |

## Non-Functional Requirements

○ *After the Final Agreement and before 11.00 a.m the Clearing Totals must be entered on the Daily Settlement Sheet.   (Timing)*

○ *The application must cope with 50 million different air fares.    (Capacity)*

○ *When Central Control fails Local Control must order signals (no priority)      (Survivability)*

○ *Authorisation must be available 100% between Mon-Fri (inc) during hours 8.00 a.m-5.00p.m. (Availability)*

○ *Billing must conform to level H2 with category D3 (External: collusive/manipulative). (Vulnerability to Fraud)*

## Interactions

○ *Limit functions available to alleviate capacity overload and therefore degradation of performance.  NF -> F*

○ *Increase in services available increase confusion of driver and decreases safety F -> NF*

○ *Increase in security encourages more usage and increases congestion. NF -> NF*

# Problems

○ *NF addressed too late (if at all)*

    *Hidden Complexity*

    *Loss of Control*

○ *Methods*

    *First Class Treatment of NF*

    *True Systems Methods*

○ *Prototyping*

    *Exhibiting NF Properties*

    *Reasoning about Interactions*

---

# 'Correct' Requirements

○ *Validation Principle & Guarantor*

○ *Principle*

    *Output - Outcome*

    *Behaviour - Effect*

○ *Guarantor*

    *Many Stakeholders*

○ *Validation Statements*

    *Proof - Weak Inference*

## Principle

    ○    *Signed off and accepted*
    *In use for several years*
    *Happy user*
    *Not failed yet*

    ○    *The system is always the servant*
    *of the 'business' and its needs*

    ○    *Is it Effective w.r.t its Guarantor*

---

## Behaviour -> Effect

**AREA TRAFFIC CONTROL**

*improve road safety*
*reduce environmental degradation*
*assist public services*
*provide information to road-users and other systems*
*provide economic benefits to the community as a whole*

**ELECTRONIC POINT OF SALE**

*increase throughput*
*guaranteed pricing*
*extra sales floor area*
*improved token handling*
*reduce fraud*
*reduce central cash administartion*

**AUTOMATED TICKET BARRIERS**

*reduce fraud*
*improve traffic information*
*reduce staff costs*
*permit flexible price structure*

## Guarantor

| | |
|---|---|
| *User* | *General Public* |
| *Customer* | *Suppliers* |
| *Sponsor* | *Beneficiaries* |
| *Maintainers* | *Victims* |
| *Employees* | *Managers* |
| *Operators* | *Regulators* |

## Problems

○ *Legitamacy*

○ *Credibility*

○ *Methods*

   *Behaviour/Outcome*

○ *Prototyping*

   *Demonstration*
   *Universal Generalisations*

# Politics & Culture

○ *Meta-Systems*

○ *Every Situation*

   *Three systems are present*

○ *Every System*

   *Changes the structures*

○ *Every Problem -- Solution*

   *Requires the three elements*

---

## Every Situation

○   *Production Systems*

○   *Belief Systems*

○   *Political Systems*

## Every System .

- ○ Can have a long timeframe
  Affects most or all of the organisation
  Involves substantial resources
  Has the potential to lead to major changes.
  Has winners and losers

- ○ Influences the political and cultural systems

## Cultural Effects

- ○ Increased alienation

- ○ Changes in status

- ○ Social isolation

- ○ Challenge to values

## Political Effects

- Shifts in balance of power
- Job losses
- Span of control shifts
- Shifting problems/threats
- Intensity of work/machine pacing
- Polarisation

- Heirarchies may change
- Working relationships my be strained
- Systems of grading, promotion, reward may become redundant
- Demarcation issues may alter
- Threats to confidentiality
- Heirarchies may change

## Every Problem -- Solution

- Requires an understanding of the three elements

Solution

|  | P | C | T |
|---|---|---|---|
| P | ↘ |  | ↘ |
| C |  | ↘ |  |
| T |  |  | ↘ |

Problem
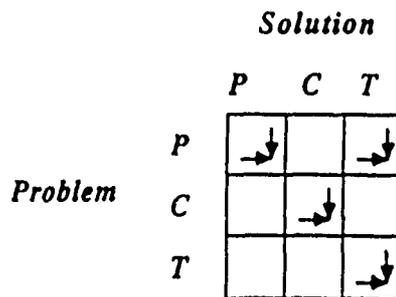
*Technical Presentation 2*

*"The Role of Requirements
in the System Development Process"*

*Mr. Edward H. Schlosser
Lockheed Software Technology Center, USA*

# Old Approach: Allocate System Requirements Between Hardware and Software Up Front
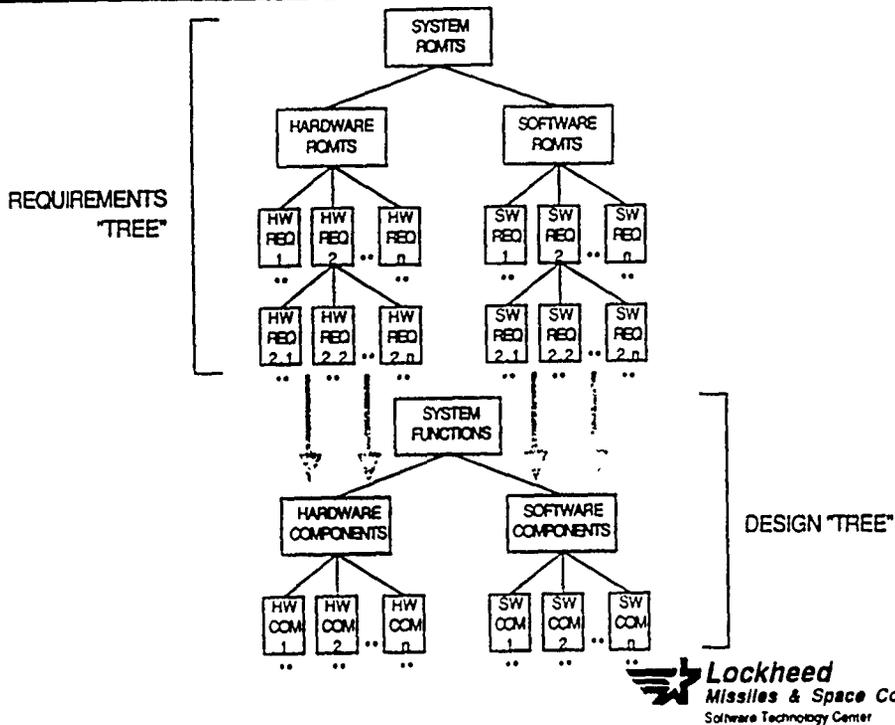


Requirements "Tree" diagram showing SYSTEM RQMTS branching into HARDWARE RQMTS and SOFTWARE RQMTS, each with HW REQ 1, HW REQ 2 ... HW REQ n and SW REQ 1, SW REQ 2 ... SW REQ n, then HW REQ 2.1, HW REQ 2.2 ... HW REQ 2.n and SW REQ 2.1, SW REQ 2.2 ... SW REQ 2.n. Design "Tree" showing SYSTEM FUNCTIONS branching into HARDWARE COMPONENTS and SOFTWARE COMPONENTS, with HW COM 1, HW COM 2 ... HW COM n and SW COM 1, SW COM 2 ... SW COM n.

**Lockheed**
*Missiles & Space Company, Inc.*
Software Technology Center        12383

---

# Don't Break Out System Requirements into Hardware and Software Requirements Up Front

## Why?

> We lack detailed knowledge up front to make good decisions about allocating requirements to hardware or software.

> A reasonable allocation to hardware or software may become inappropriate later due to changes in needs & available technology.

> All-or-nothing allocation of a requirement to hardware or software is often unrealistic.

> All-or-nothing allocation may limit or prevent exploitation of complementary hardware and software capabilities.

**Lockheed**
*Missiles & Space Company, Inc.*
Software Technology Center        12384

# Hardware/Software Differences Are Complementary

| | |
|---|---|
| Hardware wears out & breaks. | Software does not wear out or break. |
| Hardware gets out of adjustment or calibration. | Software does not get out of adjustment or calibration. |
| Hardware runs fast. | Software runs slow. |
| Manufacturability of hardware is limited by its complexity and by the laws of physics & chemistry. | Reproducability of software is not significantly limited by its complexity or by the laws of physics & chemistry. |
| Hardware is often difficult to install & configure. | Software can be made largely self-installing & self-configuring. |
| Retrofitting & upgrading hardware is often difficult. | Retrofitting & upgrading software can be highly automated. |
| User assistance and training cannot be built into hardware. | User assistance & training can be built into software. |

**Lockheed**
*Missiles & Space Company, Inc.*
Software Technology Center          12385

# Hardware/Software Cost Differences Are Complementary

| | |
|---|---|
| Developing the first copy of hardware is costly. | Developing the first copy of software is also costly. |
| Hardware is difficult and costly to manufacture to precise tolerances. | Software is easy and cheap to reproduce with precise digital fidelity. |
| Developing special tooling and processes to manufacture hardware is costly. | Standard tooling and processes can be used to replicate software. |
| Hardware design changes often require costly changes in tooling & manufacturing processes. | Software design changes usually do not require changes in tooling and processes used to replicate software. |
| It is difficult & costly to make hardware self-diagnosing. | It is easier & less costly to make software self-diagnosing. |
| The large costs of tooling for hardware manufacturing have encouraged application independence, standardized interfaces & reuse. | The minimal costs of tooling for software replication have discouraged application independence, standardized interfaces & reuse. |

**Lockheed**
*Missiles & Space Company, Inc.*
Software Technology Center          12386

# Benefits from Exploiting Complementary Characteristics of Hardware and Software

Components that exploit the complementary capabilities of hardware and software internally can provide greater capabilities at less cost than all-hardware or all-software components. Such mixed hardware/software components should have the following desirable properties:

- Early warning of failure
- Self-adjusting & self-calibrating
- Both fast & customizable
- Self-installing & self-configuring
- Self-checking & self-diagnosing
- Automated support for retrofitting

- Built-in user assistance & training
- Less costly initial tooling
- Less costly retooling as component is improved
- Fewer & less costly repairs
- Improved standardization & reuse

*Lockheed*
*Missiles & Space Company, Inc.*
Software Technology Center          12387

# Don't Break Out System Requirements into Hardware and Software Requirements Up Front

## Why?

We lack detailed knowledge up front to make good decisions about allocating requirements to hardware or software.

A reasonable allocation to hardware or software may become inappropriate later due to changes in needs & available technology.

All-or-nothing allocation of a requirement to hardware or software is often unrealistic.

All-or-nothing allocation may limit or prevent exploitation of complementary hardware and software capabilities.

## What should we do?

Defer allocating requirements to hardware or software until lower levels of design when we know more.

Allocate requirements up front to system components likely to contain both hardware & software.

Encapsulate allocations within low-level system components so they can be changed without "rippling."

Allocate functions which support the requirement, some to hardware and some to software, as appropriate.

Share responsibility for a low-level function between hardware & software when they are complementary.

*Lockheed*
*Missiles & Space Company, Inc.*
Software Technology Center          12388

# New Approach: Allocate Functions Between
# Hardware and Software at Lower Levels of Design



REQUIREMENTS "TREE"          DESIGN "TREE"
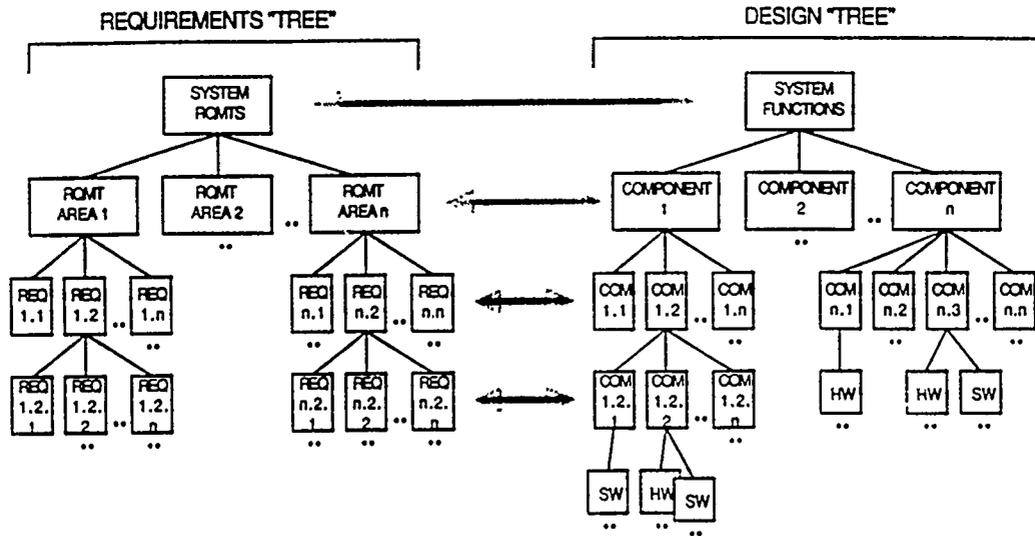
# Benefits from the New Approach

- Minimize risk of bad hardware/software allocation due to limited information

- Minimize "ripple" effect of changes in requirements and technology

- Avoid arbitrary "all-or-nothing" allocation to hardware or software

- Exploit complementary capabilities of hardware and software

This page is intentionally left blank.

*Technical Presentation 3*

*"Overview of Rapid Prototyping Systems"*

*Mr. Scott P. Overmyer*
*Contel Technology Center, USA*

# A Rapid Prototyping Tool is . . .

A tool, or set of tools which allow user-computer interface designers to QUICKLY and INEXPENSIVELY construct a high fidelity simulation of an interactive system. To be effective, a rapid prototype must not only convey the look, but also the feel of a proposed system design to users, customers, and developers.

RAPI002

# Goals of Rapid Prototyping

• Determine user requirements

• Communicate the design

• Exercise the design

• Collect human and system performance data

• Evaluate the design

• Market the design

RAPI003

# Tasks in Rapid Prototyping

- Design and develop screen layouts

- Select, design and develop dialog method

- Implement applications (in some form)

- Link screens, applications and dialog

- Make rapid iterations on simulation

- Collect and analyse user and system performance data

RAPID04

# Products of Rapid Prototyping

- "Live" user requirements specification

- Human-computer interface design
    - Dialog concept
    - H-C task allocation
    - I/O control concept
    - System and user response time requirements

- CDRL figures (screen print/plots)

- Quantitative and qualitative requirements validation criteria

RAPID05

D-25

# Transition to the Operational System

- **Throwaway rapid prototyping**
  - Brief final version of prototype and specs.
  - Deliver prototype and specs to developers
  - Monitor code and unit test
  - Compare operation of prototype to that of operational modules during integration and test

- **Evolutionary prototyping**
  - Generate user-interface management software from prototyping tool -or- Use prototype code
  - Integrate application modules
  - Make it all work together (e.g., compile and run)

RAPID06

# General Rapid Prototyping Tool Req'ts

- Foster **RAPID** prototype development
  - Coding is usually **too slow**

- Allow non-programmers to learn and use

- Allow end-user interaction
  - Pictures alone do not provide "feel"

- Allow integration of external applications

- Provide automated system and user performance data collection

- Help with generation of CDRLs
  - ICD's, HEDAD-O, HEPP, HEPR

RAPID07

# Specific Rapid Prototyping Tool Req'ts

## Screen Development

- Alphanumerics (text) display

- Graphics display
    - drawing/painting package

- Cursor or command-oriented screen construction

- Windowing
    - tiled (e.g., standard viewports)
    - overlapping (e.g., X Windows)

- "Object" creation/definition

RAP1008

# Specific Rapid Prototyping Tool Req'ts
## Dialogue Development

- Menus
    - Static, dynamic
    - Pull-down, pop-up, slug

- Forms
    - Tab back and forth between fields
    - Range and value checking for fields

- Command language (string parsing)

- Icons (direct manipulation)
    - Objects, graphics, sliders, buttons, dials, knobs

- Voice I/O

RAP1009

# Specific Rapid Prototyping Tool Req'ts
## Hardware and Device Support

- Input device handling
  - Cursor control
    - mouse, tablet, cursor keys, joystick, trackball
  - Voice
  - Gesture, eye motion

- Output device handling
  - Monochromatic displays .
  - Color displays
  - High and low resolution displays
  - Auditory displays
    - tone generation
    - voice synthesis
  - Virtual environment displays (e.g., Eyephones®)

RAPID10

# Specific Rapid Prototyping Tool Req'ts
## Database Capability

- Forms processing
  - Data entry
  - Data retrieval

- String storage
  - Command
  - Value (variable)
  - Current state

- Help
  - Context dependent
  - Context free

- General data retrieval

RAPID11

# Specific Rapid Prototyping Tool Req'ts
## Integrated Application Support (for C3I prototyping)

- Geographic projection and display
  - Vector, raster, video
  - Geographic overlay capability
    - Line and symbol display and manipulation
  - Lat/Long-based calculations
    - course, distance, trajectory
    - zoom and pan
    - satellite ground trace and/or orbit

- Graphs & plots

- Time-based simulation

- Image display & manipulation

RAPID12

# Specific Rapid Prototyping Tool Req'ts
## Display and Dialogue Linkage

- State transition based linkage
  - Link menu options to actions or "applications"
  - Link objects to actions
  - Link menu options or objects to displays
  - Link time or events to actions

- Command parsing and linkage to actions

- Sequence execution

- Possible code generation, if available

RAPID13

# Specific Rapid Prototyping Tool Req'ts

## Automated Data Collection

- Keystroke recording and timestamp

- Error data
  - Error type
  - Error frequency

- Task/thread data

- User comments

- Sequence recording and playback

- Configuration management and iteration control

RAPID14

# My Current Toolbox

- Skylights GX
  - IBM PC or compatible
  - VGA graphics
  - Elographics touch screen
  - Dragon Systems Dragonwriter 1000 VR Board
  - Microsoft Bus Mouse

- Dan Bricklin's Demo II
  - IBM PC or compatible
  - Color, but alphanumeric

- TAE Plus
  - UNIX (SUN 3/160)
  - X Windows-based
  - High-res color graphics

RAPID15

# My Current Toolbox - Part 2

- VideoWorks Interactive
  - Macintosh SE or Macintosh II
  - High-res color graphics

- Hypercard
  - Macintosh SE
  - Monochrome

- SuperCard
  - Macintosh SE or Macintosh II
  - Hypercard compatible
  - Color
  - Full-screen capabilities

- Various & sundry programming languages
  - C, PASCAL, (even ADA)

RAPID16

**CONTEL.** Technology Center

# Tool Features Matrix

| | Skylights GX | DB Demo II | TAE Plus 4.0 | VW Interactive | Hypercard | Supercard |
|---|---|---|---|---|---|---|
| Graphics | X | | X | X | X | X |
| Windowing | X | LTD | X | | X | X |
| Object Definition | | | LTD | | LTD | LTD |
| Menus | X | X | X | X | X | X |
| Forms | | | X | X | X | X |
| Command Parsing | | | X | LTD | | |
| Icons and Symbols | X | | | | | |
| Color | X | X | X | X | | X |
| DBMS | | | LTD | LTD | LTD | LTD |
| Applications | | | DRAW | DRAW | DRAW | DRAW |
| User Interactive | X | | X | X | X | X |
| System Generation | X | | X | X | X | X |
| Data Collection | | | | | | |

RAPIO17

D-31

# Editorial and Summary

- To perform effective RAPID prototyping:
  - Must be able to build <u>and</u> modify quickly
  - Tool kit is essential
  - Must present <u>both</u> look and feel

- Rapid prototyping is not a panacea

- Throwaway prototyping is worth doing
  - Validated requirements
  - Human engineered user-computer interface

- The "right" rapid prototyping tool has not yet been built
  - A multiple tool toolkit is best bet
  - New tool development **may** be money well spent
  - Acquire existing tool, and add on (good strategy)

RAPID18

## Technical Presentation 4


## "A Possible View of Requirements Engineering"



### Dr. Winston W. Royce
### SoftwareFirst, USA

## A POSSIBLE FUTURE VIEW OF REQUIREMENTS ENGINEERING

- Life Cycle Process

- Requirements Engineering Phase

- The Production Artifacts Problem

- The Manager vs. Software Designer Problem

- The Communications Problem

## LIFE CYCLE PROCESS

## LIFE CYCLE PROCESS

```
┌──────────┐      ┌──────────┐      ┌──────────────┐
│          │─────▶│          │─────▶│ Operational  │
│          │      │          │      │ Validation   │
└──────────┘      └──────────┘      └──────────────┘
                                      (≈ 1 year)
```
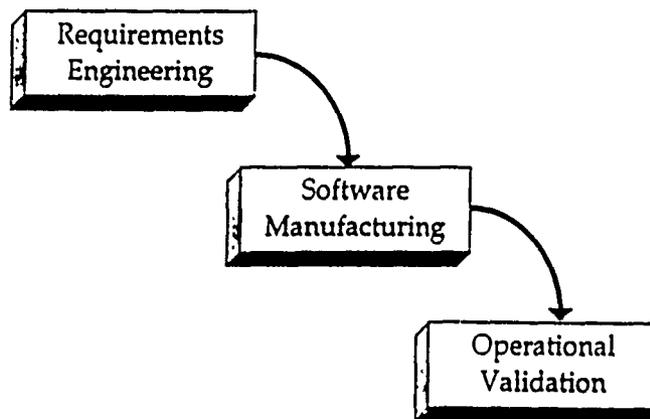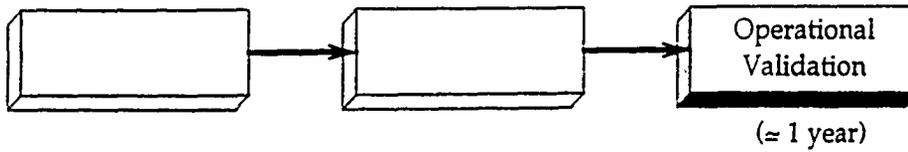
- Under Control Of Using (And Logistics) Command

- Validation Of Products

- User Achieves Confidence As Though They Built It

- Continuous (Small Scale) Change Process

## LIFE CYCLE PROCESS

```
┌──────────┐      ┌──────────────┐      ┌──────────────┐
│          │─────▶│   Software   │─────▶│ Operational  │
│          │      │ Manufacturing│      │ Validation   │
└──────────┘      └──────────────┘      └──────────────┘
                   (≈ 3/4 year)          (≈ 1 year)
```
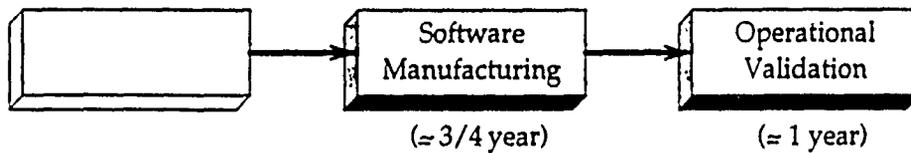
- Temporary Requirements Freeze
- Selection Of Computer Hardware
- Optimization Of Performance; Use Of Efficient Procedural Language
- Concern For Correctness
- Very Short Schedule
- Fixed Price; Warrantied; Possibly Competitive
- Modest Up-front Investment For Tools And SDE's; SDE Can Be Closed

## LIFE CYCLE PROCESS

| Requirements Engineering | → | Software Manufacturing | → | Operational Validation |
|---|---|---|---|---|
| ($\approx 1^{1}/_{2}$ years) | | ($\approx 3/4$ year) | | ($\approx 1$ year) |
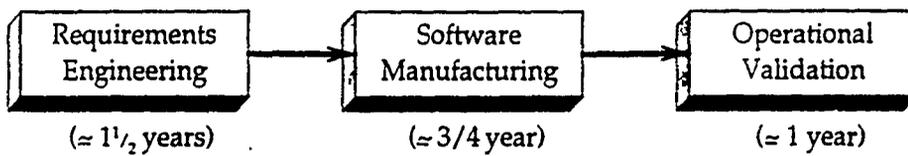
- Requirements Changes Are Encouraged
- Software Design Independent Of Computing Platform
- Highly Automated Coding; Declarative VHLL; Enormous Productivity
- Abstraction Oriented In All Things
- Prototyping; Reuse; Simulation
- Trial Deliveries Into The Field
- Evaluation Of Multi-competing Designs
- Cost Plus; Always Competitive
- Large Up-front Investment For Tools, SDE's; SDE Must Be Open

## THE PRODUCTION ARTIFACTS PROBLEM

## REQUIREMENTS PHASE

Concept of Operations

Requirements Generation

Risk Assessment

Prototype Building

Reuse

Simulation

Prototype Execution

Generate and Validate Design Transfer Spec

## THE MANAGER VS. SOFTWARE DESIGNER PROBLEM

Operations Concept

| Phenomenology 1 | · · · | Phenomenology M | Operating System 1 | · · · | Operating System N |

## THE MANAGER VS. SOFTWARE DESIGNER PROBLEM

Operations Concept

Phenomenology 1 ••• Phenomenology M Operating System 1 ••• Operating System N

System Processing Harness
- System Communications
- System Control
- System Data Handling

Operations Concept

Phenomenology 1 ••• Phenomenology M Operating System 1 ••• Operating System N

System Processing Harness
- System Communications
- System Control
- System Data Handling

## THE MANAGER VS. SOFTWARE DESIGNER PROBLEM

Operations Concept

| Phenomenology 1 | ... | Phenomenology M | | Operating System 1 | ... | Operating System N |

System Processing Harness
- System Communications
- System Control
- System Usage

Usage Layer

## THE COMMUNICATIONS PROBLEM

QA    CM    PM

Requirements Spec

Phenomenology Spec

Glueware Spec

User Interface Spec

Executable Spec

Prototype

Documents

This page is intentionally left blank.

*Technical Presentation 5*

*"Multiple Views of Requirements"*

*Dr. Alan M. Davis*
*George Mason University, USA*

# CLASSIC DEFINITION OF REQUIREMENTS

The activity that encompasses the definition of "what" the system is without decribing "how" it works.

# BETTER DEFINITION OF REQUIREMENTS

o  All activities up to and including the definition of the system's external behavior

o  It thus includes analysis of the problem domain which clearly precedes external behavior specification of the solution system

o  It thus excludes definition of any of the actual physical sub-components of the system under specification

o  Note: External behavior can be described at any level of detail and it is still requirements

# MULTIPLE DIMENSIONS OF REQUIREMENTS

o Problem Analysis vs. External Behavior of Solution System

o Levels of Abstraction

o Multiple Views

# ANALYZING PROBLEMS OR SOLUTIONS?

- <u>What</u> is the problem, not <u>how</u> are we going to solve it
  - Primarily decomposition process
  - Ambiguity/fuzziness
  - Purely in terms of problem owners
- <u>What</u> is the solution system, not <u>how</u> will it work internally
  - Primarily a descriptive (specification) process
  - Consistency
  - Springboard for design and test
  - Purely in terms of users
- Understanding so you can make intelligent choices v. external manifestation
- Problem analysis v. documenting external behavior
- Both included in requirements phase

## LEVEL OF REQUIREMENTS
## ABSTRACTION

- Communicate
- Communicate via voice
- Communicate via telephone system
- Provide local calls, call forwarding, long distance
- To make a long distance call
  - Lift phone
  - Hear dial tone within 2 seconds
  - Dial 9
  - Hear distinctive dial tone within 2 seconds.
- To make a long distance call
  - If dial tone generator available
    Then hear dial tone within 2 seconds on clock A
    Else hear reorder tone within 2 seconds on clock A

# EXAMPLES OF VIEWS

- o  Asynchronous Processes/Objects
- o  Data Structures
- o  Data Flows
- o  Data and Control Flows
- o  Finite State Machines
- o  Extended Finite State Machines
- o  Petri Nets
- o  Human/Machine Interface
- o  Hybrid

# SAMPLE OF RICHNESS: DATA FLOW VIEW
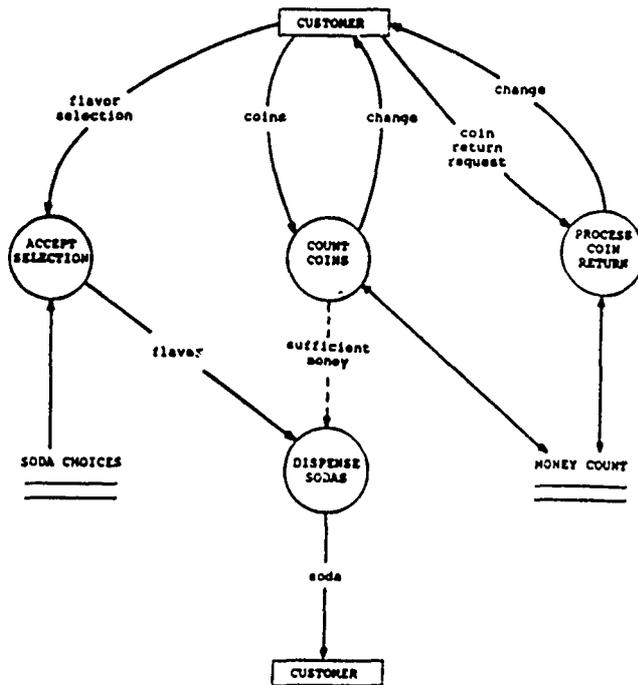


# SAMPLE OF RICHNESS: ER VIEW
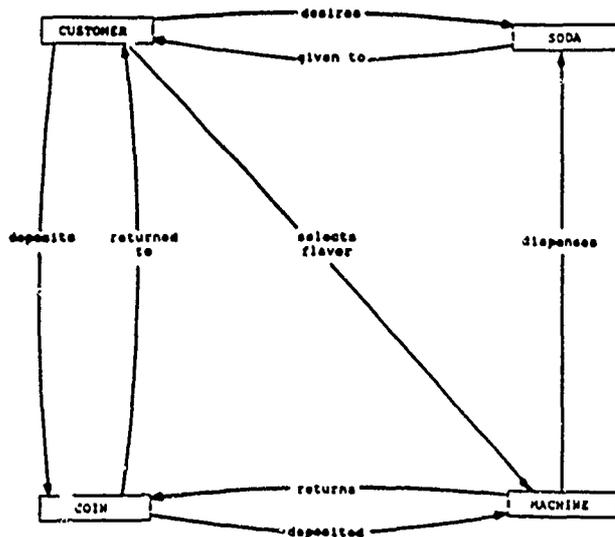
# SAMPLE OF RICHNESS: FSM VIEW
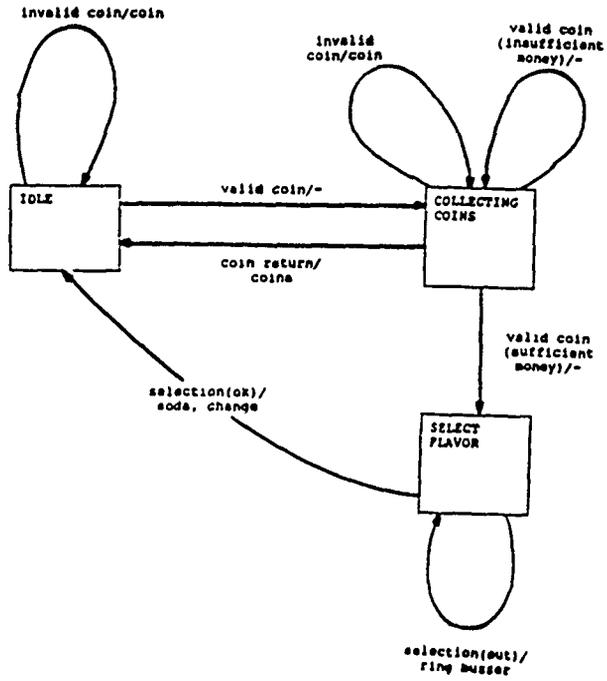


# SAMPLE OF RICHNESS: OBJECT VIEW
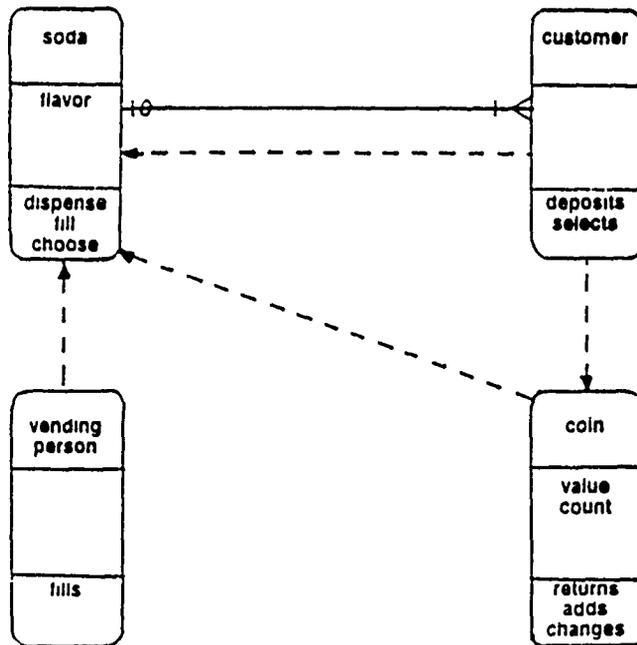
# SAMPLE OF RICHNESS: DATA STRUCTURES VIEW

---

```
SODA SELECTIONS(3)
        NAME
        PRICE
        MAXIMUM-COUNT
        CURRENTLY-AVAILABLE-COUNT
COINS-ENTERED
        NUMBER-OF-NICKELS
        NUMBER-OF-DIMES
        NUMBER-OF-QUARTERS
DATE-OF-LAST-REGULAR-MAINTENANCE
```

# SAMPLE OF RICHNESS: HMI VIEW

---

```
SELECTION #1 -- COKE CLASSIC --------->     PUSH

SELECTION #2 -- DIET PEPSI ----------->     PUSH

SELECTION #3 -- RC COLA -------------->     PUSH

COIN RETURN -------------------------->     PUSH
```

# EXAMPLES OF TECHNIQUES

| Technique | Prob Dom | Sol'n Dom | Func tion | Asyn Proc Objts | Data Strc | Data Flow | FSM Mastr | Slave | Stim | Feat | En- tity | HMI | Levels of Abst'ion |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRD | X | | | | | X | | | | | X | | 1 |
| PAISLey | X | X | X | X | | | | | | | | | 1 |
| JSD | X | | | X | X | | | | | | X | | 1 |
| OORA | X | | | X | | X | | | | | X | | 2 |
| DeMarco SA | X | | | | | X | | | | | | | N |
| Ward SA | | X | | X | | X | | X | | | X | | N |
| Hatley SA | | X | | X | | X | X | | | | | | N |
| Y'rdon MSA | | X | | X | | X | | X | | | X | | N |
| USE | | X | | | | | | | | X | | X | 1 |
| Statemate | | X | | X | | X | X | X | | | X | | N |
| REVS | | X | | | | | | | X | | | | N |
| RLP | | X | | | | | | | | | X | | 1 |

# HATLEY VS. WARD

o  Both Combine DFDs and FSMs

o  Both Add Control Signals

o  Completely Different Semantics

# THE RESEARCH GOAL

o **Enable Developers to Each Select Optimal Views for Their Aspect of the System**

o **Check Any View for Internal Inconsistency, Incompleteness, and Ambiguity**

o **Derive Parts of One View From Another**

o **Check for Consistency Among Views**

o **Transform Views Used by One Methodology into Views Used by Others**

o **"Execute" a Subset of Views While Observing Any One View**

# THE RESEARCH APPROACH

o **Fully Understand Multiple View Problem**

- **Define Meta-Model**

- **Define Views in Terms of the Meta-Model**

- **Formally Define View Ambiguity, Inconsistency, Incompleteness**

- **Formally Define Inconsistency Between Views**

- **Establish Derivation Capabilities**

o **Specify Requirements for a Requirements Environment**

- **Use Multiple Views**

o **Construct the Requirements Environment**

- **Database**

- **Single-View Checkers**

- **Multiple View Consistency Checkers**

- **Automatic View Generators**

- **The Executors**

D-49

# THE BEGINNINGS: A FIRST-DRAFT META-MODEL

o  **Object-Based**

o  **Standing on Coad's Shoulders(OOA)**

o  **A Few Views Have Been Partially Defined**

  - **Objects**

  - **Structure**

  - **Attributes**

  - **Service Names**

o  **Semantics (i.e., Service Definitions) Still Weak**

# SUMMARY

o  **Wide Spectrum of Requirements Tools/Techniques/Languages Available**

o  **Each Ideal for a Particular Aspect of a Problem**

o  **Currently Little Compatibility Exists Conceptually or Physically**

o  **ERA or Object-Oriented Meta-Models Appear to Offer Potential for Common Underlying Representations**

o  **Representation of a Few Views/Methodologies Using an OOMM Underway**

*Technical Presentation 6*

*"An Integrated Approach
to Requirements Engineering"*

*Dr. Raymond T. Yeh
International Software Systems, USA*

# Generic Problems with Requirements

Uncertainty

Volatility

Ambiguity

Inconsistency

Incompleteness

Infeasibility

Incorrectness

Insufficient Communication

Inherent Complexity

Lack of concerns for the entire life cycle

---

# The Basic Framework

• Requirements Process is Intertwined with System
Creation and Evolution Process
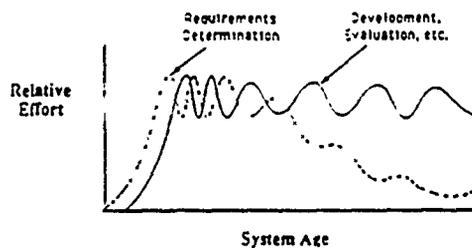


• Areas of Support for Requirements Engineering
Must be Considered in an Integrated Manner
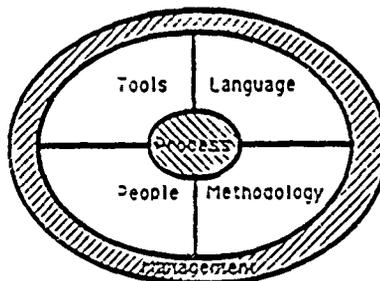
## Generic Questions in Requirements Deviation Activities

1. What is the purpose of this activity?

2. What information is needed?

3. Who is involved?

4. How is the needed information obtained?

5. What to do with the information?

6. What form or language should be used to document the new information?

7. Decision criteria as to whether to proceed?

8. What tools should be used?

## Definitions and Roles

Process Model   a model which enables defining a specific process.

Process:   a set of activities needed for a goal or purpose, and the criteria in deciding the sequencing and duration in the execution of those activities.
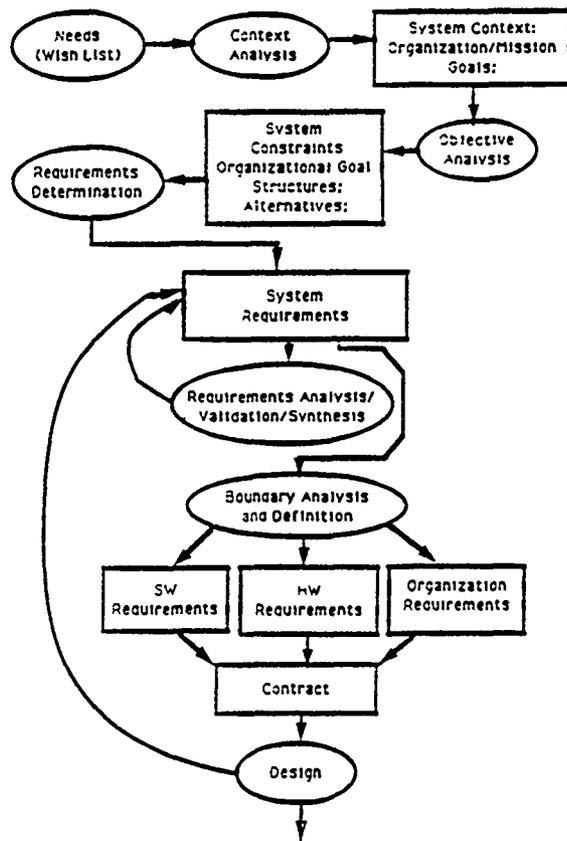
A Requirements Engineering Process Model

---

## Generic Requirements Process Activities

- Context Analysis

- Objective Analysis

- Requirements Determination
  (evaluate alternatives)

- Requirements Analysis

- Requirements Synthesis

- Requirements Validation

# Methodology
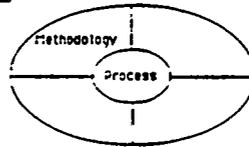


Purpose:

provide systematic plans for accomplishing
goals or implementing guiding principles.

Approaches/Issues:

· What principles guide the process?

For example:

- · Separation of concerns
- · Risk management
- · Control complexity

· What are the methods that tell you how to implement
the principles?

For example:

- · Modeling
  - · Conceptual modeling
  - · Operational modeling
    (prototyping)
- · Work structure breakdown
  - · Simplification
  - · Abstraction
  - · Partition
  - · Projection

---

# People:
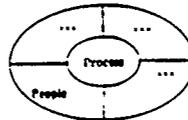


Purpose:

· Identify generic role-players (participants and
stakeholders) for the process: e.g., users,
buyers, sellers, developers

Approaches/Issues:

· What are role-player needs, i.e., their view of
an ideal system?

· What are role-player responsibilities for
activities: input, communication, feedback,
judgement?

# Language
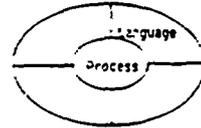


## Purpose:

Provide expression and communication for and among
different people and different concerns.

## Approaches/Issues:

- multiple languages for different major concerns.
  disciplines. and stakeholders

- multiple interfaces

- underlying commonality to support data-sharing. automated
  aid of communication

- formal languages for preciseness. automated checking

- more widespread use for enhanced support of
  information capture



---
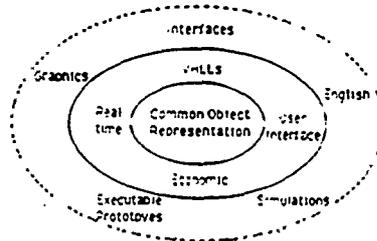
# Automation (Tools)
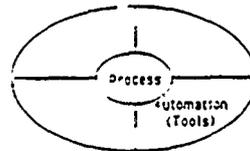


## Purpose:

Provide automated support for engineering and
management activities.

## Approaches/Issues:

- What is/are the right tool(s) to use?

- What is a right kind of architecture
  (e.g., integration platform)?

- How do you incorporate tools into practice?

# Management
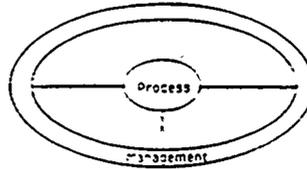
Purpose:

    direct and insure coordination of resources and
processes to accomplish goals

Approaches/Issues:

- planning and controlling allocation of resources:
  financial, human, material, information, time;
- measuring, monitoring, and controlling quality:
  of process, product, and people;
- utilizing real project data for planning;

  - getting the users involved

  - be concerned with the entire life cycle process

  - getting the baseline requirements

  - use incremental commitment

  - separate the concerns.

---

# Generic Questions Within Each Activity

Example - Objective Analysis

1. Purpose

   - Why do they want this?

   - Do they really want what they are saying?

     To make sure organizational investments (long
     term goals) are not shorthanded by the short
     term system goals.

2. What information is needed?

   - What problems currently exist in the organization?

     (problems can be seen as the difference between a
     desired value and the actual achieved value on
     one or more objective dimensions)

   - Need to have the goal/constraints structure!
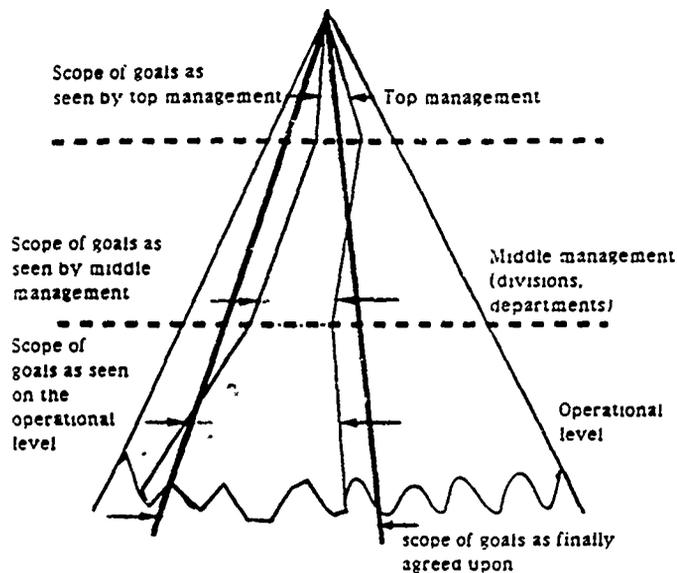
Definition of the scope of the analysis

| | start process | find objectives, constraints and related measures | structure objectives and constraints | determine importance of objectives and constraints | define alternatives to reach objective | determine impact of each alternative | select set of alternatives |
|---|---|---|---|---|---|---|---|
| high level management | A | $I^1$ v I    $J_3$ | I | I V J | I | (J) | I J V |
| steering committee | | $J_1$ | S | A I | | V J | I J V |
| other management levels | I | $I_2$ $V_2$  $J_2$    $J_3$ | I | I V | I | (I) | I J V |
| end users | | $I_3$ $V_3$ $J_3$ | | I V | I | S | I  V |
| project manager | | does not yet exist | | | | | |
| analysis team | | A | A | (A) S(V) (A) | A I | A | A I  V |
| developer manager syst designer programmer tester maintenance | | | | | S or I | S I V | S  V |
| experts of organisation business physical system | | | S S | S S | S S | I V I V I V | S  V V V |
| other sources | | I | | I | | | |

A  actor
I  information source
S  support
V  verayer
J  judge

Role/activity relationship during the OBJECTIVES ANALYSIS phase

# Generic Questions Within Each Activity

4. How to get the needed information?
   (what methods tc use?) .
   Questionaircs. Interviews. etc.

5. What to do with the information?
   (what methods. should be used to analyze the information?)

   - Static analysis
     consistency, completeness
   - Dynamic analysis
     animation of goal diagram
     What if analysis.
   - Verification
   - Validation

6. What form or language should be used to document the
   new information

7. Decision criteria as to whether to proceed?

8. What tools should be used?

   - graphical drawing tools
   - simulation models of applications

| OBJECTIVES / SUB-OBJECTIVES | | increase earnings | increase number of customers | increase price | increase service | increase quality of personnel |
|---|---|---|---|---|---|---|
| Importance | | 1 | (9) | (1) | (5) | (?) |
| Increase earnings | | ... | 0 | 0 | 0 | 0 |
| Increase number of customers | | 5 | ... | (-0 7) | 0 | 0 |
| Increase price | | 5 | (-0 7) | ... | 0 | 0 |
| Increase service | | 0 | 8 | 0 | ... | (6) |
| Increase quality of sales personnel | | 0 | 2 | 0 | (6) | ... |
| CONSTRAINTS | Importance | | | | | |
| Size of market | 8 | 0 | (-0 2) | 0 | 0 | 0 |
| Elasticity of market | 9 | 0 | 0 | (-0 8) | 0 | 0 |
| Current personnel | 6 | 0 | 0 | 0 | 0 | (-0 9) |
| Structure of labor market | 4 | 0 | 0 | 0 | 0 | (-0 3) |

Goal/sub-goal and goal/constraint matrix

Example of a Goal/Constraint Diagram

| OBJECTIVES<br>SUB-<br>OBJECTIVES | | increase<br>earnings | increase<br>number<br>of customers | increase<br>price | increase<br>service | increase<br>quality of<br>personnel |
|---|---|---|---|---|---|---|
| Importance | | 1 | ( 9 ) | ( 1 ) | ( 5 ) | ( 5 ) |
| Increase earnings | | ... | 0 | 0 | 0 | 0 |
| Increase number of customers | | 5 | ... | (-0 7) | 0 | 0 |
| Increase price | | 5 | (-0 7) | ... | 0 | 0 |
| Increase service | | 0 | 8 | 0 | ... | ( 6 ) |
| Increase quality of sales personnel | | 0 | 2 | 0 | ( 6 ) | ... |
| CONSTRAINTS | importance | | | | | |
| Size of market | 8 | 0 | (-0 2) | 0 | 0 | 0 |
| Elasticity of market | 9 | 0 | 0 | (-0 8) | 0 | 0 |
| Current personnel | 6 | 0 | 0 | 0 | 0 | (-0 9) |
| Structure of labor market | 4 | 0 | 0 | 0 | 0 | (-0 3) |

Goal/sub-goal and goal/constraint matrix

# Summary

- Use integrated approach to solve problems:

  - requirements process intertwines with system
    evolution process

  - integrate different areas of concern
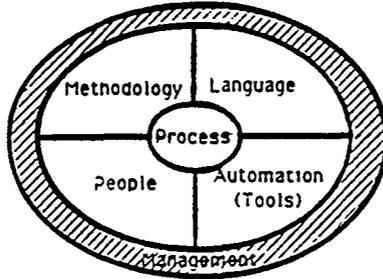
This page is intentionally left blank.

*Technical Presentation 7*

*"Knowledge-Based Requirements Assistant"*

*Mr. Douglas A. White*
*Rome Air Development Center, USA*

# The Challenges of Air Force Software

o Computer Software Dominates the Functioning of Most Military Systems
  (AF Studies Board)

o Computer Software is a problem in 7 out of 10 troubled systems.
  (AFSC/PLR "Bold Stroke" Briefing)

o Cost of AF Mission Critical Software will increase by 50% by 1995.
  (EIA Defense Electronics Market 10 yr forecast)

o Software was 5% of AF budget in 1986, will be 10% by 1990.
  (Software Growth & Logistics, AFALC/ERC)

o Demand for Software is growing at 12%/yr; Personnel 4%/yr; Productivity 4%/yr
  (Boehm, Martin)

o Maintenance Accounts for 60-90% of Software Lifetime Costs
  (Software Growth & Logistics, AFALC/ERC)

o Cost of Software Maintenance is growing by 26%/year
  (V. Castor/ OUSD(R&DT))

---

# KNOWLEDGE-BASED SOFTWARE ASSISTANT

# (KBSA)

BASIS FOR A NEW SOFTWARE PARADIGM - SHIFTING

FROM INFORMAL PEOPLE-BASED DEVELOPMENT TO

FORMALIZED COMPUTER-ASSISTED DEVELOPMENT

# KBSA DEVELOPMENT PARADIGM
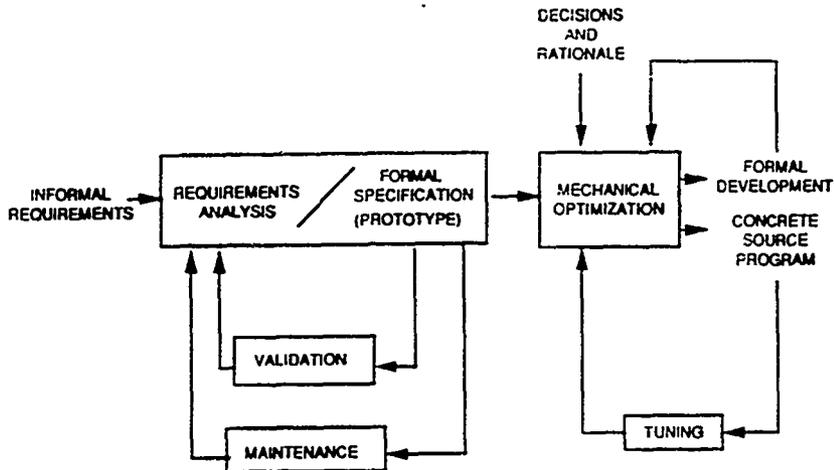


o  Machine is "in the loop"

o  All lifecycle activities machine mediated and supported

o  "Corporate Memory"

# KBSA ARCHITECTURE
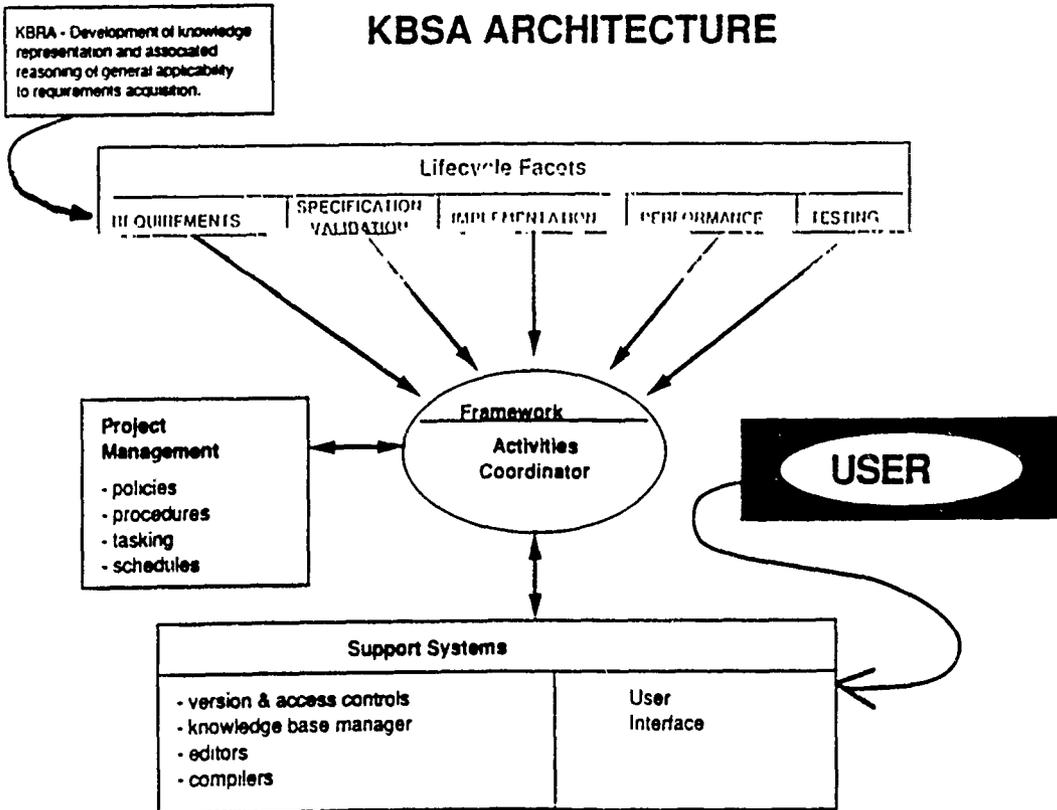
# Requirements Engineering

Acquisition, analysis, and communication of system description.

System Behavior
Boundary Conditions
Trade-off Formulas
Dependencies
Definitions

---

# TODAYS TECHNOLOGY

# FORMAL REQUIREMENTS
# (SOCLE)

**Ex. Constraint:**

```
(air-traffic-control (ako ($value (system))))
(constraint ($value ((multiplier (at* tracker initiation time)
                                 (at* objects-tracked speed)
                                 (at* geographic-coverage distance))))
```

**Ex. Formula:**

```
(multiplier (at radar-43 sweep-rate)
            (at tracker-21 number-of-radar-returns-required)
            (at tracker-21 initiation-time))
```

# KBRA THEME
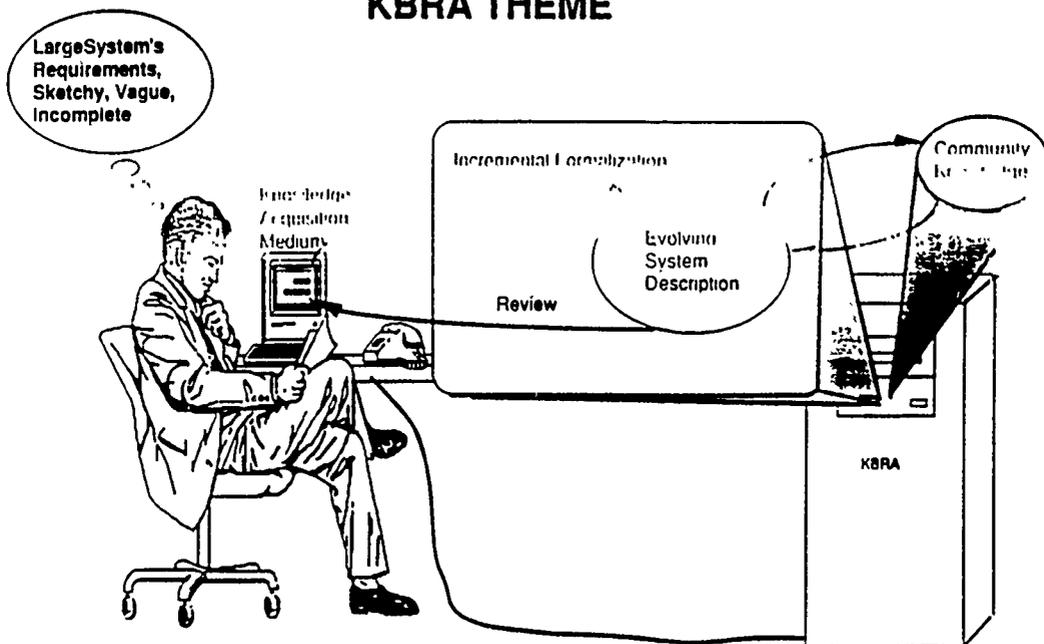


o Incremental Formalization          o Reusable Programming Knowledge
o Presentation Based Interface        o Trade-off Analysis Support

# BENEFITS

Informal   -   Multiple views, no formal computer language, postpone commitments

Consistent -   Single knowledge representation with automated reasoning and truth maintenance

Incremental -   "Catch-as-catch-can" interpretation, associative retrieval, critiquing, automatic completion

Reusable -   Libraries of application knowledge

Functional Decomposition for FLIGHT-DATA-PROCESSING

Data Flow Diagram for MAINTAIN-TRACK-BLOCK



system function diagram for AREA-CONTROL-CENTER

State Transition Diagram for MAINTAIN-TRACK-BLOCK



FORMULA BUILDING WINDOW

FORMULA BUILDING WINDOW

ASSISTANT DIALOGUE WINDOW --- Context: (ACTIVITY-1 PROCESSING-TIME (2.

Context Diagram for ATC
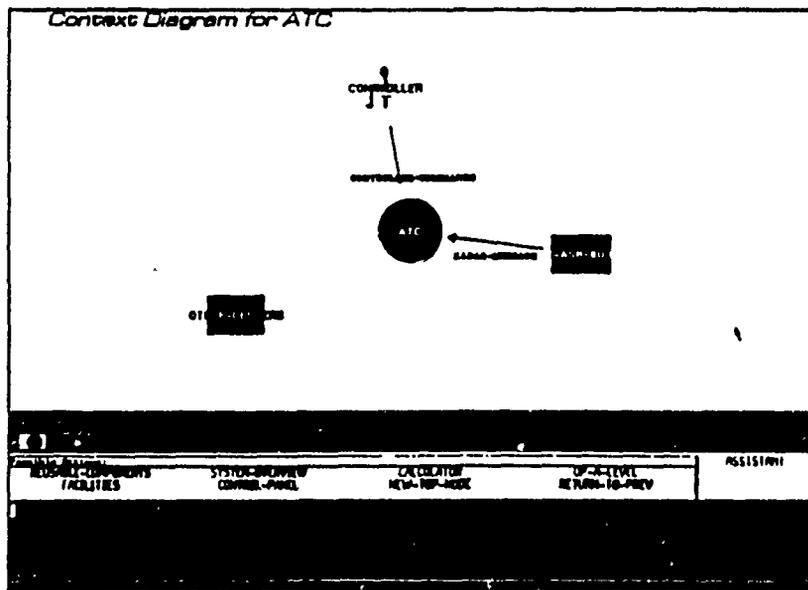
# SUMMARY

o  KBRA Demonstration Model

      Acquisition - multiple views, incremental formalization

      Analysis - automatic critiquing & completion, reusable requirements

      Communication - formal representation, requirements documents

o  Identification of knowledge representation issues

      Presentation, Structured Text, Evolving System Description

o  Formalization of reasoning processes

      Inheritance, Automatic Classification, Constraint Propagation

This page is intentionally left blank.

*Technical Presentation 8*


*"Insights Into the Influence of Shared*
*User/Customer/Contractor*
*Objectives on Project Success"*


*Mr. Michael S. Deutsch*
*Hughes Aircraft Company, USA*

# Empirical Project Success Study at Software Engineering Institute

- Motivation: paucity of significant empirical data on software project management process

- Goal:  identify factors that discriminate between success and non-success

- Feasibility investigation--

  ○ General understanding

  ○ Education

---

# Hypothetical Model of Project Success

DEPENDENT
MEASURES

PREDICTIVE
MEASURES
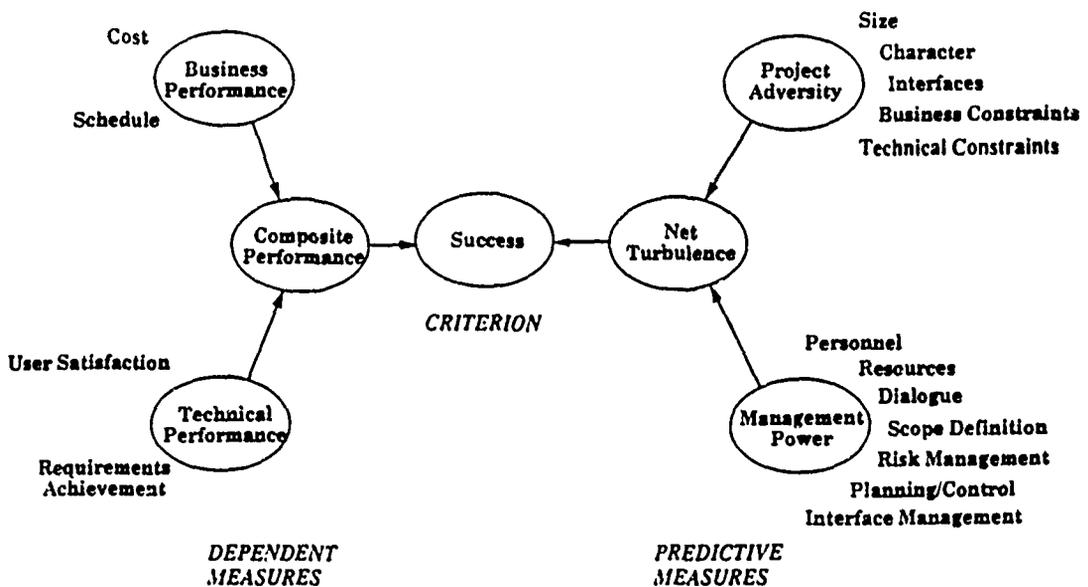
# Management Process Model    **HUGHES**



```
                    ┌──────────────────────┐
                    │ User/customer/contractor │
                    │      dialogue        │
                    └──────────────────────┘

┌──────────────┐   ┌──────────┐   ┌──────────────┐   ┌──────────────┐
│ Strategic risk│   │ Technical │   │  Personnel   │   │   Project    │
│ management    │   │  scope    │   │  resources   │   │ planning and │
│ and planning  │   │definition │   │              │   │  control     │
│               │   │           │   ├──────────────┤   ├──────────────┤
│               │   │           │   │  Physical/   │   │  External    │
│               │   │           │   │  technical   │   │  interface   │
│               │   │           │   │  resources   │   │  management  │
└──────────────┘   └──────────┘   └──────────────┘   └──────────────┘
```

---
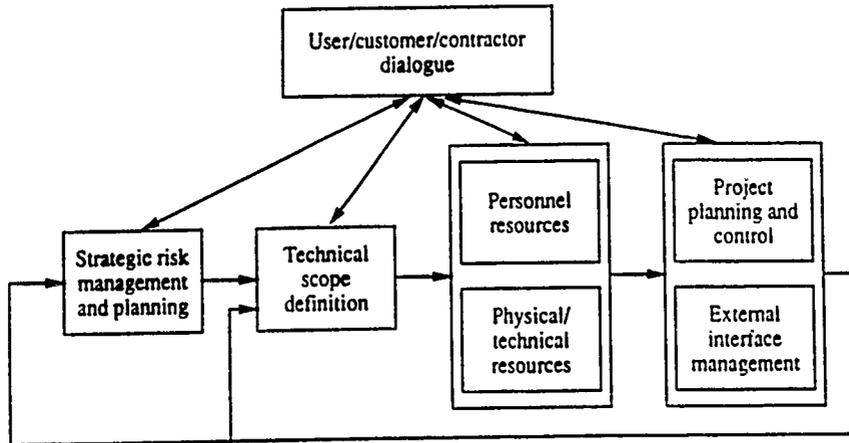
# User/Customer/ Contractor Dialogue    **HUGHES**

- ○ Reconciliation of multiple user needs

- ○ Ongoing collaborative contacts to assure correct content in technical requirements

- ○ User(s) participation in formal design reviews

- ○ Representation of user(s) and contractor on customer's change control board

- ○ Addressing of post deployment support approach

# Exploratory Data Analysis

**HUGHES**

---

**Goal: examine feasibility of conceptual model**

Data on 25 projects collected using informal questionnaire

Caveats on results:

• Insights are pointers for future study

• No statistical inferences

---

Technical
and
Business
Performance
Relationship

TECHNICAL
PERFORMANCE

**HUGHES**

RESPONDENTS' PERCEPTION:
• successful project
• unsuccessful project

BUSINESS PERFORMANCE
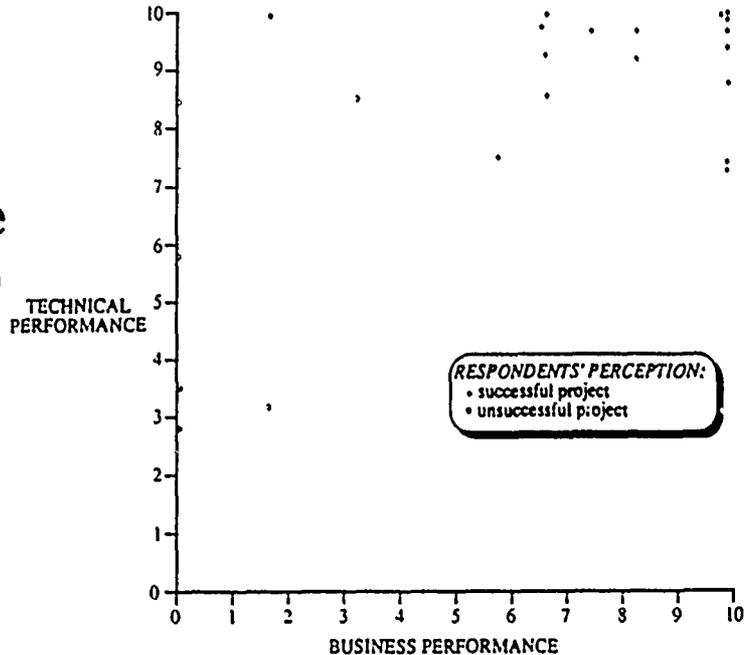
# Management Power and
# Project Adversity Relationship

**HUGHES**

RESPONDENTS' PERCEPTION:
- • successful project
- • unsuccessful project

unfavorable 10

PROJECT
ADVERSITY

favorable 0

unfavorable          MANAGEMENT POWER          favorable

---
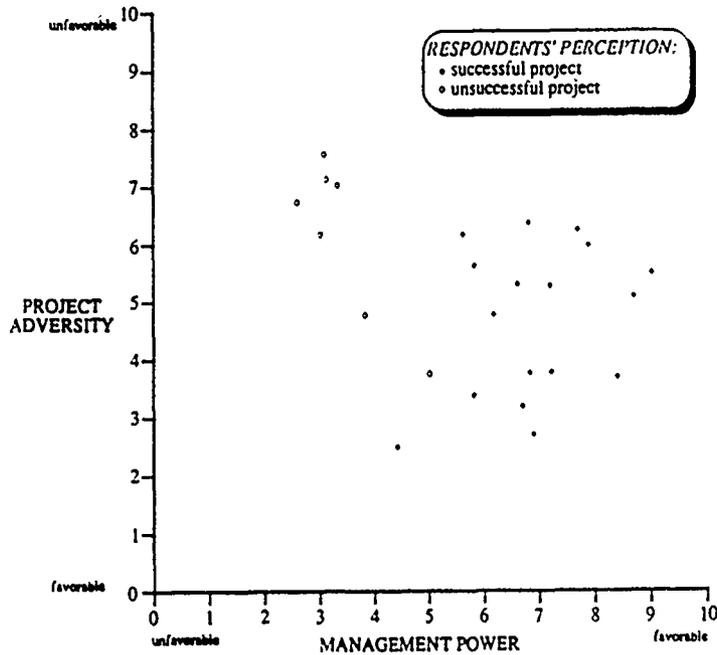
# Project Performance
# and
# Net Turbulence
# Relationship

**HUGHES**

COMPOSITE PROJECT PERFORMANCE

unfavorable          NET TURBULENCE          favorable
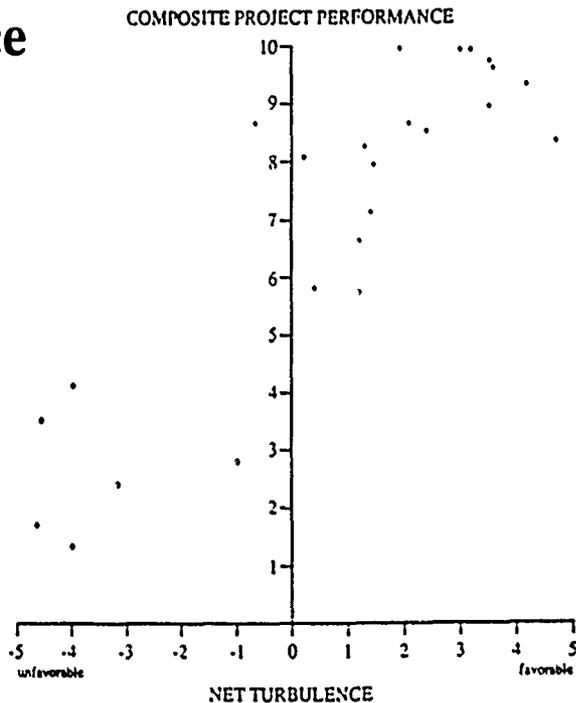
# Intercorrelations of Predictive and Performance Measures  **HUGHES**

| | All projects | | | High adversity projects | | |
|---|---|---|---|---|---|---|
| | Composite performance | Technical performance | Business performance | Composite performance | Technical performance | Business performance |
| Net turbulence | 0.81 | 0.65 | 0.79 | 0.81 | 0.70 | 0.85 |
| Management power(overall) | 0.66 | 0.72 | 0.61 | 0.78 | 0.72 | 0.76 |
| personnel resources | 0.70 | 0.63 | 0.70 | 0.85 | 0.83 | 0.82 |
| technical resources | 0.40 | 0.24 | 0.45 | 0.55 | 0.33 | 0.62 |
| user/customer/contractor dialogue | 0.63 | 0.57 | 0.59 | 0.65 | 0.70 | 0.54 |
| technical scope definition | 0.56 | 0.61 | 0.54 | 0.74 | 0.49 | 0.81 |
| strategic planning/risk management | 0.15 | 0.40 | 0.12 | 0.44 | 0.54 | 0.37 |
| project planning/control | 0.46 | 0.66 | 0.40 | 0.77 | 0.74 | 0.73 |
| external interface management | 0.48 | 0.62 | 0.42 | 0.48 | 0.58 | 0.39 |
| Project adversity(overall) | -0.61 | -0.38 | -0.62 | -0.58 | -0.41 | -0.61 |
| project size | -0.19 | 0.19 | -0.28 | 0.15 | 0.32 | 0.04 |
| project character | -0.10 | -0.04 | -0.07 | 0.36 | 0.16 | 0.42 |
| external interfaces | -0.47 | -0.19 | -0.54 | -0.36 | -0.03 | -0.53 |
| business constraints | -0.72 | -0.55 | -0.68 | -0.70 | -0.53 | -0.68 |
| technical constraints | -0.31 | -0.50 | -0.26 | -0.35 | -0.60 | -0.28 |

# Top Ten Management Considerations  **HUGHES**

| All projects | | High adversity projects | |
|---|---|---|---|
| Consideration | Correlation | Consideration | Correlation |
| Expertise of initial maintenance team | 0.78 | How well personnel and support requirements specified | 0.88 |
| Periodic cost/schedule estimates for completion | 0.67 | Expertise of initial maintenance team | 0.81 |
| Skills of personnel who remained for test/transition | 0.66 | Periodic cost/schedule estimates for completion | 0.78 |
| Reconciliation of multiple user needs | 0.63 | Periodic review and updating of risk parameters | 0.73 |
| How well personnel and support requirements specified | 0.62 | Skills of personnel who remained for test/transition | 0.72 |
| User representation on change control board | 0.57 | Periodic review of actual versus planned rate of accomplishment | 0.71 |
| Expertise of development personnel | 0.56 | Expertise of development personnel | 0.71 |
| User/customer/contractor contacts on project technical content | 0.54 | How well system qualification requirements specified | 0.70 |
| External interface stability after preliminary design review | 0.54 | Reconciliation of multiple user needs | 0.65 |
| External interface stability before preliminary design review | 0.52 | Prioritization of requirements for implement-to-schedule planning | 0.64 |
| On-going liason with interfacing elements/systems | 0.52 | | |

# Project Performance and
# Business Constraints Relationship

**HUGHES**



COMPOSITE PROJECT PERFORMANCE (y-axis, 0-10)

BUSINESS CONSTRAINT RATING (x-axis, 0-10, favorable to unfavorable)

---
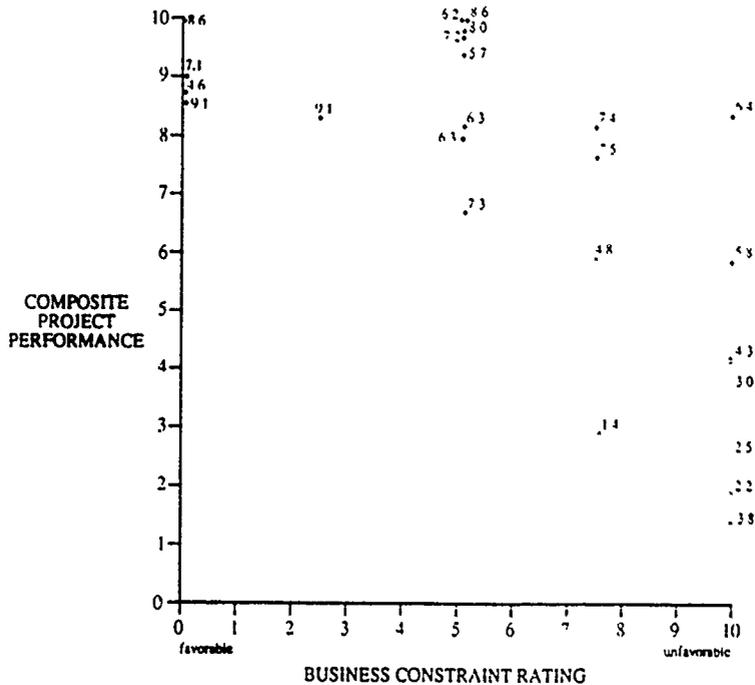
# Key Points Summary

**HUGHES**

- Empirical data confirms anecdotal experience and intuition

- Collaboration of user/customer/contractor on technical content definition affects performance

- Technical definition uncertainty with other uncertainties impacts performance

- Adverse projects require more sophisticated management including requirements

This page is intentionally left blank.

# Technical Presentation 9

## *"The Serpent User Interface Management System"*

*Mr. Reed Little*
*Carnegie-Mellon University, USA*

# Introduction

- **Problems**

- **Objectives**

- **Approach**

- **Use of Serpent**

- **Serpent Architecture**

- **Serpent Editor**

- **Transition**

- **Summary**

*89-Serpent-reed-1*

# User Interface (UI) Problems

- **User interface accounts for large portion of life cycle costs - in some interactive systems more than 70%**

- **Impacts all aspects of the life cycle**

    - requirements
    - development
    - sustaining engineering
      -- changes to user interface
      -- integration of new input/output (I/O) media

*89-Serpent-reed-2*

# Life Cycle Problems

- **Requirements**

  - **evolutionary, not well specified**
  - **written specifications inadequate for conveying "look and feel" of interface**
  - **customers may not know what is practical**
  - **customers may not know cost; time may be more important than dollars**

- **Design/implementation**

  - **very labor intensive**
  - **inadequate existing methods and tools**
  - **manual development time consuming and error prone**

89-Serpent-reed-3

# Life Cycle Problems (cont.)

- **After system completed**

  - **frequent and complex changes required**
    - **-- user interface intertwined throughout system**
    - **-- customer not able to completely comprehend interactions until system is delivered and in use**

  - **difficult to take advantage of new I/O media**
    - **-- use of particular hardware/software media permeates design and implementation**

89-Serpent-reed-4

# Objectives

- Make user interfaces easier to specify

- Support incremental development of user interfaces (prototypes)

- Provide for a "bridge" between prototype and production versions of system

- Support insertion of new I/O media during sustaining engineering

*89-Serpent-reed-5*

# Approach to Reducing UI Problems

- Provide single tool which supports incremental specification and execution of interface

- Separate concern of user interface specification and execution from rest of system concerns

- Apply non-procedural language and graphical techniques to user interface specification

*89-Serpent-reed-6*

# The Serpent UIMS

- **Has specialized** language for user interface specification

- **Supports I/O media independent applications**

- **Supports both prototyping and production**

- **Supports multiple I/O media for user interactions**

- **Supports ease of insertion of new I/O media**

*89-Serpent-reed-7*

# Serpent Use
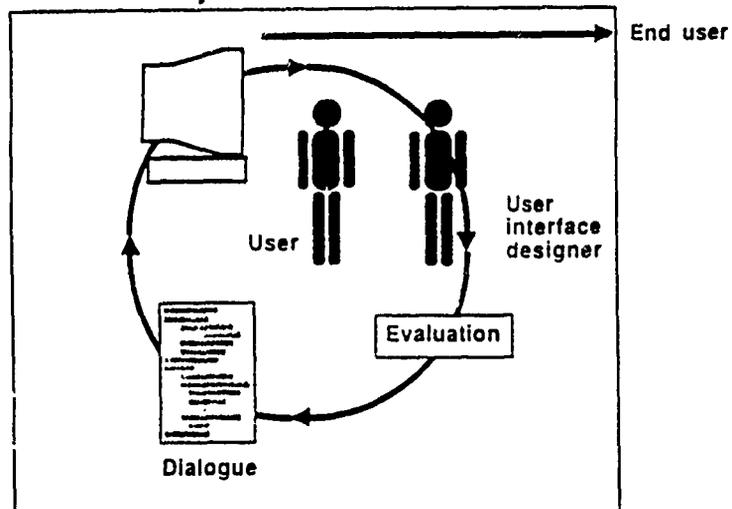


*89-Serpent-reed-8*

# Serpent Architecture
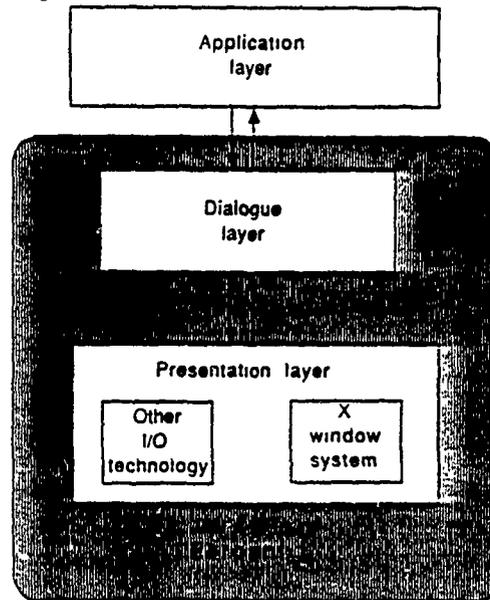
# Slang, UI Specification Language

- **Based on production model**

    - data driven
    - allows multiple threads of control

- **Provides multiple views of the same data**

    - implemented with constraint mechanism
    - re-evaluates dependent values automatically when independent values modified
    - applies to application values, I/O media display values, and local variables

# Prototyping

- **Detailed knowledge of Serpent dialogue model is not required**

- **Application not required**

- **Slang allows definition of local data**

- **Serpent automatically enforces constraints**

- **Reasonably sophisticated prototypes can be generated, e.g., visual programming**

*89-Serpent-reed-11*

VISUALLY PROGRAMMABLE CALCULATOR
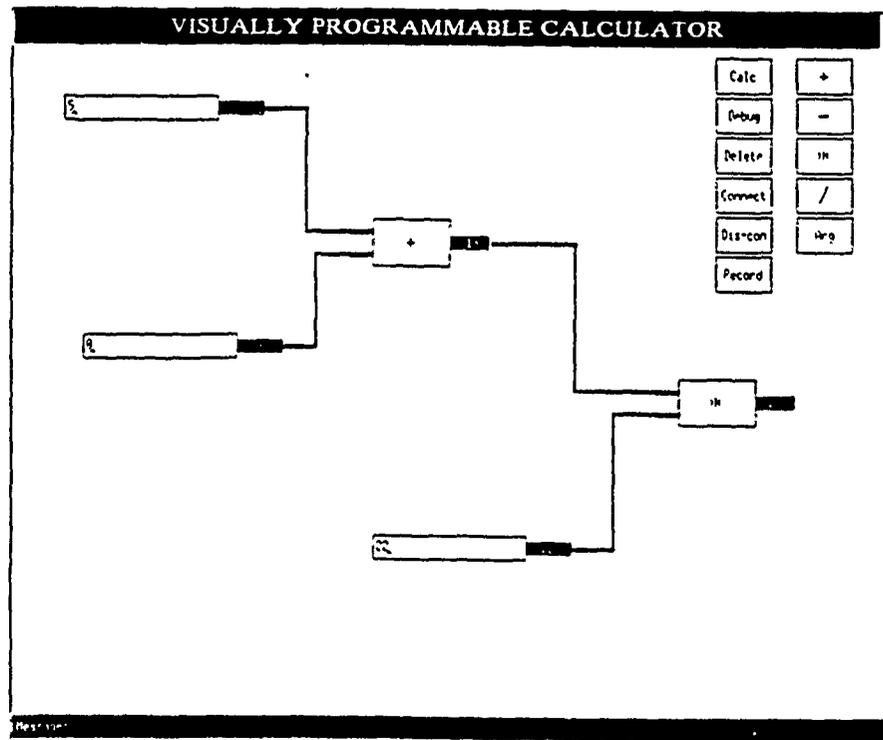
# Input/Output Media

- Serpent designed to simplify the integration of I/O media

- Currently Integrated

  - digital mapping system
  - X11 Athena widget set

- Integrations in process

  - Motif
  - Open Look
  - video-based mapping system
  - experimental gesturing system

89-Serpent-reed-13

# Application

- Can be written in C or Ada

- Views Serpent as similar to database management system

- Creates, deletes, or modifies data records

- Informed of creation, deletion, or modification of data records by dialogue layer

89-Serpent-reed-14

# Serpent Editor

- **Layouts of user interface are best specified or examined graphically**

- **Logic, dependencies, and calculations are best specified textually**

- **Serpent Editor has two portions**

  - **graphical part for examination and specification of layout**

  - **structure part for textual specification**

- **Implemented using Serpent**

*89-Serpent-reed-15*
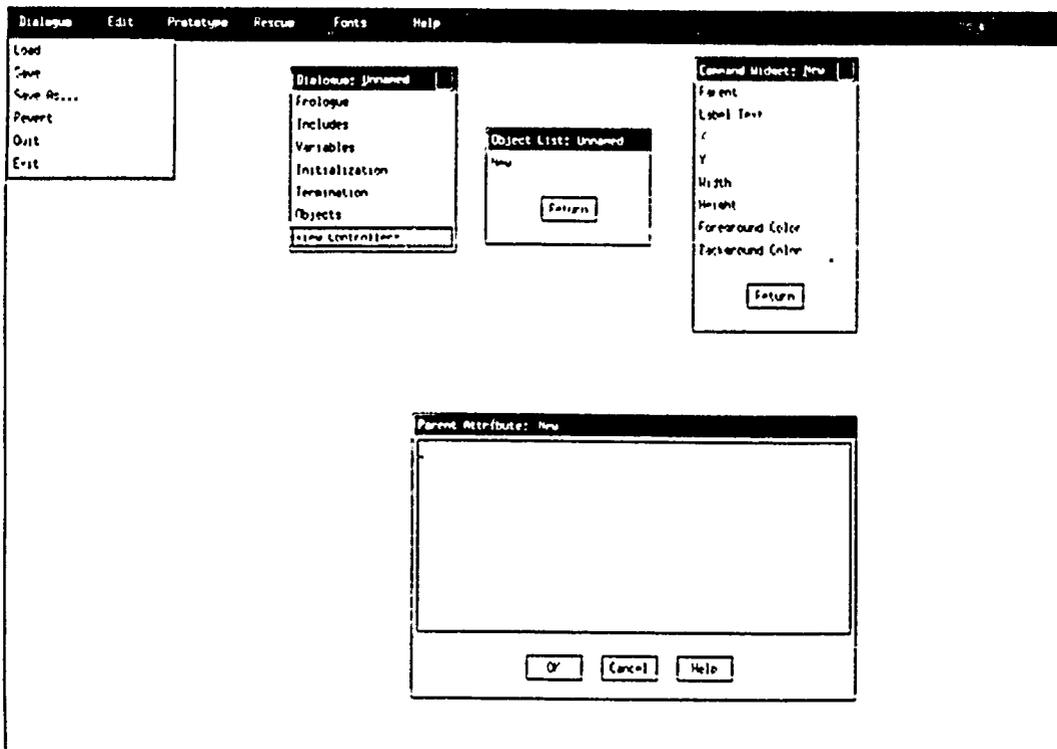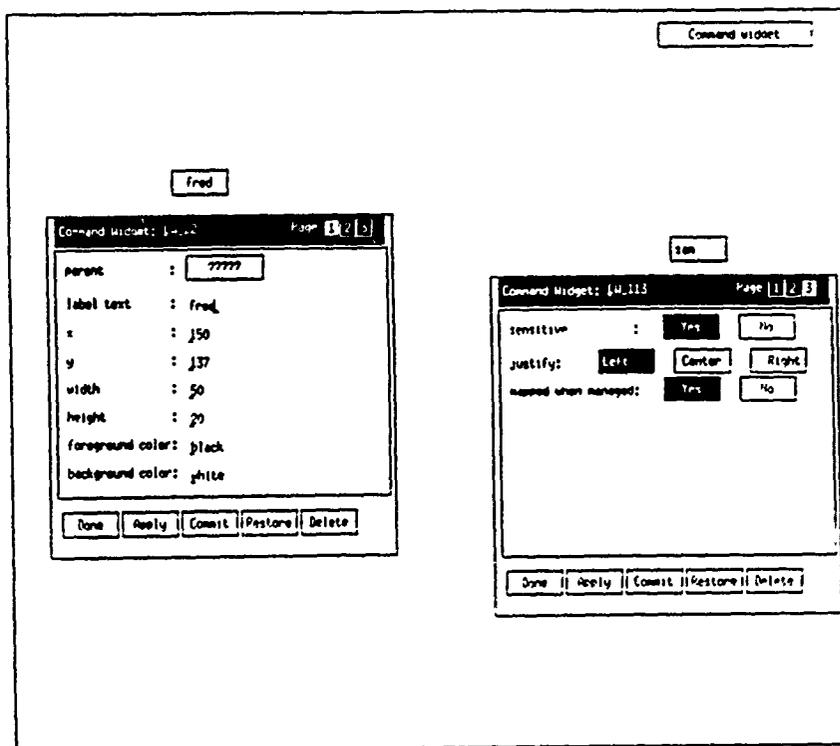
# Transition

- **Encourage use of Serpent**

- **Provide close support for selected sites during interim period**

- **Publicize Serpent**

- **Distribute via electronic media**

- **Commercialization**

*89-Serpent-reed-18*

D-92

# Status

- Serpent (w/o editor) in alpha test

- Available for SUN and VAX (ULTRIX)

- Beta version of Serpent (including editor) available Fall '89

*89-Serpent-reed-19*

# Summary

- Reduces effort for specifying/modifying user interface

- Provides for evolutionary changes of I/O media in fielded system

- Simplifies post deployment user interface modifications

- Provides seamless path from prototype to fielded system

*89-Serpent-reed-20*

This page is intentionally left blank.

*Technical Presentation 10*


*"Using Joint Application Design (JAD)*
*Techniques to Accelerate the*
*Requirements Definition Process"*



*Mr. Robert C. Fink*
*Performance Resources, USA*
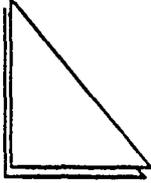
# THE CHANGING MARKET ENVIRONMENT: CAUSES

- EMPHASIS ON A GLOBAL MARKETPLACE
- EC '92 - EUROPE AS ONE TRADING PARTNER
- THE FAR EAST - AGGRESSIVE COMPETITORS
- MERGERS, ACQUISITIONS - MORE BIG PLAYERS
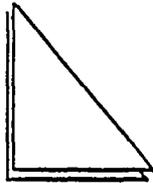
(c) 1989


# THE CHANGING MARKET ENVIRONMENT : EFFECTS

- INCREASED LEVEL OF ACCEPTABLE RISK
- NEED FOR COMPETITIVE ADVANTAGE
- HIGHER PRODUCTIVITY REQUIRED
- FLEXIBILITY TO ADAPT QUICKLY TO NEW CONDITIONS

(c) 1989
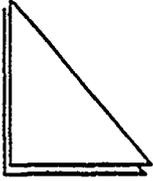
# THE CHANGING SYSTEMS ENVIRONMENT

- EMPHASIS ON IMPROVED DATA MANAGEMENT IN INFORMATION ENGINEERING

- RELATIONAL DATA BASE PRODUCTS

- IBM'S REPOSITORY - CORPORATE DATA RESOURCE

- SHIFT IN THE LIFE CYCLE

- TOOLS SUPPORTING LIFE CYCLE PRODUCTIVITY
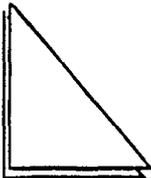
# JAD: JOINT APPLICATION DESIGN



- A GROUPWARE CONCEPT: TEAM-BASED TECHNIQUE
- LED BY A TRAINED FACILITATOR
- SUPPORTED BY A TRAINED ANALYST/DOCUMENTOR
- CENTERED AROUND A WORKSHOP
- FOCUSED ON CONSENSUS-BASED DECISION-MAKING
- USED FOR ADDRESSING INFORMATION ANALYSIS/BUSINESS MANAGEMENT ISSUES

# JAD = INCREASED PRODUCTIVITY

**WITHOUT JAD**: AS MUCH AS 35% OF FUNCTIONAL REQUIREMENTS CAN BE MISSED. ADDITIONAL REQUIREMENTS ADD NEARLY 50% MORE CODE.

**WITH JAD**: LESS THAN 10% OF FUNCTIONAL REQUIREMENTS MISSED. WITH JAD AND PROTO-TYPING, LESS THAN 5% MISSED WITH MINIMAL CODE ADDED.

-Capers-Jones, 1989

---
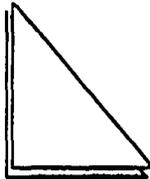
# CHANGING FOCUS

| SYSTEMS FOCUS | BUSINESS FOCUS |
|---|---|
| Technology Driven | Business Decision Driven |
| "Back Office" Transaction Driven | "Front Office" Supported - MIS and DSS |
| Hardware and Software Limiting | Increased Hardware and Software Capabilities |
| Single-Function and Organization | Multi-Function and Cross-Organization |
| Operational and Tactical Role | Strategic and Competitive Edge Role |

(c) 1989
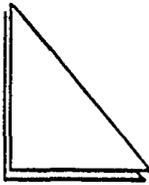
## EMERGING SYSTEMS PROFESSIONAL

| PAST | PRESENT |
|------|---------|
| Computer Expert "Gurus" | Systems Professionals are Consultants |
| Reactive to Users | Catalysts/Planners in Business Change |
| Users New to Computers | Users Have Experience with Systems |
| Technology is all Important | Choose Technology to Fit |
| Programmer/Analyst is Craftsman | Programmer/Analyst is Engineer |
| Programmer/Analyst Dominates | User - Systems Partnership |
| Maintenance - Large Role | Maintenance - Decreasing Role |

Performance Resources, Inc.                                    (c) 1989
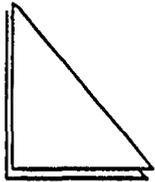
## PRODUCTIVITY FUSION: MASTERING THE BASICS

- METHODOLOGY:  INFORMATION ENGINEERING
- TOOL:  CASE
- TECHNIQUE: JAD
- ENVIRONMENT: FUSION CENTER *

* The name "Fusion Center" is drawn from the U.S. Army Corps of Engineers Management Center under the technology transfer program of the U.S. Government.

Performance Resources, Inc.                                    (c) 1989
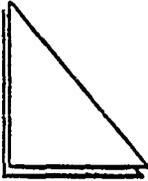
# JAD EVOLUTION

| FIRST GENERATION JAD | NEXT GENERATION JAD |
|---|---|
| FOCUS ON PROCESS | FOCUS ON DATA |
| TRANSACTION ORIENTATION | TRANSACTION + MIS/DSS |
| USER PARTICIPANTS | TIGER TEAMS = BUSINESS + SYSTEMS |
| SCRIBE AS DOCUMENTOR | DESIGN ANALYST/CASE USER |
| APPLICATION-LEVEL ONLY | ENTERPRISE, BUSINESS AREA, AND APPLICATION LEVELS |
| USER REQUIREMENTS ONLY | USER REQUIREMENTS AND LOGICAL DESIGN |
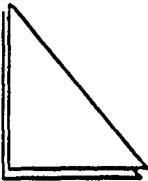
# WORKSHOP
# KEY JAD MODULES



MODULE I:
WORKSHOP
PREPARATION

Operational Guidelines   Workshop Agenda

MODULE II:
CONTEXT
MODEL

Functional Framework

Q/D/P's

MODULE III:
DATA
MODEL

ERD's

Normalized Data Model

MODULE IV:
PROCESS
MODEL

Functional Decomposition

DFD's
Dependency Diagrams
Process/Entity Matrix

Access

I/O's Interfaces    Menus

Workshop Closure
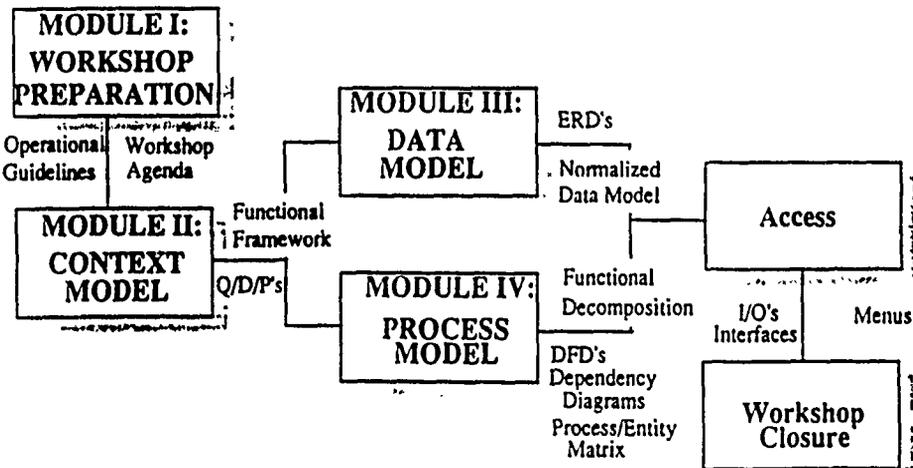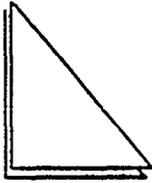
## INDEPENDENT DATA ANALYSIS

- Corporate Architecture as a Corporate Asset
- Elimination of Duplication
- Shared Data - Models Within the Architecture
- Data Separate From Business Process
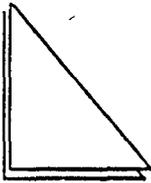
## HIERARCHICAL PROCESS ANALYSIS

- TOP-DOWN ANALYSIS OF BUSINESS
- APPLICATIONS SUPPORT CORPORATE BUSINESS STRATEGY
- FUTURE ORIENTATION
- FLEXIBLE MODEL TO MEET BUSINESS CHANGES
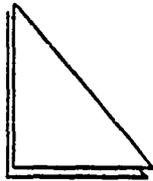
# JAD DELIVERS

- IMPROVED ANALYSIS/DESIGN QUALITY
- REDUCED ANALYSIS/DESIGN TIME/COST
- IMPROVED OWNERSHIP OF SOLUTION
- EARLY ISSUE IDENTIFICATION/RESOLUTION
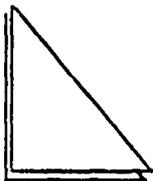
# JAD APPLICATIONS

- CORPORATE/BUSINESS AREA ARCHITECTURES
- PROCESS ENHANCEMENT IDENTIFICATION
- USER REQUIREMENTS FOR APPLICATION
- LOGICAL DESIGN FOR APPLICATION
- PROTOTYPE REVIEW/EVALUATION
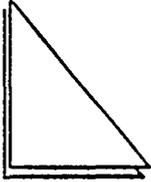
## THE FUSION CENTER
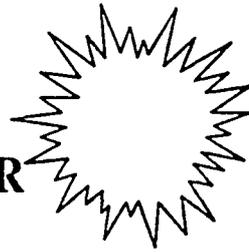
- SPECIAL FACILITY DESIGNED TO SUPPORT GROUP DECISION-MAKING

- AUTOMATED DECISION-MAKING TOOLS AND CASE TOOLS

- TRAINED FACILITATOR AND DOCUMENTOR

- USE OF SPECIAL MATERIALS - WHITE BOARD WALLS, COMPUTER PROJECTION, MOVABLE FURNITURE AND WALLS
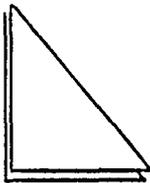
## EIGHT CRITICAL SUCCESS FACTORS

1. EXECUTIVE-LEVEL COMMITMENT
2. EDUCATED SYSTEMS AND USER TEAM
3. EXPERIENCED FACILITATOR
4. CASE SUPPORT
5. DEFINED PROJECT OBJECTIVES
6. DEFINED PROJECT SCOPE
7. DEFINED PROJECT DELIVERABLE
8. LOGISTICAL RESOURCES

This page is intentionally left blank.

# Technical Presentation 11


## "ADA Box Structures for Object-Oriented Software Development"


### Mr. Edward R. Comer
### Software Productivity Solutions, USA

## Welcome to Ada Box Structures!

- **Ada Box Structures** provides a disciplined means to analyze software systems in an object-oriented fashion.

- As an analysis method, Ada Box Structures provides a rigorous framework for describing objects from various perspectives: static and dynamic, internal and external.

- The box structures of black box, state box and clear box provide different views of any object in increasing levels of detail and with increasing visibility into the object.

- **Ada Box Structures** fills a gap in object-oriented methods by providing a rigorous method for discovering application objects.

**SPS**

## Object-Oriented Development

```
┌─────────────────────┐
│   Object-Oriented   │
│      Analysis       │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│   Object-Oriented   │
│       Design        │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│   Object-Oriented   │
│     Programming     │
└─────────────────────┘
```

**SPS**
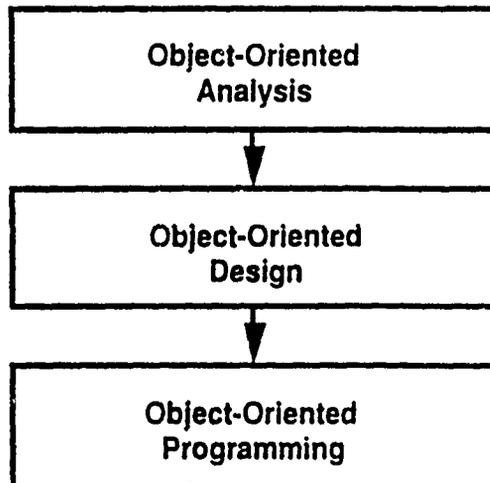
## Advantages of Object-Oriented Development

- Provides a single, consistent model that requires no "great mental leap" from analysis to design and thus increases traceability and maintainability

- Matches the technical representation of the software system more closely to the conceptual view of the application

- Provides a stable framework for analyzing the problem domain and for levying requirements

- Supports implementation using abstract data types

**SPS**

## Some Definitions

An *object* is an abstract data type, which encapsulates data and provides a set of predefined operations to manipulate and access that data.

An *object class* is a collection of object instances with common attributes and a common set of operations.

An *operation* defines an object's capacity for action, response or functioning.

A *stimulus* is an external request for an operation made upon an object.

*Transactions* are behaviorally related sequences of stimuli and responses.

**SPS**

## More Definitions

*Attributes* define the data pertinent to each instance of an object. Attributes encapsulate stimulus histories.

The *state* of an object is determined by the values of its attributes.

Objects may be nested, defining *subobjects* that contribute to the state and behavior of a parent object.

A *relation* is a mapping or association between objects.

*Constraints* denote facts about objects that specify behavior or limitations on behavior or state.

**SPS**

## Perspectives of Objects

Being able to look at problems from different perspectives is a powerful way to reason about and understand systems. These kinds of perspectives are of particular use in understanding and analyzing objects:

- Static and dynamic perspectives of objects
- External and internal object perspectives, and inter-subobject perspective

**SPS**

## Object Perspectives

|  | Static Perspective | Dynamic Perspective |
|---|---|---|
| External Object Model | • Stimuli<br>• Responses<br>• Transactions<br>• Operations | • Operation constraints<br>• External object behavior<br>• External transaction models |
| Internal Object Model | • Attributes | • Attribute constraints<br>• Operational specification of behavior |
| Inter-Subobject Model | • Subobjects<br>• Classification structure<br>• Subobject relations | • Relation constraints<br>• Interaction paths<br>• Subobject interaction models |

**SPS**

---

## The Black Box

The black box view represents the external object model.

**Black Box**

Stimulus → Object / Operation → Response
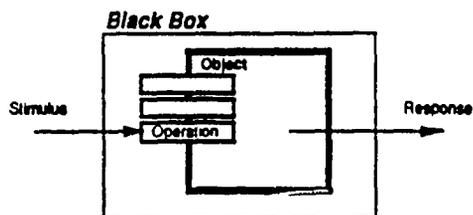
The *external object model* considers only those aspects that can be viewed from the outside.

**SPS**

D-109

# The State Box

The state box view represents the internal object model.



State Box
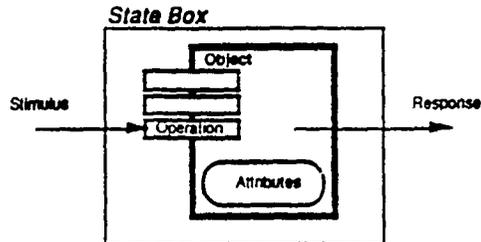
The *internal object model* statically defines the attributes of the object that the object must remember.

SPS

# The Clear Box

The clear box view represents the inter-subobject model of nested subobjects.



Clear Box
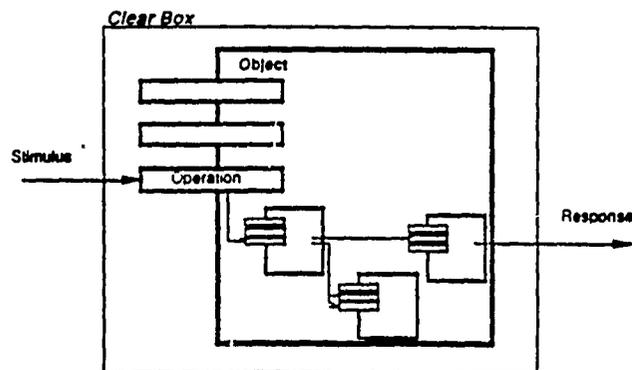
The *inter-subobject model* statically defines the subobjects that are nested within the object, analyzes the classification structure of objects, and defines the relations between subobjects.

SPS

# Box Structures Expansion
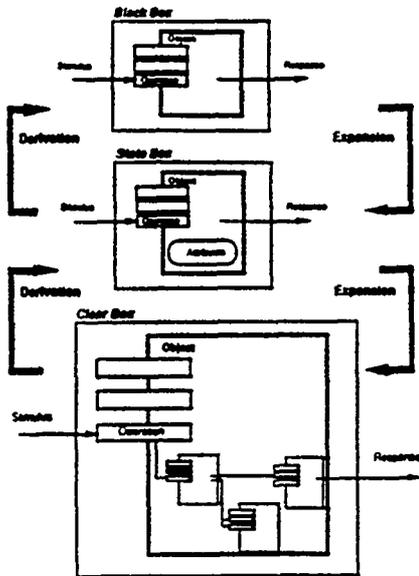
The *black box, state box,* and *clear box* provide behaviorally equivalent views of a system or subsystem at increasing levels of internal visibility.

SPS

# Box Structure Hierarchy

First Level — Black Box

Elaboration of black box into equivalent state box and clear box representations

State Box

Clear Box

Second Level
Decomposition of clear box into black box objects

Third Level

SPS
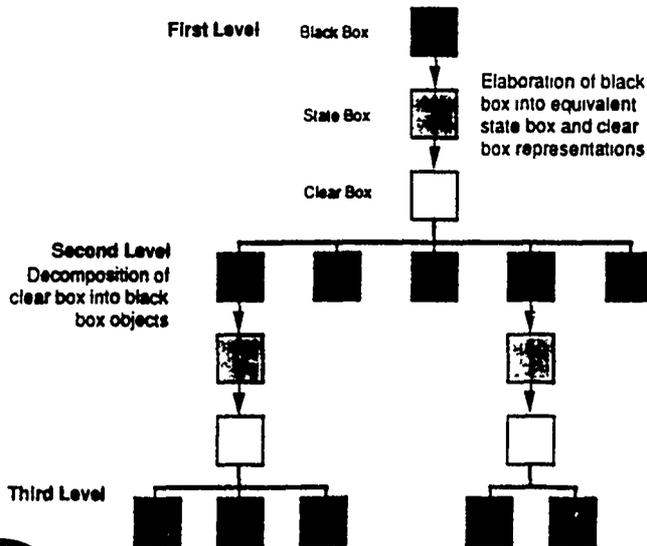
## Ada Box Structures:
### *A Framework for Systems Analysis*

The Ada Box Structures Method provides a framework for systems analysis. This framework guides the integration and application of several different analysis methods. The result of the analyses is information, expressed in text and in graphics, that records the understanding of the system.

**SPS**

## Ada Box Structures
## Work Product Representations

| | Static Perspective | | Dynamic Perspective | |
|---|---|---|---|---|
| | **Work Products** | **Candidate Representations** | **Work Products** | **Candidate Representations** |
| **Block Box** | Stimuli | • Black box diagram<br>• Stimuli list<br>• Detailed stimus specification | Operation constraints | • Formal constraint language<br>• Informal statements |
| | Responses | • Black box diagram<br>• Response list<br>• Detailed response specification | External object behavior | • Stimuli to response trace<br>• Informal assertion description<br>• Formal assertion description |
| | Transactions | • Abstract block box diagram<br>• Transaction list | External transaction models | • Stimuli/response sequence<br>• Transaction diagram |
| | Operations | • Object diagram<br>• Operations list<br>• Stimuli to operations map | | |
| **State Box** | Attributes | • Object diagram<br>• Attribute list<br>• Detailed attribute specification | Attribute constraints | • Formal constraint language<br>• Informal statements |
| | | | Operational specification of behavior | • Informal textual specification<br>• Formal textual specification<br>• Graphical representation |
| **Clear Box** | Subobjects | • Indented list of subobjects<br>• Hierarchical assembly diagram<br>• Nested object diagram | Relation constraints | • Formal constraint language<br>• Informal statements |
| | Classification structure | • Indented classification list<br>• Hierarchical inheritance diagram | Interaction paths | • Interaction matrix<br>• Interaction flow diagram |
| | Subobject relations | • Relation matrix<br>• Object relations diagram | Subobject interaction models | • Sequence of interaction list<br>• Interaction diagram |

**SPS**

## Selection of Representations

There are a number of important factors to consider in selecting specific techniques:

- **Maturity of the technique with respect to object-oriented specifications**
- **Complexity of the system to be specified**
- **Degree of detail and rigor that is desired in the specification**
- **Familiarity and experience of the organization with the technique**
- **Level of expertise of the analysis personnel**
- **Availability of training in the technique**
- **Availability of automated tools to support the technique**
- **Degree of integration possible between tools**

SPS

## The 13-Step Ada Box Structures Process
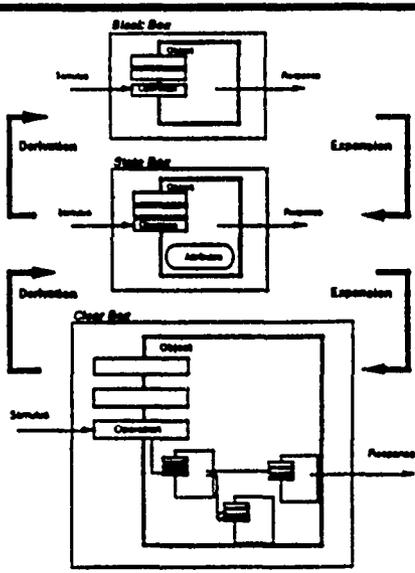


1. Define object stimuli and responses
2. Identify object operations
3. Define informal, external object behavior
4. Conduct transaction analysis

5. Discover state requirements
6. Identify attributes
7. Define operational specification of behavior

8. Conduct state analysis
9. Identify clear box subobjects
10. Classify subobjects
11. Define subobjects' relations
12. Define interaction paths
13. Conduct object interaction analysis

SPS

# Steps 1 - 4:
# Black Box Expansion

1. **Define object stimuli and responses**
2. **Identify object operations**
3. **Define informal, external object behavior**
4. **Conduct transaction analysis**



*Black Box*

---

# Black Box Expansion

## 1. Define object stimuli and responses



## 2. Identify object operations



## 3. Define informal, external object behavior

```
Begin
    Read query string from user;
    If query syntax invalid then
        Send error message to user;
    Else
        Apply query to previously selected component set to
        derive a new  component set;
        If no components found then
            Send message to user;
        Else
            Display summary results for component set;
            Set that component set as the default context for this
            user's commands;
        End if;
    End if;
End;
```

## 4. Conduct transaction analysis

# Steps 5 - 7:
## State Box Expansion

5. **Discover state requirements**
6. **Identify attributes**
7. **Define operational specification of behavior**

**State Box**



Stimulus → Object / Operation / Attributes → Response

SPS

---
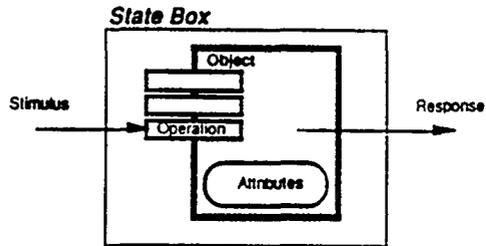
# State Box Expansion

## 5. Discover state requirements

Analysis of the external operation behavior:

```
Begin
  Read query string from user;
  If query syntax invalid then  ———————— Need facet definitions
    Send error message to user;
  Do
    Apply query to previously selected component  < Need library and component
    set to derive a new component set;              set contents
    If no components found then                      Need components and
      Send message to user;                          component facet values
    Else
      Set that component set as the default context
      for this user's session;
    Display summary results for component set;
    End If;
  End If;
End;
```

## 6. Identify attributes

| | |
|---|---|
| Archival device ID | Session component context |
| Component facet definitions | Session component set context |
| Component facet values | Session library context |
| Component sets | User names |
| Components | User passwords |
| Library names | User roles |

## 7. Define operational specification of behavior

```
Begin
  Read Query string;
  Parse Query string;
  If query syntax invalid then
    Send error message to user "Invalid query syntax...";
  Else
    Send acknowledgement message to user "Searching...";
    Get Session library context;
    Get Session component set context;
    Set New component sets to be the set of all components in the Session library context
    that are also a member of the Session component set context and whose facet values
    meet the constraints specified in the Query string;
    If New component sets empty then
      Send message to user "No components found";
    Else
      Display Query string and number of members in New component set;
      Set Session component set context to be New component set
    End If;
  End If;
End;
```

SPS

D-115

# Steps 8 - 13:
# Clear Box Expansion

8. Conduct state analysis
9. Identify clear box subobjects
10. Classify subobjects
11. Define subobjects' relations
12. Define interaction paths
13. Conduct object interaction analysis



**SPS**

---

# Clear Box Expansion

### 8. Conduct state analysis



### 9. Identify clear box subobjects



### 10. Classify subobjects



### 11. Define subobjects' relations



**SPS**

D-116

# Clear Box Expansion (con't)

**12. Define interaction paths**



**13. Conduct object interaction analysis**



SPS
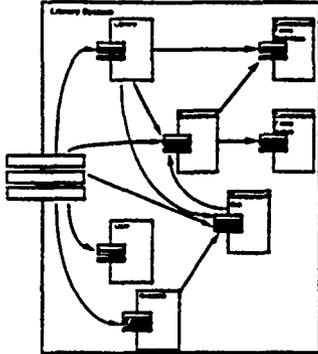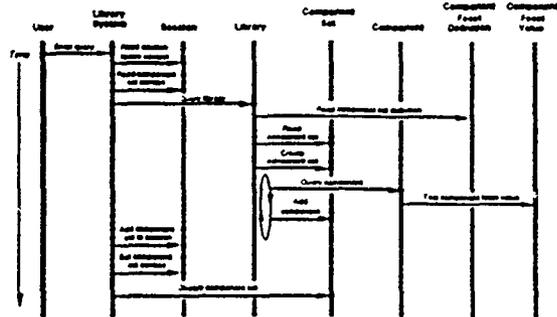
# Ada Box Structures Analysis Process

In the real-world, specifications are developed at many levels of abstraction simultaneously. The Ada Box Structures representations allow you to incrementally gather, annotate and verify system specifications.

SPS

## Incremental Expansion Process
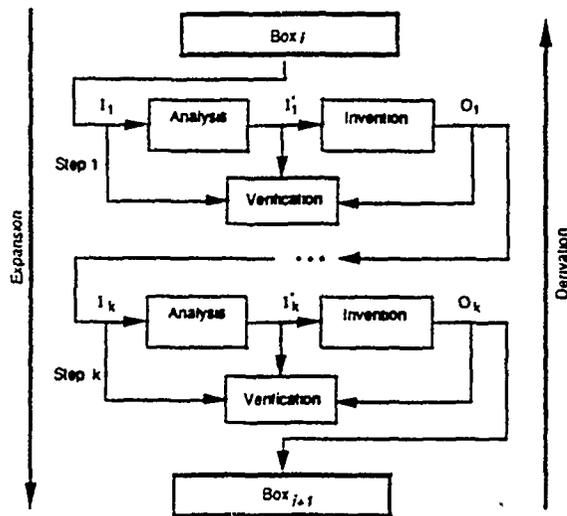


SPS

## Advantages of Ada Box Structures

- A small set of structuring concepts used repeatedly
- A rigorous process with verification
- Small steps of invention
- No restrictions placed on the order of elaboration (e.g., top-down vs. bottom-up)
- A "place notation" for documenting specification details
- Directly evolvable into an Ada object-oriented design, improving traceability and maintainability

SPS

*Technical Presentation 12*


*"A Prototyping Methodology*
*Applied to Tactical C2 Systems"*


.


*Mr. Martin Morel*
*Le Groupe CGI, Canada*

# Why Prototyping is needed...

**Conventional Methodologies**

- Impose too much responsibility on the developer with respect to the accuracy of system development.
- Deliverables prioritize heavy documentation rather than functioning and demonstratable software.
- User group review meetings become less productive and tend to be superficial as a means to gathering user requirements.

**User 's Role**

- Users have too little involvement in the development of the system.
- Lack a sense of "ownership" in the resulting system.
- Only "see" the system once it is developed - no opportunity for useful feedback during critical development stages
- System remains abstract in its early stages of design.

**Developer's Role**

- Experiences difficulty to accomplish his / her basic function:
    - -> to PRODUCE USEFUL information systems which respond to the USER'S REQUIREMENTS.
- Work serves to feed the methodology rather than the users.
- Often has to struggle to "learn the system".

**cgi**

---

# The E.C.C.O. Project

**Engineer Command and Control Operations
Mobility / Counter - Mobility Function
Canadian Land Forces**

Brief History of ECCO
Software Developments

2 Versions written to date using ...

| Conventional Methodology | | 3GL Technology |

Built with a minimum of user input

Resulting system:

| E-R Diagram of 8 entities, on 3 pages |

| Approx. 10 input screens, 10 reports |

| Only a very partial coverage of the requirements |

| Single - user, PC Based |

| Never completely accepted by the users |

**cgi**

# Requirements prior to Prototyping Approach

**Log Obstacle Tasks**

- Keep up to date specifications of Obstacle - Task activity descriptions:

| CODES | DESCRIPTIONS | STATIC QUANTITIES |
|-------|--------------|-------------------|

| ETC... |
|--------|

**Maintain Resource Descriptions**

Maintain descriptions of different types of resources used by Obstacle - Task types

**Result**

- Production of a large amount of documentation

- Little software produced for known requirements

- Requirements analysis has not gone in depth ... unknown requirements remain

cgi

# Requirements with Prototyping Approach

**(SUMMARY)**

**Obstacle - Task Planning**

- Plan and follow Mobility / Counter - Mobility tasks in a tactical situation. Support multi - plan operations.

| Mobility | Counter - Mobility |
|----------|--------------------|

| Survival | General Support |
|----------|-----------------|

**Resource and Work Schedule Calculation**

- calculate work schedules and all required resource types to carry out M / CM tasks

| Time | Personnel | Equipment | Vehicles |
|------|-----------|-----------|----------|

| Mines | Explosives | Fencing | Accessories |
|-------|-----------|---------|-------------|

**Stores Dump Management**

- Keep an update account of dump store contents and allocations (inventory control approach)

**Orders and Map Overlays Production**

- Produce detailed military orders and engineer plans, maintain a graphical representation of obstacle symbols overlays on a terrain map.

cgi

## E.C.C.O. Prototyping Status

**New Objectives**
- Ensure that user requirements are completely specified through the prototyping process.
- Use prototyping incremental approach to system's development.

> The Prototype becomes the System

**Software**
- Current architecture phase has already defined:

| Over 60 input screens | Over 80 Data Tables |
| Over 70 reporting functions | 6 Complex Calculational Functions |

**Documentation**
- With the prototyping approach, the documentation gradually builds up as the user requirements are refined. Each component of the system is documented using a data dictionary and and E - R modeling CASE tool.

> Data Model currently covers 60 entities, 7 modules displayed on 18 pages

**cgi**

---

## ECCO Technological Environment

**4GL DBMS**  Oracle with C interfacing

**Multi - User O.S.**  Unix

**Terrain Analysis Interfacing**  Geographical Information System on Graphic Workstations

**Methodology**  Protoguide - A Prototyping Methodology

**Tools**  Proto*SQL - Data Dictionary, mini Configuration Management tool, documentation generator

**cgi**

## Protoguide (introduction)

### What is ProtoGuide ?

- Development Guide
- Prototyping
- The prototype "becomes" the system

### Prerequisites

- 4th generation language
- Relational D.B.M.S.

### Advantages

- Improved user participation
- Reduced development costs
- Reduced operational costs
- Reduced duration
- Get the right system the first time

### Caution

- Manage modification requests
- Role of participants

cgi

## ProtoGuide

### Overview

General description of prototyping methodology

| Development phases | | Deliverables |

### Phases

The development is organized into phases: at the end of each phase. specific deliverables must be produced

| Preliminary Study | | Architecture |

| Prototyping | | Construction | | Installation |

### Deliverables

The deliverables consist of system components and end of phase reports

| Programs | | User documentation |

| System documentation |

| End of phase reports |

cgi

# Phases

| Preliminary Study | Actual Situation | Study and evaluate actual situation |
| | Definition | Set objectives. define the system |
| | Recommendation | Solutions. profitability. recommendation |
| Architecture | Planning | Plan overall development project |
| | Organization | Organize development project |
| | Standardization | Set development standards |
| Prototyping | Demonstration | Present an operational prototype |
| | Experimentation | Use the prototype to validate it |
| | Specification | Complete details for system construction |
| System Construction | Construction | Construct according to standards and specs. |
| | Inspection | Verify conformity to standards and specs. |
| | Preparation | Prepare installation |
| Installation | Verification | Detailed verification of correct operation |
| | Installation | Install for day to day usage |
| | Evaluation | Evaluate the system and the project |

cgi

# Overview (phases)

| Preliminary Study | Architecture | Prototyping | System Construction | Installation |
|---|---|---|---|---|
| | | | | Evaluation |
| | | | | Installation |
| | | | | Verification |
| | | | Preparation | |
| | | | Inspection | |
| | | | Construction | |
| | | Specification | | |
| | | Experimentation | | |
| | | Demonstration | | |
| | Standardization | | | |
| | Organization | | | |
| | Planning | | | |
| Recommendation | | | | |
| Definition | | | | |
| Actual Situation | | | | |

cgi

D-124

# Overview (phases)

ProtoGuide [hatched]        Conventional Methodology        [black]

| Preliminary Study | Architecture | Proto-typing | Anal-ysis | Construction | Installation |
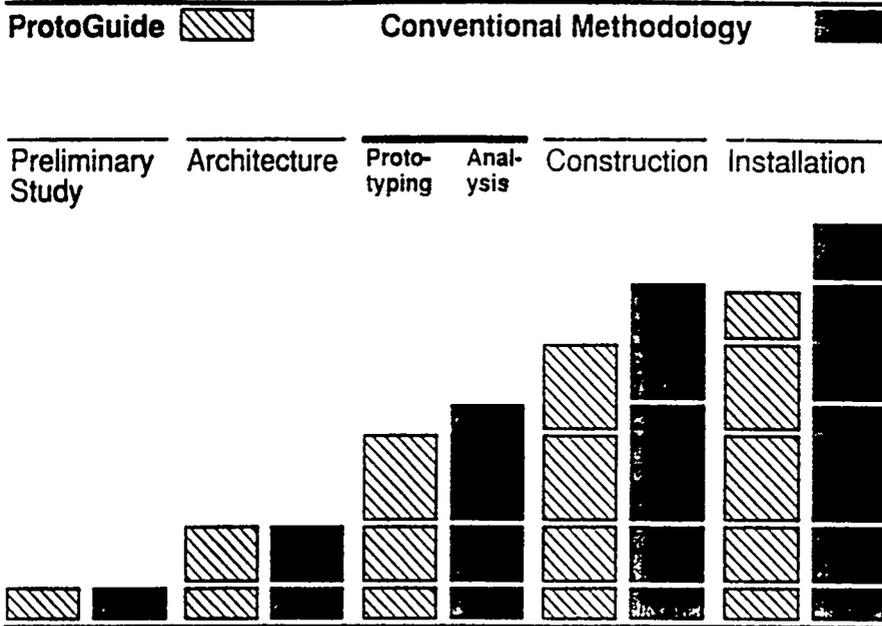
cgi

# Overview (Prototyping vs Analysis)

## Prototyping

Menus ...

**ECCO**

Minefields
Craters
Abattis
Bridges
Demolition

Screens ...

**ECCO**

Minefield Type: 3423
Size....:      300 sq. m.
Density...:  1 per 3 m.

| Mines Used | Qty. |
|---|---|
| 1231 | 10 |
| 5465 | 5 |
| 4446 | 35 |

Reports ...

**Dump Inventory**

| Store. | Desc. | Qty | Power |
|---|---|---|---|
| | Mine Resource Stores | | |
| 1231 | Conv. Mines | 13 | 500 |
| 5465 | Scatt. Mines | 50 | 350 |

Total  240KG

User Guide ...

Conv. Minefields Editor

**Validations**

| Mine Field | The mine field type may be any 4 char. code. |
|---|---|
| Mine Type | The mine type resource must exist in the dump |

**Calculations**

| Density | When qty. is modified, the total density of the field is recalculated as Size x Qty. x Density. |
|---|---|

## Analysis
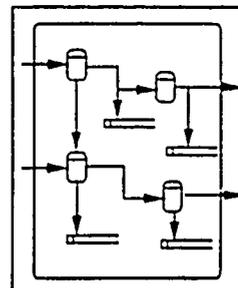
**Specifications**

| Mine - Field | The mine field code must exist in table OBS001 |
|---|---|
| Mine - Type | The mine type resource code must be validated . |
| Density | When the mine qty. is modified recalculate the total density of the mine field using the mine qty and the corresponding standard density of the mine field. The exact formula depends on . . |

cgi

D-125

## Deliverables

### Programs

| Menus |
|---|
| Interactive Programs |
| Reports |
| Batch Processing |
| Data Base Management |

### User Documentation

| System Overview |
|---|
| Training Guide |
| User Guide |
| Quick Reference Guide |
| Reference Guide |

### System Documentation

| Development Guide |
|---|
| Integrated Tests Guide |
| Conversion Guide |
| Installation Guide |
| Maintenance Guide |

### End of Phase Reports

| Preliminary Study |
|---|
| Architecture |
| Prototyping |
| System Construction |
| Installation |

cgi

## Interactive Programs

| Preliminary Study | Actual Situation | |
|---|---|---|
| | Definition | |
| | Recommendation | |
| Architecture | Planning | Summary description of programs |
| | Organization | |
| | Standardization | |
| Prototyping | Demonstration | Operational programs (function) |
| | Experimentation | Validate using real data |
| | Specification | Navigation, validation, perform.. messages |
| System Construction | Construction | Build according to standards and specs. |
| | Inspection | Verify conformity to standards |
| | Preparation | |
| Installation | Verification | Verify correct operation |
| | Installation | Install in production environment |
| | Evaluation | |

cgi

## Reports

| | | |
|---|---|---|
| Preliminary Study | Actual Situation | |
| | Definition | |
| | Recommendation | |
| Architecture | Planning | Summary description of reports |
| | Organization | |
| | Standardization | |
| Prototyping | Demonstration | Selection, sorting, report data layout |
| | Experimentation | Verify usefulness using real data |
| | Specification | Specify volumes, frequencies |
| System Construction | Construction | Performance |
| | Inspection | Verify conformity to standards |
| | Preparation | |
| Installation | Verification | Verify correct operation |
| | Installation | Install in production environment |
| | Evaluation | |

cgi

## User's Guide

| | | |
|---|---|---|
| Preliminary Study | Actual Situation | |
| | Definition | |
| | Recommendation | |
| Architecture | Planning | |
| | Organization | |
| | Standardization | Specify user interface standards |
| Prototyping | Demonstration | Describe prototype's processes and data |
| | Experimentation | Verify accordance with the prototype |
| | Specification | Specify all process details |
| System Construction | Construction | |
| | Inspection | Verify conformity with standards |
| | Preparation | |
| Installation | Verification | Verify conformity with system |
| | Installation | |
| | Evaluation | |

cgi

# ECCO Example - Menu and Screen

**Preparation Material for User Group Meeting #1**

- Menu & Menu Documentation

- Conventional Minefields Editor Screen

**Meeting Notes**

- User Group Meeting #1 - Conventional Minefields Editor

- User Group Meeting #2 - Conventional Minefields Editor

**Material Prepared after User Group Meeting #2**

- Menu and Menu Documentation

- Conventional Minefields Editor
        - User Documentation
        - Developer's Documentation

cgi

---

DREV - ECCO SOFTWARE PROTOTYPE                                           OBSMEN - 1
                     Obstacle Menu

Summary          Summary description
                 The obstacle module menu constitutes editing and listing capabilities required for the
                 maintenance of engineer standard obstacle class and types. Presently, these obstacles
                 focus on counter-mobility obstacle tasks. Future requirement analysis sessions will be
                 required to expand on other aspects of engineer activities.

Menus            Displayed Menu

                 ┌── OBSMEN ──── Defense Research Establishment Valcartier ───────
                 │ m │              Obstacle Menu                           _____
                 │   │
                 │
                 │
                 │
                 │                    Obstacle Class Editor
                 │
                 │
                 │
                 │
                 └─────────────────────────────────────────

| Options | **Summary description of menu options**<br>This section presents a summary description of each option presented in the menu |
|---|---|
| Obstacle Class Editor | The obstacle class editor is actually composed of several screens. The first of these allows the user to define and point to a "class" of obstacles which group together obstacles of a same type. By pointing to class, the user can "expand" to a secondary screen which contains the specific input data required to define an obstacle type within that class. The supported obstacle classes are: Abattis, Peacetime, Bridge Demolition, Craters, Other Demolition, Minefields, Anti-Tank Ditches and Fences. |

| Screens | **Screens displayed during processing** |
|---|---|
| Page 1 | |

```
┌── OBEOBS ──────────── ECCO Software Prototype ───────────┐
│ m│                    Minefield Data entry           Page 7/10 │
│                                                               │
│ Type   Depth    Unit of    Density  Placement  No of   Total      Stopping │
│                 Measure              Speed      rows    Density    Power    │
│ [F1]   [F2]__   [F3]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F3]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F9]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F3]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F5]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F9]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F8]       [F3]_     [F4]_      [F5]_   [F6]_       ___      │
│ [F1]   [F2]__   [F3]       [F3]_     [F4]_      [F5]_   [F5]_       ___      │
└───────────────────────────────────────────────────────────┘
```

D-129

*User Group Meeting 1*

CM

Screens | Screens displayed during processing



— ECCO Software Prototype —
Conventional Minefield Data entry — Page 7/10

| Type (1) | Width | Unit of Measure | Density (%) | Placement Speed | No.of Rows (3) | Total Density | Stopping Power (%) |
|---|---|---|---|---|---|---|---|
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |
| [F1] | [F2] | [F8] | [F3] | [F4] | [F5] | [F6] | ___ |

*"A" "G"*

*add - description*

*1st sec mines/metre*

*en mines/_*

*Fence Type*
*Fence equipment*

*add - méthode de pose "Laying Method"*
- *mece*
- *manual*
- *surface*
- *buried*
- *air*

*- 3 types of mines per mine*
  *- type*
  *- density*

---

*USER GROUP MEETING 2*

Screens | Screens displayed during processing



— ECCO Software Prototype —
Conventional Mine Field Obstacles — Page 7/11

| Type | Description | Density (/meter) | No.of Rows | Stopping Power | Placement Speed UOM | Laying Method Type | Description |
|---|---|---|---|---|---|---|---|
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |
| [F1] | [F2] | [F3] | [F4] | [F5] | [F6] [F7] | [F8] | [F9] |

Mines Used

| Type | Description | Qty |
|---|---|---|
| [V1] | [V2] | [V3] |
| [V1] | [V2] | [V3] |
| [V1] | [V2] | [V3] |

Fencing Equipment Used

| Type | Description | Qty |
|---|---|---|
| [W1] | [W2] | [W3] |
| [W1] | [W2] | [W3] |
| [W1] | [W2] | [W3] |

*Meth.*
*Man. Surf.*
*Man. Bur.*
*Mech. Surf.*
*Mech. Bur.*
*Air*

*take out description soni if abreviation soni clear enough*

*Stopping Power — en 0 et 100%*
*- perc. chance of vehicle ? mot some thing.*

*Tech - Type*
*- Desc.*
*- Weight*
*- Stopping Power*
*- Length*
*- Laying Method.*

*Result*
*- Tree*
*- Mines*
*- Equip.*

. .

| Summary | Summary description |
|---|---|

| Menus | Displayed Menu |
|---|---|

```
┌─── ENGMEN ──── Defense Research Establishment Valcartier ────────┐
│ ┌──┐                     Engineer Tasks Module              _____ │
│ │ m │                                                              │
│ └──┘                                                              │
│                                                                  │
│                                                                  │
│                                                                  │
│                                                                  │
│                                                                  │
│                  Counter Mobility Tasks Editor                   │
│                  Mobility Tasks Editor                           │
│                  Survival Tasks Editor                           │
│                  General Support Tasks Editor                    │
│                                                                  │
│                                                                  │
│                                                                  │
│                                                                  │
│                                                                  │
└──────────────────────────────────────────────────────────────────┘
```

.

| Options | Summary description of menu options |
|---|---|
| | This section presents a summary description of each option presented in the menu |
| Counter Mobility Tasks Editor | The obstacle class editor is actually composed of several screens. The first of these allows the user to define and point to a "class" of obstacles which group together obstacles of a same type. By pointing to class, the user can "expand" to secondary screens which contain the specific input data required to define an obstacle type within that class. The supported obstacle classes are: Abattis, Peacetime, Bridge Demolition, Craters, Other Demolition, Conventional Minefields, Scatterable Minefields, Anti-Tank Ditches, Fences and Booty Traps. Another generic obstacle type has also been included called "ZONES". These types are used by high level command units to plan entire field zones on which engineer counter-mobility activites are to take place. |
| Mobility Tasks Editor | The Mobility Tasks Editor is used to specify and document the resource requirements for engineer Mobility tasks. Generally, these tasks are classified as: obstacle breaching, route maintenance construction, bridging, and river crossing. |
| Survival Tasks Editor | The Survival Tasks Editor is used to specify and document the resource requirements of engineer survival tasks. Generally, these pertain to ground digging activities such as trenches and fortification. |
| General Support Tasks Editor | The General Support Tasks editor is used to specify and document the requirements of various general support activities falling under the responsibilities of engineers. Examples are: EOD, water supply, diving, facilities construction. |

Summary

**Summary Description of the Processing**
The obstacle class editor is actually composed of several screens. The first of these
allows the user to define and point to a "class" of obstacles which group together
obstacles of a same type. By pointing to class, the user can "expand" to secondary screens
which contain the specific input data required to define an obstacle type within that
class. The supported obstacle classes are: Abattis, Peacetime, Bridge Demolition, Craters,
Other Demolition, Conventional Minefields, Scatterable Minefields, Anti-Tank Ditches,
Fences and Booby Traps. Another generic obstacle type has also been included called
"ZONES". These types are used by high level command units to plan entire field zones on
which engineer counter-mobility activites are to take place.

...Page 7

```
┌─ OBEOBS ──────── ECCO Software Prototype ──────────┐
│ ⊓ │            Conventional Mine Field Obstacles        │ Page 7/11 │
│                                                                      │
│ Type Description   Density No.of Stopping Placement      Laying Method│
│                    (/meter) Rows Power   Speed UOM  Type  Description │
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ [F1] [F2]_____   [F3]_   [F4]_ [F5]_   [F6]_ [F7] [F8]____  [F9]____│
│ ───────Mines Used───────      ───────Fencing Equipment Used───────   │
│ Type    Description    Qty    Type    Description         Qty        │
│ [V1] [V2]_____ [V3]_      [W1] [W2]_____ [W3]_             │
│ [V1] [V2]_____ [V3]_      [W1] [W2]_____ [W3]_             │
│ [V1] [V2]_____ [V3]_      [W1] [W2]_____ [W3]_             │
└──────────────────────────────────────────────────────────────────────┘
```

Conventional Minefields
Table

The Conventional Minefields table is used to store the basic technical specifications of
engineer standard conventional minefield types. These types are assigned standard codes
used to quickly and uniquely identify them when assigning a obstacle-task.

[F1]Obstacle Type

CHAR 4
The conventional minefield type code is a four character field used to uniquely identify
a standard conventional minefield configuration. This code is entered through the
"Conventional Minefields Editor" and its maintenance is the responsibility of the system
pilot or administrator.

[F2]Obstacle
Description

CHAR 30
The conventional minefield description is a free text field used to associate a short
description of a standard conventional minefield to the minefield type code. This short
description is then displayed with the minefield type in other parts of the ECCO system
to enhance the significance of the minefield type mnemonic code.

[F3]Minefield Density

NUMBER 5
The minefield density describes the number of conventional mines placed per <<meter>> in a
standard conventional minefield configuration.

[F4]Number of Rows in
Minefield

NUMBER 5

The number of rows of conventional minefields that this type of minefield obstacle type
contains.

[F5]Stopping Power

CHAR 5
The stopping power is a percentage value between 0 and 100 indicating the probability of
stopping enemy vehicles from passing through the minefield.

[F6]Minefield Placement
Speed

NUMBER 6

The placement speed denotes the time required to set up this type of minefield obstacle.
Usually, this value is expessed in terms of section-hours or troop-hours.

*Counter Mobility Tasks Editor*

(F7)Minefield Placement
Unit of Measure

CHAR 4
The placement speed unit of measure is directly associated with the minefield placement speed. It is a 4 character code used to indicated the unit of measure used to describe the placement speed (usually in section-hours or troop-hours). This code is validated from the ECCO "Unit Measures Table".

(F8)Laying Method

CHAR 10
The laying method describes the principle method used to lay the conventional mines for a given conventional minefield. Its values are either manual-surface (MASU), manual-buried (MABU), mechanical-surface (MESU), mechanical-buried (MEBU).

Obstacle Resources
Table

The obstacle resource table holds the "type" codes of all consumable and non-consumable resources required by obstacle types. Used by the Obstacle editors, it defines for each obstacle type of a given class (ex: Conventional Mines, Craters etc.) the resource types required to put that obstacle into place.

(V1)Resource Type

CHAR 4
The resource type denotes a consumable or non-consumable resource code required to place an obstacle. The resource type code is defined within a specific obstacle class.

(V3)Resource Quantity

NUMBER 5
The resource quantity field denotes the quantity of a specific consumable or non-consumable resource required to place an obstacle of a given type. Its value is expressed in terms of the basic units of measure defined for a given resource.


DREV - ECCO     CODE VALUE TABLES FOR SUMMARY FORM DOCUMENTATION
ProtoSQL Form Documenter

| Field attributes | Key Triggers | Other Triggers |
|---|---|---|
| A Database field | a ClrBlk | A Post-Change |
| B Primary-key | b ClrFrm | B Pre-Field |
| C Copy field value | c ClrRec | C Post-Field |
| from block fill-in exist | d Commit | D Pre-Query |
| D Copy field value | e CQuery | E Post-Quewry |
| from field fill-in exist | f CreRec | F Pre-Insert |
| E Default value exist | g DelRec | G Post-Insret |
| F Displayed | h DupFld | H Pre-Update |
| G Input allowed | i DupRec | I Post-Update |
| H Query allowed | j EntQry | J Pre-Delete |
| I Update allowed | k ExeQry | K Post-Delete |
| J Update if null allowed | l Exit | L Pre-Record |
| K Mandatory | m ListVal | M Post-Record |
| L Fixed length | n Menu | N Pre-Block |
| M Auto skip | o NxtBlk | O Post-Block |
| N No echo | p NxtFld | P Pre-Form |
| O Auto help | q NxtKey | Q Post-Form |
| P Uppercase | r NxtRec | |
| Q List of values exist | s NxtSet | |
| R Low value exist | t PrvBlk | |
| S High value exist | u PrvFld | |
| T Help message exist | v PrvRec | |
| | w Others | |

Forms Technical Description (partial compilation)

| Block Field | Type | Len | Field Attributes | Key Triggers | Key-Fx | Other Triggers |
|---|---|---|---|---|---|---|
| OB_MINEFIELD | | | | ..............o. | .......... | ............... |
| MINE FIELD TYPE | CHAR | 4 | A....FGHI.....O....T | .............m. | .......... | A............... |
| DESCRIPTION | CHAR | 30 | A....FGHI........... | ................. | .......... | ............... |
| DENSITY | RNUMB | 5 | A....FGHI........... | ................. | .......... | ............... |
| NO OF ROWS | RNUMB | 5 | A....FGHI........... | ................. | .......... | ............... |
| STOP POWER | RNUMB | 5 | A....FGHI........... | ................. | .......... | ............... |
| PLACEMENT SPEED | RNUMB | 5 | A....FGHI........... | ................. | .......... | ............... |
| PLACEMENT SPEED_UCM | CHAR | 4 | A....FGHI........... | ................. | .......... | ............... |
| LAYING METHOD | CHAR | 10 | A....FGHI........... | ................. | .......... | ............... |
| L_M DESCRIPTION | CHAR | 10 | .....FGHI........... | ................. | .......... | ............... |
| CONV_MINE_RESOURCES_1 | | | | ..............o....t. | .......... | ..............N. |
| RESOURCE_TYPE | CHAR | 4 | A....FGHI.....O....T | .............m. | .......... | A............... |
| DESCRIPTION | CHAR | 25 | .....F.............. | ................. | .......... | ............... |
| RESOURCE_QTY | RNUMB | 5 | A....FGHI........... | ................. | .......... | ............... |
| OBSTACLE_CLASS | CHAR | 2 | A.CO............... | ................. | .......... | ............... |
| OBSTACLE_TYPE | CHAR | 4 | A.CO............... | ................. | .......... | ............... |
| RESOURCE_CODE | CHAR | 1 | A.CO............... | ................. | .......... | ............... |
| RESOURCE_CLASS | CHAR | 2 | A.CO............... | ................. | .......... | ............... |
| FENCING_EQUIPMENT_1 | | | | ..............o....t. | .......... | ..............N. |
| RESOURCE_TYPE | CHAR | 4 | A....FGHI.....O....T | .............m. | .......... | A............... |
| DESCRIPTION | CHAR | 26 | .....F.............. | ................. | .......... | ............... |
| RESOURCE_QTY | RNUMB | 5 | A....FGHI........... | ................. | .......... | ............... |
| OBSTACLE_CLASS | CHAR | 2 | A.CO............... | ................. | .......... | ............... |
| OBSTACLE_TYPE | CHAR | 4 | A.CO............... | ................. | .......... | ............... |
| RESOURCE_CODE | CHAR | 1 | A.CO............... | ................. | .......... | ............... |
| RESOURCE_CLASS | CHAR | 2 | A.CO............... | ................. | .......... | ............... |

# Conclusions

Methodology

User Group Profile

Required Tools

Participant's Objectives

cgi

# Methodology as Implemented in ECCO

**Participant's Acceptance**

- Once defined, the methodology is presented and explained to each participant:

| Project Sponsor | Users | Developers |

| Parallel Projects |
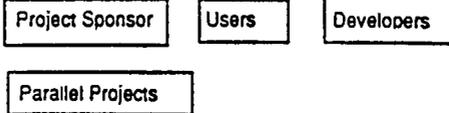
**User Group**

- The project sponsor appoints a core user group which has as a specific task the responsibility of actively participating in the development of the system.

**Technology**

- The implementation of the prototyping process requires the rapid installation of proven technologies and tools such as screen and code generation. This allows the developers to spend more time with the users, refining requirement needs rather than struggling with difficult and tedious programming in the early phases of the project.

cgi

---

# User Group Profile

**Location**

- Based in Canadian Forces Base Valcartier, Québec, CANADA 5th Engineer Reg. of Canada. This is the largest engineering base in Canada, the 2nd largest in the Canadian Land Forces

- Close proximity to the development team at D.R.E.V.

**Active Participants**

- Active participants rank from Major (project sponsor), Captain (engineer commander), sargeants and corporals

- Participants were chosen because they represent the typical profile of end users and have vast experience in engineer tactical operations.

**External Participants**

- A multi-level user group is essential to the success of the project. It therefore also includes higher ranking command officers to ensure that all vertical engineering requirements are met.

cgi

## Required Tools

E - R Diagram Data Modeling

Functional Decomposition Diagramming

Interactive Program Prototyping (4GL based)

Report Prototyping

Documentation Generation

Data and Component Dictionary

cgi

## Participant's Objectives

**Project Manager**
- User's Satisfaction
- Productivity
- Deliverables
- Reduced Costs
- Realistic Work Schedule
- Meet Requirements
- Technology

**Developer**
- More accurate analysis work
- Functioning and Valid Software
- Technological Challenge
- Recognition
- Improved professional and managerial skills

**User**
- Get a complete and correct system the first time
- Enhanced implication in development
- Rapid contact with technology
- Rapid access to deliverables
- Concrete results
- Responsibility and ownership of system

cgi

# Technical Presentation 13

## "Requirements Engineering Testbed"

Mr. William E. Rzepka
Rome Air Development Center, USA

WHAT ARE REQUIREMENTS?


REQUIREMENTS ARE PRECISE STATEMENTS OF NEED INTENDED

TO CONVEY UNDERSTANDING ABOUT A DESIRED RESULT


EXTERNAL CHARACTERISTICS

CONSTRAINTS

PERFORMANCE

RELIABILITY

SAFETY

COST

MODEL OF WHAT IS NEEDED

STATEMENT OF PROBLEM TO BE SOLVED


## IT PAYS TO CATCH ERRORS EARLY



PHASE IN WHICH ERROR IS DETECTED

# CORE

CONTROLLED REQUIREMENTS EXPRESSION



---
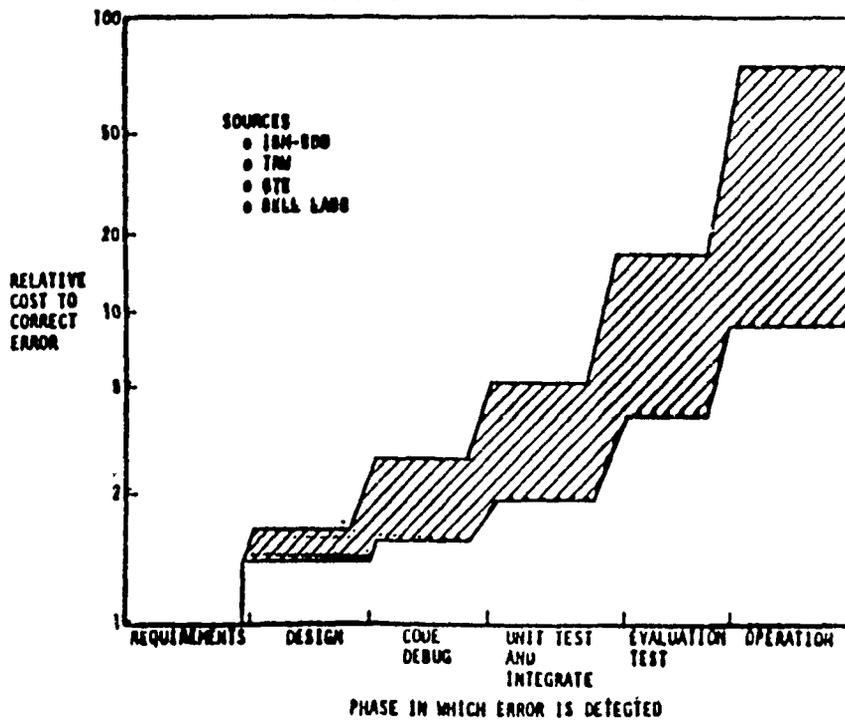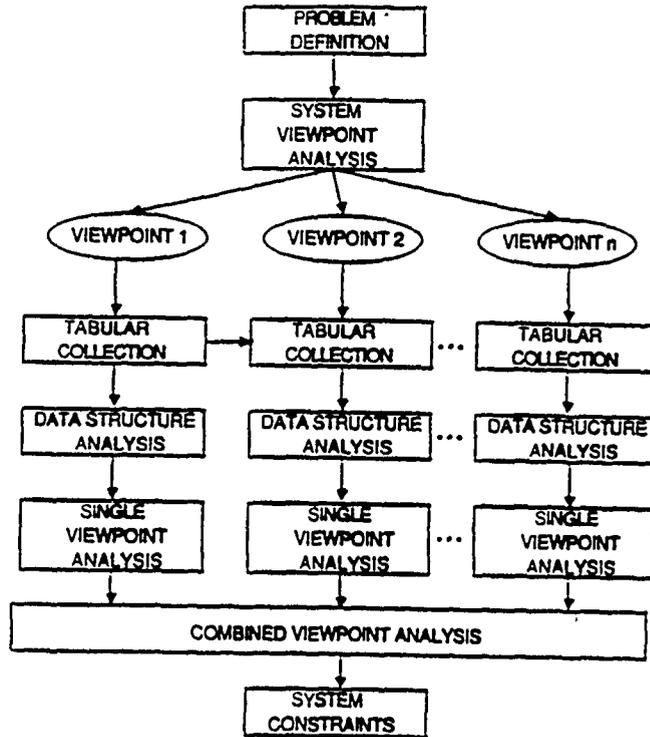
# RAPID PROTOTYPING SYSTEM

# CORE

### CONTROLLED REQUIREMENTS EXPRESSION



---
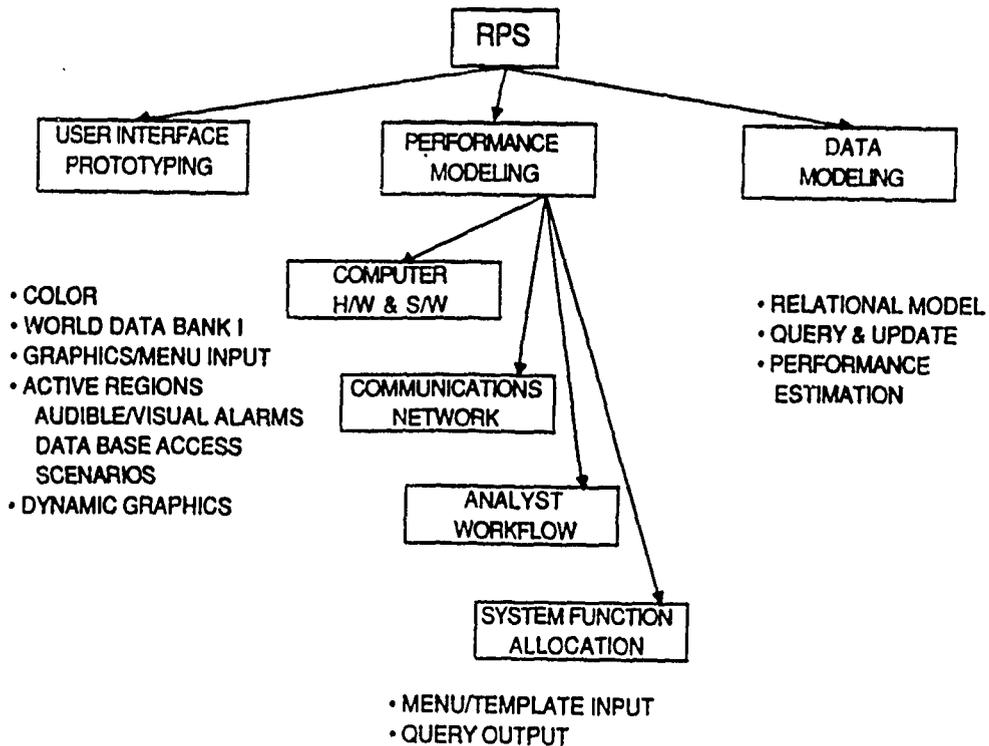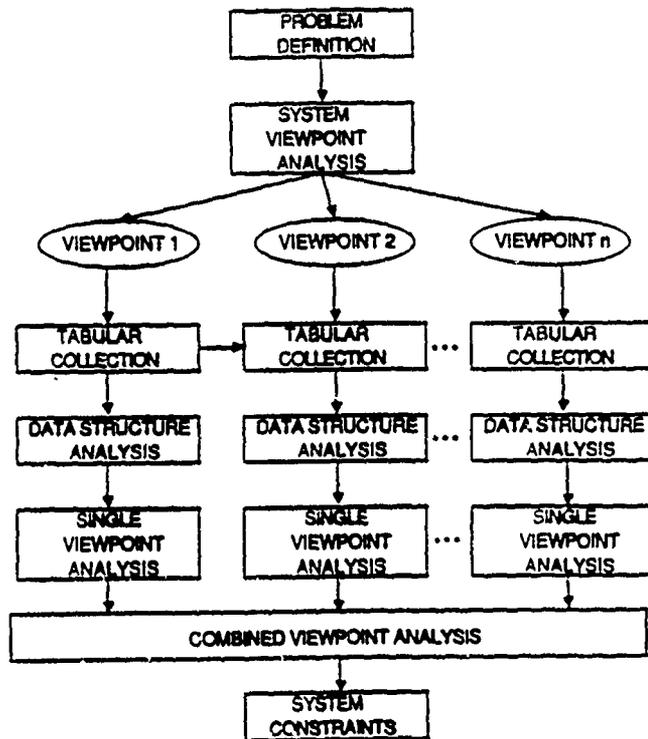
# RAPID PROTOTYPING SYSTEM
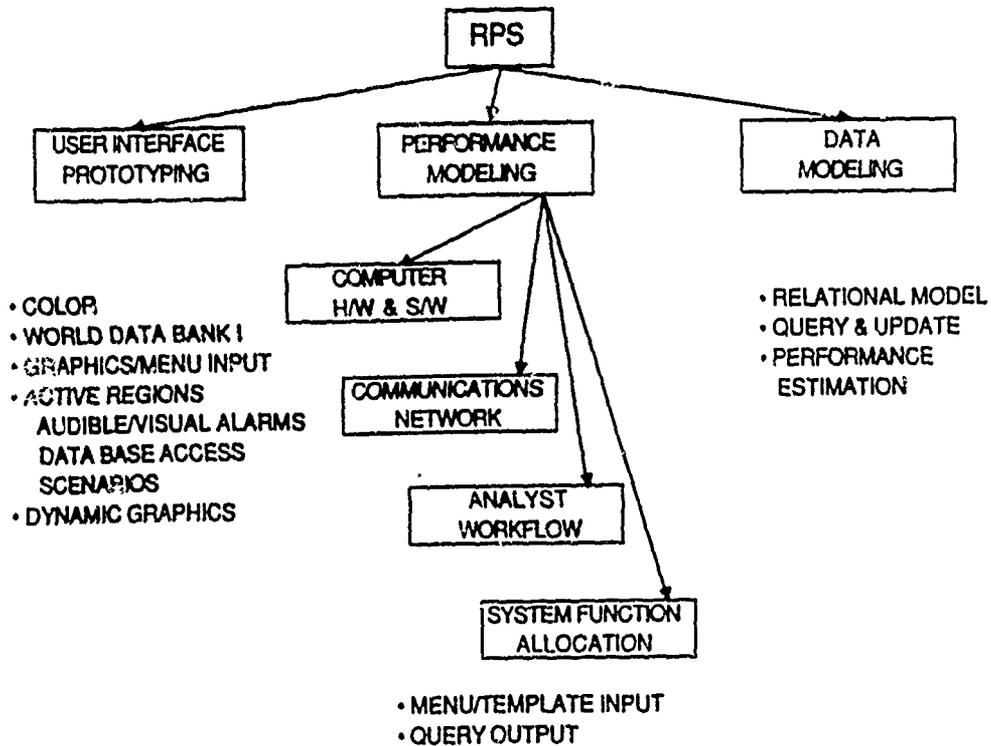
# PROTO

**OBJECTIVE**
RAPIDLY SPECIFY A PROGRAM THAT EXECUTES SPECIFIC
TARGET SYSTEM FUNCTIONS

**APPROACH**
VERY HIGH LEVEL GRAPHICAL LANGUAGE FOR
INTERCONNECTING SOFTWARE MODULES

**ENVIRONMENT SUPPORTING·**
STEPWISE REFINEMENT
CONVENTIONAL PROGRAMMING
REUSE OF APPLICATION-SPECIFIC MODULES
EXECUTABLE
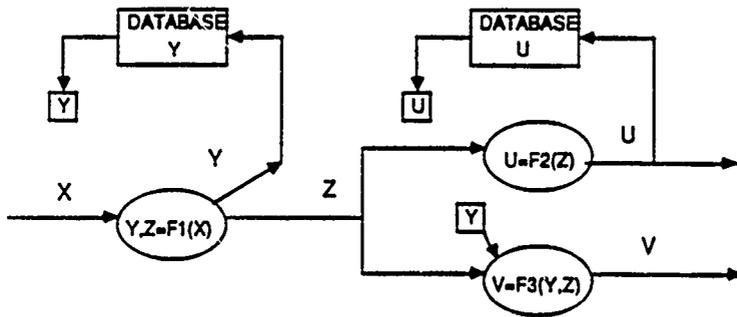


---

# RET STATUS

## RAPID PROTOTYPING SYSTEM
## TECHNOLOGY TRANSITION

**MARTIN MARIETTA INTITIATIVES:**

MX MISSILE BASING DETERMINATION

DMA (CLASSIFIED)

ORB (CLASSIFIED)

SMALL ICBM LAUNCH CONTROL

FTS 2000 COMM SYSTEM STUDY

NUCLEAR POWER PLANT, OAK RIDGE

BUREAU OF LAND MANAGEMENT

**RADC INITIATIVES:**

SOFTWARE PRODUCTIVITY CONSORTIUM

ESD/AVS C2 EVALUATION FACILITY

US ARMY CECOM

SPACE DIVISION/AEROSPACE CORP

NADC/WARFARE SYSTEMS ANALYSIS DEPT

NORAD/GRANITE SENTRY SPO

# Requirements Engineering Testbed

| INDIVIDUAL TOOL INTERFACE STYLES | | |
|---|---|---|
| CORE INTERFACE | PROTO INTERFACE | RPS INTERFACE |
| CORE | PROTO | RPS |

| DATA BASE | CORE LOGICAL DATA | DATA BASE | PROTO LOGICAL DATA | DATA BASE | RPS LOGICAL DATA |
|---|---|---|---|---|---|

# 1992 REQUIREMENTS ENGINEERING TESTBED

```
┌─────────────────────────────────────────────────────────┐
│               COMMON USER INTERFACE                      │
├──────────────────┬───────────────────┬──────────────────┤
│  CORE INTERFACE  │  PROTO INTERFACE  │  RPS INTERFACE    │
└──────────────────┴───────────────────┴──────────────────┘
```

| CORE | PROTO | RPS |
|------|-------|-----|

| LOCAL DATA | LOCAL DATA | LOCAL DATA |
|------------|------------|------------|

```
┌─────────────────────────────────────────────────────────┐
│              OBJECT ORIENTED DATA BASE                   │
└─────────────────────────────────────────────────────────┘
```

---
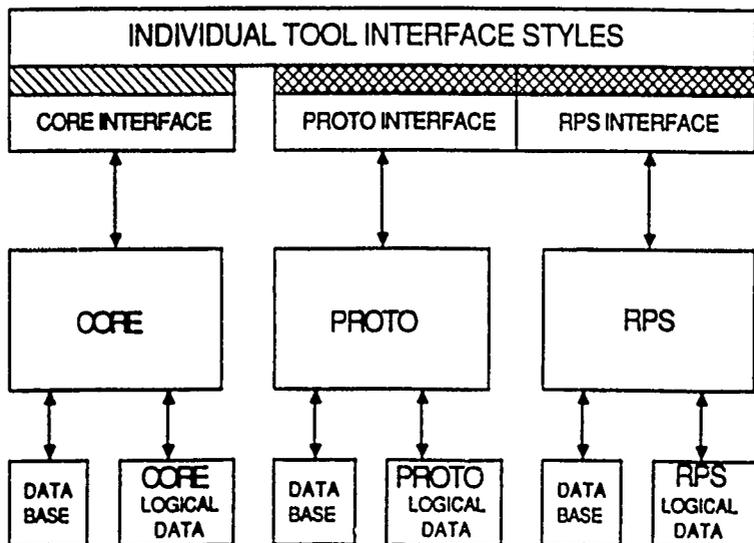
# REED ENHANCEMENTS TO THE RPS

- **USER INTERFACE MODELING**

    - **INTEGRATE INDIVIDUAL TOOLS INTO SINGLE INTERFACE**

    - **PROVIDE INTEGRATED DYNAMIC CAPABILITY**

- **PERFORMANCE MODELING**

    - **PROVIDE GRAPHIC INTERFACE TO MODELS**

This page is intentionally left blank.