DRAFT SF 298

1. Report Date (dd-mm-yy) March 1994	2. Report Type	3. Dates covered (from... to)
4. Title & subtitle Process Technologies Method and Tool Report. Volume 1		5a. Contract or Grant # 5b. Program Element #
6. Author(s) Allgood, B; Clough, A; Cunha, G; VanBuren, J		5c. Project # 5d. Task # 5e. Work Unit #
7. Performing Organization Name & Address		8. Performing Organization Report #
9. Sponsoring/Monitoring Agency Name & Address Software Technology Support Center OO-ALC/TISE 7278 4th Street Hill AFB, UT 84056-5205		10. Monitor Acronym 11. Monitor Report #
12. Distribution/Availability Statement Distribution Statement A: Approved for public release, distribution is unlimited.		
13. Supplementary Notes		
14. Abstract 19970516 153		
15. Subject Terms		

Security Classification of			19. Limitation of Abstract	20. # of Pages	21. Responsible Person (Name and Telephone #)
16. Report Unclass	17. Abstract Unclass	18. This Page Unclass			Randy Wright (801) 777-9732

STSC

Process Technologies Method and Tool Report (Volume I)

March 1994

Bruce Allgood
Anne Clough
Gary Cunha
James VanBuren
Process Technology Evaluation Project
Sam Godfrey
Technical Program Manager

This technical report was prepared by the

Software Technology Support Center
Ogden ALC/TISE
Hill AFB, UT 84056

The ideas and findings in this report should not be construed as an official Air Force position. It is published in the interest of scientific and technical information exchange.

This document is available through the STSC. To obtain a copy, please contact the STSC Customer Service Office directly at: Software Technology Support Center, Attn.: Customer Service, Hill AFB, Utah 84056; 801-777-7703 or DSN 458-7703, Fax 801-777-8069, e-mail godfreys@wpo.hill.af.mil.

Preface

This document, Volume I of the Process Technologies Report 1994, is a synopsis of the progress of the Software Technology Support Center (STSC) in evaluating process technologies. A software process is a set of activities, methods, practices, and transformations that people use to develop and maintain software and associated products.[1] Process technologies are defined as those technologies that can be applied to support an organization's software process.

Volume II of this report contains detailed information on process products, user critiques of products, and IDEF training information. To order Volume II of this report, contact the STSC customer service department at (801)777-7703 or DSN 458-7703, fax to (801)777-8069 or DSN 458-8069, or email to godfreys@wpo.hill.af.mil.

The targets of this report are organizations responsible for the development and maintenance of computer software. The information is aimed at those who must make the decisions about acquiring advanced process technology and prepare their organizations for its effective use. The assumption is that the reader will be a software manager or practitioner with limited knowledge in the area of software process technology, working in his/her organization or software project in support of software process (or sub-process) improvement.

Areas covered in the report include process assessment, process modeling, and process enactment, where process enactment is defined as the use of a formal process definition to guide and control the software process. Both process methods and computer aided process engineering tools are included. The report defines process technologies, identifies tools and software engineering environments that support process technologies, discusses the value of emphasizing process in improving software quality, and examines the effective use of process technologies. Volume II includes information about specific products in the marketplace. This report also attempts to identify the future directions of process technologies to help in planning long-range strategies.

---

[1] Paulk, M., Curtis, B., Chrissis, M.B. et. al., "Capability Maturity Model, Version 1.1," IEEE Software, Vol. 10, No. 4, July 1993.

The authors would like to thank John Brackett of the College of Engineering, Boston University, for his careful review, comments and input to this report. SEI reviewers; Steve Lipka from Lipka Software Engineering; Ralph Ganska from Draper Laboratory; and Jesse Foster, Bob Hanrahan, Dennis Barney, Judi Peterson, Larry Smith and Shane Atkinson from the Software Technology Support Center (STSC) have also provided much guidance and have made important contributions to the report. In addition, the authors would like to thank those who provided product critiques for this report. Their names have been withheld to protect their privacy.

Contents

Contents Continued

Contents Continued

Contents Continued

1 PROCESS TECHNOLOGY

In this era of increasingly complex software and mushrooming software development and maintenance costs, the software engineering community needs to improve its practices to remain competitive and to produce software that can meet complex software requirements. Projects often overrun costs, miss deadlines, and fail to meet the requirements of the customer. Now new software process technologies are evolving to help address these concerns in the areas of process *assessment, definition, simulation,* and *enactment.* A *software process* is a set of activities, methods, practices, and transformations that people use to develop and maintain software and associated products.[2] The quality of a product stems, in large part, from the quality of the process used to create it. To consistently improve products, the process used for developing them should be understood, defined, measured, and progressively improved. Software process *assessment* is the act of determining the maturity[3] of an organization's software process. Software process *definition* is the act of specifying in some detail an organization's software process. Software process *simulation* is the act of executing a software process definition. The term *modeling* will often be used in this report to encompass both process definition and simulation. The term *enactment* denotes the use of a formal process definition to guide and control the software process. In this report, software process technologies will be described, their benefits outlined, and their implications assessed. Technology maturity will be discussed and recommendations will be given for technologies that can be effectively used today.

The report provides a survey of contributors to process technology. Comprehensive lists, product sheets and critiques of methods and tools are included. An annotated bibliography is available for those who want more background information about process technologies in general or need additional information about a specific method/tool. Guidelines are supplied for adopting process technologies and for assessing the appropriate level of process technology for an organization. Checklists are provided to help prioritize technology requirements and differentiate between similar methods and tools.

---

[2] Paulk, M., Curtis, B., Chrissis, M.B. et. al., "Capability Maturity Model, Version 1.1," IEEE Software, Vol. 10, No. 4, July 1993.

[3] Process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective.

Research for this report took place over a period of 33 months and consisted of a thorough review of the technical literature, meetings with technology researchers/developers, and conversations with technology users. Applicable tools and techniques were identified by attendance at conferences and trade shows, use of tool databases[4], and information from practitioners. For each tool/technology identified, the authors spoke directly to the vendor or researcher; product sheets - using a template provided by the authors - were filled out by many vendors and researchers. Critiques were solicited from users of tools and technologies. During this period, the report authors were also actively involved in using process technologies, particularly in the areas of IDEF modeling, system dynamics modeling and process improvement. The authors participated in process technology insertion projects which included both modeling assistance and IDEF training.

1.1 The Software Process Technology Domain

In this section, we will discuss the software process technology domain - its history, recent advances and current use. To help the reader focus on particular aspects of the domain, a road map for using this document is then provided for effective use of the information in the report.

Often we think of process improvement and/or business process redesign as new concepts, when in fact similar ideas exist dating back to the early twentieth century. In 1927, Walter A. Shewhart, a statistician at Bell Labs, devised a technique to bring industrial processes into what he called "statistical control." This plan, called the *Shewhart Cycle* comprises four steps: [5]

1) Study a process to decide what change might improve it. (PLAN)
2) Carry out tests or make the change, preferably on a small scale. (DO)
3) Observe the effect. (CHECK)
4) Gather lessons learned (ACT)

---

[4] Two tool databases provided significant assistance: CASEBase and Tool Finder/Plus. CASEBase is a product of P-Cube Corporation, Brea, CA; Copyright 1990,'91'92: "CASE Product Comparison Information." Tool Finder/Plus is a PC-based, CASE tool database with automated search capabilities available from the C/A/S/E/ Consulting Group, Lake Oswego, Oregon.

[5] This information was taken from John A. Zachman's presentation, "Business Transformation: Key to Global Competitiveness," at the IDEF Users Group Conference Fall 1993.

W. Edwards Deming, whose application of statistical control in Japanese industries after World War II emphasized the importance of process, stated, " Every activity is a process and can be improved. Everybody belongs on a team to work on the Shewhart cycle to address one or more specific issues." His work has led naturally to the total quality management and process improvement initiatives.

Total Quality Management (TQM) is a philosophy that recognizes that people want to do a good job, that all tasks/systems/processes can be made to work better, and that those who are closest to and working with a particular process can best improve it. The TQM process encompasses both the search for areas that need to be and can be improved, and the implementation of improvements. The importance of process is emphasized. The software process improvement thrust can be considered in some sense to be a subset of TQM. Software process technologies can be used effectively in support of process improvement by assisting in assessing the effectiveness of software processes, defining optimized processes and automating the use of well-defined software processes.

The concepts of statistical process control have been applied in industry for many years. Definition/simulation methods have been employed in software development. However, applying these technologies to the software process is a relatively recent phenomenon; software process technology is a relatively recent addition to software technologies. In reference to software process specifically, Leon Osterweil in his paper, "Software Processes are Software Too," advocated the use of "process programming" to define software development processes in 1987.[6] Around the same time, others in the software community began to advocate the use of process orientation in software engineering environments[7].

---

[6] Osterweil, Leon, "Software Processes are Software Too", **Proceedings - 9th International Conference on Software Engineering**, March 1987.

[7] A Software Engineering Environment (SEE) is defined as a collection of integrated tools and methods providing automated support for the development and maintenance of software. See the STSC Software Engineering Environment Report, Spring 1994, for a complete discussion of the software engineering environment domain.

Realizing that focusing on process would provide an effective mechanism for improving quality, productivity, and predictability in software development, STARS[8] began to concentrate on process-driven development in the early 1990s. Their initial survey of technologies in the process area showed little uniformity of approach and largely immature, research-oriented work. During the first years, STARS process efforts were focused on experimenting with and evaluating a number of point solutions. They were supported in this by their major contractors IBM, Boeing, and Paramax. However, STARS' overall mission has always been to incorporate technologies to define the software process, install the process into an organization's work environment, and enact the process, while focusing on monitoring and measuring the process in order to evaluate and improve it. Joint service/STARS demonstration projects examining process-driven concepts and modern reuse concepts within supporting software engineering environments commenced in 1993 and will continue through 1995.

More recently, commercial interest in process technologies has become prevalent. Methods and tools are more robust. However, each of the three major process technology thrusts (process assessment, process modeling, and process enactment) are at different developmental stages. It is the authors' opinion that a number of assessment and modeling technologies are mature enough to be applied to help software development and maintenance efforts. A major benefit from using process technologies will come from assessing the maturity of existing software processes and then providing a clear process definition, defined in sufficient detail to provide a clear set of objectives that, intelligently applied and modified to suit the problem at hand, will lead to beneficial process implementation for the target product and organization. SEI assessment procedures are robust, well defined and accepted by the software community. Well defined software processes exist. Working to assess and define the software process is a realistic goal, which will position the organization for continuous process improvement.

With enactment technologies, however, the picture is a bit more confusing. When a process is enacted, that process has to be defined to a level of detail sufficient to allow the

---

[8] The Software Technology for Adaptable, Reliable Systems (STARS) Program is sponsored by the Advanced Research Projects Agency (ARPA), contracted through Air Force Electronic System Division, and involves three cooperating prime contractors - Boeing, IBM, and Paramax - and a large number of subcontractors. The STARS goal is to increase software productivity, reliability, and quality by synergistically integrating support for modern software development processes and modern reuse concepts within state-of-the-art software engineering environment technology.

process model to be followed in a uniform way by process users. This detailed definition permits software tools to be invoked at the appropriate times, guidance to be given to users of the process, project metrics to be recorded, and management reports on project status to be generated. Of course, it is possible to accomplish all these objectives without a process-driven computer environment to support the enactment. When a detailed definition is followed for a process with little or no tool/environment support, the terminology "human enactment" is used. Human enactment can be accomplished whenever a process has been defined to the necessary level of detail.

The term "computer enactment" is used to denote the situation when a process-driven computer environment is used to support process enactment. An organization is not prepared for machine/computer enactment until it has a well defined process that people are able to enact without software engineering environment support. However, even when this level of definition is available, effective environment support for process enactment may not yet be developed to the required level of sophistication, nor has anybody proved the cost effectiveness of computer enactment except for very small pieces of the software process. However, continuous monitoring and evaluation of process-driven environments, a course of action begun in this report, will allow us to determine their value as research, development, and pilot use of such environments increases.

A great deal of information is included in this report. Therefore, a road map of how to effectively use it may be helpful. The sections that are relevant depend on an organization's process maturity. An organization just beginning to address process improvement, should focus on:

Section 1.3: Assessment Technology
Appendix D.2: Assessment Method Checklist
Appendix G.1 (Bibliography): Process Assessment

Once process improvement activities begin, the parts of the report dealing with process modeling will be relevant. Process definitions can be used to support the implementation of a process at all maturity levels. The sections focusing on process modeling are:

Section 1.4: Process Modeling
Appendix A.2: Process Modeling List
Appendix B: Process Technology Product Sheets (See Volume II)
Appendix C: Process Technology Product Critiques (See Volume II)
Appendix D.3: Modeling Checklist
Appendix E: IDEF Technology
Appendix G.3 (Bibliography): Process Modeling - Overview and General Articles
Appendix G.4 (Bibliography): Process Modeling - Specific Methods and Tools

The sections of the report on process enactment, particularly computer-assisted enactment, may not address immediate process concerns. It is the authors' opinion that only the most mature organizations have an optimized process defined to the detailed level necessary to invest resources in computer enactment. However, the following sections on computer enactment will serve as an introduction to this area and a summary of state-of-the-art products and research for computer-assisted enactment:

Section 1.5: Computer Enactment
Appendix A.3: Process Frameworks List
Appendix A.4: Computer Enactment Technology List
Appendix A.5: Process Driven Environment List
Appendix B: Process Technology Product Sheets (See Volume II)
Appendix C: Process Technology Product Critiques (See Volume II)
Appendix D.4: Enactment Software Engineering Environment (SEE) Checklist
Appendix G.5 (Bibliography): Process-Driven Environment - Comparative Assessments and General Articles
Appendix G.6 (Bibliography): Specific Process-Driven Environment and Enactment Technologies

1.2 Process Technology Categories

Process technologies can be classified in a number of useful ways. The STSC uses three categories to differentiate between process technology areas. They are:

- Process Assessment
- Process Modeling[9]
- Computer Enactment

This taxonomy has proven useful for aggregating like technologies and allowing the reader to concentrate on all related technologies in a technology area.

Another category that could be discussed when looking at process technologies is process measurement. Particularly when using process definitions to improve an organization's software process, it is important to measure the current process and then measure the "improved" process to see if a proposed change is a step forward or a step backward. Measurement plays a very central role in process technology, driving the evolution and improvement of process definitions and software processes. It guides decision-making during process definition and provides evidence to evaluate which process steps are working effectively and which are not, leading to process improvement.[10] However, due to the amount of research necessary to produce a first report covering process assessment, modeling and enactment, process measurement as a separate category is not discussed in this report. It will be a candidate for inclusion in the next edition of this report.

1.2.1 Process Assessment

An organization first needs to evaluate, or *assess,* the maturity of the software process already in place. Knowing the maturity level of its development and maintenance process or processes will allow an organization to concentrate on improvements that attack immediate problems. Assessment technologies will be discussed in Section 1.3.

1.2.2 Process Modeling

In order to improve the software process, a *defined* software process is necessary. Richard Drake from STARS/IBM describes a defined process as one that is documented, taught, and applied. A defined process implies that, for some application domain, everyone

---

[9] An encompassing term that includes both process definition and simulation.

[10] Hal Hart, "Process Measurement, " STARS '91

uses the same software process and a description of that process exists. Often, the process description is a textual description that is available to all software practitioners – for example, in an organization's procedures manual. A defined process provides a basis for examining and improving the software process, predictability in cost and schedule, better understanding of roles and relationships, guidance of software professionals through choices in an orderly way, and a consistent working framework that allows staff to move easily from one project to another. The very act of precisely defining the process being used by an organization usually reveals anomalies in the process and leads to immediate improvements.

Effective definition methods and notations can support process definition. An industry-wide standard process definition method and notation does not exist. Process notations used to define the software process vary in formality from free-form English language descriptions at one end of the spectrum to formal machine-executable definitions at the other. Since effective process notation is necessary for successful process definition, research and development of notations that allow adequate precision is an important and growing technological priority. Our compilations of modeling technologies will include a number of these research efforts as well as commercially available, tool-supported methods.

Process definitions become even more powerful when they allow the *simulation* of the software process. The ability to simulate the process is a characteristic of some process definition approaches. In fact, the formality and strength of a definition technology rests, at least in part, on its ability to support simulation. Simulation allows those defining the process to test the definition for errors and completeness. "What-if" analyses can be made available to managers who wish to test the effect of alternative strategies on the overall process. In addition, formal analysis can be used to detect weaknesses and bottlenecks in the software process itself. Effective simulation technology (for example: Petri nets, real-time structured analysis, real-time systems specification and design, system dynamics) and tool support are widely available today, though not widely used for software process simulation.

The term *process modeling* will be used in this report to encompass both process definition and simulation. Process modeling will be discussed in more detail in section 1.4, the various IDEF modeling methodologies in Appendix E.

1.2.3 Computer Enactment

Ideally, in the vision of software technologists, a precisely defined software process definition will ultimately be used in a computer aided support environment to invoke software tools at appropriate times, determine compliance with the process model, give guidance to the users, record project metrics, keep management informed of the status of a project, and execute any automatable activities in the process model. When a formal process definition is used to guide and control the software process in this way, we say that the definition has become *enacted.* However, given the lack of fully realized process-driven environments, installing totally effective environment support for enactment is probably not possible at this time.

Computer enactment will be further discussed in Section 1.5, and specific products supporting computer enactment presented in Appendix F.

1.3 Process Assessment Technology

Software process assessments are used by organizations to help identify the status of their software process and to identify areas to address for process improvement. A software process assessment requires a review of the software development process for key projects in an organization. An assessment identifies an organization's key strengths and weaknesses and helps the organization establish effective improvement plans. There are a variety of approaches that can be taken to assess the maturity of a software process. However, the purpose of this document is not to evaluate assessment technologies but to introduce the need for process assessment and provide a pointer to the Software Engineering Institute (SEI) for DoD customers, since the most widely known and used approach, particularly in the DoD community, is the SEI assessment process.

'Quality Management and Quality Assurance Standards' (ISO 9000 series standards), developed by the International Organization for Standardization (ISO) in 1987, are a set of quality assurance standards that can be applied to any business. The ISO's intent is to keep the need for onsite vendor/contractor visits to examine their quality assurance methods to a minimum by creating standards for products crossing international borders and within the European Community. This assessment method has been adopted by over 60 countries, but remains primarily outside the DoD realm. These standards describe the steps

necessary to achieve an ISO 9000 certification, and were designed to address several business realms including manufacturing, service, software development, supply, and maintenance. In comparison, the CMM identifies the characteristics of an organization at a specific level of maturity, and was designed specifically for software developers. While some people working with the CMM also help in improving the ISO 9000 series, the overlap between the two methods in not clear. [11]

Process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. The Software Engineering Institute (SEI) at Carnegie Mellon University has been a leader in the area of process assessment, defining five distinct maturity levels for categorizing software processes.[12] At Level 1 (Initial), the software process is unpredictable with respect to cost, schedule and quality. Success largely depends on individual effort. At Level 2 (Repeatable), basic product and process controls are in place. This includes project management (planning and tracking), process and product assurance (SQA), and change management (requirements management and software configuration management). The necessary process discipline is in place to repeat earlier successes on projects with similar applications. At Level 3 (Defined), the software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for development and maintenance of software. At Level 4 (Managed), detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. Data is available to establish improvement priorities and to support tool and technology investment. At Level 5 (Optimizing), continuous process improvement is enabled by quantitative feedback from the process and by piloting innovative ideas and technologies.[13] When an organization uses an SEI assessment process to measure its process maturity level, specific steps are identified that enable the organization to improve its process and advance from one maturity level to another. Using this approach, an organization can

---

[11] Information on the ISO 9000 series was taken from Mark Dawood's article "It's Time For ISO 9000", published in the March 1994 issue of CrossTalk.

[12] Throughout this report the term "Level X" will be used to refer to "SEI CMM Level X."

[13] The five maturity levels are explained more fully by Watts Humphrey in his book, "Managing the Software Process," Addison-Wesley Publishing Company, Inc., 1989, and in "Capability Maturity Model for Software, Version 1.1," Technical Report CMU/SEI-93-TR-24, Software Engineering Institute, February 1993. The short descriptions above are from the technical report.

systematically move from "initial" processes to "repeatable," "defined," "measured," and finally "optimized" processes.[14]

The SEI process assessment has proved to be an effective way for an organization to identify what must be done to improve its software process. Assessors are trained to evaluate strengths and weaknesses in areas such as project planning, project management, configuration management, quality assurance, standards and procedures, training, process focus, and peer reviews/testing. An assessment determines what areas need to be improved and establishes priorities for the improvement effort. Project management is often the first area an organization needs to address. Concentrating on addressing weaknesses, in the prioritized way provided by the assessment, focuses the improvement effort in ways that can realize immediate payoff. However, more specific assistance from the SEI on how to address the weaknesses would definitely be desirable.

The SEI's process assessment procedure, with its maturity levels, improvement steps, emphasis on an action plan for improvement, concept of software engineering process groups (SEPG)[15], and process training, is rapidly becoming an industry/DoD standard. Some government agencies are now using a closely related procedure called the Software Capability Evaluation (SCE) to judge how capable private companies are at developing software. The SEI has done a thorough job in training people to apply the assessment procedure. They have conducted assessments, measured results and addressed shortcomings in the original procedures with an updated version of both the assessment procedure and the corresponding maturity levels. While the process assessment procedure is pretty stable now with infrequent changes and updates, it continues to be reviewed by a wide range of industry reviewers and will continue to be revised and updated as necessary. Though some critics have attacked the SEI approach, organizations have reported (and measured) a significant gain in the effectiveness of their software process after implementing SEI's recommended

---

14 Mr. Lloyd K. Mosemann's goal to achieve a maturity level 3 by 1998 for Central Design Activities / Software Design Activities and weapon systems Software Support Activities.

15 The Software Engineering Process Group (SEPG) acts as the focal point of a software engineering process improvement program. Working with managers and engineers from software development organizations, the SEPG tracks, screens, installs, and evaluates new methods and technology that can improve the software engineering capability of an organization.

action plan.[16] Organizations can perform self-assessments (ideally after being trained by the SEI) or the Software Engineering Institute can be contacted for a list of Software Process Assessment Associates,[17] which are companies trained, authorized and licensed by the SEI to perform software assessments. It should be recognized that assessments can be costly, whether they are done by an internal team or by an external assessment team.

Within the Air Force, XPSP (formerly TIC) is a group trained by the SEI to carry out formal software process assessments of Air Force organizations (not project teams). The Air Force Communication Command (AFCC) at Scott AFB is the primary source for these trained assessors. Their effort supports Mr. Lloyd K. Mosemann's goal "to achieve a maturity level 3 (defined process) by 1998 for Central Design Activities / Software Design Activities (CDA / SDA) and weapon systems Software Support Activities (SSA)."[18] Mr. Mosemann is the Deputy Assistant Secretary of the Air Force for communications, computers, and support systems in the office of the Assistant Secretary for Acquisition.

An alternative approach to process assessment, the Model-Based Process Assessment (MBPA), has been proposed by Clement McGowan.[19] This approach combines process modeling with process assessment. The MBPA advocates creating a model of a process and using this model as the basis for assessing a process. McGowan maintains that the modeling approach will often lead to process improvements that might have been missed using the SEI methodology. Major points of difference with the SEI approach are:

- MBPAs can be applied to any process, whereas the SEI approach is concerned solely with software process.
- An SEI assessment reviews multiple projects within an organization, while an MBPA focuses on a single process/project.

---

16 See Humphrey, W., Snyder, T.R., and Willis, R.R., "Software Process Improvement at Hughes Aircraft," **IEEE Software**, July 1991; Dion, R. "Cost of Quality as a Measure of Process Improvement," 1992 SEI Symposium; OC-ALC/LAS white paper, Oklahoma City Air Logistics Center, Tinker AFB.

17 For information on Software Process Assessment Associates, SEI training and other SEI offerings, contact Software Engineering Institute, ATTN: Customer Relations, Carnegie Mellon University, Pittsburgh, PA 15213-3890; telephone (412) 268-5800; internet: customer-relations@sei.cmu.edu.

18 This quote is taken from Lloyd Mosemann's memorandum, "Policy on Software Maturity Assessment Program," September 1991.

19 McGowan, C.L., and Bohner, S.A., "Model Based Process Assessments," **Proceedings of the 15th International Conference on Software Engineering**, pp. 202-211, Baltimore, Maryland, May 1993.

- MBPAs do not have a nominal reference model (like the CMM).
- Predefined key process areas largely determine the resulting recommendations of an SEI assessment, whereas MBPAs recommendations will come directly from studying the "as-is" model of how a process is accomplished.
- MBPAs take, on average, twice the staff labor of an SEI assessment; in an MBPA, assessors often help implement the changes that they recommend.

Despite the extra effort such an alternative would require, McGowan believes that "a process model (with true consensus on its contents) is a meeting ground for process and project staff to work jointly on improvements," and is therefore preferable to attempting to implement improvements where consensus has not been achieved.

However, it could be argued that a process assessment by definition is a comparison of a process to some reference model. By this definition, the process modeling and analysis found in the MBPA is not process assessment. It is process modeling and analysis - a classic systems analysis technique. MBPAs do not characterize process maturity. It is also true that the SEI process improvement approach (as contrasted to the SEI assessment approach), as documented in the SEPG guide[20], already includes both process modeling and analysis. Therefore, MBPAs can be considered a subset of the SEI process improvement approach and therefore not an alternative method but a complementary improvement technique. In fact, MBPA is increasingly being understood and welcomed as a complementary technique that has proven useful in a number of process improvement efforts.

1.4 Process Modeling

In general, a process model adequately models a particular process if it can be used to answer questions about the process to a specified tolerance. A great variety of process definition and simulation technologies are being used to do this. Each has strengths for answering a unique subset of questions about the process being defined. Any modeling technology must adequately support both the developer and the user (or reader) of the completed model. Therefore modeling technologies should be examined from the point of

---

[20] Fowler, P, Rifkin, S., "Software Engineering Process Group Guide," Technical Report CMU/SEI-90-TR-24, Software Engineering Institute, September 1990.

view of the questions that can be answered when both a developer and a reader use a model. This approach allows a more informed decision to be made when choosing the modeling technique for an organization. A typical set of process-related questions follows:

A Model Reader's Questions: *A model reader will look to the process model to answer basic questions about the process. The extent to which these questions can be answered (which rests heavily on the understandability and usability of the model) will be a strong indicator of a model's value to the user of that model.*

(1) What is the scope of the model?

- What questions is it designed to answer?
- What is the context for the model?

(2) What are the process steps and their results?

- What are the functions/activities that need to be performed?
- What conditions must be satisfied before an activity can take place?
- What is the sequence of activities?
- What are the rules for feedback or iteration?
- What are the constraints on the process?

(3) What are states of objects produced by or used in performing the process?

- What activities transform objects from one state to another?

(4) Who/what implements the activities?

- What is the mapping of process steps to responsible individuals? To roles? To organizations? To tools?
- Where are the activities implemented?

(5) How can I manage the process?

- What activities can take place concurrently?
- What are the critical path(s)?
- What resources are required for each activity?
- Where is there resource contention?

(6) Why does the process work the way it does?

A Model Developer's Questions: *Besides addressing the model reader's questions, the modeling technique must also meet the requirements of the model builder, as reflected in the following questions.*

(1) What do I want to model, and why do I want to model it?

(2) What notation can effectively represent the model?

(3) How difficult is it to use the modeling technology (notation, method, tool support)? How difficult is it to understand the resulting model? Is the technology more suited to analysis and design (of the process) than presentation, and if so, how can the results be presented?

(4) How widespread is the use of this technology?

(5) How can the model be used?

- Can the model be analyzed for completeness, correctness, and consistency?
- Can the model be simulated?
- Can I use the model for enactment in a software engineering environment?
- Is tool support needed to view/work with the model?

It is important to know what the modeling objectives are before modeling. Understanding the objectives helps define the model's content and defines the end-point for the work. It also circumscribes the set of questions the reader will be able to answer. This is the most important issue a modeler has to address.

The actual process that is being modeled can be viewed from a number of perspectives, such as the functional (what are the process steps?), the organizational (who/what performs each function?), the behavioral (what are the process states?), and the informational perspective (what is the information structure and what are the information relationships?). Figure 1-1[21] shows why a process being modeled needs to integrate a number of perspectives by comparing the definition of a process to a full view of a person working at a computer. You would not have a full view if you could only see from the top, from one side or the other, or from the back. All perspectives need to be taken into account to get a true picture.

---

21 Figure 1-1 is taken from the article by Curtis, Kellner and Over, "Process Modeling," in the Communications of the ACM, Vol. 35, No. 9, September, 1992.

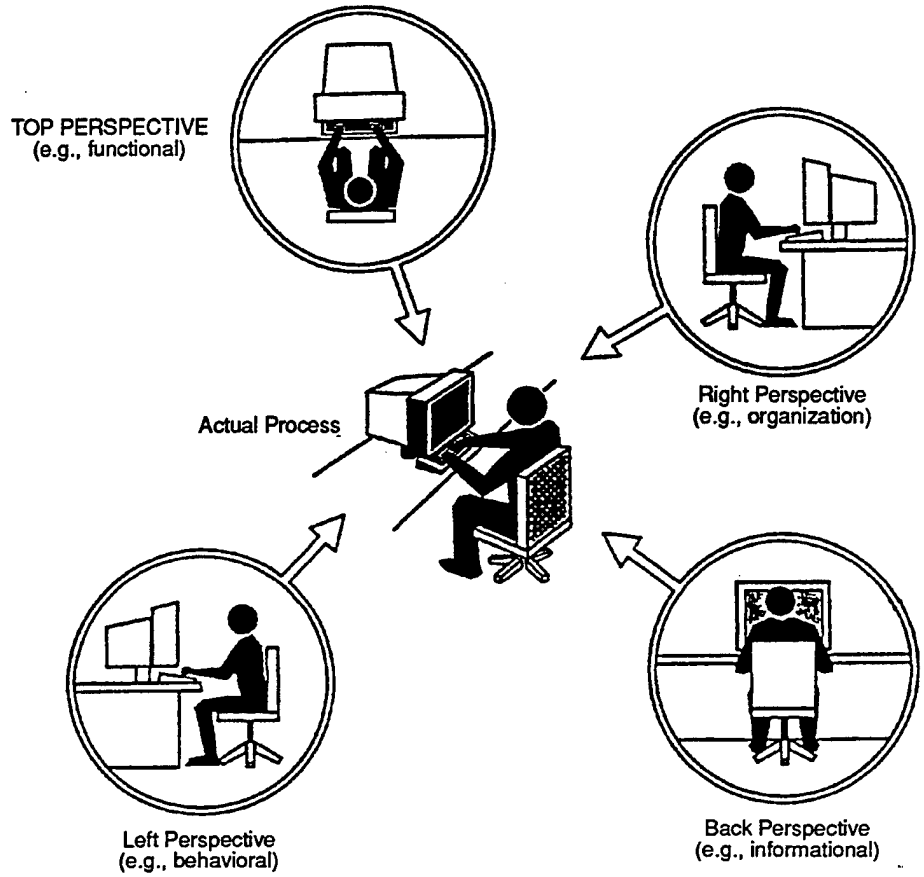

Figure 1-1. Process Perspectives

While it may seem desirable to use a modeling approach that takes all perspectives into account, most modeling approaches emphasize only one or two of these views, which provides an incomplete understanding of the process being modeled. Other perspectives are left out or are added in an incomplete way. Though there are a number of ways that combinations of perspectives can be integrated in a modeling method, no method or tool incorporates all four viewpoints in an integrated fashion.[22] Some argue that it may not be possible to effectively model all perspectives with a single method. This isn't unusual and should not be a surprise. A number of disciplines use multiple models, each of which focuses on a particular perspective. For process modeling, multiple models may also be

---

[22] This statement may need to be revised in the future. Visual Process Modeling Language (VPML), developed by ISSI and supported by their commercially available tool, ProEditor, is designed to address all four perspectives, although the support for some perspectives is minimal at present. VPML and ProEditor will become a part of the Process-Oriented Software Life Cycle Support Environment (ProSLCSE), a process-centered software engineering environment that includes editors and enactment tools.

needed at times. But it is important to note that the perspectives emphasized by a particular approach may provide the desired understanding and be sufficient for the questions the model is designed to answer. What perspective(s) are emphasized by particular modeling methods can be a very important consideration when choosing a modeling approach. Table 1-1 shows the perspectives that can be modeled using the methods that will be discussed in this report.

	Functional	**Behavioral**	**Organizational**	**Informational**
IDEF0	X		X	
IDEF1/IDEF1X				X
IDEF3		X		
Structural Analysis	X			
Real-Time Structured Analysis	X	X		
Entity Process Modeling	X	X	X	
Process Programming	X			
Petri Nets	X	X	X	
System Dynamics	X		X	

Table 1-1. Process Perspectives Supported by Modeling Methods

In the area of software process modeling, there is no single, standard, widely accepted approach. Many modeling techniques can be used for process modeling. A great variety of process definition and simulation technologies, each of which has strengths for answering a unique subset of questions about the process, is being used. Some technologies are clearly at the research stage and tool support is often lacking or rudimentary. However, other technologies (for example, SADT/IDEF0, Petri nets, real-time systems specification and design, and system dynamics, all of which will be defined and discussed in this section) are relatively mature in our opinion and have good tool support.

In section 1.4 subsections, we will look at the major process modeling approaches (structured graphics approaches, process programming approaches, systems dynamics, and Petri net modeling) being developed and/or used today from the perspective of the modeler and also from the perspective of the "reader" of the model. We will also consider the views

(functional, behavioral, organizational and/or informational) supported by the major modeling methods. The bibliography (Appendix G) contains additional information on a great variety of modeling techniques.

1.4.1 Structured Graphic Approaches

We have found the following four categories of structured graphic approaches useful in recognizing differences in methodology:

1) Model the software development functions performed.
2) Model both functions performed and behavior (i.e., model process states).
3) Model functions performed, behavior, and who/what performs each system function.
4) Model information structure and information relationships in the process.

IDEF0 is an example of a technology that falls into the first category, real-time structured analysis falls into the second category, entity process modeling falls into the third, and IDEF1/IDEF1X falls into the fourth. We will look at each separately.

1.4.1.1 Functional Modeling

The first category of structured graphic approaches, which includes structured analysis methods, Structured Analysis and Design Technique (SADT) (which is a structured analysis method), and Integrated DEFinition method (IDEF0) (which is a subset of SADT and the most popular SADT-type method), emphasizes the functional perspective and often employs a graphical hierarchical representation and diagrams which are similar to data flow diagrams. Using functional modeling, process steps can be clearly identified. The use of a data flow diagramming approach for process definition also offers the understandability of such diagrams, thus facilitating communication and information exchange.

If we look at IDEF0 as an example of this approach, so chosen because it is the most widely used of this class of approaches for process modeling, we can note the following strengths and weaknesses. In addition to the functional information noted above, IDEF0 diagrams can be broadened to indicate organizational information. In general, however, IDEF0 diagrams provide limited information; they are often imprecise about details and vague about the details of concurrency, resource conflict, timing, and state-oriented behavior.

As a result, models must be extended to allow simulation or enactment. Nevertheless, understandability of the model can make this a favorite method from a reader's point of view when compared to models that are more complete but more complex. This method is particularly useful in identifying missing process steps. For a process-immature organization, this is often the biggest improvement opportunity. It is the authors' opinion that the learning curve for readers of IDEF0 models is not steep and that training requirements for readers are minimal.

From the modeler's perspective, good tool support for these methods is available. In addition, they have been successfully used and are in widespread use.[23] Because of this wide use, method and tool obsolescence will not be as much of a concern as with other less widely used methods. Training is readily available. Standard notation, hierarchical structuring, effective rules for composition/decomposition, support for incremental development, and a reasonable learning curve are features of structured graphic approaches. Most models cannot be simulated or enacted due to the necessity of capturing more information than the notation supports; however, in a number of instances, extended notation or a bridge from structured graphic models to simulation models is available. All in all, the large community of users, the familiarity of data flow diagrams and mature tool support available for structured analysis make functional modeling methods strong contenders when choosing a process definition technology.

A more detailed discussion of IDEF0 modeling, training and tools can be found in Appendix E. More detailed tool information for other functional modeling methods is available in the tool lists in Appendix A.

1.4.1.2 Functional and Behavioral Modeling

Real-time structured analysis techniques add a state-oriented notation to data flow models, which allows behavior to be represented. Therefore, real-time structured analysis models, in addition to answering the questions that functional methods answer, also answer

---

[23] IDEF0 has seen extensive use worldwide. It has been used in such areas as hardware and software development, telecommunications, manufacturing, command and control, and real-time banking. IDEF0 has been used extensively in the commercial marketplace for business process redesign. It is also extensively used by the government market for business case analysis. The DoD Corporate Information Management Program (CIM) is using IDEF0. In addition to the DoD CIM program, IDEF0 is a functional modeling standard for the U.S. Air Force manufacturing programs (MANTECH), the CALS Concurrent Engineering Initiative, the DoD Industrial Modernization Incentive Program (IMIP), and the Advanced Manufacturing Program (CIM-OSA).

questions focusing on behavioral issues (e.g., What conditions cause objects to transition from one state to another?). In general, such techniques do not have a mechanism to indicate organizational perspective in their models. However, some structured analysis techniques do support extensions that allow the representation of "where" a function is performed.

Tool support is mature in our opinion. Points made for functional modeling in terms of answering the modeler's questions still hold for real-time structured analysis. In addition, the information captured in the real-time structured analysis notation provides the basis for simulation and enactment of the model.

However, despite its added functionality, real-time structured analysis has not been widely used for process modeling.

1.4.1.3 Functional, Behavioral, and Structural Modeling

A third structured graphical approach to software process definition offers a set of three distinct but interrelated viewpoints that can be used to define a software process: the functional view (often represented by data flow diagrams), the behavioral view (often represented by state transition notation), and the structural/organizational view (showing which elements of the process are performed by different entities). Such an approach has the necessary rigor and completeness to allow a software process to be well represented from distinct viewpoints.

These views can be represented by multiple tools. A single tool implementing all three views is STATEMATE[24], a tool for real-time systems specification and design based on a graphical language (STATECHARTS) and method developed by David Harel. In STATEMATE, the functional view is represented by enhanced data flow diagrams, the behavioral view is represented by an improved variety of state transition diagrams, and the organizational view is represented by block diagrams. A STATEMATE model can be analyzed for logical consistency and logical completeness. Structured model building approaches that allow details to be added to the model incrementally are also well supported. However, in the authors' opinion, the modeling technique has a steep learning curve and the

---

24 STATEMATE is a trademark of i-Logix, Inc., Burlington, MA.

resulting process definition is sometimes difficult to understand. STATEMATE has been used for process modeling by the process modeling research community. Though STATEMATE is being used by some organizations for process modeling[25], it is the authors' observation that STATEMATE has not been accepted or widely adopted for process definition by the software development community, despite the power of its approach.

1.4.1.4 Information and Data Modeling

IDEF1 and IDEF1X both provide a structured graphical means with which to design, analyze, and communicate the information and data portions of a system. IDEF1, developed in 1981, allows the modeler to concentrate on information collected, stored, and managed by real world objects; a high-level system information entity can be modeled as a composite of logical smaller groupings of information in its subsystems. It is often used to specify what information is currently managed, and what information will be managed in the "TO-BE" model. IDEF1X, introduced in 1985, initially focused on providing modeling support for the design of database systems, and therefore is not suited well for analysis of the "AS-IS" method. Nevertheless, IDEF1X has proved to be more expressive than IDEF1 even for non-database applications. Both IDEF1 and IDEF1X have been used widely, have mature tool support, and have been extremely useful in their intended application areas. However, IDEF1 is beginning to be used by fewer and fewer modelers. IDEF1X has more support from vendors. Consequently, IDEF1X is emerging as the IDEF modeling method of choice for informational modeling.

If a single method for process modeling is desired, neither IDEF1 nor IDEF1X is an appropriate method for this purpose. However, if other process modeling methods do not give an informational perspective and this is needed, a supplemental model using IDEF1X can be helpful to more precisely define the information discovered in an IDEF0 (or other process) model. Even this use of the IDEF1X model can be problematical, however. Since it is most useful for logical database design following a decision to implement using a relational database, the modeler sometimes has to violate some rules if he wants to use the model for process information structure and to support analysis from the user perspective. In addition, the lack of tool support for the inter-relationship of process modeling views can be a problem.

---

25 Rockwell, Naval Air Warfare Center (NAWC), and IBM Canada are three organizations currently using STATEMATE for process modeling.

1.4.2 Process Programming Approaches

Process programming languages, in which a software process is represented in the form of a program, using programming-like languages, notations, and formalisms, have been developed specifically for the description of software processes. In general, the goal of the process language approach is to support computer enactment. Some languages are essentially procedural in approach; others are rule-based. In procedural approaches, a process is broken down into a series of steps. Procedural process languages are well suited for describing control structure, hierarchy, interfaces, and for specifying the "steps" that, taken collectively, constitute a software process. In rule-based approaches, process steps are described by rules with pre- and post-conditions. Rule-based approaches are well suited for specifying what conditions must be satisfied before a process step can take place, as well as those conditions that must be met before that process step completes. Many process programming languages are being used, many in a research mode. Since no standard process programming language exists, tool support for process approaches is limited, In general, process programming is neither widely used nor considered a mature technology by software process technologists.

Process programs are likely to be inherently complex. Since the goal of the process language approach is to support computer enactment, a process language is not primarily designed for human communication. This approach is not intended for, nor suited to, ease of communication. Therefore, lack of understandability of the resulting model will often prove to be a major drawback from the model reader's perspective. Though process programming approaches are sufficiently flexible to model software products and processes at any desired level of detail, process programming provides less support than other modeling techniques for process improvement. In addition, since process programming often takes an essentially activity-based or functional view in which the process description must be followed exactly step-by-step, some fear that this may constrain the free/opportunistic decision making that is a part of any realistic software development process.

From the modeler's perspective, working with a process programming language will require familiarity with its notation and semantics. The learning curve for a process language should be roughly equivalent to the learning curve of a new programming language. As with contemporary high-level programming languages, it should be possible to follow a structured approach using a procedural process language. With essentially procedural-type languages, it should be possible to incrementally build a model in the same way that a

software application can be built incrementally. In addition, it should be possible to exploit reuse when using a process language, as well as make possible the simulation and enactment of the model. However, no standard process programming language has emerged from research efforts. With rule-based languages, a further drawback has proved to be the lack of function decomposition and overall structure.

Both procedural and rule-based languages have been chosen as the definition method of choice by some experimental process-driven Software Engineering Environments (SEEs). Procedural languages such as APPL/A in the Arcadia project, CML in the TRIAD project at Ohio State University, and Gist in the System Factory Project at the University of Southern California are representative examples. Rule-based languages are used in the MARVEL and MELMAC environments.

1.4.3 System Dynamics Approaches

Systems dynamics, developed at the MIT Sloan School of Management by Jay Forrester, applies the principles and techniques of feedback control systems to managerial, organizational, and socioeconomic systems. System dynamics models describe systems of variables and delays. Some variable values are derived from a calculation, other variable values accumulate over time. System dynamics modeling methods are widely used for systems definition in such areas as economics, ecology, avionics, and navigation. It has been found that the feedback principles of system dynamics help structure and clarify the complex web of dynamically interacting activities. Applied to the software process, this approach allows great fidelity in modeling processes, making possible both more complicated models and models of more complex systems. Modeling process steps with an emphasis on feedback in the system allows the modeler to accurately model the rework realities in the software process. The technology is also particularly strong in using feedback to accurately model dynamic behavior and interactions between activities. A primary benefit when using systems dynamics models is increased understanding of the dynamic relationships within a system. The ability to use system dynamics models for "what if" scenarios is a strength of the technology. However, no formal analysis techniques exist for system dynamics models, and complexity and understandability of very large models can be a drawback of this approach.

Tool support, both graphical and language-driven, for system dynamics modeling is quite mature in the authors' opinion. There are a number of notations used with no formal

drive for standardization. Commonly used notations are only standard in the sense that most system dynamics modelers use tools developed by a small group of vendors[26] and hence adopt their notation. Systems dynamics modeling techniques can be learned relatively quickly but this implies an understanding of the principles and techniques of systems dynamics that is not necessarily straightforward. It is also true that ease of modeling does not always translate into ease of understanding the resulting model. Models can be developed incrementally. However, there is a lack of functional decomposition. The absence of effective structuring and decomposition techniques causes the models to spread rather than decompose. There is no direct expression of state behavior.

Simulation of the model provides a vehicle for controlled experimentation in the area of software development. Since system dynamics models are good at describing the properties of systems, a system dynamics model can show the effects of various policy decisions in a way that no other model can. The approach can help in understanding the dynamics of a system but, given the reservations above, it may not provide the optimal way to describe a system. The technique of using two process models, a system dynamics model and a functional process model, can be very helpful, particularly in studying the effect of policy decisions on the process. No efforts have been made to use system dynamics models for process enactment nor is such an approach likely to be successful for enactment.

1.4.4 Petri Net Approaches

Petri net modeling techniques use a mathematically-based graphical notation (Petri nets) for modeling dynamic and distributed software process activities. Adapted to the requirements of software process modeling, these techniques apply formality and rigor to the task of process definition. Petri net analysis techniques can also be used to validate the model and to verify software process model properties such as reachability. The formalism provided by a Petri net model can contribute to consistent and precise understanding of the software process, enable automated support and open up ways to automate well-understood parts of the software process. Though the majority of questions that a user of a model might want to ask of a process model could be answered using a Petri net model, a great deal of

---

26 STELLA (or its companion tool, IThink) are the most commonly used graphical-based modeling tools for system dynamics. Tools are marketed by High Performance Systems, Hanover, NH.

DYNAMO is the most commonly used language-based modeling tool for system dynamics. It is marketed by Pugh Roberts, Associates, Cambridge, MA.

training is necessary to bring the user to the level of understanding necessary to interpret the model. Therefore, lack of understandability is a concern with Petri net technology.

From the modeler's perspective, Petri net modeling and analysis is well supported by mature tools, such as Design/CPN,[27] which provide automated simulation of the model. Other tools, such as Process Weaver, use Petri nets in their implementation while providing a more accessible interface to the modeler. When the modeler must use Petri nets directly to construct the model, the learning curve necessary to become an accomplished Petri net modeler is steep. Models of even simple systems tend to become very complex very quickly. Therefore, modeling complex systems using Petri nets can be difficult. Details can be added in an incremental manner to Petri net models; however, the use of structuring techniques/modularization and reusable net components has only recently been introduced by researchers. In addition, a number of versions of Petri net notation are being used. Nevertheless, given the power of Petri net technology for fully defining processes, Petri net models are beginning to be used in the United States and are already used widely for software process modeling and computer enactment in Europe.

Recently, modelers working with process definition methods that do not allow the completeness in process definition that more formal techniques (such as Petri nets) offer have begun to use bridges to the more formal technologies. These bridges permit the modeling of the software process using a notation and technology that is accessible and understandable, thereby facilitating communication with model readers. This model can be annotated to reflect aspects of the process that cannot be modeled with the more limited notation. Then, when the modeler is satisfied that the model accurately reflects the software process, he/she can transfer the model to the more formal technology, simultaneously adding the details noted in annotations. As an example, Meta Software provides a translator that allows IDEF0 models generated using their Design/IDEF0 tool to be translated to a Petri net framework. Adding the necessary detail (detail that cannot be formally provided in IDEF0 models except in the form of comments) will allow the IDEF0 model to be simulated using Petri net technology.

---

27 Design/CPN is a trademark of Meta Software Corporation, Cambridge, MA.

1.5 Process Enactment[28]

Once a software process is formally defined, computer enactment provides automated support for the process definition. For example, the definition can be used in a computer aided support environment to invoke software tools at appropriate times, enforce the process model, give guidance to software developers, record project metrics, keep management informed of the status of a project, and execute automatable activities in the process model. Certainly, this could produce significant corporate benefits. For technology insertion, computer enactment could facilitate the introduction of new technology by supporting the training, piloting and evolving of new processes. In the area of standards enforcement, computer enactment could enforce both process standards (inspection procedures, configuration management, etc.) and product standards (complexity, structure, etc.). Metrics could be automatically collected for both process (repeat cycles, bottlenecks, time spent in each process phase, etc.) and product (errors, product statistics, etc.). Project management information (schedule updates, resource projections, etc.) could be automatically generated. At its best, task automation provided by computer enactment will free software developers to concentrate on the creative aspects of developing systems.[29]

Computer enactment in a process-driven environment could operate in the following way: In the morning the software engineer logs into the environment. A screen shows the tasks that are available for the day. The engineer chooses a task and the task context is brought up - the status of the task, any changes that have occurred since the last work session, any messages from other members of the team. After choosing a part of the task to work on, tools are automatically invoked and the correct documents/work made available. Guidance on the process to be used is provided. If the engineer attempts to depart from the process or project standards, this will not be allowed by the environment. Metrics (exactly which metrics are determined by the process engineer who has defined and enacted the process) are gathered automatically on the project by the environment as the project progresses. Project management information is generated automatically. Any automatable parts of the process are executed automatically by the environment. Such an environment

---

28 Section 1.5 will discuss "computer enactment " - as distinguished from "human enactment."

29 Myles, D.T., "Automated Software Process Enactment," **Proceedings from the Fifth Annual Software Technology Conference: Software - the Force Multiplier**, April 1993.

would seamlessly link a software development team. The team as a whole would have a better view of the big picture and team member roles; tasks would be prioritized to improve productivity and eliminate bottlenecks. A new member of the team would learn the project and process much more quickly with the guidance available from the environment. At any time, the manager would know project status, and could use the environment capabilities to make projections when manpower and schedule changes have to be made.

Such a technology description sounds very much like a description of yet another "silver bullet" that will solve all our software development problems. It is not! When an organization wishes to enforce a process, the organization needs to have a well-defined process and to thoroughly understand that process. The organization must be committed to process driven development - not to product driven development where, at the first stress, the established processes are thrown out and a "get it done in whatever way" mentality takes over. This is the mentality found in Level 1 organizations that comprise the vast majority of software development organizations. As mentioned in Section 1.1, the level of definition detail that is necessary for either human enactment or computer enactment is far greater than exists in most process definitions. And, even if a process has been defined to a precise enough level of detail for human enactment, much more detail is often needed for a human-enactable process to become machine enactable. The cleanroom process[30] has been under development for about 20 years, its process description is around 400 pages in length, and training is extensive; the process itself can be said to be human enactable. It is thought that "writing" this to the level of detail necessary to make the process computer enactable may take three times as much detail![31]

In addition, it has not been shown that the advantages of such an approach would outweigh the disadvantages. Unless the enacted process is built very carefully, computer enactment may not effectively support training, or the piloting and evolution of software process; it could actually be a barrier to the evolution of software process. Although computer enactment may make standards readily available to encourage their use, computer enactment may not be able to really enforce standardization. Early versions of these environments have been rejected by software engineers due to their rigidity. It hasn't been

---

30 See Appendix A.1: Process Asset List for a more complete description.

31 Conversation with Paul Arnold, STARS Symposium. IBM is working on the computer enactment of selected portions of the cleanroom process.

proven that this approach really adds value. No-one has proven the cost effectiveness of computer enactment, except for very small pieces of a process - configuration management, doing regression testing, doing a build in exactly the same way. It may be that an effective use of computer enactment will be to automate repetitive tasks, dogwork, error prone tasks - things that people tend to do poorly.

Computer enactment is largely at the research stage. Though a number of efforts in the United States and abroad are making progress toward realizing process-driven environment goals, support for a true process-driven environment is not a reality at this time. Process-driven environments that do exist tend to incorporate vendor-espoused methodologies. More flexible architectures for defining (or at least tailoring) the software process are needed.

Three major approaches to process enactment (using frameworks, customizable environments, or process definitions as a basis for enactment) will be discussed in Appendix F, supplemented with examples of products supporting each approach.

1.5.1 Enactment Lessons Learned

The authors have had conversations with several organizations attempting some level of process enactment[32]. These efforts have ranged from an attempt to build process definition into a configuration management system[33] to an experiment to fully enact a portion of a software process[34]. A number of lessons have been learned: First, there is a lack of robustness noted with the technology – fixes are often necessary; deficiencies are common. This is not surprising due to the relative newness of computer enactment technologies. Secondly, turnkey enactment environments that exist, particularly in the information systems area, will probably not be acceptable to organizations, at least outside the information systems realm. In general, it is necessary to have the ability to tailor/change process descriptions to fit company practices. Thirdly, the level of commitment to the technology

---

[32] Out of respect for their corporate privacy, company names providing lessons learned have been withheld.

[33] Using a product such as Caseware/CM.

[34] A good description of the experiment to fully automate a portion of the software process can be found in Myles, D.T., "Automated Software Process Enactment," Proceedings from The Fifth Annual Software Technology Conference: Software - the Force Multiplier, April 21, 1993.

must be very high. Certainly, computer enactment can be accomplished, even given the maturity of enactment technology at this time; however, organizations will probably underestimate the amount of effort needed to install enactment technology, both in sufficiently defining the process to the necessary level of detail and in implementing it with enactment tools. The learning curve for enactment tools is much higher than with other tools, largely because traditional tools do not enforce a process. A lot of engineering/definition is needed to define and constrain the process and to take into account all the exceptional cases as the enactment environment will often not provide much flexibility. This is why it is important to pilot small portions of the process before attempting to enact an organization's entire software process. And, last but not least, introducing computer enactment to an organization requires considerable effort to sell the idea to users. In general, programmers tend to mistrust and resist computer enactment.

2 METHODS/TOOLS/TRAINING SUPPORTING PROCESS TECHNOLOGIES

In order to support the search for appropriate process technology methods, tools, and training, comprehensive information lists are included in this report. Lists of vendors and researchers working in the areas of process modeling and computer enactment are provided. In addition, a complete list of process assets in the Process Asset Library (PAL) is provided.[35]

For most tools in the lists, more detailed product sheets are also included as well as a number of product critiques from users. Since a number of STSC customers are beginning to use IDEF0 modeling for process definition, a list of training offerings is provided for this technology.

Lists were developed from four types of sources: personal experience, process technology literature searches, tool/vendor materials, and attendance at relevant conferences.

2.1 Process Technology – Method/Tool/Asset Lists

An important and necessary step in the technology selection process is to identify candidate tools and methods. In order to assist in this, Appendix A provides comprehensive lists of process technology methods and tools. Surveys of methods and tools in the areas of modeling and enactment have uncovered much information about researchers and vendors working in these areas. Appendix A contains lists providing a complete overview of the field and summarizing contact/product information. Because the SEI assessment method is primarily the only such method relevant to both DoD organizations and software

---

[35] The Process Asset Library (PAL) is a reuse library for software processes, containing examples of experience-tested processes. As members of the software engineering community begin to define their software processes formally, there will be significant opportunities for reuse. The PAL will provide these important resources to the engineering community. The PAL is a joint STARS/SEI product. STARS funded the development and made it available on ASSET, a facility supplying computer access to software reuse libraries, catalogs, and information via wide area networks and telecommunications; SEI provided oversight. At the time of this report's publication, the PAL was only available to the STARS community, but efforts were being made to make it obtainable to all. To get an account on ASSET, call (304)594-1762. Those not affiliated with STARS will not have full privileges online.

development specifically, we have not provided a list of assessment technologies Readers are referred to the Software Engineering Institute for more assessment information.[36]

The lists provided are: Process Asset List (A.1), Process Modeling List (A.2), Process Frameworks List (A.3), Computer Enactment Technology List (A.4), and Process-Driven Environments List (A.5). The Process Asset List encompasses information about process assets contained in the STARS/SEI Process Asset Library V2.0. The Process Modeling List contains information about tools and languages supporting process definition, modeling, and simulation. The Computer Enactment Technology List contains information about technologies and tools supporting computer enactment. The Process Frameworks List contains information about frameworks[37] that have been used to support the creation of process-driven environments. The Process-Driven Environments List contains information about existing environments supporting process-driven development.

2.2 Process Technology – Product Information Sheets

Volume II of this report contains the technology product sheets for most of the process technologies and tools in the technology/tool lists. These sheets provide detailed information on process technologies and tools. Information on pricing, contacts, support, process technology areas covered, intended users of the technology or tool, intended application area, primary methodology base, hardware platforms, and general tool capabilities is included. Users of these reports can make preliminary tool assessments based on the provided information. The information in the reports was obtained either directly from the vendor or from the vendor's literature. In most cases, the vendor has supplied the information. There are tools in the tool lists for which there is no associated technology product sheet. This condition occurs because there was insufficient available information to create the technology product sheet, either because the vendor did not supply information in time for publication or because the tool was added to the tool list too late for the creation of a technology product sheet.

---

[36] For information, contact Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890,; telephone: (412) 268-5800; internet: customer-relations@sei.cmu.edu.

[37] A framework provides the architectural basis of an environment and provides a set of services as a basis for environment construction.

The STSC can be contacted for both unpublished and updated reports that may be available. Please contact the STSC for information on how to obtain Volume II of this report. STSC contact information is located in the beginning of this document.

2.3 Process Technology – Product Critiques

The STSC solicited product critiques from experienced tool users. These are included in Volume II of this report, and highlight the experiences (both positive and negative) of actual tool users.

We would like to expand the number of critiques and the technology areas included in this section in subsequent reports. If you are a user of a tool that is or should be included in the tool list and would like to write a critique, please contact the STSC. A Product critique form is provided at the end of Appendix C.

The STSC can be contacted for both unpublished and updated critiques that may be available. Please contact the STSC for information on how to obtain Volume II of this report. STSC contact information is located in the beginning of this document.

2.4 Process Technology – IDEF Training

An IDEF0 course training matrix is provided in Appendix E with a representative list of IDEF0 courses. For each set of courses, use of software process examples, use of tools in the course, and the willingness to customize the course is indicated. More complete information on each training course is provided in IDEF Training Information Forms, which can be found in Volume II of this report. While we have attempted to provide a comprehensive list of training options, we continue to update this list. Please contact the STSC to provide information about course offerings not on our training list. Also, since training offerings change frequently, the companies in the list should be contacted to receive the latest information on courses offered.

3 SELECTION AND USE OF PROCESS TECHNOLOGIES

3.1 When to Use Process Technologies

The overriding reason to use process technologies is as a means to improve the software development and maintenance process. Though US software statistics collected by the SEI are confidential, findings support the following claims: In terms of software quality (measured in terms of software defects shipped), 100 to 1 improvement is possible looking at proven examples from large systems. In terms of software productivity 10 to 1 improvement is possible and 100 to 1 improvement is projected.[38]

In addition, specific cases of measured return on investment (ROI) can be cited. The Swedish Navy has measured an initial productivity improvement on their ship system FS 200 of 118% which translates into a 55.1% reduction in cost of the actual work being measured, a 55.1% reduction in the critical path schedule, and a 50% reduction in testing time.[39] In this country, the Hughes Ground Systems Group, Software Engineering Division, has seen a reduction in cost overrun targets of 50% ($1 million) and a return on investment of 5 to 1.[40] At Raytheon Equipment Division, Software Systems Laboratory, annual investment in process improvement is $1 million with a resultant savings in cost of quality of $5.8 million/year, a productivity gain of 29% ($11.2 million/year) and an improvement ROI of between 5.8 to 1 and 11.2 to 1.[41] At the Oklahoma City Air Logistics Center, Tinker AFB, a direct labor savings of $2.935 million and improvement ROI of 6.4 to 1 was measured.[42]

In all of these cases, attention to software process and the use of process technologies was a key factor in the improvements measured. However, the SEI is also

---

38 McKeehan, David, "Planning a Software Process Improvement Program," Tutorial: The Fifth Annual Software Technology Conference, Salt Lake City, Utah, 19 April 1993.

39 Ibid.

40 IEEE Software, July 1990.

41 Ray Dion briefing, 1992 SEI Symposium.

42 OC-ALC/LAS white paper.

seeing many groups that fail. It is important to consider when it is appropriate to apply a given technology. Often, using a process technology before the use of that technology is appropriate can be counter-productive. Therefore, we will consider in this section the appropriate timetable for process technology insertion for each technology area.

Process assessment technologies have been defined as those technologies that enable an organization to characterize the maturity of its process. Before any other process technology is considered, an organization should take advantage of this technology. For assessment with management commitment, the benefit is very high. An assessment should ideally identify the most critical process issues and facilitate the initiation of process improvement actions. Without this assessment, it will be difficult to use the other process technologies to their full advantage. If assessment has not already taken place, this should be a first priority. DoD organizations should contact the SEI for further information before initiating the assessment process.[43]

Once the assessment has been completed, an organization will have identified an action plan that may require process definition. Process definition technologies can then be used to support the formal definition of an organization's software development process. One effective use of process definition at this stage is to model the "as-is" process and then use that definition to develop a model of the "to-be" process.

For some added benefit, simulation and analysis of the model can be performed. After a proposed process has been defined, simulation and analysis will allow an organization to see any bottlenecks or unworkable flows in the system. For example, simulating a process may show clearly how several activities are repeatedly delayed until one activity completes. Once this bottleneck in the process is recognized, then steps can be taken to improve that aspect of the process. Analysis may show that certain activities do not get the resources that they need. In these ways, simulation and analysis technologies can undeniably provide extra insight into problems with an existing or proposed process. However, it is important to note that much benefit and the biggest ROI will be achieved by a

---

[43] For information, contact Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890,; telephone: (412) 268-5800; internet: customer-relations@sei.cmu.edu.

thorough definition of the process; much can be learned, including information about flaws in a system, from a process model without simulating it.

Process enactment technologies have been defined as those technologies that will support the execution of a software process definition. By this we mean any technologies that will allow the process definition to be "installed" in the sense that the enacted definition will manage and/or control the process, and capture measurements as the development process proceeds. Only after an organization has thoroughly worked the process definition aspect until the process is well defined, well understood, and consistently used by the organization, should that organization consider the steps that need to be taken to enact or automate it. Some feel that process enactment is only sensible for level 5 organizations. For most organizations, this will be a long-term goal.

3.2 How to Select a Process Technology

In order to evaluate methods, tools, and software engineering environments, it is necessary, first of all, to define goals and requirements. A method is then chosen that supports the user's goals and requirements. Lastly, a tool/environment is selected that supports the chosen method. In all of this, it is important to distinguish between a method and tools/environments that support method(s). A method provides a step-by-step approach; a tool provides automated assistance for a method; and an environment provides an integrated set of tools in support of method(s).

Within each process technology area, for each technology being evaluated, we must ask first: is it a method or a tool/environment? If it is a method, depending on the process technology area the method addresses, a particular set of functional and quality characteristics must be evaluated as well as the level of tool support available for the method. If it is a tool, a different set of functional, quality, and support characteristics must be addressed. When it is an environment, yet another set needs to be addressed.

In order to facilitate the evaluation of process technologies, checklists are provided in Appendix D: Process Technology Taxonomy, for assessment methods, modeling methods, modeling tools, and Software Engineering Environments (SEEs) designed for software process model enactment. These checklists can be used to characterize the features,

strengths, and weaknesses of a technology and hence assist the user in choosing methods or tools with the appropriate attributes for a given project. Use the checklists to determine the essential properties of a tool or method and then use the resulting short list of "must-have" features when evaluating a process technology.

An effective decision process could follow the following steps:

(1) Identify the process technology area on which to concentrate.

(2) Define goals and requirements.

(3) Use the taxonomy (Appendix D) to identify essential method/tool attributes.

(4) Consult the appropriate candidate tool list (Appendix A) for a comprehensive list of researchers and vendors working in the chosen process area.

(5) Use the essential attributes, product sheets and product critiques (Volume II of this report) to shorten candidate tool list.

(6) Use recommended reading list (Appendix F) to learn more about methods/tools on shortened list. Interview tool users.

(7) Test method or tool in-house.

(8) Make decision.

3.3 How to Use Process Technology

The overall goal of using process methods and tools should be to improve the software development process. To start the improvement cycle, it is necessary to establish an organizational baseline against which to measure improvements. It is often helpful to develop a quick picture of the organization's current software processes for use later on by the assessment team and by the working groups. It is also necessary to establish an infrastructure for continuous process improvement. A Software Engineering Process Group (SEPG)[44] working closely with a management steering council can be an effective way to initiate and sustain this continuous improvement. The SEPG organizational element has been used throughout the industry with some success; when the SEPG works closely with a

---

[44] The "Software Engineering Process Group Guide" (CMU/SEI-90-TR-24) published by the SEI provides a good introduction to the operation of an SEPG. It discusses the rationale behind process groups, specifies activities that are performed in establishing a process group, identifies activities that an established process group would perform on an ongoing basis, and addresses organizational issues.

management council, continuous improvement has been effectively initiated and sustained. It is necessary to be committed to this effort as the top priority of the software organization; efforts have failed when they were put on a back burner to deal with the crisis of the day.

3.3.1 Assessment

After an infrastructure for continuous process improvement (such as the SEPG) is in place, an assessment will help an organization to take the next steps. Though a number of the guidelines in this section on how to use an assessment effectively come from SEI sources, such as the recent STSC conference tutorial, guidelines could be applied equally well to both SEI assessments and Model Based Process Assessments (MBPAs). In either case, in order to effect change, an organization needs to first understand its existing process. The objective of an assessment is to understand current software engineering practices, identify key areas for process improvement, and facilitate the initiation of process improvement actions to provide a framework for action. In addition, an assessment team can help obtain sponsorship and support for action and help establish support at all levels for improvement efforts.

During an SEI process assessment, a trained team of experienced software professionals appraises an organization's existing software process, based on a review of four or five key projects, responses to an assessment questionnaire, and in-depth discussions with project leaders and practitioners. Strict confidentiality will always be observed. SEI's appraisal identifies where an organization should focus its efforts by comparing organizational process capability to a standard reference model (the CMM) and local expert opinion. It yields a measure of organizational process maturity and findings based on current and desired states of organizational maturity. The collective assessment team knowledge and experience allows the assessors to interpret the data gathered and prepare a report of findings, an assessed maturity level, and recommendations for improvement. The team will provide improvement action planning, which will include both long-term recommendations and short-term quick fixes to capitalize on momentum created by the assessment process.

In an MBPA, the team conducts interviews and constructs a model of the process, analyzes the process and plans improvements, implements improvements, and then measures, compares results against the goal, and continues to improve the process. MBPAs identify where an organization should focus its efforts through analysis by local experts.

After an assessment, the organization either should be prepared to take action or they should not undergo the assessment. If expectations are raised and then action is delayed, the question will be, "What happened to process improvement?" and morale will be negatively affected. It is often important to start the improvement phase with one small effort and demonstrate success early. Keep improvement efforts visible by using newsletters, seminars, etc. Use pilot projects to try new practices. Evaluate success and reformulate practices before institutionalization.

The initial project on which the technology is used must be selected with care. Implementing a change may cause a negative effect on productivity, particularly when initial training is taken into account. Because of this, time for introducing the new technology must be budgeted in the pilot project schedule. Since proper transition to the new technology is important, the issues of training and technology transition are of particular concern. Management must budget and plan for the necessary training, or the effort will fail. To get people through this time of transition, it is necessary to be specific about what is not going to change and provide sufficient support for the changes that will take place.

3.3.2 Modeling

As standard practices begin to be defined and adopted by an organization, it is appropriate to adopt a standard approach for defining/modeling those processes. A number of the process modeling approaches are good candidates for this. It is important to realize that an organization should start to define pieces of the overall software development process as soon as process improvement efforts begin to take place, the pieces that will be defined being the areas of the overall process on which the improvement effort will concentrate. Even though Level 3 emphasizes process definition, that does not indicate that all definition/modeling should therefore be restricted to this level and should not be attempted until an organization is at Level 3. Rather, at Level 3, all components defined at earlier levels are defined/modeled in an integrated fashion.

Figure 3-1 (adapted from teaching materials developed at the SEI with the sponsorship of the DoD) shows how modeling can be used to support the improvement process. Modeling technologies can be used to define the "as-is" process being used in an organization; the process is then analyzed, monitored and modified to incorporate improvements. The proposed "to-be" process can then be modeled; this model can help

people understand the proposed process, and evaluate it for consistency and effectiveness. Iterations of the model will take place until reaching consensus on an appropriate approach. The model can then be used for implementation of the proposed process on pilot projects. This process then becomes the "as-is" process which will be analyzed, monitored and modified as necessary. Process improvement should be continuous as indicated in Figure 3-1 and modeling will help this continuous improvement.

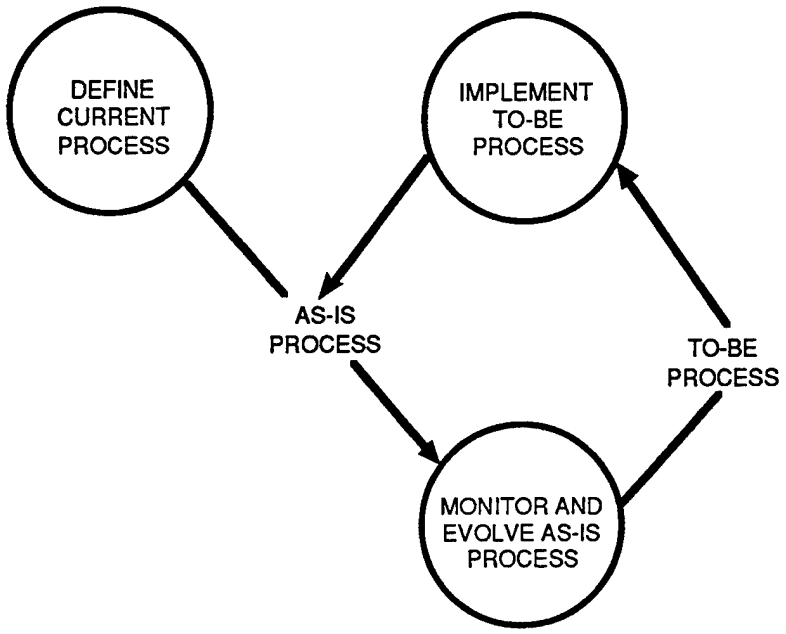

Figure 3-1. Using Process Modeling to Support Process Improvement

3.3.3 Enactment

While process assessment and process modeling can allow an organization to reap important benefits and assist in process improvement, these technologies appear to lead quite naturally to process enactment technology as the next logical step in the continuum of process innovations. However, before an organization attempts to enact its process, commitment to formal process – i.e., structured methods, formal software inspections, formalized configuration control automation, formalized requirements analysis, prototyping, training – should all be in place. Enactment is not a first step! Use of enactment technology should be in a pilot program context, enacting one small portion of a key process area. It is the opinion of the authors that the technology is too new for production programs.

4 FUTURE DIRECTIONS

Process technologies have proven to have real benefits. Process assessment helps organizations to take the first steps in improving their process. Defining processes, either existing processes or proposed processes, helps organizations to uncover process flaws and design effective and improved processes. However, software process is a new technology area and, in the opinion of the authors, some process technologies are far from mature. While the SEI assessment methods are well on their way to being an assessment standard, both in the DoD and in the commercial world, standard techniques in the areas of modeling and computer enactment have not been adopted. Over time, de facto standards in the areas of modeling and computer enactment will probably emerge as these areas become more widespread. Initially, a variety of methods and tools will be used; however, effective methods and tools will be retained and less effective methods and tools will be discarded.

In the process modeling area, methods and tools aimed specifically at process modeling will emerge. Today, most of the modeling methods and tools being advertised as suitable for process modeling were not designed specifically for that purpose; many were used for software design, task decomposition, management/enterprise models. Some of these general-purpose modeling methods/tools will be useful in the process modeling area; some will not. When surveying modeling tools, the authors found that a number of vendors who advertise "process modeling" in their promotional literature will admit that their tools have never actually been used for that purpose! Most modeling tools being actively used for process modeling today support some graphical representation of the process being modeled. It seems fair to predict that graphical methods for process modeling will be well-represented in future process definition methods and tools. However, it is difficult to predict the future direction for process languages. The fact that no standard language seems to be emerging from the research efforts in this area, coupled with the appeal of a graphical approach for communicating the contents of a model, combine to put widespread use of process languages for modeling in doubt. Nevertheless, process languages may still have a role in some environments supporting process enactment.

When predicting future directions for process technologies, no area is as murky as process enactment. First, the technology has seen little real use so far. Second, there is some confusion about just what process enactment really means. Third, the problem of enactment has been approached from quite disparate directions, as described in Section 1.5. Computer

enactment technology is surely a promising technology. However, much of the work being performed in this area is in a research mode and few commercial products are available. The actual number of organizations using computer enactment operationally is quite small. In the short term, the number of players in this area of process technology will probably increase. A number of research efforts will begin to commercialize their work, some with the support of ARPA. Over time, there will probably be a decrease in major players, as successful computer enactment products will become adopted and less successful approaches fail. For the present, those who want to experiment with enactment should investigate computer enactment approaches on a very small scale with well-defined parts of an overall process.

5 SUMMARY

This report summarizes the STSC's work to date in the area of process technologies. A comprehensive compilation of process technologies has been gathered in the areas of process assessment, modeling, and enactment. Major approaches in each area have been discussed. A taxonomy of process method/tool characteristics has been developed to aid in the choice of an appropriate process technology. Vendors and researchers in the area of process technology have provided information on their methods and tools. Method and product critiques have been supplied by users. An annotated bibliography provides sources for more in-depth knowledge. Detailed information has been provided on IDEF methods, tools, and training.

This is the first report that the STSC has issued in the process technology area. We will maintain and evolve the report to reflect both increased understanding of the technology area and changes in the information about methods and tools. The STSC will continue to update the Method/Tool lists. Additional tool reports and user critiques will be solicited and published. Emphasis will be placed on methodologies and technologies that have been successfully used on specific applications. Finally, Software Development and Support Activities (SDSAs) will be supported in their process technology selection and insertion efforts when they so request. This report will be republished next year.

Appendix A - Process Technology Tool Lists

This appendix contains five separate lists, each of which concentrates on a particular process technology area. They are:

- The Process Asset List
- The Process Modeling List
- The Process Frameworks List
- The Computer Enactment Technology List
- The Process-Driven Environments List.

The Process Asset List contains information about process assets contained in the SEI Process Asset Library V2.0. The Process Modeling List contains information about tools and languages supporting process definition, modeling, and simulation. The Computer Enactment Technology List contains information about technologies and tools supporting computer enactment. The Process Frameworks List contains information about frameworks[45] that have been used to support the creation of process-driven environments. The Process-Driven Environments List contains information about existing environments supporting process-driven development. Note that because of the widespread acceptance and use of the SEI assessment method and the lack of alternative, mature assessment methods, we have not provided a list of assessment technologies.

Lists were developed from four types of sources: personal experience, process technology literature searches, tool/vendor materials, and attendance at relevant conferences. In each list provided, the tools are listed alphabetically by method or tool name. The information includes the tool or technology name, the developer (in the case of pertinent technologies) or vendor (for commercial tools), as well as the developer or vendor's address and phone number. Since a number of technologies span more than one process area or provide a variety of process functions, a technology category column is provided on most lists, as well as a column which allows a brief comment about the tool/technology. In addition, a number of tools, particularly modeling tools, are focused on supporting specific methods. The class of platform on which the tool runs is often of primary importance to the potential "buyer." Vendors and researchers often develop, market, and optimize their

---

[45] A framework provides the architectural basis of an environment and provides a set of services as a basis for environment construction.

methods and tools to target specific user application areas. The lists contain columns for method type, platforms, and target audience where appropriate.

The following abbreviations are applicable to all five lists. Note that, through the term modeling was used to encompass both definition and simulation in much of the report, the specific terms, definition and simulation, are used here. The reason for this is to enable us to identify those tools that explicitly support process simulation in addition to process definition.

Process Technology Category Abbreviations

A:	Technology and Tools Supporting Process Assessment.
D:	Tools and Languages Supporting Process Definition.
S:	Tools and Languages Supporting Process Simulation.
E:	Technology and Tools Supporting Computer Enactment.
SEE:	Software Engineering Environment (SEE)/Integrated Programming Support Environments (IPSEs) supporting process-driven development.
SEE/F:	SEE/IPSE Frameworks supporting the creation of process-driven environments. (A framework provides an interface for the building of SEEs.)

Targeted Application Area Abbreviations

MIS:	Management Information Systems.
TECH:	Technical.
CM:	Configuration Management.
OTHER:	Real-Time, Scientific Development, Transaction Processing, Embedded System Development, etc.; specific area targeted specified where space permits – otherwise information is available on individual Product Sheets.
ALL:	Tools/methods targeted to all market segments.

Platform Abbreviations

DT: Desktops, including Macintoshes and PCs.

WS: Workstations, including computers classed as mini-computers.

MF: Mainframes.

Many of the process technology tool lists use additional abbreviations relevant only to a single tool list. In that case, abbreviations applicable to a specific list are noted at the beginning of the relevant list.

Additional information on specific tools or methods can be obtained using the Product Information Sheets in Volume II of this report. For information on how to order Volume II, please contact the STSC customer service department at (801)777-7703 or DSN 458-7703, fax to (801)777-8069 or DSN 458-8069, or email to godfreys@wpo.hill.af.mil.

Due to the very dynamic nature of the process technology area, lists may contain inaccuracies and omissions. If you are aware of any, please contact the STSC using the information above.

A.1 Process Asset List

The Process Asset Library (PAL) is a reuse library for software processes, containing examples of experience-tested processes. The PAL is a joint STARS/SEI product. STARS funded the development and made it available on the Asset Source for Software Engineering Technology (ASSET), which supplies computer access to software reuse libraries, catalogs, and information via wide area networks and telecommunications. SEI provided oversight. This list contains information about process assets contained in the SEI Process Asset Library V2.0. Assets are presented in alphabetical order. The information includes the component or asset name, contact information, and a brief comment about the process asset.

At the time of this report's publication, the PAL was only available to the STARS community, but efforts were being made to make it obtainable to all. To get an account on ASSET, call (304)594-1762. Those not affiliated with STARS will only be able to read the abstracts online.

COMPONENT/ASSET	DEVELOPER/VENDOR	COMMENTS
Cleanroom Engineering Software Process	Paul G. Arnold IBM Federal Systems Co. 800 N.Frederick Ave., 182/3M34 Gaithersburg, MD 20879 (301)240-7464	Process guide with process model in a graphical box structure notation
Domain Specific Software Architecture Process Lifecycle	J.W. Armitage Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213-3890 (412)268-6589	Process guide, information mapped™, IDEF0
IEEE Standard 1074-1991, IEEE Standard for Developing Software Life Cycle Processes, Section 3 Project Management Processes	Neal Reizer Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213-3890 (412)268-2854	IDEF0 Model, with English text supplemental information
IEEE Standard 1074-1991, IEEE Standard for Developing Software Life Cycle Processes, Section 3 Software Quality Management and V+V Process	Marc I. Kellner Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213-3890 (412)268-7721	done using STATEMATE
Quality Function Deployment	Kenneth Y. Nieng AT&T Bell Laboratories Room 2A-223 263 Shuman Blvd. PO Box 3050 Naperville, IL 60566 (708)713-4746	English text process guide, embedded SADT model

Requirements Elicitation Process	Kenneth Y. Nieng AT&T Bell Laboratories Room 2A-223 263 Shuman Blvd. PO Box 3050 Naperville, IL 60566 (708)713-4746	English text process guide, embedded SADT model
Software Configuration Management Process	Warren Mosely Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213-3890 (412)268-5174	process guide, SPMS model
Synthesis	Mark D. Kasunic Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213-3890 (412)268-5039	process guide, IDEF0 model
The TRW Ada Process Model	Fred A. Maymir-Ducharme Paramax (STARS Program) 12010 Sunrise Valley Drive Reston, VA 22091 (703)620-7596	English text process guide, embedded SADT (using Design IDEF)

A.2 Process Modeling List

This list contains information about tools and languages supporting process definition, modeling, and simulation. The technologies and tools are presented in alphabetical order. The information includes the tool or technology name, contact information, the method supported by the tool, the platform the tool runs on, the target audience for the technology, and, a brief comment about the tool.

Modeling Method Abbreviations

IDEF/SADT:	Supports IDEF or SADT methods
SA:	Supports structured analysis
RTSA:	Supports real time structured analysis methods
EPM:	Supports entity process modeling
PPL:	Process programming language
SD:	Supports system dynamics modeling
PN:	Supports Petri Nets
DM:	Supports data modeling
AI:	Builds models using artificial intelligence techniques
OO:	Supports object-oriented model design
MI:	Methodology independent
D/S:	Supports definition and simulation of a process
D:	Supports definition of a process solely
E:	Method provides support for enactment

TOOL TECHNOLOGY	DEVELOPER/VENDOR	METHOD/ --- PLATFORM	TARGET --- COMMENTS
AI0	Knowledge Based Systems Inc Contact: David N. Rice 2746 Longmire College Station, TX 77845-5424 Phone: (409)696-7979 FAX:(409)696-7277 BBS:(409)696-7055	IDEF/SADT: D --- DT	MIS, TECH, REAL-TIME --- IDEF0 package for PC. (Issue of simulating their IDEF0 models being worked.)
AI4	Knowledge Based Systems Inc Contact: David N. Rice 2746 Longmire College Station, TX 77845-5424 Phone: (409)696-7979 FAX:(409)696-7277 BBS:(409)696-7055	IDEF: D --- DT, WS (planned)	MIS, TECH, REAL-TIME --- An interactive Unix-based tool for object-oriented design.
AGE/ASA	Verilog, Inc. 3010 LBJ Freeway Suite 900 Dallas, TX 75234 contact: R. Wesley Hair 214-241-6595	IDEF/SADT:D/S --- DT, WS	TECH, REAL-TIME --- Rqmts and testing workbench based on SADT/IDEF0.
Ada Process Programming Language based on Aspen (APPL/A)	Computer Science Dept. University of Colorado Boulder, Colorado 80309-0430 contact: Stanley M. Sutton, Jr. phone: (303) 492-7906 fax: (303) 492-2844	PPL: D/S/E --- N/A	ALL --- Ada superset providing machine-executable support (developed as part of the DARPA Arcadia project).
AP5	contact: N. Goldman and K. Narayanaswamy	AI: D	Approach using AI methods including rules pre/post- conditions, events and triggers. Also uses object modeling including class types and instances, hierarchy and inheritance.
Authormate	Eclectic Solutions Corporation Contact: Pat Duran/Al Irvine 5580 LaJolla Boulevard Suite 130 LaJolla, CA 92037-7692 Phone (619)454-5781 (619)457-4511 FAX: (619)450-9949	IDEF/SADT:D --- DT, MF	ALL --- Used to be called "COINS." This is a complete modeler's workstation environment.
AutoSADT	TRIUNE Software, Inc. 2900 Presidential Dr. Ste 240 Fairborn, OH 45324 ph: (513)427-9900 fax: (513)427-9964	IDEF/SADT:D --- DT	MIS, TECH --- MAC + PC Windows tool version that has less functionality than other IDEF/SADT tools, and is less expensive.
BPwin	Logic Works Contact: Jeffrey D. Mershon 1060 Route 206 Princeton, NJ 08540 Phone: (609)252-1177 FAX: (609)2521175 Compuserve: 70262,1135@compuserve.com	IDEF/SADT:D --- DT	BPwin supports the IDEF0 activity modeling notation, and supports creation of an IDEF0 kit. BPwin also supports Activity Based Costing

Business Design Facility (BDF)	Texas Instruments Plano, TX phone: 1-214-575-5599	DM: D --- WS	MIS --- Analysts can create "as-is" and "to-be" models of an organization to aid in process engineering decisions.
Design/CPN	Meta Software Corp. 150 CambridgePark Dr. Cambridge, MA 02140 (617) 576-6920	PN: D,S --- DT, WS	ALL --- Tool that can be used to support process definition using the formal principles of hierarchical colored Petri nets.
Design/IDEF	Meta Software Corp. 150 CambridgePark Dr. Cambridge, MA 02140 (617) 576-6920	IDEF/SADT:D --- DT, WS	MIS, TECH --- Tool can be used to support process definition using IDEF graphical notation. Design/CPN can then be used for simulation if that is desired. (A MetaSoftware tool exists for translating IDEF diagrams to Petri Net format.)
DYNAMO	Pugh Roberts Associates, Inc. 41 William Linskey Way Cambridge, MA 02142 617-864-8880	SD: D,S --- DT	MIS, TECH, OTHER --- System Dynamics approach.
EasyCASE Professional	Evergreen CASE Tools 8522 154th Avenue, NE Redmond, WA 98052 phone: (206) 881-5149 fax: (206) 883-7676	IDEF1X, SA, RTSA, IDEF1X: D --- WS, DT	ALL --- Process Modeling supporting graphical editing, with an underlying data dictionary and report generation
EasyCASE System Designer	Evergreen CASE Tools 8522 154th Avenue, NE Redmond, WA 98052 phone: (206) 881-5149 fax: (206) 883-7676	IDEF1X, DM, SA, RTSA, IDEF1X: D --- WS, DT	ALL --- Same functionality as EasyCASE Professional, but sed for data modeling, extracts chema from your E-R diagrams, view entity attributes right on your charts.
ERWIN	Logic Works Contact: Jeffrey D. Mershon 1060 Route 206 Princeton, NJ 08540 Phone: (609)252-1177 FAX: (609)2521175 Compuserve: 70262,1135@compuserve.com	IDEF:D --- DT	TECH, MIS, REAL-TIME --- Products concentrate on data modeling (IDEF1, IDEF1X).
ESF/SPECIMEN (SPECIMEN is a project within ESF oriented around the creation of process programs)		PPL: D	ALL --- Process Control Language (PCL) provides the process control mechanisms of the environment.
Excelerator II	Intersolv 3200 Tower Oaks Blvd. Rockville, MD 20852 tele: (301) 230-3200 fax: (301) 231-7813	SA, RTSA, OO, DM: D,E --- MF, WS, DT	MIS --- Users can model data-driven, process-driven, and event-driven systems using methodologies off-the-shelf or tailor them using the Customizer.
Formal Software Process Notation (FSPN)	Software Productivity Consortium SPC Building 2214 Rock Hill Road Herndon, VA 22070-9858	PPL: D, E	Notation designed to map to a software development environment to permit process management.

GRAPPLE	Computer and Information Science Department University of Massachusetts Amherst, MA contact: Karen Huff	AI: D	A system based on an AI planning paradigm.
The Hierarchical and Functional Software Process (HFSP) description and enactment language	Department of Computer Science Tokyo Institute of Technology contact: T. Katayama	PPL: D/E	Functional language with constructs supporting enaction.
IDEFine	Wizdom Systems Inc Contact: Don Sloan 1300 Iroquois Avenue Naperville, IL 60563 Phone: (708)357-3000 FAX: (708)357-3059	IDEF/SADT:D --- DT	TECH, MIS, REAL-TIME --- PC-based IDEF0 and IDEF1X process definition tools; exports in format readable by CACI's SIMprocess for simulation.
IDEF/ LEVERAGE	D. Appleton 1334 Park View Ave Suite 220 Manhattan Beach, CA 90266 Phone: (310)546-7575	IDEF/SADT:D --- DT, WS, MF	ALL --- Personal IDEF/ LEVERAGE is a PC based IDEF0 modeling package. Regular IDEF/ LEVERAGE on large host mainframe can read these models via a standard modeling language (SML, AML) for reporting, analysis, merging, etc.
Integrated Model Development Environment (IMDE)	TASC 2555 University Blvd. Fairborn, OH 45324 tel. (513) 426-1040 fax. (513) 426-8888 contact: Patrick Clark	OO: D/S --- WS	TECH --- Supports graphical, object-oriented construction of simulation models.
LBMS Project Engineer	LBMS, Inc. 1800 West Loop South Suite 1800 Houston, TX 77027 (713) 682-8530 1-800-231-7515	MI: D/E --- DT	ALL --- Project planning and estimation tool; can be used to support any development method.
MicroWorld Creator	MicroWorlds Inc. 347 Broadway Street Cambridge, MA 02139 (617) 547-9898	SD: D/S --- DT	TECH, MIS, REAL-TIME PROJECT-MGMT --- System Dynamics approach.
MVP-L	The Multi-View Modeling Project (MVP) University of Maryland College Park, MD 20742 contact: Christopher M. Lott cml@cs.umd.edu	PPL: D	Rule-based textual process representation language.
ObjectMaker	Mark V	SA, OO, DM --- DT, WS	ObjectMaker is a software development workbench using a variety of methods and supporting analysis, design, code generation, and reverse engineering. Model attributes stored in a common repository, and can be viewed using different notations.
Object Modeling Workbench (OMW)	James Martin and Co. Rosemont, IL and Intellicorp Mountain View, CA phone: 1-415-965-5634	D/S	Supports execution of process models defined through the semi-formal graphical notation espoused by Martin and Odell; aimed for the business market.

PACE	Grossenbacher Elektronik AG Spinnereistrasse 8 9008 St. Gallen Switzerland tel. 011 42 72 26 31 51 fax: 011 41 71 24 04 06 contact:Robert Schopflin	PN, OO: D/S	Graphical interactive tool based on high level Petri Nets, combined with object oriented data modeling.
PLAN/1	Andersen Consulting 69 West Washington St. Chicago, IL 60602 tele: (312) 580-0069 fax: (312) 507-2548	SADT/IDEF, DM: D --- DT	MIS, TECH --- functional modeling using diagrams with SADT-type syntax; in addition a data modeling editor for E-R diagrams and data modeling. One of 5 major modules in Andersen's life-cycle application development environment, Foundation.
PMDB	Info taken from article titled "Process Modeling," as seen in Communications of the ACM, Vol 35, No. 9, September, 1992.. Authors were Bill Curtis, Mark Kellner, and Jim Over.	D/E	Method emphasizes data modeling including E/R, relations, and structured data declarations; produces executable software process models; has been used to develop and analyze models of actual software processes.
ProCAP	Knowledge Based Systems Inc Contact: David N. Rice 2746 Longmire College Station, TX 77845-5424 Phone: (409)696-7979 FAX:(409)696-7277 BBS:(409)696-7055	IDEF: D --- DT, WS	MIS, TECH, REAL-TIME --- PC/MAC/UNIX. Process Description Capture: Uses the new IDEF3 technology, which has a different look/syntax. than IDEF0. Captures precedence & causality relations between situations and events giving models a concept of time. IDEF0 doesn't capture those concepts; SADT does to some degree.
Prototype Engineering Information System (PREIS)	Honeywell Systems and Research Center 3660 Technology Dr. Minneapolis, MN 55418 contact: John Kimball	OO:D	**PREIS employs an object-oriented engineering approach which features a control points and policies mechanism for automating administrative actions.**
Process Modeling Language (PML)	Praxis Systems PLC 20 Manvers Street Bath BA1 1PX England contact: Clive Roberts	PPL: D/S/E	Object-Oriented Conceptual Modeling Language: - undertaken as part of IPSE 2.5 project.
ProTEM	SCIL 13812 SE 240th Kent, WA 98042 (206) 631-4212	PN: D/S --- DT	ALL --- Petri net-based modeling tool.
Programmer Command Center	Compuware Corporation 31440 Northwestern Highway Farmington Hills, MI 48018 (313) 737-7300		Process control product for mainframes.
ProEditor	Cecil Martin International Software Systems, Inc. 9430 Research Blvd Bldg.4, #250 Austin Texas 78759 (512) 338-5741	PPL: D/E --- WS	TECH, MIS, REAL-TIME --- This tool supports graphical editing/creation of process models, and supports the VPML notation (see VPML below)

RDD-100 (Requirements Driven Development)	Ascent Logic Corporation 180 Rose Orchard Way Suite 200 San Jose, CA 95134 phone: 408-943-0630 fax: 408-943-0705	EPM: D/S --- DT, WS	ALL --- The RDD language and notation (Behavior Diagram notation) combines data, control and functions in one graphical display.
Role Interaction Nets (RIN)	Microelectronics and Computer Technology Corp. (MCC) Austin, TX contact: B. Singh and G. Rein	PN: D	Technique models the role interaction structure of a project using a Petri net-based representation and language; formalism can be used as an underpinning for coordinating activities in a process-driven environment. Has now been used in a commercial product; strong in representing roles, dependencies and process elements; however, its representation of artifacts is weak.
SIMprocess	CACI Contact: Ron Flauaus 1100 North Glebe Road Arlington, VA 22201 (703)841-4430	IDEF/SADT:D/S --- DT, WS	MIS, BUSINESS PROCESS ENG. --- Simulates imported Wizdom Systems IDEF0 models. Can also simulate their own NON-IDEF models.
Software Process Management System (SPMS)	SAIC STC Cielo Center One, Suite 380 1250 Capital of Texas Hwy. S. Austin, TX 78746 contact: Jim Terrel Adam Linehan	D,S, E	Employs reuse-based mechanisms for developing project-specific software processes. Still in R&D stage. A fully functional SPMS will eventually be embedded in a process-centered STARS* SEE in the 1993-1996 time period.
Software through Pictures (StP)	Interactive Development Environments 595 Market Street, 10th Floor San Francisco, CA 94105 telephone: (415) 543-0900 fax: (415) 543-0145	RTSA: D --- WS	ALL --- Upper CASE tool which supports structured and object-oriented process definition development
STATEMATE	I-LOGIX, INC. 22 Third Ave. Burlington, MA (617) 272-8090	EPM: D/S --- WS	REAL-TIME --- Systems engineering tool which can be used to define and simulate a process definition.
STELLA II (and a similar product by High Performance Systems: IThink)	High Performance Systems, Inc. 45 Lyme Road Hanover, NH 03755 603-643-9636	SD: D/S --- DT	ALL --- System Dynamics approach.
System Architect	Popkin Software & Systems, Inc. 11 Park Place New York, NY 10007 (212) 571-3434	IDEF/SADT,RTSA, EPM, OO: D --- DT	ALL --- Supports a variety of methodologies, OOD, structure charts, state transition, decomposition, entity-relationship diagrams, flow charts.

* The Software Technology for Adaptable, Reliable Systems (STARS) Program is sponsored by ARPA, contracted through Air Force Electronic System Division, and involves three cooperating Primes - Boeing, IBM, and Paramax - and a large number of subcontractors. The STARS goal is to increase software productivity, reliability, and quality by synergistically integrating support for modern software development processes and modern reuse concepts within state-of-the-art software engineering environment technology

Teamwork	Cadre Technologies, Inc. 222 Richmond Street Providence, RI 02903 telephone: (401) 351-5950 fax: (401) 351-7380 contact: Barry Ackerman	RTSA: D --- WS	ALL --- Upper CASE tool which can be and has been used for process definition
TurboCASE	Structsoft 5416 156th Ave. SE Bellevue, WA 98006	SA, OO, EPM: D --- DT	TECH, REAL-TIME --- Modeling tool supporting the traditional analysis and design techniques as well as the newer O-O analysis and design methodologies.
Visual Programming Language (VPL)	Department of Computer Engineering, Royal Military College of Canada Kingston, Ontario Canada K7K 5LO contact: Terry Shepard fax: (613) 547-3053	PPL: D/E	Minimal tool support.
Visual Process Modeling Language (VPML)	Cecil Martin International Software Systems, Inc. 9430 Research Blvd Bldg.4, #250 Austin Texas 78759 (512) 338-5741	PPL: D/E --- DT,WS (GOTS Soon)	TECH, MIS, REAL-TIME --- Process language embodying three distinct perspectives: process modeling (ProVision), Information modeling, and resource modeling. Of these, only ProVision is currently implemented.

A.3 Process Frameworks List

This list contains information about SEE/IPSE frameworks supporting the creation of process-driven environments. A framework provides an interface for the building of SEEs. The technologies and tools are presented in alphabetical order. The information includes the tool or technology name, contact information, process technology category (type), platform, and, a brief comment about the tool.

TOOL TECHNOLOGY	DEVELOPER/VENDOR	PLATFORM --- COMMENTS
Atherton Software BackPlane	Michael Wendt Atherton Technology 1333 Bordeaux Drive Sunnyvale, CA 94089 Phone: (408) 734-9822 FAX: (408)744-1607	WS --- The Software Backplane was developed as a framework for building an Integrated Project Support Environment (IPSE), independent of tools, methodology, language or platforms, using an object-oriented tool integration methodology.
Common Ada Programming Support Environment Interface Set (CAIS-A)	Controlled by AJPO contact: Gary Pritchett 10875 Rancho Bernardo Rd. Suite 100 San Diego CA 92127 (619) 451-9301	WS --- Does not provide for process modeling; however, it provides lower-level capabilities upon which one may build those capabilities.
ESPRIT Portable Common Tool Environment * (PCTE)	W. Wohlschlegel, CEC 25 rue Archimede 1040 Brussels, Belgium 32-2-236-0257 suppliers: Bull Louvveciennes, France Software Sciences London, England	Repository interface specification standard; does not provide explicitly for process modeling services; however, it provides lower-level capabilities upon which one may build those capabilities.

* The project's goal was to provide a common interface for the Esprit program and define a complete set of framework services to support tool development. The PCTE interfaces were submitted for ISO standardization in 1990.

SoftBench 3.0	Hewlett-Packard Software Engineering Sys Div. 3403 /e. Harmony Rd. Ft. Collins, CO 80525-9599 tele: (303) 229-3800	WS -- Tool integration platform upon which a custom development environment can be built.

A.4 Computer Enactment Technology List

This list contains information about technologies and tools supporting process enactment. The technologies and tools are presented in alphabetical order. The information includes the tool or technology name, contact information, process technology category (type), platforms supported, and, a brief comment about the tool.

TOOL TECHNOLOGY	DEVELOPER/VENDOR	PLATFORM	COMMENTS
Apt	Computer Science Department University of Colorado Boulder, CO 80309-0430 contact: Dennis Heimbigner phone: (303) 492-6643 fax: (303) 492-2844 Part of Arcadia project	WS	Apt translates programs written in a subset of the AAPL/A process programming language into an equivalent Ada program that may be compiled and executed. (APPL/A is a process programming language used in some process-driven environments.)
CASE* Method	Oracle Corp. 500 Oracle Parkway Suite 4 Redwood Shores, CA 94065 (800) 345-3267	WS	Support for process mgmt in a Single vendor toolkit - not tailorable wrt methodology.
CaseWare/CM	CaseWare, Inc. 8300 Boone Blvd., Suite 500 Vienna, VA 22182 contact: Lesli Mangeri phone: (703) 848-9272 fax: (703)848-4586	WS	Uses a company-defined language, ACCENT, for process definition; once the process is defined, CaseWare/CM controls a project so that software and documents will be developed according to the process defined.
FastPACE	James Martin & Company Reston, VA phone: 1-312-693-5040 fax: 1-312-693-3112	WS	Tailored for the Information Engineering method as implemented under KnowledgeWare's ADW; integrated with other JMC tools.
firstCASE [Cross Life Cycle product under IBM's AD/Cycle]	AGS Management Systems 880 First Avenue King of Prussia, PA 19406 (215) 265-1550	WS	Process management for component CASE; interfaces to several major CASE tools and unifies the mgmt of tools, techniques, policies and procedures.
FirstEP	Jim Dutton International Software Systems, Inc. 9430 Research Blvd Bldg.4, #250 Austin Texas 78759 (512) 338-5741	WS	An enterprise and process description method. Enterprise modeling includes process modeling, infrastructure modeling, and information modeling. Used in the Pro-SLCSE environment.
Formal Software Process Notation (FSPN)	Software Productivity Consortium SPC Building 2214 Rock Hill Road Herndon, VA 22070-9858	Currently unimplemented	Notation designed to map to a software development environment to permit process management.
The Hierarchical and Functional Software Process (HFSP) description and enaction language	Department of Computer Science Tokyo Institute of Technology contact: T. Katayama		Functional language with constructs supporting enaction.

Integration Softboard	Denise Boucher Atherton Technology 1333 Bordeaux Drive Sunnyvale, CA 94089 (408) 734-9822	WS	Allows the behavior of any software engineering process written in any programming language to be used by the Atherton Backplane to enable the automation of development rules.
KI-Shell (Knowledge-based Integration Shell)	UES, Inc. 5162 Blazer Memorial Pkwy Dublin, Ohio 43017-1339 contact: Dale Upshaw phone: (614) 792-9993 fax: (614) 792-0998	WS	Advertised as "The best off-the shelf object-oriented product for process modeling and enactment."
METHOD/1	Andersen Consulting 69 West Washington St. Chicago, IL 60602 tele: (312) 580-0069 fax: (312) 507-2548	WS,DT	Provides an automated system for work planning and project control based on Andersen's proprietary information planning development process. One of 5 major modules in Andersen's life-cycle application development environment, Foundation.
Process Weaver	CAP Gemini America 5120 Goldleaf Circle Suite 130 Los Angeles, CA 90056 contact: Mr. Larry Proctor (213) 291-7804	WS	Research for this tool was done in the Eureka Software Factory (ESF); product is a commercially available, tailorable process management tool.
SynerVision	Hewlett-Packard Software Engineering Sys Div. 3403 /e. Harmony Rd. Ft. Collins, CO 80525-9599 tele: (303) 229-3800	WS	Process engine that can be used to provide process automation, definition, guidance and metrics in a SoftBench process environment.
Toolbuilder	IPSYS Software Marlborough Court Pickford Street, Macclesfield SK11 6JD, Cheshire, UK local distributor: Newbury, MA tele: (508) 463-0006 fax: (508) 462-9198	WS	A Meta-CASE environment that enables rapid prototyping and development of customized CASE environments.
Virtual Software Factory (VSF) Limited	8300 Boone Boulevard Vienna, VA 22182 contact: Dr. Mel Selwood (703) 848-9282	DT, WS	Workbench that allows user to specify methodology; methodology independent - can be configured to support any method; has been used for SADT, OOD and IDEF.

A.5 Process Driven Environments List

This list contains information about SEE/IPSEs supporting process-driven development. The technologies and tools are presented in alphabetical order. The information includes the environment name, contact information, enactment approach (method), platform, target users, and, a brief comment about the tool.

Environment Abbreviations

F: Environment built on a framework that supports enactment
CEE: Customizable enactment environment
PDEE: Environment generated by process definition
TPDE: Non-customizable turnkey process-driven environment
D/E: Environment supports both definition and enactment
D/S/E: Environment supports definition, simulation and enactment

TOOL TECHNOLOGY	DEVELOPER/VENDOR	METHOD --- PLATFORM	TARGET --- COMMENTS
Arcadia*	Richard Taylor Information and Computer Science, University of California Irvine, CA 92717	PDEE: D/E	More advanced than usual university prototype/Hybrid approach combining procedural and rule-based paradigms.
Concerto	Sema Group France	SEE	European Software Engineering Factory; targeted for the support of technical applications; open and configurable architecture; on the market now.
EPOS (Engineering and Project-management Oriented Support System)	GPP Kolpingring 18a Oberhaching, Munchen D-8024, Germany tele: +49-89-613041 fax: +49-89-61304-294	TPDE: D/E --- MF, WS, DT	MIS, TECH, REAL-TIME --- Fully integrated, yet open life-cycle environment allowing integration of user or third-party tools.

* The Arcadia Consortium consists of a collection of separately funded, informally coordinated research and development projects at the University of California at Irvine (UCI), the University of Colorado at Boulder (UCB), the University of Massachusetts at Amherst (UMass), Stanford University, Incremental Systems Corporation, and TRW Defense Systems Group. This Consortium was formed in August 1987. The funding of these projects is provided by the Advanced Research Projects Agency (ARPA), and is administered by the National Science Foundation (NSF), and the Navy's Space and Naval Warfare Systems Command (SPAWAR). The universities also have other sources of funding, and the corporations have made internally funded investments in the work of the Arcadia Consortium.

ESF** (EUREKA Software Factory)	ESF Hohenzollemdamn 152 D-1000 Berlin, Germany	PDEE: D/E	Process model-driven hybrid approach; completion 1996.
EUREKA Advanced Software Technology (EAST) Project	SFGL 14, rue de le Ferme 92100 Boulogne France tel: (33-1) 47 61 05 20 fax: (33-1) 47 61 92 15	SEE: D/E	Project aimed at producing an integrated software factory/PCTE-based open systems CASE environment for a number of application domains. This is now commercially available.
Foundation	Anderson Consulting 69 W. Washington Chicago, IL 60602 tele: (312) 580-0069 fax: (312) 507-2548	CEE: D/E --- MF, WS, DT	MIS, TECH --- An integrated CASE environment to automate and manage the entire systems development life cycle.
I-CASE	Harris Strategic Alliance contacts: Jeff DePasquale, Harris (407) 242-5223 Amy Snare, Paramax (703) 620-7163	SEE: D/S/E --- WS	ALL – Uses PCMS configuration mgmt system to define process (roles, entry-exit conditions, etc.); env enforces process; context and tools brought up automatically when user clicks.
Information Engineering Facility	Texas Instruments P.O. Box 869305, M/S 8474 Plano, TX 75086 (214) 575-4405	TPDE: E	Support for process mgmt. for business information and activities in a single vendor toolkit - based on Information Engineering Methodology (IEM) - not tailorable wrt methodology.
Inscape Environment	AT&T Bell Laboratories Murray Hill, NJ contact: Dewayne E. Perry	SEE: D/E	Pre-conditions and post-conditions used as points of interconnections between software development activities.
ISPW	Benchmark Technologies Suite 300, 839 Fifth Ave. S.W. Calgary, T2P 3C8 Alberta, Canada tele: (403) 269-7499 fax: (403) 265-2379	CEE: D/E – MF	MIS --- Automated application management environment - can be customized for an organization's own process and standards.
ISTAR	Imperial Software Technology 60 Albert Court Prince Consort Road London SW7 2BH England	SEE: D/E	An early commercial example of a process-driven environment; supports a "contractual" approach to process definition.
Maestro	Softlab Inc. 188 The Embardcadero Bayside Plaza, 7th Floor San Francisco, CA 94105 (415) 957-9175	TPDE: E	Support for process, tightly woven with methodology espoused by vendor - focuses primarily on project mgmt, large-scale sw engineering information collection, storage, retrieval, and control.
MARVEL	Columbia University New York, NY/ Software Engineering Institute Carnegie Mellon Univ. Pittsburgh, PA 15213	PDEE:E	Rule-based approach; MARVEL is an instance of a rule-based process server. A recent prototype in C is available for trial use.

** Over two hundred people spread across more than twenty sites in five countries are involved in the ESF project. The companies involved represent computer manufacturers, research institutions, CASE tool producers, and system developers. By 1991, halfway into the 10-year project, ESF has defined a reference architecture, completed the first implementation of a supporting framework and various tools and tool prototypes, and has undertaken several factory-integration experiments.

MATE (Methods and Tools Expert)	Advanced Development Methods 49 Solomon Pierce Rd. Lexington, MA 02173 tele: (617) 861-7848 fax: (617) 861-3817	CEE: D/E --- WS, DT	MIS Primarily --- knowledge-based system for managing the systems development process; the ADM methodology is a turnkey process which comes loaded into MATE - it may be modified or replaced with custom methods.
MELMAC	Dortmund University Dortmund, Germany	PDEE: D/E	Uses FUNSOFT nets (high-level Petri nets) to define software processes; mechanism based on modification points used to cope with software process model modifications during software process execution.
Navigator Systems Series	Ernst & Young 600 East Las Colinas Boulevard Irving, TX 75039 contact: Clark Norman phone: 214-444-2165 fax: 214-444-2102	TPDE: D/E	Information engineering-based turnkey system which defines the process for information engineering system development automated and integrated through CASE technology.
PACT ***	The PACT Project Bull S.A. 68, Route de Versailles 78430 Louveciennes France	SEE/F: D/E	Environment that uses the framework services of the environment framework, PCTE.
Pro-SLCSE	Cecil Martin International Software Systems, Inc. 9430 Research Blvd Bldg.4, #250 Austin Texas 78759 (512) 338-5741	SEE: D/E --- WS	TECH, MIS, REAL-TIME --- Uses Visual Process Modeling Language (VPML) and a data-flow paradigm to define processes.
SLCSE	Frank LaMonica USAF Rome Laboratory C3I Directorate Software Technology Division Software Engineering Branch Griffiss AFB, NY 13441 (315) 330-2054	SEE: D/E	Minimal rule-based capabilities largely related to tool invocation.
Softman environment	Decisions Systems Dept. University of Southern Calif. Los Angeles, CA 90089	SEE: E	Part of USC's System Factory project.
Summit Process	Coopers & Lybrand P.O. Box 5258 Princeton, NJ -8540 contact: John Dilley phone: 609-520-6161 fax: 609-452-0177	CEE: D/E --- DT	MIS --- Incorporates a fully automated methodology which covers the entire development life cycle; methodology can be modified or rewritten; major customer emphasis - information systems.
Synergy	CASE Methods Development Corporation 100 North Central Expressway Suite #710 Richardson, Texas 75080 tele: (214) 437-9700	SEE/F: D/E --- WS	ALL --- Allows definition of process via a textual definition language and populates a database; tools can be launched.

*** The PACT project is part of the ESPRIT program and is partially funded by the Commission for the European Communities. The PACT project members include Bull SA, Eurosoft, GEC, Software Ltd, ICL, Olivetti, Siemens, Syseca, and Systems and Management.

System Factory	Dr. Walt Scacchi Decisions Systems Dept. University of Southern Calif. Los Angeles, CA 90089 (213) 740-4782	PDEE: D/S/E -- WS	TECH, MIS, REAL-TIME --- A configurable environment of software process engineering technologies for large-scale development applications; being transferred into commercial products at this time.
TRIAD/CML	Ohio State University/ Universal Energy Systems (UES) Columbus, Ohio	PDEE: D/E	Imperative approach using Conceptual Modeling Language (CML).

(This page is intentionally left blank)

Appendix B - Process Technology Product Sheets (See Volume II)

Volume II of this report contains technology product sheets for most of the process technologies and tools in the technology/tool lists. These reports provide detailed information on process technologies and tools. Users of these reports should be able to make preliminary tool assessments based on the provided information. Information on pricing, contact, support, process technology areas covered, intended users of the technology or tool, intended application area, primary methodology base, hardware platforms, and general tool capabilities is included. The information in the reports was obtained either directly from the vendor or from the vendor's literature. In most cases, the vendor has supplied the information.

There are tools in the tool lists for which there is no associated technology product sheet. This condition occurs because there was insufficient available information to create the technology product sheet, either because the vendor did not supply information in time for publication or because the tool was added to the tool list too late for the creation of a technology product sheet.

The STSC can be contacted for both unpublished and updated reports that may be available. For information on how to order updated reports and/or Volume II of this report, contact the STSC customer service department at (801)777-7703 or DSN 458-7703, fax to (801)777-8069 or DSN 458-8069, or email to godfreys@wpo.hill.af.mil.

Appendix C - Process Technology Product Critiques (See Volume II)

Volume II of this report contains a number of method/tool critiques and a template that can be used for additional process method/tool critiques. We welcome input from readers of this report who have used process technology methods and tools and would be willing to share their experiences with others, and for that reason, a critique template has also been included here in Volume I. All user critiques are written by actual tool users. Editing by the STSC is kept to an absolute minimum. A critique has seven sections: (1) Reviewer's Background Information, (2) Tool Name, (3) Project Information, (4) Notable Strengths, (5) Notable Weaknesses, (6) Advice for Potential Users of Method/Tool, and (7) Vendor Comments. The purpose of the Reviewer's Background and Project Information sections is to give a critique reader an idea of the background of the evaluator and the nature of the project using the method/tool so that the applicability of the critique to the reader's context can be judged. Actual names of reviewers or reviewer affiliations do not appear in this report. The Strengths, Weaknesses, and Advice for Potential Buyers sections allow free-form commentary about the tool by the evaluator. The Vendor Comments section is allows for vendor response.

For information on how to order Volume II of this report, please contact the STSC customer service department at (801)777-7703 or DSN 458-7703, fax to (801)777-8069 or DSN 458-8069, or email to godfreys@wpo.hill.af.mil.

Sample Product Critique Sheet

Reviewer's Name: _____
Company Name: _____
Position/Title: _____
Main Duties: _____
_____
Tool used for software process: yes/no
Years of software experience: _____
Years of experience with the tool: _____
Last time tool used:
Currently **6 months** **1 year** **>1 year**
I am a software:
Manager **Engineer** **Programmer** **Novice**
Date of Review: _____

Tool Name: _____
Vendor: _____
Version: _____
Hardware platform: _____
Operating system: _____
Memory used: _____
Disk space used: _____
Enhancements: _____
(accelerator, large monitor, graphics card, etc.)
Overall impression of the tool?
Excellent **Good** **Fair** **Poor**
Quality of vendor support?
Excellent **Good** **Fair** **Poor** **Unknown**

Project Information:

Notable Strength(s) of the Tool:

Notable Weakness(es) of the Tool:

Advice for Potential Buyers of this Tool:

Vendor Response:

STSC Product Sheet Version 2.1

(This page is intentionally left blank)

Appendix D - Process Technology Taxonomy

D.1 Evaluation Taxonomy for Process Technologies

In order to facilitate the evaluation of process technologies, checklists are provided for assessment methods, modeling methods, modeling tools, and SEEs designed for software process model enactment. Each checklist incorporates two types of entries:

(1) Entries that represent attributes that can be checked if a technology has the given attribute.

(2) Entries which provide a list of options, one of which can be checked for the technology being evaluated.

These checklists can be used to characterize the features, strengths and weaknesses of a technology and hence assist the user to choose methods or tools with the appropriate attributes for a given project.

In the checklists provided for process technology methods and tools, an effort was made to include in the list entries that can be determined quantitatively. Entries are provided for functional characteristics. For methods, entries are also provided for level of tool support available for the method; for tools, specific entries allow the level of support a particular tool supports to be indicated.

Quality characteristics are not included in the checklist. However, when distinguishing between methods and tools with similar functionality and level of tool support, quality characteristics, which often cannot be quantitatively measured, must also be considered. Therefore, a list of pertinent quality characteristics for process technologies are provided in section D.1.3.

Functional characteristics, level of tool support, and quality attributes are discussed in sections D.1.1, D.1.2, and D.1.3 respectively.

D.1.1 Functional Characteristics

Functional characteristics for each technology level, to a very detailed level, have been provided. Such characteristics have to be considered carefully when methods, tools or environments are selected because it is at this level of detail that tradeoffs are made and priorities are set.

Some major input to the organization of the list of enactment characteristics was provided by the "Reference Model for Frameworks of Computer Assisted Software Engineering Environments[47]." Many of the process assessment characteristics were taken from Watts Humphrey's book, "Managing the Software Process[48]," and Terry Bollinger's article, "A Critical Look at Software Capability Evaluations[49]." Some of the process definition characteristics were taken from Marc Kellner's article "Software Process Modeling: A Case Study[50]." A number of others, in particular characteristics which allow one to evaluate the representative power of a process definition approach, were adapted from a list of process description characteristics included in the Software Productivity Consortium document, "Process Definition and Process Modeling Methods[51]."

D.1.2 Level of Tool Support

Tool support can vary from minimal support at one extreme to full environment-assisted support at the other. For process assessment, assessment procedures may be provided in manuals, on-line help may be available, or there may be some limited integration of tools available to support the assessment process. Similarly, modeling methods may be supported with a written description of the method at one extreme and full tool support in a process-driven SEE at the other. Modeling tools may provide various levels of support. Enactment technologies can lead to human enactment or computer enactment; support of a

---

47 "A Reference Model for Frameworks of Computer Assisted Software Engineering Environments," NIST draft version 1.3, prepared by the NIST ISEE Working Group, July, 1991.

48 Humphrey, Watts, **Managing the Software Process**, Addison-Wesley Publishing Company, Inc., 1989.

49 Bollinger, Terry, B., McGowan, Clement, "A Critical Look at Software Capability Evaluations," IEEE Software, July, 1991.

50 Kellner, Marc I., Hansen, G.A., "Software Process Modeling: A Case Study," IEEE, 1989.

51 Lai, Robert, "Process Definition and Process Modeling Methods," Software Productivity Consortium, SPC-91084-N, 1991.

defined process or control of the process. The evaluation checklists ask the evaluator to specify the level of support by choosing a level of support from the following five levels:

(1) *Manual-Based Methodology Support.* Manual-based process methodology support refers to the level of support in which standards, manuals, documents, and practices are provided for method practitioners to follow. However, no level of electronic tool support is available.

(2) *Method Definition Available On-Line.* The process practitioners are expected to follow is provided on-line (electronically). Though standards, manuals, documents, and practices are provided on-line within a software engineering environment, preferably in some hypertext/hypermedia form, this level of support provides no further automated support for carrying out the process.

(3) *Limited Integration of Tools Supporting Process.* At this level, additional support for carrying out the defined process is provided by tools that are either not integrated or provide limited integration. Interaction between tools will most likely be file-based interaction. Methodology guidance may be tied into tool invocation scripts or in logon and logoff scripts.

(4) *Process-Supported SEE.* In a process-supported SEE, the main unit of work is the invocation of tools. This level of support can also include monitoring activities in the environment and the ability to present to each user the appropriate current view of the development to select the next activity from choices that are appropriate. A project member logs in and is presented with the available activities that can be performed by the user. The user then selects an activity, which results in either a screen of subprocess activities or results in direct invocation of applications for work. Such a system leads the user through steps by presenting a choice of activity-specific menu options and then returns control to the user to perform creative aspects of a task. Strict guidance can be provided by the system or, alternatively, the system can provide advice and a series of options. Integration of tools where tools interact through the SEE database is customary at this level of support.

(5) *Process-Driven SEE.* In a process-driven SEE, the main unit of work is the invocation of process steps, where tools and data are then made available for

performing each process step. At this level of support, action item messages are sent to project members when they log in with appropriate instructions for accomplishing activities. Completion of one action item creates another action item that must be addressed. This level of support emphasizes control goals. Some necessary work steps may be automatically executed by the environment. Fundamental interaction with the environment framework is assumed. Knowledge-based components and a rule-based approach are customary at this level of support. In a rule-based approach, activities are described by rules with pre- and post-conditions. In general, the SEE promotes a mutual assistance between a well informed initiative engine and human developers.

D.1.3 Quality Attributes

Evaluating methods, tools, and environments in terms of overall quality and the quality of each supported functional capability will permit the evaluator to differentiate between entities with equivalent functionality. Though quality attributes cannot, as a rule, be quantitatively evaluated and therefore are not included in the checklists, guidelines are provided here to facilitate the assessment of quality when distinguishing between tools/methods/environments of similar functionality. The following comprehensive list of quality attributes are taken from Rome Air Development Center's "Specification of Software Quality Attributes: Software Quality Evaluation Guidebook[52]." The Rome study defined quality attributes for delivered software. Nevertheless, with some tailoring, the characteristics apply equally well to methods, tools, and environments.

While this is a comprehensive list, depending on the technology being evaluated, a subset of quality attributes will sometimes be most useful when distinguishing between methods, tools or environments with equivalent functionality. For example, when evaluating assessment methods, the quality characteristics of efficiency, usability, verifiability and expandability are most important. For the evaluation of modeling methods, the quality characteristics of efficiency, verifiability, usability, and reusability are most pertinent, with the most important quality characteristic being usability. However, when looking at

---

[52] Bowen, Thomas P., Wigle, G. and Tsai, J., "Specification of Software Quality Attributes: Software Quality Evaluation Guidebook," RADC-TR-85-37, Rome Air Development Center, Griffiss AFB, NY, February 1985. The Rome study was later extended by the Ada Evaluation and Validation project, the purpose of which was to "provide information that will help users to assess [Ada Programming Support Environments] APSEs and APSE components," Resulting in the Ada E&V Guidebook, Version 3.0, 14 February, 1991.

modeling tools or enactment SEEs, the entire list of quality characteristics must be considered: efficiency, integrity, reliability, survivability, usability, correctness, maintainability, verifiability, expandability, interoperability, reusability, and transportability.

(1) *Efficiency.* When judging the efficiency of a method, the ratio of the effective or useful output to the total input of human effort must be considered. In tool and environment evaluation, three areas that need to be looked at are processor/human time to complete a task, secondary storage requirements for computer-based tools/methods, and I/O/network considerations for multiprocessor systems and/or multi-user systems. When a tool or environment performs adequately with respect to these resources for a particular process definition, modeling and/or enactment scenario, its efficiency is judged acceptable.

(2) *Integrity.* Integrity needs to be assessed for tools and environments. Process definition/modeling/management failures due to unauthorized access or the corruption of the tool or SEE database will cause a low rating in this category.

(3) *Reliability.* Reliability also needs to be assessed for tools and environments. Reliability can be defined as the absence of failures. Indicators such as maturity, published error reports, and errors uncovered during testing will have to guide the tool and environment assessment in this area.

(4) *Survivability.* Survivability deals with the ability of the tool or environment to perform even when portions of the system have failed. This is a desirable characteristic, especially in a distributed system (e.g., a client/server system). Therefore, it is included in the quality list. However, this characteristic probably will not serve as a distinguishing characteristic in environments being evaluated today.

(5) *Usability.* Usability is the extent to which resources required to acquire, install, learn, operate, prepare input for, and interpret output of a tool or environment are minimized. User interfaces, user documentation, and extent of training necessary must be evaluated.

(6) *Correctness/Verifiability.* Verifiability in terms of assessment or modeling methods refers to the ability of the developers of the method to prove that products produced using the methodology are complete, consistent, and correct. For tools and environments, correctness is the extent to which tool or environment design and implementation conforms to specifications and standards. Verifiability gauges the extent of testing and verification of correctness that has been completed for a tool or environment.

(7) *Maintainability.* Maintainability, as applied to software process tool and environment technology, refers to the ability of the vendor or researcher to deliver tool or environment maintenance in a timely manner.

(8) *Expandability.* Expandability is the ease with which a method can evolve: for a tool, the ease with which current functions can be enhanced, and new functions can be added; for a SEE, the ease with which new tools can be added to a process-driven environment. Expandability also refers to the ease with which the method, tool, or environment can be changed to meet new requirements. It is an especially important characteristic when evaluating process definition technologies and process-driven environments.

(9) *Interoperability.* Interoperability tests the ability of tools to communicate with other tools (e.g., the extent to which open architecture standards are adhered) and of environment databases to support the need for tools to exchange information without conversion.

(10) *Reusability.* Reusability is the extent to which a component developed in one application can be adapted for use in another application. This is an especially important attribute for methods and tools supporting process definition. The use of assets from a process definition library to define a software process is probably essential given the formality and rigor necessary for adequate software process definition.

(11) *Transportability.* Transportability is the ability of a tool or environment to be installed in a different development configuration without extensive changes. The number and variety of platforms and operating systems with which a tool or environment can be used provides a good measure of its transportability.

D.2 Assessment

Process assessment technologies are defined as those technologies that enable an organization to characterize the maturity of its process. An assessment should ideally identify the most critical process issues and facilitate the initiation of process improvement actions. Characteristics in the Assessment Method Checklist can be used to judge the suitability of a particular assessment method.

Assessment Method Checklist

Assessment Method Checklist	
Assessment method uses a rigorously proven standard or maturity model	
Orderly, systematic assessment steps	
Assessment method is driven by a standard set of questions	
Grading system used:	
Numerical grading system	
Maturity level grading system	
No explicit grading (neither leveled or graded; instead, a summary of findings)	
Assessment includes methods for recording and analyzing process issues	
Assessment method requires process-modeling	
Assessment of an organization's software process takes into account:	
Management practices	
Software process	
Software Tools	
Software Technology	
Software Standards	
Software Testing	
Software process definition	
Configuration controls	
Quality assurance practices	
Project estimation practices	
Data collection and analysis practices	
Defect prevention	
Efficiency of practices used by an organization	
Method is statistically reliable	
Sufficient data and quality measures are gathered to permit reliable analysis of complex organizational practices	
Data collection and analysis approach:	
Dense data approach **	
Sparse data approach*	
Level of tool support available:	
Textual-based assessment procedure descriptions exist	
Assessment procedures available on-line	
Limited integration of tools available to support assessment process	

* A small number of widely spaced data points (e.g., small number of questions, with little overlap between question topics) are used to assess maturity.

** A large number of closely spaced data points (e.g., much information is gathered; many questions are asked in each area of concern) are used to assess maturity.

D.3 Modeling

The terms "process definition," "process simulation," and "process modeling" are often used interchangeably in the literature. The term process modeling is used in this taxonomy to describe the process technologies that allow both definition and simulation of the software process.

Methods that support process definition must support the definition of a software process with a sufficient degree of formality. Characteristics in the Modeling Method Checklist can be used to judge the suitability of a particular definition method.

While few tools targeted specifically for process modeling are being produced by vendors at this point, tools built for other purposes can often meet the needs for process modeling. A number of existing tools can be used to capture a process definition. In addition, many tools currently available can assist in the simulation of a process definition. Criteria in the Modeling Tool Checklist can be used to judge the suitability of tools for supporting the area of software process modeling.

Modeling Method Checklist

Modeling Method Checklist	
Type of process notation used:	
Free-form English language descriptions	
Semi-formal, structured English	
Process programming language (e.g., MVP-L)	
Graphical notation (e.g., Data-flow/SADT/IDEF/real-time structured analysis type diagrams)	
Set of declarative rules/knowledge-based definition	
Formal machine-executable definition notation (e.g.. APPL/A)	
Petri nets	
State charts	
System dynamics	
Strategy for model building:	
Build from scratch	
Generic (building block) structure - using building blocks, asset library, "build-to" templates for process component types and/or process examples	
Standard software process models which can be customized for each project's needs	
Process description:	
Permits necessary flexibility in the process	
Requires step-by-step following of the process	
Support for multiple, complementary viewpoints of the process:	
Functional	
Behavioral	
Organizational (who implements the activities and where they are implemented)	
Data modeling	
Support for hierarchical representation of a process - vertical decomposition	
Support for multiple levels of abstraction	
Support for the defining of software development as a dynamic and distributed activity	
Support for the representation and analysis of constraints on the process (such as regulations, standards)	
Support for definition of resource requirements:	
Roles users take while performing a task	
Software artifacts that are needed, created or enhanced during a task	
Tools used	
Information about a task's schedule and its expected duration	
Support for creation and management of variants, versions and reusable process description components	

Modeling Method Checklist	
Representative power:	
Precision - the process description allows all necessary actions to be specified	
Scalable for different sizes of software projects	
Automation can be applied on the process described	
Communicable to humans	
The method defines the following:	
Process steps	
The mapping of individual process steps to tools or humans	
Pre-conditions for execution of a process step	
Post-conditions for completion of a process step	
What constitutes a process enactment state	
Pre- and post-conditions for enactment	
Constraints or policies to be checked or enforced during process enactment	
Project data operated upon, both input and generated	
Process control services including overall specification of control activities, sequencing, and generation of project plans	
Allowable concurrency and synchronization (if any) with other processes	
Product and process metrics to be collected	
Ability of method and its notation to support analysis:	
Internal consistency checks	
Completeness	
Correctness	
Definition support for the capturing of metrics and measurements	
Definition notation allows simulation of the process	
Approach can be integrated with other useful approaches - for example, management (e.g., PERT) or analysis methods (e.g., CPM)	
Level of tool support available:	
Textual-based definition descriptions exist	
Modeling method descriptions available on-line	
Tool support of method (with limited integration to other tools)	
Support of modeling method available in a process-supported SEE	
Support of modeling method available in a process-driven SEE	

Modeling Tool Checklist

Modeling Tool Checklist	
Fully supports a process modeling method	
Tailorable to support multiple process modeling methods	
Tool is sufficiently independent so that a change in execution environment in which it is being used does not impact the process description	
Support for defining and using/reusing process assets	
Capability for incremental evolution	
Configuration management of models during development	
Simulation:	
Ability to simulate:	
Concurrency	
Asynchronous processes	
Distributed development (i.e., shared information is not assumed)	
Human interaction	
Integration of software development tools	
Iteration	
Timing of process steps	
Simulation capabilities are directly integrated with model representation	
Interactive simulation/animation capabilities*	
Batch simulation capabilities provide a detailed trace	
Capability to make predictions regarding the effects of change:	
Qualitative: Looking at the behavior and reactions of the process to various events and circumstances	
Quantitative: Prediction of numerical outcomes (time-to-completion, manpower requirements, quality measures)	
Analysis support	
Level of tool support available:	
Tool provides modeling definition/help on-line	
Tool support for method; limited integration to other process tools	
Tool is available as part of a process-supported SEE	
Tool is available as part of a process-driven SEE	

* Interactive simulation/animation capabilities such as the following: Simulation can be started from any valid state of the process model; changes can be specified, and events be generated to emulate external influences or internal changes; user can step through the simulation one step at a time.

D.4 Enactment

Process enactment technologies are defined as those technologies that will support the execution of a software process definition. By this we mean any technologies which will allow the process definition to be "installed" in the sense that the enacted definition will manage and/or control the process, and capture measurements as the development process proceeds.

Processes may be enacted by SEEs, tools, and/or humans. In each case, issues of visibility, status, control and resource management must be considered when judging the suitability of a particular enactment technology.

During enactment, visibility to product information as well as to process information is necessary. An executing process will need access to product data, history, or state in order to accomplish its goal. Project members will also require appropriate access to product information as well as information regarding the status of a project. Management of process state and history data is necessary at least to the level that will permit management reporting on the state of process and project activities. State monitoring and event management will also permit appropriate actions to be taken in response to process state.

In addition, numerous control services must be provided to permit such functions as history/metrics collection, auditing and accounting, scheduling, configuration management, quality assurance, and policy enforcement.

Finally, resource management services must be provided. In a software development project, the people involved typically have roles to play throughout the project. However, a person may have many roles during a project and the roles may change through the project's life span. Process resource management services handle information about people and roles, and the relationship between them.

The Enactment Technology Checklist is provided to assist in assessing the level of support provided by enactment technologies and environments currently available.

Enactment Technology Checklist

Enactment Technology Checklist	
Enactment approach:	
Process-Control*	
Process-Support**	
Execution environment:	
Contains knowledge of the process	
Provides support for tailored project-specific processes	
Supports network-based collaborative development	
Supports multiple platforms	
Enactment tool/technology is:	
Hierarchical	
Intuitive to both user and the process implementor	
Maintainable	
Process execution drives:	
The creation of a plan	
The scheduling of resource usage consistent with the execution environment	
Process visibility and scoping:	
Access to information provided where appropriate	
Visibility to information precluded where appropriate	
Ability to associate control access rights with designated operations	
Process state:	
Work context memory	
Status is attached to each task or action in the enacted software process definition***	
Access to the process' current status	
Access to process definition information	
Access to the state of work products	

* Involves action items with pre- and post-conditions that must be satisfied (control points); completion of one action item creates another action item that must be responded to. Approach emphasizes control goals.

** Monitoring activities in the environment allow the user to have an appropriate current view of the development for selecting the next activity from choices which are appropriate. Approach emphasizes support.

*** Such as no status, allocated, ready, active, stopped, broken, done, not available for execution.

Enactment Technology Checklist	
Process control:	
Workflow	
Process metrics collection service	
History collection	
Policy enforcement	
Query capability	
Process resource management:	
User role management service	
Role definitions	
Level of support:	
Human enactment	
Limited tool support for enactment	
Process-supported SEE	
Process-driven SEE	

Appendix E - IDEF Technology

E.1 Case Study

In Section 1.4, major modeling approaches that can be and have been successfully used for process modeling were presented and discussed. After a careful evaluation of these modeling approaches, the STSC is recommending the use of IDEF0 for software process modeling to its customers. In this section, reasons for that recommendation will be given, as well as an overview of the IDEF family of methods and tool support for IDEF0, IDEF1/IDEF1X, and IDEF3 modeling methods.

The information in Appendix E was collected as part of a case study performed during spring 1992. The purpose of this study was to investigate the current state of process technologies and then choose a promising method based on our process requirements to be used on pilot projects. In this ever changing field, the timeliness and correctness of tool/method information is imperative, but difficult to keep up with. View this information in this appendix intelligently; understand that it does not address every vendor/tool/detail of IDEF. Using this as a building block, remember that it is a good business practice to contact the vendors and get the most current information, always being aware of new technologies/tools/vendors that come about in the process community.

E.1.1 STSC Modeling Technology Requirements

To assist us in keeping sharp focus of the case study goals, certain requirements were defined and reviewed. The following information is taken from the study, and is explained in detail below.

To define the STSC requirements for process definition technology and understand the need for such technology, an understanding of the STSC's customers and their problems is necessary. Studies have shown that almost all software organizations can be characterized as having an initial (SEI CMM Level 1) or repeatable (Level 2) process maturity. Very few organizations, and none of the STSC's customers to date, have mature (Level 3: Defined, Level 4: Managed or Level 5: Optimizing) processes. A variety of technological and managerial solutions are needed to address the needs of these STSC customer organizations. Process definition technology is one of these technologies.

The primary STSC functional requirement for a process definition method is that it must adequately capture and document a software process for the purpose for which the model is being used. There must be sufficient formality and rigor in the method to allow processes to be modeled faithfully. Ideally, there should be support for multiple, complementary views of the process. The method should produce a structured definition and provide techniques for composition and decomposition. In addition, technical growth aspects that are desirable include support for incremental stages of process building, tool support for simulation, ability to be used for enactment of the software process, and the existence of well-defined analysis techniques that can be applied to the model.

Process definition in level 1 and 2 organizations will be used primarily to identify process improvement opportunities, facilitate training of team members in the organization's standard software process, and facilitate communication about processes throughout the organization. Thus, a fundamental requirement for the method is its understandability. If the method and its notations are going to be used for communication, often between people of different technical levels, the notations must be easily understood. The method and notation must also have a fast learning curve. Additional requirements are effective tool support and a high probability of the method's acceptance in the software community. This likelihood of acceptance should be based on the usability and effectiveness of the method, the existence of a support base for the methods (e.g., users groups, newsletters, bulletin boards, etc.), familiarity with similar methods, and/or acceptance of the method as an emerging standard. A tested, widely-used approach with a good track record is most desirable.

E.1.2 IDEF0 Method Recommendation

One of the goals of the case study was to target a methodology as the best fit at that time given our requirements. The results of this goal are discussed below.

In Section 1.4, the strengths and weaknesses of major modeling approaches were discussed. It is important to note that it would be premature to point to a particular process definition technology as the one correct way to define the software process. There is no clear consensus about which method is best, possibly because either none of the existing techniques is ideally suited to the problem or the problem is so broad that no one technique can be ideally suited. Therefore, any method recommendation has to be made with the

caveat that research and development in this area continues and significantly more suitable techniques may be on the horizon.

A summary of evaluation results mapped against STSC requirements appears in Table E-1. Certain categories (Process Modeling Track Record, Likelihood of Acceptance for Software Process Modeling, Modeler Learning Curve and Reader Understandability) were rated on a subjective scale after surveying the field. They are scored on a scale of 0 to 4, with 0 being worst and 4 best. "Yes/r" in the tool support column of the matrix indicates the situation where tool support consists mostly of research prototypes and commercial tools do not exist.

No method can be recommended as the one suitable method for all process definition. However, after studying the available methods with regard to the requirements of the STSC, structured graphics methods were recommended because of their familiarity, understandability, maturity, available tool support, and widespread use in the industry for process definition. The graphical hierarchical representation and data-flow-like diagrams offer the readability of data flow diagrams for software process modeling, thus facilitating communication and understanding. In addition, these methods have been successfully used

	Capture and Document SW Process?	Tool Support	Can be Used to Support Computer Enactment	Tool Support for Simulation	Analysis Methods	Reader Understandability*	Modeler Learning Curve*	Likelihood of Acceptance for Software Process Modeling*	Process Modeling Track Record[53]*
Structured Analysis	Yes	Yes	No	No	Some	4	4	3	3
IDEF0	Yes	Yes	No	No	Some	4	4	4	4
Real Time Structured Analysis	Yes	Yes	No	Yes	Some	2	3	3	2
Process Programming	Yes	Yes/r	Yes	Yes	Some	1	2	1	2
System Dynamics	Yes	Yes	No	Yes	Some	2	3	2	3
Petri Nets	Yes	Yes	Yes	Yes	Yes	0	0	0	3
Rule Based	Yes	Yes	Yes	Yes	Little	1	1	1	2
Object Oriented	Yes	No	No	No	Some	2	2	2	1
Entity Process Modeling	Yes	Yes	No	Yes	Yes	2	2	3	2

Table E-1. Modeling Method Survey Results

54. This is a subjective criteria and is scored on a scale of 0 to 4, with 0 being worst and 4 best.

and are in widespread use by the process definition community. Three major variants of structured graphics - structured analysis methods[54], structured analysis methods with real time extensions,[55] and IDEF0 - were considered.

The IDEF0 functional modeling approach was chosen for a number of reasons. IDEF0 was designed for process modeling, whereas other structured analysis techniques grew out of analysis techniques used for software development. The standardized graphical syntax used in IDEF0 diagrams (as contrasted with free form data flow diagrams) makes IDEF0 diagrams easier for the reader to follow. IDEF0 provides excellent activity decomposition and hierarchical structure. The IDEF0 method also provides the ability to build a large model using multiple sub-models. IDEF0 features – such as feedback arrows to indicate the need for rework, control arrows, mini-specs associated with the diagrams at all levels, and IDEF0 configuration management – also make IDEF0 superior for process modeling. Mechanism and control arrows in the IDEF0 diagram allow the modeler to show who or what performs a given activity. These arrows also allow the separation of input from control. Tool availability of IDEF0, as with all structured analysis methods, was considered mature. IDEF0 tools are available on several platforms and several tools allow platform interoperability.

Not only do IDEF0 models form the basis on which to build the requirements for improvement, but IDEF0 can also be used as a training tool, can be used for process optimization, and can be used for problem solving. Using IDEF0 graphics as a thinking tool has the potential to provide a powerful and cost-efficient way to improve the software development and maintenance process. Experienced users of IDEF0 praise the method for producing easy-to-understand, structured information. IDEF0 diagrams are far easier to understand than bubble-type data flow diagrams with unstructured mazes of arrows. In IDEF0 diagrams, each page has a definite structure. However, diagrams produced by IDEF0 have fairly sparse information content; they are vague about concurrency details, conflict for resources, rules for feedback, and timing. As a result, models must be extended to allow simulation or enactment. To get a usable, transferable model that is understandable to someone without firm domain knowledge, it is necessary to supplement the basic IDEF0

---

54 DeMarco, T., "Structured Analysis and System Specification," Yourdon Press, New York, 1978.

55 Hatley, D.J., and Pirbhai, I.A., "Strategies for Real-Time System Specification," Dorset House, 1987.

diagrams with significant information. IDEF0 is flexible enough to allow this, but it is necessary to have knowledge of what information needs to be added and have the discipline to add this necessary information.

IDEF0 has seen extensive use worldwide. It has been used in such areas as hardware and software development, telecommunications, manufacturing, command and control, and real-time banking. IDEF0 has been used extensively in the commercial marketplace for business process redesign. It is also extensively used by the government market for business case analysis. The DoD Corporate Information Management Program (CIM) is using IDEF0. In addition to the DoD CIM program, IDEF0 is a functional modeling standard for the U.S. Air Force manufacturing programs (MANTECH), the CALS Concurrent Engineering Initiative, the DoD Industrial Modernization Incentive Program (IMIP), and the Advanced Manufacturing Program (CIM-OSA). With the recent interest in software process technologies and software process improvement, IDEF0 methods have seen increased use in the software sector as well. The widespread use and understanding of IDEF0 diagrams and mature tool support available make IDEF0 a good choice, particularly for organizations being introduced for the first time to modeling technology.

E.1.3 IDEF Family of Methods

IDEF is a set of methods and notations, accepted as standards by the IDEF Users Group (representing both government and commercial interests) and maintained by the Air Force. Over time, numerous IDEF graphical techniques have been developed. The IDEF family of methods has been designed to support system description, modeling, and simulation. IDEF methods are called a "family" of methods because they are all formulated to support a common objective: the analysis and design of complex systems. Having multiple methods in the family is intended to supply special-purpose methods with applications limited to specific problem types rather than a super method that attempts to address all problems.[56]

---

[56] Much of the material about the IDEF family in Section E.1.3 comes from the following source: Mayer, R.J., Painter, M.K., deWitte, P.S., "IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications," Knowledge Based Systems, Inc., 1992.

IDEF0, the most mature and widely used of the IDEF methods, was derived from the Structured Analysis Design Technique (SADT) method and supports the functional modeling of a system. While newer IDEF methods (IDEF1, IDEF1X, IDEF2, IDEF3, IDEF4, IDEF5, ---) have each been developed to address known and obvious voids in existing methods, the decision was made to develop a family of methods rather than to re-formulate or revise existing methods. Each method can be highly effective for system description from a particular viewpoint. One method is not right and another wrong. Later methods, such as IDEF3, are not intended as replacements for earlier IDEF methods. The methods are additive, complementary. Therefore, the question is not, "What method is best?" but, "What method is most directly pertinent to the task at hand?" It is often useful to utilize more than one IDEF method to get the deepest understanding of a process - for example, IDEF0 for the functional view, IDEF1/IDEF1X for data modeling.

IDEF0 is widely used for functional modeling, concentrating on activities performed, and providing a static description of a system. IDEF1 concentrates on data modeling and is used primarily for information modeling, business data modeling, and relating data records hierarchically. IDEF1X, an extension to IDEF1, is used to support database design. IDEF2 is a simulation IDEF that is now largely out of use; it has been recommended, though not approved, as a standard. There is little tool support for IDEF2; however, the whole discussion of a simulation IDEF has led some vendors to add vendor-specific, non-standard simulation notations, methods, and capabilities to their own toolset. IDEF3 is a process description method that aids knowledge acquisition by providing a method and representation medium to support the capture, storage, and manipulation of real-world descriptions of a system in an event-model style. IDEF3 shows how activities are time-related or sequentially-related. The behavioral aspects of an existing or proposed system can be captured. The main description mechanism is the process flow description diagram, which describes an ordered sequence of events or activities. To supplement process flow diagrams, major objects in the system are described using object state transition network diagrams that summarize the allowable transitions of an object. IDEF3 has not yet been accepted as a standard by the Air Force or by the IDEF Users Group. IDEF3 tool support is relatively limited. IDEF4, IDEF5, et al. are little used and not standardized; many of these additional methods are manufacturing-oriented and domain-specific. The more standardized IDEF0 and IDEF1/IDEF1X appear to be best suited for software process definition and are the IDEF methods that a large number of process modelers in industry use. Consequently, when evaluating IDEF toolsets, we have primarily concentrated on toolsets that support IDEF0 and IDEF1/1X.

E.1.4 IDEF Family Method Integration

As the case study progressed, customers often showed interest in the integration capabilities between the different areas of the IDEF technology. For this reason, we spent some time investigating this, and the results follow.

Although the IDEF technology is extremely useful and has a wide range of applications, it is our opinion that some aspects of the tools supporting the methodology remain immature. One such aspect concerns the data dictionary. A data dictionary defines all information quantities in an IDEF model; it contains both definitions and structural descriptions of the objects and activities of the system being studied. Unfortunately, IDEF toolsets do not have a good data dictionary compared to data dictionaries supplied by the average structured analysis toolset. In general, vendors have developed varying levels of data dictionary/repository support for their tools. Without question, an increased level of integrity and integration support is needed.

When developing large-scale models in IDEF, it often becomes necessary to use more than one IDEF methodology. The different flavors of the IDEF methodology allow the modeler to represent the system from different perspectives. In investigating the issue of sharing records/types in a data dictionary between tools supporting the IDEF0 and IDEF1X methodologies, several problems were encountered. Most importantly, there is no standard method of integrating or hand-shaking data between IDEF0 and IDEF1X tools. The level of interfacing between the two is left to the discretion of tool vendors. As a result, users wishing to have a seamless sharing of data dictionary information often find that a large amount of additional effort is necessary to achieve this. Increased integrity in the data dictionary is also needed. When records or types are used in several places and across methodologies (IDEF0 and IDEF1X for instance), any modifications to these records or types should be reflected in all instances. Being forced to manually modify all instances of a record or type conflicts with the whole purpose of types – to provide a one-point source of change across the board.

A third related problem involves the availability of records/types already defined by the user. IDEF tools do not support the use of a common data dictionary when the modeler uses more than one IDEF method. After modeling in IDEF0 and creating records in the data dictionary, it seems logical that, when you begin creating other IDEF models, you should be able to make use of the types you have already put in the database. Some tools

require you to enter these types into the data dictionary a second time when it is to be used in a second IDEF method. In addition, there often is no link between these identical types across the different methodologies. Because of this, the diagrams are harder to maintain and may often not be accurate.

Tool vendors need to consider a more open-systems approach to their IDEF tools. An open-systems approach would lead to increased productivity when incorporating a data dictionary into IDEF models, and encourage more non-users to begin using these tools the data becomes useful on several levels.

A summary of tool vendor support for tool integration, particularly in providing links between IDEF0 and IDEF1X tools, follows. The matrix indicates whether or not a tool supports IDEF0 and/or IDEF1X, provides information about links between IDEF0 and IDEF1X tools, hardware platform information, cost information, vendor contact information, and notes on integration capabilities.

TOOL IDEF0 IDEF 1x	LINKS TO IDEF0 & IDEF1X TOOLS --- WINDOWS 386 CLASS MACHINE	$$ --- VENDOR CONTACT	NOTES
AI0 **IDEF0** √ **IDEF 1x**	ASCII export is currently the only link to other tools. --- DOS, Windows	DOS: $1695 / copy Windows: $2995/copy Discounts for volume sales --- David N. Rice Project Mngr - IDEF tools Knowledge Based Systems International One KBSI Place 1408 University Drive East College Station, TX 77840 phone:(409)260-5274 fax: (409)260-1965	KBSI has a wide range of IDEF tools, including those supporting IDEF3.
AI1X **IDEF0** **IDEF 1x** √	ASCII export is currently the only link to other tools. --- Windows	$5000/copy Discounts for volume sales --- David N. Rice Project Mngr - IDEF tools One KBSI Place 1408 University Drive East College Station, TX 77840 phone:(409)260-5274 fax: (409)260-1965	This tool supports both IDEF1 and IDEF1X.

Authormate **IDEF0** √ **IDEF 1x**	ASCII export is currently the only link to other tools. --- DOS (can still be run in Windows environments)	$2500 / copy $2000/copy for 5+copies $1500/copy for 25+ copies --- Al Irvine Eclectic Solutions Corp. 5580 La Jolla Boulevard Suite 130 La Jolla, CA 92037-7692 phone: (619)454-5781 fax: (619)459-3025	At the Spring '93 IDEF conference, the vendors announced a link between Authormate and Evergreen CASE Tools' IDEF1X tool EasyCASE. In this link, Authormate processes their IDEF0 models and identify which types are likely candidates to be included in an IDEF1X model.
AutoSADT **IDEF0** √ **IDEF 1x**	ASCII export is currently the only link to other tools. --- Macintosh, PC Windows	$495 --- Doug Bernard TRIUNE Software 2900 Presidential Dr. Ste 240 Fairborn, OH 45324 ph: (513)427-9900 fax: (513)427-9964	Mac and PC Windows tool that is very affordable.
Design/IDEF **IDEF0** √ **IDEF 1x** √	No common link in the data dictionary between the IDEF 0 and IDEF1X sections of this tool. --- Windows, Macintosh, Sun SPARC, HP Workstations	$4000/1-4 copies, discounts for volume sales. --- Robert L. Seltzer Meta Software Corp. 125 CambridgePark Drive Cambridge, MA 02140 phone: (617)576-1203 fax: (617)661-2008	The same data dictionary is used for IDEF0 and IDEF1X, but no link between the two methodologies is apparent, even with the common database.
BPwin **IDEF0** √ **IDEF 1x**	Supports DDE (Dynamic Data Exchange) --- PC w/ Windows	$1995 / copy --- Ronnette Kelly Logic Works, Inc 1060 Route 206 Princeton, NJ 08540 Phone: (609)252-1177 FAX: (609)2521175 Compuserve: 70262,1135@compuserve.com	Supports IDEF0 and allows the user to capture all information necessary to create complete IDEF0 kits. BPwin also supports Activity Based Costing.
ERwin-ERX **IDEF0** **IDEF 1x** √	ASCII export is currently the only link to other tools --- Windows	$2500/copy Discounts for DOD customers and volume sales. --- Jeffrey D. Mershon Manager, Product Support Logic Works, Inc. 1060 Route 206 Princeton, NJ 08540 Phone: (609)252-1177 FAX: (609)2521175 Compuserve: 70262,1135@compuserve.com	Entity Relationship database design tool that help the user quickly design and build databases using an intuitive graphical approach.

IDEF Leverage **IDEF0** √ **IDEF 1x** √	TBD --- IBM Mainframe, VAX Workstations	IBM $45K-$65K VAX $10K-$30K --- B. Neil Snodgrass Senior Vice President D. Appleton Company, Inc. 222 W. Las Colinas Blvd. Suite 1141 Irving, TX 75039 phone: (214)869-1066 fax: (214)869-1099	This company is primarily in the business of selling their expertise and consulting, often selling their tools in the process. However, selling tools is not their highest priority.
IDEFine-0 **IDEF0** √ **IDEF 1x**	Wizdom's IDEF Glossary product creates an integrated glossary of each ICOM in the IDEF0 model and the entities and attributes in the IDEF1X model. It provides a reference for the user as to where each term is used. It maintains consistency across models. --- DOS (can be run in Windows environment)	$5000/copy (including glossary). Discounts for volume sales on any of Wizdom's tools --- Kristy Hanna Wizdom Systems 1300 Iroquois Avenue Naperville, IL 60563 phone:(708)357-3000 fax: (708)357-3059	Windows version available
IDEFine-1x **IDEF0** **IDEF 1x** √	Wizdom's IDEF Glossary product creates an integrated glossary of each arrow/connection in the IDEF0 model and the entities and attributes in the IDEF1X model. It provides a reference for the user as to where each term is used. It maintains consistency across models. --- DOS(can be run in windows environment)	$3500/copy. Discounts when buying more of the same or different Wizdom Systems tools --- Kristy Hanna Wizdom Systems 1300 Iroquois Avenue Naperville, IL 60563 phone:(708)357-3000 fax: (708)357-3059	Windows version available
ModelPro **IDEF0** **IDEF 1x** √	ASCII export is currently the only link to other tools. --- Windows, Macintosh	$1200/copy --- B. Neil Snodgrass Senior Vice President D. Appleton Company, Inc. 222 W. Las Colinas Blvd. Suite 1141 Irving, TX 75039 phone: (214)869-1066 fax: (214)869-1099	A common repository effort is under development. This company is primarily in the business of selling their expertise and consulting, often selling their tools in the process. However, selling tools is not their highest priority.

E.2 IDEF Tool Survey

Special emphasis has been given in the modeling tool list to IDEF tools, given the selection for STSC use. This section summarizes the primary requirements for tools supporting IDEF from the STSC's perspective and provides supplementary material on available IDEF tools.

The basic STSC functional requirement for an IDEF process definition tool is support for IDEF0. Support for data modeling through IDEF1 or IDEF1X support is important but is not initially crucial for STSC efforts. Secondly, the ideal IDEF modeling tool should non intrusively support the modeling activity. If an IDEF tool is going to be used by the modeler as a thinking tool, it must be easy to use – the modeler should not have to change the processes in order to accommodate the tool. And, finally, the STSC will need to support process definition on a number of different desktop platforms. The PC architecture must be supported at the STSC. However, the STSC cannot place platform requirements on its customers; therefore, as a further requirement, a wide range of commonly available desktop platforms must be supported by the STSC's process definition tools. This requirement may be met by a suite of tools with some form of bridging built between them.

Information on available IDEF tools was gathered from the following sources: the IDEF Users Group,[57] CASEBase,[58] RAD'92,[59] Tool Finder/Plus,[60] and personal knowledge. The following IDEF tools were identified: Design/IDEF from Meta Software Corporation, AI0, AI1, and ProCAP from Knowledge Based Systems, Inc., ModelPro from D. Appleton, IDEFine from Wizdom Systems, Inc., Authormate from Eclectic Solutions Corp., AutoSADT from TRIUNE Software Inc., Erwin and BPwin from Logic Works, and SIMprocess (a simulation tool for Wizdom Systems with no IDEF modeling capability) from CACI. Tool survey information is summarized in Table E-2.

---

[57] The IDEF Users Group is a forum for the sharing of information and experience on issues related to IDEF and associated methods. Conferences are held quarterly. The conference attended in May 1993 featured tutorials, product and service displays, conference sessions, and workshops, and provided much information on the current state of IDEF modeling and tools.

[58] CASEBase is a product of P-Cube Corporation, Brea, CA; Copyright 1990,'91'92: "CASE Product Comparison Information."

[59] Requirements Analysis and Design Tools Report, Software Technology Support Center, Ogden ALC/TISE, Hill AFB, Utah, April 1992.

[60] Tool Finder/Plus is a PC-based, CASE tool database with automated search capabilities available from the C/A/S/E/ Consulting Group, Lake Oswego, Oregon.

	IDEF0 Support	IDEF1 or IDEF1X Support	Platforms
Design/IDEF	Yes	Some	IBM PC, Mac, and SPARC (interchangeable)
AI0, AI1, AI1X	Yes	Yes	IBM PC
IDEF Leverage	Yes	Yes	IBM Mainframe; VAX
IDEFine-0 & 1X	Yes	Yes	IBM PC, Sun
Authormate	Yes	No	IBM PC and VAX
AutoSADT	Yes	No	Macintosh
BPwin	Yes	No	IBM PC
ERwin	No	Yes	IBM/ Macintosh
SimProcess	No	No	IBM PC, Macintosh, UNIX

Table E-2. IDEF Tool Survey Results

E.3 IDEF Training

Many of the STSC's customers were using, or were interested in using, the IDEF technology. We thought it would be extremely important to aid them in finding proper training for the different levels of IDEF usage that exist. All too often tools are bought and never used because proper training was never acquired. We had a goal of assisting organizations learn about process tools, but additionally give them information on where to look for training to get them up and working in the targeted methodology area.

When considering the IDEF technology for a software process modeling group, there are several crucial issues that need to be addressed. These issues include understanding what you expect from IDEF, researching IDEF vendors for a tool that fits your needs, and researching IDEF training sources. Training should be considered of extreme importance because a modeling group may have an exceptional tool and well thought out goals, but, without proper training, IDEF modelers will be substantially less effective. It takes proper training and an iterative review process for useful and correct models to be produced. How

roles should be distributed in a modeling group is discussed in this section. Different categories of IDEF training according to modeling roles will be presented, as well as guidelines for what is expected of people in these training categories.

There are several different types of IDEF training. The IDEF Users Group defines specific training categories which correspond to group member roles in an IDEF modeling group. The following is an outline which summarizes responsibilities and necessary levels of training for each role:

IDEF Authors

- Create diagrams and interview experts.
- Make comments to other authors about IDEF practice, alternate decomposition, diagram changes, and analysis principles.
- Make comments to other authors about technical accuracy of a specific application.
- Require an IDEF author's course and project workshop.

IDEF Reviewers

- Read diagrams for content and technical accuracy of material.
- Make comments to author on accuracy of diagram, appropriateness of the decomposition, correctness of the interfaces, and diagram quality.
- Require both an IDEF reader and an IDEF reviewer course and project workshop.

IDEF Readers

- Read diagrams for content.
- Make informal remarks on diagrams.
- Require an IDEF reader course and project workshop.

The philosophy of this type of tiered training is that correct distribution of group roles is crucial. Every member of an IDEF modeling team does not need to be an IDEF author, who has the skill and experience to extract information from documents and experts and the ability to select information to be incorporated into an IDEF model. Instead, it is beneficial to formulate a role scheme, where each role has a defined scope. In this way, each group member can thoroughly concentrate on his/her specific role.

This method of organization brings with it a structure that is difficult to achieve when all group members try to be everything to all people. Assigning different roles to group members yields increased objectivity and heightens their effectiveness. For example, when the roles are distributed, the reviewers have a level of separation from the authors, and are therefore more objective.

Every member of the modeling group, including managers, should enroll in a reader's course. The length of this course is typically one or two hours. Knowledge gained from such a workshop will enhance communications within and outside of the modeling group, while allowing all participants to benefit from the work done.

IDEF modeling, performed by the IDEF authors, is almost always more than a one-person job and is often effective when performed by a small core of workers rather than a large group. These authors will have the heaviest up-front work load. Members of the author's team will be interviewing experts, reviewing documents, and examining information, ultimately deciding what parts should be incorporated into the models. A comprehensive course for authors will usually take about one week.

The reviewer group is often the same size as the author's group, or marginally larger. IDEF reviewers should take an additional course specifically focusing on critiquing models after taking the IDEF readers course with the rest of the group. A reviewer course provides reviewers with extra insight into IDEF modeling. A course concentrating on IDEF model review will usually be about one or two hours in addition to the reader's course.

While recognizing that the production of correct and useful models should be a result of an entire team effort, some practitioners feel that the reader's course, taken in isolation from the concepts taught in an author's course, is not really very helpful. They ask

the question, "Why not let everyone take the author's course?" In this view, even if the students will be basically a reader of models, the extra information available in an author's course will enable them to "read" an IDEF model with more understanding.

Whichever philosophy you subscribe to, when evaluating IDEF courses, it is important to note that IDEF0 is taught by a small number of teachers, a number of whom frequently teach for multiple vendors. It is important to choose instructors who have actually used SADT and/or IDEF0 on major projects rather than instructors with limited experience applying IDEF0. You want your instructor to have that major, hands-on experience.

An IDEF0 course training matrix is provided in section E.3.1 with a representative list of IDEF0 courses. For each set of courses, length of course, use of process examples, use of tools in the course, and the willingness to customize the course is indicated. More complete information on each training course is provided in IDEF Training Information Forms in Volume II of this report.

E.3.1 IDEF Training Matrix

	Use SW Process Examples	Willing to use tools in Course	Willing to Customize Course
D. Appleton	-	√	√
Eclectic Solutions	-4	√5	√
Knowledge Based Systems Inc.	√	√	√
Lipka SW Engineering. Inc.	√	√	√2
Meta Software Corporation 3	-	-	-
New England Business Consultants	-	√6	√
Productivity Solutions	-	-1	-
Wizdom Systems, Inc.	-	√	√

Table E-3. IDEF Training Matrix

1 - Productivity Solutions (John Stockenburg) has never taught the IDEF0 author's course with tools. He usually makes use of paper and pencil. He's familiar with a number of tools but the customer would have to provide computers, software, etc.

2 - Steve Lipka's course is very tailored to the customer's needs. The last day in his course is spent on the client's case problem.

3 - Meta Software offers consulting services, but is not in the IDEF training business. They spoke highly of a few particular consultants that offer training services and are familiar with Meta's Design/IDEF tool.

4 - Eclectic is really a business process company, but do have a SW process example/case they can use.

5 - Eclectic is willing to use various tools, but are sensitive to the fact that, because they are familiar with their own tool more than others, they don't know what impact other tools will have on the course. Will it be a burden to students? Is it robust? etc.

6 - NEBC will use tools if that is desired, but are concerned that the tools may get in the way. They advise learning the methodology first, with tool training later.

E.3.2 IDEF Training Information Forms (See Volume II)

The IDEF Training Information Sheets appear in Volume II of this report. For information on how to order Volume II of this report, please contact the STSC customer service department at (801)777-7703 or DSN 458-7703, fax to (801)777-8069 or DSN 458-8069, or email to godfreys@wpo.hill.af.mil.

Volume II contains IDEF Training Information on the following companies:

- D. Appleton Company, Inc.
- Eclectic Solutions
- Knowledge Based Systems, Inc.
- Lipka Software Engineering, Inc.
- Meta Software Corporation
- New England Business Consultants
- Productivity Solutions
- Wizdom Systems

(This page is intentionally left blank)

Appendix F - Enactment Technology

Our analysis identified three mechanisms for achieving computer enactment:

1) An environment framework that provides a set of basic services for environment construction can be used as the architectural basis of an enactment environment.
2) Vendors can provide customizable environments that provide a fixed process; in such environments, the user can often make small modifications of particular parts of the process.
3) Using a process model of the process, an environment can be built up "from scratch" to support the model.

The separation between these three approaches is not always clear-cut, and there is a great deal of variation in examples in a specific category. Nevertheless, keeping the three basic approaches in mind helps us organize the great variety of work being done in this area. These approaches will be further explained in the following sections, supplemented with examples of products implementing each approach.

F.1 Frameworks Supporting Enactment

Frameworks generally provide low-level services (e.g., data modeling, control and user-interface support) that are needed in virtually all software production environments. Some are also tool integration platforms. Examples of some frameworks are: Atherton Technology's Software BackPlane, CAIS-A (Common APSE Interface Set), PCTE (Portable Common Tool Environment), SLCSE (Software Life-Cycle Support Environment), DSF (Distributed System Factory), and Hewlett Packard's SoftBench.

Atherton Software BackPlane is an open-systems object-oriented software development repository that, because it is object-oriented, can store not only the data that is created by a tool, but also the software engineering process as well. The BackPlane provides repository services – version control, configuration management, consistent user environment and interface, context management, access control – and allows user customizations and extendibility. A new Atherton product, called Integration SoftBoard, will allow the type hierarchy to be extended to include the behavior of any software engineering process written in any programming language, thereby enabling a user to automate established development rules and procedures or encapsulate rules and procedures that have previously been automated. Atherton BackPlane has been used by organizations to build SEEs to integrate

CASE tools from multiple vendors. It has been used for process enactment by Loral Corporation to define and then enact a standard software development process; BackPlane was used to provide a common medium for tool control and interchange; the environment built supported the multiple computing platforms used by Loral's divisions.

CAIS-A (under control of the Ada Joint Program Office (AJPO)) is a set of Ada interfaces designed to act as a high-level virtual operating system, providing an object management system on which process control can be based. The PCTE (Portable Common Tool Environment) is a European standard that is seeing increased use as a basis for integrating tools. SLCSE provides a framework, a common database, a set of common schemata, and tools designed to support DoD-STD-2167A. Distributed Software Factory (DSF) provides a framework for a distributed wide area network.

Hewlett Packard's SoftBench, based on the UNIX system, supporting X-Windows, is a tool integration platform upon which a custom development environment can be built. The basic framework includes a comprehensive set of integrated program construction tools, most connected with the development of code (static analyzer, automatic makefile generator, compiler, linker, builder, dependency graph generator, editors, debuggers, etc.) Other tools spanning the full CASE life cycle have been integrated into the environment by third-party vendors and by HP, and are available for an extra cost. A new HP product called SynerVision is advertised as a "process engine." Customizable process templates are used to define the architecture and the individual steps of specific processes. A user can automate any process by adding templates and their own tools or tools of a third party. A turnkey change request management environment called ChangeVision, built using SynerVision and sold by HP, combines metrics, reporting capabilities, and an interface to a change request tracking system.

F.2 Customizable Enactment Environments

Customizable environments are based on the existence of a high-level, fixed process and a core of services or capabilities. The construction method allows the user to specify a set of extensions or changes to the core capabilities. This specification is then combined with the core capabilities to form an environment. This set of approaches provides an environment that is easier to instantiate than frameworks but less general than those that can be created using frameworks. However, the line separating the two is not always clear.

The criteria used in this report to separate the two approaches is that frameworks, though providing a variety of services, do not support any particular process; customizable environments do support specific processes.

Much of the work in customizable environments comes from two specific domains: information systems development and configuration management. The information systems world seems to be well ahead of other software development application areas in the process technology area, particularly in the area of computer enactment. Process technologies developed for this application area are also advertised as supporting other software application areas (technical, real-time, etc.) as well. This support is largely untested. Often, when customizable environments are hard-wired to a specific vendor process, the likelihood that it will be useful for other application domains is rather doubtful. Even though some tailoring of the process is allowed, it is usually minimal. For the amount of tailoring that would be necessary to make an information systems environment applicable to another domain, it would probably be more straightforward, and the end result more useful, to use a different approach to building an enactment environment. Nevertheless, some of these process-specific enactment environments have been included in the technologies represented in this report to support information-systems-type applications. The other specific area with strong support for computer enactment is configuration management. A number of turnkey configuration management systems are available commercially. These cannot be considered to be enactment environments; however, some legitimate computer enactment has resulted from configuration management toolsets which have allowed process definition to be added and supported as the process itself becomes configuration controlled.

Some examples of customizable environments are EAST (the Environment of Advanced Software Technology), CaseWare/CM, Summit Process, firstCASE, and Navigator System Series. In the EAST environment, built on an implementation of PCTE, management is a priority. The environment supplies a set of management and development tools, and provides complete configuration management. EAST furnishes standard process models (IEEE Standard 1002-1987, ESA Software Engineering Standard, DoD-STD-2167A). Models can be refined and adapted to the tasks of a specific project.

Summit Process incorporates a hypertext front end and provides a fully automated process for the entire information systems life cycle. Though it is said that Summit Process can be used to support any process, it is most suitable for use in building information

systems. Another information systems environment, firstCASE, is based on the vendor's automated (customizable) systems methodology. Navigator System Series is a third information systems turnkey environment; in this instance, customers must buy into Navigator's methodology for building information systems; the only tailoring that is allowed permits a user to attach guidelines to process blocks. A number of other information-systems-oriented enactment environments are also available commercially.

CaseWare/CM, targeted mainly to the UNIX market, is advertised as a configuration management system that "provides complete control over development and maintenance activities." Support for tool integration, configuration management, and problem tracking is already built in. The system can be customized to implement, automate, and enforce unique organizational development processes. The desired software process must be specified in the system using a fourth generation language (4GL) process modeling language called ACcent so that the system can provide automated support of the process. Turnkey systems are available – CaseWare/CM has integrated their toolset with FrameMaker, SoftBench, Saber-C, Saber-C++, and the Alsys Ada environment; plans to integrate Process Weaver (see Section F.3) are being made. The CaseWare/CM system can be used to support the modeling of project/organization-specific process and project-specific tools. Although the tool was developed by building capabilities onto a basic configuration management tool, CaseWare/CM has begun to approach a basic process modeling/enactment capability. A number of customers who have chosen this type of system are basically satisfied with the resulting environment and feel that such a system provides the level of computer enactment support that can realistically be achieved commercially at this time.

F.3 Process-Driven Enactment Approaches

When a software process model or process language is used to generate the environment, the process itself is of primary importance. Some of the products used to implement this approach are: EUREKA Software Factory (ESF), MARVEL, IPSE2.5, MELMAC, Virtual Software Factory (VSF), KI-Shell, and Process Weaver. ESF uses process models as the basis of tool integration; this research effort is investigating a number of different schemes and modeling formalisms for use in enacting the process. MARVEL uses process definitions using pre-condition/action/post-condition formalisms. In IPSE2.5, a particular execution of the process models provides the environment to the users; a process model execution facility serves as the means of invoking tools and presenting choices to

environment users. MELMAC is an environment based on the execution of high-level Petri net representations of software process models.

A product called Process Weaver, developed by Cap Gemini in France, allows modeling, instantiation, guidance, and support for interfacing with CASE tools. This product uses a combination of hierarchy of activities, descriptions of activities in terms of input objects, output objects and roles (or types of people) involved in the performance of the activity, and Petri nets to model a process. For each node of the "activity tree," a cooperative procedure is defined that represents the ordering of activities and conditions that trigger the move from one activity to another. Any tool on any supporting platform can be integrated using the concept of a "software bus," rather than using a common database for all tools; the bus requires only that tools provide a programmatic interface that can be accessed via the protocols that the software bus stipulates. After enactment, the user of the system is presented a screen that lists the set of tasks that can be performed. By clicking on a specific task, the Workcontext window of that task will be brought up. By clicking on particular input/output objects, the associated tools are invoked with the appropriate template, file, or information. Metrics (determined by the builder of the model) can be gathered automatically as a project proceeds. This product has generated a lot of interest in the process technology community. At this point, it has only been used to enact small portions of practice areas – for example, to model/enact the review cycle for software components. Process Weaver seems to have a lot of promise but needs to be piloted and used operationally before being unconditionally recommended for full life-cycle enactment.

KI-Shell is an object-oriented, workflow-based product for process modeling and computer enactment. To define workflow, a workflow activity model (looking very much like an IDEF0 model providing a functional view of activities to be performed) is used. How each activity is performed is specified by using the C programming language to define procedures (rules) to be executed during activity performance and by calling utilities from the system integration library. KI-Shell directs and manages user activities, tools, and data according to a planned workflow. It can handle complex synchronization and ordering requirements, and complex information structures. Because KI-Shell uses the object-oriented approach, changes are localized to impacted objects. The model can be changed quite rapidly to meet changing user requirements. It has been successfully used to model and enact concurrent engineering, manufacturing/assembly processes, and financial and medical/health processes. KI-Shell operates in a client-server mode – a network of workstations with one

workstation designated as the process server (containing a database such as Oracle to store process descriptions and process instances) and others performing as machines for users with specific roles.

A product with an approach that differs from the others described above is the Virtual Software Factory (VSF), that is really a process-driven CASE tool and CASE tool environment builder. A language called Cantor, based on set theory and predicate logic, is used to define a process model. VSF then enables the process engineer to rapidly establish high-performance, multi-window, menu-driven CASE tool support. They call their approach a "modeling approach" as contrasted with Process Weaver and KI-Shell's "constructive approach" (where COTS tools are the components). The environment uses a common database repository for all tools; because the VSF system actually generates the tools that will be used in the environment, all information in the database is managed at a detailed semantic level. This may be an ideal solution for organizations and projects that have their own design methods or documentation standards to support as it allows the user to establish support for the precise method required and provides the ability to evolve over time. The downside is the necessity to build all tools from scratch, even though the vendor asserts that it is possible to build a basic process-driven environment with all the necessary tools in three to six months. A number of "complete" VSF solutions (which have resulted from VSF working with customers in building environments) are now available.

(This page is intentionally left blank)

Appendix G - Bibliography

This appendix contains articles, books, and presentations that will provide information on process technologies discussed in this report. In order to help the reader find relevant materials more easily, the recommended readings are arranged into categories based on process focus rather than providing all references alphabetically. Within a process category, readings are grouped alphabetically by author - where authors are not known, by title.

The first page is a table of contents for the recommended readings. This is followed by a short list of references recommended to the beginning reader in process technologies for a good overview of the field. The full annotated bibliography follows this short list.

Table of Contents: Recommended Readings

I. Recommended Readings for a Process Technology Overview

Assessment:

Humphrey, Watts, "Managing the Software Process," Addison-Wesley Publishing Company, Inc., 1989.

The definitive book on the SEI's software assessment process. The book describes the SEI software maturity framework, the use of this framework in process assessment and the steps required to initiate effective software process change. The five maturity levels are described (chaotic/ad-hoc, repeatable, defined, managed and optimized) as well as the steps required to move from one level to another.

Bollinger, Terry, B., McGowan, Clement, "A Critical Look at Software Capability Evaluations," IEEE Software, July, 1991.

This article provides a critical view of the government's Software Capability Evaluation program, particularly the use of the SEI capability maturity model by the government to determine whether or not companies are capable for a software job.

The paper argues instead for a global, top-down approach to process optimization that features some type of structured, preferably graphical, method (such as SADT) for recording the details of an organization's existing processes, asserting that the act of doing this will lead to more effective company-specific process improvement.

Dawood, Mark, "It's Time For ISO 9000," CrossTalk, March, 1994.

This article provided an overview of the ISO 9000 series standards, what it takes to become certified, why it's important, who is currently utilizing ISO 9000, and comparing it to SEI's CMM.

Modeling:

Curtis, Bill, Kellner, M.I., Over, J., "Process Modeling," Communications of the ACM, Vol. 35, No. 9, September, 1992.

Article presents an overview of current state-of-the-art in software process modeling. It provides a conceptual framework for discussing software process modeling and carefully defines terms used in the discussion. Topics covered include: Uses for process models, perspectives that a model may provide (functional, behavioral, organizational and informational), and reviews of five representation approaches for process information (process programming, functional approach, plan-based, Petri nets and quantitative models).

Enactment:

Myles, D.T., "Automated Software Process Enactment," Proceedings from The Fifth Annual Software Technology Conference: Software - the Force Multiplier, April 21, 1993.

This paper provides a good introduction to process enactment. It discusses the typical roles for process enactment and lists the necessary elements of a process enactment technology. Myles summarizes the lessons learned by the IBM Federal Systems Company, Houston, Texas; FSC has been actively engaged in automated software process enactment since early 1990. Myles presents principal enactment tool requirements that have grown from this experience and briefly describes their use of the enactment tool, Process Weaver. He emphasizes the need for a disciplined process development methodology before process enactment is attempted. He argues for the concept of "piecemeal enactment" of small segments of the process and stresses the importance of pilot projects to confirm the utility of process enactment.

II. Full Annotated Bibliography

G.1 Process Assessment

Bollinger, Terry, B., McGowan, Clement, "A Critical Look at Software Capability Evaluations," IEEE Software, July, 1991.

This article provides a critical view of the government's Software Capability Evaluation program, particularly the use of the SEI capability maturity model by the government to determine whether or not companies are capable for a software job. The article emphasizes aspects of process assessment that they feel are inadequately addressed in the SEI assessments process - aspects such as taking into account technology issues, confidence in the model used, view of the software process embodied in the model, statistical reliability, rating systems, etc.

Argues that the assessment program (and the Software Capability Evaluation (SCE) program used by the government to determine whether or not companies are capable for a software job) are seriously flawed by their reliance on the SEI's unproved process-maturity model. Criticizes the lack of importance of technology issues on an organization's rating, the artificial ordering of the maturity levels, the model's lack of statistical reliability, its reliance on a small set of yes-no questions, and the assembly-line view of the software process, which emphasizes maintenance, not design. The paper also raises the issue of the model's inhibiting effect on introducing any change - even change that could be beneficial - to a software process once it has reached an acceptable maturity level.

The paper argues instead for a global, top-down approach to process optimization that features some type of structured, preferably graphical, method (such as SADT) for recording the details of an organization's existing processes, asserting that the act of doing this will lead to more effective company-specific process improvement. And finally, it argues that the government should use the proven track records of companies for producing quality software on time and within budget when letting new contracts rather than depending on the SCE program to do the screening.

Humphrey, Watts, "Managing the Software Process," Addison-Wesley Publishing Company, Inc., 1989.

The definitive book on the SEI's software assessment process. The book describes the SEI software maturity framework, the use of this framework in process assessment and the steps required to initiate effective software process change. The five maturity levels are described (chaotic/ad-hoc, repeatable, defined, managed and optimized)) as well as the steps required to move from one level to another.

Humphrey, Watts, Snyder, T.R., Willis, R.R., "Software Process Improvement at Hughes Aircraft," IEEE Software, July 1991.

Lessons learned and return on investment (ROI) for Hughes Aircraft in using the SEI assessment procedure to improve the software process. Includes: SEI recommendations for improving the software process from the 1987 and 1990 assessments; portions of the Hughes action plan to address recommendations; summaries of actions taken.

Humphrey, Watts, and Curtis, Bill, "Comments on 'A Critical Look'," IEEE Software, July 1991.

A rebuttal to many of the points made in the Bollinger article in the same issue of IEEE Software. First, the concerns of the SEI assessment process (to help software organizations improve their own capabilities) and the SCE (to help acquisition groups evaluate suppliers) are quite different. Questionnaire scores alone are not used in SCE evaluations; the SEI instructs SCE auditors not to base their contract-award recommendations on maturity grades, but rather trains them to evaluate an organizations' strengths and weaknesses in eight key areas (project planning, project management, configuration management, quality assurance, standards and procedures, training, process focus and peer reviews and testing) using a clearly defined and widely reviewed method using public, generally recognized criteria. In general, for either assessment or evaluation, the questionnaire cannot be separated from the process by which it is used.

Secondly, asking more questions is not necessarily better; statistical methods are rigorously applied by the SEI to examine the reliability of the questions chosen to identify the key problem areas. (Some additional questions will be added to the next version of the maturity questionnaire to resolve shortcomings identified in the current version.) Thirdly, the SEI feels that the use of the maturity levels to promote an evolutionary improvement approach, supported by the use of maturity levels, is a sound one. And, lastly, the SEI has concluded that specific technologies should not be addressed in this framework until there is a broader consensus on the most effective technologies. The authors comment that the use of technology by low-maturity organizations will probably have limited success. The SEI urges organizations to use technology to address only the problems they truly understand.

McGowan, C.L., and Bohner, S.A., " Model Based Process Assessments," Proceedings of the 15th International Conference on Software Engineering, pp. 202-211, Baltimore, Maryland, May 17-21, 1993.

This paper presents an approach that combines process modeling with process assessments. It describes the creation of an SADT (IDEF0) model of a large software maintenance process and its use as a basis for assessing the process. The model led to process improvements that might have been missed otherwise. The model based process assessment (MBPA) approach is contrasted to the SEI Process Assessment approach and recommended as either a replacement for or adjunct to the SEI approach.

Mosemann, Lloyd K. II , Memorandum "Policy on Software Maturity Assessment Program", September 1991.

This memorandum outlines the Air Force commitment to improving the software acquisition, development, and support process of their software intensive systems. In addition, it states a goal "to achieve a maturity level 3 (defined process) by 1998 for Central Design Activities / Software Design Activities (CDA / SDA) and weapon systems Software Support Activities (SSA)."

Paulk, M., Curtis, B., Chrissis, M.B. et al., "Capability Maturity Model for Software," Technical Report CMU/SEI-91-TR-24, Software Engineering Institute, August 1991.

Using knowledge acquired from software process assessments and extensive feedback from both industry and government, an improved version of the process maturity framework has been produced called the Capability Maturity Model for Software (CMM). This paper is an introduction to the revised model. Specifically, it describes the process maturity framework, the structural additions that compose the CMM, how the CMM is used in practice, and future directions of the CMM. It is hoped that the report will clear up some of the misconceptions associated with the earlier model and questionnaire, particularly the practice of equating the vehicle for exploring process maturity issues, the maturity questionnaire (a simple tool for identifying areas where an organization's software process needed improvement) with the model itself.

Paulk, M., Curtis, B., Chrissis, M.B. et al., "Capability Maturity Model, Version 1.1," IEEE Software, Vol. 10, No. 4, July 1993.

This paper presents an overview of the current version of the CMM. It discusses immaturity versus maturity in an organization, the five maturity levels of the CMM, and the CMM operational definition (i.e., internal structure; key process areas; goals - which are used to determine if an organization or project has effectively implemented a key process area; attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting; and key practices). In addition, it summarizes the differences between CMM Version 1.0 and Version 1.1; in general, the new version has more consistent wording and should be easier to use. The revision is based on more than six years of experience with software-process improvement and the contributions of hundreds of reviewers. Although the CMM is considered a living document that will be improved, The SEI anticipates that CMM Version 1.1 will remain the baseline until at least 1996.

Paulk, M., Humphrey,. W.S., Pandelios, G.J., "Software Process Assessments: Issues and Lessons Learned," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

The software process assessment method developed by the Software Engineering Institute at Carnegie Mellon University is being used by a growing number of US government and industrial software organizations. This paper describes the key organizational issues found by using this assessment method over the past four years and relates them to traditional US industrial practices. Some of the SEI's experiences are described as well as the lessons learned from assessing

over 60 organizations and interviewing approximately 2,300 software professionals and managers.

Weber, C.V., Paulk, M.C., Wise, C.J., Withey, J.V., et al., "Key Practices of the Capability Maturity Model," Technical Report CMU/SEI-91-TR-25, Software Engineering Institute, August 1991.

This report describes the key process areas that correspond to each maturity level in the CMM. Key process areas are building blocks that indicate the areas an organization should focus on to improve its software process and to identify the issues that must be addressed in order to achieve a maturity level. What is meant by maturity at each level is elaborated and a guide that can be used for software process assessments, software capability evaluations, and process improvements is provided.

For each level in the maturity model, key process areas are defined (i.e., requirements management, software project planning, software project tracking and oversight, software subcontract management, software quality assurance and software configuration management for Level 2: Repeatable). Then, for each key process area identified, goals are established and the key practices needed to accomplish these goals specified.

G.2 Process Assets

Gates, L., et al., "Process Definition Advisory Group Workshop Summary Report," Software Engineering Institute Special Report (SEI-91-SR-15), December 1991.

This report is a summary of the October, 1991, Process Definition Advisory Group (PDAG) workshop. The purpose of the PDAG (members drawn from STARS prime contractors, affiliates and academia) is to support the STARS software process definition project. STARS/SEI is identifying, collecting and analyzing existing life-cycle models, process descriptions, procedures, methods and other related process documentation in order to distinguish process component types, attributes, instances and priorities so that component templates can be designed based on these features. The motivation for this task is that making tailorable, adaptable examples of modern experience-tested software processes readily available will facilitate their use. The task will culminate with the initial definition and demonstration of a reuse library for software process, i.e., the Process Asset Library (PAL). The library or repository will contain reusable, tailorable, adaptable, experience-tested, modern software engineering processes. Methods and criteria for composing project-specific processes from components will also be developed.

This workshop discussed long-term and short-term usage scenarios for the library, general architectural concepts in the context of software process and the library, and provided guidance on which process components should initially be collected and installed in the PAL. Components judged to be most useful included support tools, process fragments, examples of defined processes and process notations.

G.3 Process Modeling - Overview and General Articles

Carney, Dave, IDA, "Obstacles to Automating the Software Process," AIAA Computing in Aerospace 8, October, 1991.

"Much, perhaps most, of the technology for automated Process Management is in its infancy. And the core problem itself, in its scope, tractability, etc., is hardly known. So my concern is that expectations, at least expectations of rapid technological developments and rapid dissemination of them, are perhaps too great."

Clough, A.J., "Choosing an Appropriate Modeling Technology," C/A/S/E Outlook, Vol. 7, No. 1, pp. 9-14.

Choosing an appropriate process modeling technology is not a straightforward task. An organization must first determine what questions it wishes to answer by using the model. This paper provides lists of questions from the model reader and model developer's perspectives. Choosing the questions that an organization wishes to answer can help an organization establish priorities for process modeling technologies, since each has strengths for answering certain kinds of questions. Once an organization has determined what questions are important in its development and use of a software process model, a more systematic and directed technology choice is possible.

Curtis, Bill, Kellner, M.I., Over, J., "Process Modeling," Communications of the ACM, Vol. 35, No. 9, September, 1992.

Article presents an overview of current state-of-the-art in software process modeling. It provides a conceptual framework for discussing software process modeling and carefully defines terms used in the discussion. Topics covered include: Uses for process models, perspectives that a model may provide (functional, behavioral, organizational and informational), and reviews of five representation approaches for process information (process programming, functional approach, plan-based, Petri nets and quantitative models). Specific approaches reviewed include APPL/A, HFSP, GRAPPLE, Petri net role interaction model, and system dynamics. Issues in process modeling discussed include: formality, granularity and precision, prescriptive vs. descriptive vs. proscriptive models, multi-paradigm representations. It also discusses the use of models in process improvement and software project management. Process-based software development environments discussed include Arcadia, ISTAR, and MARVEL.

Curtis, W., Krasner, H., Shen, V., and Iscoe, N., "On Building Software Process Models Under the Lamppost," ACM, 1987.

A criticism of most process models as representing a manufacturing orientation that does not really describe how software is developed. The authors'

criticism of process programming points out that procedural descriptions are not likely to assist software engineers in performing their tasks with greater efficiency or accuracy, unless their problem was not knowing what to do next. Such descriptions do not provide managers with greater insight into impending problems. Further dangers of process programming: Procedures may represent idealized processes that may not map accurately to actual development behavior. Secondly, we may automate processes that do not match the way people, and often outstanding people, work. Premature proceduralization may thus actually interfere with efficient performance of a complex, or creative, task.

The authors visited 19 projects to study their development processes. It is their conclusion that, if software process models are to offer more than illusory comfort, then we must focus them on something other than phase-ending events and activity descriptions that are useful only when there is little uncertainty. It is this paper's primary thesis that the focus should be on the activities that account for the most variation in software productivity and quality. Current process models are prescriptive. Descriptive process models of the way software is actually developed are needed.

Curtis, Bill, "Three Problems Overcome with Behavioral Models of the Software Development Process," Proceedings: 11th International Conference on Software Engineering, May 1989.

This paper clarifies the distinction between a functional approach and a behavioral approach to process modeling.

Functional approach: software development processes focus on the software artifact at given stages in its evolution and the nature of the transformations being applied to it during these stages; the software process is bounded by those activities that initiate and terminate the development of a specific software product. Three problems that this approach can lead to are: 1) the progression of stages through which the artifact evolves gets confused with the organization of the processes through which people develop software, 2) project processes that do not directly transform the artifact are not analyzed for their productivity and quality implications, and 3) the process is treated as discrete rather than continuous in time (i.e., each project invokes a separate process).

Behavioral approach: a behavioral analysis of software development and the factors that control its productivity and quality takes into account cognition and motivation, group dynamics and organizational behavior. Such an analysis does not replace traditional models of software product evolution - rather, it supplements them with a greater understanding of what controls project outcomes. A layered behavioral model has been proposed to analyze problems experienced in developing large software systems. The layers that must be present in a comprehensive process model include: individual (cognition and motivation), team and project (group dynamics), and company and business milieu (organizational environment).

Finkelstein, Anthony, " 'Not Waving but Drowning': Representation Schemes for Modeling Software Development," Proceedings: 11th International Conference on Software Engineering, 1989.

Given the requirements that a modeling representation must be formal, explicit and enactable and, in addition, support various roles (i.e., environments, active method guidance, meta-programming and development analysis), it is not

surprising that it has proved impossible to find a single representation that is sufficient. The paper rejects the standard repertoire of schemes and argues that what is needed is not incremental improvement or minor changes to existing schemes but a quantum jump in the type of representation scheme to be used. He suggests a "pluralist" approach in which descriptions in a variety of formalisms can be used to complement each other and advances some ideas (such as non-standard logical formalisms) in this vein for further exploration.

Hatley, Derek J., "The Case for Parallel Development," Embedded Systems Programming, January 1991.

A layered model of the software development process is presented that is considerably more complex than other (i.e., sequential) development process models, which are considered by the author to be too simplistic to be useful.

Kellner, M.I., and Hansen, G.A., "Software Process Modeling," Technical Report, Carnegie-Mellon University/Software Engineering Institute, CMU/SEI-88-TR-9, May 1988.

This technical report pulls together much of the same material as is represented in shorter papers in this bibliography. It includes: an overview of software process modeling; a more in-depth overview of the Post Deployment Software Support (PDSS) area focused on in SEI's modeling efforts; an overview of modeling approaches considered and a more in-depth overview of software process modeling with STATEMATE; a summary of results and lessons learned.

Kellner, Marc I., "Representation Formalisms for Software Process Modeling," Proceedings of the 4th International Software Process Workshop: Representing and Enacting the Software Process, ACM, 1988.

Kellner uses process model synonymously with process definition. He provides a good list of requirements for an effective process model (i.e., definition) and a brief comparison of the capabilities of STATEMATE when compared to the list of requirements.

Notkin, David, "Applying Software Process Models to the Full LifeCycle is Premature," 1988.

Notkin focuses on process programming to define software processes. His thesis: sufficient experience in building many complete instances (look at compiler and operating system technology as a good example) is necessary before you can hope to generate instances. And even that is not sufficient if enough formal notations, useful for the actual parameterization, have not been developed. Add to this the fact that there are currently no commercially successful full life cycle environments, how can we expect to construct viable process-driven environments? We need to continue our efforts to develop full lifecycle environments and focus on narrow ranges of lifecycle activities with the intention of producing parameterizable efforts in these areas before attempting to apply software process models to the full lifecycle.

Rombach, H.D., "A Specification Framework for SW Process: Formal Specification and Derivation of Information Base Requirements," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

Author asserts that most existing approaches to specifying software engineering processes are not satisfactory. They are frequently incomplete, inconsistent and imprecise. Part of the reason for this is that no single language can satisfy the needs of software engineers as well as the designers of the information base. This belief led the researches at the University of Maryland to propose a specification framework idea. The framework consists of three representation levels: 1) the application level, which allows the user to specify the relevant aspects of a software engineering scenario in a natural way, 2) the intermediate level, which allows for a "complete" (i.e., executable) specification of a software engineering scenario and 3) the information base level, which allows the specification of a software engineering scenario in terms of objects to be stored and operations on these objects. A first version of a software process specification language exists. The author is currently working on first ideas for languages at the application and information base level.

Tully, Colin, "Software Process Models and Iteration," IEEE, 1987.

This short paper makes some useful distinctions between prescriptive versus descriptive models, activity-based versus deliverable-based models and iteration in models vs. backtracking.

Williams, Lloyd G., "A Behavioral Approach to Software Process Modeling," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

Presents an argument for a behavioral approach to software modeling which focuses on the effects which the activities produce rather than the specific procedures (or algorithms) used to produce those effects, i.e., what an activity is supposed to do rather than how it is to be done. The software process is described in terms of events which occur in the development effort, rather than the state of the product. The software process model is described as a set of activities where an activity is defined to consist of a set of preconditions, an action, a set of postconditions and a set of messages.

G.4 Process Modeling - Specific Methods and Tools

G.4.1 Entity Process Models/STATEMATE

Humphrey, W., and Kellner, M.I., "Software Process Modeling: Principles of Entity Process Models," Technical Report, Carnegie-Mellon University/Software Engineering Institute, CMU/SEI-89-TR-2, February 1989.

This paper outlines the principles of entity process models and suggests ways in which they can help to address some of the problems with more conventional, functional or task-oriented approaches to modeling software processes. Many traditional process models are extremely sensitive to task sequence; many are unrealistic thereby biasing the planning and management system. Entity-based models deal with real entities (for example, requirements, the finished program, program documentation design) and the actions performed on them The process of producing an entity process model entails: 1) identifying the process entities and their states, 2) defining the triggers that cause the transitions between these states, 3) completing the process model without resource constraints, and 4) imposing the appropriate limitations to produce a final constrained process model.. This process is illustrated by a detailed example using the STATEMATE toolset. Entity process models focus on the dynamic behavior of a process, are formal and enactable, and permit automated tests and analyses.

Kellner, Marc. I., "Software Process Modeling Example," Proceedings of the 5th International Software Process Workshop: Experience with Software Process Models, October 1989.

Brief paper conveys the flavor of the SEI approach to software process modeling through the presentation of an example model fragment.

Kellner, Marc I., "Experience with Enactable Software Process Models," Proceedings of the 5th International Software Process Workshop, October 1989.

This short paper describes using STATEMATE to model the post deployment software support process used by the US Navy to support the avionics system for F-14A aircraft and by the Air Force to support the avionics system for the F-16A/B aircraft. It describes the amount of research that preceded the modeling and points to a number valuable outcomes that have resulted form the modeling efforts (such as a substantial increase in understanding of the process and help in highlighting problems and areas of opportunity for process improvements). Qualitative examination is described. The models, as they currently exist, are not quantified and cannot predict things like manpower requirements or schedule.

Kellner, Marc I., Hansen, G.A., "Software Process Modeling: A Case Study," IEEE, 1989.

Describes the use of STATEMATE for software process modeling. Covers: the major objectives of software process modeling, primary capabilities required of

software process models, requirements for a modeling approach (an excellent and comprehensive list), and a list of potential modeling approaches. STATEMATE, the modeling method chosen, supports multiple viewpoints: behavioral (using Statecharts, an improved variety of state transition diagrams), functional (using Activity charts, which are enhanced data flow diagrams), and structural (using Module Charts to represent organizational units or individuals who perform the activities depicted in the activity chart). The STATEMATE modeling methods were applied to the task of improving the process used by the Air Force to modify Technical Orders. General lessons learned from this exercise are included. The SEI continues to investigate and apply promising approaches from the set of existing methodologies and toolkits that may be applied to software process modeling

G.4.2 GRAPPLE

Huff, Karen, "Probing Limits to Automation" Towards Deeper Process Models," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

The author explains how artificial intelligence technology can be applied to reasoning about a process with respect to its "purpose", that is, the goal of the process in the particular context. At the University of Massachusetts, an intelligent assistant called GRAPPLE, has been designed for supporting the process of software development, based on an AI planning paradigm, where plan recognition is used to detect and avert process errors and planning is used to cooperatively automate the process. A Prolog implementation is now operational, providing a means for defining a process, including process constraints and default rules; given such a definition, sequences of actions can be recognized and potential process errors diagnosed.

G.4.3 IDEF/SADT

Cullinane, T.P., and Mayer, R., "IDEF0 & IDEF3: Methods for Implementing a Warehouse Control System," Proceedings of the IDEF Users' Group Conference, Washington DC, October 1992.

The purpose of this paper is to demonstrate how IDEF0 and IDEF3 can be utilized to enhance the effectiveness of feedback control in a system. IDEF0 has proven to be an excellent method for developing models of systems that require control. The significant variables that must be monitored and adjusted to keep a system within appropriate limits of control become much more obvious when placed in the context of a function model. IDEF3, excellent for capturing the knowledge about how a particular process or event should work, then offers a means for detailing the logic of the operations that must be performed within each functional area. The IDEF3 method has proven to be extremely useful for describing "how to bring a system back into control" when it is necessary to make some adjustments in the way in which the system is being operated.

Duran, P., "Why IDEF is Better than Structured Analysis for Process Modeling," Eclectic Solutions Corporation, May 1992

In the course of trying to sell IDEF concepts, many of us find ourselves facing organizations that have been using some form of Structured Analysis (SA) (Yourdon/DeMarco, Gane and Sarson, McMenamin and Palmer, Ward and Mellor, Hatley/Pirbhai..) They want to know how IDEF differs and why they should consider using anything other than their brand of SA. They probably also have a CASE (Computer Aided Software Engineering) tool, even if nobody is using it. This paper is geared to the IDEF professional who needs some answers to these questions. Thus some knowledge of IDEF and SA is assumed in the process of doing the comparison. This is not meant to be a tutorial in either method.

Duran, P., and Irvine, C.A., "Planar Views: Multi-Layer IDEF0 Modeling," Eclectic Solutions Corporation, 1991.

This paper presents an approach that provides a solution to several different difficulties that occur on IDEF0 projects. The approach involves a new use of FEO (For Exposition Only) diagrams. This approach solves some of the most serious problems of complexity of models, provides a means for dealing with multiple viewpoint models, and addresses several other long-standing modeling issues.

Fritz, M.W., Hoffman, S.L., Cullinane, T.P., "Utilizing IDEF0 for Structuring a System Support Process," Proceedings of the IDEF Users' Group, Albuquerque, NM, May 1992.

The purpose of this paper was to outline how IDEF0 is utilized to structure a system support process (both hardware and software) for a large avionics system maintenance project. Emphasis is on the positive and negative aspects of getting started with IDEF0 in a project team environment. The unique benefits derived from the use of IDEF0 for planning a future "to-be" system (without the benefit of having an "as-is" system) was described. The resulting model served to bring the team together (unified vision, standard vocabulary) and to unify its goals and understanding. Potential conflicts, duplication of tasks and recognition of activities that are critical to the system were discovered. For project management: The model helped in the creation of an effective work breakdown structure; the fact that it did not reflect the dynamic aspects of the support system could be worked around when used in conjunction with flow charts and process flow diagrams. For program management: the model helped users to understand how to structure the organization to gain the greatest amount of management control while still maintaining some level of flexibility.

"IDEF3 Process Description Capture Method Overview," Knowledge Based Systems, Inc., 1992.

One of the primary mechanisms used for communicating information about a situation is to describe an ordered sequence of events or activities. The IDEF3 Process Description Capture Method was developed to provide a mechanism for collecting and documenting processes. IDEF3 captures precedence and causality relations between situation and events in a form that is natural to domain experts. The goal of IDEF3 is to provide a structured method for the expression of domain expert's knowledge about how a particular system of organization works

Irvine, C.A., "IDEF0 Model Dynamics, Activation and Simulation," Eclectic Solutions Corporation, 1989.

IDEF0 is a method that contains the appropriate framework for specifying the active and static constituents of a system and how they operate to carry out the actions of a system. IDEF0 models can describe the dynamic performance of a system, including real-time characteristics. The fundamental nature of IDEF0 diagrams and models is often misunderstood. The description of specification of the dynamics of a system that is explicit in IDEF0 diagrams and models is seldom appreciated. This paper will explain the dynamics expressed in an IDEF0 model and show how the dynamics of a system may be examined and studied by automated simulation of a system as specified in the IDEF0 model. Also discussed is how to include the required information in an IDEF0 model and how to analyze it to answer questions about the dynamic behavior of the that it models. Simulation based on automated execution of IDEF0 models is also described.

Marca, D.A. and McGowan, C.L.,"SADT: Structured Analysis and Design Technique," McGraw-Hill, NY, 1988.

The definitive book on SADT/IDEF0 modeling methods. The book presents the general system concepts inherent in SADT, explains how SADT is commonly practiced, describes how the SADT modeling process is managed, discusses current automated support for SADT and presents real-life examples.

Mayer, R.J., Cullinane, T.P., deWitte, P.S., et al., "Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report," AL-TR-1992-0057, Knowledge Based Systems, Incorporated, May 1992.

This document provides a method overview, practice and use description, and language reference for the IDEF3 Process Description Capture Methods developed under the Information Integration for Concurrent Engineering (IICE) project, funded by Armstrong Laboratory, Wright-Patterson Air Force Base, Ohio. IDEF3, as defined in this report, is being proposed as a standard to the IDEF Users Group.

Mayer, R.J., Painter, M.K., deWitte, P.S., "IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications," Knowledge Based Systems, Inc., 1992.

This report presents the concept of an IDEF family of methods, which, taken together, provide a comprehensive modeling capability. The discussion centers on experiences in the use of the IDEF methods to perform modeling activities in support of CIM and CE system development. Experience with three such methods is described in detail, namely, IDEF0 Function Modeling, IDEF1 Information Modeling, and IDEF1X Data Modeling. Following this discussion, the emerging IDEF methods, including IDEF3 Process Description Capture, IDEF4 Object-oriented Design, IDEF5 Ontology Description and IDEF6 Design Rationale Capture are introduced and their envisioned application potential for CIM implementation described.

Mayer, R.J., Cullinane, T.P., et al., "IDEF3: A Process Flow Description Capture Method," Progress in Material Handling Research, 1992, ISBN 1-88-27-89-00-0, Milwaukee, May 1992.

IDEF3 is a method for capturing process descriptions and object state transitions corresponding to process executions. IDEF3 differs from traditional process flow modeling techniques in that it is focused on facilitation of the capture and organization of facts rather than the formulation of idealized models. IDEF3 descriptions differ from unrestricted text and ad hoc diagrammatic descriptions because the language is based on a formal mathematical semantics. Thus, an IDEF3 description base can be used as the basis for the design of many different types of models ranging from cost and schedule models to simulation models. This paper presents a succinct overview of the method and describes the potential applications for this technology in the development of material handling systems.

Mogilensky, Judah and Stipe, Dennis, "Applying Reusability to Software Process Definition", Tri-Ada Conference Proceedings, 1989.

This paper introduces the notions of applying reusable process step components to the definition of the software development process thereby allowing the construction of custom project life cycles from reusable process step components. Process steps are selected and retrieved from a library during the life cycle design activity. Each process step accepts as an input a relatively abstract model of the system under development, and produces as an output a new model of the system that is relatively less abstract. The notation and modeling approach is IDEF, The Integrated DEFinition Method. Project life cycle design should eventually have an automated tool that assists the software engineer in defining the key characteristics of a given project and applying rules to the activities of process step selection and life cycle design. The paper also relates this method of process definition to process maturity improvement efforts using SEI assessment levels.

G.4.4 IMDE

Clark, P., Honious, J., Renken, K., (TASC: Fairborn, Ohio), "The Integrated Model Development Environment," a paper presented at Elector '92, Boston, MA.

Paper describes the Integrated Model Development Environment (IMDE), an object-oriented environment that provides graphical description and construction of models, maintains configuration control of models (including input and output files and their constituent parts), analyzes simulation results and allows documentation to be integrated with the models. It is argued that, by allowing more time to be spent in designing models (rather than in tracking down syntax errors, maintaining configuration control and sifting through mountains of raw data), the IMDE will allow for a significant reduction in the complexity involved in building discrete event simulations. In addition, IMDE allows the analyst (as opposed to the programmer) to get much closer to the actual model implementation which uses Petri Nets.

G.4.5 Petri Nets

Jensen, K., "Coloured Petri Nets: A High Level Language for System Design and Analysis," Advances in Petri nets 1990, Lecture Notes in Computer Science, vol. 483, Spring, Berlin Heidelberg New York 1990, pp. 342-416.

The paper describes Coloured Petri Nets and support for modeling and analysis by tools such as Design CPN.

Jensen, K., and Rozenberg, G., "High-level Petri Nets," Springer-Verlag, 1991.

This book contains reprints of some of the most important papers on the application and theory of high-level Petri nets. High-level Petri nets make it possible to obtain much more succinct and manageable descriptions than can be obtained by means of low-level nets ,while, they still offer a wide range of analysis methods and tools. High-level Petri nets are now widely used in both theoretical analysis and practical modeling of concurrent systems and have been used, particularly in Europe, to define and model the software process.

Kramer, Bernd and Luqi, "Petri Net-Based Models of Software Engineering Processes," Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, Jan. 1990.

Presents a Petri net-based process model (PNP model). Claims the formalism provided by a Petri net model contributes to consistent and precise understanding of software process, enables automated support to enhance the reliability and reusability of process models and opens ways to automate well-understood parts of software processes. This is a behavior-oriented software process model which is effective in modeling dynamic and distributed software process activities.

G.4.6 Process Languages

"Domain Specific Environment Repository Process Programming Language Experimentation," Informal Technical Report for the Software Technology for Adaptable, Reliable Systems (STARS), Publication No. GR-7670-1254(NP), 15 November 1991.

This report presents the results of a process modeling and programming experimentation task carried out by TRW, Fairfax, VA, under a subcontract to Unisys Defense Systems, Reston, VA, as part of the STARS program. The experiment was conducted over a six-month period, February through July, 1991, and studied two process programming languages, MVP-L and APPL/A (which represent the current state of the art), and their prototype tools. The study found process programming to be difficult and time consuming. It was felt that process

experts were vital for any success in this area. In general, process programming was deemed an immature area.

Lehman, M.M., "Some Reservations on Software Process Programming," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

Paper expresses the author's strong reservations about process programs and the role they can or should play in software development. He questions whether they will provide more insight into the software development process, produce better understanding of that process, or lead to its significant improvement. He fears the popularization of the concept of process programming. Process programming cannot adequately capture the heavily context dependent software development process in which process structure and composition cannot be predicted. A program based model may actually limit the scope and power of what can be achieved in the software development process. The author feels that such descriptions will ultimately only have usefulness if used in a limited way to guide IPSE and tool architecture and to help control IPSE usage.

Osterweil, Leon, "Software Processes are Software Too", Proceedings - 9th International Conference on Software Engineering, March 1987.

This paper advocates the use of "process programming" to define software development processes. Provides examples of process programming, outlines its advantages in describing process, points to the need for process programming language studies and relates it to software environment architecture research (in particular, Arcadia).

Osterweil, Leon, UC, Irvine, (Arcadia Project) "Process Centered Software Environment as Interpreters of Software Process, Programs, AIAA Computing in Aerospace 8, October, 1991.

"Software processes can be specified quite precisely using programming-like approaches, and can thereby be used to indicate just how software tools are to be applied by computing devices and humans. Work carried out as part of the Arcadia project has enabled us to specify such software objects as requirements specifications, designs and test plans using programming language like constructs and to create process programs for building these objects."

Ramanathan, J., and Sarkar, S., "Providing Customized Assistance for Software Lifecycle Approaches," IEEE, 1988.

This paper describes a tightly coupled environment architecture that uses underlying representations of the software development process, the objects and relationships being manipulated, the functionalities of the tools, and the roles of the various project members to provide automated support for enforcing the discipline required to ensure the success of large multi-person projects. The paper focuses on features of a Conceptual Modeling Language (CML) for specifying such representations. A prototype implementation has been constructed as part of the TRIAD project at Ohio State University.

Roberts, Clive, "Describing and Acting Process Models with PML," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

*The language Process Modeling Language (**PML**) combines features from both specification and programming and permits the integration of not only tool and data, but also the activities constituting the process itself. Using a role/interaction paradigm, it allows the description of behavior which is both generic and dynamic - classes can be refined and changed on-the-fly, minimum restrictions on sequence and concurrency are necessary. Classes are the main objects of the language and primitive object groups and their behavior are built into the language. A prototype simulator implemented in Smalltalk 80 and based on the Smalltalk environment has been developed to allow enaction of the PML process model. A mature PML machine would act as a fully extensible IPSE, tailored to the process required of the users.*

Scacchi, Walt, "Modeling Software Evolution: A Knowledge-Based Approach," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

*This paper argues that the complexity of objects, attributes, relations, constraints, procedural and non-procedural control structures, and process actions that must be described by a process language will be better realized by using a knowledge representation language, rather than a programming language, to define the software process. At the University of Southern California (as part of the USC System Factory Project), **Gist** is being used as an operational knowledge specification language, supplemented by the use of tools such as specification generators, language-directed editors, specification analyzers, functional simulators and a software hypertext system to construct, formalize and manage the knowledge descriptions. A commercial expert system development environment, Knowledge Craft from Carnegie Group, Inc., has extended the processing capabilities to accommodate inferential reasoning mechanisms and abstractions.*

Shepard, T., Sibbard, S., Wortley, C., "A Visual Software Process Language," Communications of the ACM, Volume 35, Number 4, April 1992.

*Describes **VPL (Visual Programming Language)**, a formal programming language designed to visually represent and permit enaction of software development processes. A VPL model of a software process is a directed graph of nodes and edges and combines some of the features of the object-oriented paradigm, Petri nets, and logic flowcharts. An object in a VPL program represents all the artifacts associated with a currently active individual work assignment. Non-linear control is available via activation and deactivation mechanisms and the archiving of earlier versions of an object. In this paper, language symbols are explained, examples of language application are provided and the use of VPL to describe the maintenance of aircraft software for the Aurora Software Development Unit (ASDU) in Canada discussed. The language appears to be useful in describing a software process. However, tool support is minimal. Authors are from the Department of Computer Engineering at the Royal Military College of Canada.*

Their work was supported in part by the Canadian Department of National Defense (DND).

Taylor, Richard, "Concurrency and Software Process Models," IEEE, 1987.

Emphasizes that concurrency must be a fundamental descriptive (or structuring) mechanism in a process definition "language." Any language without support for concurrency should be judged inadequate.

G.4.7 RDD 100

Alford, Mack, "Strengthening the Systems Engineering Process," paper presented at NCOSE, October, 1991.

This paper describes an executable "behavior diagram" notation (which can be directly executed by RDD-100) for describing system functionality which eliminates the deficiencies of existing system analysis notations. Contains a good critique of the shortcomings of many of the conventional modeling techniques.

"Requirements Driven Development: An Overview," Ascent Logic Corporation, San Jose, CA, December, 1989.

This paper presents the Requirements Driven Development (RDD) method, language, and tool suite. It discusses the factors in inter-group comprehension, compares RDD to other high level definition languages, and argues for the need for systems design automation.

G.4.8 SPM

Lai, Robert, "Process Definition and Process Modeling Methods," Software Productivity Consortium, SPC-91084-N, 1991.

Process description characteristics are presented as a prelude to the major purpose of the report, a presentation of the SPC's approach to software process modeling .

Presents the SPC's approach to software process modeling based on a generic process model, a two-level state model that models artifact states (A-states) and process states (P-states). The two-level model allows the separation of the process description from the representation used for the artifacts.

Pre-conditions and post-conditions applied to activities enable this method to model a process description with more freedom than a procedural method in which the process description must be followed exactly step-by-step. The SPC opportunistic process allows more freedom being constrained only by the requirement that, before beginning an activity, the necessary preconditions exist.

The process modeling process described is essentially a form-filling process. Each form is a template that asks for some data to be provided to describe the artifacts connected with the process (mostly documents), the relationship between

artifacts, process states, operations that can be performed in a process state, analyses which can take place within a process state, role definitions, action definitions. For convenience in representing relationships between roles, activities, artifacts, A-states and P-states, a graphical notation has been developed to supplement the forms. Steps that may be followed in applying this method are included. Various scenarios are described (bottom-up, top-down, inside-out, etc.)

It is stated that the two-level model of a software process can be viewed both as a specification for the process and as a specification for a process-centered software environment to be used to support the process. However, other than suggestions for implementation, no available automatic support is mentioned.

G.4.9 SPMS

Krasner, Herb, Terrel, J., Linehan, A., Arnold, P. and Ett, W.H., " Lessons Learned from a Software Process Modeling System," Communications of the ACM, Vol. 35, No. 9, September, 1992.

Describes the use of the Software Process Management System (SPMS) for process model development. SPMS development was funded as a breakthrough initiative by DARPA/STARS in 1990. The SPMS includes capabilities for process model definition, validation, automated generation of project-specific plans with tailorable execution constraints, continuous process evaluation and model-driven enaction in the STARS Software Engineering Environment (SEE). The process model is constructed using model components available in the SPMS process database. SPMS introduced the notion of integrating a quantitative software quality model with the process model at both the generic and project-specific levels. SPMS is designed to handle dynamic process replanning resulting from redirection, rework and contingency analysis.

SEI has been alpha testing SPMS as a definition tool that can be used to create assets for the Process Asset Library (PAL). SPMS is still in the R&D phase. Results: SPMS was found to handle behavioral, functional and organizational process information well. Additional facilities are needed to better support the information modeling perspective.

The first functional prototype was available in June 1991. A fully functional SPMS will eventually be embedded in a process-centered STARS SEE for supporting active models of the software process in the abstract and in action on demonstration projects in the 1993 to 1996 time period.

G.4.10 Structured Analysis Methods

DeMarco, T., "Structured Analysis and System Specification," Yourdon Press, New York, 1978.

This landmark book describes the structured analysis approach, which like SADT/IDEF0, uses a graphical language to build models of systems. There are four basic features in structured analysis: data flow diagrams, data dictionaries, procedure logic representations and data store structuring techniques. SA data flow diagrams are similar to SADT/IDEF0 diagrams, but they do not indicate mechanism and control, and an additional notation is used to show data stores.

Hatley, D.J., and Pirbhai, I.A., "Strategies for Real-Time System Specification, Dorset House, 1987.

An important book presenting extensions to structured analysis that make it more appropriate for the analysis of real-time systems

Ward, P.T., and Mellor S.J., "Structured Development for Real-Time Systems," Yourdon Press, Englewood Cliffs, New Jersey, 1985.

This book describes the real-time structured analysis approach that adds a state-oriented notation to structured analysis dataflow models, which allows behavior to be represented.

G.4.11 System Dynamics

Abdel-Hamid, T., and Madnick, S.E., "Software Project Dynamics: An Integrated Approach." Prentice Hall, 1991.

This book describes the use of the system dynamics modeling approach to develop an integrative model of software development project management - developed on the basis of an extensive review of the literature supplemented by focused field interviews of software project managers in five organizations. The model divides the software development and management activities into four areas: (1) human resource management, (2) software production, (3) controlling, and (4) planning. Two key features of this model that distinguish it from most others are that it is integrative and it is a system dynamics model. It is integrative in the sense that it integrates the multiple functions of the software development process, including the management-type functions (e.g., planning, controlling, and staffing) as well as the production-type functions that constitute the software development life cycle (e.g., designing, coding, reviewing and testing). Benefits: overall understanding is increased, problem diagnosis and solution evaluation are increased by being able to model interactions and interdependencies, the chain of effects from intervention to first, second and third-order consequences can be traced.

System dynamics is the application of feedback control systems principles and techniques to managerial, organizational, and socioeconomic systems. As applied to software process, this approach allows modeling of system structure and uses computer simulation to enhance understanding of system behavior. Benefits: greater fidelity in modeling processes, making possible both more complex models and models of more complex systems, and providing a vehicle for controlled experimentation in the area of software development.

"System Dynamics Modeling," American Programmer, Vol. 6, No. 5, May, 1993.

The May, 1993, issue of the American Programmer focuses on the use of system dynamics to model software development and project management systems. Articles include Abdel Hamid's recommendation to software managers to begin to "think in circles," a description of a software process model developed at Draper

Laboratory that can be used as a "flight simulator" for software managers, using system dynamics to enhance executive dialog and debate, modeling the rework cycles of defense and commercial software development projects, and modeling the impact of quality initiatives over the software product life cycle.

G.4.12 VPML

VPML: A Commonsense Approach to Enterprise and Process Modeling for the Domain-Specific Software Process Automation Technology Program, Technical Report, Appendix A, ISSI-A92A00002, 27 July, 1992.

A set of representation schemes and techniques are presented for defining all facets of the software process. This paper considers three modeling problems - process modeling, infrastructure modeling, and information modeling - as three related aspects of the larger problem of enterprise modeling. The report presents modeling techniques aimed at all three areas and collectively referred to as the Visual Process Modeling Language (VPML). VPML will become a part of the Enhanced Software Life Cycle Support Environment (E-SLCSE) project (now renamed ProSLCSE), the goal of which is to produce a process-centered software engineering environment that includes editors and enactment tools utilizing these methods.

G.5 Enactment Technology/ Process-Driven Environments - Comparative Assessments and General Articles

Curtis, Bill, Kellner, M.I., Over, J., "Process Modeling," Communications of the ACM, Vol. 35, No. 9, September, 1992.

Article presents an overview of current state-of-the-art in software process modeling. It provides a conceptual framework for discussing software process modeling and carefully defines terms used in the discussion. Topics covered include: Uses for process models, perspectives that a model may provide (functional, behavioral, organizational and informational), and reviews of five representation approaches for process information (process programming, functional approach, plan-based, Petri nets and quantitative models). Specific approaches reviewed include APPL/A, HFSP, GRAPPLE, Petri net role interaction model, and system dynamics. Issues in process modeling discussed include: formality, granularity and precision, prescriptive vs. descriptive vs. proscriptive models, multi-paradigm representations. It also discusses the use of models in process improvement and software project management. Process-based software development environments discussed include Arcadia, ISTAR, and MARVEL.

Drake, Richard, IBM, "Process Support Requirements - A View from the Top", AIAA Computing in Aerospace 8, October, 1991.

"It is dangerous to automate something before you have experience doing it manually."

"Environment Frameworks: Assessments," a report (N69-86-C-0415) prepared for the Naval Air Development Center (NADC) by Software Productivity Solutions, Inc., Melbourne, Florida, May 9, 1989.

This report assesses several commercially available environment frameworks. The focus is primarily on framework aspects that contribute to environment tool integration. Atherton Software BackPlane and SLCSE are among the environments evaluated.

Hevner, A., Becker, S., Pedowitz, L., "Integrated CASE for Cleanroom Development," IEEE Software, March 1992.

This paper addresses the difficulty in CASE environments when you are unable to integrate the results of one phase and its tools transparently into another phase and its tools because the phases' underlying concepts and representations differ. What is needed is a seamless methodology and representation that supports the entire development process. Using the concepts of one of the most successful formal methods, IBM's Cleanroom approach to systems engineering, a model is proposed in which stimuli and responses are basic information units - state information, procedural behavior and a usage hierarchy are added to complete the

definition of a process component. Though cleanroom techniques may offer advantages, much research and development is needed before an integrated Cleanroom environment can become a reality.

Huseth, Steve, Honeywell, "Obstacles to Automating Software Process Support," AIAA Computing in Aerospace 8, October, 1991.

Outlines the major technical, economic, and organizational obstacles that must be overcome to achieve software process support in practical software engineering environments.

Karrer, A.S., Scacchi, W., "Meta-Environments for Software Production," Journal of Software Engineering and Knowledge Engineering, December, 1992.

The term meta-environment is used to include generic environments, environment generators and other approaches to environment construction. This paper provides a good overview of the current approaches to environment construction by focusing on the five general topics: environment frameworks, customizable environments, process modeling, process programming and tool integration. A very extensive list of researchers/developers is included with descriptions of their products.

Lonchamp, J., Benali, K., Derniame, J.C., and Godart, C., "Towards Assisted Software Engineering Environments", Information and Software Technology, vol. 33, no. 8, October, 1991.

The paper characterizes Assisted Software Engineering Environments (ASEE) historically and by functionalities.

Lonchamp, J., Benali, K., Godart, C., Derniame, J.C., "Modeling and Enacting Software Processes: an Analysis," Proceedings: Fourteenth Annual International Computer Software and Applications Conference, IEEE Computer Society Press, 1990.

This paper establishes a list of requirements for designers, managers and developers in a model-driven environment. Strengths and weaknesses of three third generation IPSEs (TRIAD/CML, MARVEL and IPSE 2.5) are highlighted in terms of the list of requirements.

Mi, Peiwei and Scacchi, Walt, "Process Integration in CASE Environments," IEEE Software, March, 1992.

Paper distinguishes between environments that use a tool-invocation chain to support the use of tools from those that use a resource-transformation chain which progresses from initial artifacts to intermediate ones and then to the final product (where artifacts produced early in the life cycle are used later to create other artifacts). The paper proposes process integration to make the task execution chain explicit, flexible and reusable and which will provide mechanisms to guide, manage, monitor and control the progress of development. A software process model specifies an activity hierarchy and resource requirements. A process driver

interprets and executes this model according to its activity hierarchy. After enactment and during software development, for each task or action in a software process model, a status (ready, active stopped, broken, done, etc.) is attached. Using the Softman environment, which was developed as part of the University of Southern California's System Factory project, the authors implemented process-driven CASE environments using existing CASE environments. The resulting environment provided interfaces for both developers and managers.

Myles, D.T., "Automated Software Process Enactment," Proceedings from The Fifth Annual Software Technology Conference: Software - the Force Multiplier, April 21, 1993.

This paper provides a good introduction to process enactment. It discusses the typical roles for process enactment and lists the necessary elements of a process enactment technology. Myles summarizes the lessons learned by the IBM Federal Systems Company, Houston, Texas; FSC has been actively engaged in automated software process enactment since early 1990. Myles presents principal enactment tool requirements that have grown from this experience and briefly describes their use of the enactment tool, Process Weaver. He emphasizes the need for a disciplined process development methodology before process enactment is attempted. He argues for the concept of "piecemeal enactment" of small segments of the process and stresses the importance of pilot projects to confirm the utility of process enactment.

Osterweil, Leon, "Automated Support for the Enactment of Rigorously Described Software Processes," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

Osterweil's summary of the state of the art indicates how far we have to come before effective enactment facilities will be available. Process description languages, although many in number and approach, in general are not yet addressing what the author believes is the most central and difficult requirement in a process coding language - namely the extraordinary degree of dynamism that is required. Few if any existing languages support the dynamic changes necessary in the software development process. Secondly, environment architectures are often closely aligned with work on specific process coding languages. Here too, most researchers do not yet seem to be adequately addressing the need for dynamism - namely the need to support alteration of the process itself, perhaps even while the process is being enacted. Thirdly, the need to establish, analyze and maintain an enormous and bewildering welter of relations among environment object stores is such that we should not expect that database researchers will provide us with the solution to object storage and management problems in the near future. Lastly, user interface issues and the lack of adequate testing and evaluation for various description approaches and process enaction mechanisms gives us little reason to be optimistic that we can develop effective, reliable process enactment software any time soon.

"A Reference Model for Frameworks of Computer Assisted Software Engineering Environments," NIST draft version 1.3, prepared by the NIST ISEE Working Group, July, 1991.

The functional description of a set of services needed to describe software engineering environment frameworks. Process Management Services (Section 4) provides a functional description of process definition, process enactment, process

visibility and scoping, process state, process control and process resource management services that need to be supplied by a process-driven SEE.

"Representing and Enacting the Software Process," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

The purpose of this workshop was to focus attention on languages and notations in which formal models of software processes could be represented and "enacted." Proceedings include outline reports of the conference discussions (on topics including enaction formalisms, constructing enactable models, enacting the models and emerging issues) and position papers submitted by all attendees. Papers represent a wide variety of approaches and little consensus on a standard approach for either definition language, enactment mechanisms or environment support.

"Software Engineering Environment Capability and Viability Review," JIAWG SWEAT Team Report, 1989.

Report reviews Atherton Backplane, SLCSE and NASA SSE by answering a number of directed questions covering SEE characteristics, functional capabilities and programmatic issues.

Tate, G., Verner, J., and Jeffery, R., "CASE: A Testbed for Modeling, Measurement and Management," Communications of the ACM, Volume 35, Number 4, April 1992.

The important developments in CASE, software maturity and improvement, software metrics, and software process modeling and enaction, are brought together in this article and their relationships explored. Authors contrast their software process modeling goals (measurement and management) with process programming and state transition process modeling. The paper advocates a data flow diagram approach both for automating the measurement process and for producing notation that is easy to understand for software managers and developers. Enaction consists of a developer selection of one component and one type of operation; the single-purpose session is important as a measurement unit. The paper proposes that the CASE development environment be surrounded by a metrics envelope which in turn is "surrounded" by the software process model. The developer does not interact with the CASE tool set; but rather "enacts" a suitable process model as described above.

Weiderman, N.H., Habermann, A.N., Borger, M.W., and Klein, M.H., "A Methodology for Evaluating Environments," ACM SIGPLAN Notices, Jan. 1987.

This paper provides the requirements for an effective environment evaluation methodology, the individual steps of the methodology, and an example of how the methodology has been applied in practice.

G.6 Specific Process-Driven Environments and Enactment Technologies

G.6.1 Distributed System Factory (DSF)

Scacchi, Walt, "The Software Infrastructure for a Distributed System Factory," Software Engineering Journal, September, 1991.

Based on experience in creating and evolving the System Factory project at USC, a new experimental project, called the Distributed System Factory (DSF) project is being developed to provide a software infrastructure suitable for engineering large-scale software systems with dispersed teams working over wide-area networks. This software infrastructure is the central focus of this paper. The paper describes the information structures that can be used to model and create the infrastructure, software services that populate and execute within this infrastructure, and capabilities for growth. Some discussion of software engineering processes and DSF's software-process modeling framework.

G.6.2 ESF

Fernstrom, C., Narfelt, K., and Ohlsson, L, "Software Factory Principles, Architecture, and Experiments," IEEE Software, March 1992.

This paper, written by participants in the Eureka Software Factory project, distinguishes between two types of CASE vendors: component vendors, the makers of the factory "equipment," and factory vendors, the builders of environments, who select the most suitable equipment, integrate it, and customize it to fit a client's organization and production process. The paper points out that the requirements for process integration are higher for environments than for services. In the ESF project, process designers create and maintain process descriptions using a graphical notation that is like SADT, with detailed task descriptions and task synchronization described using colored Petri nets. To enact a process, the process designer attaches actions, expressed in an action language, to the Petri net transitions. Process enactment is provided by a factory process engine, which implements the runtime support for process programs with a set of service components present in every factory environment. A software bus provides a common understanding of the data exchanged among components, independent of their actual representation. A communication mechanism provides remote procedure calls and notification-based component interoperation.

G.6.3 ISTAR

Dowson, Mark, "ISTAR - An Integrated Project Support Environment," ACM SIGPLAN Notices, January 1987.

This paper describes ISTAR, an integrated, language independent, commercially available project support environment developed by Imperial Software Technology in London, England, which was an early commercial example of a process-driven environment. It included a comprehensive and extensible set of tools covering every aspect of the software and system development process. ISTAR was organized to support software process definition using a contractual approach to software and system development based on the recognition that every activity in the software process has the character of a contract between a contractor and a client. However, ISTAR did not enforce the sequence of activities in a process definition but simply provides support for them if required.

Graham, M.H., Miller, D. H., "ISTAR Evaluation," Technical Report CMU/SEI-88-TR-3, Software Engineering Institute, July 1988.

ISTAR's process definition is based on a "contract model" whose primary objective is that every individual in the organization know what is expected of him or her. To accomplish this, the relationships among the individuals of the organization are modeled as contracts. Each contract has a specification of the work to be performed under it, a person to whom it has been assigned, and a person for whom the work is being done. The collection of all contracts (forming a tree) constitutes the process definition. ISTAR makes no attempt to enforce any rules or standards on the data flows into and out of a contract. Contract databases serve as repositories of controlled project knowledge.

Strengths: the project and configuration management tools. Criticisms: Work area versions and database versions of software units are permitted; it is relatively easy for these versions to diverge inadvertently. The ISTAR model works best when a development project is well planned in advance and the resulting plan is executed without modification. However, the contract model does not always accommodate changes (particularly changes in the structure of the project) very well.

The bulk of the report deals with the tool sets supplied with ISTAR. Overall, ISTAR can be judged an emerging product, not a completed one. Use of ISTAR by customers tends to be experimental.

G.6.4 KI-Shell

KI Methodology for Workflow Knowledge Acquisition, Technical Report, No. 11, Update 2, copyright UES, Inc., August 1992.

Description of the KI Shell workflow modeling methodology which can be easily mapped to workflow enaction. The notation used to describe activities, information objects, applications, workers, and their inter-relationships are presented followed by a discussion of how models using these views are mapped to the workflow enaction.

G.6.5 MARVEL

Kaiser, Gail E., "Rule-Based Modeling of the Software Development Process, Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

Describes MARVEL, a model-driven environment, which uses rules both as a formalism for modeling the software process and a mechanism for automating the menial aspects of this process. A MARVEL rule consists of three parts, a precondition that must be true before a particular software activity can be executed, an activity, and a set of postconditions, exactly one of which becomes true after the activity terminates. MARVEL provides a form of controlled automation that the developers call opportunistic processing because MARVEL invokes tools as the opportunity arises. Controlled automation is accomplished by forward chaining and backward chaining on the rules, automatically invoking tools as soon as their preconditions are satisfied and as soon as one of the postconditions is required (i.e., as in the case where a user explicitly invokes an activity through a command.)

G.6.6 MELMAC

[Deiters, W., and Gruhn, V., "Managing Software Processes in the Environment MELMAC," ACM, 1990.

One key problem in the design of software process modeling languages is the variety of purposes that a model is used for (i.e., guidance, understanding, coping with changes). This variety does not lead to one software process modeling language which is well-suited for fulfilling all purposes. Some implementation approaches have tried to deal with this problem by distinguishing between an application level (oriented towards the need of the software process modeler), an intermediate level (which will permit the representation to be executable) and an information base level (mainly concerned with questions about object storing and retrieving).

This paper describes the approach taken in the environment MELMAC. They have adopted a view-based approach to model the software process (which includes an object type and activity view, process view, project management view, feedback view, distribution view and simulation view). Representation of each view is tailored towards the specific information of that view (i.e., pre-condition/activity/post-condition for the object type and activity view, Petri nets for the process view, specification of allowable modification points for changing process models on the fly in the feedback view, etc.) Then, all the information specified in the views on the application level is presented on the intermediate level in a uniform way to fulfill the purposes of execution and analysis of software process models - in MELMAC, FUNSOFT nets, described in some detail in this paper, are used.

Deiters, W., Gruhn, V., "Software Process Model Analysis Based on FUNSOFT Nets," Systems Analysis - Modeling - Simulation, vol. 8, West Germany, 1991.

This paper describes a high level Petri net type that has been adapted to the requirements of software process management. The paper emphasizes in particular software process model analysis by showing how these nets can be used to validate software process models and to verify software process model properties. The work was partially funded as part of the ESPRIT program. The MELMAC environment provides automated tool support for FUNSOFT modeling, instantiation and execution as well as support for the incremental management of process models.

G.6.7 PREIS

Kimball, John and Thelen, Karen, Honeywell, (PREIS - Prototype Engineering Information System), "Engineering Process Enactment: Requirements of Environment Frameworks", AIAA Computing in Aerospace 8, October, 1991.

Describes modeling and enacting processes with PREIS

G.6.8 SFINX

Bux, G, Marzano, G., "Software Process Design: A "job function" approach in the context of the SFINX project, AICA Annual Conference Proceedings, Bari, Italy, 1990.

This paper presents the general approach to software process modeling being defined and adopted in the context of the SFINX project. A library of predefined software process models together with rules and mechanisms for their customization is described. The adopted technique for the functional description is an "event-based" one (using the Event Graph language which is an extension of the Petri net formalism). This event-based approach is derived from the "behavioral approach" to software process modeling (see Williams).

G.6.9 TRIAD

Ashok, V., Ramanathan, J., Sarkar, S., and Venugopal, V., "Process Modeling in Software Environments," Proceedings of the 4th International Software Process Workshop, ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 4, June 1989.

Description of TRIAD, an integrated project support environment driven by a process model. Also describes the features of the process modeling language and the execution of the process model.

Ramanathan, J., and Sarkar, S., "Providing Customized Assistance for Software Lifecycle Approaches," IEEE, 1988.

This paper describes a tightly coupled environment architecture that uses underlying representations of the software development process, the objects and relationships being manipulated, the functionalities of the tools, and the roles of the

various project members to provide automated support for enforcing the discipline required to ensure the success of large multi-person projects. The paper focuses on features of a Conceptual Modeling Language (CML) for specifying such representations. A prototype implementation has been constructed as part of the TRIAD project at Ohio State University.

G.6.10 VSF

Bloor, Robin, The Software Tools' Software Tool," Software Development Monitor, March 1990.

A concise summary of the Virtual Software Factory (VSF) approach. VSF is described as not being a CASE product at all but rather a software workshop that can be used to build other tools.

Pocock, J.N., "The Case for Meta-CASE," paper available from Virtual Software Factory (VSF) Ltd., Vienna, Virginia.

This paper argues that the reasons for failure of CASE tools and environments to effectively address the "software crisis" derive from a mis-match between the specialist information needs of CASE users and the information manipulation provided by the tools, and from a lack of knowledge integration between the different tools required in order to enable a full spectrum of functionality to be provided over the whole development life-cycle. It argues that CASE can only be made truly effective by the application of a technology which allows the tools to be tailored for specific information environments. The terms "meta-CASE" is used to refer to the ability to build a CASE environment specifically tailored to the needs of a user organization, in terms of both standards and practices. A modeling based meta-CASE tool, VSF, is described as an effective way of enabling the rapid development and evolution of commercial quality CASE tools.

Pocock, J.N., "VSF and its Relationship to Open Systems and Standard Repositories," paper available from Virtual Software Factory (VSF) Ltd., Vienna, Virginia.

In this paper, the nature of the requirement for an open system approach to CASE tools and the relative roles of standard repositories and specialist "point" tools in such an environment is discussed. How the VSF facilities can be used to support the open system approach is outlined.

Pocock, J.N., "Frameworks and Tools for the Integration of Models," paper available from Virtual Software Factory (VSF) Ltd., Vienna, Virginia.

The fundamental problem associated with the provision of computer assistance to enterprise modeling as a whole, is the ability to compose multiple modeling paradigms in such a way as to allow the different models to interact without causing excessive information management load on either the support system itself or its users. VSF's facilities for formal definition have been validated

for 25 or more different combinations of modeling techniques. How this can be accomplished using the VSF system is described.

G.7 General Process Resource Materials

Bowen, Thomas P., Wigle, G. and Tsai, J., "Specification of Software Quality Attributes: Software Quality Evaluation Guidebook," RADC-TR-85-37, Rome Air Development Center, Griffiss AFB, NY, February 1985.

Quality factors identified in this guidebook each represent an aspect of quality that can be used to specify the types of qualities wanted in a particular product. Thirteen quality factors (efficiency, integrity, reliability, survivability, usability, correctness, maintainability, verifiability, expandability, flexibility, interoperability, portability and reusability) are identified encompassing performance, design and adaptation. These are presented along with the user concern that characterizes the need for each type of quality.

Clark, Peter G., Bard, Crawford S., "Evaluation & Validation Guidebook Version 3.0," TASC No. TR-5234-4, Ada Joint Program Office, 14 February 1991

This guidebook provides information that will help users to assess Ada Programming Support Environments (APSEs) and APSE components.

Clough, A.J., "Software Process Technology Analysis," CrossTalk, Number 34, June/July 1992.

Overview of software process technology maturity. August issue of CrossTalk then speaks more specifically of modeling technology.

Feiler, P.H., and Humphrey, W.S., "Software Process Development and Enactment: Concepts and Definitions," Technical Report SEI-92-TR-004, Software Engineering Institute, Pittsburgh, PA, , copyright 1992 by Carnegie Mellon University.

This report includes descriptions of some basic "core" software process terms. Its purpose is to provide a common communication framework for the software process and to reflect the views and findings of leading software process researchers.

IEEE Standard Glossary of Software Engineering Terminology (Approved September 23, 1982: IEEE Computer Society; approved August 9, 1983: American National Standards Institute), Copyright 1983 by The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY.

Software engineering is an emerging field. New terms are continually being generated, and new meanings are being adopted for existing terms. The Glossary of Software Engineering Terminology was undertaken to document this vocabulary. Its purpose is to identify terms currently used in software engineering and to present the current meanings of these terms. It is intended to serve as a useful

reference for software engineers and for those in related fields and to promote clarity and consistency in the vocabulary of software engineering. It is recognized that software engineering is a dynamic area; thus the standard will be subject to appropriate change as becomes necessary.

This glossary was prepared by the Terminology Task Group of the Software Engineering Standards subcommittee of the Software Engineering Technical Committee of the IEEE Computer Society.

NIST ISEE Glossary, version 1.0, compiled and edited by the NIST ISEE Working Group, April 1991.

The National Institute of Standards and Technology (NIST) Integrated Software Engineering Environments (ISEE) Working Group realized that there was a need for the use of a common terminology to facilitate discussions between the members of the working group. This effort represents an initial attempt to compile and edit a list of terms to eventually serve as a comprehensive glossary for software engineering environment activities. In several cases more than one definition is presented for a term. The glossary is viewed as an evolutionary activity. Updates are disseminated on an annual basis.

Proceedings from The Fifth Annual Software Technology Conference: Software - the Force Multiplier, 19-23 April 1993.

Software process definition tutorial notes; various presentations in the areas of process improvement, process modeling and process enactment..

STARS '91 Conference, Proceedings from the Process-Driven Development track, December, 1991.

Notes from Track 1, Process Driven Development: Process Driven Development Vision, Strategies, and Achievements; Process concepts; Process Asset Library; Experiment in Process Definition and Representation; Enacting the software process; and Process measurement.

STARS '92 Conference, Proceedings from the Process-Driven Development track, December, 1992.

Notes from Track 1, Process Driven Development: Process Driven Development Objectives/Motivation, Process Assets and the Process Asset Library (PAL), Process Definition and Modeling, Process Measurement, Experience in Process Driven Development, Process Improvement Perspective, and Experience with Automated Process Enactment.

Appendix H - Glossary and Acronyms

H.1 Glossary

The definitions listed here have been derived from the sources listed at the end of this appendix.

Arcadia Consortium - The Arcadia Consortium consists of a collection of separately funded, informally coordinated research and development projects at the University of California at Irvine (UCI), the University of Colorado at Boulder (UCB), the University of Massachusetts at Amherst (UMass), Stanford University, Incremental Systems Corporation, and TRW Defense Systems Group. This Consortium was formed in August 1987. The funding of these projects is provided by the Advanced Research Projects Agency (ARPA), and is administered by the National Science Foundation (NSF), and the Navy's Space and Naval Warfare Systems Command (SPAWAR). The universities also have other sources of funding, and the corporations have made internally funded investments in the work of the Arcadia Consortium.

Capability Maturity Level (SEI) - The Software Engineering Institute (SEI) at Carnegie Mellon University has defined five distinct maturity levels (initial, repeatable, defined, managed and optimizing) for categorizing the maturity of an organization's software processes.

Computer Aided Software Engineering (CASE) - Computer-Aided Software Engineering identifies a sector of the computer software industry concerned with producing software development environments and tools. The main components of a CASE product are individual tools to aid the software developer or project manager during one or more phases of software development (or maintenance).

Data Dictionary - A collection of entities used in a system together with attributes of those entities.

DoD-STD-2167A - A US Department of Defense documentation and review standard for the development of mission-critical software systems.

Enactment - The use of a formal process definition to carry out a process.

Entity Process Modeling - Entity process modeling is a structured graphical modeling approach that offers a set of three distinct but interrelated viewpoints that can be used to define a system or process: the functional view (often represented by data flow diagrams), the behavioral view

(most often represented by state transition notation), and the structural/organizational view (showing which entities perform specific activities in a system).

EUREKA Software Factory (ESF) Project- Over two hundred people spread across more than twenty sites in five countries are involved in the ESF project. The companies involved represent computer manufacturers, research institutions, CASE tool producers, and system developers. By 1991, halfway into the 10-year project, ESF has defined a reference architecture, completed the first implementation of a supporting framework and various tools and tool prototypes, and has undertaken several factory-integration experiments.

Framework - The infrastructure for tool integration. A product whose main role is to integrate a set of CASE tools while providing little direct functionality of its own. A framework provides the architectural basis of an environment and provides a set of services as a basis for environment construction.

Integrated Project Support Environment (IPSE) - A software environment that connects software tools, allows data to be freely interchanged, and makes it easy to manage project data. An IPSE also contains tools dealing with project management aspects of the software life cycle.

Integration - The property of different components working well together. Specifically, having varied CASE tools operated from a common user interface, sharing data, and accessing each other's functions.

ISTAR - An early integrated, language independent, commercially available project support environment developed by Imperial Software Technology in London, England, based on a "contractual" approach to software and system development.

KI-Shell - (Knowledge-based Integration Shell): object-oriented, workflow-based product for process modeling and computer enactment.

Life Cycle - The stages and processes through which software passes during its development and operational use.

Management Information Systems (MIS) - A computer based system of processing and organizing information - as distinguished from computer based systems which do not have a high

information content and concentrate more on non-informational objectives, (such as control, for example).

MARVEL - A software engineering environment using rule-based languages to support process definition.

MELMAC - MELMAC is an environment based on the execution of high-level Petri net representation of software process models.

Method - A method is a sequence of specific steps taken to perform an activity. For example, the activity of process modeling is accomplished using a particular method.
Methods are often performed using tools.

Methodology - A methodology is a collection of methods, rules, and postulates employed by a discipline. A software development/maintenance methodology is often a life cycle model customized by methods, techniques and tools.

Metric - Quantitative analysis values calculated according to a precise definition and used to establish comparative aspects of development progress, quality assessment or choice of options.

Object-Oriented Analysis - Examination of a problem by modeling it as a group of interacting objects.

PACT Project - The PACT project is part of the ESPRIT program and is partially funded by the Commission for the European Communities. The PACT project members include Bull SA, Eurosoft, GEC, Software Ltd., ICL, Olivetti, Siemens, Syseca, and Systems and Management.

Petri Nets - A Petri net is a mathematically-based, graphical, state-oriented notation for modeling dynamic and distributed software process activities.

Platform - Hardware architecture; particular model or family of computers.

Process - See Software Process.

Process Assessment - An appraisal, by a trained team of experienced software professionals, of an organization's current software process.

Process Assets - Life-cycle models, process descriptions, procedures, methods and other related process documentation, including process components and component templates.

Process Asset Library (PAL) - A reuse library for software process assets being developed by the SEI.

Process Definition - A partially ordered set of process steps that is enactable.

Process Enactment - The use of a formal process definition to guide and control the software process.

Process-Driven Environment - An environment supporting the software process by promoting a mutual assistance between a well-informed initiative engine and human developers. This level of support emphasizes control goals.

Process-Supported Environment - Environment supports the software process mainly through the invocation of tools from the SEE and possibly providing some process guidance.

Process Language - Process definition in which a software process is represented in the form of a program, using programming-like languages, notations, and formalisms.

Process Model - An abstract representation of a process architecture, design, or definition.

Process Programming - The use of a process language to express a process definition.

Process Simulation - Execution of a process model; the activation of a process model's dynamics.

Process Weaver - A product, developed by Cap Gemini in France, which allows software process modeling and instantiation, and provides guidance to the user and support for interfacing with CASE tools.

ProSLCSE - Software Engineering Environment which will provide support for process definition (using its language VPML) and enactment when completed.

Real-Time - Immediate response (for example, systems that must provide instant response to signals sent to them) - or any electronic operation that is performed in the same time frame as its real-world counterpart.

Real-Time Structured Analysis - A specification approach and graphical notation to describe the logical, physical, and behavior views of a real-time system.

Repository - In an environment, a database that defines and contains all of the information relevant to the components manipulated within the system.

Rule-Based Systems - Reasoning systems built around set rules; computer programs or systems that use rules to represent knowledge and consist of collections of antecedent-consequent rules.

Software Life Cycle Support Environment (SLCSE) - An SEE whose capabilities are provided through a preliminary rule base. Minimal working capability exists. Pro-SLCSE has largely superseded the work on SLCSE.

Software Engineering - The disciplined development and support of software using recognized methods and tools that help assure the quality of the product and the efficiency of the process.

Software Engineering Environment (SEE) - A software based system which provides automated support for the engineering of software systems and for the management of the software process.

Software Process - A set of activities, methods, and practices that guide people in the production of software.

Structured Analysis - A graphical language which can be used to build models - incorporating data flow diagrams, data dictionaries, procedure logic representations, and data store structuring techniques.

System Dynamics - A method which applies the principles and techniques of feedback control systems to the construction of models.

TRIAD - Software Engineering Environment developed at Ohio State University which supports process enactment using the language CML.

Tool - An individual CASE tool which automates one individual, focused activity in the life-cycle process.

Technology Insertion - The process by which an organization identifies, prepares for, acquires, implements, and institutionalizes new technology.

H.2 Acronyms

ACC - Air Combat Command

ACPIN - Automated Computer Program Identification Number

AETC - Air Education Training Command

AFCC - Air Force Communication Command

AFMC - Air Force Material Command

AFOTEC - Air Force Operational Test and Evaluation Center

AFSC - Air Force Systems Command

AFSPACECOM - Air Force Space Command

AI - Artificial Intelligence

AJPO - Ada Joint Program Office

AMC - Air Mobility Command

ANSI - American National Standards Institute

APPL/A - Ada Process Programming Language based on Aspen (used in the Arcadia project)

APSE - Ada Programming Support Environment

ARPA - Advanced Research Project Agency

ASCII - acronym for American Standard Code for Information Interchange. The ASCII code allows a standard representation of text characters in electronic form.

ASEE - Assisted Software Engineering Environment

ASSET - Asset Source for Software Engineering Technology - a facility supplying computer access to software reuse libraries, catalogs, and information via wide area networks and telecommunications.

CAIS-A - Common APSE Interface Set, revision A

CALS - Computer-Aided Acquisition and Logistics Support

CASE - Computer Aided Software Engineering

CEE - Customizable Engineering Environment (abbreviation used in long lists)

CIM - Corporate Information Management

CM - Configuration Management

CML - Conceptual Modeling Language (Process language used in the TRIAD environment)

CMM - Capability Maturity Model

COTS - Commercial Off-The-Shelf

D - Tool/method supports process Definition (an abbreviation used in the long lists)

D/E - Tool/method supports process definition and enactment (abbreviations used in the long lists)

DM - Supports Data Modeling (an abbreviation used in the long lists)

D/S - Tool/method supports process Definition and Simulation (an abbreviation used in the long lists)

D/S/E - Tool/method supports process Definition, Simulation and Enactment (an abbreviation used in the long lists)

DSF - Distributed System Factory

DoD - Department of Defense

DT - DeskTops, including Macintoshes and PCs (an abbreviation used in the long lists)

E - Tool/method support process Enactment (an abbreviation used in the long lists)

EAST - EUREKA Advanced Software Technology environment

EISE - Extendible Integration Support Environment

EPM - Tool/method supports Entity Process Modeling (an abbreviation used in the long lists)

E/R - Entity Relationship

ESF - EUREKA Software Factory

ESIP - Embedded computer resources Support Improvement Program

ETVX - Entry-Task-Validation-Exit

F - Environment built on a Framework which supports enactment (an abbreviation used in the process-driven environments long list)

FSPN - Formal Software Process Notation

GUI - Graphical User Interface

HFSP - Hierarchical and Functional Software Process description and enactment language

I-CASE - Integrated Computer-Aided Software Engineering

IDEF - Integrated DEFinition

IEEE - Institute of Electrical and Electronics Engineers

IMDE - Integrated Model Development Environment

IMIP - Industrial Modernization Incentive Program

IPSE - Integrated Project Support Environment

MBPA - Model-Based Process Assessment

MF - Mainframes (an abbreviation used in the long lists)

MIS - Management Information Systems

OO - Object-Oriented

PAL - Process Asset Library

PCTE - Portable Common Tool Environment

PDEE - Process Definition-generated Engineering Environment (an abbreviation used in the long lists)

PDSS - Post-Deployment Support System

PERT - Program Evaluation and Review Techniques

PML - Process Modeling Language

PN - Petri Nets (an abbreviation used in the long lists)

PPL - Process Programming Language (an abbreviation used in the long lists)

PREIS - PRototype Engineering Information System

RDD100 - Requirements Driven Development

RTSA - Real-Time Structured Analysis

S - Supports Simulation (an abbreviation used in the long lists)

SA - Structured Analysis

SADT - Structured Analysis and Design Technique

SCC - Software Control Center

SCE - Software Capability Evaluation

SD - System Dynamics (an abbreviation used in the long lists)

SDSA - Software Development and Support Activities

SEE - Software Engineering Environment

SEE/F - SEE/IPSE Frameworks supporting the creation of process-driven environments (an abbreviation used in the long lists)

SEI - Software Engineering Institute

SLCSE - Software Life Cycle Support Environment

SPMS - Software Process Management System

STARS - Software Technology for Adaptable, Reliable Systems

STSC - Software Technology Support Center

TECH - Technical (an abbreviation used in the long lists to denote technical application area)

TIS - Technical Information Sheets

TPDE - Non-customizable Turnkey Process-Driven Environment (an abbreviation used in the long lists)

TQM - Total Quality Management

USAF - United States Air Force

VPML - Visual Process Modeling Language

VPL - Visual Programming Language

VSF - Virtual Software Factory

WS - Workstations, including computers classed as mini-computers (an abbreviation used in the long lists)

References for definitions:

Feiler, P.H., and Humphrey, W.S., "Software Process Development and Enactment: Concepts and Definitions," Technical Report SEI-92-TR-004, Software Engineering Institute, Pittsburgh, PA, , copyright 1992 by Carnegie Mellon University.

Hanrahan, R., Peterson, R., Peterson, J., and Barney, D., "Software Engineering Environment Report, " March 1992.

IDEF Users Group Members Guide, October 1993.

IEEE Standard Glossary of Software Engineering Terminology (Approved September 23, 1982: IEEE Computer Society; approved August 9, 1983: American National Standards Institute), Copyright 1983 by The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY.

NIST ISEE Glossary, version 1.0, compiled and edited by the NIST ISEE Working Group, April 1991.

Software Engineering Institute (SEI), Training Handout, "Glossary for 'An Integrated Approach to Software Process Improvement' ". Software Engineering Symposium, August 1993.

NOTE: Acronyms are used so freely in product names that, to avoid an overly long list of acronyms, product acronyms were not included unless that product had been referred to explicitly in the report or appendix text.

(This page is intentionally left blank)

Appendix I - STSC Services & Information

I.1 The Software Technology Support Center

The mission of the Software Technology Support Center (STSC) is to transition technologies and exchange information to help DoD Software Development and Support Activities continuously improve their software quality and life cycle productivity.

A planned approach is necessary for successful transition. In general, transitioning effective practices, processes, and technologies consists of a series of activities or events that occur between the time a person encounters a new idea and the daily use of that idea. Conner and Patterson's Adoption Curve [Conner 82], shown in Figure G-1, illustrates these activities.

After encountering a new process or technology, potential customers of that technology increase their awareness of its usage, maturity, and application. If the process or technology is promising, then customers try to better understand its strengths, weaknesses, costs, and applications. These first activities in the Adoption Curve take a significant amount of time.

Next, the customer evaluates and compares the processes and technologies that show the most promise. To reduce the risk, customers usually try new processes or technologies on a limited scale through beta tests, case studies, or pilot projects. A customer then adopts processes or technologies that prove effective. Finally, refined processes and technologies become essential parts of an organization's daily process (institutionalization).

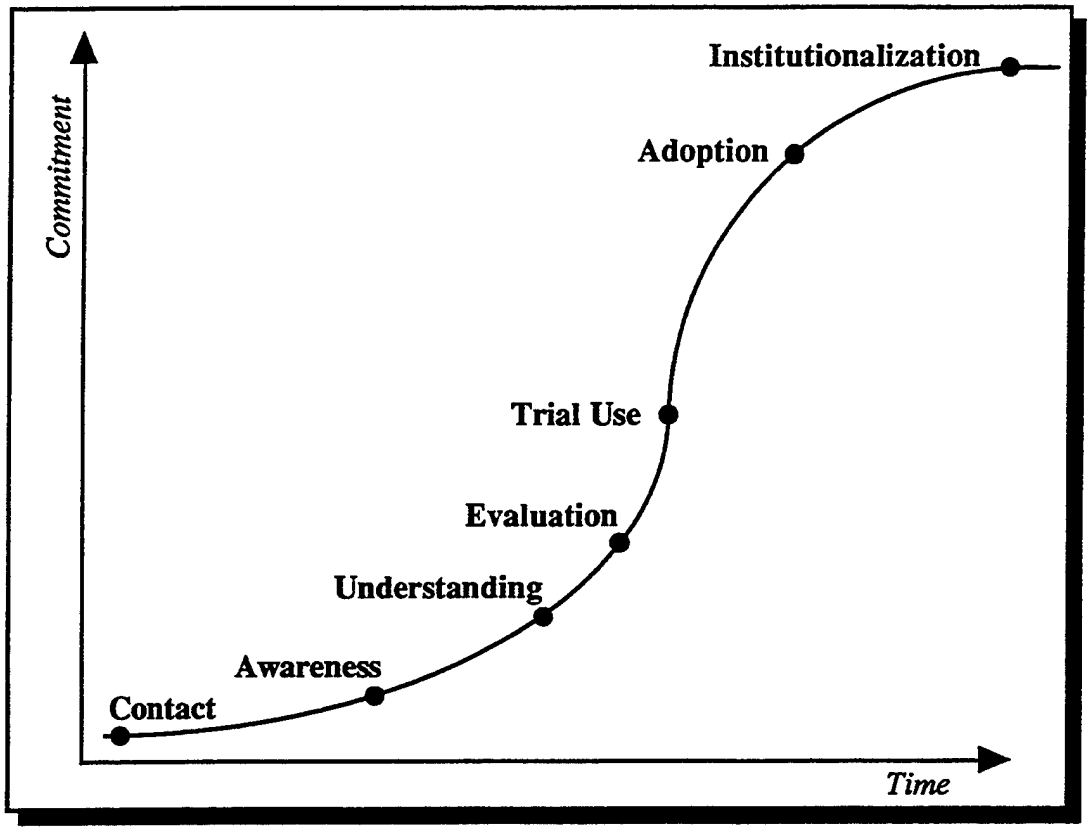

Figure I-1. Adoption Curve

Word processors are essential in most organization's daily operations. Yet, thirty years ago they did not exist. The institutionalization of word processors in many organizations followed a series of events similar to those identified in the Adoption Curve.

The STSC is researching and collecting information about technologies that will reduce the time and resources it takes to become aware, understand, evaluate, test, try, and adopt effective practices, processes, and technologies. The STSC has developed the following objectives to accomplish its mission:

- Technology Evaluation

  Identify, validate, classify, and evaluate effective processes and technologies.

- Information Exchange

Facilitate the exchange of better software business practices, processes, and technologies within the DOD.

- Insertion Projects

  Analyze and improve processes, adopt new methodologies as needed, evaluate and select effective tools, receive appropriate levels of training, and perform pilot projects to try out and confirm the technology insertion efforts.

- STSC Associates

  Develop STSC Associates who can infuse effective process and technology improvements through the use of STSC products, services, and processes.

I.2 STSC Technology Transition Approach

This section describes the STSC's approach to meeting the objectives identified in the previous section.

I.2.1 Technology Evaluation

The first technology transition objective involves identifying, validating, and classifying processes, methods, and technologies that can potentially improve the quality or productivity of software development and maintenance. Many organizations are so focused on deadlines and customer needs that they lack the resources and time to thoroughly investigate options for improvement, leaving them vulnerable to marketing hype. The STSC has developed the infrastructure to provide information on all types of applicable technologies. Product critiques, which are essentially brief evaluations from experienced technology users, are collected. Quantitative evaluations, which are detailed, comparable, and objective, are performed on the most promising tools, methods, or processes.

I.2.2 Information Exchange

This technology transition objective involves exposing potential customers to available technologies and, conversely, customer requirements to technology developers. Referring to the Adoption Curve, this objective focuses on contact, awareness, and understanding. STSC products that accomplish this objective include *CrossTalk* (a monthly

engineering journal), the annual Software Technology Conference, specific technology reports, and electronic customer services.

I.2.2.1 *CrossTalk*

Over 10,500 software professionals receive *CrossTalk* monthly. This publication provides a forum for the exchange of ideas. Articles cover leading edge, state-of-the-art, and state-of-the-practice processes and technologies in software engineering.

I.2.2.2 Software Technology Conference

The annual Software Technology Conference is held each April in Salt Lake City, Utah. This conference brings together over 2,000 software professionals from government, industry, and academia to share technology solutions and exchange ideas and information.

I.2.2.3 Technology Reports

STSC technology reports provide detailed information on specific software engineering technologies; and this report is an example. The current list of reports include:

- Test Preparation, Execution, and Evaluation
- Documentation
- Project Management
- Software Cost Estimation
- Requirements Analysis and Design
- Reengineering
- Source Code Static Analysis
- Software Engineering Environments

These reports provide awareness and understanding of each topic in preparation for evaluation and selection of corresponding technologies. Over 30,000 of these reports have been distributed.

I.2.2.4 Electronic Customer Services

Along with the services mentioned above, the STSC also provides customers with electronic access to information via Electronic Customer Services (ECS). ECS includes a bulletin board system which is available to obtain additional information, leave messages, add information, and confer electronically. In addition, a computerized database of practice, process, and technology information is coming on-line. ECS can be accessed via the INTERNET at address 137.241.33.1 or stscbbs.a1.mil or by calling 801-774-6509 with modem at 2400 or 9600 baud, 8 bit word, 1 stop bit, and no parity.

I.2.3 Technology Insertion Projects

STSC technology insertion projects are customer oriented projects that evaluate, select, and pilot the use of new processes, methods, and technologies for a specific customer. These projects can include process definition, process improvement, methodology insertion, tool insertion, and development of a technology road map. Referring to the Adoption Curve, Figure G-1, an insertion project helps cement understanding of a process or technology, tailors an evaluation of the process or technology for the customer, and pilots the use of that process or technology with appropriate levels of training. Customers move closer to adoption of the process or technology through hands-on experience. It is important to try out technology improvements in a pilot project to confirm that the technology is appropriate for the organization and that the organization is ready and able to adopt the new technology.

I.2.4 STSC Associates

Fowler and Przybylinski [Fowler 88] propose that transitioning new technologies from a developer to a consumer requires an advocate to push the technology and a receptor to pull the technology into an organization. This concept is illustrated in Figure G-2.

Effective change comes from within the organization. The STSC Associates' objective is to develop technology receptors within individual Air Force SDSAs. These receptors and STSC Associates, are trained in the use of the STSC's information, products,

and services to enhance their organization's ability to incorporate advanced practices, processes, and technologies.

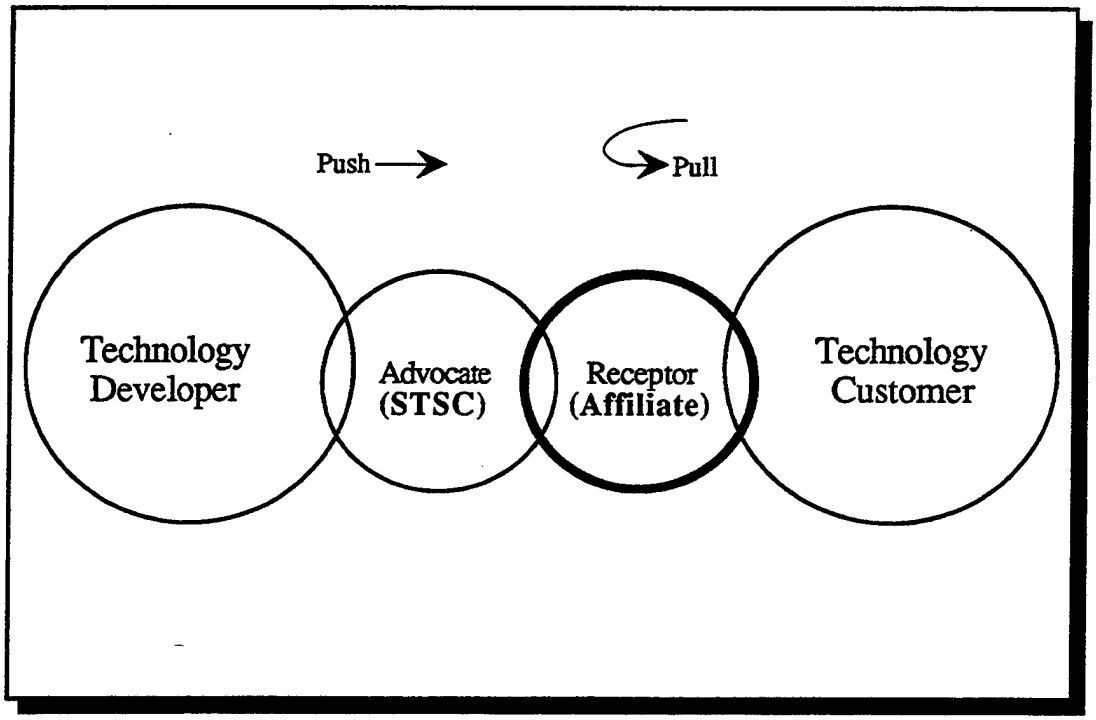

Figure I-2. Transitioning Technology

Referring to the Adoption Curve in Figure G-1, STSC Associates complete the trek to institutionalization. Associates coming from within the organization should be politically astute and aware of internal organizational requirements. They have the highest probability of influencing the adoption and daily use of effective business practices, processes, and technologies.

I.3 Embedded Computer Resources Support Improvement Program (ESIP)

The STSC operates out of the Ogden Air Logistics Center at Hill Air Force Base, Utah, under the direction and guidance of the ESIP Steering Group. An Air Force program, the ESIP has the goals of reducing the software backlog and increasing software quality and

productivity. Its mission is to provide an infrastructure to assist in the transitioning of technology to support all categories of embedded computer systems throughout the acquisition cycle and improve the readiness of Air Force weapon systems. ESIP is directed by an Air Force program management directive (PMD3118) and is led by a major command level steering group. The steering group had representation from the following organization: AFMC, AFSPACECOM, USSTRATCOM, ACC, AFOTEC. The voting members of ESIP are:

Col. Charles Fuller, DSN 458-2435, commercial 801-777-2435, fax DSN 458-9034

Capt. Mike Helsabeck, DSN 576-8189, commercial 618-256-8189, fax DSN 576-8939

Capt. Jonathan Liles, DSN 787-2151, commercial 513-257-2151, fax DSN 787-0841

Maj. Barbara Nelson, DSN 692-5054, commercial 719-554-5054, fax DSN 692-3350

Capt. Sean O'Connell, DSN 271-3278, commercial 402-294-3278, fax DSN 271-1020

Capt. Carl Scott, DSN 574-5700, commercial 804-764-5700, fax DSN 574-6060

Mr. Jeffrey Wiltse, DSN 246-5310, commercial 505-846-5310, fax DSN 246-5145