FINAL TECHNICAL REPORT

Contract No. F30602-96-C-0274

9/30/96 to 8/15/98

Amount of Contract: $285,612.00

Principal Investigator: Prof. Barry Boehm,
USC Center for Software Engineering
Phone: (213) 740-8163

Title:

KBSA Life Cycle Evaluation

Authors:
Barry Boehm, A. Winsor Brown
USC Center for Software Engineering
Prasanta Bose
George Mason University

Volume I
Sections A-I

A.1 Table of Contents

A.2   List of Figures

A.3   List of Tables

B.   **Executive** Summary

The *objective* of this research effort has been to develop and validate technical approaches for evaluating the effects of Knowledge Based Software Assistant (KBSA) process concepts and technology on software development effort and schedule, and to use the approaches to perform comparative evaluations of KBSA and other sources of software technology.

The research *approach* has involved four tasks.

Task 1. *Characterize KBSA and other sources of software technology in the context of recent and emerging software trends.* We provide a summary of KBSA technology, concentrating on the KBSA Advanced Development Model developed by Andersen Consulting. We also summarize two other comparative sources of software technology: the commercial marketplace and the DARPA/AFRL Evolutionary Design of Complex Software (EDCS) program. To our knowledge, this is the first software technology payoff analysis done in comparison to concurrent advances in commercial software technology.

Task 2. *Develop models and an evaluation framework for assessing the effects of KBSA and other sources of software technology on software development effort and schedule.* Our recently developed and calibrated COCOMO II model provides an approach for evaluation based on the effects of alternative software technologies on the model's effort-driver parameters. The model's calibration to over 100 1990's software projects also provides a 1990's baseline from which to evaluate the technologies' effects.

For assessing schedule effects, we have developed another model, CORADMO, for evaluating the effects of rapid application development (RAD). Our evaluation framework also includes a domain focus: DoD warfighting systems; and a particular evaluation example: a representative embedded, high-assurance, real-time (EHART) missile software project. We have also developed a spreadsheet version of the evaluation model. This enables technology decision makers to perform tradeoff and sensitivity analyses of alternative software technology investment strategies.

Task 3. *Use the models to evaluate KBSA, EDCS, and commercial technology with respect to the baseline.* The primary result is shown in Figure 1, which compares the technologies' relative effects on development effort, using relatively conservative assumptions. It shows that commercial technology is likely to reduce development effort of the EHART 1998 baseline project by a factor of 2.5 in 8 years (2006) and another factor of 3 in 15 years (2013). Relative to commercial technology, a fully-supported mix of KBSA and EDCS technologies could reduce development effort by another factor of 3 in 8 years and another factor of 6 in 15 years. These assessment results are consistent with such analyses of commercial software technology impact as [Bernstein, 1997].
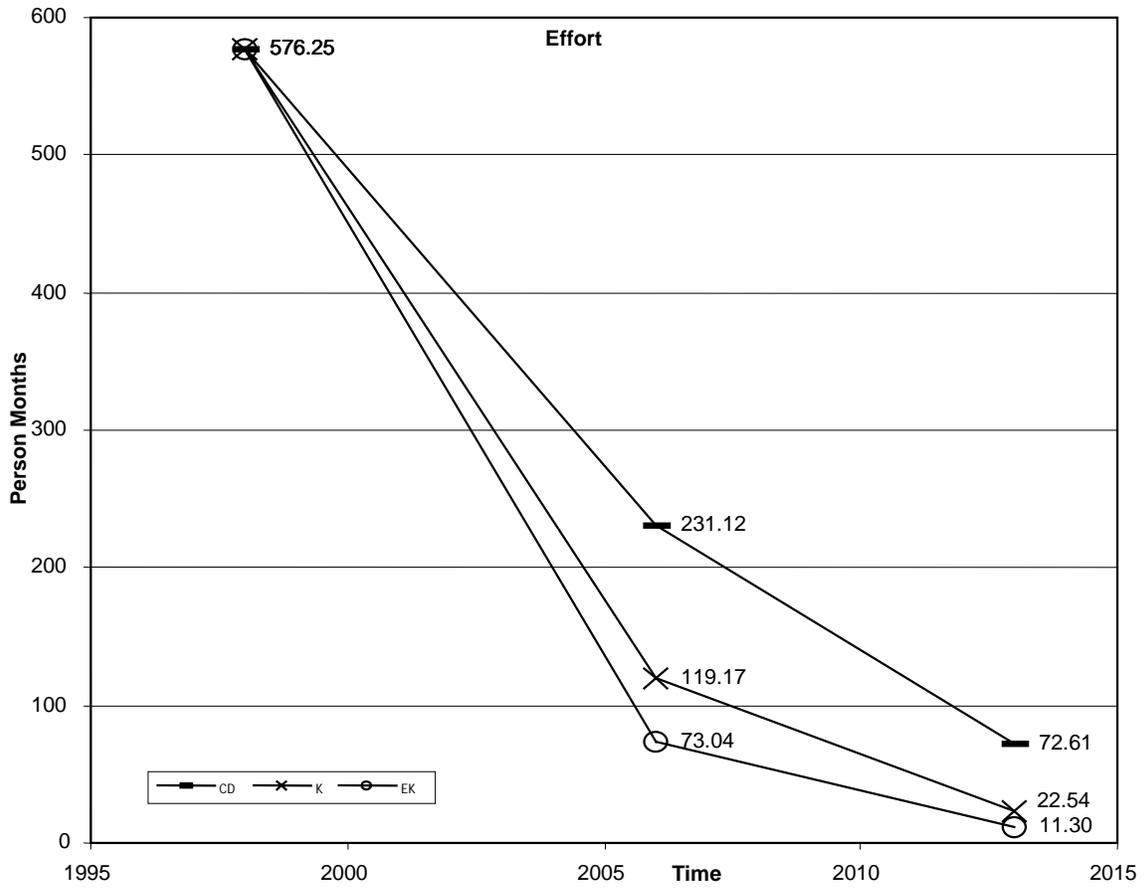
Figure 1. Effects of Commercial/Basic DoD (CD), KBSA (K), and EDCS/KBSA (EK) Technology on Effort

Task 4. Formulate study conclusions and recommendations.

The six primary conclusions are:

1. The KBSA Advanced Development Model was not sufficiently robust and scalable to provide appreciable benefits to critical DoD software projects.

2. However, the KBSA concepts, if otherwise realized, have strong potential for reducing costs and schedules for DoD-critical warfighting software applications, The key to realizing this potential is to develop the KBSA technology in the context of DoD warfighting domain knowledge.

3. The KBSA Decision Support area provides an attractive new direction for Air Force and DoD software technology investments.

4. EDCS technology investments and payoffs have both commonalities and complementarities with respect to KBSA technology investments and payoffs.

5. The Software Technology Impact Analysis spreadsheet model provides a useful tool for assessing alternative software technology assessment strategies.

6. There is no software technology silver bullet for DoD warfighting systems, including reliance on commercial technology.

The two primary recommendations are:

1. The Air Force and DoD should continue a strong level of investment in software technology, particularly for warfighting domains.

2. The software technology investments should be part of an integrated DoD software management/process/technology strategy to ensure DoD software supremacy, particularly in warfighting domains.

The conclusions and recommendations are not particularly surprising. They are quantitative counterparts of the qualitative conclusions and recommendations about the need for DoD investments in software technology in recent Air Force Scientific Advisory Board studies [SAB, 1995; SAB, 1997] and recent DoD-level studies [NAE, 1995; Fox et al., 1995; NRC, 1997]. What is surprising is that in response to this consistent set of signals, DoD has been reducing its level of investments in software technology.

C. Objectives of the Effort

The *objectives* of this research effort have been to develop and validate technical approaches for evaluating the effects of Knowledge Based Software Assistant (KBSA) process concepts and technology on software development effort and schedule, and to use the approaches to perform comparative evaluations of KBSA and other sources of software technology.

These objectives relate to the strong Air Force needs for improved software technology, particularly in areas not well served by commercial software technology. These needs are expressed in such Air Force Scientific Advisory Board studies as New World Vistas [SAB, 1995] and Air Expeditionary Forces [SAB, 1997]. Similar needs have been expressed at the DoD level by recent reviews of DoD/DARPA software technology programs [NAE, 1995; Fox et al., 1995], and the National Research Council "Ada and Beyond" report [NRC, 1997].

D. Approach

The research *approach* has involved four tasks:

Task 1. *Characterize KBSA and other sources of software technology in the context of recent and emerging software trends*. Section E.1 summarizes KBSA technology, concentrating on the KBSA Advanced Development Model developed by Andersen Consulting. It also summarizes two other comparative sources of software technology: the commercial marketplace and the DARPA/AFRL Evolutionary Design of Complex Software (EDCS) program. Details of the KBSA ADM evaluation are in Section M.

Task 2. *Develop models and an evaluation framework for assessing the effects of KBSA and other sources of software technology on software development effort and schedule*. Our recently developed and calibrated COCOMO II model provides an approach for evaluation based on the effects of alternative software technologies on the model's effort-driver parameters. The model's calibration to over 100 1990's software projects also provides a 1990's baseline from which to evaluate the technologies' effects.

Section E.2 summarizes COCOMO II and an additionally-developed model, CORADMO, for evaluating the effects of rapid application development (RAD) strategies on software development schedules. It also summarizes the evaluation domain focus -- DoD-critical embedded, high-assurance, real-time (EHART) projects -- and the evaluation timeframe: current 1998 baseline efforts and schedules on a representative DoD EHART project starting in 1998, and counterpart project efforts and schedules if the project were started in 2006 or 2013 with the benefit of commercial technology or well-supported KBSA and EDCS technology.

Section F provides a summary of a spreadsheet-based Software Technology Impact Analysis model, which can be used to evaluate other software technologies or to perform sensitivity analyses on the evaluations performed in this study. Details of the models are provided in Section J, K, and O.

Task 3. *Use the models to evaluate KBSA, EDCS, and commercial technology with respect to the baseline*. Section E.3 presents example worksheets showing the technology assessments in terms of their COCOMO II and CORADMO effort and schedule driver ratings, with rationales for each. The primary benefits of well-supported KBSA technology come from domain-KB-based applications generation and KB software decision aids, particularly in the EHART domain and other DoD warfighting domains, which are less supported by commercial technology. The primary benefits of well-supported EDCS technology come from domain architectures, general architecture and collaboration technology, and high-assurance software technology. The complete set of worksheets and spreadsheet calculations are in Section L.

Task 4. *Formulate study conclusions and recommendations*.

These are provided in Section E.4, and were summarized in the Executive Summary (Section B).

E. Discussion of Tasks

E.1 KBSA and Other Technologies

E.1.1 The 1983 KBSA Vision

The "Report on a Knowledge Based Software Assistant" [Green et al., 1983] provided a vision for achieving major improvements in software development and maintenance. It involved the following primary technical advances:

a. Automatic generation of desired software systems from formal specifications.

b. Life cycle maintenance and evolution of software via modification of specifications and re-derivation of the resulting software, rather than modifying the source code itself.

c. A life cycle process model organized around development of specifications, iteration of automatic program generation directives to achieve desired target software performance, and subsequently iterating the specifications to accommodate software changes.

d. Knowledge based "activity coordination" expertise built into software development and management tools, to ensure effective software acquisition and evolution management.

E.1.2 Developments Since 1983

E.1.2.1 Evolution of Conventional Technology Toward KBSA Vision

Since 1983, conventional software development technology has been undergoing significant changes, reorienting it more toward aspects of the KBSA vision. Large, powerful COTS software components (e.g., Microsoft Foundation Classes) now form the basis for many domain-specific applications generators, such as spreadsheets, business fourth generation languages (4GL's), and graphic user interface (GUI) builders. The process model for using these packages largely follows the KBSA process model (except generally for the lack of a performance tuning cycle), as it is much more natural and cost-effective to modify applications at the 4GL level than at the target-code level.

Concurrently, large software organizations such as AT&T, HP, DoD, and its contractors have been making significant investments in software product line management based on domain specific software architecture (DSSA)-based reusable components. The DSSA's are based on a domain engineering process, which is a mix of systems engineering and knowledge engineering focused on the applications domain of interest.

However, significant limitations remain with respect to the evolving conventional technology. Combinations of large COTS components are extremely hard to integrate, often leading to factors of 4-5 in budget and schedule overruns [Garlan et al., 1995]. Various commercial and government interface standardization initiatives have been formed to address this problem (e.g., OMG's CORBA, DoD's Defense Information Infrastructure–Common

Operating Environment (DII-COE)). Wrapper and glue-code technology has been developed to address the problem also, but the problem remains significant.

E.1.2.2 Evolution of KBSA Technology

The Rome Laboratory KBSA program began with a number of exploratory research efforts addressing the various facets of a KBSA capability. These facets included prototype KBSA tools supporting software project management, requirements acquisition and analysis, formal specification development, transformation of specifications into code, and performance optimization. Some additional research efforts focused on facets of an infrastructure capability such as tool integration, activity coordination, and communication coordination [White, 1991].

Concurrently with the later stages of the facets research, Andersen Consulting developed an exploratory Concept Demonstration version of an overall KBSA system. It demonstrated the integration of portions of the requirements acquisition, code development, and project management facets in an Air Traffic Control domain [De Bellis et al., 1992].

As the program has proceeded toward its ultimate objectives, it has spawned a number of productivity-enhancing spinoffs such as the Refine-based software reengineering and testing tools, and domain-specific applications generators covering portions of avionics, transportation planning, and military message processing applications. These have had significant productivity benefits for portions of a process or product, but await further capabilities to be able to address a full life cycle process or a full system product.

The KBSA activity-coordination area began with the work on automating configuration management and producing software project management decision aids. A number of additional exploratory software decision support aids have been produced in such areas as risk assessment, process model selection, review preparation, and schedule-slip assessment, e.g., [Bose et al., 1995]. More recent AFRL-sponsored work has been for decision support aids for DoD warfighting domains such as for security and survivability.

A major effort to integrate KBSA technology has been the KBSA Advanced Development Model described in the next section.

E.1.3 The KBSA Advanced Development Model

The KBSA Advanced Development Model developed as part of the Rome Laboratory's Knowledge-Based Software Assistant effort is aimed at improving software development productivity and software quality. The fundamental approach in achieving the above goal is providing automated support that mediates, automates and documents all activities throughout the development lifecycle for both individual developers and teams of developers. The challenge is building such computer based assistants as elaborated in the KBSA program vision [Green et al., 1983].

The key concepts to meeting the above challenge are based on the understanding that software development is a knowledge intensive activity. Creating large software based systems requires knowledge of the domain (typically multi-disciplinary in nature), knowledge of the process context, knowledge of existing components and hardware, and personnel resources. The

KBSA approach is then to provide means for capture and effective use of such knowledge with the goal that such use of such knowledge by the stakeholders will lead to timely production of high-quality software.

Given the above understanding, the key idea in the KBSA approach is exploiting artificial intelligence (AI) concepts and representations to capture and use knowledge. The different types of product specific knowledge are user requirements, system specifications, code, test scenarios and documentation. Process specific knowledge corresponds to the software development plans, resources and status of the project. Some major problems here are: a) integrated usage of the knowledge [Selfridge, 1992]; b) managing change; and c) managing complexity. Significant progress has been made in addressing the above problems individually [Johnson et al., 1991, Mi et al., 1990, Smith, 1991]. The ADM objective was to build on such advances to provide an integrated set of concepts and tools that address the problems within a single framework.

The goals of our analysis were restricted to evaluating the ADM concepts and support features. In particular we focused on: a) understanding the ADM conceptual model, b) analyzing ADM facilitated software development processes, c) analyzing ADM in the context of specific application development case study, and d) mapping ADM contributions into the parameters of a cost benefit evaluation of the ADM support features. This report documents our findings with respect to the above goals. Here is a summary of the findings:

1. The ADM environment framework has good technical concepts (model-view controller architecture; process-driven environment; persistent object base).

2. These concepts have several critical issues regarding the feasibility of their use on large, complex projects (scalability; overconstraining people-collaboration processes).

3. The ADM was not fully-enough implemented to resolve these issues.

4. Much of the ADM has been overtaken by commercial technology (e.g., Rational Rose).

5. The ADM has some good concepts such as the use of critics for software project decision assistance or conceptual debugging.

6. For the foreseeable future, KBSA technology will have more impact on software project costs and schedules if pursued in terms of specialized tool enhancements of commercial environments (e.g., domain-specific application generators; critics or software project decision aids), rather than in terms of alternative environments.

E.1.4 Related Technology

Most technology impact analyses simply compare the downstream effect of applying their technology to the results achievable with current practice. This is generally unsatisfactory, because it leaves the reader with no way to assess the gains via the technology being analyzed to the gains which would have accrued via commercial technology evolution, non-technology (e.g., process and management) gains, or via alternative technologies.

With our COCOMO-oriented baseline of current practice and technology assessment approach, we can do better than this. In addition to assessing the downstream effect of KBSA technology, we also assess the counterpart downstream effects of commercial software technology, DoD non-technology initiatives (e.g., integrated product teams, process maturity, existing-asset reuse initiatives), and the major alternative DoD software technology initiative, the DARPA/AFRL Evolutionary Design of Complex Software (EDCS) program.

Commercial technology can reduce software costs and schedules in a number of ways captured by COCOMO cost drivers. Moore's Law increases in hardware speed and memory capacity can reduce execution time and storage constraints. Commercial off-the-shelf (COTS) software, class libraries, and open architecture frameworks can reduce costs and schedules via increased reuse. CASE tool improvements reduce software specification, development, test, and documentation costs. Commercial network-based collaboration tools can reduce both costs and schedules.

The EDCS program has five technology clusters focused on two primary themes. One is improved support for incremental software evolution (contrary to popular perception, evolution technology begins paying off not only after delivery but after the first specifications are formulated). The second is technology development in DoD mission contexts, such as aircraft, missiles, satellite systems, and sensor systems. Its five technology clusters are described in the recent EDCS Demo Days 98 document [Salasin – La Monica, 1998] as follows:

Rationale Capture and Software Understanding: Rationale is captured in a knowledge base and represented in a manner that permits explanation, exploration, and management. Constraint maintenance techniques can be used to monitor rationale, such as design decisions, so that future changes to the system do not violate previous requirements.

Architecture and Generation: Architecture activities combine descriptions of component functionality with specifications of how a component interacts, coordinates, cooperates, and communicates with other components. The goal is to recognize the fundamental qualities imparted to systems by these various interconnection strategies, verify that implementations achieve these qualities, and encourage informed choices. Executable code can be generated or composed in the context of the specified architecture.

High Assurance and Real-Time: Design, development, deployment, integration, interoperation, and evolution of systems that provide evidence supporting high confidence that all critical system requirements will be met. Two approaches are being explored. The first approach combines testing and analysis of executable code. The second emphasizes providing run-time assurances that the system will conform to desired properties.

Design Management: Addresses a scaleable information substrate that is aimed at the retention, organization, and exploitation of all useful software information. It will: provide shared views of the information infrastructure; support information conceptualization, representation, and manipulation of multi-media software artifacts; enable effective collaboration in a distributed setting; provide consistency management of all software-pertinent artifacts; and, support processes for the controlled evolution of the substrate.

Dynamic Languages: The capabilities of dynamic languages to support incremental changes, rapid prototyping, and optimized software performance, are combined with techniques to increase reliability and efficiency, and facilitate interoperability with more conventional static languages. Dynamic languages also promote implementation decisions that are made at the latest possible stages of system development to avoid costly rework.

E.2 KBSA Evaluation Framework and Models

E.2.1 Overall Evaluation Framework

We have used the COCOMO II cost and schedule estimation model [Boehm et al., 1995] as the primary model used in performing the KBSA impact evaluation. The rationale for using COCOMO II involves four primary factors:

1. It was designed as a revision of the original Constructive Cost Model (COCOMO) [Boehm, 1981] to better address current and future software life cycle practices.

2. Its internals are in the public domain, and it is available either as a free basic tool or in several more powerful commercial versions.

3. It is well-calibrated to recent projects. The COCOMO II.1998 calibration produces estimates within 30% of project actuals 71% of the time for its sample of 161 projects, and 76% of the time when its coefficient is calibrated to individual organizations [Chulani et al., 1998].

4. Its project database provides a baseline for the KBSA evaluation. Over 100 of the projects in the database were completed in the 1990's. Their average cost driver ratings provide a current-practices baseline from which we can assess the impact on the cost drivers of KBSA and other technologies. We can then use the model to derive the resulting cost and schedule estimates associated with these resulting cost drivers.

We have found that the COCOMO II schedule estimation model is good for traditional software projects, but not for Applications Composition or other forms of Rapid Application Development (RAD). COCOMO II, along with COCOMO 81 and most other software cost-schedule estimation models, uses a variant of the classic cube-root model for schedule estimation:

$$M = 3 \sqrt[3]{PM}$$

where M is the development schedule in months and PM is the development effort in person-months.

Recently, many organizations, including DoD mission-critical system organizations, have found that it is more important to minimize software development or modification schedule (cycle time) than to minimize cost. They have also found a number of techniques, such as rapid prototyping, development process reengineering and critical path streamlining, collaboration

```
┌─────────────────────┐
1998 EHART project      │                     │──▶ Resulting 1998
baseline cost drivers ──▶│                     │    EHART cost,
                         │                     │    schedule estimates
                         │                     │
Projected 2006 and 2013  │   COCOMO II,        │
EHART cost drivers for:  │   CORADMO           │──▶ 2006, 2013 EHART-CD
                         │                     │    cost, schedule estimates
• Commercial tech. & ───▶│                     │
  DoD practices (CD)     │                     │──▶ 2006, 2013 EHART-
                      ───▶│                     │    KBSA cost schedule
                         │                     │
• KBSA technology     ───▶│                     │──▶ 2006, 2013 EHART-
  improvements over CD   │                     │    EDCS cost schedule
                         │                     │    estimates
                      ───▶│                     │──▶ 2006, 2013 EHART-
• EDCS technology        │                     │    KBSA/EDCS cost,
  improvements over CD   │                     │    schedule estimates
                         └─────────────────────┘
```

technology, architecture-based parallel development, and prepositioning of assets, which can make schedules faster than the cube root model.

In concert with its Affiliates, USC-CSE has developed a RAD Opportunity Tree providing a structural approach to such methods [Boehm-Chulani-Egyed, 1997]. We have recently completed a COCOMO II extension called CORADMO for estimating the schedule effects of various mixes of RAD strategies (see sections E.2.2 and K for further details).

Figure 2 provides an overview of how we use COCOMO II and CORADMO to perform the software technology impact assessments for the representative DoD embedded, high-assurance, real-time (EHART) missile software project.

Figure 2. Overview of Impact Assessment Approach

E.2.2 Evaluation Approach

Figure 3 provides more specifics on how the evaluation is conducted. It shows the ranges of the effort multipliers and scale factors in the 1998 calibration of the COCOMO II model. A diamond on each of the ranges shows the average multiplier for the 106 1990's projects in the COCOMO II database.
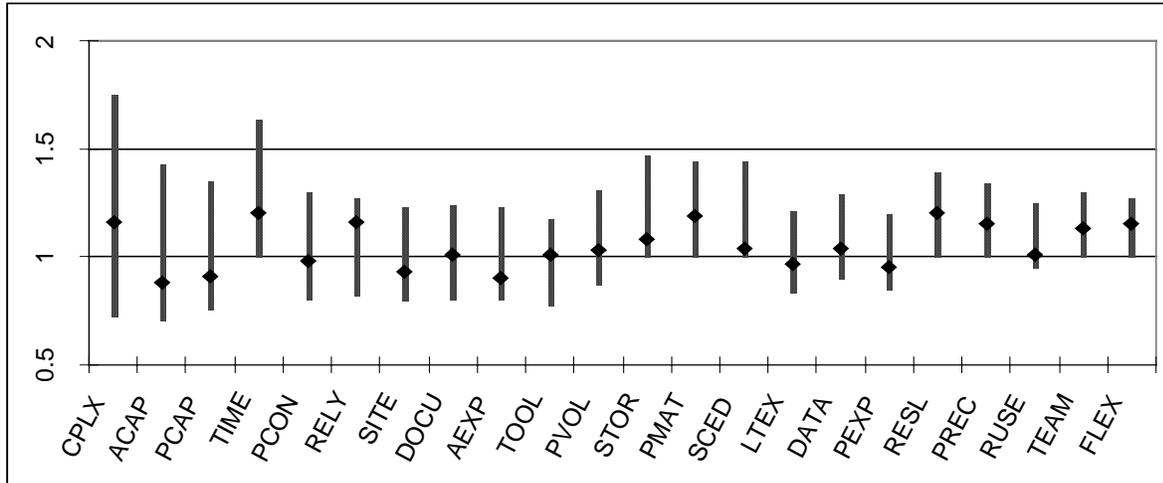
♦ **Average Multiplier for 1990's projects**

Figure 3. COCOMO II.1998 Productivity Ranges and Current Practice

For example, the range of multipliers for the TOOL (Use of Software Tools) cost driver runs from a multiplier of 1.17 for a Very Low project use of software tools (17% less efficient than a project using a Nominal level of tools), to a multiplier of 0.78 for a Very High project use of software tools (22% more efficient). Detailed rating scales for TOOL and the other variables are in Section J.

In the COCOMO family of models, software development or modification effort in person months (PM) is calculated as:

$$PM = A \cdot \Pi \text{ (cost driver multipliers)} \cdot \text{(Effective Size)}^{B + \text{Scale Factors}}$$

Cost is calculated from PM via an average cost/PM factor. Effective size is converted into equivalent thousands of lines of source code (KSLOC), including effects of reuse, breakage, function points and language level.

The average TOOL multiplier for the 106 1990's projects in the COCOMO II database is 1.01. For TOOL and each of the other COCOMO II cost drivers and scale factors, we have provided a worksheet in Section L providing our assessments of the effects of commercial/normal DoD, KBSA, and EDCS improvements on the project's TOOL rating and multiplier for the years 2006 and 2013.

For example, our assessment was that commercial technology would decrease the project's TOOL multiplier from 1.01 to 0.94 by the year 2013 (about 1/3 of the way to the lowest multiplier of 0.78). We assessed that KBSA technology would lower the multiplier farther, to 0.90; and that the combination of KBSA and EDCS technology would lower the project's TOOL multiplier to 0.86, about 2/3 of the way to the lowest multiplier. The rationale for these reductions was that KBSA and EDCS technology would be providing stronger tools for EHART applications than mainstream commercial technology would provide.

Once all the cost driver multipliers and scale factors (and the reductions in effective size due to reuse, very high level languages, and reduced breakage) were assessed, we could enter them into COCOMO II to generate estimates of the required effort and schedule for the various combinations of technologies and assessment dates. A similar approach was used to adjust the estimates for RAD improvement via CORADMO.

E.2.3 Overview of COSSEMO and CORADMO Models

E.2.3.1 Introduction

The **CO**COMO **RAD MO**DEL (CORADMO) is currently implemented in two parts: a front end staged schedule and effort model, **CO**COMO **S**tage **S**chedule and **E**ffort **MO**DEL (COSSEMO), and a back end RAD model

E.2.3.2 COCOMO Stage Schedule and Effort MODEL (COSSEMO)

The COSSEMO model is based on the lifecycle anchoring concepts discussed by [Boehm, 1996]. The anchor points are defined as Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC). Figure 4, adapted from Rational Corporation [Rational, 1998] shows the stages around the anchor points.
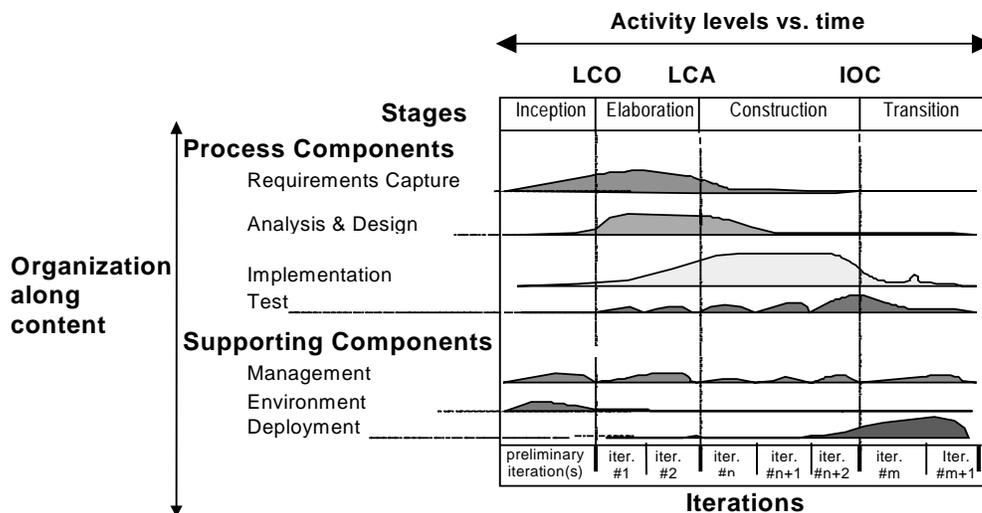
Figure 4. COCOMO Stages (Rational Phases) and Anchor Point Milestones

17

COCOMO II, COSSEMO & CORADMO use the word "stage" so it is not confused with the classic waterfall phases which correspond to the activities shown in Figure 4: Requirements Capture, Analysis & Design, Implementation, and Test.

COCOMO's effort and schedule estimates are focused on Elaboration and Construction (the Stages between LCO and IOC). Inception corresponds to the COCOMO's "Requirements" activity, which is actually an additional (fixed percentage) effort, above and beyond the effort calculated by COCOMO.

Another important difference of COSSEMO/CORADMO's schedule estimation from COCOMO II's simple schedule estimation is the use of a more complex calculation for the low effort situations. The initial COCOMO II baseline schedule equation roughly follows the classic "three times the cube root of the effort" model. For low-effort situations, for example PM = twenty seven (27) person months, this yields a very pessimistic and unlikely duration of M=nine (9) months at an average staff level of P= three (3) people. As a result, a new mathematical function is used to calculate (predict) the calendar months for a given amount of effort: the function is only radically different in low (under 25) person-month efforts where it seems more normal to have an equal number people and months to accomplish the task. At the higher (greater than 64) person-month efforts, the traditional COCOMOII-1998 function is used which is based on the traditional cube-root-like function of effort. A smooth curve is fit within these ranges. An example using the square root and classic 3-times-cube-root formulas is shown in Figure 5.
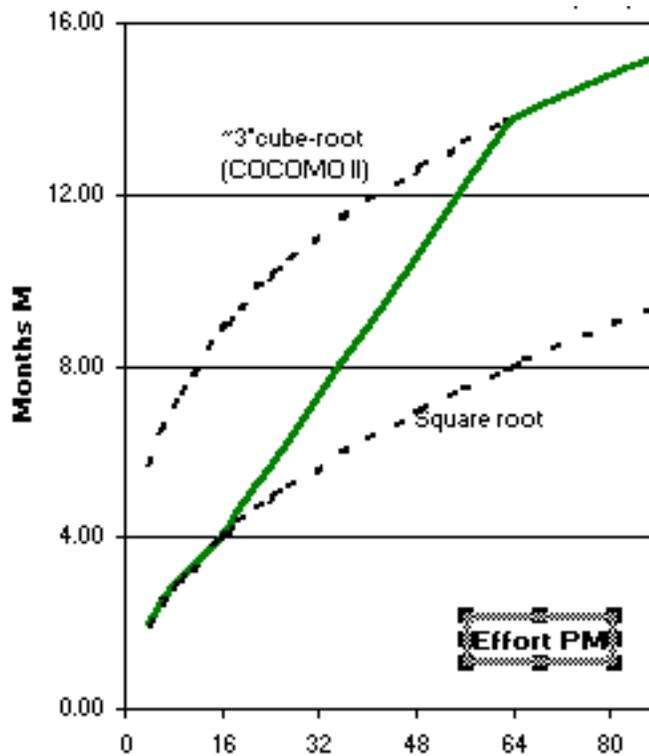
Figure 5. CORADMO Revision to COCOMO II Schedule Estimator

E.2.3.3. COCOMO RAD MODEL (CORADMO)

The COCOMO RAD model has its roots in the results of a 1997 CSE Focused Workshop on Rapid Application Development [Boehm - Chulani - Egyed, 1997]. RAD is taken to mean application of any of a number of techniques or strategies to reduce software development cycle time. A "RAD Opportunity Tree" presented at the workshop identified 5 classes of strategies that whose degree of implementation can be used to parameterize a schedule estimate given an effort estimate produced by COCOMOII-1998. These strategies replace the simple add-staff strategy embodied in the COCOMO II SCED cost driver. The five classes are development process re-engineering (DPRS), re-use and very high level languages (RVHL), collaboration efficiency (CLAB), architecture and risk resolution (RESL), and pre-positioning of assets (PPOS). RESL corresponds to the COCOMO II scale factor; the other four are new. All have their effects reflected as multipliers on effort (person months, PM), schedule (months, M) and/or average number of personnel (P). Person months of effort can actually be increased because certain pro-active strategies, like pre-positioning of assets, are only possible with extra effort.

CORADMO utilizes the COSSEMO extension which allocates effort and schedule to the stages which are anchored at the LCO/LCA/IOC points in a development life cycle. Staged schedule and effort distribution is needed because the effects of the RAD strategies identified above is different for the different stages.

The intent of CORADMO is to calculate/predict the schedule (months, M), personnel (P), and adjusted effort (person-months, PM) based on the distribution of effort and schedule to the various stages, and impacts of the selected RAD strategy schedule driver ratings on the M, P, and PM of each stage.

E.2.4 Evaluation Cases Considered

E.2.4.1 EHART

It is important to note that the embedded, high-assurance, real-time (EHART) applications domain is not the only domain to which this analysis applies. The EHART domain is just one example of a class of domains called "warfighting" domains in the NRC "Ada and Beyond" report [NRC, 1997] for which commercial technology solutions are unlikely to satisfy many of DoD's application needs.

Other examples of warfighting domains from the NRC report are electronic warfare, wideband surveillance, battle management, and battlefield communications. The assessment of commercial, KBSA, and EDCS technology impact presented in this report for the EHART domain would be applicable in large measure for the other warfighting domains as well. For DoD commercially-dominated domains such as finance and logistics the impact of commercial technology would be considerably higher, and the impact of KBSA and EDCS technology considerably lower.

The particular representative EHART application used in this analysis is a 100,000 source line of code missile software suite (100KSLOC; 1400 function points if using Ada and the 71 SLOC/FP ratio for Ada in [Jones, 1990]). It is starting its development in 1998. The application has a higher required reliability (RELY[1]), execution time constraint (TIME[2]), and main storage constraint (STOR[3]) than the COCOMO II 1990's database averages; these values have been adjusted upwards for the analysis.

---

1 Required Software Reliability (RELY): This is the measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only slight inconvenience then RELY is low. If a failure would risk human life then RELY is very high.

2 Execution Time Constraint (TIME): This is a measure of the execution time constraint imposed upon a software system,. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource. The rating ranges from nominal, less than 50% of the execution time resource used, to extra high, for 95% of the execution time resource is consumed.

3 Main Storage Constraint (STOR): This rating represents the degree of main storage constraint imposed on a software system or subsystem. The rating ranges from nominal, less that 50%,   to   extra   high,

The analysis assumes that the counterpart missile software projects starting in 2006 and 2013 have substantially the same functionality, but their accuracy and performance requirements are adjusted upwards to capitalize on improvements in computer, sensor, and guidance technology. The net effect of these on key COCOMO II cost drivers is the following:

- SIZE: the application and architecture remain similar enough to enable significant gains via domain-engineered reuse and applications generators.

- CPLX: the application complexity level remains essentially constant

- TIME, STOR: hardware advances reduce the resource constraints, but do not eliminate them due to the increased accuracy and performance requirements.

These assumptions are also applicable to applications in the other warfighting domains discussed above.

E.2.4.2 KBSA

The analysis separately considers the application of the two major KBSA technology branches – Applications Generation (KG) and Decision Support (KD) – and their combined KBSA application (K). The analysis assumes that programs have been put in place to develop, extend, and transition the indicated technologies, with a particular emphasis on the development and use of solutions capitalizing on domain knowledge for the selected DoD applications area (in this case, EHART missile applications, but comparably applicable to other DoD warfighting domains).

The KBSA Applications Generation (KG) program would focus on domain models, domain architectures, tailorable components, domain-oriented very high level languages for visual programming of applications, and general and domain-oriented applications generator technology enabling efficient generation of both originally-specified applications or modifications to such applications.

The KBSA Decision Support (KD) program would focus in general and domain-oriented technology providing effective guidance on such software engineering decisions as process model selection; domain architecture selection; architecture view integration; achievement of system properties such as performance, reliability, safety, survivability, security, interoperability, adaptability, usability, etc.; risk assessment and risk management planning; COTS integration; cost and schedule management; verification and test planning and management. The intent would be to bring decision support for such decisions to the level of maturity and applicability as it currently is for such decisions as change control or process control via control limits.

E.2.4.3 EDCS

The analysis assumes that a strong EDCS Phase II effort would be put in place, with a particular emphasis on developing and transitioning domain-oriented solutions in the selected warfighting domains. It also assumes that a strong EDCS followon technology development effort would be put in place to expand the capabilities being developed in such areas as requirements

determination, architecture, software collaboration, incremental specification/generation/verification, and achievement of high-assurance software.

E.2.4.4 Commercial Technology

The analysis assumes that commercial software technology will continue to focus on mass-market domains and tool support such as broad-use requirements, design, programming, test, project management, and configuration management aids. As indicated in the [NRC, 1997] study, commercial software tool vendors will focus primarily on technology investments with near-term payoff. Thus, commercial tools for niche markets such as DoD warfighting applications will develop slowly.

E.2.4.5 Technology Combinations

The analysis begins with the 1998 baseline. It uses the 100 KSLOC EHART missile application and the average 1990's cost driver ratings from the COCOMO II database. These ratings might be considered optimistic, as they come from organizations sufficiently advanced to have collected thorough data on their software projects. However, this is offset by the fact that the average date of the projects' completion is 1994. On balance, the ratings should be well-representative of 1998 EHART project practice, with the EHART adjustments noted for the RELY, TIME, and STOR cost driver ratings.

The commercial technology and DoD practice (CD) ratings for 2006 and 2013 are assessed next. The KBSA Applications Generation (KE) and Decision Support (KD) ratings for 2006 and 2013 are assessed relative to the CD rating levels, followed by a similar assessment for the combined KBSA technology impact ratings. Finally, a similar set of ratings for 2006 and 2013 EDCS technology impact is assessed, followed by ratings for the combination of KBSA and EDCS technology.

The assessments are made with the aid of a spreadsheet implementation of the technology impact analysis model. This enabled us to easily perform sensitivity analyses with respect to our technology ratings. We provide one set of sensitivity analyses indicating the results of our assessments if KBSA and EDCS technology were optimistic by a factor of 2 with respect to commercial technology. Even with this conservative assessment, the results indicate a strong DoD payoff for the technology investments.

E.2.5 Example of Model Driver Analysis

We presented a simple summary of our model driver analysis approach for the COCOMO II TOOL cost driver in Section E.2.2. Figure 6 below presents an example of the actual model driver analyses performed for this study, using the COCOMO II Architecture and Risk Resolution (RESL) scale factor. The full set of model driver analysis worksheets are provided in Section L.

RESL: Architecture/Risk Resolution

Driver		Baseline	CD@+8	CD@+15	KG@+8	KG@+15	KD@+8	KD@+15	K@+8	K@+15	E@+8	E@+15	EK@+8	EK@+15
RESL	default	3.97	3.50	3.00	3.20	2.50	3.20	2.50	3.00	2.20	3.00	2.20	2.70	1.70
RESL	new													

RESL Current/New

(chart: legend — CD, KG, KD, K, E, EK, ▲ SF; SF values: ▲ 0 EH, ▲ 1.41 VH, ▲ 2.83 H, ▲ 4.24 N, ▲ 5.65 L, ▲ 7.07 VL)

CD:	Significant gains due to open systems technology, commercial OO architecture technology and DoD emphasis on risk management
KG:	Significant additional gains over CD via domain architectures
KD:	Significant additional gains over CD via architecture and risk advisor technology
K:	Complementary gains from KG and KD
E:	Major additional gains over CD via domain architecture, general architecture technology, high assurance technology, and rationale capture
EK:	Complementary gains from E and K

Figure 6. RESL: Architecture/ Risk Resolution Analysis

RESL and other COCOMO II scale factors add increments to the exponent B in the COCOMO II effort equation:

$$PM = A \cdot \prod (\text{Effort Multipliers}) \cdot (\text{Size})^B \text{, where}$$

$$B = B_o + 0.01 \cdot \Sigma (\text{Scale Factors}).$$

The figures by the triangles show the calibrated scale factors for the RESL ratings, ranging from 0 for very thorough architecture and risk resolution to 7.07 for very lax architecture and risk resolution. The effect of a scale driver is higher on larger-size products: for a 100 KSLOC product, it is a factor of $100^{1.0707}/100 = 1.38$; for a 1000 KSLOC product, it is a factor of $1000^{1.0707}/1000 = 1.63$. The behavioral explanation for the difference is that projects with lax architecture and risk resolution spend much more effort in rework due to late risk and defect removal and communications overhead across poorly-defined interfaces. These effects are larger for larger products.

The 1998 baseline average from the COCOMO II project database is a RESL scale factor of 3.97. The effect of commercial technology and DoD practices (CD) on this scale factor was assessed as significant: a reduction to 3.5 by 2006 and to 3.0 by 2013. This is due partly to ongoing commercial and DoD open systems initiatives, such as CORBA and the Defense Information Infrastructure Common Operating Environment (DII-COE); improved commercial object-oriented architecture support largely based on the Unified Modeling Language; and DoD

emphasis on risk management in DoDD 5000.1, DoDI 5000.2, and the DoD Software Best Practices Initiative.

However, the effect is not larger because these advances do not fully address a number of EHART domain issues (e.g., distributed real-time deadline management). Thus, there are a number of added advances that can be achieved via KBSA and EDCS technology improvements, particularly if pursued within an EHART domain context.

The KBSA Applications Generator (KG) technology when pursued in an EHART domain context would provide improved domain architectures and avoidance of various classes of architecture and real-time scheduling risks. The KBSA Decision Support (KD) technology would provide complementary improvements via decision aids for such issues as architecture choices, risk identification and resolution, and choices of scheduling algorithms, process models, and verification strategies. Each of these is assessed as providing significant additional gains down to a scale factor of 3.20 in 8 years (2006) and down to 2.50 in 15 years (2013). The sources of gains are complementary, so that the effect of a 2-prong KBSA program (K) is assessed to reduce the scale factor to 3.0 by 2006 and 2.2 by 2013.

The EDCS program (E) would provide some of the same improvements, particularly via its EHART domain coverage. It would not cover some of the KBSA Decision Support technology, but would provide stronger capabilities than the KBSA program in such areas as general architecture description and analysis, high-assurance test and verification, risk reduction via rationale capture, and evolutionary/incremental specification, development, and verification techniques. The net improvement in the RESL scale factor over the CD values would be about the same as those for a full KBSA program, to 3.0 by 2006 and 2.2 by 2013.

Since the KBSA and EDCS programs provide some complementary benefits, there is an additional scale factor reduction assessed for a combined KBSA-EDCS program (EK) down to 2.70 in 2006 and 1.70 in 2013. The overall improvement contribution in reducing the RESL scale factor from 3.97 to 1.70 is about 12% for a 100 KSLOC product.

E.2.6 Spreadsheet Model Overview

A multi-sheet Excel Workbook has been developed to show the impacts of the COCOMOII and CORADMO drivers projected over time and technology type on a selected domain's typically sized application. This spread sheet model has the name "Software Technology Impact Analyzer". The sheets include a description of the other sheets and the COCOMOII-1998 Calibration values and ranges for reference, and sheets for the COCOMOII and CORADMO drivers and their impacts.

The workbook also has several protected sheets which are used for the detailed layout of the drivers to facilitate the graphing shown in the "Drivers" sections. There are also protected sheets for the default values (i.e. the USC Center for Software Engineering assessed values) of the COCOMO-II.1998 and CORADMO drivers.

E.2.6.1 COCOMO-II Drivers, Calculations and Impacts

There are three sheets in this grouping. The first, "CII SF&EM Drivers", has the projected drivers over time. The second, "CII SF&EM Data", aggregates the driver data and does the COCOMOII calculations. The third, "CII E&S Impact", has graphs showing the effort and schedule impact of the COCOMO-II.1998 drivers projected over time.

"CII SF&EM Drivers" shows our assessed values for each of the drivers projected over time and our rationale, and allowing the input of new values and additional or modified rationales. A graph of the current values of the driver projected over time is included; the data points on this graph change when new values are entered.

"CII SF&EM Data" has the assessed COCOMO-II.1998 drivers, both scale factors and effort multipliers, organized in a compact, single page sheet along with the calculations of the COCOMOII effort and schedule. The calculations use the COCOMO-II.1998 model equations for effort and the COSSEMO equations for schedule (different schedule formulas for three ranges of months: 0 to 16; 16 to 64; and 64 and up). Each data column of the table performs the full set of COCOMO-II calculations for a particular year and one technology-type combination.

E.2.6.1.1 Example Spreadsheet Calculations

An example of the comparative COCOMO II effort (PM) calculations is shown in Table 1. The columns correspond to the 1998 baseline and pairs of 2006 and 2013 projections of the effort effects of commercial/general DoD technology (CD); the KBSA Applications Generator (KE), Decision Support (KD), and Combined (K) effects; the EDCS (E) effects; and the combined EDCS and KBSA (EK) effects.

The six rows starting with row 7 show the assessed values of the five COCOMO II scale factors and their sum Σ. For example, the values for Architecture and Risk Resolution (RESL) are exactly those explained in Figure 6. Row 13 shows the impact of the various technologies in reducing the COCOMO II diseconomy-of-scale factor B from its 1998 baseline value of 1.076 to a 2013 EK value of 0.991. Recall that the COCOMO II effort estimation equations in Section E.2.5 are:

$$PM = A \cdot \prod (\text{Effort Multipliers}) \cdot (\text{Size})^{B} \text{ , where}$$

$$B = B_{o} + 0.01 \cdot \Sigma (\text{Scale Factors})$$

For COCOMO II.1998, the calibrated values of A and B_o are 2.94 and 0.91, respectively.

Rows 14-30 of Table 1 show the corresponding assessed effects of the various technologies on the COCOMO II effort multipliers. For example, the reduction in the TOOL effort multiplier from a 1998 baseline value of 1.01 to a 2013 value of 0.86 for the combination of KBSA and EDCS (EK) technology is exactly as discussed in Section E.2.2. The resulting effect of the ensemble of effort multipliers is to reduce their product Π (row 31) from a 1998 baseline of 1.21 to a 2013 EK value of 0.38, a factor of more than 3.

Cost-Driver	Baseline (now) 1998	CD 8Yr 2006	CD 15Yr 2013	KG 8Yr 2006	KG 15Yr 2013	KD 8Yr 2006	KD 15Yr 2013	K 8Yr 2006	K 15Yr 2013	EDC 8Yr 2006	EDC 15Yr 2013	EK 8Yr 2006	EK 15Yr 2013
PREC	3.1	2.8	2.5	2.5	2.0	2.8	2.5	2.5	2.0	2.5	2.0	2.5	2.0
FLEX	3.2	2.8	2.5	2.5	2.0	2.6	2.3	2.4	1.8	2.4	1.8	2.2	1.5
RESL	4.0	3.5	3.0	3.2	2.5	3.2	2.5	3.0	2.2	3.0	2.2	2.7	1.7
TEAM	2.7	2.4	2.0	2.4	2.0	2.0	1.4	2.0	1.4	2.1	1.6	1.8	1.1
PMAT	3.7	3.0	2.2	3.0	2.2	2.8	1.8	2.8	1.8	3.0	2.2	2.8	1.8
Σ	16.6	14.5	12.2	13.6	10.7	13.4	10.5	12.7	9.2	13.0	9.8	12.0	8.1
B	1.076	1.055	1.032	1.046	1.017	1.044	1.015	1.037	1.002	1.040	1.008	1.030	0.991
RELY	1.16	1.14	1.12	1.14	1.12	1.14	1.12	1.14	1.12	1.10	1.06	1.10	1.06
DATA	1.04	1.01	0.98	1.01	0.98	1.01	0.98	1.01	0.98	1.01	0.98	1.01	0.98
CPLX	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16
RUSE	1.01	1.05	1.03	1.10	1.08	1.05	1.03	1.10	1.08	1.10	1.08	1.10	1.08
DOCU	1.01	0.97	0.95	0.93	0.91	0.95	0.93	0.92	0.89	0.92	0.89	0.92	0.89
TIME	1.20	1.12	1.08	1.12	1.08	1.12	1.08	1.12	1.08	1.08	1.04	1.08	1.04
STOR	1.08	1.04	1.02	1.04	1.02	1.04	1.02	1.04	1.02	1.04	1.02	1.04	1.02
PVOL	1.03	1.01	0.98	1.01	0.98	1.01	0.98	1.01	0.98	0.98	0.94	0.98	0.94
ACAP	0.88	0.88	0.88	0.88	0.88	0.86	0.82	0.86	0.82	0.88	0.88	0.86	0.82
PCAP	0.91	0.91	0.91	0.91	0.91	0.89	0.85	0.89	0.85	0.91	0.91	0.89	0.85
PCON	0.98	0.98	0.98	0.98	0.98	0.96	0.93	0.96	0.93	0.96	0.93	0.95	0.91
AEXP	0.90	0.89	0.87	0.87	0.84	0.89	0.87	0.87	0.84	0.87	0.84	0.87	0.84
PEXP	0.95	0.94	0.92	0.94	0.92	0.94	0.92	0.94	0.92	0.94	0.92	0.94	0.92
LTEX	0.97	0.95	0.92	0.93	0.89	0.95	0.92	0.93	0.89	0.93	0.89	0.93	0.89
TOOL	1.01	0.98	0.94	0.98	0.94	0.96	0.90	0.96	0.90	0.94	0.88	0.94	0.86
SITE	0.93	0.91	0.88	0.91	0.88	0.91	0.88	0.91	0.88	0.88	0.84	0.88	0.84
SCED	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04
Π	1.21	0.93	0.67	0.89	0.63	0.83	0.52	0.81	0.49	0.72	0.45	0.68	0.38
SCED%	95.71	95.71	95.71	95.71	95.71	95.71	95.71	95.71	95.71	95.71	95.71	95.71	95.71
SIZE	100	60	30	40	15	60	30	40	15	35	12	30	10
CII_PM	505.48	204.55	65.76	124.00	23.97	175.65	43.05	108.83	21.51	85.66	16.37	66.71	10.90

Table 1. Comparative COCOMO II Effort Calculations

The other major influence on software effort in the COCOMO II equations is SIZE, roughly equivalent to the number of source instructions one needs to program. Technology can

reduce this via reuse, reduced breakage, or very high level languages, particularly in concert with domain specialization and knowledge. The SIZE worksheet in Section L provides the rationale for the size reduction effects of the various technologies; the resulting reductions are shown in row 35 of Table 1.

Given the values of B, Π, and SIZE, the COCOMO II equations produce the corresponding effects on effort (CII_PM) of the various technologies (row 36). The effects will be displayed graphically and discussed in Section E.3 below. The full set of spreadsheet calculations for cost/effort and schedule impacts are provided on Section L.

E.2.6.2 CORADMO Drivers, Calculations and Impacts

Like the COCOMOII-1998 sheets, there are three sheets in this grouping. The first, " RAD Drivers ", has the projected drivers over time. The second, " CORADMO Data ", aggregates the driver data and does the CORADMO calculations. The third, "RAD SM Impact", has graphs showing the resulting impacts of the COCOMO-II.1998 and CORADMO drivers projected over time.

"RAD Drivers" has our assessed values for each of the CORADMO schedule multiplier drivers projected over time and our rationale, and allowing the input of new values and additional or modified rationales. A graph of the current values of the driver projected over time is included; the data points on this graph change when new values are entered.

"CORADMO Data" aggregates the CORADMO drivers, both scale factors and effort multipliers, organized in a compact, single page sheet along with the calculations of the CORADMO effort and schedule. The sheet also allows for the distribution of effort and schedule to the various stages as specified in the COSSEMO model. The calculations use the CORADMO model equations to distribute schedule and effort based on the selected percentage allocations and the schedule multiplier driver ratings. Each PM & M pair of data columns of the table performs the full set of CORADMO calculations for a particular year and one technology-type combination.

"RAD SM Impact" has graphs showing the effort, schedule and Full-time Software Personnel (FSP) impacts of the entered CORADMO drivers projected over time. All three are shown for each stage (Inception, Elaboration, and Construction) since the schedule multipliers can impact both effort and schedule; and the FSP values are then simply the result of dividing effort (in person months) of a stage by its duration (in months).

E.2.6.3 Technical Impact Final Results

At the end of the "RAD SM Impact" worksheet, following the RAD impacts by stage, are the summary charts for effort and schedule by technology over time that result from the COCOMO II and CORADMO driver changes over time. The data for these charts is actually shown on the second page of the "CORADMO Data" sheet. The effort and schedule results are generated by adding the effort or schedule, respectively, for all three stages. Since COCOMO II only calculates the effort and schedule for development, a second set of summary charts was generated so the COCOMO-II model results could be easily compared to the CORADMO model

results. The second set of charts totals effort and schedule only for the Elaboration and Construction stages. Along with each chart are copies of the rows of the appropriate data from "CORADMO Data" sheet.

E.3 KBSA Comparative Impact Analysis Results

E.3.1 Cost/Effort Impact

Figure 7 shows the relative assessed effect of the various technologies in reducing the amount of effort required to develop the 100 KSLOC EHART missile application. The 1998 baseline effort is 505 person months (PM). The assessed effect of commercial technology and DoD practices (CD) is to reduce this effort by a factor of 2.5 (to 204 PM) by 2006 and another factor of 3 (to 64 PM) by 2013.
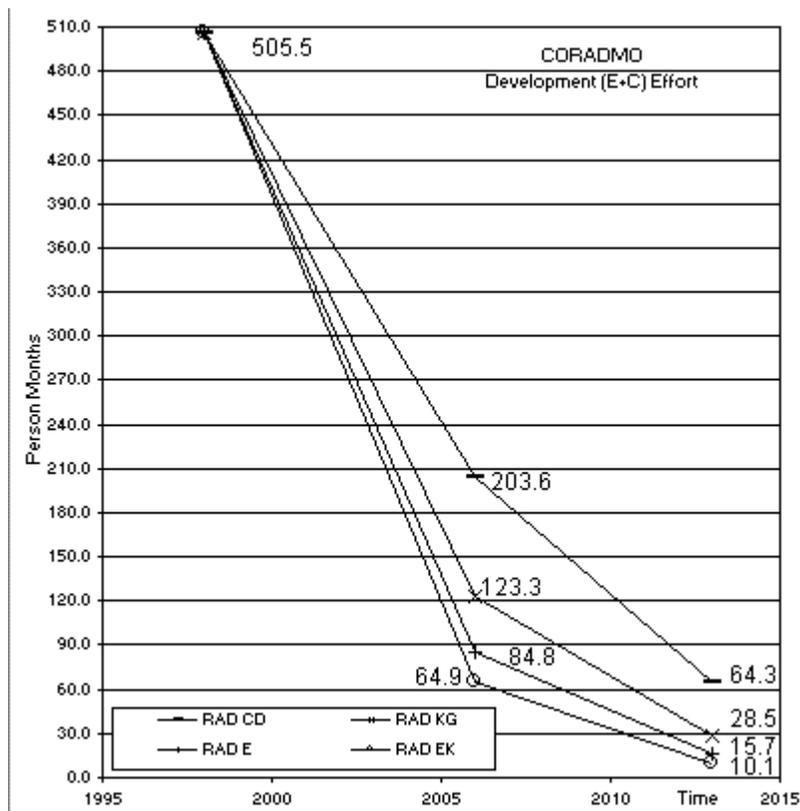
Figure 7. Impact of Technologies on Software Effort or Cost

These are impressive gains, but it must be remembered that most of them are also within reach of any rogue nation or terrorist organization purchasing commercial technology. To keep a competitive military edge, USAF and DoD need to have superior software management and technology capabilities.

The remainder of the curves indicate that significant improvements are available via investments in KBSA and EDCS technology. A combination of EDCS and KBSA technology investments is assessed to provide another factor-of-3 gain over commercial/DoD technology and practices (to 65 PM) by 2006 and another factor-of-6 gain (to 10 PM) by 2013.

An analysis of the major factors contributing to the additional technology gains indicates that the primary source is the reductions in effective size due to combinations of software technology and domain knowledge engineering. These account for almost all of the gains from KBSA Applications Generator (KG) technology, and much of the gains from EDCS (E) technology.

However, as seen by the relative position of the KG effort values, domain engineering is not a single silver-bullet solution for achieving DoD warfighting software productivity gains (nor is commercial technology). Other particularly strong contributors were:

- High assurance technologies, strongly improving the cost driver factors for RESL (Architecture and Risk Resolution), RELY (Required Reliability), TIME (Execution Time Constraints), and TOOL (Use of Software Tools);

- Requirements, collaboration, and rationale capture technologies, strongly improving the cost of driver factors for RESL (Architecture and Risk Resolution), TEAM (Team Cohesion), PCON (Personnel Continuity), and SITE (Multisite Development);

- KBSA decision support technologies, strongly improving the cost driver factors for RESL (Architecture and Risk Resolution), TEAM (Team Cohesion), ACAP and PCAP (Analyst and Programmer Capability), PCON (Personnel Continuity), and TOOL (Use of Software Tools), particularly in the longer term (2013).

- Software architecture and incremental evolution support technologies, strongly improving the cost driver factors for RESL (Architecture and Risk Resolution), TOOL (Use of Software Tools), and MCF (Maintenance Change Fraction, factored into SIZE).

Some software technology areas receiving recent DoD support which did not generate significant cost and schedule impacts over commercial technology impacts were Integrated Software Environments and Formal Methods. Integrated Software

Environments are being supported at such a critical-mass level by commercial technology that they become too difficult to match via independent DoD efforts. Formal Methods are very important for ultra-high assurance DoD applications as computer security and information warfare; they also provide support for small-scale applications generation capabilities. But they do not apply well or scale up well to large COTS-intensive applications.

E.3.2 Schedule Impact

Figure 8 shows the counterpart results of the CORADMO-based analysis of the impact of the various technologies on the development schedule for the representative EHART project. The detailed spreadsheet analysis for this and related effects are in Section L.
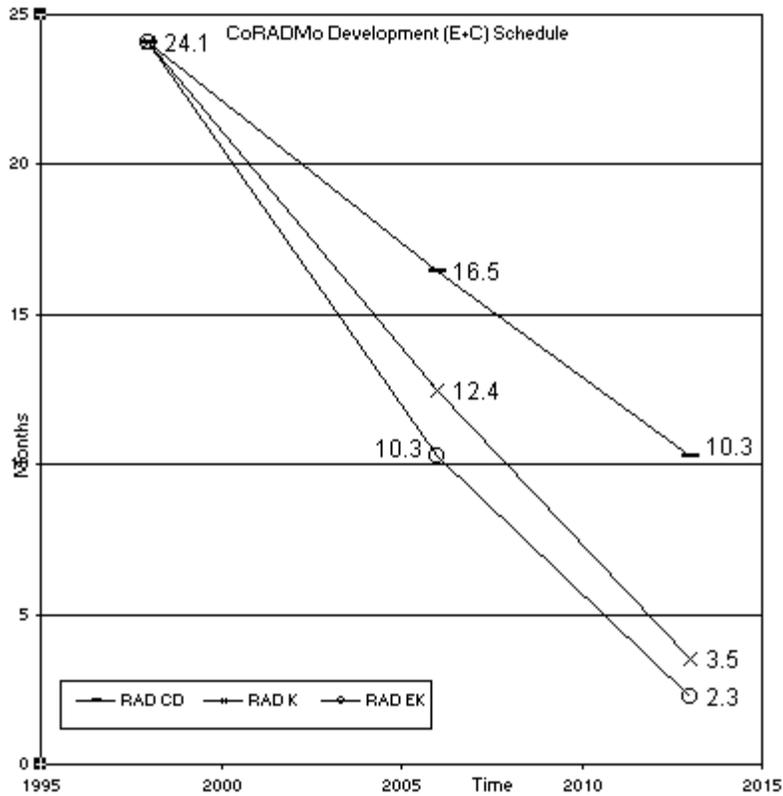
Figure 8. Impact of Technologies on Software Schedule

The general trends of the results are similar to those in the cost/effort analysis shown in Figure 7. The main highlights are:

- Commercial technology and DoD practices (CD) provide significant gains, reducing the 1998 baseline EHART schedule from 24.1 months to 16.5 months by 2006 and to 10.3 months by 2013. Besides the contributions via reduced effort, the primary added RAD contributions came from added collaboration efficiencies via commercial collaboration technology gains and DoD Integrated Product Team practice gains.

- KBSA technology has the potential to reduce the development schedule even lower, to 12.4 months by 2006 and to 3.5 months by 2013. Besides reducing effort, the added RAD gains come from rapid prototyping via domain knowledge-based application generators and from KB decision aids for accelerated processes and collaboration.

- EDCS technology can reduce the schedule even further, via domain application generators plus other architecture-based prepositioning of assets and via collaboration technology.

- A unified EDCS/KBSA/commercial technology/DoD practices approach is assessed to reduce the schedule even further, to 10.3 months by 2006 and to 2.3 months by 2013. The main reason for the significant additional shortening by 2013 is that the

project size has been reduced to the point that schedule and staffing can follow the square-root relation in Figure 5 rather than the 3-times-cube-root relation.

The CORADMO analyses also address effects of the RAD strategies on development effort, by the net results for the EHART project were not much different from the COCOMO II estimates. The detailed results are in Section L.

E.3.3 <u>Sensitivity Analyses</u>

The technology-impact results in Figure 7 and Figure 8, and in the detailed analyses in Section L, are based on the KBSA ADM evaluation and the authors' experience-based judgement. It is always possible that these judgements are overly optimistic or pessimistic. To reduce these possibilities, we have performed or provided several cross-checks:

- Boehm and Brown independently assessed the effects of the technologies on the model parameters, resulting in considerable iteration of the parameter values.

- We reviewed the values and results with two industry software metrics experts and did a preliminary briefing of the results for AFRL/IFTD, resulting in further suggestions and iteration of the results.

- We have done a comparative analysis with the results of similar studies (in Section E.3.4 below), and have found the results fairly consistent.

- We have developed the spreadsheet model so that anyone with different technology assessments can enter them in the model and see the results. The model and its usage instructions are in Section N, and also available via our Web site at http://sunset.usc.edu/COPROMO/KBSA_LCE/kbsa_lce.html.

- We have used the spreadsheet model to perform various analyses of the sensitivity of the results to the model's inputs. The most significant example is presented next; others are included in Section L.

Figure 9 shows the result of a sensitivity analysis in which we assumed that all of our assessments of the influence of KBSA and EDCS technology were optimistic by a factor of 2. For example, in the last column of Table 1 (Section E.2.6.1.1), the effect of the combined 2013 EDCS and KBSA technology was to reduce the scale factor sum Σ from its 2013 CD value of 12.2 down to 8.1. Similarly, the assessment reduced the effort multiplier product Π from 0.67 to 0.38, and the effective size from 30 to 10 KSLOC.

For the analysis in Figure 9, these differences, and all the other differences between the 2013 CD assessed values and the assessed values in the columns to the right of it were reduced by a factor of 2. For the 2013 EK values above, the Σ value became 10.2, the Π value became 0.52, and the SIZE value became 20 KSLOC.
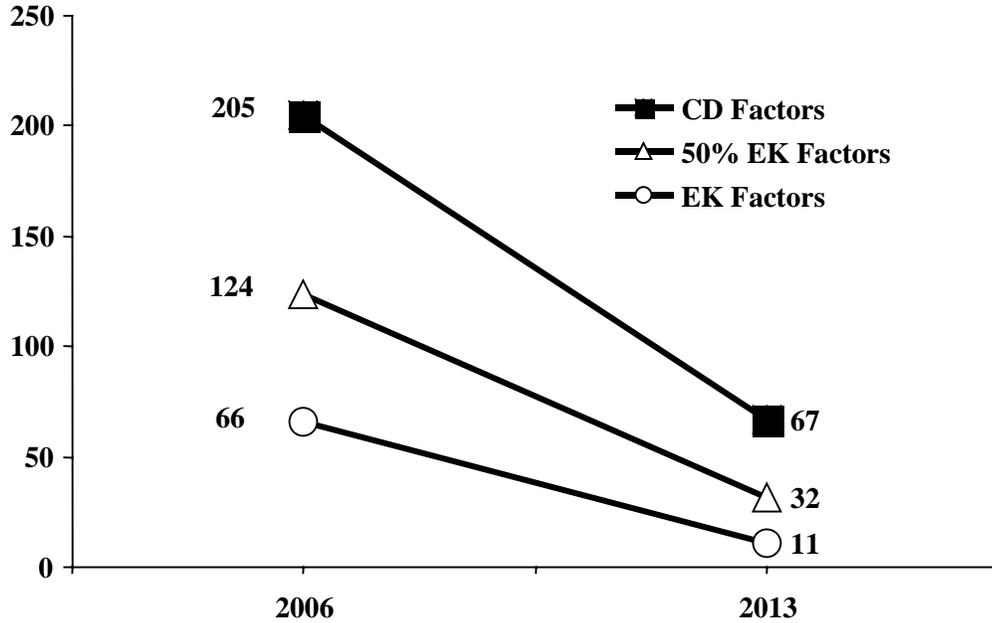
Figure 9. Effect of 50% Reduction in Effort Improvement Factors

The results in Figure 9 show that reducing the EK improvement parameters by a factor of 2 leads to reduction in effort savings by a factor of 1.7-1.75 (e.g., for 2013, (67-11)/(67-32) = 1.75). For the other technology comparisons, the results were similar: a savings reduction close to but somewhat less than a factor of 2. Even with the more conservative payoff assessments, the resulting benefits appear well worth the technology investments.

E.3.4 Comparison to Other Studies

A good comparative study of commercial technology impact on software effort or productivity is provided in [Bernstein, 1997]. Bernstein's Figure 10 shows the trend in "expansion factor," the ratio of machine language instructions per line of generated source code, between 1960 and 1995. The trend line in Figure 10 indicates a factor-of-10 improvement every 20 years, a figure very consistent with this study's factor-of-7.7 improvement from commercial technology over 15 years. Bernstein's 1995 and projected 2000 expansion factors rise above this trend line, and appear to be due to a strong degree to domain-oriented reuse gains. This is again consistent with this study's determination of additional productivity benefits from domain-oriented technology in DoD warfighting domains not well covered by commercial technology.
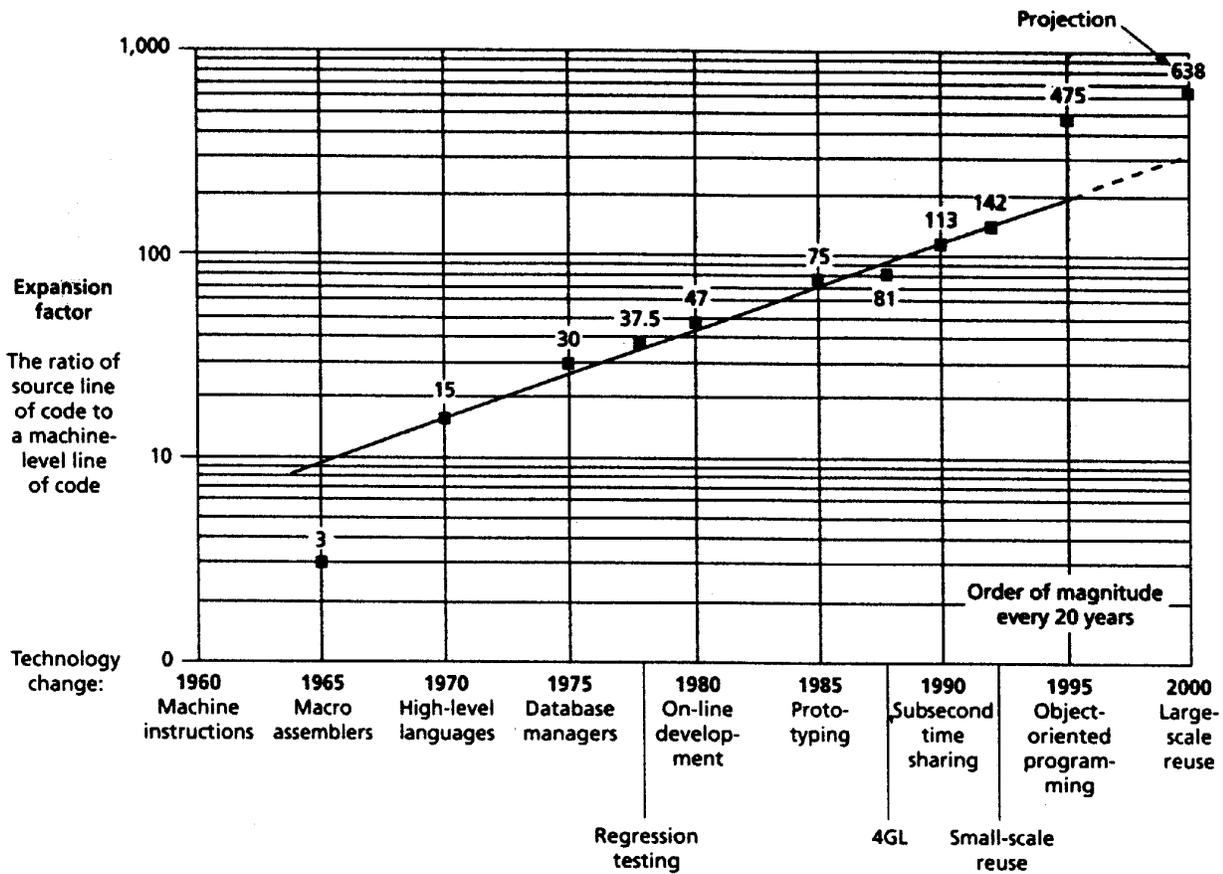
Figure 10. Software Expansion Factor vs. Time [Bernstein, 1997]

A similar study to this was done for the DoD STARS program [Boehm – Standish, 1983]. It predicted a factor-of-4.34 effort reduction over 10 years, but did not distinguish between STARS-based and commercial technology-based gains.   The DoD Software Technology Strategy [DoD, 1991] used a different value-chain approach to estimate effort reductions for each type of software activity (requirements, design, code, test, documentation, management, etc.).   It indicated a potential factor-of-12 savings in software effort over 15 years, but a more likely factor-of-4 savings in 15 years due to technology transition inertia.   It did not distinguish between savings due to DoD technology investments vs. those due to commercial technology.

Both of these studies indicated that effective size savings via reuse was the largest source of savings, although only the latter emphasized domain-specific reuse. All three of the major recent books on software reuse, [Poulin, 1996; Jacobson et al., 1997; Reifer, 1997], indicate that reuse is the most attractive source of software savings, and that domain-specific reuse is far more effective than domain-independent reuse.

E.4 Conclusions and Recommendations

E.4.1 Conclusions

1. The KBSA Advanced Development Model was not sufficiently robust and scalable to provide appreciable benefits to critical DoD software projects. We found its framework difficult to scale to large projects, and its toolset to be small with respect to large project needs.

2.   However, the KBSA concepts, if otherwise realized, have strong potential for reducing costs and schedules for DoD-critical warfighting software applications. The key to realizing this potential is to develop the KBSA technology in the context of DoD warfighting domain knowledge.   For KBSA Applications Generation, this implies focusing on DoD warfighting areas where a good start exists towards domain engineering, domain architecting, and development and use of reusable domain assets. Some of the USAF/ESC Product Line areas are good candidates, as are some avionics, missile, and sensor processing domains.

For KBSA Decision Support, this involves both domain-specific decision aids and more generic decision aids for security, survivability, risk assessment, architecture definition, and process definition and management.

3. The KBSA Decision Support area provides an attractive new direction for Air Force and DoD software technology investments. Knowledge-based and agent-based mixed-initiative technical approaches are generating high payoffs in decision support areas similar to software engineering such as manufacturing and logistics. A similar initiative for software engineering would build on existing technology in attractive new directions.

4.   EDCS technology investments and payoffs have both commonalities and complementarities with respect to KBSA technology investments and payoffs. To some extent, good portions of a KBSA Applications Generation program have been pursued under EDCS. A KBSA Decision Support program could build upon and provide new directions for some current EDCS efforts in such areas as rationale capture, design webs, and architecture technology.

5.   The Software Technology Impact Analysis spreadsheet model provides a useful tool for assessing alternative software technology assessment strategies.   It enables technology managers to relate candidate technologies' effects on software cost and schedule drivers to resulting project cost and schedule improvements.   It is well calibrated and baselined with respect to current software practice and commercial technology and productivity trends.

6. *There is no software technology silver bullet for DoD warfighting systems, including reliance on commercial technology*. A mixed strategy of commercial technology and complementary DoD warfighting software technology is necessary to address DoD's full range of needs.

E.4.2 Recommendations

1. *The Air Force and DoD should continue their investments in software technology, particularly for warfighting domains.* Both a strong DoD integrating emphasis and strong service domain-focused applications emphasis are needed. A sustaining EDCS Phase I-level technology generation program with continuing EDCS Phase II-level technology insertion efforts appears to be an appropriate magnitude for such a program.

Given their commonalties, a combined EDCS-KBSA program could be pursued at roughly the initially-planned EDCS Phase I and II levels of investment. If complementary DoD management and commercial technology assimilation strategies were in place, the program could generate software cost and schedule benefits similar to those estimated in the analysis in Section E.3. As noted in [NAE, 1995] and [Fox et al., 1995] the resulting return-on-investment ratio for DoD would be very high.

From Section E.3.1, the strongest software technology investment impacts are likely to come from DoD warfighting domain-based reuse and applications generation; high assurance technologies; requirements, collaboration, and rationale capture technologies; and software architecture and incremental evolution support technologies. Areas with less strong general investment impacts were integrated software environments and formal methods, although formal methods are important for some small or ultra-high assurance DoD applications areas.

2. *The software technology investments should be part of an integrated DoD software management/process/technology strategy to ensure DoD software supremacy, particularly in warfighting domains.* Besides the technology development and transition investments, elements of the integrated strategy are OSD management initiatives toward integrated Capability Maturity Models and best practices; acquisition initiatives such as the USAF/ESC Product Line strategy; domain engineering initiatives such as the SEI Product Line Systems program; and enterprise architectures building on the DII-COE.

The conclusions and recommendations are not particularly surprising. They are quantitative counterparts of the qualitative conclusions and recommendations about the need for DoD investments in software technology in recent Air Force Scientific Advisory Board studies [SAB, 1995; SAB, 1997] and recent DoD-level studies [NAE, 1995; Fox et al., 1995; NRC, 1997]. What is surprising is that in response to this consistent set of signals, DoD has been reducing its level of investments in software technology.

F. Discussion of Tools: Technology Impact Analyzer

F.1 Tool Overview

The Technology Impact Analyzer is a multi-sheet Excel Workbook that shows the impacts of the COCOMO II and CORADMO drivers projected over time and technology-type on a selected domain's typically sized application. The sheets include an overview and sheets for the COCOMO-II.1998 , COSSEMO and CORADMO drivers, data and their impacts.

The overview sheet includes abbreviations and descriptions of the other sheets on the first page, Figure 11. More information on the overview and the other sheets is available Appendix 5.

Techonology Impact Analyzer Abbreviations										
CD=	Commercial technology and DoD general practice					CII=	CoCoMo II-1998			
KBSA=	Knowledge Based Software Assistant					SF=	Scale Factor		EM=	Effort Multiplier
KG=	KBSA Applications Generators, including KB domain engineering (+CD)						RAD=	CoRADMo (schedule & effort)		
KD=	KBSA Project Decision Support (SE decision assistant concept) (+CD)						SM=	Schedule Multiplier		
K=	Both KG & KD						PM=	Person Months		
E=	EDCS or Evolutionary Delivery of Complex Software Sysetms						M=	Months		
EK=	both EDCS & KBSA (KG & KD)						FSP=	Fulltime Software Personnel		
EHART=	Embedded, High Assurance, Real Time [baseline application domain]					SSE=	Staged Schedule and Effort			
						I=Incepstion	E=Elaboration	C=Construction		
The Techonology Impact Analyzer Workbook has several worksheets covering										
Technoloy Impact Analyzer Overview Sheet ("TIA" tab): This sheet with										
	1.	Abbreviations and worksheet overviews								
	2.	COCOMOII-1998 Calibration values and ranges								
COCOMOII-1998 Scale Factor & Effort Multiplier Drivers										
	COCOMOII-1998 Scale Factor & Effort Multiplier Drivers projected over time ("CII Drivers" tab)									
		Individual parameters displayed with default and new/current numeric values, and graph of current values.								
	COCOMOII-1998 Scale Factor & Effort Multiplier Data ("CII Data" tab)									
	1.	Parameters organized in a compact, single page for review, along with schedule & effort calculation.								
	2.	Calculates effort according to the COCOMOII-98 rules and schedule according COSSEMO rules								
		(different schedule formulas for three ranges of months--0 to 16; 16 to 64; and 64 and up).								
COCOMOII-1998 Effort and Schedule Impact ("CII Impact" tab)										
		Displays the Effort & Schedule impacts that result from the driver values' change over time.								
COSSEMO: Stage (Inception, Elaboration and Construction) percentage distribution of Schedule and Effort ("SSE %" tab)										
	1.	Input of inception, elaboration and constrution stages' schedule and effort percentages								
	2.	Chart of distribution of schedule and effort impacts on the current COCOMO II calculations								
CORADMO: Schedule and Effort Multipliers Projected Over Time for KBSA Evaluator										
	CoRADMo Drivers projected over time ("RAD Drivers" tab)									
		Individual parameters displayed with default and new/current numeric values, and graph of current values.								
	CoRADMo Drivers projected over time ("RAD Data" tab)									
	1.	Parameters Organized in compact single page for review								
	2.	Calculates effort, schedule & FSP according CORADMO rules after distribution of effort & schedule per COSSEMO rules.								
	CoRADMo Schedule and Effort Multipliers Impact ("RAD Impact" tab)									
Final Results:===>	Displays the Effort, Schedule and FSP impacts that result from the driver values' change over time									

`|◄|►|►|\ TIA / CII Drivers / CII Data / CII Impact / SSE % / RAD Drivers / RAD Data / RAD Impact / |◄|`

Figure 11. TIA Abbreviations and Sheet Descriptions

F.1.1 COCOMO-II Drivers, Calculations and Impacts

There are three sheets in this grouping. The first, "CII Drivers", has the current projected scale factors and effort multipliers drivers over time and allows for changing the default values to their new values. The second, "CII Data", aggregates the driver data and does the COCOMO II calculations. The third, "CII Impact", has graphs showing the effort and schedule impact of the COCOMO-II.1998 drivers projected over time. Only "CII Drivers" allows input. More information on "CII Drivers" is available the next section. More information on all these sheets is available in Appendix 5.

F.1.2 CoSSEMo Schedule and Effort Percentage Distributions per Stage

This sheet, "SSE %", allows the input of percentage distributions of effort and schedule to the various stages, Inception, Elaboration, and Construction, as required for the COCOMO II Staged Schedule and Effort Model (COSSEMO). The impact of these distributions on the COCOMO-II.1998 baseline results is shown in the chart at the end of the worksheet.

F.1.3 CORADMO Drivers, Calculations and Impacts

Like the COCOMO-II.1998 sheets, there are three sheets in this grouping. The first, "RAD Drivers", shows the new or default projected drivers over time. The second, "RAD Data", aggregates the driver data and does the CORADMO calculations. The third, "RAD Impact", has graphs showing the resulting impacts of the CORADMO drivers projected over time when applied to the corresponding COCOMO-II.1998 results with the COCOMO drivers projected over time. At the end of the page of the "RAD Data" sheet are the summary calculations for totals of schedule and effort across stages allowing comparison with the results of COCOMO-II.1998. Only "RAD Drivers" allows input. More information on "RAD Drivers" is available the next section. More information on all these sheets is available in Appendix 5.

F.1.4 Technical Impact Final Results

At the end of the "RAD Impact" worksheet, following the nine RAD impacts by stage charts, are the summary charts for effort and schedule by technology over time that result from the COCOMO-II.1998 and CORADMO driver changes over time. The effort and schedule results are generated by adding the effort or schedule, respectively, for either all three stages or just for the Elaboration and Construction stages. More information on this sheet is available in Appendix 5.

F.2 COCOMO II Drivers Display, Modification and Rationale

The worksheet with the "CII Drivers" shows all of our assessed values for each of the scale factor or effort multiplier drivers, projected over time and technology, and our rationale. Each page of this worksheet has the current projected COCOMO-II.1998 drivers, both scale factors and effort multipliers, over time and allows for changing the default values to their new values. The rationales for the default settings of the drivers are included; they should be modified when "new" values are provided. Figure 12 shows the scale factor PREC's information.

PREC: Precedentedness

Driver	Baseline	CD@+8	CD@+15	KG@+8	KG@+15	KD@+8	KD@+15	K@+8	K@+15	E@+8	E@+15	EK@+8	EK@+15
PREC default	3.06	2.80	2.50	2.50	2.00	2.80	2.50	2.50	2.00	2.50	2.00	2.50	2.00
PREC new													

CD:	Some gains due to better general understanding of EHART domains, but not large
KG:	Solid gains due to stronger domain understanding and technology, but offset by continuing need for more advanced systems
KD:	No additional domain gains over CD
K:	Same as KG
E:	No additional domain gains over KG
EK:	Same as E and K

Figure 12. PREC's Driver Entry, Modification and Display

The default and current values of the driver, projected over time and technology, are shown in a small table above the chart of the current values. The last row of this table accepts the input of new values of the driver, projected over time and technology. The chart below the table shows the driver's current values over time for each technology combination. The data points on this graph change when new values are entered.

Since each value of a driver should have a rationale, the rationales for the default values (our assessed values) are shown below the chart. The area below the rationales for the default values allows the input of additional or modified rationales.

F.3 COSSEMO Distribution of Schedule and Effort per Stage

There are two parts to this worksheet: 1) Input of inception, elaboration and construction stages' schedule and effort percentages; and 2) Chart of distribution of schedule and effort impacts on the current COCOMO II calculations.

Input of schedule and effort percentage distributions per stage, Inception, Elaboration, and Construction, is required for the COCOMO II Staged Schedule and Effort Model (COSSEMO). To help visualize these distributions, their impact on the COCOMO-II.1998 100K EHART baseline is displayed in the chart at the end of the worksheet. Figure 13 shows the entire content of this worksheet.
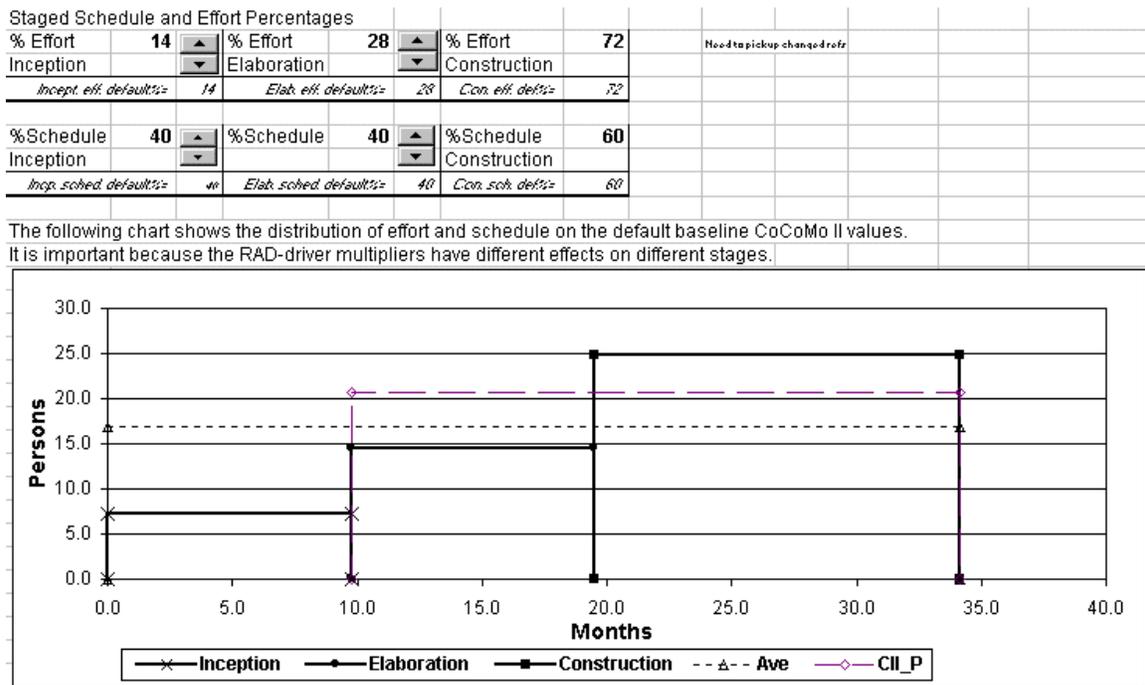
Figure 13. Staged Schedule and Effort Distribution

The values of the Inception and Elaboration percentages for schedule and effort are adjusted by clicking on the up/down arrows (spinners) shown to the right of their values. The current values are displayed in bold, along with the corresponding calculated values for the Construction stage. The default values for all the percentages are shown in italics.

F.4 CORADMO Drivers Display, Modification and Rationale

"RAD Drivers" has our assessed values for each of the relevant CORADMO schedule and effort multipliers projected over time and our rationale. It also allows the input of new values and additional or modified rationales. A graph of the current values of each driver projected over time and technology is included; the data points on this graph change when new values are entered.

Schedule		RVHL: Reuse and Very High Level Language											Inception	
Inception	Baseline	CD@÷8	D@÷15	KG@÷8	G@÷15	KD@÷8	D@÷15	K@÷8	K÷@15	E@÷8	E@÷15	EK@÷8	K@÷15	
RVHL-M defau	1.00	0.99	0.98	0.98	0.96	0.98	0.96	0.97	0.94	0.97	0.94	0.96	0.92	
RVHL-M new														

RVHL Projection Rationales

Baseline:	Relatively low current capability and experience in EHART domain (standard 3GL module reuse)
CD:	As indicated under SIZE in the Effort impact analysis, commercial technology and DoD EHART domain initiatives will provide some but not much improvement over standard 3GL module reuse
KD:	Some gains over CD via domain oriented reuse asset identification and decision support
KG:	Some gains over CD via domain oriented prototype applications generation
K:	Complementary gains from KD and KG
E:	Significant gains over CD via domain architecture technology and associated prototype applications generation
EK:	Some complement any gains from E and K
NOTE:	RVHL effects in construction accounted for with regular COCOMOII effort adjustment

Figure 14. RVHL's Inception Stage Schedule Multiplier Driver Information

The default and current values of the driver, projected over time and technology, are shown in a small table above the chart of the current values. The last row of this table accepts the input of new values of the driver, projected over time and technology. The chart below the table shows the driver's values over time for each technology combination.

Since each value of a driver should have a rationale, the rationales for the default values (our assessed values) are shown below the chart. The area below the rationales for the default values allows the input of additional or modified rationales.

F.5 Final Results: Technology Impacts

At the end of the "RAD Impact" worksheet, following the nine RAD impacts by stage charts, are the summary charts for effort and schedule by technology over time that result from the COCOMO-II.1998 and CORADMO driver changes over time. The "new/current" data for the summary charts is actually shown at the end of the "RAD Data" sheet.

There are three different types of charts:

1. Overall (effort or schedule for all three stages or just for development (elaboration plus construction), with some of these having alternative axes layouts;

2. COCOMO-II.1998 compared to CORADMO (final) results, with some of these charts showing only the major technology groupings (CD, K and EK);

3. Final results of default driver settings compared to new/current driver settings' results.

The list of all the charts corresponding to final results is shown in the Figure 14, below. More detailed information and examples of these charts are given in an Appendix.

Number	Title
1.	CORADMO Total Effort (effort on x axis)
2.	CORADMO Total Effort (years on x axis)
3.	CORADMO Total Effort (only for CD, K and EK)
4.	CORADMO Development (E+C) Effort with COCOMO II Development (E+C) Effort
5.	CORADMO Development (E+C) Effort with COCOMO II Development (E+C) Effort (only for CD, K and EK)
6.	CORADMO Total Schedule (schedule on x axis)
7.	CORADMO Development (E+C) Schedule with COCOMO II Development (E+C) Schedule(only for CD, K and EK)
8.	CORADMO Development (E+C) Schedule with COCOMO II Development (E+C) Schedule
9.	New/Current CORADMO Total (I+E+C) Effort with Default CORADMO Total (I+E+C) Effort
10.	New/Current CORADMO Total (I+E+C) Schedule with Default CORADMO Total (I+E+C) Schedule

Table 2. Final Result Charts

F.6 Sensitivity Analysis Example

One of the reasons for allowing input of new values for the drivers is to permit sensitivity analysis. Suppose it is believed that the impact of reuse and very high level languages has been over estimated, and that only 50% of the originally estimated impact seems to be justified.

F.6.1 COCOMO-II.1998 Driver Modification

First, the impacted COCOMO-II.1998 drivers need to be adjusted. Since there is no explicit driver for reuse, but rather the effective size is modified to reflect the decreased amount of reuse and/or use of a very high level language. This can be accomplished by filling each cell in the "new" row except the baseline, which remains the same, by a formula that subtracts fifty percent of the difference between the baseline and the default. The new values are shown in Figure 15.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
657				SIZE: KSLOC (repeated for new values and justifcation)															
658	Driver			Baseline	CD@+8	CD@+15	KG@+8	KG@+15	KD@+8	KD@+15	K@+8	K@+15	E@+8	E@+15	EK@+8	EK@+15			
659	SIZE		default	100.0	60.0	30.0	40.0	15.0	60.0	30.0	40.0	15.0	35.0	12.0	30.0	10.0			
660	SIZE		new	100.0	80.0	65.0	70.0	57.5	80.0	65.0	70.0	57.5	67.5	56.0	65.0	55.0			

Figure 15. Fifty Percent SIZE Impact

The "new" values, except baseline, in the table above the chart, like that in E632, were calculated with a formula like: "=$D659-($D659-E659)*(50/100)". The new values are reflected in the chart below. Also, the rationale is noted in the section following the chart.

F.6.2 CORADMO Driver Modification

Since RVHL reflects the impacts of reuse and/or very high level languages, it is the only driver that needs to be adjusted. Figure 16 shows the new values for the RVHL schedule multiplier for the inception stage.

Schedule		RVHL: Reuse and Very High Level Language											Inception	
Inception		Baseline	CD@-8	D@-15	KG@-8	G@-15	KD@-8	D@-15	K@-8	K+@15	E@-8	E@-15	EK@-8	K@-15
RVHL-M	defau	1.00	0.99	0.98	0.98	0.96	0.98	0.96	0.97	0.94	0.97	0.94	0.96	0.92
RVHL-M	new	1.00	1.00	0.99	0.99	0.98	0.99	0.98	0.99	0.97	0.99	0.97	0.98	0.96

RVHL Schedule Multipliers--Inception

— CD —□— KG —◇— KD —✳— K —┼— E —○— EK ▲ SM

RVHL Projection Rationales

Baseline:	Relatively low current capability and experience in EHART domain (standard 3GL module reuse)
CD:	As indicated under SIZE in the Effort impact analysis, commercial technology and DoD EHART domain initiatives will provide some but not much improvement over standard 3GL module reuse
KD:	Some gains over CD via domain oriented reuse asset identification and decision support
KG:	Some gains over CD via domain oriented prototype applications generation
K:	Complementary gains from KD and KG
E:	Significant gains over CD via domain architecture technology and associated prototype applications generation
EK:	Some complement any gains from E and K
NOTE:	RVHL effects in construction accounted for with regular COCOMOII effort adjustment
ALL:	Only 50% impact as part of sensitivity analysis

Figure 16. RVHL Inception Stage Adjustment for 50% Impact
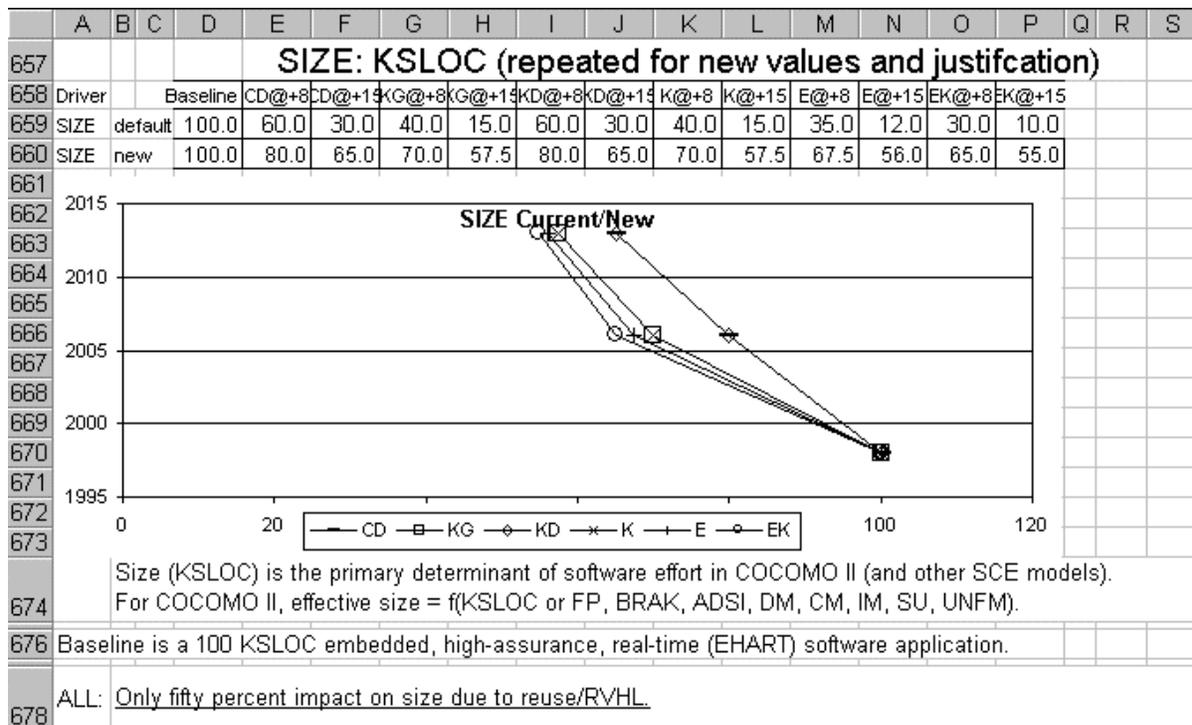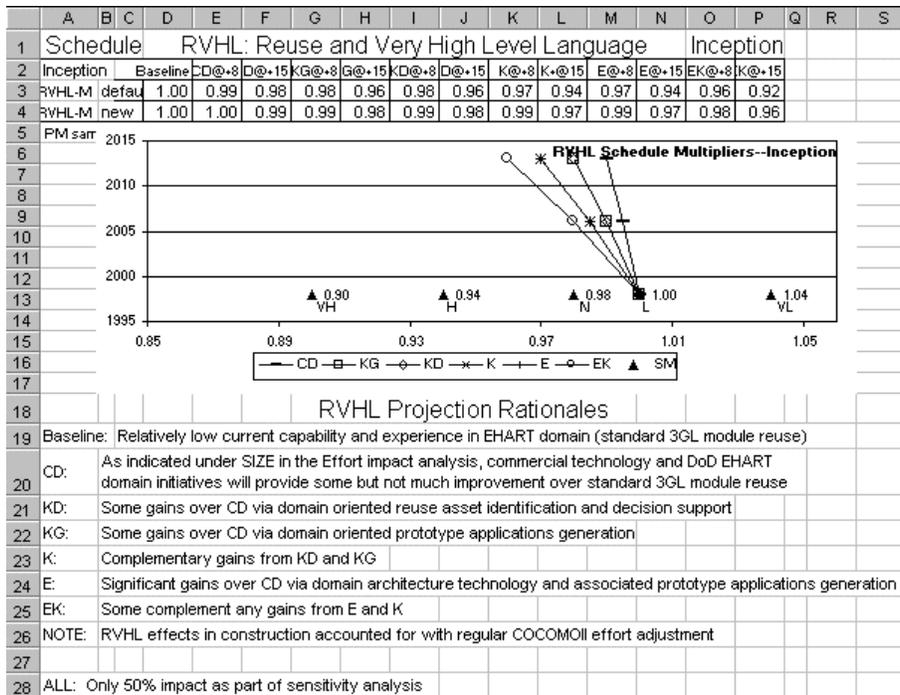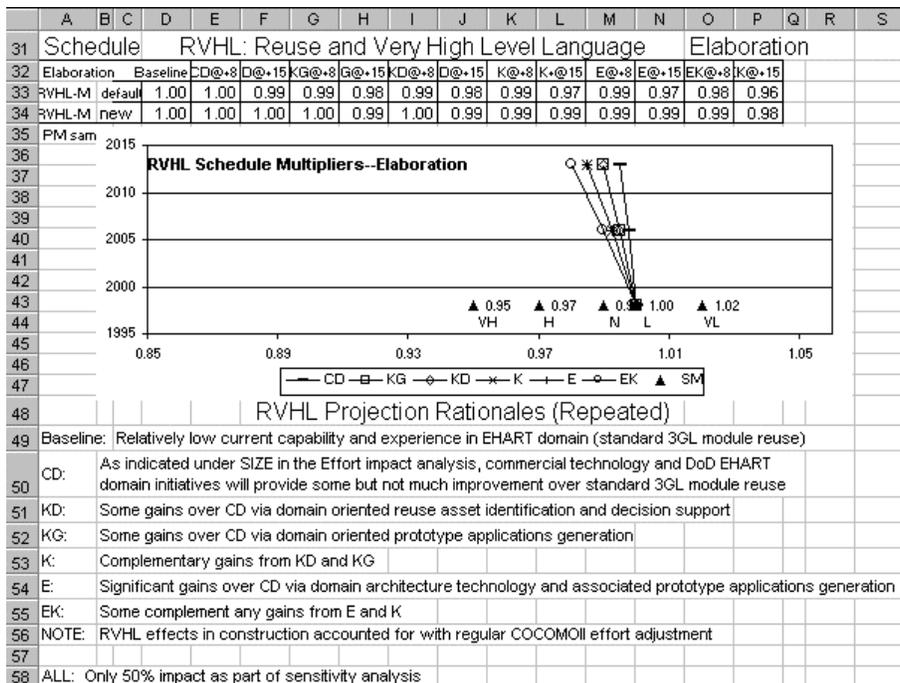
The "new" values, except baseline, in the table above the chart, like that in E4, were calculated with a formula like: "=$D4-($D4-E4)*(50/100)". The new values are reflected in the chart below. Also, the rationale is noted in the section following the chart.

Similarly, Figure 17 shows the new values for the RVHL schedule multiplier for the elaboration stage as well as the rationale for the modification.

Schedule		RVHL: Reuse and Very High Level Language											Elaboration	
Elaboration		Baseline	CD@-8	D@-15	KG@-8	G@-15	KD@-8	D@-15	K@-8	K+@15	E@-8	E@-15	EK@-8	K@-15
RVHL-M	default	1.00	1.00	0.99	0.99	0.98	0.99	0.98	0.99	0.97	0.99	0.97	0.98	0.96
RVHL-M	new	1.00	1.00	1.00	1.00	0.99	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.98

RVHL Schedule Multipliers--Elaboration

— CD —□— KG —◇— KD —✳— K —┼— E —○— EK ▲ SM

RVHL Projection Rationales (Repeated)

Baseline:	Relatively low current capability and experience in EHART domain (standard 3GL module reuse)
CD:	As indicated under SIZE in the Effort impact analysis, commercial technology and DoD EHART domain initiatives will provide some but not much improvement over standard 3GL module reuse
KD:	Some gains over CD via domain oriented reuse asset identification and decision support
KG:	Some gains over CD via domain oriented prototype applications generation
K:	Complementary gains from KD and KG
E:	Significant gains over CD via domain architecture technology and associated prototype applications generation
EK:	Some complement any gains from E and K
NOTE:	RVHL effects in construction accounted for with regular COCOMOII effort adjustment
ALL:	Only 50% impact as part of sensitivity analysis

Figure 17. RVHL Elaboration Stage Adjustment for 50% Impact

F.6.3 Sensitivity Analysis Results: Technology Impact Estimates

The effort and schedule results after applying the COCOMO-II.1998 and CORADMO driver modifications identified above are shown in Figure 18. This result can be compared to the default driver settings results shown in Figure 20. CORADMO Total Effort in Figure 18 refers to the same values as CORADMO Total (I+E+C) Effort in Figure 20.

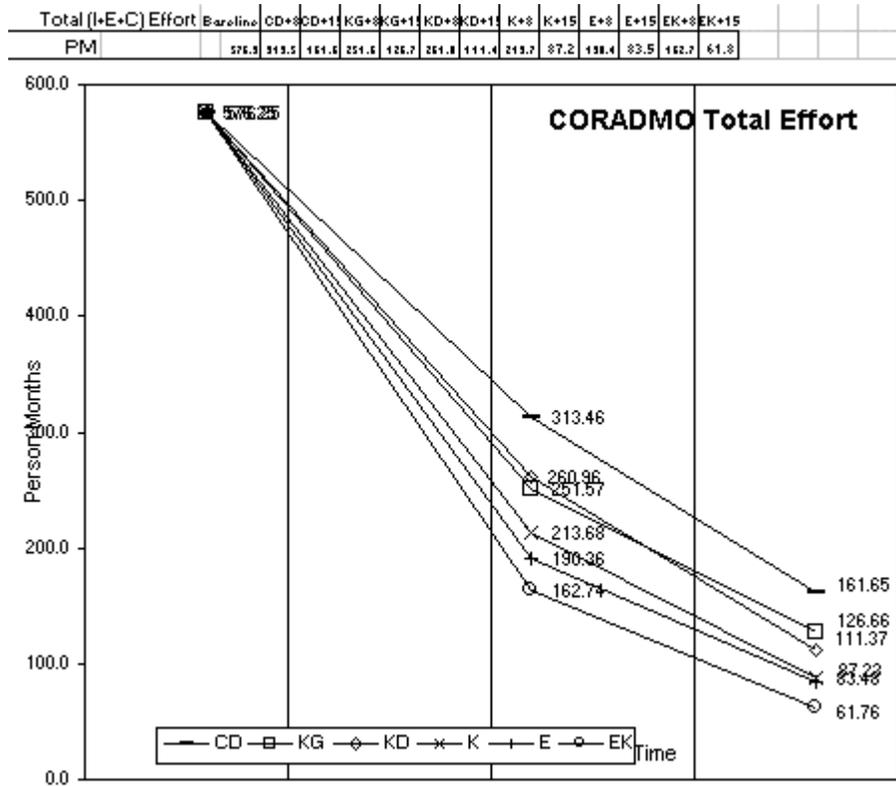| Total (I+E+C) Effort	Baseline	CD+8	CD+15	KG+8	KG+15	KD+8	KD+15	K+8	K+15	E+8	E+15	EK+8	EK+15			
PM		576.3	313.5	161.6	261.6	126.7	261.8	111.4	213.7	87.2	138.4	83.5	162.7	61.8		

Figure 18. Total Effort after applying modified COCOMO-II.1998 & CORADMO Drivers

Figure 19 a comparisons of COCOMO-II.1998 only results and the final CORADMO with only 50% of the reuse and high level language impact.

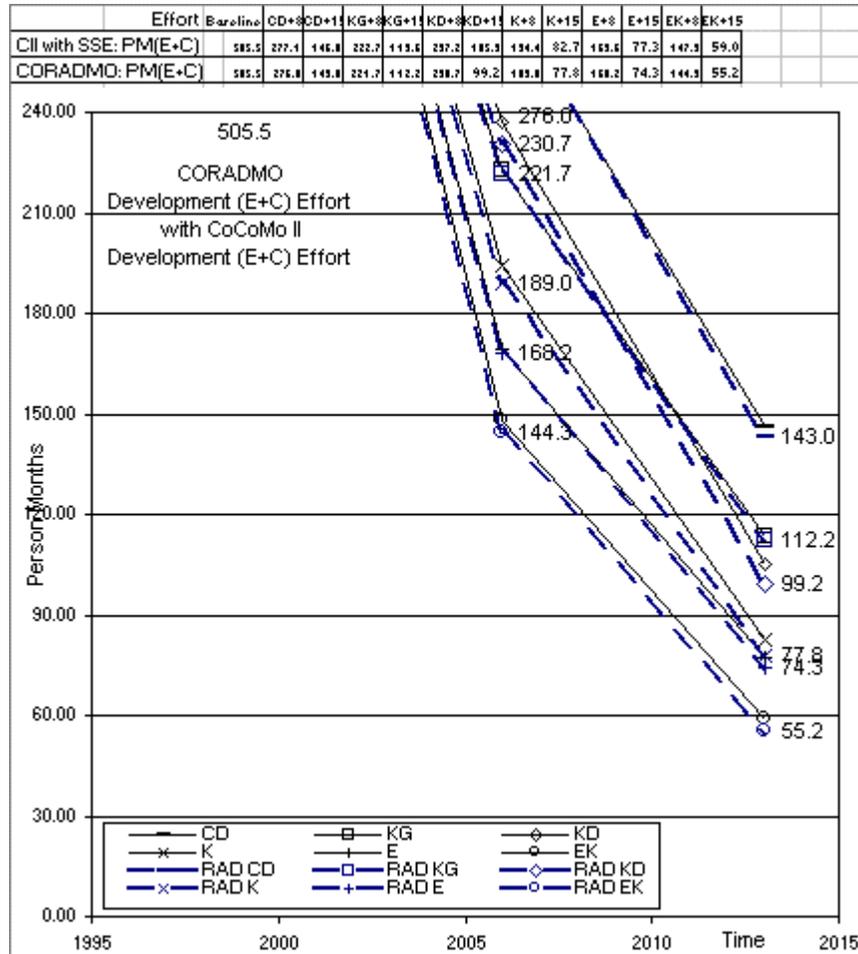| Effort	Baseline	CD+%	CD+15	KG+%	KG+15	KD+%	KD+15	K+%	K+15	E+%	E+15	EK+%	EK+15		
CII with SSE: PM(E+C)	505.5	277.1	146.1	222.7	113.6	297.2	185.3	154.4	82.7	163.6	77.3	147.3	59.0		
CORADMO: PM(E+C)	505.5	276.8	145.1	221.7	112.2	298.7	99.2	153.8	77.8	168.2	74.3	144.3	55.2		

Figure 19. One of the comparisons of COCOMO-II.1998 only results and Final Results

Here, both the COCOMO-II.1998 set of calculations and the final results calculations are shown in the table above the chart. Again, the final results row's values will contain the results based on the "current" CORADMO driver values, and thus may have changes anytime there is input in the "new" row of the drivers. While only the data associated with the top row of the table, which contains the COCOMO-II.1998 calculation results, is shown in the chart, the final results values are evident due to the dashed lines appearing in the chart.

For sensitivity analyses, the set of comparison charts especially usefull. They show the overall effort and schedule results using the default driver values and the new/current driver values. Along with each chart are copies of the rows of the appropriate data from "CORADMO Data" sheet. Figure 20 shows a comparison of final CORADMO results for default and new drivers (with the only driver change being SIZE (change amount reduced by 50%).
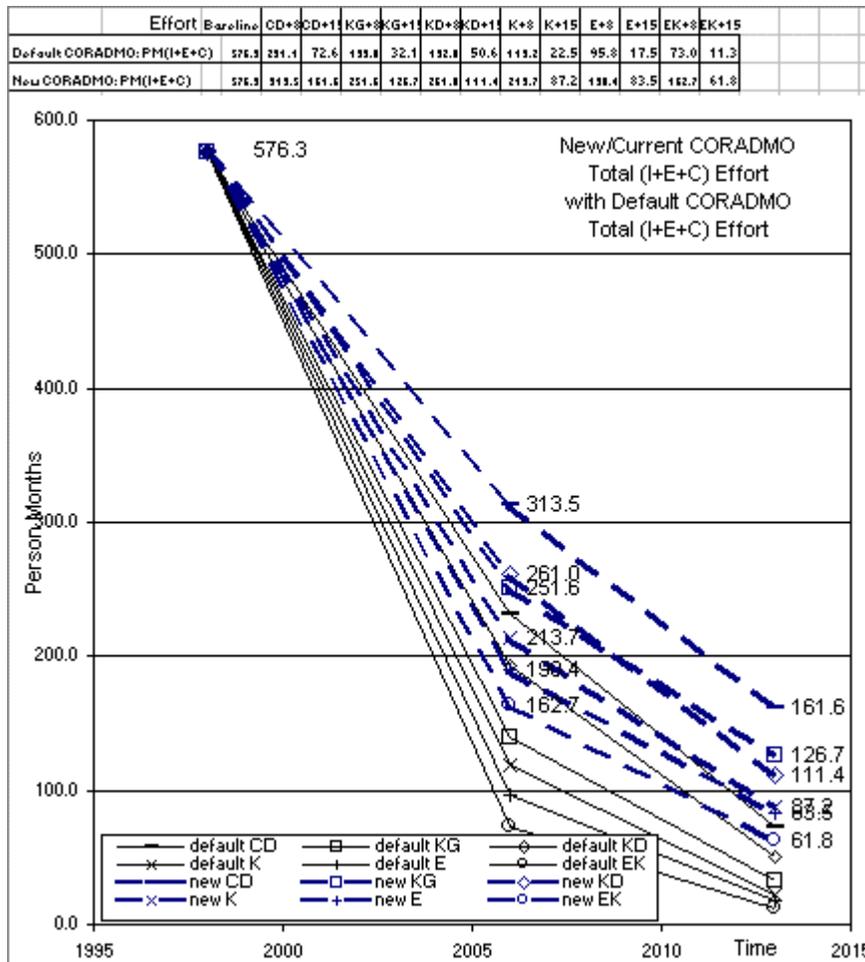
Effort	Baseline	CD+%	CD+15	KG+%	KG+15	KD+%	KD+15	K+%	K+15	E+%	E+15	EK+%	EK+15
Default CORADMO: PM(I+E+C)	576.3	291.1	72.6	133.1	32.1	192.1	50.6	113.2	22.5	95.8	17.5	73.0	11.3
New CORADMO: PM(I+E+C)	576.3	313.5	161.6	251.6	126.7	261.1	111.4	213.7	87.2	198.4	83.5	162.7	61.8

Figure 20. Comparisons of Effort Final Results for Default and New Drivers

Here, both the default and new final results calculations are shown in the table above the chart. Again, the new final results row's values will contain the results based on the "current" CORADMO driver values, and thus may have changes anytime there is input in the "new" row of the drivers. While only the data associated with the bottom row of the table, which contains the new calculation results, is shown in the chart, the default results values are evident due to the solid lines appearing in the chart.

The corresponding schedule final results comparison is shown in Figure 21.

46

Schedule (I+E+C)		Baseline	CD+8	CD+15	KG+8	KG+15	KD+8	KD+15	K+8	K+15	E+8	E+15	EK+8	EK+15
Default CORADMO: M(I+E+C)	M	33.8	23.0	14.3	18.9	6.7	20.7	9.6	17.3	4.9	16.1	4.1	14.3	3.08
New CORADMO: M(I+E+C)	M	33.8	25.3	18.5	22.7	16.2	22.8	14.8	20.8	13.3	20.0	13.5	18.3	10.5
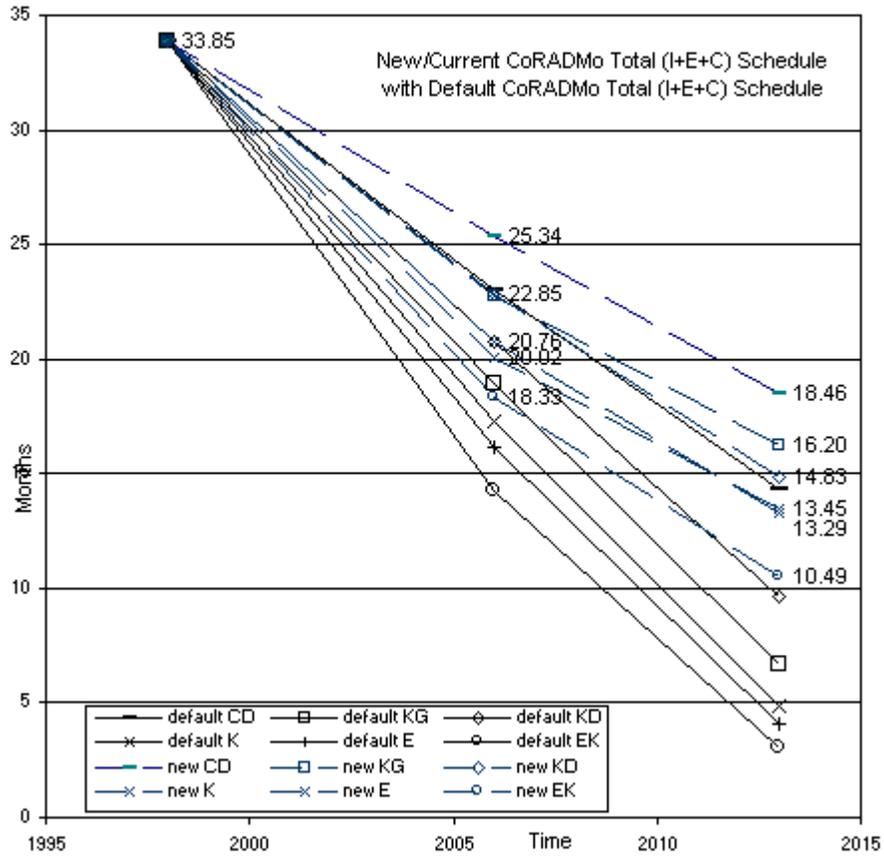
Figure 21. Comparisons of Schedule Final Results for Default and New Drivers

G. Technical Transition Activities

The Bayesian approach for calibrating parametric models to a mix of expert-consensus-determined values and project data [Chulani et al., 1998] has been applied to COCOMO II.1998, and results reported at the 1997 COCOMO/Software Cost Modeling Forum, the 1998 International Conference on Software Engineering, and the 1998 ISPA/SCEA Conference. It is also being applied to USC-CSE's Constructive COTS Integration cost model (COCOTS) and Constructive Quality Estimation Model (COQUALMO), initially reported at the 1997 California Software Symposium [Chulani, 1997].

These and the other recently completed models discussed in this report (CORADMO, COSSEMO, and the Software Technology Impact Assessment Model) were presented at the 1998 COCOMO/Software Cost Modeling Forum in October 1998. Concurrently with this Forum, USC-CSE held an Affiliates' Workshop on COCOMO II Extensions to provide the models to the Affiliates and discuss their usage and refinement.

The preliminary results of the KBSA Life Cycle Evaluation were briefed to AFRL/IFTD personnel at Rome on July 8, 1998. The final results will be briefed to selected Air Force and DoD software technology managers.

As indicated in Section I, we are also making the results and model available via the USC-CSE Web site.

H. Summary and Lessons Learned

1. *The KBSA Advanced Development Model was not sufficiently robust and scalable to provide appreciable benefits to critical DoD software projects.* We found its framework difficult to scale to large projects, and its toolset to be small with respect to large project needs.

2. *However, the KBSA concepts, if otherwise realized, have strong potential for reducing costs and schedules for DoD-critical warfighting software applications. The key to realizing this potential is to develop the KBSA technology in the context of DoD warfighting domain knowledge.* For KBSA Applications Generation, this implies focusing on DoD warfighting areas where a good start exists towards domain engineering, domain architecting, and development and use of reusable domain assets. Some of the USAF/ESC Product Line areas are good candidates, as are some avionics, missile, and sensor processing domains.

For KBSA Decision Support, this involves both domain-specific decision aids and more generic decision aids for security, survivability, risk assessment, architecture definition, and process definition and management.

3. *The KBSA Decision Support area provides an attractive new direction for Air Force and DoD software technology investments.* Knowledge-based and agent-based mixed-initiative technical approaches are generating high payoffs in decision support areas similar to software engineering such as manufacturing and logistics. A similar initiative for software engineering would build on existing technology in attractive new directions.

4. *EDCS technology investments and payoffs have both commonalities and complementarities with respect to KBSA technology investments and payoffs.* To some extent, good portions of a KBSA Applications Generation program have been pursued under EDCS. A KBSA Decision Support program could build upon and provide new directions for some current EDCS efforts in such areas as rationale capture, design webs, and architecture technology.

5. *The Software Technology Impact Analysis spreadsheet model provides a useful tool for assessing alternative software technology assessment strategies.* It enables technology managers to relate candidate technologies' effects on software cost and schedule drivers to resulting project cost and schedule improvements. It is well calibrated and baselined with respect to current software practice and commercial technology and productivity trends.

6. *There is no software technology silver bullet for DoD warfighting systems, including reliance on commercial technology.* A mixed strategy of commercial technology and complementary DoD warfighting software technology is necessary to address DoD's full range of needs.

I. World Wide Web Home Pages and References

I.1 World Wide Web Home Pages

1. USC Center for Software Engineering:
   http://sunset.usc.edu/

2. Technical Reports:
   http://sunset.usc.edu/TechReports/electronicopy.html

3. COCOMO II Research Program and Results:
   http://sunset.usc.edu/COCOMOII/cocomo.html

4. Software Technology Impact Assessment Model:
   http://sunset.usc.edu/COPROMO/KBSA_LCE/kbsa_lce.html

I.2 References

[Basili et al., 1995]. V. Basili, M. Zelkowitz, F. McGarry, J. Page, S. Waligora, and R. Pajerski, "SEL's Software Process Improvement Program," <u>IEEE Software</u>, November 1995, pp. 83-87.

[Bernstein, 1997]. L. Bernstein, "Software Investment Strategy," <u>Bell Labs Technical Journal</u>, Summer 1997, pp. 233-242.

[Boehm, 1981]. B. Boehm, "Software Engineering Economics," Prentice Hall, 1981.

 [Boehm – Standish, 1983]. B. Boehm and T. Standish, "Software Technology in the 1990's: Using an Evolutionary Paradigm," <u>IEEE Computer</u>, November 1983, pp. 30-37.

[Boehm et al., 1995]. B. Boehm, B. Clark, E. Horowitz, R. Selby, and C. Westland, "Cost Models for Future Software Processes: COCOMO 2.0," <u>Annals of Software Engineering</u>, Vol. 1, No. 1, 1995.

[Boehm, 1996]. B. Boehm, Anchoring the Software Process," <u>IEEE Software</u> Vol. 13, No. 4, July 1996, pp. 73-82.

[Boehm - Chulani - Egyed, 1997]. B. Boehm, S. Chulani, and A. Egyed, "Knowledge Summary: USC-CSE Focused Workshop on Rapid Application Development," USC-CSE Technical Report, June 1997.

[Chulani, 1997]. S. Chulani, "Modeling Software Defect Introduction," <u>Proceedings, 1997 California Software Symposium</u>, UCI/USC, November 1997, pp. 41-49.

[Chulani et al., 1998]. S. Chulani, B. Boehm, and B. Steece, "Calibrating Software Cost Models Using Bayesian Analysis," USC-CSE Technical Report CSE-98-508, June 1998.

[De Bellis et al., 1992]. M. De Bellis, K. Miriyala, S. Bhat, W. Sasso, and O. Rambow, "Final Report: KBSA Concept Demonstration System," Rome Labs Contract F30602-89-C-0160, October, 1992.
[DoD, 1991]. U.S. Department of Defense, "Draft DoD Software Technology Strategy," ODDR&E, December 1991.

[Fawcett et al., 1997]. J. Fawcett, B. Brunk, K. Ganesh, and U. Parvate, "Evaluation of a KBSA Advanced Development Model," Syracuse University Report, May 1997.

[Fox et al., 1995]. J. Fox et al., "ARPA Software Review Panel: Final Report," October 1995.

[Garlan et al., 1995]. D. Garlan, R. Allen, and J. Ockerbloom, "Architectural Mismatch: Why Reuse Is So Hard," <u>IEEE Software</u>, November 1995, pp. 17-26.

[Green et al., 1983]. C. Green, D. Luckham, R. Balzer, T. Cheatham, and C. Rich, "Report on a Knowledge Based Software Assistant," Kestrel Institute Report, prepared for RADC, June 15, 1983.

[Jacobson et al., 1997]. I. Jacobson, M. Griss, and P. Jonsson, <u>Software Reuse</u>, Addison Wesley Longman, 1997.

[Jones, 1990]. C. Jones, "Applied Software Measurement," McGraw-Hill, 1990.

[NAE, 1995]. National Academy of Engineering, "Defense Software Research, Development, and Demonstration: Capitalizing on Continued Growth in Private-Sector Investment," NAE Workshop Report, 1995.

[NRC, 1997]. National Research Council, "Ada and Beyond: Software Policies for the Department of Defense," National Academy Press, Washington D.C., 1997.

[Poulin, 1996]. J. Poulin, <u>Measuring Software Reuse</u>, Addison Wesley, 1996.

[Rational Corp., 1998]. "Rational Objectory Process 4.1 – Your UML Process," available online at http://www.rational.com/support/techpapers/toratobjprcs/, Rational Corporation, 1998.

[Reifer, 1997]. D. Reifer, <u>Practical Software Reuse</u>, John Wiley and Sons, 1997.

[SAB, 1995]. USAF Scientific Advisory Board, <u>New World Vistas: Information Technology Volume</u>, USAF-SAB, 1995.

[SAB, 1997]. USAF Scientific Advisory Board, <u>Report on United States Air Force Expeditionary Forces</u>, SAB-TR-97-01, November 1997.

[Salasin – La Monica, 1998]. J. Salasin and F. La Monica (eds.), "EDCS Demonstration Days '98: A Revolution in Software Evolution," DARPA/AFRL-ITD, July 1998.
[Sasso, 1997]. W. Sasso, "Empirical Evaluation of KBSA Technology," Rome Laboratory Report RL-TR-97-49, July 1997.
[Sasso-Benner, 1995]. W. Sasso and K. Benner, "An Empirical Evaluation of KBSA Technology," <u>Proceedings, KBSE 95</u>, IEEE Press, November 1995, pp. 71-78.
[White, 1991]. D. White, "The Knowledge-Based Software Assistant: A Program Summary," <u>Proceedings, Sixth Annual Knowledge-Based Software Engineering Conference</u>, IEEE, September 1991, pp. 2