AD-A258 967

PROCESS DEFINITION AND MODELING GUIDEBOOK

DTIC
ELECTE
S JAN 6 1993 D
C

SPC-92041-CMC

VERSION 01.00.02

DECEMBER 1992

92 12 28 136

PROCESS DEFINITION AND MODELING GUIDEBOOK

DTIC QUALITY INSPECTED 8

SPC-92041-CMC

VERSION 01.00.02

DECEMBER 1992

CONTENTS

FIGURES

TABLES

This page intentionally left blank.

ACKNOWLEDGEMENTS

This page intentionally left blank.

1. INTRODUCTION

1.1 REQUEST FOR FEEDBACK

This guidebook is not intended to advance the state of the art. It does, however, use state of the art principles as a means for advancing the state of the practice. The Consortium's objective with this version of the guidebook is to provide sufficient tools, techniques, guidelines, and examples to enable you to acquire the necessary understanding and skills to capably perform *process representation*. It must be emphasized that a deliberate attempt has been made to stay with fundamental principles and techniques that can be easily learned and readily applied.

The Consortium is actively interested in end-user reaction, case studies, feedback, and any suggestions or recommendations you have for improving and advancing the content of this guidebook. While insights from researchers and technologists are also appreciated, this guidebook is designed to provide information useful to process *project managers*, process line engineers, and others whose daily activities are consistently constrained by crowded schedules and tight budget considerations. It is from this audience that the Consortium is especially interested in feedback.

Although this guidebook combines both theory and practice, the Consortium has consistently put the emphasis on practical considerations and practical application. The Consortium encourages you to apply the information presented herein, and strongly encourages you to let us know your thoughts on how to make this information progressively more useful to others like yourself.

1.2 SUMMARY PURPOSE AND SCOPE

The purpose of this guidebook is to provide a tailorable approach for process definition and modeling that an organization can use to accomplish process-related objectives. To achieve this purpose, this guidebook provides a flexible set of templates and techniques for capturing and representing *processes* and for representing the relationships and *constraints* between and within the artifacts, *resources*, and activities comprising the processes. Additionally, this guidebook provides explanation on how the templates and corresponding techniques can be used:

- As a common foundation for process analysis, design, development, and documentation.

- To facilitate the development of process-oriented guidebooks.

- To improve the usability of process-oriented guidebooks.

- To facilitate process training and education.

- To reduce the cost of developing process-oriented guidebooks.

- To create a defined process from a repeatable process.

- To create a repeatable process from the analysis of independently successful process activities.

- To facilitate process management.

- To facilitate process measurement.

- To facilitate process improvement.

- To facilitate process automation.

- To design and develop *process models*.

- As a migration path to and from process models based on existing notations (Structured Analysis and Design Technique [SADT], Entry-Task-Validation-Exit [ETVX] paradigm, State Transition Diagrams [STDs], etc.).

Although there are many aspects to the development, analysis, maintenance, and application of process representation, this guidebook limits its scope to the construction of process representations and their interpretation to gain process insights that facilitate the above uses. As indicated above, this guidebook provides introductory material on the conversion of template-based representations to other notations, and then briefly evaluates the relative advantages and disadvantages of these alternative notations with regard to *process modeling*. The notations discussed include:

- ETVX

- SADT

- STDs

- Petri nets

- Statecharts

- Process and Artifact State Transition Abstraction (PASTA)

- Role Interaction Nets

This guidebook includes several examples (in the appendixes) that illustrate a variety of approaches for constructing process models using different process notations.

One eventual goal of process definition and modeling is to construct models which are formal enough that they enable a process to be, literally, executed in an integrated, automated environment. This would also directly support *project management* activities through the development of project plans derived from the instantiation of one or more process models. Although this long-range objective is beyond the immediate scope of this guidebook, relevant issues are discussed and both manual and semi-automated alternatives to model-supported project management are presented.

1.3 MOTIVATION

An organization can be characterized by the resources it requires, the *products* it produces, and the processes it uses. Successful organizations typically become so through improvements in resource

usage, product quality, and process efficiency. Organizations are increasingly recognizing the fact that the quality of a product is a direct reflection of the quality of the process that creates it. The challenge has been, and will continue to be, for organizations to understand their processes and the effects of those processes on product quality and resource allocation. Consequently, the primary purpose of this guidebook is to provide you with *guidance* on how to describe your organization's processes.

Although improved quality, efficiency, productivity, moral, and profitability all depend extensively on process quality, the means for communicating about, analyzing, and improving organizational processes are often inconsistent or nonexistent within many organizations. Whether an organization wants to improve its process depends upon that organization's decision makers. However, once the decision is made to improve process quality, an organization still needs tools and techniques for performing the work. Therefore, another primary objective of this guidebook is to provide you with a set of tools and techniques that can be used to improve process analysis and communication, and thereby facilitate organizational process improvement.

Consequently, a key question forming the development of this guidebook was, "What is the best approach for communicating knowledge about an organization's processes and the practices and activities of which those processes are built?" The communication of knowledge traditionally takes one or both of two major forms: words and pictures. Words have the advantage of formality and *precision*, but the disadvantage of restricting you to serial communication. Pictures have the advantage of simultaneously communicating several related concepts and principles, but have the disadvantage of being too easily misinterpreted or misunderstood.

To improve communication, there needs to be a way of using both words and pictures to represent processes. This guidebook proposes an approach that strives to combine the best of both; however, to improve communication it is important to have a common understanding of terms. In the following material (and throughout the guidebook) *process definition* is considered to be the act of representing the important characteristics of a process in a way that facilitates understanding and communication. Process modeling both extends and constrains process definition by requiring the process model to adhere to a predefined set of objects, relationships, methods, and structural conventions. Process representation is a general term that refers to the combined or sequential efforts of jointly performing process definition and process modeling. From this perspective, your challenge becomes how to best represent your organization's processes via some combination of process definition and process modeling techniques.

1.4 GOAL-DRIVEN PROCESS REPRESENTATION

Process representation spans the entire effort spectrum from exceedingly easy to exceedingly difficult as a function of process complexity and desired level of detail. Before an organization undertakes process definition and modeling, it needs to clearly define the specific objectives that it expects. If the goal is to analyze the representation for process bottlenecks, this will influence the type of model that should be constructed. If the goal is to identify and remove process *redundancy*, to remove areas where the process remains largely undefined, or to seek insights into process risk, all these considerations influence not only the type of definition or model constructed but also the approach taken in researching and constructing those representations.

You should note that the magnitude of your goals must be directly related to the magnitude of organizational support for process representation. If the organization can only nominally support a

process representation effort, the goals that you strive for must be kept realistically and correspondingly small. Conversely, if an organization is prepared to give considerable support, then the goals of your process representation effort can be proportionately larger.

The key point is that it is important to allow your goals to drive your process representation effort. Initially, you may find that relatively simple diagrams of isolated parts of your process are sufficient to achieve your immediate goals. Later, you may find that you can achieve additional goals by extending your existing representation, adding more detail, and maybe capturing primary interrelationships between your growing inventory of "process asset" models. Still later, other goals may be achieved through even further expansion of scope, detail, abstraction, and information. It is important to note that this guidebook, and the tools and techniques it proposes, have been designed to facilitate precisely this type of goal-driven, incremental approach to process definition and representation.

The Consortium believes that it is important for organizations to have a modeling option which attempts to combine the strengths of text- and graphic-based representations while minimizing their respective weaknesses. The templates and associated graphical conventions described in this guidebook are specifically designed to support process representation, including both process definition and process modeling. The templates have fields for text-oriented descriptions of activities, pre- and post-conditions, internal processing, comments, and revision history. However, through a variety of relationships they also convey an explicit architecture directly supporting graphical rendering and analysis. It is believed that this combined template- and graphically-based approach provides you greater opportunity for the optimal combination of both text and diagrams toward the cost-effective development and use of process representations.

1.5 THE NEED TO MOVE TOWARD MODELS

Existing examples of process representations include *policy*, procedure, and operational manuals developed by organizations to inform and guide their employees in the performance of their responsibilities. Most organizational process guidebooks only define the process, and few make use of process models. Those that do, often use relatively high-level or simple models. Although process modeling is a comparatively rare technique for representing organizational processes, it is a well-known and mature technique for representing processes implemented by computer systems. Examples include Statecharts, SADTs, and the ETVX paradigm.

Due to fundamental parallels between defining and modeling organizational processes and computer processes, many techniques from computer process representation can be applied to organizational process representation. Similarly, many of the advantages and benefits derived by building computer process representations can also be derived from organizational process representations. As highlighted below and elaborated throughout the remainder of this guidebook, the Consortium believes that an organization can achieve significant benefits through the proper use of both process definition and process modeling techniques.

Science and industry have long recognized the value of models. A model enables you to gain detailed insights into important, and sometimes crucial, characteristics of whatever is being modeled. The risks of building flawed models are usually less than the risks of proceeding with flawed plans, products, or practices. The costs of recovering from the flaws, correcting the problems, and eventually achieving an acceptable model are significantly less than attempting such efforts on something already in production or in practice. Additionally, following something that is well defined is far preferable—and

1. Introduction

far more successful—than attempting to follow or adhere to something which is poorly defined or not defined at all.

This is where models are especially useful. Models can communicate key component objects and organizational relationships, highlight key aspects of the process, and abstract out information not crucial for the perspective being presented. In practice, however, even process models have proven to be difficult to develop and maintain, especially for large-scale processes. Modeling notations (such at SADT, Petri nets, and Statecharts) have traditionally been used to represent processes occurring within a computer. These same notations can be used with greater or lesser degrees of success for constructing corporate or organizational process models. However, due to the inherent volatility of organizational processes (based in large part on the need to respond progressively faster to market forces) even these models can be inadequate and of questionable value. As with documentation, the model must reflect reality. If reality advances but the model (or documentation) does not, its relevancy and its value drop drastically.

This guidebook presents a combination of tools and techniques for leveraging the best that is offered by process definitions and models while minimizing their less advantageous characteristics. This template-based approach can be used for both defining and modeling a process. These templates can be used, for instance, to succinctly capture organizational process policies, procedures, and operational guidelines. Additionally, by using the graphical conventions provided in Section 3.3, you can use them to construct graphical process models.

Since processes and objectives can vary greatly between different organizations, the template-based process representation technique presented in this guidebook is specifically designed for flexibility and tailorability. Guidelines are presented in Section 4 for modifying this process representation technique to accommodate site-specific objectives and to facilitate site-specific, cost-effective implementation and application.

1.6 GUIDEBOOK ORGANIZATION

Figure 1-1 shows the guidebook's organization and the possible paths you can follow.
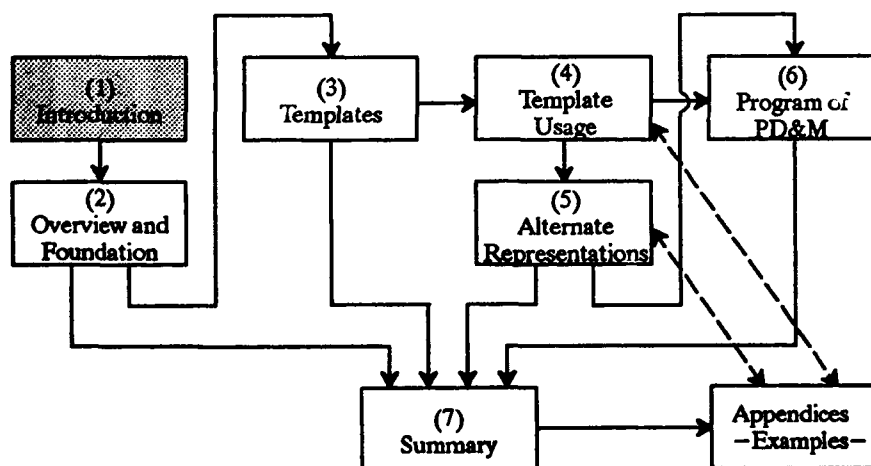
Figure 1-1. Guidebook Organization View 1

Outlined below is a summary of information presented throughout the remainder of this guidebook:

- Section 2, Overview and Fundamental Concepts, provides an overview of the value of and techniques for process definition and modeling. Section 2 starts with a brief discussion on the need for representing processes. Section 2.2 provides an overview of defined processes and process maturity. Section 2.3 provides a brief introduction to alternative process modeling notations, and Section 2.4 examines common characteristics in process representation. Section 2.5 discusses how you choose a process representation notation as a function of comparative analysis of key characteristics with regard to your specific environment and goals. Next, in Section 2.6, there is a brief examination of the benefits to using the template-based approach presented in this guidebook. Section 2 concludes with a basic introduction to key concepts, terms, and definitions relevant to the domain of process representation. As shown in the diagram, after Section 2 the casual reader may want to directly go to Section 7.

- Section 3, Process Definition Templates, provides:

  - The rationale behind the selection and use of this particular set of process templates.

  - The process templates and discusses how they can be used (Section 3.2).

  - The graphical extensions used to diagrammatically depict template-based process descriptions and example process diagrams highlighting a variety of graphical characteristics and advantages (Section 3.3).

- Section 4, Template Usage, examines using templates to facilitate the development and analysis of process guidebooks and it expands upon the domain of use and applicability by discussing specific characteristics of process models. Section 4.3 examines process models from the key perspective of representative power. Section 4.4 changes to a practical orientation, and develops a detailed examination of a suggested approach for using templates to construct both text and graphic-based process models. Section 4.5 discusses how you can improve the usability of template-based process models both by improving data presentation and by modifying or tailoring the templates to site-specific needs. Section 4 closes by examining how you can potentially use the templates to facilitate organizational process improvement. As is shown in the diagram, it is suggested that if Section 4 is read, Section 5 should likewise be read (before jumping to Section 7).

- Section 5, Alternative Process Notations, commences with a discussion on process objects and relationships which bind those objects. A process meta-model is constructed to highlight process objects represented by the templates, and legal relationships between those objects (Section 5.1). Section 5.2 examines various opportunities and techniques for supplementing template-based representations with other notations, and vice-versa. ETVX, SADT, State-charts, Petri nets, PASTA, and Role Interaction Nets are each described and evaluated.

- Section 6, Process Representation Programs, discusses issues, approaches, and options relevant to commencing and continuing a program for process representation. Section 6.1 presents material on introducing process definition and modeling into your organization. Section 6.2 introduces the subject of metrics and discusses how process definition can support a program of process and product measurement. Section 6.3 provides an overview description of the Evolutionary Spiral Process from the perspective of process representation. Section 6.4 discusses the use of process representation as a tool for process management, and Section 6.5

looks beyond conventional process management approaches and discusses the support process representation gives to automated process management within highly integrated environments. Section 6.6 provides a brief summary of this material.

- Section 7, Summary, briefly highlights the key principles in the guidebook, and presents a summary of future work planned to appear in the next release of this guidebook.

1.7 INTENDED AUDIENCE

The primary intended audience for this guidebook is practitioners interested in the tangible benefit derived from using process representation tools and applying process representation techniques. This audience includes line engineers, project managers, and anyone working on or interested in the area of process analysis, design, development, or improvement. It is assumed that most readers are not necessarily familiar with the theoretical issues and aspects of process representation; therefore, motivation and rationale are considered important and are provided throughout the guidebook.

1.8 TYPOGRAPHIC CONVENTIONS

This report uses the following typographic conventions:

Serif font . . . . General presentation of information.

Italicized serif font . . . . Words, expressions, abbreviations, and acronyms found in the Glossary, mathematical expressions, and publication titles.

Boldfaced serif font . . . . Section headings and emphasis.

This page intentionally left blank.

2. OVERVIEW AND FUNDAMENTAL CONCEPTS

This section provides an introduction to the fundamental concepts of process representation as they relate to process definition and modeling. This section discusses:

- Motivation for process representation.

- The relationship between process representation and process maturity.

- Alternative techniques for process representation.

- Characteristics for comparing alternative approaches for process representations.

- Considerations when choosing an approach to process representation.

- Benefits of the template-based approach proposed in this guidebook.

- An overview of process-related terminology. .

As you can see in Figure 2-1, after completing this section you may elect to skip Sections 3 through 6 and complete your review of this material by reading Section 7. This approach provides the essentials needed to discuss and understand the key concepts of process definition and modeling. For a more in-depth understanding, continue with Section 3.
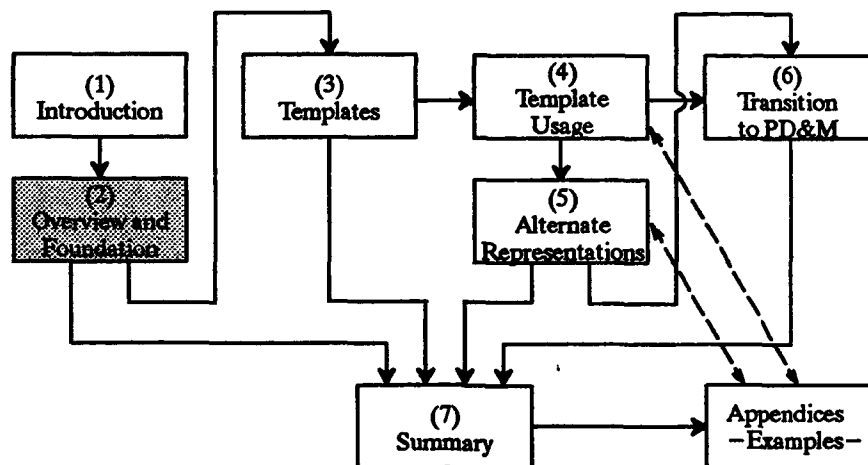
Figure 2-1. Guidebook Organization View 2

2.1 THE NEED FOR REPRESENTING PROCESSES

Process representations satisfy four key needs:

- To do successful work.

- To preserve lessons learned.

- To facilitate training.

- To analyze and compare existing and proposed processes.

2.1.1 Do Successful Work

An organization needs the ability to repetitively and consistently do successful work. Not only is it necessary for one division or group to successfully do their work repeatedly, but it may also be important for multiple divisions or groups to simultaneously perform the same type of work in the same way. Having a defined process directly supports having a repeatable process. Although work may be successfully repeated in the absence of a defined process, such repetition depends entirely on the efforts of individuals. Conversely, once a process is defined it can become institutionalized and preserved by the organization as part of its corporate legacy.

2.1.2 Preserve Lessons

Another key advantage to representing a process is the preservation of lessons learned and insights gained from the performance of that process. Typically, processes evolve. People involved in either managing or participating within the process often find a variety of ways in which to improve how they perform their work. Unless a process is defined, there is an increased risk that as individuals move to work in other areas, their process improvement insights move with them. By having a representation of a process and by updating that representation to reflect process improvements the insights and lessons learned through the actual performance of a process are preserved as part of the corporate legacy.

2.1.3 Training

Another key use of process representations is to facilitate training. Explaining to new employees the process they should follow is one means of educating them, but the results can be inconsistent and, in many ways, unpredictable. Training employees in their respective processes improves the likelihood that they understand the key characteristics of the process they should be following, which can more quickly lead to higher employee productivity and efficiency. Note that it is not the process representation itself that yields these benefits, but the training opportunities that result from having a representation of the process.

2.1.4 Analyze and Compare

Finally, one of the most important key uses of process representation is to facilitate the analysis and comparison of both existing and proposed processes. In the absence of process representations, detailed analysis of existing or proposed processes is difficult, if not impossible. All discussion on the process would be based on individual experience and opinion. Typically, there needs to be a consensus on what the process is before the focus can be turned to how the process can be improved. Without a process representation, valuable time and energy are lost while analysts attempt to reach agreement on their differing conceptual impressions of the process. By defining and potentially modeling a process, the key characteristics of the process are communicated more consistently and less ambiguously. This tends to minimize confusion about what the process is and frees the process analysts to focus upon how to analyze and improve that process.

2.2 DEFINED PROCESSES AND PROCESS MATURITY

Process improvement is now commonly considered to be a function of increasing process **maturity**. Process maturity, especially the software engineering process, is an area of extensive *research* at the Software Engineering Institute (SEI) at Carnegie Mellon. One of the cornerstones of the SEI software process maturity model is that software process maturity can be evaluated in terms of the following five levels (Humphrey 1990):

- Initial

- Repeatable

- Defined

- Managed

- Optimizing

An organization can gain insights into its process and its opportunities for process improvement by evaluating its process maturity.

- *Initial Level.* The lowest level of process maturity is reflected by initial or ad hoc processes. At this level, processes are generally chaotic. They are essentially ad hoc processes characterized by managers that react to situations as opposed to managers who attempt to direct situations. Successful *projects* within this type of environment typically owe their success entirely to the efforts of individual people—people who succeed in spite of the process, not because of it. Key problem areas which, if addressed, can facilitate the processes' advance to the next level of maturity include efforts to improve project management and planning, configuration management, and initial steps toward software quality assurance.

- *Repeatable Level.* The next higher level of process maturity is characterized by repeatable processes. Repeatable processes typically include project plans, sizing and estimation measures, productivity factors, scheduling and project tracking, planning models, software configuration management, and quality assurance efforts. However, processes at this level still depend extensively on the efforts of individuals, and people are sometimes successful only because they intuitively understand the problem domain, and not because of any special process or management support. Key areas of improvement include training, increased use of technical quality evaluation practices (such as reviews, walkthroughs, inspections, etc.) and an increased focus on the process itself.

- *Defined Level.* Defined processes are characterized by software standards and guidelines, software inspections and reviews, and more formalized testing (including test plans, test support tools, and methodologies). At this level, there are one or more ongoing software engineering process groups that work together to consistently improve the quality and maturity of the processes in use. Areas for improvement include process measurement, process analysis, and quantitative quality plans.

- *Managed Level.* The managed process is characterized by process data gathering, especially by management's response to insights derived from the analysis of such data. To be useful, the data must contribute to one of the four following areas:

- Understanding

- Evaluation (compliance with criteria)

- Control

- Prediction

Also critical to this level is the application of the collected data to such areas as quality motivation, quality estimation, quality goals, quality plans, and general tracking and control of software quality. Key areas where further process improvement efforts can be focused include changing technology, problem analysis, and a special focus on problem or defect prevention.

- *Optimizing Level.* Finally, the optimizing level is generally characterized by efforts in which improvement insights are directly translated into process changes. These changes include efforts toward defect prevention, improved development environments, and an overall trend toward automating the software process. Note that at this, the highest level of process maturity as defined by SEI, process enactment, monitoring, and enforcement are still principally a human-intensive and largely manual process. Consequently, improvements at this level may be achieved by relegating progressively greater process-oriented responsibilities to **automatic** execution, analysis, and enforcement.

Using these levels of process maturity, summarized in Table 2-1, you can see that process representation directly supports the second (repeatable) level of process maturity, and is critical at level three (defined). Developing a repeatable process is directly facilitated by having a common definition of the process to be repeated and by using that definition for training, reference, and as a guide to project management. Having a repeatable process that has been both defined and institutionalized is a cornerstone to reaching and exceeding the third level of process maturity. Because each level of maturity above the first level augments the practices of the preceding level, having a defined process remains crucial for organizations seeking to or manifesting process maturity at the fourth and fifth levels.

Table 2-1. Software Engineering Institute Process Maturity Level and the Problem Areas

Level	Attributes	Problem Area
(5) Optimizing	Improvements fed back to process	Automation
(4) Managed	Quantitative — Measured process	Changing technology Problem analysis Problem prevention
(3) Defined	Qualitative — Institutionalized	Process measurement Process analysis Quantitative quality plan
(2) Repeatable	Intuitive — Depends on individuals	Training Technical practice Process focus
(1) Initial	Choice — Controls and plans ineffective	Project management Configuration management Software quality assurance

Deciding to define (or represent) a process is only the first step. The next step is developing an understanding of the various options, methods and alternative techniques available by which processes can be represented.

2.3 ALTERNATIVE TECHNIQUES FOR PROCESS REPRESENTATION

The most common technique for representing or defining processes is the use of descriptive text. Essentially, all operations manuals are text-based representations that describe one or more processes. Organizational policy and procedure manuals are also potential sources of process descriptions. One significant advantage to text-based descriptions is that there are virtually no constraints placed on the description itself. Since the notation is the entire language of words, anything that can be talked about, can be described. This, however, is also the most significant disadvantage to text-based process descriptions Since there are no constraints (other than grammar) on the structure of the description, there is the potential for considerable inconsistency, ambiguity, uncertainty, and inaccuracy. Text-based descriptions are typically the least formal type of process representation.

At the other end of the formality spectrum is mathematics. Although mathematically provable process models are an ideal goal, such methodologies are still an area of active research, and there are not any clearly cost-effective and widely applicable math-based methodologies. However, between the informality of descriptive text and the rigorous formality of mathematics, there are many options for representing processes at varying degrees of formality.

Having graphical support to a notation can directly contribute to higher degrees of formality. Usually, a graphically-oriented notation comes coupled with a methodology that imposes rules on placement, use, and connections between the graphical objects. These rules constrain the types of structures that can be built and increase the formality of the resulting depictions. Additionally, graphical depictions are especially useful for portraying abstract or high-level relationships. The resulting diagrams allow for insights into the process that can be virtually impossible to derive from descriptive text.

Fortunately, there are enough parallels between the need to represent human or organizational processes and the need to represent software-based processes that many of the techniques and notations developed for use in software engineering can be extended for use in general process representation. From a software perspective, any software program can be written using five basic constructs:

- Sequence

- Selection

- Iteration

- Dispatch

- Rendezvous

These are exactly the same constructs needed to represent organizational processes. Any type of process can be represented if you have:

- Some means for indicating a series of *events* occurring in sequence.

- A selection between two or more events.

- The repetitive execution of one or more events.

- The parallel initiation of two or more events.

- The synchronization of two or more events executing in parallel.

2. Overview and Fundamental Concepts

Examples of software-based techniques that can be used in general process modeling include STDs, ETVX, SADT, Statecharts, and Petri nets. These and similar alternative notations are examined in detail in Section 5 and are only briefly introduced here.

2.3.1 STATE TRANSITION DIAGRAMS

STDs are often used for describing finite automata (finite state machines). Any process that can be described in terms of a finite automaton can be represented using an STD. Generally, a finite automaton accepts some series of input symbols and typically produces some series of output symbols (Kolman and Busby 1984). Each such output symbol is a function of the relevant input symbol, the current state of the automaton, or both. Additionally, as input symbols are received by the automaton, its state may change. With regard to process modeling, input symbols and output symbols typically represent events occurring in reality, and states within the machine represent different milestones or activities within an overall process. Thus, finite state machines can be seen as one possible representation for modeling sequences of events within some defined domain.

2.3.2 ENTRY-TASK-VALIDATION-EXIT

The premise behind the development of ETVX was the necessity to find a means of embedding methods and tools into a common framework for intellectual and management control (Radice and Phillips 1988). ETVX (see Figure 2-2) is a quasi-diagrammatic representation of IBM's Programming Process Architecture (PPA). The authors of ETVX take the position that PPA is the highest representation of the software process; and that although it contains the necessary elements for representing software engineering environments and activities, it also is applicable across a much more broad framework (Radice and Phillips 1988). Therefore, it holds promise for use in general process definition and modeling.

2.3.3 STRUCTURED ANALYSIS AND DESIGN TECHNIQUE

When applying the SADT to software systems, the overall approach consists of identifying activities then identifying the inputs and outputs of those activities, identifying factors that constrain the activities, and identifying resources or materials that support the activities (Marca and McGowan 1988).

As Figure 2-3 depicts, activities are represented diagrammatically as boxes. Inputs to an activity are labeled arrows arriving at the left side of the box. Outputs from an activity are labeled arrows departing from the right side of the box. Constraining influences are labeled arrows arriving at the top of the box, and enabling mechanisms are labeled arrows arriving at the bottom of the box. This approach has both a simple methodology and a clear graphical convention. Consequently, process definition and modeling can readily be based upon SADT.

2.3.4 STATECHARTS

Statecharts are an extension of the basic notation used for finite state machines. Statecharts allow a finite automaton to be decomposed into a representation which models two or more interacting or communicating subsystems. Statecharts also support hierarchical decomposition of transition diagrams so that various levels of abstraction can be independently represented (Sanden 1992b). Figure 2-4 shows an example of Statecharts adapted from Marc Kellner's SEI technical report (Kellner 1989). The top portion of this diagram represents a functional perspective of a process while the bottom portion represents a behavioral perspective.

2-6

Modified ETVX Paradigm

- - - - - Controlled by Activity Management

Instantiate with methods
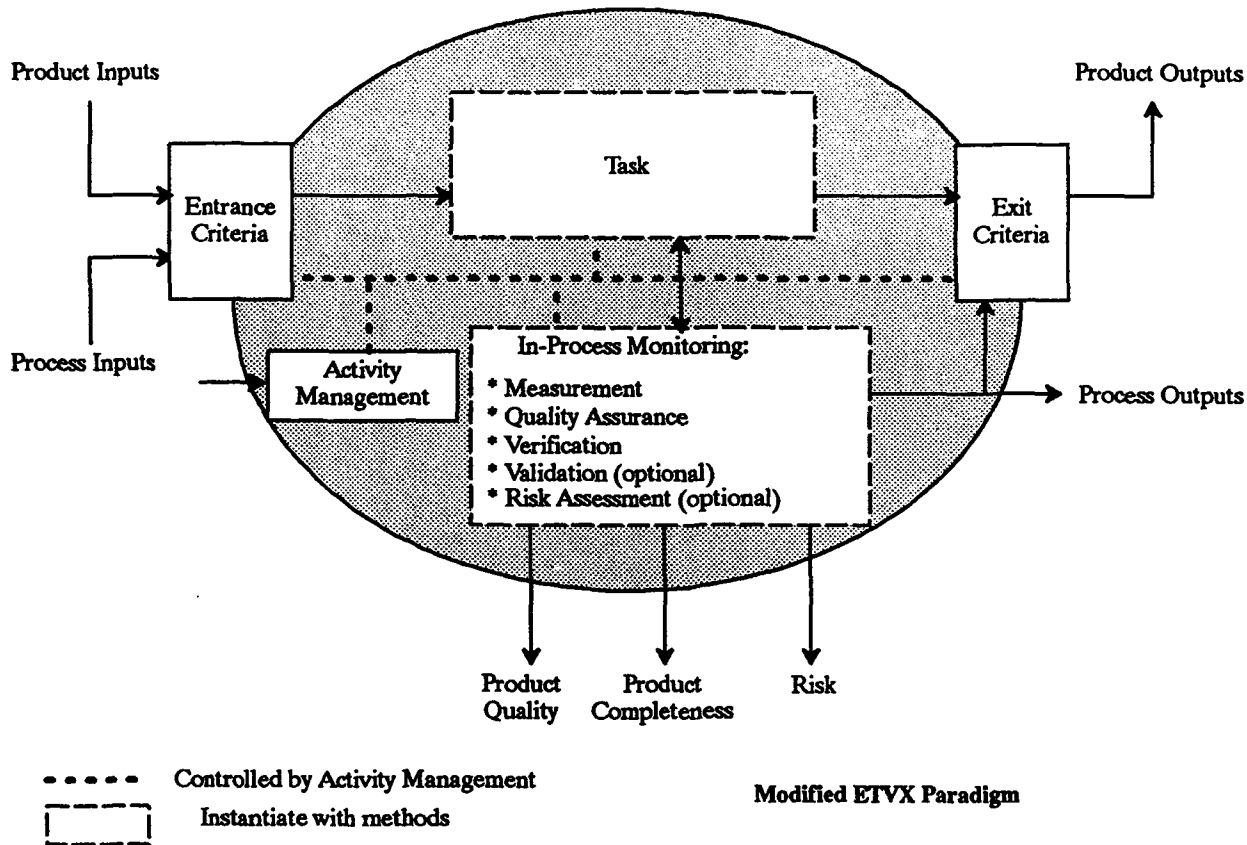
Figure 2-2. Entry-Task-Validation-Exit Diagram
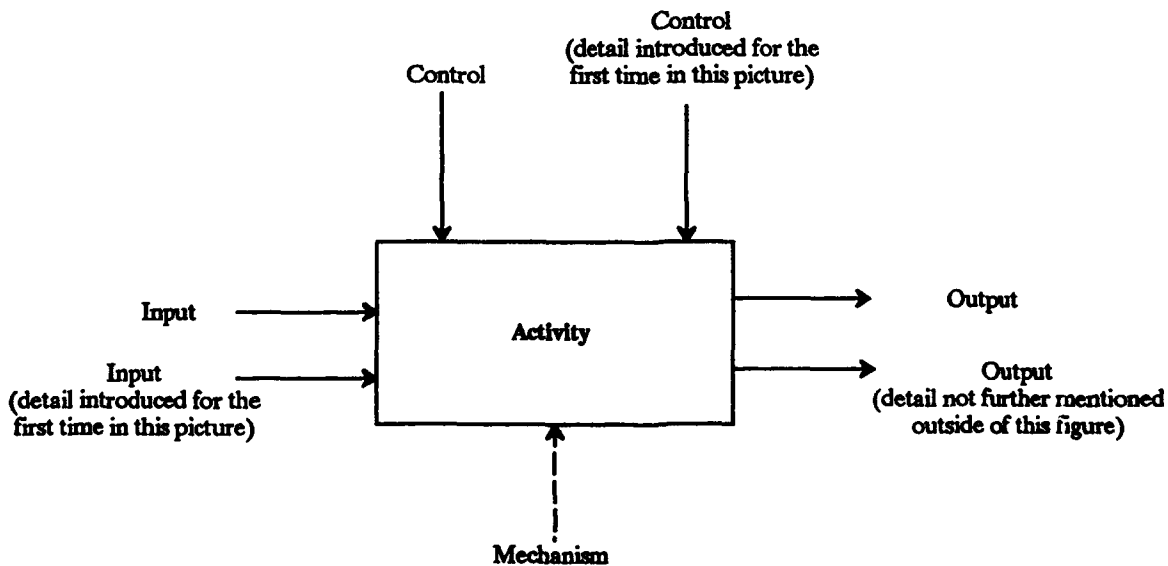
Figure 2-3. Structured Analysis and Design Technique Diagram

Though Statecharts typically require a considerable level of detail, their formality allows you to construct process depictions that can be readily converted to executable instructions. Consequently, Statecharts have excellent potential for providing a means to gain insights into process dynamics.
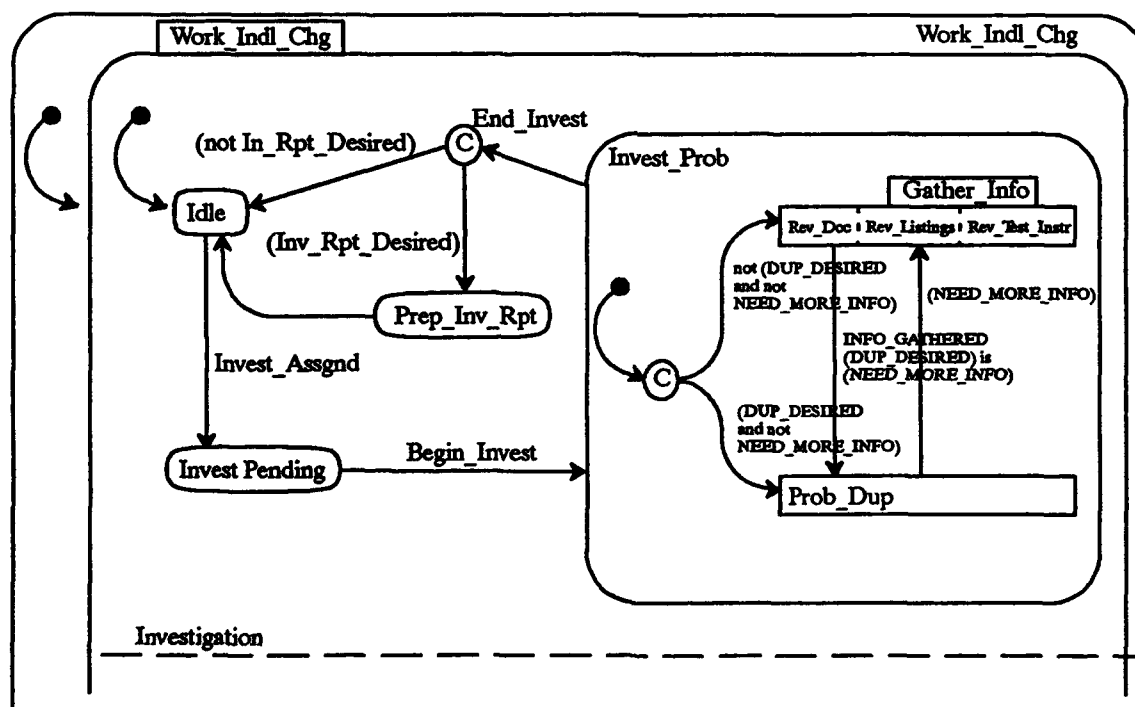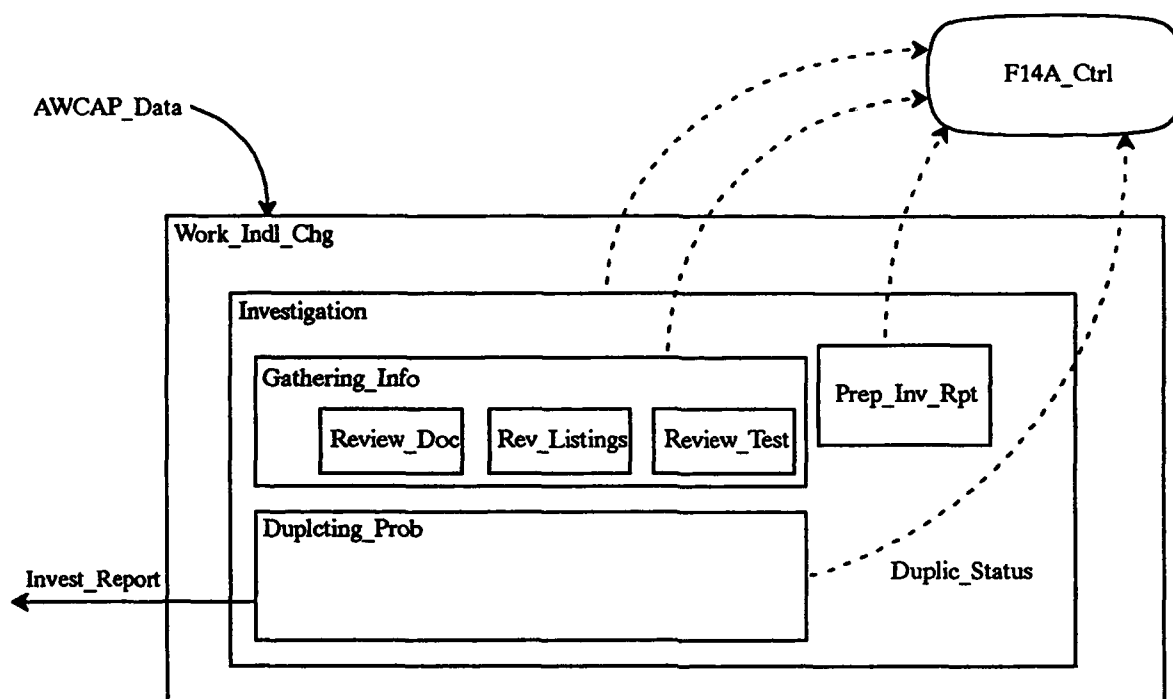
Figure 2-4. Statechart Example 1

2.3.5 PETRI NETS

Petri nets (a variation is shown in Figure 2-5) are becoming progressively more widely used as a means for building a representation of a wide variety of processes. Petri nets have been successfully used to model manufacturing processes, chemical processes, and hard real-time embedded processes. One

of the most important characteristics of Petri nets is the fact that they capture the dynamic behavioral characteristics of the system being modeled. In effect, Petri nets can be executed.
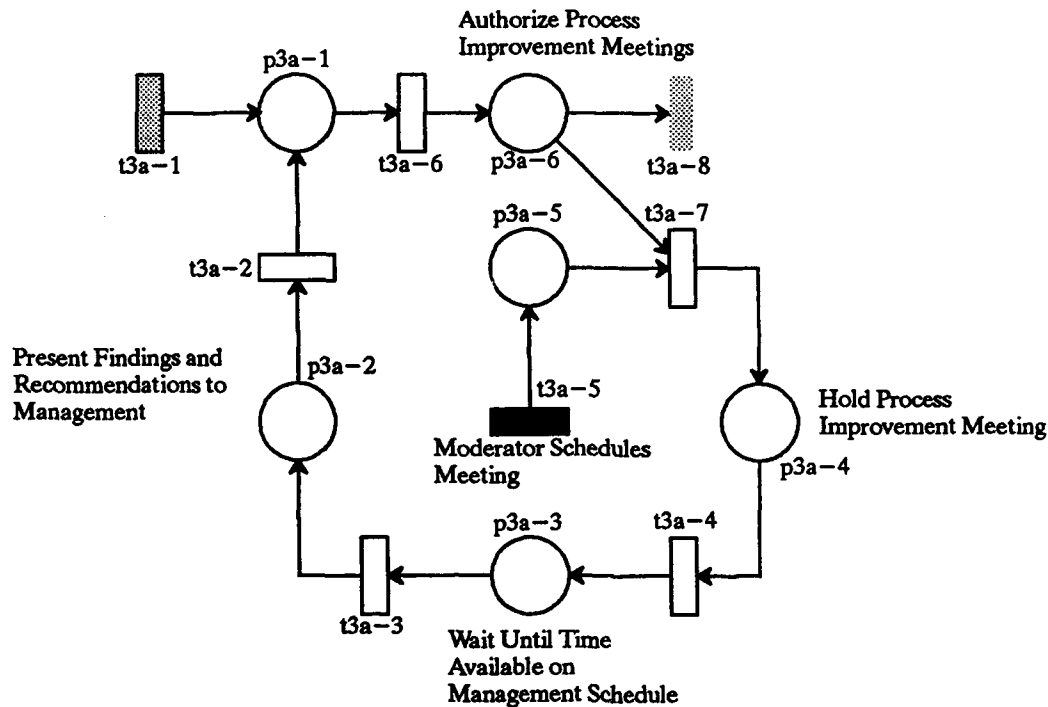
Figure 2-5. Petri-Type Net

In addition to the graphical notation, Petri nets also come with a significant body of mathematical formalism. By relying on the mathematical substructure of these diagrams, it is possible to do a static structural analysis of the system's dynamic behavioral characteristics without resorting to actually running a simulation. This is of key importance since it allows formal interpretation and analysis of a process model for both desirable and undesirable characteristics (Levis 1992).

Another key factor contributing to the facility of Petri nets is its graphical nature. The basic graphical representation principles are conceptually simple to learn and understand, and yet they can be used to build detailed representations of complex systems or models.

2.3.6 TEMPLATE SUPPORT

Other notations exist (and new notations are being developed), but this set of notations represents some of the most commonly used tools for representing software processes. Again, Section 5 will examine how these notations might be used to specifically model organizational processes. Nevertheless, one obvious question now is with this variety of notations already available, why are templates even needed?

Several advantages to the template-based approach were presented in Section 1, and further benefits are examined in Section 2.6. However, one additional use for the templates is their direct support of any of these alternate notations. Templates can be used as an analytical tool to facilitate the organized collection of information that will later be used to develop process models in SADT, ETVX, etc. Furthermore, due to their compatibility with and use of descriptive text components (in the template fields) the templates are capable of capturing more information about a process than might otherwise

be captured using an alternative notation. From this perspective, if a process representation already exists in the format of an alternative notation (such as ETVX, SADT, etc.), the templates can be used to augment or extend that representation by including additional information and relations that the templates provide.

To summarize, there are several alternative process representation notations examined in this guidebook, and all are compatible with the proposed set of templates. Although the templates can be used as a stand-alone technique for representing a process, they can also be used prior to, in conjunction with, or subsequent to alternative notations such as SADT, ETVX, Statecharts, Petri nets, etc. To determine which approach or notation to use, it is important to discuss common characteristics of process definition and modeling.

2.4 COMMON CHARACTERISTICS IN PROCESS REPRESENTATION

Process notations always imply some type of methodology by which the notation is used. When comparing alternative notations and methodologies, relative advantages and disadvantages can be highlighted by contrasting the various approaches from the perspective of key characteristics. The key characteristics described in the following subsections include:

- Scalability

- Applicability

- Flexibility

- Readability

- Maintainability

- Learnability

- *Robustness*

- Relative formality

2.4.1 SCALABILITY

Scalability represents the degree to which a notation can simultaneously tolerate both abstraction and details. Some notations may be especially good at representing high-level abstractions of a process, but rather inadequate at capturing process details. Other notations may excel at representing details, but may suffer severe "scale-up" problems when it comes to representing high-level views of the process. An ideal characteristic for a notation is to efficiently provide both abstract views and detailed views of a process.

2.4.2 APPLICABILITY

A notation can be more or less applicable to the specific needs of a given domain of processes. For example, some notations may be quite poor at representing parallelism; others may be quite poor at representing the people involved in a process; and others may provide only a static view of a process

but no insights into dynamic characteristics. If, for example, parallelism, people, or process dynamics are important to the domain of processes being represented, it is ideal to select a notation that can capture and represent these and similar details important for and applicable to your specific domain.

2.4.3 FLEXIBILITY

Flexibility is the degree to which a notation can be altered and customized to better achieve specific objectives. Note that this is a different quality than maintainability (described in Section 2.4.5). Flexibility is the characteristic of being adaptable outside of a specific domain. A notation is relatively flexible to the degree that it can be tailored for your specific needs.

2.4.4 READABILITY

A process notation is relatively readable to the degree that it allows a reader to quickly, clearly, and accurately derive insights into the described process. An additional consideration with readability is the amount of training or expertise required before a notation becomes readable. Complex mathematical expressions may require years of relevant education before they can be accurately interpreted; whereas, a simply flow-chart type approach may convey meaningful information even to the uninitiated.

2.4.5 MAINTAINABILITY

A process notation is more or less maintainable as a function of the ease with which changes can be made to existing representations. Process improvement implies process change. Any process definitions or models that you use to document the existing process will, therefore, need regular updates. The ease with which such updates can be done is a direct reflection of the maintainability of that notation. Although maintainability is affected by, for example, readability, flexibility, and learnability, it is a separate consideration. Just because something is readable, flexible, and learnable does not guarantee that it is maintainable.

2.4.6 LEARNABILITY

The learnability of a process notation is derived from three factors:

- The average level of expertise needed by individuals before they can receive training in using the process notation.

- The average amount of training required before they are capable of using the notation.

- The average amount of time it requires a trained person to transition from just being capable to actually being proficient and adept at using that notation.

A highly learnable notation requires little or no related expertise, perhaps a day or two of training, and it allows process engineers to become proficient in a few weeks or months of actual usage.

2.4.7 ROBUSTNESS

A process notation is robust if it can be used "as is" on a comparatively larger set of different processes. Note that higher robustness can, to some degree, compensate for a lack of flexibility. There is less need

for a notation to be tailorable for different uses when that notation is usable in standard form across a wide variety of applications. Likewise, a low level of robustness in a notation can be offset by a relatively high degree of flexibility.

2.4.8 FORMALITY

The formality of a notation is a combination of a variety of factors. At a minimum, relative formality is reflected by the degree to which a notation is unambiguous and deterministic. Formality increases to the degree that it constrains the set of elements with which processes can be described, and it defines the operations and transformations permissible on those elements. An ideal level of formality for a process notation is one that allows process representations or models to be machine compilable, linkable, and executable. Even more formal are "provable" notations where it can be proven (using mathematical techniques) that the design representation is consistent with and complete against an analysis representation, and the enactment representation is consistent with and complete against the design.

2.5 CHOOSING A PROCESS REPRESENTATION NOTATION

When deciding how to initiate a program of process representation, the selection of a process notation is of central importance—it is the language you will be using to communicate about your processes. The key characteristics described in Section 2.4 are important for guiding your thinking in choosing a process representation notation. However, primary considerations when choosing an approach to process representation depend upon your relative comparisons of characteristics that you consider to be important.

Remember that all notations imply some type of methodology by which the notation is used. When considering alternative process notations, you need to simultaneously consider the associated methodology. Both the notation and its accompanying method of use determine its relative scalability, readability, and robustness.

Some of the questions you will need to answer include:

- Can the notation represent details and also abstractions? (Scalability)

- Can the notation be used to represent your processes? (Applicability)

- Can the notation be adapted for use in representing different processes? (Flexibility)

- Are process descriptions resulting from this notation easily interpreted? (Readability)

- Can process depictions be easily updated to reflect changes? (Maintainability)

- Can average people become quickly competent without extensive training? (Learnability)

- Is the notation capable of representing a large variety of processes? (Robustness)

- Does the notation yield machine-interpretable process models? (Formality)

There are no single answers to these questions. In all cases, the answers depend on several important factors relevant to your organization. As discussed in Sections 2.5.1 through 2.5.5, these factors

include the type of environment you have, the resources available, budget constraints, history, and your immediate and future goals.

2.5.1 ENVIRONMENTS

Environments vary significantly between organizations, and they can even vary significantly between divisions within a single organization. From the perspective of process representation, one major consideration is the relative volatility of your environment. Are the processes you intend to model relatively stable or highly dynamic? Are the changes relatively nominal, or are they often radical? Are the changes predictable, or are they often unexpected? Additionally, is the environment dedicated to a single domain of processes, or are there processes occurring within a variety of domains? Are the processes of relatively short duration, or are they long-term? Are the processes highly iterative, or typically nonrepeating? Is management largely a human-intensive activity, or is there a large degree of automated project management support? These are examples of environmental considerations that can strongly influence which process notational characteristics are applicable to your specific site.

2.5.2 RESOURCES

Another consideration is the resources you have available to perform process analysis, design, and representation. From the perspective of people, do you have highly trained or experienced process engineers already available or will you need to have people trained? Are you anticipating a large initial effort (involving dozens of people), or will you initially involve only a few people? Will your process engineers be full-time, dedicated resources; or will process-related work be a matrixed responsibility added to their current responsibilities? Are you intending to use the process representations to support project management in a automated, integrated environment? If so, do you currently have the tools and techniques needed to support that approach, or will you have to acquire them?

2.5.3 BUDGET CONSTRAINTS

Budget constraints are another major factor. How much funding is available to you for initiating the process representation effort? After start-up costs, how much funding is available to support the ongoing program? Is immediate cost-justification important, or can the costs be amortized over a longer period of time?

2.5.4 HISTORY

Organizational history must also be considered. Is there an existing repository of process representations? If so, are they rendered in one or multiple notations? Is there an existing experience pool for the performance of process engineering? Was process engineering an effort previously done but canceled long ago or has it been an ongoing effort? Is the staff allocated to process engineering being held relatively constant, or is it being significantly increased or decreased?

2.5.5 GOALS

Finally, and most importantly, you must consider both the organization's immediate and long-term goals. As emphasized earlier, it is important to take a goal-driven approach to process representation. "Improved process maturity" is too ambiguous a goal to help answer many of the questions presented

here. Instead, explicit and well-defined goals are needed. Is the goal to develop a process guidebook? If so, is the guidebook intended to be a technical reference or a self-teaching tutorial? Is the goal to develop a guidebook of a **proposed** process that can then be distributed to reviewers and process analysts? Is one of the goals to develop material that will facilitate training employees in organization processes? Is one of the goals to have a representation that facilitates gaining insights into dynamic and static process characteristics? Does the representation need to facilitate developing automated process models that can be executed? Is the goal of process model execution to gain insights into performance issues such as deadlocks and bottlenecks? Is it a goal to use the representations to facilitate project management?

2.5.6 SUMMARY

As stated, the answers to these questions vary from one organization to another; and even within the same organization, the answers will change with time. Nevertheless, there is a general mapping that can be made between the key characteristics of alternative notations and organizational issues:

- Your environment

- Available resources

- Budget constraints

- History

- Your immediate and future goals

Examining these relations will help you select your approach to process representation.

If your environment is volatile, then a notation with a high degree of maintainability is desirable. Ideally, as your process goes through significant change, the notation and methodology you have chosen will remain usable. If you will be defining and modeling processes across multiple domains, robustness and flexibility become important. You will need a notation that is either capable of handling a variety of processes or one that can be easily adapted to your changing needs. Conversely, if you will be targeting a single domain, applicability is of key importance. Long-term processes imply a need for a notation that exhibits a high degree of maintainability; and highly integrated, automated environments may convey a need for greater formality in the selected notation.

The type of available resources also affects your choice of a notation. If you anticipate a high degree of turnover in the personnel performing as process engineers, then you will need a notation that has a high degree of readability and learnability. Conversely, if many of the process engineers are already familiar with a particular notation, then learnability is less important—your resources already have much of the fundamental training and experience.

If you must work within a very small or very tight budget, you will need a highly scalable notation. This gives you the opportunity to build either very high-level abstract representations or, if it is more expedient, to build very low-level detailed representations of isolated or highly cohesive sub-processes. Either way, a considerable amount of progress can be accomplished with comparatively little time and effort. Similarly, if you need short-term confirmation of beneficial results, then both scalability and learnability are crucial. Under such budget or schedule constraints, you cannot afford a notation that requires either extensive training or labor- and time-intensive application.

History is a key consideration, but so is your view on history. It may be that your organization has built a small resource pool of people familiar with the use of a particular notation. If the notation is applicable to your process domain, then it is sensible to give close consideration to that notation for representing your processes. However, it also may be time to depart from history and introduce a new notation. In short, the applicability, learnability, and maintainability of a notation can rapidly become more important to you than history of usage within your organization.

A notation's relative support of your current and future goals can involve several, if not all, of the key characteristics already discussed. If your goal is to develop process guidebooks or self-teaching tutorials, for example, then readability and learnability are primary considerations. If your goal is to develop process models and study them for temporal or behavioral characteristics, then you need a notation capable of capturing process dynamics. This implies you need a notation with a relatively high degree of formality. If one of your long range goals is integrated process automation, then formality may become of central importance.

As a general rule, the more flexible, robust, maintainable, and learnable a process representation approach can be, the more usable it is. As discussed in this section, certain environmental, budget, resource, historical, or goal-oriented constraints become more important, others become less important. Nevertheless, the Consortium believes that there are advantages to having an approach to process definition and modeling that manifests all of these characteristics to a high degree. As discussed in Section 2.6, this is the rationale behind the template-based process representation approach proposed in this guidebook.

2.6 BENEFITS TO THE TEMPLATE-BASED PROCESS REPRESENTATION

This guidebook presents a set of templates and corresponding graphical conventions for use in process definition and modeling. These templates can be used in both pre- and post-support of other process representations. More importantly, the templates can also be used as a stand-alone technique for process definition and modeling. The following material presents additional benefits to template-based process representation by examining the templates from the perspective of key characteristics.

2.6.1 SCALABILITY

The templates have a very high degree of scalability. Each template contains a set of fields for representing inclusion relations. Therefore, arbitrarily long chains of ancestries can be defined. A template can be established as a parent template and other templates can serve as its "children." Any or all of those child templates may, in turn, have further (grand) children which likewise may have still other children. Consequently, you can use the templates to capture the most exacting and detailed process information, the most general or abstract process information, any level or "view" between those extremes, and any combination of these views.

2.6.2 APPLICABILITY

You will also find the templates to be highly applicable to your domain because the templates have been explicitly designed for representing organizational (as opposed to computer-executed) processes. The different template types (managerial and production events, *roles*, resources, products, research, *internal constraints* and *external constraints*) immediately provide a rich set of fundamental conventions for capturing many of the important characteristics of your process.

2.6.3 FLEXIBILITY

As described in Section 3, you will find the templates to be an extremely flexible and highly tailorable tool for process representation. The structural, hierarchical architecture that underlies the templates readily accommodates the addition of new levels to the hierarchy, new types (or "meta-classes") at the highest level, or new sublevels under any existing or new level. Additionally, field and content changes can be made to the templates at the lowest level or at higher levels where the changes can then be "inherited" by lower levels. Though the templates are completely useable as is, they can also be easily adapted to your specific needs.

2.6.4 READABILITY

The templates preserve the detailed readability of pure text-based representations because the majority of template fields allow unconstrained descriptive text. Additionally, the templates also support organizational or global readability because of template interrelationships and especially because of the accompanying graphical conventions diagraming template-based process architectures and models.

2.6.5 MAINTAINABILITY

The maintainability of template-based process representations is quite high because template usage is conducive to the construction of relatively modularized process elements. Each template allows you to define an element both in terms of its internal characteristics and in terms of how that element relates to the other elements in the representation. As a result, "information hiding" (the encapsulation of implementation details so that they are invisible to the outside world) is directly achievable through template usage. You can change a template without having to perform compensating changes in numerous other templates. This yields architectures that are highly tolerant of maintenance, updates, enhancements, and similar modifications.

2.6.6 LEARNABILITY

Learnability is one of the chief benefits to the template-based approach described in this guidebook. The associated tools and techniques are comparatively quite simple, yet they can be used to capture and depict even highly complex processes. The use of these templates does not require any special background or education, and trainees can be easily constructing process representations by the end of their first day of training. There are relatively few rules that need to be followed, yet when adhered to, the result is well-defined process representations.

2.6.7 ROBUSTNESS

The templates achieve a high degree of robustness by not attempting to focus on just one specific domain, but instead by focusing on the phenomenon of process itself. The only assumption behind these templates is that the representation being constructed is intended to design and/or model a process. The type of process is not particularly important. Whatever that process is, if you describe the flow and relationship of events, the passage and evolution of products and research, the people and other resources needed by that process, and the constraints the process is subject to, you will capture enough information to understand and improve that process. All of this can be achieved by using the templates in their current form. Though tailoring the templates is certainly an option, the templates are robust enough that you may prefer—especially initially—to use them just as they are.

2.6.8 FORMALITY

Finally, the templates can support differing degrees of formality. This preserves the ability to choose the degree of formality appropriate to your needs. Since increased formality typically translates to increased cost, you may decide to use the templates to build low or moderately formal representations. These representations would be characterized by the relatively high use of free-form text-based descriptions when providing or filling in field values. Increased formality can be achieved by, for example, instituting a policy that all text-based fields must use some form of structured English (similar to that used when writing pseudo-code for a software program). You can achieve still higher degrees of formality by further constraining text-based fields to structured English and first-order predicate calculus. If you layer onto this the explicit use of states and state transitions (presented in Section 3), you can achieve a sufficiently precise and nonambiguous degree of formality. From here, the template-based representation can be entered into and mechanically executed within an integrated, automated process support environment.

2.6.9 SUMMARY

To summarize, the benefits of template-based process representation include their versatility to be used in both top-down (functional decomposition) or bottom-up (object oriented) methodologies for process analysis, design, development, and verification. Additionally, the template field values can be incrementally established which allows template usage to employ cyclic or *spiral* development techniques. There are explicit fields on the templates for referencing, or binding to, other templates so that both the characteristics of a given template and the relationships it has with other templates are quite visible and accessible. This facilitates maintaining and updating not only the information contained within the templates, but also the overall process architecture represented by the templates.

Template usage is based on a few simple conventions and requires no special background or experience; therefore, training requirements will be nominal. Furthermore, they can be used as a foundation for building representations using extensions of other notations (such as SADT or Petri nets). The templates can also be used to augment any existing representations based on other notations.

Another benefit of the template-based approach is that using these templates is amenable to automation and automated tool support. Initially, you may decide to provide automated support via some commercial tool designed for electronic forms management. More versatile automation can be achieved by forms management that includes hyper-text, or "hot" links between instantiations of the various templates. Coupled with a "windowing" computer environment, the construction, use, and application of the templates or the resulting process representations can be readily and directly supported through automation. At this point, it is a relatively small step to achieve executable process representations that support both project-level and process-level management.

Finally, you should remember that the templates were designed to be tailorable to site-specific or organization-specific needs. If you work under special circumstances or have uncommon requirements or needs, you can readily alter the templates so that they provide you with improved support exactly where you need it. Should your needs change, the templates can be altered to conform to the new priorities. The templates not only support you in efficiently changing your process, they also support efficiently changing the way you change your process.

2.7 PROCESS REPRESENTATION TERMINOLOGY

The study of process representation critically depends on clear communication. Process-specific communication depends upon the study of terms, definitions, and word meanings commonly used

within this field. Process representation is a relatively new field, and it is common for there to be inconsistency in the use and meaning of terminology. The following definitions are not meant to imply that this is industry standard usage. Nevertheless, the usage presented here is as close to industry consensus as can be determined at this time.

2.7.1 PROCESS REPRESENTATION

Process representation is a general term referring to the combined or sequential efforts of jointly performing process definition and process modeling. Typically, process definition is the more abstract side of process representation, and process modeling is the more detailed side.

2.7.2 PROCESS DEFINITIONS

Process definition is the act of representing the important characteristics of a process in a way that facilitates understanding and communication. Process definitions can be entirely text-based and un-structured, or they can be of varying degrees of increasing structure and formality. As a representation becomes progressively more structured and formal, it transitions from a process definition to a process model.

2.7.3 PROCESS MODELS

Process modeling both extends and constrains process definition by requiring that the process model adheres to a predefined set of methods and structural conventions; the latter are often rendered graphically. Models can by syntactically correct: all the rules and conventions of using the modeling notation have been complied with. Models can also be symantically correct indicating that the model is an accurate representation of a real-world process.

2.7.4 PROCESS EVENTS

Events are processes, activities, or *tasks*. There are few objective criteria for distinguishing between a process, an activity, and a task. Most attempts to discretely distinguish various levels are subjective. Events are organized into collections; the events in a collection typically form a tree or a network of trees. The root of a tree is referred to as a process; interior nodes are activities; and the leaf nodes are considered tasks.

It is useful to allow a "view of interest" consisting of subtrees of the total process. Each subtree has its own unique root process, set of activities, and tasks. By definition, a process is a single node that has no parent and contains 0 to n activities and tasks. Each activity may also contain 1 to m activities and tasks (an activity must contain at least one task, otherwise it is, by definition, a task). Each task is atomic and, as such, cannot be further decomposed within the current degree of detail of a given view. In Figure 2-6, E1 is the process, E2, E3, E4, E6, E10, E14, and E16 are activities, and everything else (E7, E8, E9, E11, E12, E13, E17, E18, E15, E19, and E20) is a task.

The shaded region in Figure 2-7 defines a view of interest. Like the total process, it too contains a root process and a set of activities and tasks. This view's "process" is E10, which is comprised of only one activity (E16) and three tasks (E15, E19, and E20).

As a final example, consider the same overall structure with a different view of interest (Figure 2-8). If this is the view of interest under examination or discussion, E2 is the process and E6 is the only activity.
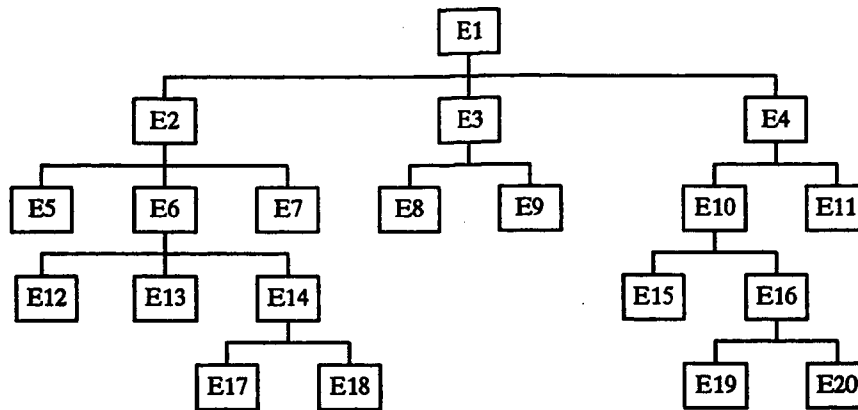
Figure 2-6. Event Structure
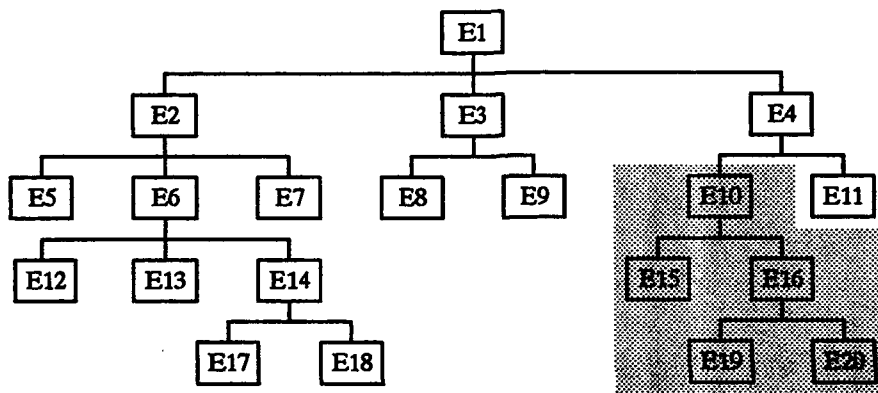
Figure 2-7. Event Structure View 1

The tasks are E7 and E14. Note in this example that the view of interest does not necessarily have to include the original tasks of some larger perspective. Activities in the overall structure can become leafs (tasks) in a subview (as with E14 in Figure 2-8). You can select a view that is in the "middle" of a larger perspective yet not look as high or as low as the original perspective. Also, note that when selecting what had been an activity, you can optionally include none of the original subactivities or tasks (as with E14 in Figure 2-8), all of them (as with E16 in Figure 2-7), or a partial selection (as with E6 in Figure 2-8).

To summarize, with these distinctions, it is necessary to consider the view of interest before defining "what is a process" and "what is a task." Put in more formal terms (using a simplified process tree in which all branches are of the same length), if "m" signifies the uppermost level of a process model, "n" signifies the lowermost level, "a" signifies some number of levels below the top, and "b" signifies some number of levels up from the bottom. Then when looking at an m+a, n−b view of the process hierarchy (defining the highest and lowest levels of interest, respectively), the m+a level activity (of which there is typically only one) becomes the "process" and all n−b activities become tasks (i.e., the lowest level "atomic" events in our particular view of interest). The remaining nodes between m+a and n−b are the activities.

2.7.4.1 Process

As discussed under "events," process is a relative term. For the purposes of this guidebook, a process is considered to be an event that is superordinate to all events of which it is comprised. Processes can
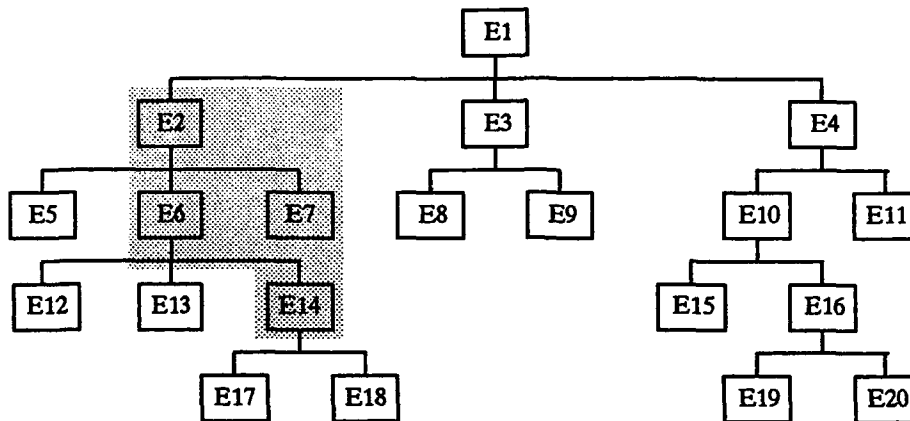
Figure 2-8. Event Structure View 2

be comprised of one or more activities and tasks. In some terminologies, a process is considered to be something that is perpetually ongoing; e.g., a quality assurance process.

2.7.4.2 Activity

An activity is an event composed of one or more subactivities or tasks. It has at least one parent event and implies at least one child event. Activities are typically used to initially partition or group work without regard to the resources needed, sequencing, or other enactment details. In some terminologies, an activity is considered something of finite duration that explicitly starts and stops, e.g., the series of inspection activities associated with inspecting a particular artifact. Activities are modeled using all four types of relations. As discussed in Section 2.7.20, these relations are:

- Inclusion

- Sequence

- Specialization

- Reference

2.7.4.3 Task

A task is an event: one of the basic components in a process description. A task is atomic; it has no child events and has at least one parent event. A task is typically enacted by a human, which requires process planning and control; and it is typically composed of an arbitrarily complex ordering of steps. Tasks can participate only as a subordinate in inclusion relations, but may participate unrestricted in sequence, specialization, and reference relations. As with activities, the term "task" is typically used to define something of finite duration.

2.7.5 PROCESS THROUGHPUTS

A *throughput* is either a tangible or intangible artifact. Throughputs usually refer to intermediate and final (sub)products of the software development process. They can be a physical artifact, usually modeled as a product (such as a module, a document, or a schedule), or an intangible artifact, such as the knowledge gained from having performed research.

2.7.5.1 Products

Product artifacts are tangible throughputs within a process. This type of throughput typically represents the vast majority of artifacts that pass through a process. Examples include code modules, end-user guidebooks, circuit boards, and anything else tangibly produced by a process. Products can be decomposed into subproducts, sub-subproducts, etc. This decomposition is captured within a model by the inclusion relation.

2.7.5.2 Research

Research is a by-product of a process; but it differs from products in that research is considered intangible. If for instance, the research leads to a technical paper, that technical paper is modeled as a product. However, if experiments or investigations are being performed within one or more events, but nothing tangible is available, the throughput can still be explicitly modeled as a research (intangible) artifact. As with products, research can be decomposed into subresearch, sub-subresearch, etc. This decomposition is captured within a model by the inclusion relation.

2.7.6 PROCESS SUPPORTS

A *process support* is any nonthroughput item needed by an event for it to be performed. Events need throughputs, as that typically is the purpose of events: to accept one or more throughputs; modify, manipulate, inspect, and possibly create one or more new throughputs; and pass those along to other events. However, more is needed by an event than just the throughputs. These nonthroughput items are all modeled as supports. Two common types of support include roles and resources.

2.7.6.1 Roles

Roles commonly represent either individual humans or humans working in concert toward a common goal or set of goals. Consequently, "programmer," "manager," "clerk," etc., all define roles that can be assumed by individuals. However, "programming team," "inspection department," and "quality assurance division" also define roles. In the latter case, the roles are organizational roles as opposed to individual. For process definition, roles can be defined at all levels of abstractions.

2.7.6.2 Resources

Resources are nonhuman items needed to support an event. Examples include equipment, office space, supplies, and funding. All items that might be required to support an event can be modeled as resources. Resources can be decomposed (using the inclusion relation) so that while one level of event abstraction shows that the training building is required, at a lower or more detailed level of abstraction the support might show that only a small classroom is actually required.

2.7.7 PROCESS CONSTRAINTS

Process constraints describe the limiting conditions associated with the activation, performance, or cessation of an event. Whereas supports can be viewed as those things required to enable or make the right things happen, constraints can be viewed as those things required to disable or prevent the wrong things from happening. In this guidebook, constraints have been divided into two general types: internal constraints and external constraints.

2.7.7.1 Internal Constraints

Internal process constraints are typically managerial in nature and usually take the form of authority and permission. Examples of internal constraints include management authority or permission required before an event can commence. Internal constraints also convey authority to roles to suspend events, cancel events, recommence events, or cease events. In all cases, internal constraints are always coupled with a role (typically a role signifying lead or managerial responsibility, but in all cases a role signifying—by definition—some form of authority). As a rule, internal constraints are those constraints that you have authority to change, countermand, or enforce.

2.7.7.2 External Constraints

External constraints include all factors that may limit or constrain how an event proceeds and that are not directly attributable to local authority (which are modeled as internal constraints). Examples of external influences that may constrain an event include quality requirements, corporate standards, division policies, engineering procedures, process guidelines, and management directives. External constraints differ from internal constraints in that they are usually not subject to discretionary use—they are intended to be, and expected to be, explicitly followed regardless of project-specific issues.

2.7.8 PROCESS ANALYSIS

Process analysis is a general term that refers to several different types of analysis. One form of process analysis is the analytical effort performed so as to construct or formulate a definition of a process. This "upstream" form of analysis usually involves study of corporate guidelines and procedures and interviews with management and engineering staff. Another form of process analysis involves examining an existing representation of a process. This "downstream" form of analysis is employed as a key step in performing process improvement. In addition to the analysis of the static representation of a process, analysis can also be performed on the dynamic characteristics of a process. Commonly, this involves the collection and interpretation of process-related metrics. In rare cases, the representation itself may yield a model that can be executed to simulate process dynamics, and analysis can be performed against this dynamic model.

2.7.9 PROCESS DESIGN

Process design is the act of translating process requirements and objectives into a cohesive system. Like process analysis, the design effort may be directed toward an initial representation of a process or toward altering an existing process representation so as to design a new process. High-level designs can be built entirely of references to events, throughputs, supports, and constraints; detail designs may explicitly reference processes, activities, tasks, products, research, roles, resources, and internal and external constraints.

2.7.10 PROCESS INSTANTIATION

When process design is completed, the next stage usually is to instantiate a project-level process plan from the process representation. This involves replacing variables with actual values. Hence, if a design calls for two people in the role of programmers, instantiation involves filling those roles with the names of actual people. Likewise, resource requirements are mapped to actual resources, and general product types are replaced by actual product names. The result of process instantiation is a project process model.

2.7.11 PROJECT PROCESS MODEL

A project process model is an instantiation of a process representation that explicitly states the names of actual people, products, and resources intended for use within a specific project. The project process model is used as a management aid, and ideally includes such information as milestone dates and cost data. Once the process is enacted, actual schedule and cost data can be compared against the planned version.

2.7.12 PROCESS ENACTMENT

Process enactment is the execution of a project process model. Enactment indicates that actual work is being performed or, at a minimum, that work has at least commenced and is expected to continue. Currently, most environments enact a process through traditional or manual management methods. In highly integrated, automated environments, there is excellent potential for process management through process automation.

2.7.13 PROCESS AUTOMATION

Process automation involves environment-based support of process management. The degree to which an environment can participate in and facilitate process management can vary widely. At one end of the spectrum, an instantiated process model can be regularly polled by an environment; and electronic mail sent to notify participants that certain activities have completed, other activities are cleared to commence, and certain resources are now available. At the other end of the spectrum, these environments can automatically collect process-related metrics, detect potential or actual process bottlenecks or deadlocks, provide management with summarized reports of project status, and potentially can even provide dynamically sensitive recommendations on alternative process options.

2.7.14 PROCESS MATURITY

Process maturity is a phrase used to convey levels of process quality. Generally, higher levels of process maturity indicate improved productivity and reduced risk. Currently, process maturity is considered to map into five general levels. From lowest to highest, the process maturity levels are ad hoc, repeatable, defined, managed, and optimizing. This concept was examined in greater detail in Section 2.2.

2.7.15 PROCESS IMPROVEMENT

Process improvement takes many forms, but it is commonly considered to be the evolution of a process from lower levels to higher levels of process maturity. Pragmatically speaking, process improvement can also involve specific efforts to reduce costs, reduce staff turnover, mitigate risk, improve product quality, increase corporate process flexibility, improve moral, reduce time-to-market, increase customer satisfaction, reduce process redundancy, reduce backlogged orders, accelerate accounts receivable collections, and essentially all other areas of improving organizational competitiveness.

2.7.16 PROCESS ASSETS

Process assets are self-contained process "pieces" or modules that can be used in combination with other process assets to rapidly develop tailored process models. Process assets may involve predefined

2. Overview and Fundamental Concepts

collections of roles into teams, predefined collections of subproducts into products, and predefined collections of tasks and activities that can be used in concert to guide work in combination with a general process model. Process assets that consist entirely of events and event relations are called *process architecture* assets. Process assets that do not involve any events are process elements (roles, products, and resources). Process models can be constructed of process assets by selecting a process architecture and then incorporating the necessary process elements into that architecture.

2.7.17 PROCESS ASSET LIBRARY

Process asset libraries are common repositories for the storage and retrieval of process assets. As with all librarian-oriented tools, a process asset library needs effective query or search mechanisms, import and export facilities, and version management.

2.7.18 ASSET GRANULARITY

Asset granularity is a phrase used to refer to the relative degrees of abstraction or detail conveyed within a process asset. "Large" or "high" granularity refers to assets that have comparatively little detail and that use abstraction extensively. Conversely, "small" or "low" granularity conveys assets that have comparatively more details and use relatively less abstraction.

2.7.19 ASSET INTERFACE

Assets can be described, stored, and matched for compatibility as a function of their interface with other assets. The most general view of an asset's interface is the set of all references originating from that asset. These references include sequence, inclusion, specialization and reference relations. Specialized examination of an asset's interface can be constrained to just the sequence relations (applicable only to assets containing events) or to just the throughput references.

2.7.20 PROCESS RELATIONS

Processes can be modeled as sets of objects that are bound to each other by a variety of pre-defined relations. This guidebook uses four types of relations to bind process objects. These are sequence relations, inclusion relations, specialization relations, and reference relations.

2.7.20.1 Sequence Relations

Sequence relations are pre- and postset relations that exist between and among events. The preset of events to a given event are all events which immediately precede the event of interest. Similarly, the postset is the set of all events which have the event of interest in their preset. From a minimalist perspective, all that is required to construct a process model is a set of events and an ordering or partial ordering of that set. Sequence relations are used to define the ordering of events.

2.7.20.2 Inclusion Relations

Inclusion relations are the most general form of "parent/child" relations. All events, throughputs, supports, and constraints can be (optionally) decomposed by using inclusion relations. Products can be shown as including subproducts, and teams can be shown as including individual roles.

2-24

2. Overview and Fundamental Concepts

2.7.20.3 Specialization Relations

Specialization relations are a means for indicating inheritance. The child object in a specialization relation is a specialized form of the adult. For example, a role object described as "Programmer" may have specialized forms called "C Programmer," "Ada Programmer," and "Fortran Programmer."

2.7.20.4 Reference Relations

Reference relations are all relations within process modeling that are not sequence, inclusion, or specialization relations. A reference relation conveys that one object or asset either references or is referenced by another object or asset. Reference relations include an event referencing an external constraint it is subject to, a role referencing an internal constraint within its authority, or a product referencing an event it evolves through.

2.7.21 ASSET COUPLING AND COHESION

The use and meaning of coupling and cohesion within and among process assets is identical to its use in the field of software engineering. High degrees of coupling between assets indicates that there are a comparatively large number of sequence or reference relations that exist between those assets. Low coupling indicates a relatively minimal number of interrelations. High asset cohesion indicates that the asset employs strong "information hiding" techniques and that all aspects of its internal work is closely related. As a rule, low coupling and high cohesion are considered desirable qualities.

2.7.22 PROCESS ARCHITECTURE

A process architecture is a representation of a process that limits its contents to only events, sequence relations, event inclusion relations, and event specialization relations. Abstracted out of the architectural view are such issues as throughputs, supports, and other elements which, when combined with an architecture, can yield a detailed process model.

2.7.23 PROCESS NOTATION

A process notation is any predefined set of symbols which, when combined with a methodology, can be used to render a depiction or representation of a process. In the most abstract sense, text-based process definitions use the character set and grammar rules as their notation and methodology, respectively. More commonly, however, process notations should include some degree of graphical ability. SADT, Petri nets, Statecharts, ETVX, and STDs are each distinctly different notations that can be used for representing processes.

2.7.24 DEGREES OF FORMALITY

Relative formality is of critical importance for automated or executable process models. At a minimum, process models should provide a consistent view of the process being modeled. A process model needs to communicate nonambiguous information. With increasing formality, there is reduced ambiguity. At its most formal, a process notation is coupled with a supporting calculus, algebra, or both.

This page intentionally left blank.

3. PROCESS DEFINITION TEMPLATES

The preceding material has presented an introduction and overview of key concepts of process definition and modeling. The purpose of this section is to describe and detail a set of tools and practical means for applying process definition and modeling. Section 3.1 explains the reason for choosing this specific set of templates to support process representation. Section 3.2 presents details on the process templates and explains how these templates (or some subset) can be used to initiate process definition. It should be noted that this is simply a recommended set of templates with recommended fields for each template. They are designed to be applicable across the greatest variety of process representations possible. As discussed later in Section 4.5.2, the approach taken readily allows tailoring these generalized process data collection templates to the specific characteristics and needs of a particular corporate or project environment or process representation. Section 3.3 presents a graphical notation that supports diagrammatic depictions of template-based process models. The graphical notation is also based on the reasoning presented in Section 3.1; it therefore directly corresponds to and supports the templates presented in Section 3.2.

As noted in Figure 3-1, after reading Section 3 you may skip to the summary chapter. However, for more details on using the templates, optimization, and additional material on process modeling and definition, the Consortium recommends that you continue with Sections 4 and 5.
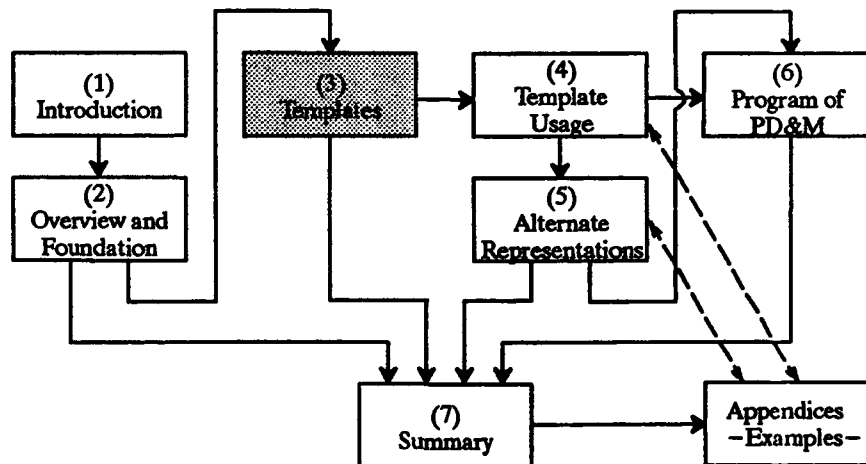
Figure 3-1. Guidebook Organization View 3

3.1 DERIVING PROCESS REPRESENTATION TEMPLATES

Models are abstractions. With any abstraction, the objective is to emphasize important characteristics and reduce or eliminate nonessential information. However, for an abstraction or model to be widely applicable, it must capably represent all important characteristics in its target domain.

Your interest in process extends beyond just the events themselves. You are also interested in the types of products produced by the process; the resources required; and the relationships between products, resources, and events. To determine what the model needs to capture, you need to methodically analyze the concept of a process. Shown below are a series of figures that construct the model used as the foundation for developing the templates.

As shown in Figure 3-2, at the simplest or most abstract level, one view of a process is that it is anything that uses inputs and produces outputs.
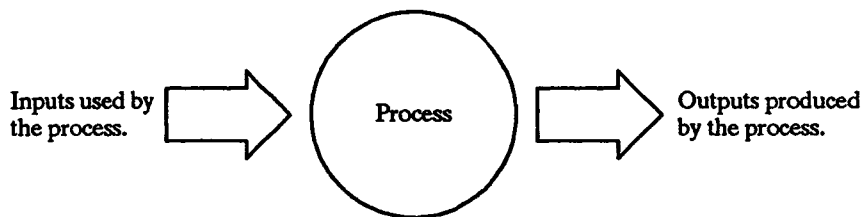
Figure 3-2. Process Model Construction—Level 1

However, it can be seen that not everything required by a process is consumed in producing the outputs. Although some inputs are converted, consumed, or otherwise transformed as the process constructs its outputs, other inputs act only as catalysts. To distinguish these two classes of inputs, variable inputs are those used, consumed, or included in the outputs; and their rate of use tends to vary (in the most general sense) in proportion to the volume of outputs being produced. Catalytic inputs—those not "consumed" by the process—are referred to as fixed inputs (Figure 3-3) because they are less likely to vary in direct relation to the volume of outputs. Examples of variable inputs include raw materials, supplies, and subcomponents. Examples of fixed inputs include machines, meeting rooms, management staff, engineering staff, and administrative staff.
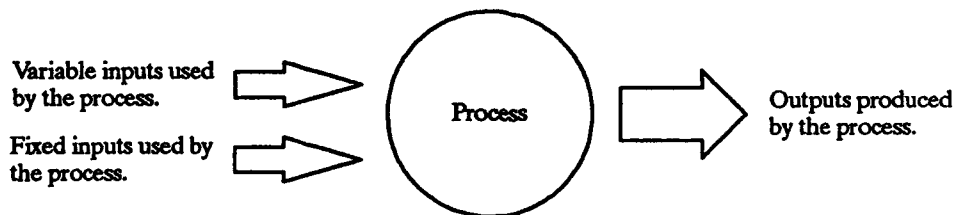
Figure 3-3. Process Model Construction—Level 2

Since fixed inputs generally tend to remain available to the process, it is diagrammatically advantageous to move these inputs to the bottom of the process "bubble." The revised diagram, as depicted in Figure 3-4, implies that variable inputs are directly related to the outputs, but both inputs and outputs depend upon the process "catalysts" or fixed inputs.

At this point, the model represents a process as something that converts inputs to outputs by relying on a set of resources. Clearly, all this does not happen accidently. The model must also show, as indicated in Figure 3-5, that a deliberate process is a set of events subject to control. Your definition of a process is now: a **controlled** set of **events** that uses **inputs** to produce **outputs** by relying on a set of **resources**. (Readers familiar with SADT or integrated computer-aided manufacturing definition language (IDEF0) will note that this perspective is analogous to the approach used in those representations.)

Finally, a process does not always have the advantage of working with tangible phenomena or objects. For example, although a meeting room is tangible, the responsibility of people using that room is not.

Variable inputs used
by the process.

Process

Outputs produced
by the process.

Fixed inputs used to
support the process.
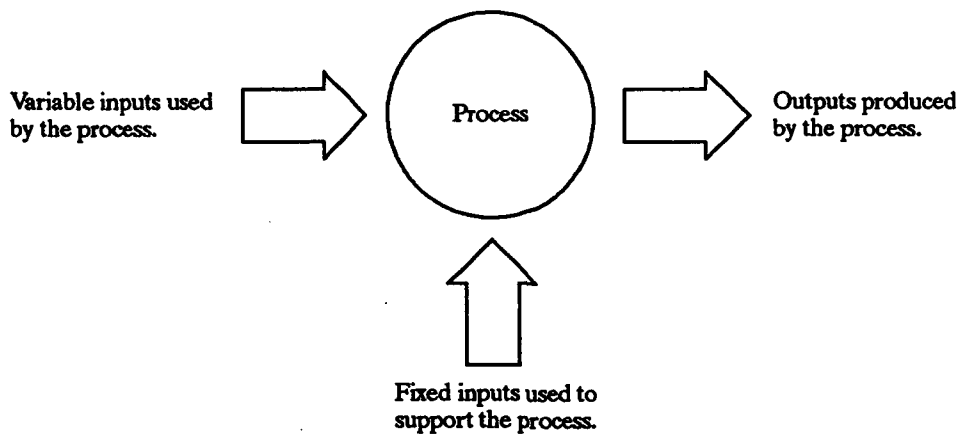
Figure 3-4. Process Model Construction—Level 3

Change control

Variable inputs used
by the process.

Set of
Events

Outputs produced
by the process.

Fixed inputs used to
support the process.
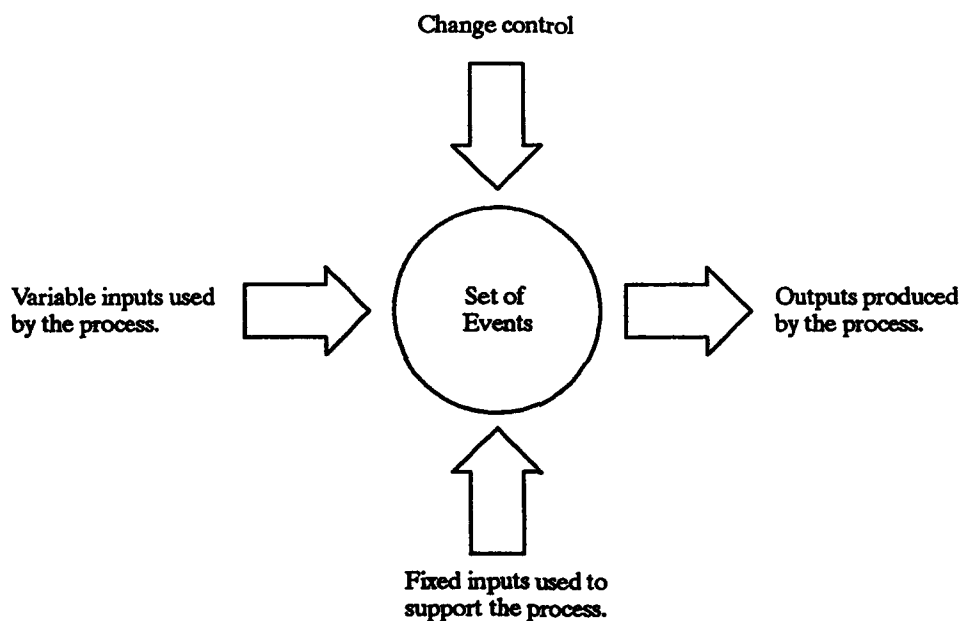
Figure 3-5. Process Model Construction—Level 4

Although a software requirements document is tangible, the research that went into developing the requirements is not. As shown in Figure 3-6, inputs, outputs, supports, and controls can all be divided into tangible and intangible classes.

The abstract model shown above, and the detailed model (which distinguishes tangible from intangible) both directly correspond to the templates discussed in Section 3.2. As shown in Figure 3-7, the constraint templates are used to capture information regarding change control. Throughput templates are used to capture information about both the inputs and the outputs of a process. Support templates are used to capture information about people and objects needed to support a process. Event templates capture information about the managerial and production activities and tasks occurring within a process.

As explained in Section 3.2, constraint, throughput, support, and event templates are all "meta-class" templates. Each has subordinate templates at the class level to represent tangible and intangible instances. As shown in Figure 3-8, external constraints are considered intangible in the sense that
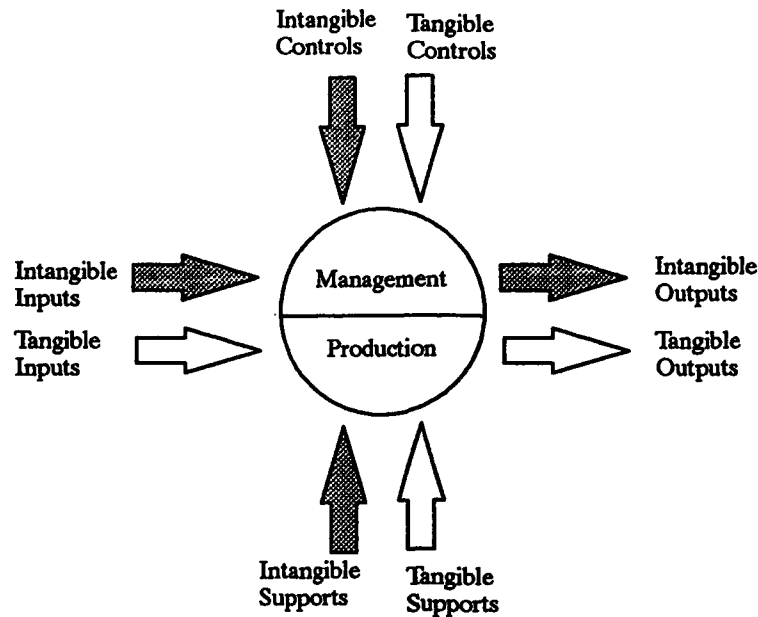
Figure 3-6. Process Model Construction—Level 5
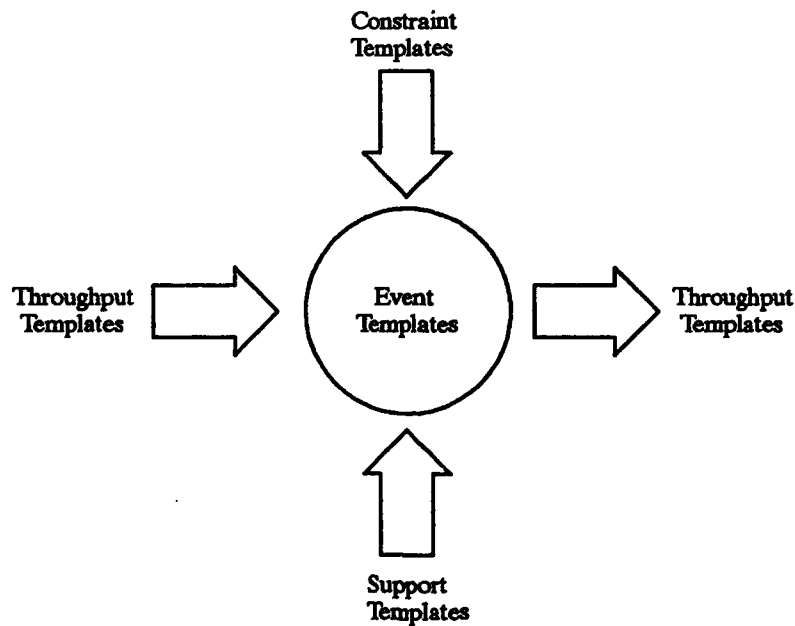
Figure 3-7. Meta-Class Templates

external constraints are constraints that you cannot influence. This contrasts with internal constraints which represent permission or authority and, therefore, are subject to "local" or internal decision making. Product templates represent tangible throughputs (such as software modules or design documents), and research templates represent intangible throughputs. Resource templates represent tangible objects that support the process (such as machines or facilities), and role templates represent responsibilities (and, in combination with internal constraint templates, authorities) conveyed by roles to people performing the process. Management templates and production templates represent intangible and tangible types of events, respectively.

External
Constraint
Templates

Internal
Constraint
Templates

Research
Templates

Management
Template

Research
Templates

Product
Templates

Product
Template

Product
Templates

Role
Templates

Resource
Templates
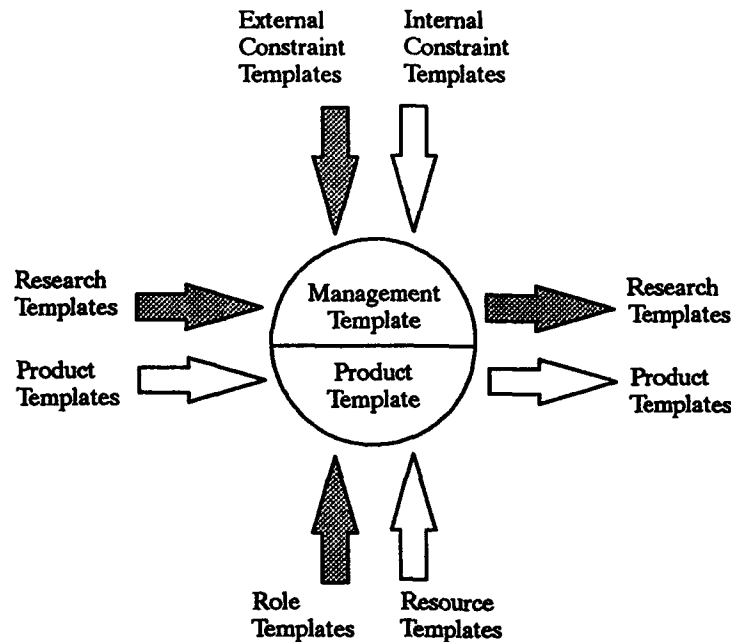
Figure 3-8. Class Templates

It is important to note that this set of templates will allow you to construct process models that are subject to a variety of control paradigms. For example, if you intend for your process model to depict an object-oriented type of process, then the product templates (and states of products) will play a key role in influencing the flow of events. Entry criteria, internal processing, and exit criteria will all make extensive use of references to products and changes in the states of products. From this orientation, it is not the flow of events that define the states of products, it is instead the states of products which define the flow of events.

If you intend for your process model to depict a process driven by functional decomposition, then the event templates (and states of events) will be the primary factors driving the process flow. In this case entry criteria, internal processing, and exit criteria all make extensive use of references to events and changes in the states of events. These models also make extensive use of inclusion relations (within and between levels of events) to capture the hierarchy of events. Using this approach, the events will be the greatest factor influencing not only the process but the products produced and the resources involved.

As a final example, you may want to model a process that is essentially governed by management. In this case, it is not the state of products nor the occurrence of events that triggers or controls what occurs next. The evolution of the process depends almost entirely on the decisions and actions of managers. A model of this type of process makes extensive use of constraints, especially internal (or permission) constraints. Entry criteria, internal processing, and exit criteria are largely stated in terms of numerous internal constraints. In effect, events start when management states that they can start, and they end when management declares them to be over, and so on.

The templates are not designed to be biased toward any particular process execution paradigm. Object-oriented processes, functional processes, management-intensive processes, etc., can each be depicted using this same set of templates. The primary difference is the detail allocated to the various classes of templates and the types of references used in the entry criteria, internal processing, and exit criteria.

To summarize, the underlying model used for constructing the templates is quite simple. A process is a set of events that typically has inputs, produces outputs, requires enabling supports that allow the right things to happen, and is subject to disabling constraints that prevent the wrong things from happening. Events, throughputs, supports, and constraints; virtually all processes can be defined and modeled using these constructs. As described in Section 3.2, the templates provide a simple means for capturing, manipulating, organizing, and analyzing such information.

3.2 TEMPLATE DEFINITION AND LAYOUT

This section describes the templates and provides examples of paper-based representations. Using paper to capture template information requires minimal start-up costs. However, any paper-only approach to process representation or documentation rapidly becomes quite difficult to maintain on real-world projects. From the perspective of scale-up, it is ideal to use electronic forms management, hyper-text, or some other automated support to facilitate capturing process information and maintaining the integrity of interrelations between that information. The paper-based approach works well for training, examples, and possibly even small pilot projects and case studies. However, large projects will benefit from automation. When you consider automated support for a template-based approach, you should reconsider template (or table) design based on the strengths of the particular automated tool you intend to apply and tailor as appropriate. The design of the templates shown on the following pages is not nearly as important as the information they contain, and the types of interrelationships they capture.

As seen in Figure 3-9, the set of templates is derived from the following structure.

- One "Root" Template

- Four "Meta-Class" Templates

- Eight "Class" Templates
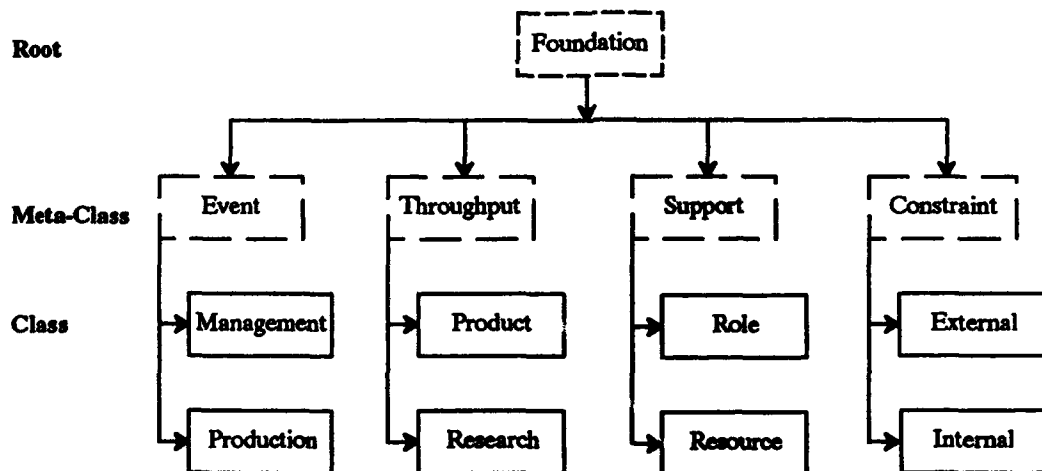
Figure 3-9. Template Structure (Generic)

For the purposes of this guidebook, class templates are derived from meta-class templates which are derived from a common root template. The root and meta-class templates are not actually used during process definition; only class templates are used (in the diagram, class templates are the only

templates shown with a solid boarder). However, the root and meta-class templates are useful for discussing fields common to groups of templates and for simplifying tailoring the templates for site-specific objectives. This structure allows easier construction and discussion of the templates and their fields. Furthermore, this approach facilitates the development and use of an automated tool for designing, maintaining, and using electronic versions of the templates.

All templates have the following fields in common: **Name, Unique Identifier, Purpose, Comments, and Revision History.** Any field intended to be used on all class templates, regardless of type, is shown on the foundation (root) template. The discussion of such globally common fields is found in the sub-section that describes the foundation template. All meta-class templates "inherit" these common fields from the root template. Each meta-class template adds additional information common to class templates of that meta-class and not common to class templates of a different meta-class. For example, under the event meta-class, there are class templates for management and production events. These two classes have certain fields in common with each other. For example, they have fields for entry criteria and exit criteria, and each class provides space for a logical description of these criteria. Using the principle of inheritance, any field shown at the meta-class level of a template will also appear on the class templates subordinate to that meta-class. (Also shown will be all fields that the meta-class template inherited from the root template.)

As mentioned, these are generalized templates and it is expected—indeed, encouraged—that they be tailored to the needs and characteristics of the corporate or organizational environment in which they will be employed. Similarly, these templates can also be tailored toward the actual process notation favored by a given environment. The purpose of the hierarchical architecture in the template design is to explicitly facilitate the tailoring process.

When considering alternative template designs, consider the scope of any new field being added. Will the new field be used on all eight of the class templates? If so, put the new field on the foundation template (all other templates will then inherit this field). Will a the new field be used on both the role and the resource templates? If so, change the support template (at the meta-class level) and allow the role and resource templates (and any other defined support templates) to also inherit it as a common field. This approach makes it easy to introduce and document changes to the templates, yet preserves the underlying consistency between templates.

Most of the templates described in this section make reference to the concept of state and state descriptions. Understanding the state of something conveys insight into how that phenomenon can change over time. A traffic light might be described as a three state device. That is, it has a state of being red, yellow, or green. Ideally, the set of states for a given item will not overlap (i.e., the light can not be simultaneously red and green), and the set of states for a given item will be complete (e.g., traffic lights cannot turn purple). One of the primary advantages of using states in process modeling is that they allow conditional expressions to be written in terms of states of items. For example, one activity may need to be in a state of complete before another activity can transition from standby to an active state. For most of the following templates, there is a recommended set of default states that can be used to facilitate more precise representation of the relationships that exist between the various events and components participating in an overall process.

In the following material, each template is discussed in turn. The contents of the foundation template are discussed first. Each meta-class is discussed; and within each of these areas, the subordinate class templates are described. Discussion of the graphical depiction of template-based process representations is deferred until Section 3.3.

3.2.1 FOUNDATION TEMPLATE

The Foundation template (Figure 3-10) is the repository for all common fields shared by all template classes. It reduces the input labor by concentrating all shared data on one template.
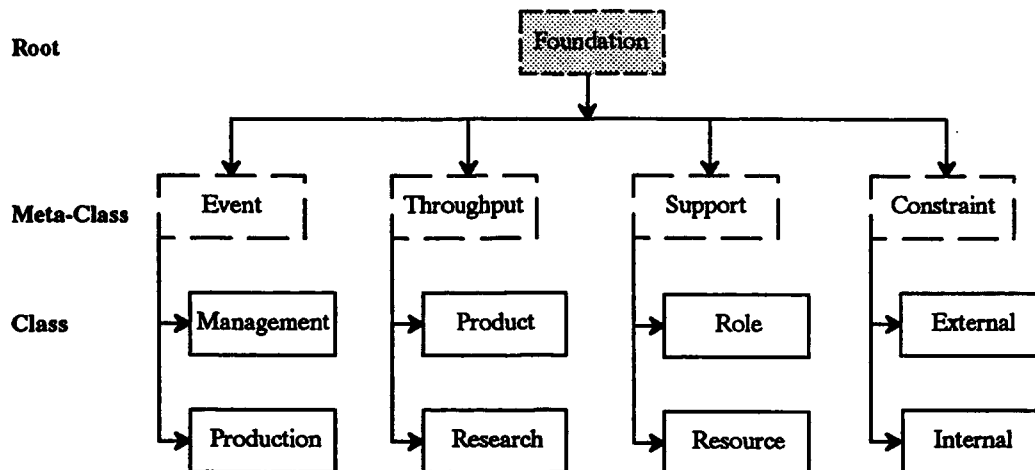
Figure 3-10. Template Structure (Generic)

Ideally, the layout and format of the Foundation template (reflected in the examples shown in this section) also conveys the layout and format of the common fields on the class templates. Therefore, if a **Name** field is shown in the upper-left corner of the Foundation template, all templates will show a **Name** field in the upper-left corner.

The Foundation template example (Figure 3-11) contains the following fields:

- Level #

- Version Number and Date

- Name

- Unique Identifier

- Purpose

- Comments

- Revision History

The **Level #** field is an optional field that can be useful in representing levels of templates. All of the templates can be used to represent a family of items. For example, a role might be constructed so that at the highest level it is represented by a team. This could be the Level 1 template. Underneath the team, there may be other role templates (such as team leader, principle engineers, and support technicians) that detail how the team was comprised. These latter templates could be Level 2 templates. The level number is used as a convenience to help you quickly understand some of the hierarchical relationships between templates.

Level #	Foundation Template Template Type	Version # and Date
Name		**Unique Identifier**
Purpose		
Comments		

Figure 3-11. Foundation Template

The **Version # and Date** field is shown as a single concatenated field. This is consistent with the concept of segmented identifiers described in the preceding paragraph. It is often useful to know what date a particular version was released or, conversely, to search for the version that was in effect on some given date. Nevertheless, although version and date are combined into one field in this example, this template can be tailored to show these as two distinct and separate fields (possibly even adding a third field for the initials of the person who authored that particular version).

The **Name** field is used to assign a descriptive name to the item described by the template. The **Unique Identifier** field is used to assure that each template can be identified uniquely. It should be noted that there is considerable advantage to developing a segmented identifier in which each segment further classifies the item being identified. For example, all product identifiers might begin with the letters IP, representing an input-class template product item. Similarly, all task identifiers might begin with ET, representing the event meta-class and the task class. The goal is to develop unique identifiers that convey meaning in addition to uniqueness. A sequential number can be appended to the prefix to assure uniqueness.

The **Purpose** field is used to provide a sufficient description of the template's intended purpose. This should be kept as short as is possible; however, if a more elaborate description is needed, the **Comments** field may also be used.

The **Comments** field is for virtually any use deemed advantageous by whomever is using the templates. Aside from more detailed descriptions of the item's purpose described by the template, the **Comments** field can also note potential problem areas, possible future enhancements, and questions for things to be researched later.

At the bottom of the Foundation template, the **Revision History** field is a means to track the evolution of information contained within the templates. Especially in the case of large sets of templates, it is typically more practical to simply update and replace individual templates, as required, than it is to repeatedly generate entire new printouts of the full set of templates. When doing individual updates, the **Revision History** field can facilitate a manual confirmation of document sets having the latest versions.

Note that between the header section of the templates (all the common fields except for **Revision History**) and the footer section (**Revision History**), there is a large blank section. This section of the template is reserved for lower level (meta-class and class) templates. Therefore, if the recommended approach is followed, all class-level templates will have header and footer sections that, in format, are identical to each other. Format variations that exist between class templates are contained within the large reserved section in the Foundation template.

3.2.2 EVENT TEMPLATES

The Event template (Figure 3-12) is, from one perspective, the most crucial template of this entire set. In essence, a process can be formally defined using nothing more than the Event template. It is the only required meta-class template. Strictly speaking, all that is necessary for modeling a process is some nonambiguous means for ordering (or partially ordering) the flow of events that comprise that process. All else (the resources required by the process, the products produced by the process, etc.) is optional. The flow of events, however, must be captured; this is the essence of process definition and modeling.
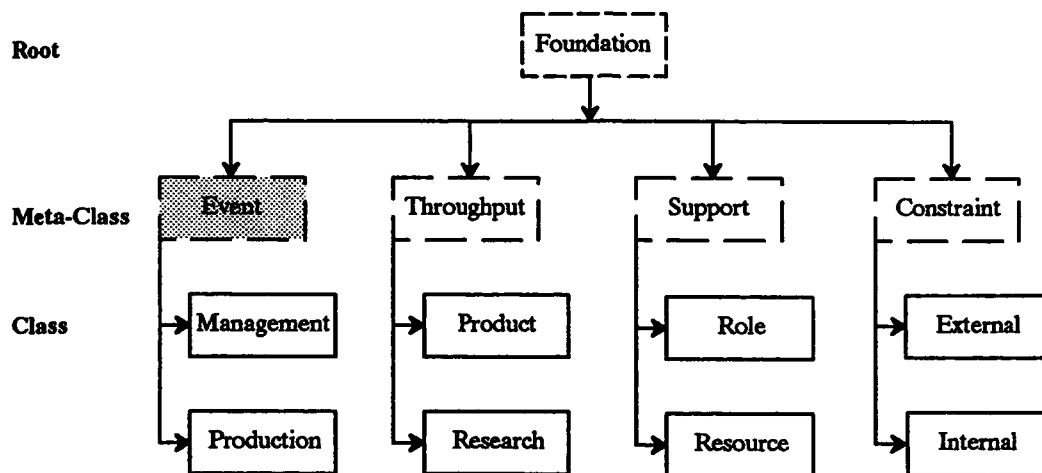
Figure 3-12. Template Structure (Generic)

To facilitate defining the entry and exit criteria for events, in terms of other events, the following state terms are used:

- Pre-enabled

- Enabled

- In-progress

- Disabled

- Suspended

- Cancelled

- Completed

For example, the entry criteria to a given activity might state that activities A, B, and C all be in a state of completed and that activities D and E be in a state of in-progress.

It is recommended that the state of a task (the lowest level of events described within a given model) be described using only the default set of state terms (those listed above). Typically, this allows a task to assume a state of pre-enabled, enabled, in-progress, or completed (the other state terms are used for comparatively unusual situations).

For more complex activities, intermediate transition terms could be used to allow further distinction of current state. Specifically, for activities and processes the state term, in-progress, would be expanded into any number of intermediate state terms. This approach is useful because even mildly complex events may be desirable to discretely distinguish different stages of work within that event. For example, in a given event, the in-progress state might be replaced by in-analysis, in-design, and in-review states.

Both Event templates (Figure 3-13) share the **Entry Criteria** fields, **Internal Processing** fields, and **Exit Criteria** fields. **Entry Criteria** and **Exit Criteria** fields can be readily separated into those that are tangible and those that are intangible.

Tangible entry criteria include:

- Products required (typically these are outputs produced by prior events).

- Roles required (possibly at various levels such as employees, groups, or departments).

- Resources required (computers, test equipment, etc.).

Intangible entry criteria include:

- Required research.

- Required coordination.

- Required authorization.

Required research is similar to required products, except that typically there is nothing tangible to represent the research. Required coordination is simply a means to establish coordinating relationships between events that otherwise have nothing in common; they do not pass products or research between each other. Nevertheless, it is often necessary to coordinate events due to considerations that are not otherwise reflected in the model. The required coordination fields (which can specify conditions in terms of the state of other events) provide a means to easily represent such criteria. Required authorization is (within this guidebook) modeled as constraints (specifically, internal constraints). The purpose of required authorization is to facilitate modeling those types of events whereby they cannot start until someone (with sufficient authority) gives permission for the event to start. In all these examples, the entry criteria have been intangible.

Tangible exit criteria are entirely in terms of products produced by a given event.

Intangible exit criteria include research conducted by an event, and may also include coordination (such as discussed above), and satisfaction of either (or both) external and internal constraints (discussed further below).

With **Entry Criteria** and **Exit Criteria**, these fields can also be used to document logical relations. For example, the entry criteria to an event might be logically stated as "(Activity-A and Activity-B must be in a state of Done) or (Activity-A must be in a state of Done and (Activity B must be in a state of Mostly_Done or Activity-C must be in a state of Done)) or Product-P must be in a state of Completed." As discussed in Section 4.6.2, one of process modeling's goals is to keep the complexity of such logical relations at an absolute minimum.

Level #	Event Template	Version # and Date
	Template Type	

Name

Unique Identifier

Purpose

Comments

Additional States

Entry Criteria

Internal Processing

Exit Criteria

Parent Event(s)

Child Events

Revision History

Figure 3-13. Event Template

The **Internal Processing** field describes the work that occurs within the event being defined. This description is typically composed of a series of steps detailing how the work should proceed. This description usually contains a variety of logical constructs (such as "if" statements, "while" loops, etc.) to accurately capture the details of the work.

These templates also contain a field for **Additional States**. Events typically comprise two or more sub-events (activities or tasks) and often require progress descriptions providing more detail than just "in_progress" (the only default state that conveys work being done). The process analyst may elect to replace the in_progress state with an enumerated set of terms. For example, progress through an activity might be described using the terms: Early_Stages, Underway, First_Build_Complete, Second_Build_Complete, Ready_For_Evaluation, Rework_Underway, and Passed_Evaluation.

The Event template also includes the fields **Child Events** and **Parent Event(s)**. The **Child Events** field is used to list the "first generation" activities and tasks that participate in this event. The **Parent Event(s)** field is used to specify those activities in which this event participates.

Two small (gray) regions of this template are left available for unique fields at the class level. The uniqueness is introduced as a function of priorities. As shown on the following page, the Management template gives more distinction to constraints, whereas the production template gives more distinction to throughputs.

3.2.2.1 Management Templates

The first class under Event is the Management template (see Figure 3-14).
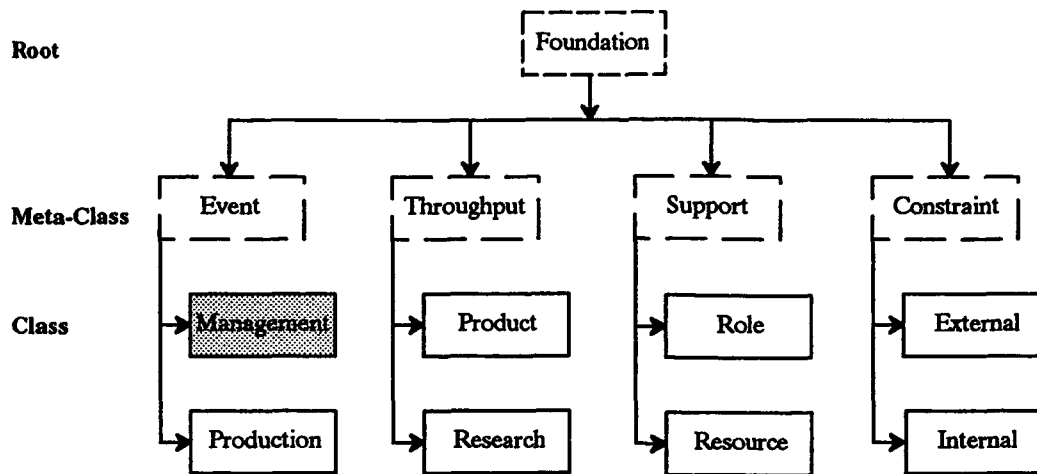
Figure 3-14. Template Structure (Generic)

In this version of the templates, there is little that explicitly distinguishes management events from production events. Furthermore, many production-oriented tasks involve people who have the authority to make management-level decisions, and many management-oriented tasks can involve production-specific work. From the perspective of process analysis, the rationale behind distinguishing between management activities and production activities is largely based on metrics. To facilitate the measurement and evaluation of alternative management and alternative production options, it is necessary to distinguish between the two. Is one management approach better than another with regard to how a particular process executes? Unless data distinguishes between management and production events, detailed comparative insights can be more difficult to achieve.

Both Management and Production templates have a region dedicated to noting applicable constraints. However, the Management template, depicted in Figure 3-15, includes specific fields that distinguish between internal constraints and external constraints. The **Internal Constraints** and **External Constraints** fields capture unique identifiers from those respective templates. These represent constraints on the management event that cannot be readily described in terms of supports, throughputs, or other events.

External constraints refer to those influences that constrain a process outside the responsibility of that process. For example, if the approval of a vice president is needed before some event can commence, that would be noted in the **Entry Criteria** field as an external constraint. Conversely, internal constraints are those influences that constrain a process but which are within the responsibility of that process. This includes virtually all authority that is associated with the roles used in support of the process. Each of these areas will be discussed in more detail as each template is discussed in turn.

Other fields on the Management template include **Throughputs** and **Supports**. Each of these fields is a cross-reference field that establishes relations between and among the templates. The **Throughputs** field is used for the unique identifiers of either or both Product or Research templates (which, in turn, define details of the throughputs that are input to, manipulated by, or output from the event). The **Supports** field is used for unique identifiers from Role and/or Resource templates. These templates define the people and the tools, respectively, needed for the event to occur.

Level #	Management Template Template Type	Version # and Date

Name	Unique Identifier

Purpose

Comments

Additional States

Entry Criteria	Internal Processing	Exit Criteria

Throughputs	Supports		Internal Constraints	External Constraints

Parent Event(s)

Child Events

Revision History

Figure 3-15. Management Template

3.2.2.2 Production Templates

The second class under Event is the Production template (see Figure 3-16).

Root   Foundation

Meta-Class   Event   Throughput   Support   Constraint

Class   →Management   → Product   →   Role   →   External

         →Production   → Research   →   Resource   →   Internal
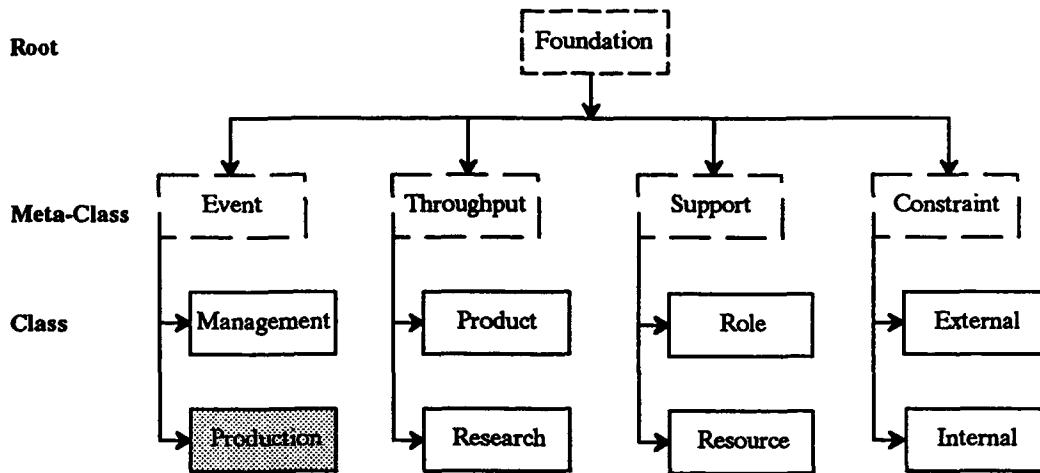
Figure 3-16. Template Structure (Generic)

The Production template (Figure 3-17) is fundamentally identical in form and content to the Management template. The Production template has entry criteria that are likely to be bound to the exit criteria of other events, activities, or tasks. Conversely, the Management template is more likely to be stated entirely in terms of the Constraint template (i.e., internal authorizations [via roles] or possibly directives as defined on external constraint templates).

As with Management template, the Production template contains a field for establishing references to applicable constraints. However, whereas the Management template specifically distinguishes between internal and external constraints, this is less important at the production level. Therefore, all constraints, regardless of whether they are internal or external, are noted together in the area reserved for constraints.

The **Throughput** and **Support** fields are also available on the Production template, except that throughput is explicitly divided into product-based throughputs and research-based throughputs. Hence, on the Production template, the region reserved for noting throughputs is divided into dedicated space for noting and distinguishing between products and research.

Again, throughputs, supports, products, research, internal and external constraints are all examined more closely in the following pages as their respective templates are discussed.

Level #	Production Template	Version # and Date
	Template Type	

Name	Unique Identifier

Purpose

Comments

Additional States

Entry Criteria	Internal Processing	Exit Criteria

Products	Supports		Constraints

Parent Event(s)

Child Events

Research

Revision History

Figure 3-17. Production Template

3.2.3 THROUGHPUT TEMPLATES

The Throughput template (Figure 3-18) consists of Product templates and Research templates. These and similar items are modeled as throughputs because they tend to be passed from one event to another. That is, the output products or research of one event are used as input products or research to later events. This continues until no more events are required and the throughput is considered finished.
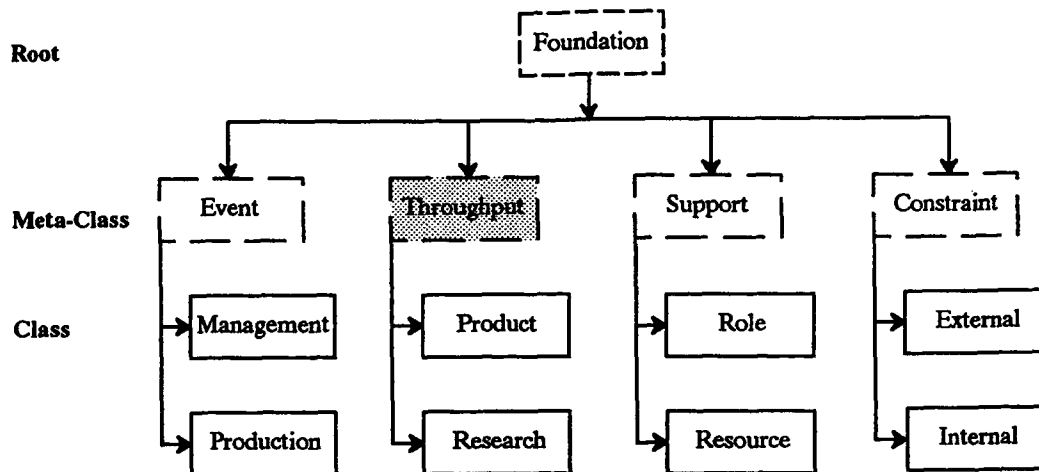
Figure 3-18. Template Structure (Generic)

Generally, it is ideal to attempt to model throughputs as tangible throughputs, i.e., products. Hence, the Research template should be rarely used; its primary purpose is to capture important relationships between events in which nothing physical is available to support the model. For example, often when research is done, some type of paper or document is produced. Maybe it is just a simple "white paper" or perhaps a checklist or form where someone states that something is done. It could be just a weekly or monthly status report. In all these cases, it is important to note that something physical, or tangible, is produced and can be used in constructing the process model (e.g., "Status-Report-A must be in a state of Complete before Activity-C can commence"). A problem arises, however, when even incidental products such as these are not available to represent work that is occurring. In such cases, the Research template can be used and that portion of the process model can specify relationships in terms of intangibles. Again, this is less than ideal from the modeling perspective, and research templates should only be used when no other alternatives exist.

The Throughput template (see Figure 3-19) adds two fields common to both the Product and Research templates: **Composed Of** and **Part Of**. The **Composed Of** field defines the subproducts used to build a product, and the sub-research efforts of which a given stage of research is composed. Both products and research can have multiple "parents" in the sense that they can be used in a variety of higher level outputs. Note, however, that products are only composed of other products, and research is only composed of other research. Although research may be required to support the development of a particular product (and occasionally, vice versa), this binding would occur within Event templates and not within the Throughput templates. The **Part Of** field defines the opposite relationship of **Composed Of**. The "parent" throughput (if applicable) of the throughput item currently being defined is listed in the **Part Of** field.

The **Evolves From Events** field lists those processes, activities, and tasks that contribute to the particular output. Often only the highest levels are listed. If an activity is needed to produce an output, and that

Level #	Throughput Template	Version # and Date
	Template Type	

Name   **Unique Identifier**

Purpose

Comments

Additional States	Descriptions	Part Of
		Composed Of
Evolves From Events	**State Transitions (Event/Step)**	
		External Constraints
Revision History		

Figure 3-19. Throughput Template

3. Process Definition Templates

activity is composed of two tasks, only the activity needs to be listed. If only one task within that activity is relevant to producing the described output, just that task is listed. However, there are times when a throughput may reference explicitly events that are not at the "highest level." In such cases, the **Evolves From Event** field should show events at whatever level they are referenced. This is important because this is a cross-reference field (i.e., on the Event template, there is a field for indicating throughputs; on the Throughput template, this is the field for indicating events). In all cases, explicit references between templates need to match each other.

As with the Support and Event templates, it can be useful to have a state attribute associated with the throughput item being described. These states allow more explicit logical relations to be established within the **Entry Criteria** and **Exit Criteria** fields. For Throughput templates, the information is captured in the **Additional States** and **State Transition (Event/Step)** fields. In the case of Throughput templates, the recommended state terms are:

- *Unauthorized.* The described throughput has not yet been authorized; therefore, it cannot be created and no work (processes, activities, or events) can commence yet.

- *Authorized.* The throughput item is authorized. Any events that commence work on this item can begin (given that all other event-specific criteria have also been satisfied).

- *In Progress.* The throughput is under development.

- *In Rework.* The throughput did not successfully pass quality or other compliance criteria and is being reworked.

- *Disabled.* Work on this throughput item has ceased due to a lack of necessary resources.

- *Suspended.* Work on this throughput item has ceased due to a removal of work authorization. When permission is received, work will recommence.

- *Cancelled.* Work on this throughput has ceased and no further work is contemplated or expected. The throughput is not complete; it has been abandoned.

- *Completed.* All planned work on this throughput is done.

Finally, there is a field on the Throughput template for listing **External Constraints**. This is a cross-reference field and is used to bind throughputs and applicable external constraints to each other. External constraints are discussed in detail later; but in general, they are used to represent influences on a process that originate from outside that process.

The Throughput template consists of Product templates and Research templates. Currently, both templates are comprised of the same set of fields. As a result, the following material does not introduce any new fields.

3-21

3.2.3.1 Product Templates

The first template under Throughput is Product (see Figure 3-20).
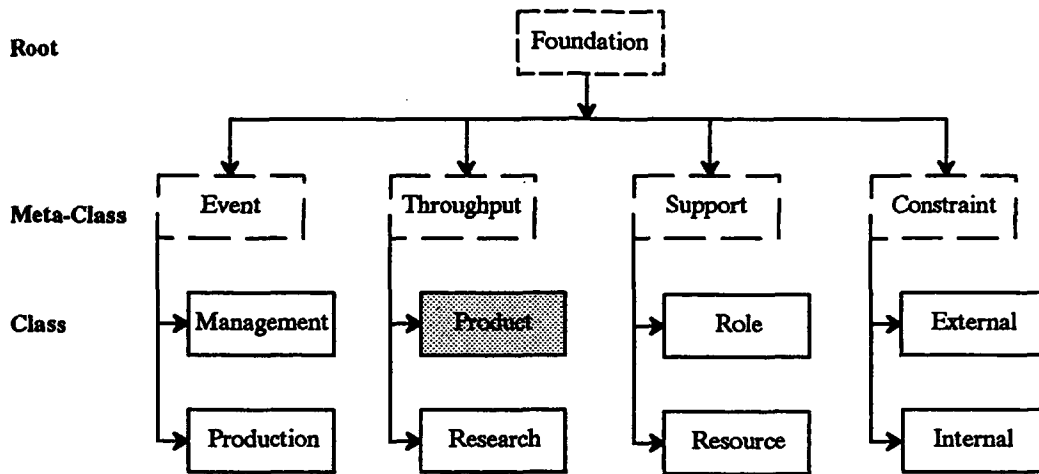
Figure 3-20. Template Structure (Generic)

The Product template (Figure 3-21) defines the tangible artifacts that result from one or more contributing events. These artifacts can be software modules, documentation, reports, results of test-runs, etc. As discussed earlier, virtually anything tangible can be modeled as a product; therefore, checklists are products: as are status reports, forms, invoices, and logs.

When defining a proposed process model, the definition should occur in terms of actual products. If necessary, incidental products (forms or checklists) should be proposed as part of the overall recommended model. When there are no actual products to track, it can be difficult to represent, manage, or audit the process that actually occurs. Tangible products, by there mere existence, leave actual evidence of events having occurred. (They at least document that specific individuals are claiming the events occurred.) Such evidence is highly beneficial in analyzing existing processes and in proposing new processes.

Level #	Product Template	Version # and Date
	Template Type	

Name

Unique Identifier

Purpose

Comments

Additional States	Descriptions	Part Of
		Composed Of
Evolves From Events	**State Transitions (Event/Step)**	
		External Constraints

Revision History

Figure 3-21. Product Template

3.2.3.2 Research Templates

The Research template (Figure 3-22) is the second template under Throughput.
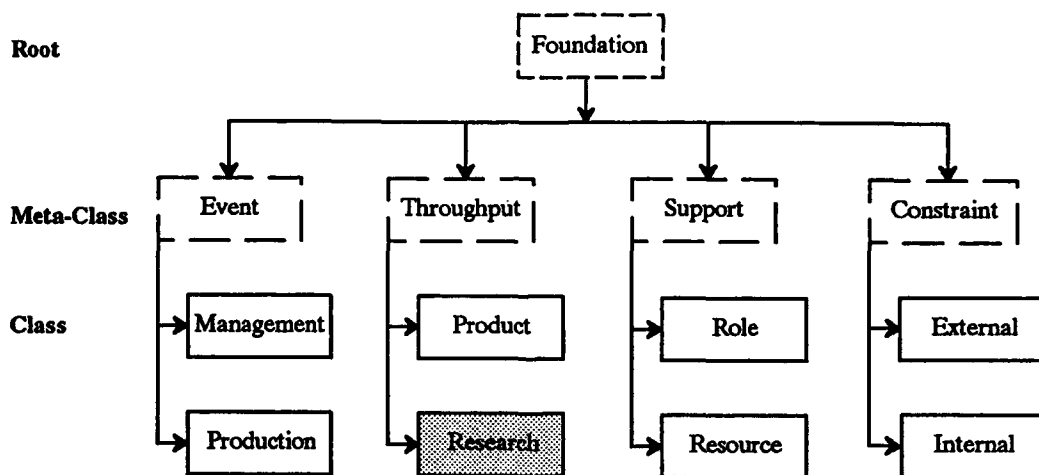
Figure 3-22. Template Structure (Generic)

Research templates (Figure 3-23) describe the intangible artifacts that result from one or more interacting events. These templates are a means to capture the fact that there are times when valuable work is done, but no tangible by-products are created. Examples include surveys, investigations, experiments, and all other forms of information gathering or evaluation.

As mentioned earlier, you should note that various research efforts often will produce some type of white-paper, checklist, summary of findings, progress report, or a chronological log of the effort performed. In such cases, the process analyst should model that artifact as tangible, and use the Product templates. As a general rule, it is easier to analyze, monitor, and model tangible, as opposed to intangible, artifacts. The Research template exists as an aid in those cases where nothing else will work.

Level #	Research Template	Version # and Date
	Template Type	

Name

Unique Identifier

Purpose

Comments

Additional States	Descriptions	Part Of
		Composed Of
Evolves From Events	**State Transitions (Event/Step)**	
		External Constraints

Revision History

Figure 3-23. Research Template

3.2.4 SUPPORT TEMPLATES

Support templates (Figure 3-24) consist of Role templates and Resource templates.
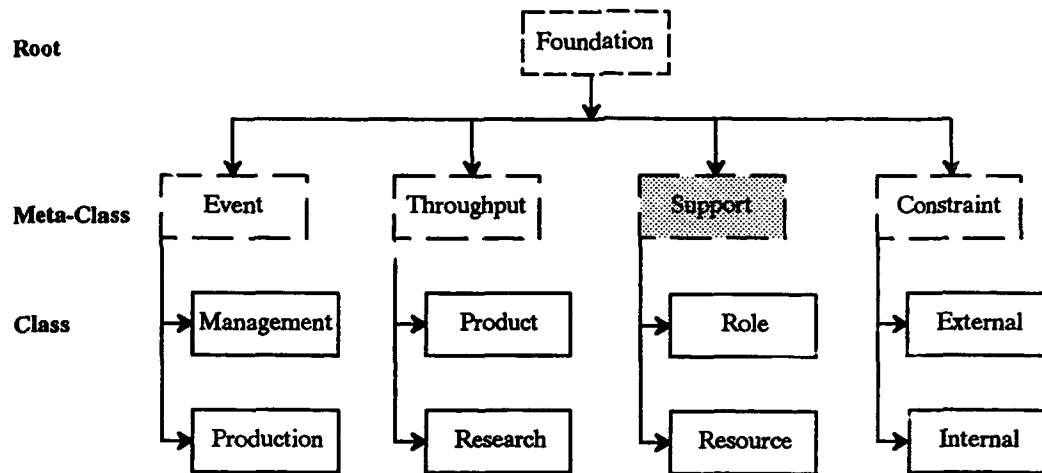
Figure 3-24. Template Structure (Generic)

Support templates (Figure 3-25) add several fields common to both Role and Resource templates: **Additional States/Descriptions, Part Of, Composed Of,** and **Supported Events.**

Composed Of and **Part Of** define the child and parent relationships, respectively, within support items. As related in an earlier example, the role of engineering_team might be composed of a team_leader, primary_engineers, and support_personnel.

The **Supported Events** field lists all processes, activities, and tasks which require the role or resource. Usually, only the highest level support relationships are listed; therefore, if a resource is used by all the tasks within an activity, only the activity needs to be listed as a supported event. Note, however, that to determine the total set of roles (or resources) required by an event, it is necessary to combine the roles (or resources) explicitly defined for that event with all those defined for all subordinate events of which it is comprised. Since this is a cross-reference field (binding support templates with event templates), any explicit reference by an event to a support results in a corresponding reference from the Support template to that specific event (using the **Supported Events** field).

As with the Throughput and Event templates, it is useful to have a state attribute associated with the support item described. In the case of Support templates, any additional states can be detailed in the **Additional States/Descriptions** field. As reflected by this set of state terms, a particular person, team, or department (i.e., roles) may need to be available_exclusively or available_shared for part or all of an event. These states make such criteria more rigorous and potentially more formal. The recommended default set of state terms is:

- *Available_Exclusively.* The support item is available for either shared or exclusive use.

- *Available_Shared.* This item is in use by one or more events, but it is still available for shared use by other events.

- *Not_Available.* This item is in use exclusively or is otherwise engaged to capacity and is not available for further uses (until one or more current uses are released).
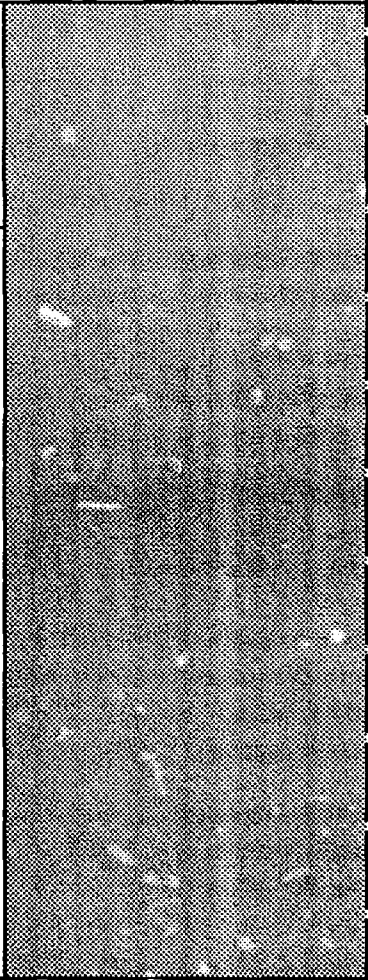
Level #	Support Template	Version # and Date
	Template Type	

Name	Unique Identifier

Purpose

Comments

Additional States/Descriptions

Part Of

Composed Of

Supported Events

Revision History

Figure 3-25. Support Template

- *Disabled.* This item is not available for use due to the absence of one or more critical components needed for this item to function.

- *Suspended.* This item has been declared "not available for use." Although no critical components are absent (i.e., the item is not disabled), a decree has been issued to the effect that this item (at least temporarily) cannot be used by any event.

3.2.4.1 Role Templates

The Role template (Figure 3-26) is the first of the Support templates.
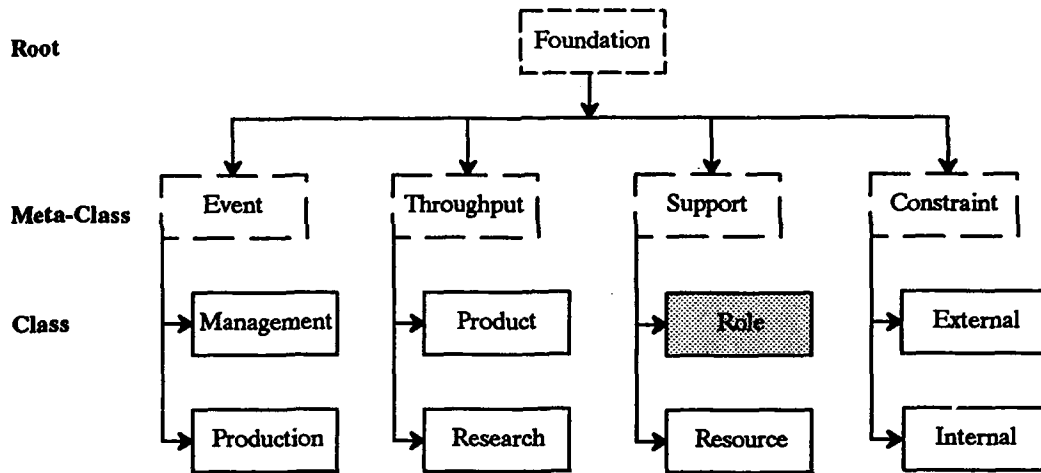
Figure 3-26. Template Structure (Generic)

Role templates (depicted in Figure 3-27) include an **Associated Authority** field not shown on the Support template. Such authority is defined in the Internal Constraints template; this field serves as a cross-reference field between the Role templates and Internal Constraint templates. As discussed under the Internal Constraint template, the authority associated with roles is represented as the authority for someone acting in a given role to declare that some type of state change has occurred. Ideally, an activity is modeled as completed when all the tasks comprising that activity are completed. In certain environments, however, someone in a manager's role might have the authority to declare that a given activity is "complete" even though some of the tasks within that activity have not been performed. If such authority exists, it needs to be captured within the process model. This can be done through the joint use of Role templates and Internal Constraint templates (detailed examples of this are presented in Appendix A).

Also note that Role templates can be subject to the **External Constraints** field. Although this is a common field shared with Resource templates, its importance differs significantly between these templates (as does the space allocated). For Role templates, there are likely to be comparatively few **External Constraints,** and they can be represented by a single source such as a manual containing position descriptions, experience required for those positions, and scope of authority.

Level #	Role Template	Version # and Date
	Template Type	

Name	Unique Identifier

Purpose

Comments

Additional States/Descriptions	**Associated Authority (via Internal Constraints) and Applicable Events**

Part Of	**Supported Events**	
Composed Of		
		External Constraints

Revision History

Figure 3-27. Role Template

3.2.4.2 Resource Templates

The Resource template (Figure 3-28) is the second class of templates under Support.
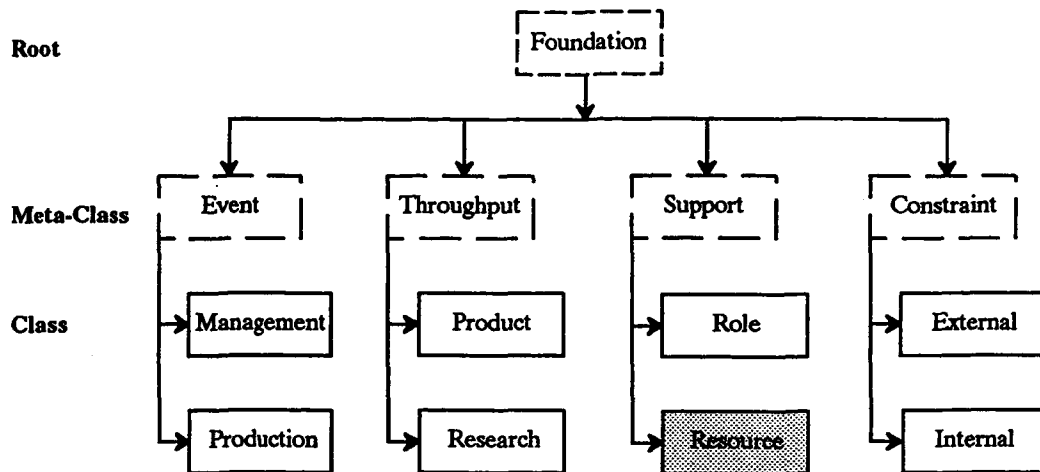
Figure 3-28. Template Structure (Generic)

Resource templates (Figure 3-29) contain two new fields: **Operational Guidelines/Limits** and **Required Supplies/Materials**. They provide a larger space for cross-referencing to **External Constraints.**

Both the **Operational Guidelines/Limits** field and the **Required Supplies/Materials** field provide information that can be of considerable significance with dynamic limitations placed on the availability of a given resource. For example, a particular process may require that a delivery_van resource be available to make a daily run to the post office. Implied in this is the availability of a driver for the van. Instead of developing a large collection of Support templates to capture such interdependencies, the required supplies/material can be used to document these necessities.

The **Operational Guidelines/Limits** field can also be used to document constraints or issues that impact the availability or usage of a given resource. For example, suppose there was an informal policy governing the use of a particular machine requiring that no one under the age of 18 be allowed to use it without supervision. These types of constraints, which might be difficult to capture can easily be represented using descriptive text. At a minimum, this field can be used as a temporary expedient to capture constraints that will later be replaced by more formal (and probably more complex) representations.

As on other templates, any documented policies, procedures, guidelines, or standards can be represented via the **External Constraints** field. Also note that when a resource is shared with or used by a process external to the one being modeled, the logistics of sharing or availability can be captured using external constraints.

Level #	Resource Template	Version # and Date
	Template Type	

Name	Unique Identifier

Purpose

Comments

Additional States/Descriptions

Operational Guidelines/Limits

Part Of

Supported Events

Composed Of

Required Supplies/Materials

External Constraints

Revision History

Figure 3-29. Resource Template

3.2.5 CONSTRAINT TEMPLATES

Constraint templates capture the influences that govern any set of events and simplify the definition and representation of the corresponding control structure. This guidebook suggests two classes of Constraint templates (Figure 3-30).
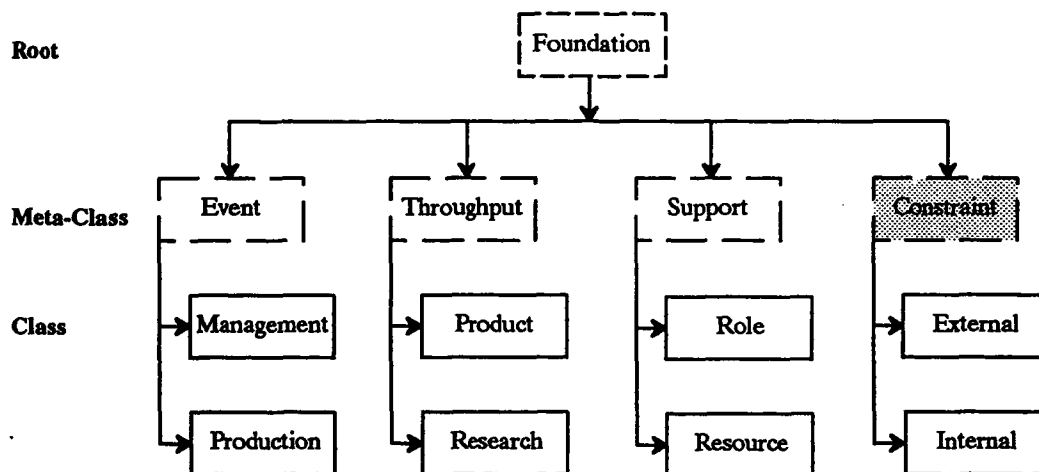
Figure 3-30. Template Structure (Generic)

The first class is the External Constraint template. This template can be used to represent controlling or governing influences exerted on a process that originate from outside the scope of the process being modeled.

The second class of Constraint templates is the Internal Constraints template, and it is used to represent internal constraints within a process. Internal constraints are primarily modeled as the authority of various roles to declare state transitions with respect to events, throughputs, and supports. In all cases, this class of Constraint templates is associated with one or more roles. The net result of this effort yields a process model that explicitly (and often more simply) models the influence of management and authority within a process description.

The Constraint template (Figure 3-31) shares three new common fields (in addition to those inherited from the Foundation template): **Special Form Of, General Form Of,** and **Constrained Events.**

The **Special Form Of** and **General Form Of** fields designate parent and child relationships, respectively. These relationships can be of virtually any kind (keeping in mind that the Constraint templates are used for representing all important constraints on a process readily captured using Throughput, Support, and Event templates). Two major types of relationships relevant to the **Special Form Of** and **General Form Of** fields are "whole/part" relationships, and "generalized/specialized" relationships.

An example of whole/part relations would be an aircraft: the "whole" of the aircraft is comprised of definite "parts" such as wings, power plant, and fuselage. Conversely, the "generalized" concept of an aircraft is an abstraction of the more "specialized" concepts of "propeller-driven aircraft," "jet-propelled aircraft," and "fixed-wing aircraft." Both types of relationships provide very useful insights into the relationships between concepts at a higher level and concepts at a lower level. Since constraints can be virtually any kind, it is worth noting that the relationships used to represent "layers" of those constraints can likewise be any kind.

Level #	Constraint Template	Version # and Date
	Template Type	

Name	Unique Identifier

Purpose

Comments

Special Form Of (parent)	

General Form Of (list children)	Constrained Events	

Revision History

Figure 3-31. Constraint Template

Finally, the Constrained Events field is used as a cross-reference to the Event template. Whenever an event lists a constraint as applying to that event, the constraint should show a corresponding reverse reference indicating the event being constrained.

3.2.5.1 External Constraints Templates

The External Constraint template (Figure 3-32) is the first class of Constraint templates.
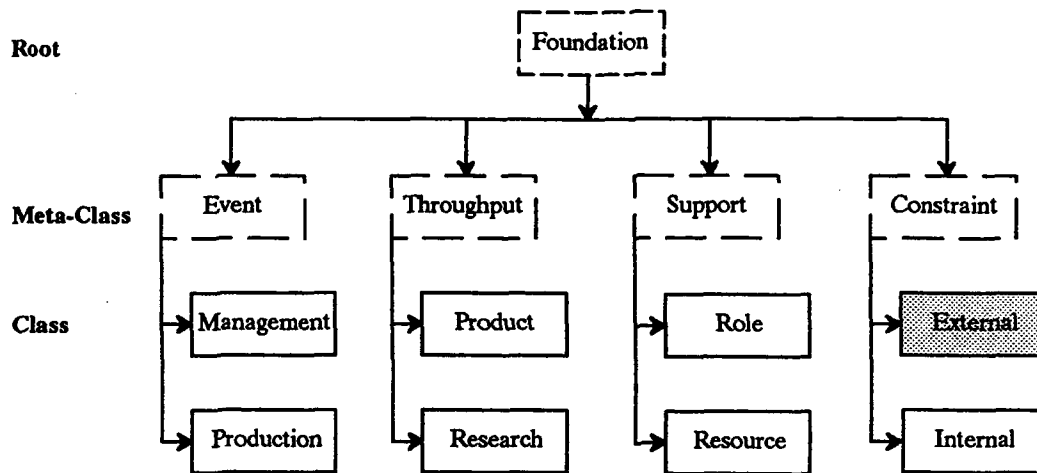
Figure 3-32. Template Structure (Generic)

The primary purpose of the External Constraint template (Figure 3-33) is to capture those controls that constrain a process but which originate from outside the scope of the current process model. Examples of possible external sources of constraint include people, policies, regulations, projects, and weather conditions (when an activity, for instance, has to occur outdoors).

This template is the most expedient means of coordinating independent process models while keeping inter-process coupling at a minimum.

Suggested states for this template include:

- *Compliance Unknown.* This state implies that from the perspective of a given throughput, support, or event, compliance with the external constraint has not yet been evaluated.

- *Compliance Under Evaluation.* In this state, external constraint is being actively referenced to evaluate compliance.

- *Compliance Achieved.* This state indicates that compliance with the external constraint has been achieved. Typically, this allows corresponding state changes in either throughputs or events.

- *Compliance Failure.* This state indicates that from the perspective of the referenced external constraint, a throughput item or an event has failed to achieve compliance with the constraint. This usually either prevents throughput or event state transitions or else causes transitions into states similar to in_rework, etc.

- *Compliance Waived.* When an external constraint is in this state, its constraining influence is removed (only temporarily).

In all cases, note that external constraint states are only meaningful with regard to some particular event or throughput. Therefore, although a particular style guide might be an external constraint for four different events, the state of that external constraint would always be a function of each event's

Level #	External Constraint Template Template Type	Version # and Date

Name	Unique Identilier

Purpose

Comments

Special Form Of (parent)	Additional States/Description

General Form Of (list children)	Constrained Events	Constrained Throughputs
		Constrained Supports

Revision History

Figure 3-33. External Constraint Template

frame of reference. For one event, the state of the style guide may be compliance achieved, for another event the state of the style guide may be compliance failure, and for yet another event the state of the style guide may be compliance waived. Just as internal constraints are always coupled with roles (discussed below), external constraints always coupled with events, throughputs, or supports.

The External Constraint template also contains the fields **Constrained Throughputs** and **Constrained Supports**. (Although these fields are common with internal constraints, they are given considerably more space on the External Constraint template. This is because internal constraints are defined in terms of roles, and will likely have less applicability to non-event issues.) The **Constrained Events**, **Constrained Throughputs**, and **Constrained Supports** fields can be used to facilitate cross-referencing between External Constraint templates and Event, Throughput, and Support templates. Although the evaluation of compliance with an external constraint must occur within some type of event, it can be quite convenient to also detail all the constraints (such as system design standards and coding standards) that are applicable to a given throughput (such as a software product).

3.2.5.2 Internal Constraint Templates

Internal Constraints (Figure 3-34) is the second class of templates under Constraint.
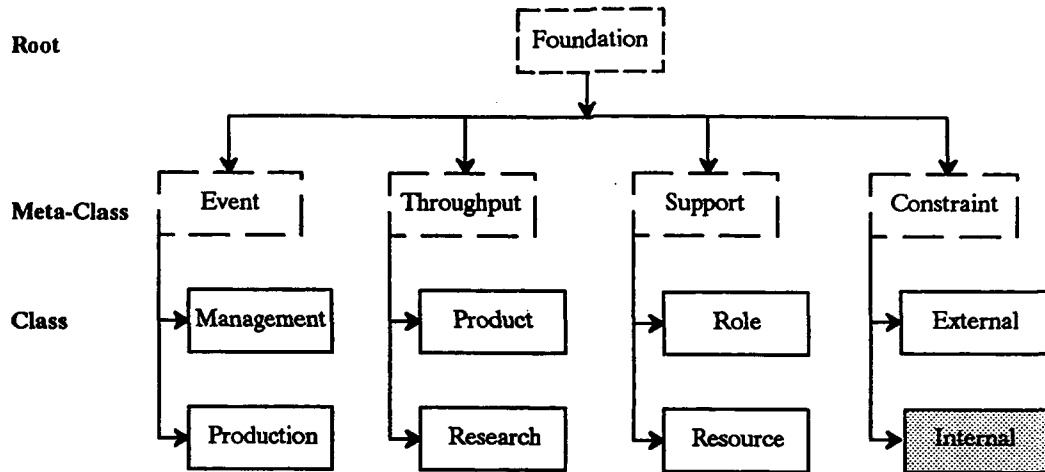
Figure 3-34. Template Structure (Generic)

The primary purpose of Internal Constraint templates (Figure 3-35) is to model various classes of permission. (It must be emphasized that the use of these templates can be expanded to model all types of internal constraints.) These templates either make possible or simplify the definition and representation of the control structures that coordinate and govern various events.

One type of event flow control mechanism discussed earlier describes entry conditions for events in terms of satisfied exit conditions of other events. This, however, introduces at least two problems. First, it makes parallelism more difficult to capture. That is, suppose event "B" usually starts once event "A" is almost done. Forcing the use of exit conditions could result in a definition that states "A" must finish before "B" may start. A second problem is exponentially increasing complexity. If dozens of exit conditions must be satisfied by numerous events before a given "downstream" event can start, it may be easier to simply describe the downstream event as needing management authorization to start. You accomplish this by asserting in the entry criteria in an event that commencement (permission) must be granted by someone in an appropriate role (i.e., a role that has been granted such authority).

The Internal Constraint templates allow the process engineer to define classes of permission, but such permission does not directly map to events. Instead, Internal Constraint templates define permission classes associated with roles. Roles, in turn, are one of the Support templates that describe (in combination with Resource templates) the support items required by events. As implied by the parallel boxes in Figure 3-35, any event subject to an internal constraint is also matched with an associated role that can grant (or withhold) the necessary permission represented by that constraint.

At a minimum, any set of templates should define the following set of internal constraints. This list of permissions serves as a subset of the internal constraints that govern a particular process model.

- *Commencement Permission.* This permission, when associated with a role, allows people within that role to grant permission to commence work associated with an event (assuming that all other entry criteria have been satisfied).

- *Suspension Permission.* This permission, when associated with a role, allows people within that role to decree that all work on a given event cease.

Level #	Internal Constraint Template	Version # and Date
	Template Type	

Name		Unique Identifier

Purpose

Comments

Special Form Of (parent)	Constrained Throughputs	Constrained Supports
General Form Of (list children)	**Constrained Events**	**Associated Roles**

Revision History

Figure 3-35. Internal Constraint Template

- *Recommencement Permission.* This permission allows one to authorize those associated with a ceased event to resume work.

- *Cancellation Permission.* This permission allows one to cancel work associated with a given event. Such cancellation is considered to be permanent (whereas suspensions are temporary).

- *Resurrection Permission.* This permission allows reversing a cancellation decision and recommencing work in an area that previously had been "permanently" cancelled.

- *Completion Permission.* Someone in a role which has this permission can decree that the work associated with an event has been successfully completed (assuming that all other exit criteria have been satisfied).

- *Override Permission.* Someone in a role which has this permission can decree that the work associated with an event has been successfully completed despite the fact that other exit criteria still remain to be satisfied. Similarly, events can be ordered to start even though one or more entry criteria remain to be satisfied.

- *Executive Permission.* Someone in a role which has this permission can decree that the work associated with an event has been successfully completed, and further, they can decree that currently unsatisfied exit criteria are, in fact, satisfied. Similarly, entry conditions can be declared "satisfied" and events ordered to start.

Again, the Consortium suggests that all process models establish a basic set of Internal Constraint templates that include the above internal constraints. Much of what governs the ongoing execution of a process are people in various roles of authority granting or withholding permissions at various stages of work. The Internal Constraint templates readily capture and formally define the relationships between these constraints and the other factors discussed (i.e., external constraints, throughputs, supports, etc.).

It should be noted that internal constraints, as presented here, do not carry the concept of state: internal constraints essentially represent the authority of someone in a given role to declare a state change for an event, a throughput, or a support. To attempt to represent a change of state as itself being subject to changes of state is an unnecessary complexity that adds little, if any, value. Instead, internal constraints are matched directly with the states of other items in the process model (especially event states reflected by the recommended default set of internal constraints described above). If it is possible for anyone to declare that a state change has occurred, that state change:

- Must be represented as having an internal constraint which represents authority to issue such declarations.

- Must be represented by binding that internal constraint to an explicit role (defined using the role templates).

Whenever that role is modeled as participating in a given event, that role can declare any of the state changes for which it is authorized.

It should also be noted that on the Role template the cross-reference field to the Internal Constraint template allows stating that a particular authority is available to a given role only with respect to an

applicable set of events: a manager with the authority to declare some event types to be complete does not mean they have the authority to declare all event types complete. If it is necessary to restrict a role's use of authority granted by an Internal Constraint template; those restrictions are detailed on the role template, and not the internal constraint template.

3.3 TEMPLATES AND GRAPHICAL MODELS

The templates are essentially a text-based device for capturing and organizing information relative to a process, and further details regarding their usage is presented in Section 4. However, when a process has been defined using, for example, hundreds of templates, it can be quite difficult for someone to efficiently use the templates to get a high level understanding of the overall process. To overcome this problem, the graphical notation described in this section can be used in conjunction with the templates to provide diagrammatic abstractions of the process model.

This graphical notation consists primarily of objects and relations. As shown in Figure 3-36, there are four types of process modeling shapes for depicting process objects:
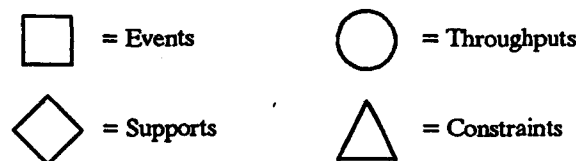
Figure 3-36. Process Object Modeling Shapes

As shown in Figure 3-37 below, these process objects, or elements, can be connected to each other by four different types of relations. Sequence relations are used to connect one event to another and convey a temporal ordering (e.g., event B happens after event A, but before event C). Inclusion relations show parent/child or "part of" relations. For example, inspection preparation activity may be "part of" an overall inspection process. Specialization relations show relationships between the general form and the special form of something. That is, "programmer" may be a general form, whereas "Ada programmer," "C programmer," etc., would be special forms (i.e., a specialized form of the more general term). Finally, reference relations represent any relation between process objects aside from sequence, inclusion, and specialization. The graphical form of these relations is shown in Figure 3-37.
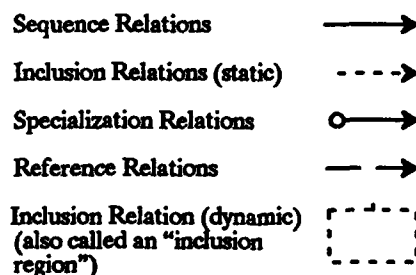
Figure 3-37. Process Relationship Modeling Shapes

There are several rules that define the use of relationships. Sequence relations can only connect one event with another event. Static inclusion relations can only connect like elements. For example an inclusion relation may bind an event to a subevent or a product to a subproduct, but it never binds an event to a subproduct. Dynamic inclusion relations (valid only for events) combine a set of process elements as all working in concert within the parent event (examples are provided in the following

pages). Specialization relations, similar to static inclusion relations, can only connect like elements. For graphical simplicity, all relations, or arcs, may split or join with other arcs. However, care must be taken to ensure that ambiguity is not introduced.

Any element can be decomposed either through using the inclusion relation or by indicating that an expanded depiction of the inclusion relation is detailed on another page. As shown in the following example, off-page references are indicated by using one of two different conventions. One approach is to indicate the off-page reference with a double border object that uses a broken or dotted line for the interior border. Alternatively, an off-page relation can also be shown through explicit use of an octagon object. Details of using these and additional techniques are provided in the following series of diagrams and supporting text.

The next five figures (Figure 3-38 through Figure 3-42) show a simple representation of a hybrid formal inspection process. Some organizations have adopted the use of inspection "SWAT" (Special Weapons and Tactics) teams whose primary purpose is to perform rapid, frequent, efficient, and effective inspections on specific domains of artifacts. This scenario presented in this set of figures outlines an example SWAT inspection team process. The first figure, SWAT–D1, shows the highest level view. SWAT–D2 shows an expanded view of the resources required to support this process. SWAT–D3 is an expanded view of the process improvement activities, and SWAT–D4A and –D4B show an expanded view of the inspection activities.

Figure 3-38 is an example of the primary process elements of a formal inspection process. The bold outline on the formal inspection process event (square) is used to aid a reader in quickly determining the central component of a diagram. In this example, the constraints (triangles), supports (diamond), throughputs (circles), and sub-events or activities (squares), are all shown in relation to the bold inspection process event.

Note that this example shows joining two reference relations from two separate constraints into one arc that intersects with the inspection process event. Joining arcs in relations is permissible, but only if the relations are identical (in this case, both are reference relations). Also note that arcs indicate flow. Hence, "material to be inspected" is shown as throughputs required by the inspection process event, and "inspected material" is shown as throughputs generated by or released from this process.

However, inspection metrics is shown as an outbound throughput; also, note that there is a slash across the arc just before the arrowhead. This slash indicates that this is an optional relation to an object that may or may not exist in an actual instantiation of this model. In this case, the diagram indicates that inspection metrics are an optional output of the inspection process.

The dotted-line arcs show inclusion relations. These relations on SWAT–D1 indicate that the inspection process is composed of (or includes) three major process activities: process improvement, the inspection activities, and causal analysis. For inclusion relations, the arrow always points from the child to the parent.

Finally, note on SWAT–D1 that three of the elements have dotted-line interior borders. One of these is the support (diamond) "Inspection resources," both the others are events (squares): "process improvement" and "inspection activities." The dotted-line interior border indicates that an inclusion relation exists but is shown on another page. In all cases, each such element should have an associated unique identifier that tells the reader which page has the expanded diagrams. As shown on SWAT–D1, expanded diagrams can be found on figures labelled SWAT–D2, SWAT–D3, and SWAT–D4A.
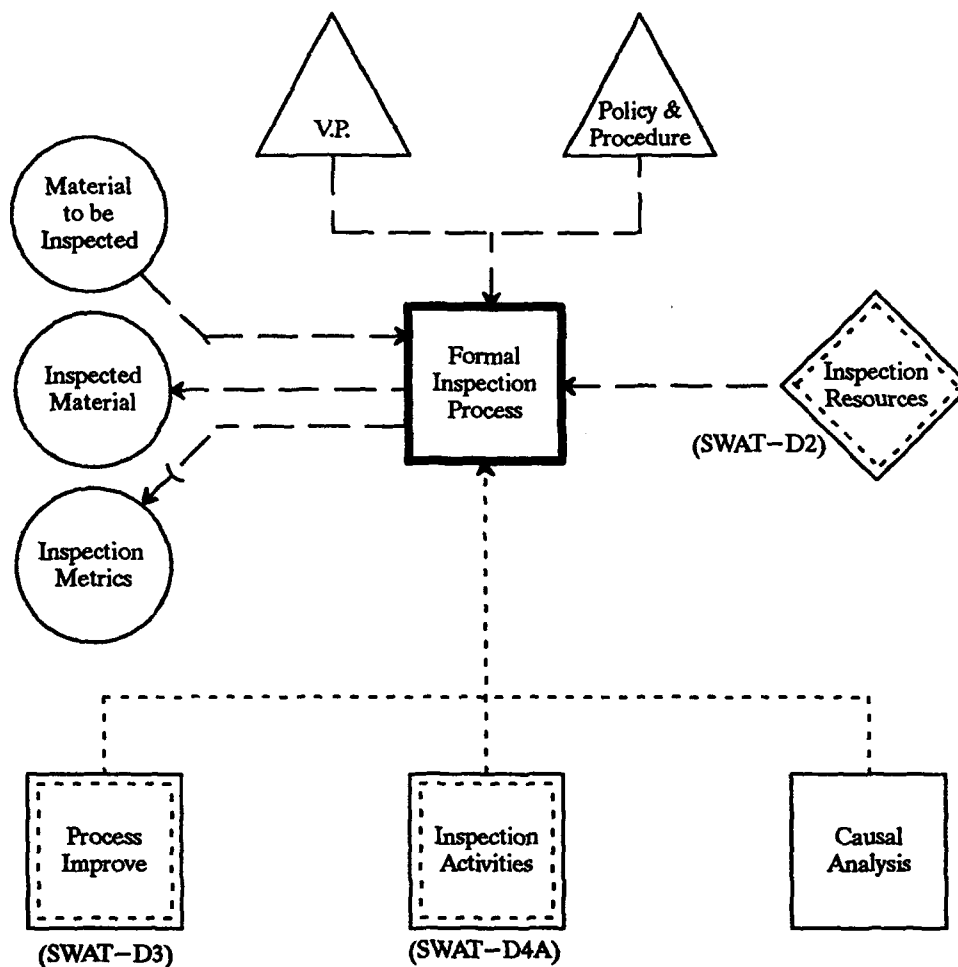
Figure 3-38. SWAT—D1

Any support, throughput, event, or constraint can be diagrammatically expanded using hierarchical decomposition techniques similar to that shown in SWAT—D2 (Figure 3-39). In this example, inspection resources are shown as a support that includes, as indicated by the dotted-line inclusion relation, super moderator resource(s), inspection team resource(s), and facility resource(s). Additional inclusion relations on the diagram show that inspection team resources are composed of moderator(s), reader(s), inspector(s), scribe(s), and author(s).

The inspector element has two specialization relations connecting to it from the two types of inspectors. This relation indicates that the general concept of an inspector may be used in some parts of the model; however, in actuality, inspectors are either key inspectors or regular inspectors.

SWAT—D2 also shows that inclusion relations should explicitly show cardinality. When indicating cardinality, there are at most two pairs of numbers (separated by a ":") used to indicate the cardinality of the inclusion relation. The first pair of numbers indicates, from the child's perspective, the minimum and maximum number of parents required. The second pair of numbers indicates, from the parents perspective, the minimum and maximum number of children required. Typically, one of three values is used: 0, 1, and N. Whenever the first and second number are identical, only a single instance of that value is used (so, "00:11" would simply be shown as "0:1").
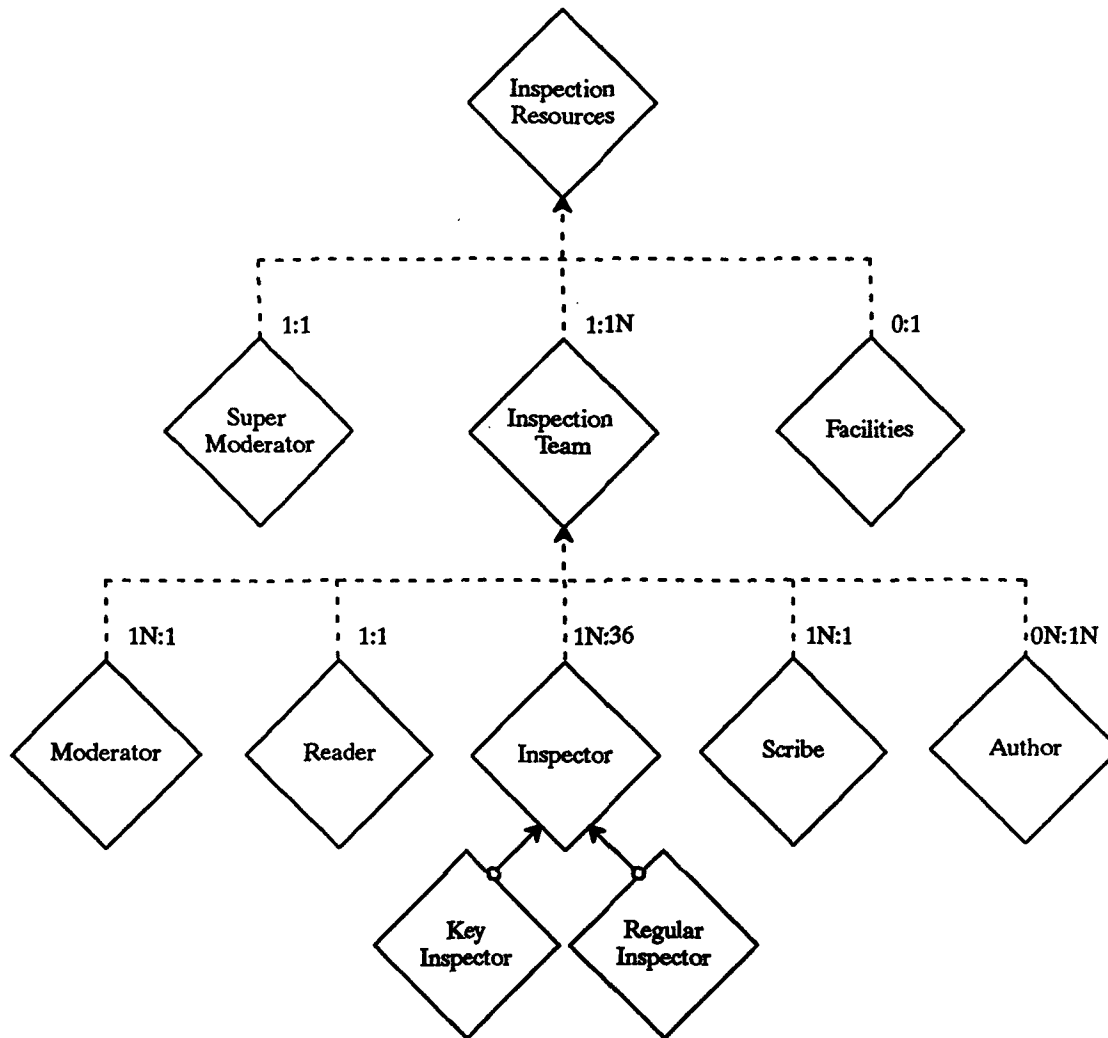
Figure 3-39. SWAT–D2

Cardinality labels on the SWAT–D2 diagram indicate the following. Look at the inclusion relations indicating the roles that make up an inspection team. The cardinality labels show that for a moderator to exist, there must be at least one inspection team but that one person may be the moderator of multiple (or "N") inspection teams (1N:...). From the perspective of an inspection team, there must be at least one moderator and at most one moderator for that team (...:1). For a reader to exist, there must be at least one inspection team and at most one inspection team (1:...). In effect, this states that the same person cannot simultaneously be a reader on more than one team. From the perspective of an inspection team, there must be at least, and at most, one reader for that team (...:1). For an inspector to exist, there must be at least one inspection team, but the same inspector can simultaneously participate on multiple inspection teams (1N:...). However, from the team perspective, each team requires at least three inspectors but is not allowed to have more than six inspectors (...:36). This is an example of where the more general form "N" has been replaced by explicit values; if either of the numbers requires two or more digits, a comma should be used to delimit the values. As shown in SWAT–D2, the cardinality of the Scribe is identical to that of the moderator. Finally, from the perspective of an author, the author requires zero inspection teams (that is, a person is an author of something independent whether an inspection team exists or not), but they may have authored products for "N" or

multiple inspection teams (0N:...). From the team perspective, there needs to be at least one author (whose work is being inspected); but there may be multiple authors, such as when, for instance, an entire design group develops one section of a high-level design document (...:1N).

Again, Figure 3-39 only shows the composition of one support object. Similar diagrams can be used to show the composition of anything. Furthermore, the decomposition need not necessarily be hierarchical. Certain items might be better represented by using a network or directed graph. In all cases, a strict ordering of elements is not required. However, each decomposition will typically proceed from a single, common root.

SWAT−D3 (Figure 3-40) is an expansion of the process improvement element shown on SWAT−D1 (Figure 3-38). As with SWAT−D1, this diagram gives a high-level view of the resources (diamond), constraints (triangle), throughputs (circles), and events (squares) that participate in this subprocess. Different from SWAT−D1, however, is the use of a large dotted-line box to show an immediately expanded view of the composition of the event "process improvement." This too is an inclusion relation (indicated by the dotted-line boundary), but because it defines a region (i.e., a box) it indicates a dynamic view of the elements included in "process improvement." The only difference between a dynamic view of an expanded object (shown on SWAT−D3, SWAT−D4A, and SWAT−D4B) and a static view (shown on SWAT−D1) is that the dynamic view connects the events with other events using sequence relations. That is, a static view states that a set of child events are included in a parent event, but it does not convey any information about in what order the child events might occur. The dynamic view both states that a set of child events are included in a parent event, and that the child events occur in some sequence or order. Additionally, as shown on SWAT−D3, one of the child events is shown with a bold outline to indicate that it is the waiting, default, or initiating activity in a series of activities. This is especially useful when (as shown in SWAT−D3) the events are essentially cyclic, and therefore there is no obvious starting place.

Note that SWAT−D3 also shows a pair of inclusion relations (terminating at the event, "hold meeting") and the use of optional reference relations. For instance, as an outbound throughput from the process improvement event, there is optional output of "new policy proposals." Also note the double-headed arc connecting the "summarize recommendations" event with the "recommendations database" throughput. In this example the double-headed arc indicates that the event "summarize recommendations" has the option of getting data from the recommendations database, and also has the option of outputting data to the database.

SWAT−D4A (Figure 3-41) and SWAT−D4B (Figure 3-42) together show an expansion of the "inspection activities" event. One main difference seen in this pair of diagrams is the use of octagon off-page connectors. The octagon shape is used to allow what is essentially a single picture to be shown across multiple pages. Note that any of the relations may enter into and exit from an off-page connector. Also notice that the off-page connectors are bidirectional. They may show outbound relations from the current page to a remote page and simultaneously show inbound relations from the remote page to the current page (as is shown on the off-page connector "SWAT_J2" on diagram SWAT−D4A [Figure 3-41]).

Again, the event expansion diagram shows the internal details of the "inspection activities" event. Since this pair of diagrams show dynamic process characteristics (that is, sequence-ordered events) the inclusion region is used in lieu of simple inclusion arcs. Typically, when a picture must be spread across multiple pages, one of those pages serves as the root diagram, and all other pages are either directly or indirectly bound to the root diagram by relations connecting through off-page symbols. In
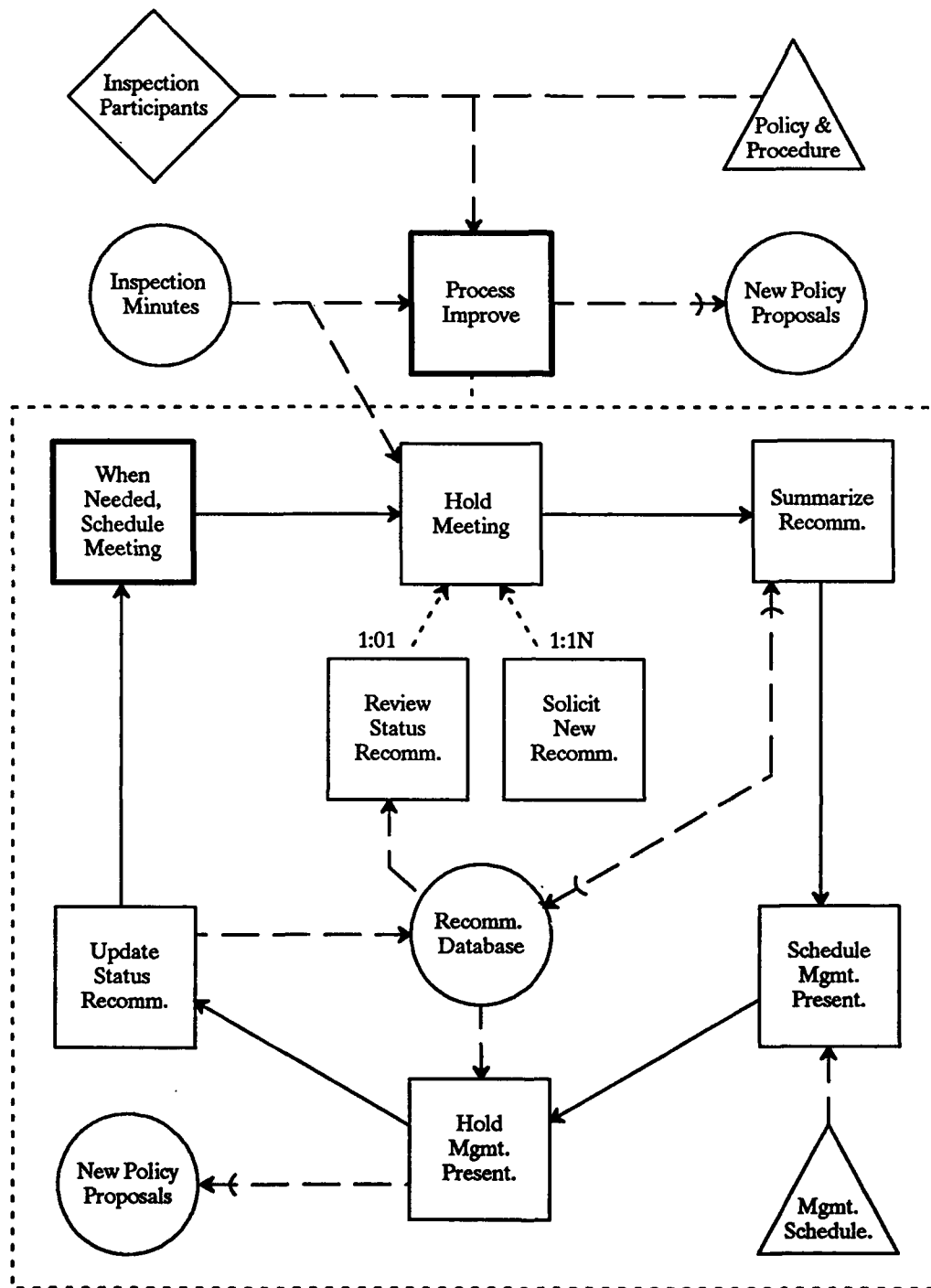
Figure 3-40. SWAT—D3
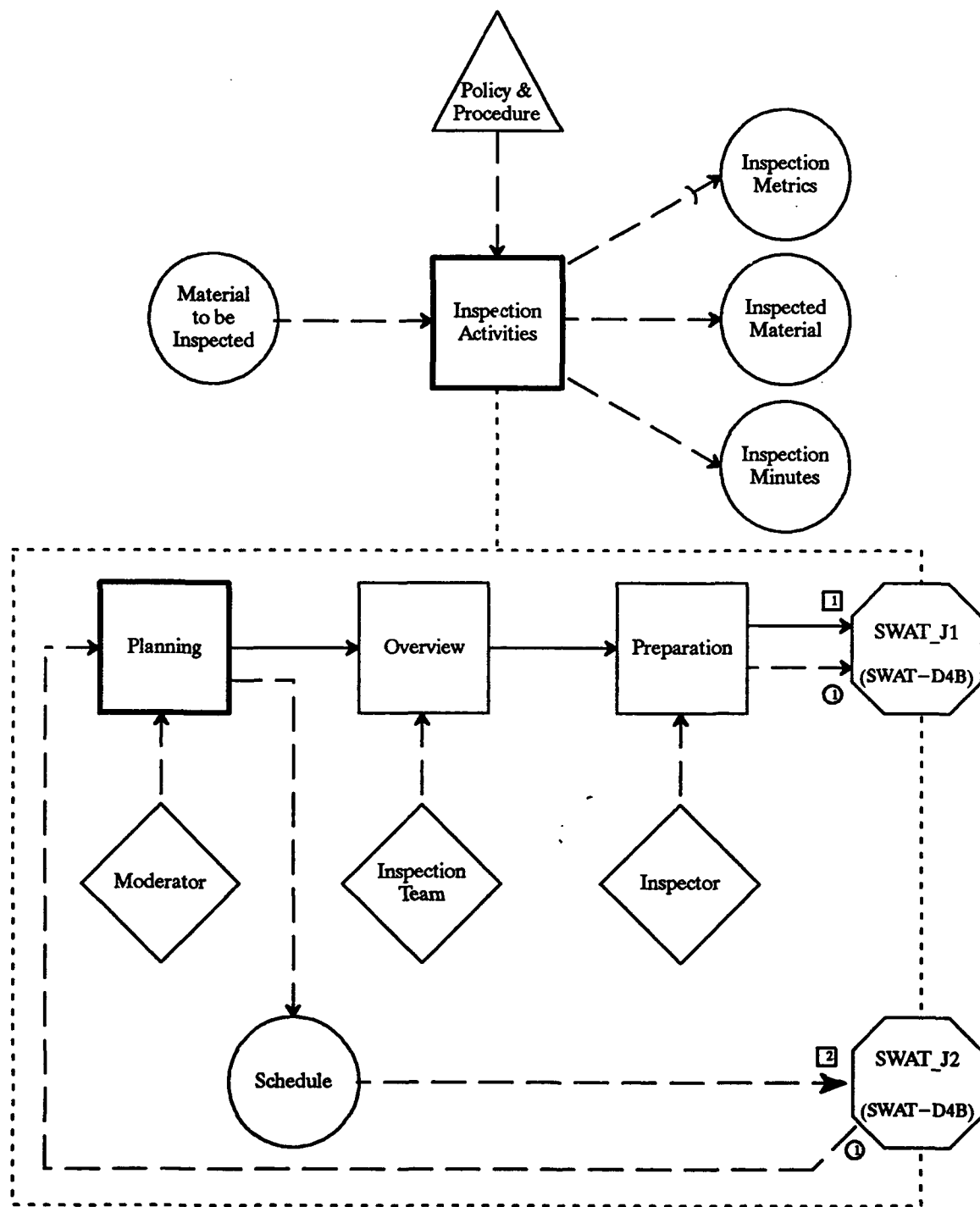
Figure 3-41. SWAT–D4A
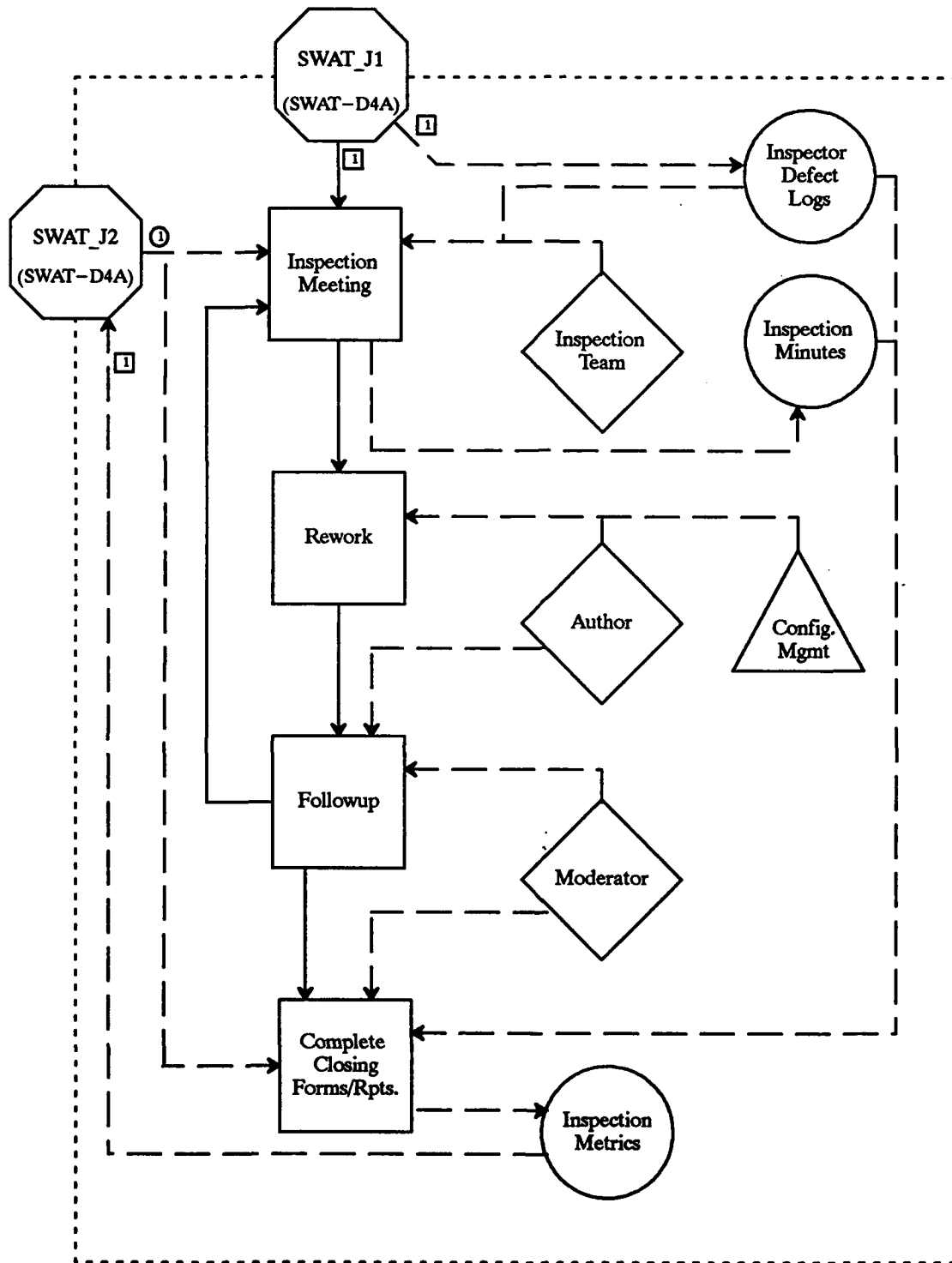
Figure 3-42. SWAT–D4B

this example, SWAT−D4A is shown as the root diagram, and SWAT−D4B shows detail information for which there was no room on SWAT−D4A. Note that the use of off-page connectors requires each connector to have its own identifier (such as SWAT_J1) and that all such connectors always exist in multiples (i.e., you would never have only one). For example, SWAT_J1 is shown on SWAT−D4A as a source connector, and it is also shown on SWAT_D4B as a destination connector. (Although not shown in this example, it is permissible to have multiple sources and/or multiple destinations, but this is discouraged as it tends to reduce, as opposed to improve, readability.) In all cases, an off-page connector must connect to someplace; therefore, you always have at least two diagrams using that connector's unique identifier.

To enhance readability, relations arriving at or departing from off-page connectors should show the symbol of their destination or source elements (respectively) and a count of those elements if a joined arc was used. On SWAT−D4A, the "preparation" event has a sequence relation that terminates at the off-page connector SWAT_J1. At the off-page connector, the relation is labelled with a small box indicating that on the remote diagram this relation terminates at an event. (To the knowledgeable reader, this simply confirms what is, in fact, a rule: sequence relations can only connect events with other events.) Additionally, "preparation" also has a reference relation that terminates at the off-page connector SWAT_J1. In this case, the reference relation indicates that on the remote diagram the relation refers to a throughput element (represented by the circle). At the off-page connector SWAT_J2 on diagram SWAT_D4A, a reference relation from "schedule" shows that it connects to two events (indicated by the small square with a "2" inside) on the remote diagram. Note that the off-page connector, in addition to its own unique identifier, also shows exactly which diagram the references refer to.

Figures 3-41 and 3-42 are a series of examples that show graphical depictions of process architectures that are consistent and compatible with the set of process templates described earlier in this section. Graphical renderings are an ideal way to quickly and efficiently model process elements and their relationships before proceeding with the more detailed work of filling in the templates. Additionally, even if templates were used prior to developing a graphical model, there is a direct mapping between the information captured on the templates and a graphical depiction of that information. Consequently, if you want to have a graphic-based and a text-based representation of your process, you can use the graphical notation prior to, concurrent with, or subsequent to your use of the process templates. If you will be doing process modeling without the assistance of automated tools, then the chief value of the graphical notation is likely to be in designing and altering initial architectures for a process model. After the graphical process architecture has stabilized, the templates can be populated with details and bound with each other.

4. TEMPLATE USAGE

The material in this section provides information and guidance on template usage. Section 4.1 examines using templates to support developing process guidebooks. Section 4.2 extends this discussion by examining template support of general process definition and modeling. Section 4.3 shifts perspective by focusing on the notation itself. This section introduces the concept of representative power and examines how the template graphical extensions increase representative power. Section 4.4 presents an example template usage scenario. Section 4.5 discusses tailoring and improving the usability of the templates, and Section 4.6 presents a simple example of using template-based process representations to support process improvement efforts. As indicated in Figure 4-1, while reading this section you may find it advantageous to regularly refer to the template-based example presented in the appendixes.
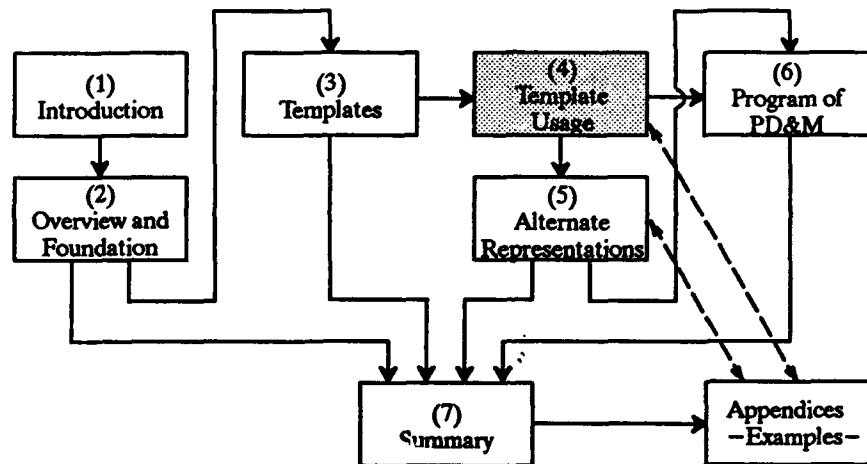
Figure 4-1. Guidebook Organization View 4

4.1 PROCESS GUIDEBOOKS

Software process guidebooks (SPG), such as the Consortium's Evolutionary Spiral Process or Reuse Adoption Process guidebooks, are designed to be shared by all software developers in an organization, division, or team. The SPG documents current and planned software development policies and procedures found to be most suitable for the organization as a whole, and it is typically written in natural language.

Improved communication is often achieved through greater use of any of the following: unified and consistent term definitions, section formats, decision tables; forms and templates; activity diagram types; and software process and product matrix collection tables. Predefined forms and table types, for example, can then be used to check the consistency and "usability" of the process. The SPG provides a mechanism for the software developer to reason, reference, discuss, and follow work descriptions. Practicing the process documented in the SPG also allows verification of process correctness and completeness.

SPGs can be of greater or lesser degrees of formality. A formalized SPG implies a software process model (SPM). An SPM is essential for mapping between the SPG and the representation of a process using a definition or modeling notation. An SPM serves as both a conceptual model for the software process developer and as an internal model for the complete and consistent implementation and modification of the SPG. The SPM should describe possible relations among different kinds of artifacts in the software process, possible activities and their sequence in a process, and possible partitioning of software products.

Two well-known process models are the Waterfall model and the Spiral model (the basis of the Consortium's Evolutionary Spiral Process). Each software organization may have its own SPM that reflects its software development experiences, procedures, policies, and conventions.

A process notation (PN) serves to represent the concepts, guidelines, and heuristics of the SPM to the software engineer. It should map to the software development environment so that a process represented using a PN can be interpreted, maintained, improved, and (ideally) used to support the development of process or project management plans.

The overall sequence of events that characterize an organization's ongoing efforts to develop and improve SPGs is as follows:

1. Analyze the existing process by building an abstract SPM of the process elements.

2. Extend the SPM by adding additional process details (throughputs, supports, etc.).

3. Analyze the model and use it as a foundation for writing an SPG.

4. Modify the SPM based on insights gained while developing the SPG.

5. Increase the formality of the SPM.

6. Update the SPG to reflect the latest SPM.

7. Use the SPG to develop software development project plans.

8. Analyze project metrics and evaluate possible areas of process improvement.

9. Use a variety of SPMs to evaluate alternative processes and potential process improvements.

10. Construct a composite SPM of approved changes.

11. Go to step 3.

These steps can result in defining an increasing set of reusable process assets that eventually can be used to support the implementation and execution of Environment-Based Process Management (EBPM) tools (see Figure 4-2).

Specific projects are instantiated from the model. The instantiation bounds the sequence of events and constrains that the throughputs and supports permit when forming a project plan. A project planner will then use this information in combination with information from the process execution environment to allocate resources and determine a feasible schedule. When a project plan has been instantiated from the process model, it is ready for enactment.
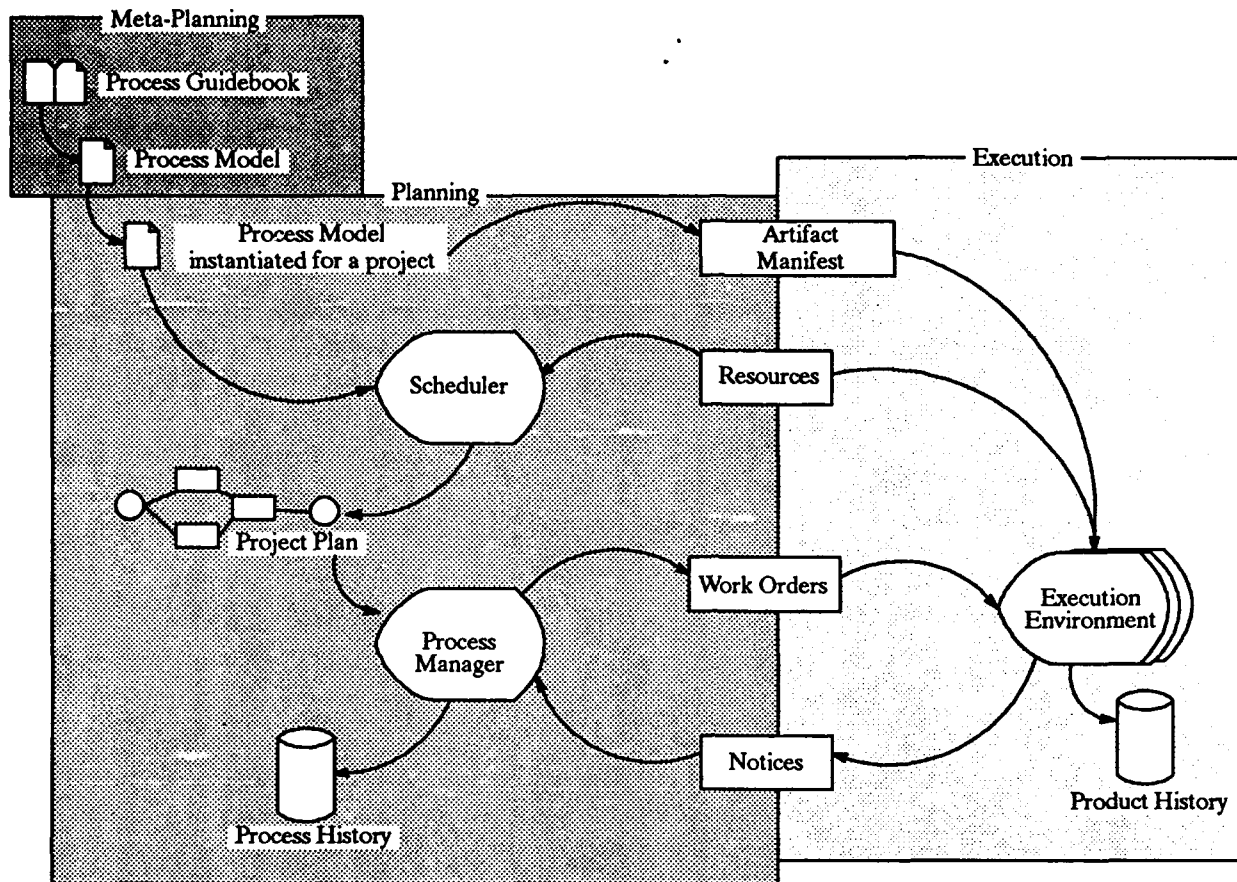
Figure 4-2. Environment Based Process Management Concept

During enactment, the project manager issues work orders into the execution environment in accordance with the schedule and monitors the project's execution. Data collected from the evolution of events and products form the basis of software process and product measurement. These metrics support process monitoring and eventually contribute to the insights that lead to ongoing process improvement. Continuous and measurable improvement is central to advancing software maturity, and such efforts can be facilitated by the combined use of process representations to construct SPMs which, in turn, serve as the foundation for SPGs.

It should be noted that one of the key advantages to template-based development of process guidebooks is the improved potential to automatically construct and print process guidebooks directly from an electronic inventory of templates. Since the templates are tolerant of free-form descriptive text, comments within, for example, entry conditions, exit conditions, description fields, etc., can all be reformatted and presented in guidebook format.

There are several advantages to this approach, especially if a versatile automated tool is used to maintain and update template-based process representations. First, this approach potentially removes the need to maintain both the guidebook and its underlying model. When guidebooks can be derived directly and automatically from the model (or templates), only the templates need to be maintained. Second, the templates can be automatically checked for consistency, completeness, conformance to structure rules, etc. In effect, the process model can be "compiled" and checked for "syntax" errors. Guidebooks would only be generated if the process model "compiled clean." Third, and possibly most important,

the process templates and associated graphical models can be used to train personnel in an organization's process. Training directly out of a multi-hundred page free-text process guidebook is often inhibited by a plethora of details and the resulting extremely low-level perspective. The templates impose a natural organization to this information and provide explicit layers of abstraction for improved conceptual overview. When combined with graphical depictions of template relationships, even complex processes can be more readily and efficiently taught.

4.2 PROCESS MODELS

This section continues to look at the use of process models but from an increasingly larger perspective. One method for analyzing the various potential uses of process models is to examine models from the perspective of support for process improvement. This is a function of how well the model facilitates planning deliberate process change.

Continuous process (re)definition and improvement can occur at a variety of levels within an organization. Each process level, or layer, is examined, and related models are developed through successive layers by repeatedly (re)defining the model to produce greater detail and refinement of information. The concept of refinement maps abstractions of the process model from one layer to the next. For example, the refinement of the process model that represents the corporate's process policy can be mapped into the process layer of the division. In the same way, the process model can be further refined, or instantiated, to produce the project model. Each of the refinements inherits the essential process definitions that support process benchmarks and metrics and allows the collection of process-specific data that can be used in a program of corporate process improvement.

4.2.1 PROCESS LAYERS

The process definition layers, presented in Figure 4-3, are adopted from the Boeing Software Technology for Adaptable Reliable Systems (STARS) project. The layers are:

- *Organizational Layer.* In this layer, the process model is embodied in textual descriptions collected into corporate policies and procedures.

- *Architectural Layer.* Here, the process model is described in a relatively rigorous graphical/textual notation such as ETVX and SADT.

- *Design Layer.* This is where detailed information is added to the architectural model, yielding detailed process designs and the instantiation of project-level process representations.

- *Enactment Layer.* This is the layer where one or more actual or virtual machines initiate and monitor process tasks.

- *Execution Layer.* These are the environment(s) where process tasks are performed.

As described above, process improvement can be facilitated by continuously refining the process model and forming more defined or specific layers as the model from one layer is instantiated to form the next layer. In Figure 4-3, on the side of the diagram, you see process description, process modeling notation, and EBPM; these are the three technologies that support the five layers of process development.

Central to each of these layers are the events' descriptions depicted at that layer of abstraction. Events are constrained and organized with respect to each other largely though the information relayed in
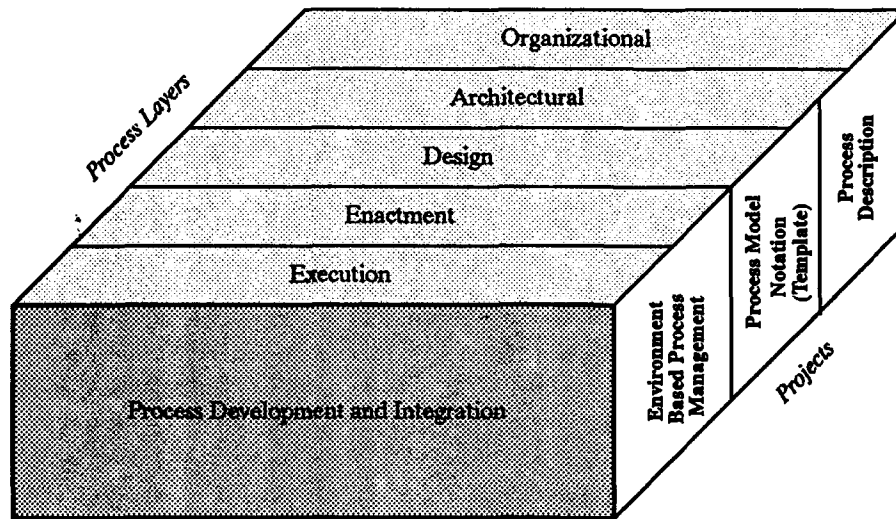
Figure 4-3. Process Definition Layers

the **Entry Criteria** and **Exit Criteria** fields. The information related at various levels of the process model will vary as a function of the information necessary to describe that level of construction. In all cases, the tradeoff is always between abstraction and details. Nevertheless, when constructing template-based models of the different layers, there are at least six kinds of information that can be placed in the **Entry Criteria** and **Exit Criteria** fields of the Event template. What you present and how you present it, depends upon the process layer you are working upon. The six kinds of information are:

- *Throughput Produced.* At higher levels, a text-based approach will communicate just the name of the throughput and its description. At progressively lower levels, the process engineer may apply state information by listing the throughput with its state. At progressively lower levels, the analyst may elect to express entry or exit criteria on throughputs using first-order predicate calculus.

- *Roles Required.* At higher levels, roles are described in isolation and are simply referenced by the event descriptions. At progressively lower levels, it becomes increasingly important to convey the authority associated with the roles and how that authority influences or constrains the flow of events.

- *Resources Required.* At the highest levels, resources are likely to be defined using abstract or collective terms. For example, a process may require the resources of a particular division but without clarifying which resources within that division are specifically necessary. With increasingly lower models, divisions' references, for example, would be broken out to department references that may, in turn, be broken out into group or team references, etc. At the enactment layer, it becomes necessary to bind, for example, the names of actual people with roles they are scheduled to perform.

- *Coordination Required.* In the entry(exit) criteria, coordination constraints provide a means to establish coordination relationship between events that otherwise have nothing in common. Coordination references will tend to be more common within higher level models and less common as details become available or expressible Ideally, at the enactment level coordination can be defined entirely in terms of throughputs, supports, or related events and their respective states.

- *Authorization Required.* Similar to coordination required, authorization required will tend to be more common on higher-level models. Authorization, especially as an external constraint, is a means to simplify the expression of relationships in highly abstract representations of a process. As the representation becomes less abstract, authorization can be replaced by, for example, references to conformance with published standards, successful completion of evaluations or inspections, etc.

- *State Information.* State information will likely become increasingly important as models are refined to deeper levels of detail. Events, throughputs, products, and constraints can all potentially have state information associated with them. For the enactment layer, and especially for the execution layer (if EBPM is being employed), state information can become a significant part of the model. Typically, however, state information implies detailed information; therefore, it is likely to become progressively less important when modeling higher-level processes.

The templates are designed to easily capture informal descriptions of the process but to do so in a way that automatically organizes that information to facilitate analysis, updates, and the construction of graphical models. The material in this subsection is intended to encourage the process engineer, when using the templates, to consider the advantages of deferring process details until lower level models are constructed. Simultaneously, the process engineer also needs to think forward and prepare for the next *cycle* of more detailed, or formal, process definition. As understanding of the process increases for the process engineer, he will migrate information in increasing detail between and among Throughput, Support, Constraint and Event templates. As the process depiction becomes more detailed, the process information will likely evolve into state-sensitive relationships between events, throughputs, etc. This entire effort is a process of constructing and maturing organization process definitions and, over time, using this information for organizational process improvement. Software engineers may start with undefined processes with unclear constraints; but through the systematic application of process definition and modeling, software engineers can evolve into an environment constrained by known, defined, and measured processes.

4.3 REPRESENTATIVE POWER

Regardless of the notation used for defining or modeling a process, or the level of the process being captured, each notation (and supporting methodology) can be viewed from the perspective of its representative power. The representative power of a process description reflects the factors of granularity, practicality, redundancy, modularity and information hiding. These factors are not mutually exclusive and, to greater or lesser degrees, each one overlaps with all of the others. Nevertheless, these are key factors that a process engineer needs to consider when matching the level of a process model with an appropriate process modeling strategy.

4.3.1 GRANULARITY

Granularity is the degree of detail depicted in a process description. The amount of effort necessary to support process modeling grows rapidly with increased granularity. If not automated, growing cross reference information increases the expense of process management and coordination. Granularity should be gradually increased with increasing understanding of the process. Note that it is not necessary to maintain the same level of granularity for the whole process model. For example, the basic repeated routines may be defined in more detail than others: i.e., important and well understood process elements can be defined with increased granularity.

For templates, granularity can be increased in two ways: by refining the template class structure to describe more specialized process elements or by using the parent-child (inclusion) relationships of the existing templates to create hierarchies of finer grained objects. Typically, increased granularity is a function of iteratively increasing detail.

The capability to represent the decomposition hierarchy of process elements is an important characteristic of event definition templates, e.g., the parent-child relationships in the proposed set of templates. Both the management template and the production template is expected to support a tree of subactivities through its parent-child relationships—the deeper the tree becomes the higher the granularity. A similar situation exists establishing decomposition hierarchies within each and all of the other templates. This ability supports both a top-down, or a bottom up process modeling strategy, depending on whether you intend to extend your models through adding greater detail, or through distilling progressively more abstract characteristics, respectively.

4.3.2 PRACTICALITY

Practicality is the sum of the degrees of ease of use, ease of understanding, ease of implementation, and applicability to the existing process. The proposed template set represents a level of granularity deeper than that currently practiced by most of the software industry. The practicality of the scheme depends on the ability of the software engineers to follow the process defined to the level of detail in the templates. This ability relates to the size of the organization, the technical level of the organization, the degree of formality applied, and the amount of automation available to manage the details. These considerations encourage simplicity. On the other hand, perceived discrepancies between the real process and the model discourages its use. This leads to complexity which conflicts with ease of use and implementation. Practicality requires the process designer to strike a balance between complexity and understandability.

Initially, keep the number of fields in the templates to a minimum. Avoid purely abstract process data and concentrate on that information that leads directly to process analysis, guidance, and support. For example, include fields that can be used to generate activity checklists. Where possible, use automation to manage details and to support analysis of template or process interrelationships.

The ability to represent or model an existing process depends upon existing process descriptions, how dynamic the process is, and whether automated tools exist to support the development and analysis of process representations. Process models are dynamic and change enormously before they become stabilized in a software development environment. One risk-mitigation-based approach is to use a spiral process that cyclically performs process analysis, definition, modeling, usage, and assessment which then transitions back to process analysis and the beginning of the next cycle. A process model needs to be used, evaluated, and refined before becoming a fixed process asset. You might begin with a subset of the templates and then expand the number of templates and the level of detail as your understanding increases. Similarly, you can begin with a high level definition of the existing process, then cycle through process refinement, making liberal use of the graphical notations to rapidly build and refine alternative views of the process architecture and its principle elements. Using this approach, process definition and modeling can become an important step in an overall program of process improvement.

4.3.3 REDUNDANCY

Redundancy is the amount of overlapping information in one process description. In theory, it is possible to design the templates so that each piece exists in one and only one place in the template and that

cross-references are used to tie them together. In practice, any paper-intensive approach becomes progressively more difficult to use as the number of pages increases. Large volumes of paper typically require the reader to constantly flip between pages in order to understand the relationships and develop high-level understanding. This problem can be mitigated through the use of, for example, "hyper-text" based tools. A template browser can be easily implemented using hyper-text links to navigate the cross-references. Additionally, in cases where a guidebook is automatically generated from process definition templates, it may be desirable to deliberately allow redundant information within the guidebook so as to improve guidebook usability for the reader.

Another redundancy issue is consistency. As soon as redundant information is introduced, we create the possibility that the information will not be consistent. For example, consider a set of parent-child relationships where a child is not included in the list of its parent's children. Increasing the redundancy in a process design may contribute to its readability, but it increases the overhead necessary to validate the design's consistency. Again, automation can be useful by supporting analysis which identifies inconsistencies in a process definition.

4.3.4 MODULARITY AND INFORMATION HIDING

A process provides the ability to represent a process fragment (or activity) as a self-contained unit similar to a procedure in an Ada package. The engineer can treat the description as a reusable process asset, i.e., he can compose existing process fragments to form a new process. Just as an abstract datatype is one way to encapsulate a generic algorithm, a process fragment can encapsulate an activity.

A module of a process is defined in terms of a set of activities or working stages for software development. The process description provides the opportunity to group a set of activities into a single super-activity or to decompose an activity into a set of subactivities. The process modeler needs to provide the description of interfaces between this activity and its subactivities and the interface between two activities which support the modularity of a process description.

The proposed templates use the event meta-class as one encapsulating mechanism. Generally, the "interface" of an event is a function of its entry criteria and its exit criteria. If two process fragments have identical entry and exit criteria, then they at least have the potential to be substituted for each other. Another advantage to modularity and applicable to the existing template design is that modularity offers a consistent means (via interfaces) to automatically search, retrieve, and match related templates from a process fragment library.

4.4 TEMPLATE USAGE SCENARIO

The preceding material has presented a discussion on a variety of principles important to process representation. However, understanding is based upon both principles and practices. Therefore, this section gives an example of how you might proceed with practicing process definition and modeling. This is a recommended approach, but by no means is this the only approach. It can serve as a good point of departure for an organization that does not have a history of or strong preference for using a different technique.

There are a variety of ways that the templates can be used to perform process definition and modeling. It is highly recommended that regardless of the approach selected, there should be some element of cyclic or top-down refinement. A relatively low-risk, cost-effective approach to building process

models is to initially build a complete—if high-level and simplified—graphical representation of the process. When this has stabilized, the templates can be used to capture increasingly detailed information. This enables a greater degree of detail to be gradually introduced and a more progressively explicit and complex model to be constructed. The overall model is then cyclically extended to include progressively more information in both the graphical and template-based representations. The example scenario described below follows this approach.

4.4.1 ACTIVITY ONE: DEFINE EVENT RELATIONSHIPS

The organization and relationships between the events that comprise a process is critical to defining and modeling that process. Although you can build process representations that do not include roles, resources, or products, it is essential that all representations capture the ordering (or partial ordering) of events. The tasks within this first activity define a suggested approach for capturing events and their interrelationships.

4.4.1.1 Task 1: Build an Indented List of Events

The first step of the first cycle is to simply build an indented list of events. The purpose of the indentation is to reflect the organization of higher-level events and their subevents. For each item on this list, you provide a unique identifier and a brief explanation of the event's purpose and description.

At this stage, it is sufficient to attempt to define only the first two or three layers of the domain of interest. The first, or highest, layer defines the major processes occurring within the domain. Processes are distinguished from activities in the sense that a process is a perpetual or ongoing event whereas activities have definite start and stop times. For example, in a particular domain there is likely to be a management process, an engineering process, and a quality assurance process.

The second layer defines the primary activities that comprise each process. Activities similar to the management process include:

- Perform cost/benefit analysis.

- Evaluate alternative solutions.

- Construct PERT chart.

At the third layer, list the primary tasks that comprise each activity. For example, the "evaluate alternative solutions" activity might decompose into:

- "Evaluate commercial solutions."

- "Evaluate alternative in-house solutions."

- "Evaluate risk of doing nothing."

4.4.1.2 Task 2: Build a High-Level Graphical Model of Events

Using the indented list as a reference, construct a high-level model based upon the graphical notation discussed in Section 3.3. This model will contain one type of object (events); and although all four types

of relations might be needed, the most likely relations will be inclusion (to show decomposition of events) and sequence (to show ordering of events). Generally, the indentation levels on this list should capture the majority of the parent/child or inclusion relationships.

4.4.1.3 Task 3: Establish a Template for Each Event on the Indented List

For each item on the indented list, develop a useful, consistent, and ideally intuitive naming convention for constructing unique identifiers for the listed item. Transfer this and other basic information to an appropriate template (i.e., a Management or Production template). Fill in any other fields with the information already known (e.g., the **Version #** and **Date** fields, the **Purpose** field, and **Comment** field).

4.4.1.4 Task 4: Identify Event Abstractions

Each template has fields for noting relative event abstractions by using the Child Events and Parent Event(s) fields. On both managerial and production templates, note the parent and child events. The graphical model developed in Task 2 of this activity should provide direct insights, though tracing inclusion relations, into the parent and child relationships needed by the templates. Detail this two-way relationship for all events.

4.4.1.5 Task 5: For Each Nontask Event, List All Additional States

Events can have the default state of in_progress expanded into an enumerated set. Consider each event and evaluate whether it is necessary to expand the number of states used to describe that event. If so, provide a state name and a brief description. (*Note:* The list of possible states can be adjusted and refined at any time; hence, it may be more efficient to initially use just the default states.)

4.4.1.6 Task 6: Describe Internal Processing

An event's internal processing or flow is the principle description of the work that characterizes it. This description is done in terms of logical flow (using if, then, and, or, not, while, repeat, until and similar logical constructs) of activities, tasks, and steps. Note that lowest-level work (or "leaf" events) can be likewise described, except that terminal events, by definition, do not contain subordinate events (i.e., these are atomic events).

A simple example of a logically described work-flow is:

```
Step 1: do ...
Step 2: commence Activity-A1 and commence Activity-A2
Step 3: do...
Step 4: if Activity-A1::In_Testing then
        Step 4a1: do...
        Step 4a2: commence Activity-A3
    else
        Step 4b1: do...
Step 5: when Activity-A2::Completed
        Step 5a: commence Activity-9
Step 6: if Activity-A1::Suspended then
        Step 6a: goto Step 3
Step 7: do...
```

It is not necessary to precede each line with a "tag" or step number. However, tags greatly facilitate discussion and are convenient for logically describing certain forms of process flow. (*Note:* The "goto" statement is as undesirable here as it is in conventional logic programming.) If there is sufficient familiarity the conventions of logic-based flow charts, a more formal and more graphical depiction can be made of the events' internal processing.

4.4.1.7 Task 7: Describe Event State Transitions

Describe any event state transitions as a reflection of the internal process flow just described in the preceding step. One approach for defining state transitions is to use ":=" as a state assignment operator, then intersperse such assignments within the process flow description.

Alternatively, a separate list can be built explicitly describing at which step in the work flow the state transition occurs. A convention should be established and used throughout. In this guidebook, the notation "::" is used to represent a particular item in a given state.

Examples of using this convention include:

```
Step 1: Do...
Step 2: Set state of Some_Event to Some_State
Step 3: Do...
Step 4: If Some_Other_Event::Some_Other_Event_State then
    Step 4a1...
else
    Step 4b1...
...
```

4.4.1.8 Task 8: Define Entry and Exit Criteria

For each event (from top to bottom) define the entry and exit criteria in terms of event states. Entry conditions are defined in terms of the state of external events, and exit conditions are defined in terms of the state of internal events (on rare occasion, the state of external events).

For example:

Entry Conditions

(Activity-A::Complete and Activity-B::Complete) or
(Activity-C::Nearly_Done)

Exit Conditions

(Step-12::Complete and Activity D3::Complete)

As the entry and exit criteria are defined, assure that they are consistent with the implications of the graphical model built in Task 2. If need be, update or alter the graphical model as insights are gained from ongoing analysis and the effort of filling in the templates.

4.4.2 ACTIVITY TWO: DEFINE EVENT THROUGHPUTS

Having a first pass at an event-based model, the next activity is to expand both the graphical and template-based information describing that model. One of the primary purposes of processes is the

creation or modification of a product or service that justifies the need for the process. Therefore, the second activity in this suggested approach is to expand the template-based definition to include Throughput templates. Throughputs, abstractly speaking, are anything (tangible or intangible) that evolve through one or more events. Though typically throughputs are products, they can also be used to represent research.

4.4.2.1 Task 1: Build an Indented List of Throughputs

As with the prior cycle, the first step is to build an indented list. However, this time the list will consist of throughputs. To construct it, note the significant products or research evolving from work performed in the domain being modeled. Also attempt to develop a useful convention for developing unique identifiers.

An example of an indented product list is:

```
Software_Product
    Software_Source
    Software_Binary
    Technical Documentation
        Administrator Guide
        User Guide
    etc.
```

An example of an indented listing of research throughput is:

```
Investigate Machine Performance
    Investigate Bus Throughput
        Test Internal Bus
        Test I/O Bus
    Investigate CPU cycle time
    Investigate Disk Access
        Test Hard Disk
        Test Floppy Disk
    etc.
```

4.4.2.2 Task 2: Extend the Graphical Event Model to Include Throughputs

Using the throughput indented list as a reference, extend the event model using the graphical notation discussed in Section 3.3. This model will now contain two types of objects (events and throughputs). Although all four types of relations might be needed, the most likely relations will be inclusion (to show decomposition of events), sequence (to show ordering of events), and reference (to bind throughputs and events to each other).

4.4.2.3 Task 3: Establish a Template for Each Throughput

For each item shown on the indented list (and added to the graphical model), transfer the basic information to an appropriate template (i.e., either a product template or a research template). Fill in any other fields that information is already known (e.g., the **Version #** and **Date** fields, **Purpose** fields, and **Comments** fields).

4.4.2.4 Task 4: Identify Throughput Abstractions

There are spaces on the templates for noting parent/child relations between products and subproducts and research and subresearch. Using the graphical model as guidance (specifically, throughput inclusion relations) use the **Composed Of** and **Part Of** fields to document these relationships to whatever depth is necessary. (*Note:* These relationships will also closely follow the indentation levels in the throughput indented list.)

4.4.2.5 Task 5: Identify Any Additional Throughput Item-Specific States

Throughput items come with the following default set of states:

- Unauthorized

- Authorized

- In Progress

- In Rework

- Disabled

- Suspended

- Cancelled

- Completed

For each Throughput, Product, or Research template, add any additional states that are necessary or useful for defining relationships between throughput and events.

4.4.2.6 Task 6: For Each Event, List All Throughputs `

For each of the Event templates, list in the **Throughputs** column all throughputs generated or manipulated by that event. Processes will typically reference higher level throughputs, and tasks will reference lower level throughputs. If meaningful, also indicate whether a given throughput, within a given event, is required or optional. (The graphical model can help you confirm you've documented all the relationships; the graphical model should also be updated as appropriate.)

4.4.2.7 Task 7: For Each Throughput, List All Contributing Events

Once all the Event templates have been updated to reference applicable Throughputs, each of the throughput templates need to be updated to reference the applicable events (i.e., Evolves From Events). In this way a cross-reference exists to facilitate verifying the integrity of the process definition. It may be useful to specify, by event, whether the throughput is required or optional (this is especially true if this convention was used in task 5 of this activity). The graphical model can help you confirm that you have documented all the relationships; update the graphical model as appropriate.

4.4.2.8 Task 8: Define Throughput State Transitions

If you desire increased detail, then for each throughput, define throughput state transition as a function of Event:Step#:new_state. That is, for each throughput, define how the throughput advances through various throughput states as a function of the events that govern the evolution of that throughput.

4.4.2.9 Task 9: Update Internal Event Flow to Include Throughput::State References

Now that throughputs and their corresponding states have been explicitly defined, the internal work flow within an event can be upgraded to include references to the states of throughputs. This can be especially useful for modeling inspection or quality assurance phases of a particular event.

4.4.2.10 Task 10: Update Entry and Exit Criteria to Include Throughput::State References

Similar to the previous step, for all event templates, the entry and exit criteria can be upgraded to include references to the states of throughputs that are either expected by or released by a particular event.

4.4.3 ACTIVITY THREE: DEFINE EVENT SUPPORTS

This step defines and references the support roles and resources needed by a particular work domain. As detailed in the following tasks, this sequence of steps proceeds in a similar fashion to those in the previous cycles.

4.4.3.1 Task 1: Build an Indented List of Supports

This indented list of supports should detail both the roles and resources needed to support the work domain. The term "roles" is used in the broadest sense; therefore, not only do individuals perform roles but so do teams, entire divisions, etc.

4.4.3.2 Task 2: Extend the Graphical Model to Include Supports

Using the support indented list as a reference, extend the graphical model using the notation discussed in Section 3.3. This model will now contain three types of objects: events, throughputs, and supports. It will likely contain all four types of relations: inclusion relations (to show decomposition of events; throughputs, and supports); sequence (to show ordering of events); specialization (to show specialized instances of classes of process elements); and reference (to bind events and throughputs, and to bind events and supports).

4.4.3.3 Task 3: Establish a Template for Each Support on the Indented List

As in the prior cycles, when establishing the template, attempt to fill in any information already known (especially those fields in common with the Foundation template).

4.4.3.4 Task 4: Identify Support Abstractions

These abstractions are two-way relationships. They can be noted under the **Composed Of/Part Of** fields of the Role and Resource templates.

An example of levels of abstractions within roles is:

    Programming Team
        Team Leader
        Programmer (3-6)
        Technical Support (1-2)

Line Manager
Administrative Manager
V&V Department
  Statistical Group
    Statistical Group Manager
    Statistical Group Engineer
  Dynamic Test Group

   ...

  ...

 ...

Note that the indentation on the support indented list and the graphical model support inclusion relations can both be used for initial guidance on documenting composed of/part of relationships on the templates.

4.4.3.5 Task 5: Identify Any Additional Support States

The default state set for support items is:

- Available_Exclusively

- Available_Shared

- Not_Available

- Disabled

- Suspended

For each Support, Role, or Resource template, add any additional states that are necessary or useful for defining relationships between supports and events.

4.4.3.6 Task 6: For Each Event, List All Needed Supports

If appropriate and desirable, also note whether a support is needed exclusively by an event, whether it can be shared, and whether or not that support can be considered optional. For example, secretarial support or a delivery van might be needed in a shared capacity, whereas a particular hardware device or printing press might be needed exclusively.

4.4.3.7 Task 7: For Each Support, List All Supported Events

This is the cross-referencing step. It details the companion relationship between supports and events (whereas the prior step defined the relationship between events and supports).

4.4.3.8 Task 8: Update Internal Processing Fields to Include Support::State References

With supports and their corresponding states explicitly defined, the internal processing within an event is upgraded to include references to the states of the roles and resources applicable to the event.

4.4.3.9 Task 9: Update Entry and Exit Conditions to Include Support::State References

This step is similar to the previous step. For all event templates, the entry and exit criteria are upgraded to include references to the states of supports needed by an event.

4.4.4 ACTIVITY FOUR: DEFINE EXTERNAL EVENT CONSTRAINTS

Even establishing just a few External Event templates has considerable potential to simplify the organization and representation of a process model. Often, especially in management intensive environments (as opposed to function intensive), the governing flow of control is driven by management directives. In other words, "the time to start something is when management says it is time to start." Such factors are readily modeled as external (and sometimes internal) constraints.

4.4.4.1 Task 1: Build an Indented List of External Constraints

This is fundamentally the same effort as in the other cycles, except that the list being built focuses on itemizing external constraints.

The following are examples of external constraints:

- Policies

- Procedures

- Standards

- Guidelines

    - Management guidelines

    - Technical guidelines

    - Support guidelines

- Management plans/budgets

- Directives (from "outside" people; typically verbal)

4.4.4.2 Task 2: Extend the Graphical Model to Include External Constraints

Using the external constraint indented list as a reference, extend the graphical model using the notation discussed in Section 3.3. This model will now contain all four types of objects: events, throughputs, supports, and constraints. It will likely also contain all four types of relations: inclusion relations (to show decomposition of events, throughputs, supports, and constraints); sequence (to show ordering of events); specialization (to show specialized instances of classes of process elements); and reference (to bind events and throughputs, events and supports, events to constraints, constraints to throughputs, and constraints to supports).

4.4.4.3 Task 3: Establish a Template for Each External Constraint on the Indented List

As in prior cycles, when establishing the template, attempt to fill in any information already known.

4.4.4.4 Task 4: Identify External Constraint Abstractions

Use the **Special Form Of** and **General Form Of** fields to note the parent/child relations that exist within the external constraints. Recall from prior material that these may include "whole/part" relations or "generalized/specialized" relations. Again, the graphical model can greatly facilitate determining the relations that need to be documented on the templates.

4.4.4.5 Task 5: Define Additional States for Each External Constraint

As in the other areas, it may occasionally be necessary to expand upon the default set of external constraint states for some of the external constraints.

The following list includes existing external constraint states:

- Compliance_Under_Evaluation

- Compliance_Achieved

- Compliance_Failure

- Compliance_Waived

4.4.4.6 Task 6: For Each Event, Note All Applicable External Constraints

These can be listed in the external constraint column in the Event templates.

4.4.4.7 Task 7: For Each External Constraint, Note All Applicable Events

As with the other activities, this task establishes a cross-reference between templates to support verifying template integrity.

4.4.4.8 Task 8: For All Events, Update Internal Processing *to* Include External_Constraint::State References

With external constraints and their corresponding explicitly defined states, the internal processing within an event can be upgraded to include state-sensitive references to applicable external constraints.

4.4.4.9 Task 9: For All Events, Update Entry and Exit Conditions to Include External_Constraint::State References

Similar to the previous step, for all event templates the entry and exit criteria can be upgraded to include references to the states of external constraints that are governing an event.

4.4.5 ACTIVITY FIVE: DEFINE INTERNAL EVENT CONSTRAINTS (PERMISSION CLASSES)

In this example, the last major components introduced into the process definition are the internal constraints. This effort is separated from the external constraint definition effort because internal constraints must be explicitly bound to a role before they can be introduced into the event descriptions.

4.4.5.1 Task 1: Build an Indented List of Internal Permission Constraints

The following internal (permission) constraints should be included in virtually any list of constraints:

- Commencement permission

- Suspension permission

- Recommencement permission

- Cancellation permission

- Resurrection permission

- Completion permission

- Override permission

- Executive permission

4.4.5.2 Task 2: Update the Graphical Model With Internal Constraints

Generally, since internal constraints are always manifested through roles, there is little need to graphically depict them. However, if you desire a detailed graphical model, internal constraints are graphically depicted using the same symbol as external constraints (a triangle). They are bound to events, supports, or throughputs using reference relations, and they are bound to other internal constraints using the inclusion relation.

4.4.5.3 Task 3: Establish a Template for Each Permission Class on the Indented List

When establishing the template, attempt to fill in any information already known.

4.4.5.4 Task 4: Identify Permission Class Abstractions

These abstractions are two-way relationships. They are found in the **General Form Of** and **Special Form Of** fields of the Internal Constraint template. They can be derived from examining internal constraint inclusion relations on the graphical model.

4.4.5.5 Task 5: For Each Role, Note All Applicable Permission Classes

On each Role template, there is a field for listing all applicable internal constraints exercised by that role. Use this field to indicate whether the role can grant permission for cancellation state changes, override state changes, etc.

4.4.5.6 Task 6: For Each Internal Constraint Class, List Associated Roles

While the prior step documents all the internal constraints exercised by a role, this step documents all the roles exercised in a given internal constraint. As in prior cycles, these should always occur as paired relationships.

4.4.5.7 Task 7: For Each Event, Update Internal Processing to Reference Role::Permission

With explicitly defined internal constraints, the internal processing within an event can be upgraded to include references to applicable roles exercising authority as described within the internal constraint templates.

4.4.5.8 Task 8: For Each Event, Update Entry and Exit Conditions to Reference Role::Permission::State

For all Event templates, the entry and exit criteria is upgraded to include references to roles and internal constraints relative to that event.

4.4.6 ACTIVITY SIX: SIMPLIFY/CLARIFY TEMPLATE CONTENTS

At this stage in the evolution of a process' template-based definition, the templates have been through successive modification cycles. Before repeating the above activities with the goal of adding several more layers of detail (i.e., expanding tasks into activities, subactivities, etc.), it would be useful to improve and clarify the existing process definition.

This is the subject of Section 4.5.

4.4.7 ACTIVITY SEVEN: GENERATE ALTERNATIVE/OPTIMIZED PROCESS MODEL

After several passes through the above series of activities, a detailed and explicit model of the processes within a given domain emerges. An ideal subsequent effort is to use that template-based process definition as a means of implementing process improvement. This will be discussed in detail in Section 4.6.

4.5 IMPROVING TEMPLATE USABILITY

There are a variety of ways to improve the usability of the templates. One technique is to improve the way you present data using the existing templates. Another technique, especially useful if you have specific or uncommon requirements, is to modify the templates and tailor them to accomplish site-specific objectives. Both techniques are discussed below.

4.5.1 IMPROVING DATA PRESENTATION

In addition to the virtually unlimited variety of approaches you can take for using the templates, there are likewise a great variety of ways for representing information within the templates. However, readability is a critical goal; therefore, it is important to examine methods in which the presentation of information within the templates can be simplified and made clearer.

One simple technique is to determine a naming convention for building unique identifiers that facilitate ease of understanding. Consider the following example. For any given process model, select a unique, brief identifier to represent that system. (The inspection process model example used in several sections of this guidebook uses the prefix SWAT.) For any template used within that system, append a two character code where the first character represents the meta-class template, and the second character represents the class template. In only one case does this yield duplication and that is on the Role template (both role and resource begin with "R"). As shown below, the character "P" can be used (think of the word "people") to represent the Role template. This convention yields the following suffixes:

EM	Event	Management
EP	Event	Production
TP	Throughput	Product
TR	Throughput	Research
SP	Support	Role ("People")
SR	Support	Resource
CE	Constraint	External
CI	Constraint	Internal

Therefore, SWAT_TP_UI-CODE might represent the user-interface code (UI-CODE) that is a throughput product (TP) of the Inspection Process Model (SWAT).

Using a consistent pattern to construct meaningful unique identifiers is one of the key techniques for improving the representation of data within the templates. The templates are intentionally constructed with fields for intertemplate binding, and these fields always contain unique identifiers. Properly constructed identifiers greatly improve the reader's ability to understand the nature and purpose of these numerous intertemplate relations. Unique identifiers of children should indicate their lineage of parents and ancestors using a "pathnames" approach common within operating systems. Continuing with the above example, suppose the product template defining UI-CODE indicates that it is "composed of" MENU-SYS and several other modules. In turn, MENU_SYS indicates it is composed of "PULL-DOWN" and "POP-UP." The constructed unique name for the pull-down menus might be SWAT_TP_UI-CODE_MENU-SYS_PULL-DOWN. (*Note:* In this example, the use of the underscore character indicates separation of levels, and concatenated terms within a level [such as PULL-DOWN] are always separated with a hyphen.) This not only easily distinguishes what a component is, but how many parent generations it has and who those ancestors are.

The process analyst can further assist the reader by attempting to detect and reduce redundancy, especially between levels of abstraction. For example, if an external constraint is already modeled at a higher level, it is not necessary to repeat the representation of that constraint on each of the activities subordinate to that level. If a guidebook is subject to an external constraint represented by a style guide, it is also true that each chapter of that guidebook is likewise subject to that same external constraint. If each chapter is shown, due to the breakdown of events, as a separate product, the model might explicitly show that each of these chapters is itself also constrained by the style guide. Although this is certainly correct, it leads to cluttered templates. This clutter can be reduced by verifying that the external constraint is applied against the "parent" product then remove the (essentially redundant) reference to the external constraint from each of the subordinate templates. Note, however, that the integrity of cross-reference fields must be maintained at all times. If references to the external constraint are removed from chapter-level products, it must be verified that the External Constraint template does not have chapter-level cross-references, but instead it only references the product at the level of the overall guidebook.

It cannot be overstressed that simplicity is crucial. In the words of Einstein, "Make everything as simple as possible, but no simpler." For each template, the information recorded on that template needs to be viewed from the substance of information it conveys. At some point, if there is little value to some additional information, that information must be considered as potentially obscure or confused. A crucial step to optimizing any representation is not only to determine what is missing (and should be added) but also to discern what is excessive (and hence, should be removed). "Pruning" representations is as crucial to improving their readability as are growth and extension.

Recall that a model is a representation of reality that captures interesting characteristics and abstracts out all noninteresting characteristics. This seems especially applicable to process models. From a clarity perspective, the question is not simply whether a given template captures something accurately or not. It is a question of whether it is both accurate and interesting with respect to the objectives of the definition and modeling effort. Avoiding and, if need be, removing uninteresting information from the templates is crucial to improving the overall clarity of the model the templates represent.

4.5.2 IMPROVING AND TAILORING THE TEMPLATES

Occasionally, you will be able to achieve significant increases in the readability and usability of template-based process models by not only using the techniques discussed above, but also by altering, adding, or otherwise modifying the basic set of templates themselves. The arrangement of the templates into a hierarchy of a root template, meta-class templates, and class templates is deliberately intended to simplify and support the tailoring process. There is such a vast diversity of processes and a correspondingly vast diversity of objectives when constructing process models, that allowing tailoring of the templates to accommodate site-specific goals is a key objective for this guidebook.

The templates' current version attempts to keep information and relationships at an absolute minimum, thereby leaving the greatest flexibility for altering the templates to meet site-specific needs. It should be stressed that the templates are completely usable "as is." In certain environments, however, especially after several template-based process models have been constructed, it may become clear that you can make advantageous extensions or alterations to the templates. This directly translates into improved ease of use, greater clarity, greater applicability, and even increased formality.

For example, suppose that the processes occurring within a particular site all follow the ETVX paradigm (discussed in greater detail in Section 5). This paradigm essentially requires that all events have a validation stage (the 'V' in the paradigm acronym). In their current form, the templates do not have an explicit type of event dedicated to validation work. How might the process analyst respond to this?

Initially, the process analyst could elect to use the templates as they currently are and model a process "tree" (i.e., a process with hierarchically supporting activities and tasks) dedicated to validation. He can then build process models for all other work or processes occurring in the domain. To verify that the regular work processes are adhering to the ETVX paradigm, the process analyst reviews the templates and verifies that each event does, in fact, make a "call" to one of the validation event templates to perform a validation effort. Tasks that do not have such a call (at least according to the model) violate the ETVX paradigm.

Alternatively, the process analyst can elect to define a new meta-class template and call it, for example, "Measurement." Underneath this meta-class, he can construct one or more related class templates such as "Product Quality Assurance" and "Project Efficiency." Fields within the quality assurance template capture cross-reference information regarding throughputs subjected to quality assurance, and they cross-reference to events indicating where the quality assurance measurements should occur. Similarly, the efficiency template can be tied to throughputs and events and, depending on site-specific objectives, even to resources. The efficiency template formally represents artifacts that support monitoring process progress and, subsequently, project histories. From this perspective, the heuristic for assessing whether a process model achieves ETVX compliance is to investigate and verify that all events have a cross-reference to a quality assurance class template under the measurement meta-class. (The implication being that this represents a throughput being measured, evaluated, and assessed from the perspective of achieving predefined quality goals.)

This alternative approach of tailoring the templates may be more labor-intensive at first than the initial approach, but it has the distinct advantage of clearly representing key characteristics (or their absence) of various renditions of process models. In this case, by adding a new meta-class and a couple of new class templates, process analysts can easily and explicitly capture verification- and validation-specific activities and artifacts that they consider important to their ETVX-oriented environment.

This is just one example, but the principle remains the same for other types of tailoring. If a site wanted to tailor the templates toward DOD-STD-2167A-oriented process models, they benefit by expanding on the throughput meta-class and adding class templates to explicitly represent DOD-STD-2167A artifacts. Conversely, if a given site follows processes intended to be compliant with IEEE 1074 (Standard for Software Life Cycle Processes), they can elect to expand upon the event meta-class. The existence of and order of events is one of the principle characteristics of IEEE 1074. Furthermore, since IEEE 1074 explicitly references the need for seven processes, these processes can all be given a dedicated template format tailored toward collecting and representing the specific characteristics of those individual process types.

When tailoring the templates, you will also need to update your documentation describing the usage of the templates. The recommended approach is to pattern such documentation on the format used in Section 3.2. Specifically, if the tailoring effort results in a new field that needs to be added to all templates, that field should be added to the Foundation template. The field's purpose and use should be discussed and included with the rest of the discussions on Foundation template fields. If you need to add a new field to all throughputs, you should show it on the throughput meta-template. A discussion of the field's purpose and use should be included with the discussion of Throughput template fields.

You should note that there is no particular reason to constrain the tailoring effort to the current three level structure of root, meta-class, and class templates. In certain environments, it can be useful to convert to a four level structure containing root, meta-class, class, and type templates. Another environment may find that another layer, "subtypes," should be added to the four-level approach.

A word of warning: all else being equal, increased complexity virtually always means increased risk. Regrettably, it is far easier to justify adding new fields, new templates, or even new levels than it is to justify removing fields, templates, or levels. Consequently, the general trend is often inexorably toward ever-increasing complexity. At some point, ease of use starts to decrease and ease of understanding starts to diminish. In all cases, tailoring the templates should be evaluated from a cost/benefit perspective and with a keen respect for the elegance of simplicity. The objective is not merely **more** information, the objective is **more useful** information.

4.6 USING TEMPLATES TO FACILITATE PROCESS IMPROVEMENT

The first use of the templates is typically to define one or more existing processes. In such cases, the process exists and the templates are used to establish a base-line process model. The next step is to use the templates to define a proposed process and to analyze that proposed process for desirable or undesirable structural characteristics. The goal is to define a process that is an improvement over the existing approach. The long-term objective is to incrementally achieve discrete process improvements, and a steady evolution toward an optimized process.

4.6.1 PROCESS IMPROVEMENT VIA INCREASED PROCESS MATURITY

There are many methodologies for achieving process improvement. In the software engineering industry, process improvement is often perceived to be the systematic progression to increasingly higher levels of process maturity as defined by SEI's Capability Maturity Model (CMM). The chief contribution template-based process representations make toward improved process maturity is their support to organizations transitioning from ad hoc (Level 1) processes to repeatable (Level 2) processes, and from repeatable processes to defined (Level 3) processes.

4.6.1.1 Level 1

If an organization is currently using Level 1, or ad hoc processes, it is important for that organization to begin performing the work in a regular, repeatable manner. Typically, this means deciding which process they want to use then training their people to perform that process. Selecting a process, particularly in an environment where each process tends to be a new approach, can be a difficult challenge. It is made more difficult by the fact that communicating about a process can be both subjective and inconsistent—one person sees and thinks about it one way, another person sees it from their own perspective. Both are right in their view; neither sees the whole picture. This is precisely where process representations can help.

A process representation using consistent terms, objects, and relationships (and possibly supported by a graphical model) will allow people to more readily compare and contrast their different views and more easily come to a common understanding. Furthermore, groups of people can simultaneously coordinate work on process descriptions and models, and each can efficiently verify that their work is consistent with work done by the others.

Once the process analysts have, with management, constructed an approved model of the process the organization would like to follow, that same model can serve as the foundation for the development of process guidebooks, process quick-reference guides, process-oriented training classes, etc. Compared to attempting to develop a multi-hundred page operations manual of an ad hoc process, the development and analysis of a process model offers a relatively low-risk, low-cost, and high benefit alternative. As the organization trains its managers and personnel in using the intended process, and as management and personnel become increasingly adept at repeatedly doing so, the organization moves toward having a Level 2 (repeatable) process.

4.6.1.2 Level 2

After achieving a repeatable process, the CMM describes the next major level as a defined process. As mentioned in Section 4.6.1.1, the process templates and accompanying graphical depictions can serve as a foundation for the development of process guidebooks. If the templates are constructed and maintained electronically, the possibility also exists for automatically deriving process guidebooks based on information maintained within the templates. If you prefer to keep the information contained on the templates to a minimum, you can still use the templates (in an automated environment) to provide the skeleton of a process guidebook. Process analysts would then only need to annotate the information in the initial draft by adding supplemental material that expands upon that which was directly extracted from the templates.

4.6.1.3 Level 3

Even without automated support, the templates can assist an organization in efficiently developing its process guidebooks and thereby moving toward a Level 3 process. With text-based guidebooks, readability, completeness, and consistency are relatively high-risk considerations. As guidebooks become increasingly larger, it also becomes increasingly difficult to ensure that:

- All areas of the guidebook are defined to the same level of detail.

- Each area is compatible with information in other areas.

- Some areas are not redundantly described in multiple locations

- No critical areas have been overlooked.

Process models, especially when rendered graphically, allow much easier insight into each of these characteristics. Consequently, work on a process guidebook can be preceded by work on developing a process model, analyzing that model to assure that it uses a common level of granularity, that the interfaces match between diagrams, that all major activities have been represented, and none twice, etc. When a satisfactory model has been built, the development of a process guidebook is primarily an effort to translate the process model into descriptive-text. Since the templates may already contain a considerable amount of descriptive-text, even this step is simplified.

4.6.1.4 Summary

To summarize, process representations using the combined template and graphical approach described in this guidebook can be an important tool in an overall program of process improvement. The primary contributions derived from this approach are improved understanding of your process and an improved ability to communicate and teach that process to others.

4.6.2 PROCESS IMPROVEMENT VIA REDUCED COUPLING AND INCREASED COHESION

Process improvement can also be performed on a simpler foundation. For example, software engineers have long understood the advantages to designing modularized systems that exhibit the desirable qualities of low coupling and high cohesion. The same principle can be applied to the development of an improved organizational process. This subsection briefly examines structural analysis of process models (and the modification thereof) as a means toward incremental improvements in a process architecture. Although the following discussion makes repeated references toward making changes to information captured on the templates, the goal is not simply to improve the clarity of this information on the templates (such as was discussed in Section 4.5.1). Instead, the primary goal is to fundamentally improve the way a process is conducted as represented by the information on the templates. Improving a process by changing the templates is only the first step. The second step, once the analyst is satisfied with the alternative improved model, is for management to use that model as the foundation for actually altering the way work is performed.

The following approach was developed to work specifically with template-based process representations. However, it can also be used to improve any process description that includes entry criteria, internal structure, and exit criteria as part of the representation. To summarize, this approach recommends that the process analyst attempt, in priority order, to simplify exit conditions, simplify entry conditions, and simplify the internal processing of all activities and tasks participating within a given process.

At the basis of this prioritization are the issues of "fan-in" and "fan-out" and the general problem of excessive process coupling. Consider exit criteria. Suppose one event referenced five other events in its exit criteria. Suppose each of these referenced five more events, and each of those referenced another five. Assuming no overlap, within three "steps" the fan-out has now spread to 125 different events. The coupling and interrelationships between events can rapidly become overwhelming when exit criteria contain unnecessary complexity. Similar problems exist with fan-in or with complex entry criteria. Fortunately, fan-in implies that the complexity exists "upstream" from the event, and the event is (if its fan-out is low) potentially reducing overall process complexity. Note that although template-oriented terms are being used (such as "exit-criteria"), what is actually being discussed here are events actually performed within an organization and the amount and kind of relationships that exist between those events. Complex exit criteria directly corresponds to complex relations of the events being modeled by the templates.

The complexity of an event's internal processing is least important. This is based on principles of information hiding. From one perspective, if the complexity of an event is hidden from the rest of the process, it does not adversely affect any event other than itself. The benefits gained by reducing internal complexity are generally limited to the event only and are not likely to propagate to other events or activities.

Using entry condition complexity, internal structure complexity, and exit condition complexity, the following eight levels of complexity can be defined:

1. Low   Low   Low
2. Low   High   Low
3. High   Low   Low
4. High   High   Low
5. Low   Low   High
6. Low   High   High
7. High   Low   High
8. High   High   High

Any activity or tasks can be characterized by one of these levels of complexity.

These are not steps to be followed in inverse order to reduce the complexity of relationships between and within events. Instead, the ranking is intended to highlight where the greatest benefit can be achieved. For example, a Level 8 event (HHH) can be advanced either to Level 7 (HLH) or to Level 6 (LHH) or to Level 4 (HHL) by either simplifying the internal structure, the entry criteria, or the exit criteria, respectively.

From this perspective, developing an improved model to reduce process complexity and process coupling can proceed quite simply. First, the process analyst evaluates each of the events within a process and assigns a complexity ranking, from 1 to 8, based on where complexity exists within that event. After this rank ordering, the analyst examines the events to decide which ones are potentially the easiest to change. Generally, Level 8 events are examined and optimized first (since they are the most complex and are likely to have the greatest potential for simplification). Proceed with Level 7 then to Level 6, etc. As discussed earlier, the purpose is not to improve an event by one complexity rating, but to achieve the greatest change practical. If, by using the above ordering, it is equally possible to simplify either the entry criteria or the exit criteria, then the greatest gain can be achieved by simplifying the exit criteria.

This technique is a simple approach for using a template-based process model as a foundation for examining and reducing process coupling and increasing process cohesion. There are certainly other approaches that lend support to new models of proposed improved processes. These models, in turn, can guide the development of new guidelines for performing the process.

For example, one technique is to analyze the structure of a process model and reduce the number of events a throughput must transition through before it is considered complete. This is effectively an effort to eliminate unnecessary work from the product development cycle. Another approach is to assure that there are a minimum number of long length serial event chains. Long serial event chains imply a risk of critical path events with no parallelism to ensure the progression of work. In such series of events, delays early in the chain cause delays of all later dependent events.

In these approaches to reduce process coupling and increase cohesion, the common denominator is the analysis of the structural characteristics of the process model. The goal is to alter or rebuild the model so that it depicts a process structure with increased advantageous characteristics and decreased disadvantageous characteristics.

Again, it is important not to confuse improving the model with improving the process it is representing. Improvements to a model are always done from the perspective of the process implications of changing some given event: especially with respect to its impact on other events. Often, changes that appear easy to make on the templates are actually quite hard to enact within the real process. Conversely, what appears difficult or awkward to change within a model (graphical or templates) may actually be very simple to change in the process itself.

In all cases, the process analyst's orientation must be on improving the process. The model is only a facilitation tool: a means toward an objective. It is important to remember that there are distinct differences between improving the clarity of a template model (Section 4.5.1), improving the usefulness of the templates themselves (Section 4.5.2), and improving the process represented by the templates (Section 4.6).

4.6.3 INCREMENTAL CHANGE

These templates and their structure, relationships, and associated data fields are designed to support incremental application and incremental efforts toward process optimization. As stated previously, all that is needed to define a process are the event templates. Role, Resource, Product, Research, and Constraint templates can all be ignored or deferred until later. Furthermore, the essential characteristics of the Event templates are fairly nominal: name, identification, purpose, comments, and entry and exit criteria in terms of other events.

As described in detail in Section 4.4, the first pass at a process model can be built very rapidly by constraining process collection and definition to just event objects, sequence relationships, and inclusion relationships. If further information is desired, a second pass can be done defining authority templates, a few basic roles to exercise the authority (maybe even just two roles: managerial and non-managerial), and relationships between events and the roles required to support those events. A third pass might introduce Resource templates. A fourth pass might be done to break large scale tasks (such as clean proofing the document) into smaller scale activities and tasks (such as reviewing the document, then clean proofing the table of contents, main sections, and appendixes). A fifth pass might begin to include Product templates and relationships to the events that produce the product. Maybe the next pass introduces Research templates, etc.

To summarize, the key importance in the use and optimization of the templates is the principle that defining a process model proceeds one stage, or cycle, at a time. Each stage should be comparatively small and relatively low risk. Each stage should be evaluated in terms of the value derived from process definition in relation to the effort required to construct that definition. This technique corresponds directly to the "goal-driven" approach advocated in Section 1.

Finally, the process analyst should strive to constrain proposed or improved process models to those that are realistic within an organization. From the perspective of support for process improvement, the goal is not to develop the ideal model; the goal is a model that gives insights into the ways that **feasible** incremental process improvement can be achieved.

5. ALTERNATIVE PROCESS REPRESENTATIONS

This section examines opportunities for constructing or extending process models by supplementing the templates with other representations, or vice versa. A set of alternative process representations are introduced (STDs, ETVX, SADT, Statecharts, Petri nets, PASTA, and Role Interaction Nets [RINs]), and each is examined from the perspective of process modeling.

As can be seen in Figure 5-1, after completing this section you might elect to skip Sections 6 and complete your study of this material by reading Section 7. This approach provides all the essential information relevant to formal process definition and modeling. However, for further guidance on establishing a process definition and modeling program, continue with Section 6 after completing this section.
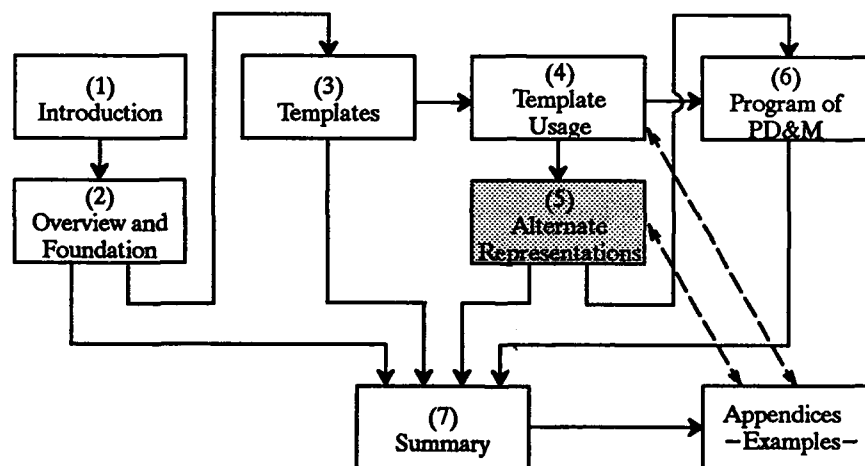
Figure 5-1. Guidebook Organization View 5

5.1 PROCESS OBJECTS AND RELATIONSHIPS

A model is an abstraction of a real-world object or phenomenon. Alternative approaches to process modeling can be contrasted by examining the objects elaborated by each approach and the relationships permitted among those objects. There is no universal standard for what is a "better" approach and what is "worse." All that can be evaluated is which approach has the greatest potential for helping you achieve your goals, in your environment, using your resources, given the funds available in your budget, etc. Additionally, significant changes in any of these variables may render one approach progressively less beneficial and another approach more beneficial. Therefore, the purpose of this section is not to provide a rank-ordering of better and worse approaches to process definition and modeling; instead it is to provide a basic framework that you can use when performing your comparative analysis, and to present information that you can use in determining the approach best suited to your requirements.

Before examining each of the alternative notations, the following material constructs a meta-model composed of template objects that graphically shows legal use of the relations which can be used to

connect template process objects. After briefly discussing this material, components from these diagrams will repeatedly be used to compare and contrast the alternative notations.

5.1.1 TEMPLATE META-MODEL

As presented in Section 3, the template-based modeling techniques described in this guidebook define objects and the relationships that connect those objects. There are four major classes (meta-classes) of objects:

- Event objects

- Throughput objects

- Support objects

- Constraint objects

Events include both managerial and production events. Throughputs include products and research. Supports include roles and resources. Constraints include internal and external constraints.

In order to bind objects to other objects, there are four major types of relationships that have been defined:

- Inclusion relations

- Sequence relations

- Specialization relations

- Reference relations

As reviewed in Sections 5.1.2 through 5.1.4, inclusion relations are used to indicate object decomposition, sequences relations are used to order events, specialization relations allow inheritance, and reference relations are used to capture any type of relation not already covered by the other three.

In the following material, STDs, Statecharts, and other notations are examined in terms of these objects and relations. Generally, this set represents a superset of the objects and relations applicable to the various notations. In other words, each of the alternative notations can be examined in terms of the support they give to modeling event, throughput, support, and constraint objects and the support given to modeling inclusion, sequence, specialization, and reference relations.

5.1.2 REFERENCE RELATIONS

From the perspective of the templates and as shown in Figure 5-2, event objects can establish reference relations to each of the three other objects, and events may also make references to other events (recall from Section 3 that reference relations represent an object making any type of reference to another object and were indicated using dashed lines). When constructing a template-based process model, the process analyst might elect to take an event-driven approach and proceed by first defining events and then defining (reference) relationships between events and other principle process objects (i.e., throughputs, supports, constraints, and other events).

Sequence ——→

Inclusion ----→

Specialization ○——→
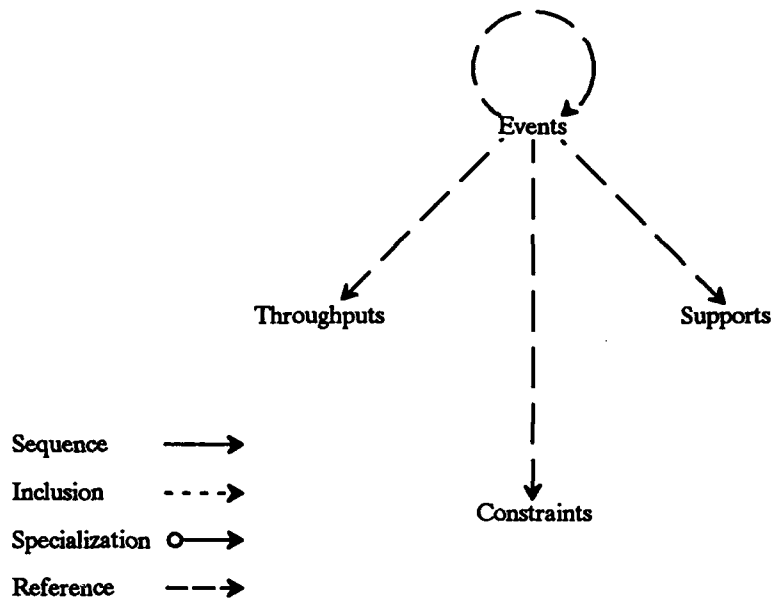
Reference ———→

Figure 5-2. Reference Relations: View 1

However, it is also permissible, as shown in Figure 5-3, to have throughputs and supports making direct reference to constraints. Throughputs may also directly reference other throughputs, supports may directly reference other supports, and constraints may reference other constraints. These are shown on the diagram as a loop (dashed-line) both originating from and terminating at the same object.

Sequence ——→

Inclusion ----→

Specialization ○——→
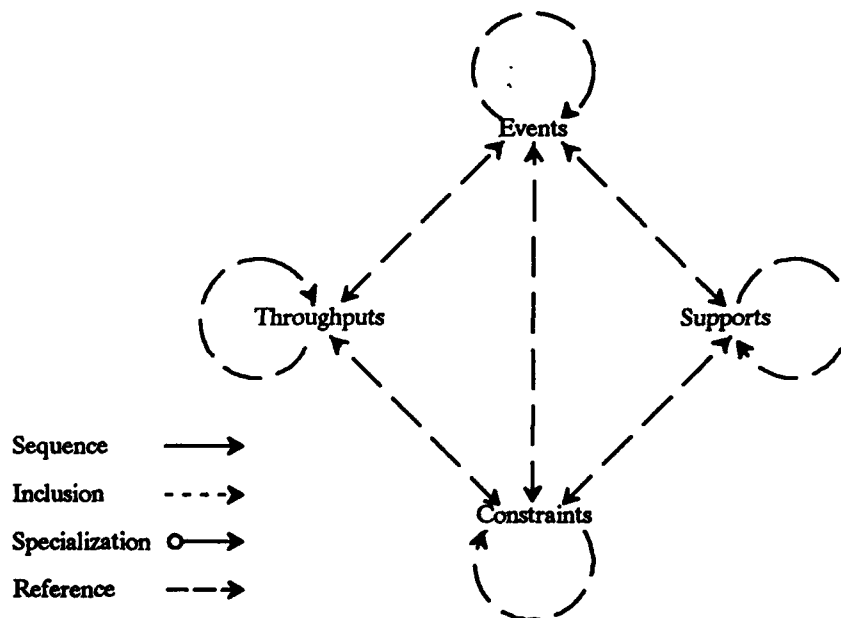
Reference ———→

Figure 5-3. Reference Relation: View 2

Additionally, all relationships exist as two-way links. If, for example, an event references a support, then that support must also reference the event. This provides a direct means to verify the consistency of the relationships that have been incorporated within a model. Figure 5-3 shows these two-way

relations as double-arrow lines connecting dissimilar objects. As indicated in this diagram, with only one exception all objects can directly reference all other objects, and each object can reference itself. The only exception is that the templates do not allow direct relations between throughputs and supports. This is because a support (role or resource) can only work on a throughput (product or research) via some event. Similarly, a throughput is only affected by a role or resource as a function of some event. Therefore, these relationships always exist as support-event-throughput relation sequences or as throughput-event-support relation sequences.

5.1.3 SPECIALIZATION RELATIONS

In the inspection example shown in Section 3, the inspector role was shown as having two specialized forms: key_inspector and regular_inspector. In a similar way, any other object can be represented with a "child" being a specialized form of the parent, and the parent being a "generalized" form of the child. Therefore, in addition to reference relations (discussed in Section 5.1.2, and shown in Figures 5-2 and 5-3 as dashed-lines) the templates support the specialization relation as shown in Figure 5-4. This relation is depicted (both in graphical models and in this template meta-model) as a solid line that has a small circle as a "tail."
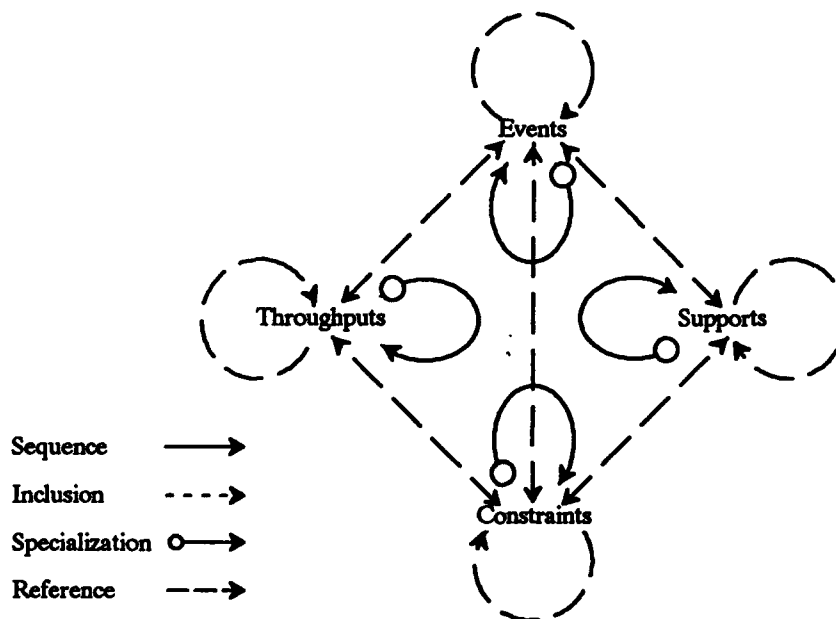
Figure 5-4. Addition of Specialization Relations

As indicated in Figure 5-4, specialization relations are restricted from combining dissimilar objects. Consequently, events may only be a specialized form of other events, supports may only be a specialized form of other supports, etc. Note, however, that all objects may be depicted using the specialization relation, if necessary.

Specialization relations are always shown with the arrow originating at the child and terminating at the parent. The reverse link for this relation is considered, as introduced above, a generalization relation. This relation depicts parents as generalized forms of children. Finally, note that specialization and generalization are all relative. Take, for example, a specialization hierarchy that is five levels deep. Objects on the third level would be specialized forms of objects on the second level (their parents), but they would be generalized forms of objects on the fourth level (their children).

5.1.4 SEQUENCE AND INCLUSION RELATIONS

The final diagram in this section (Figure 5-5) adds the remaining two relations that you will use when examining the other representations. The first of these relations is the sequence relation. This is shown on Figure 5-5 as a dark, solid line that loops from the event object back onto itself. The sequence relation is the only relation that is restricted or dedicated to one specific type of object: events. The sequence relation shows an ordering or partial ordering on a set of events by pointing from "pre-set" events to "post-set" events. That is, for any given event, all events showing a solid arrow terminating at the given event have exit criteria that must be satisfied before the given event's entry criteria can be satisfied (the pre-set of events). Similarly, the post-set of events for the given event are all those events whose entry criteria depend on the given event's exit criteria: pointed to by solid arrows originating at the given event and terminating at the post-set of events.

Sequence
Inclusion
Specialization
Reference
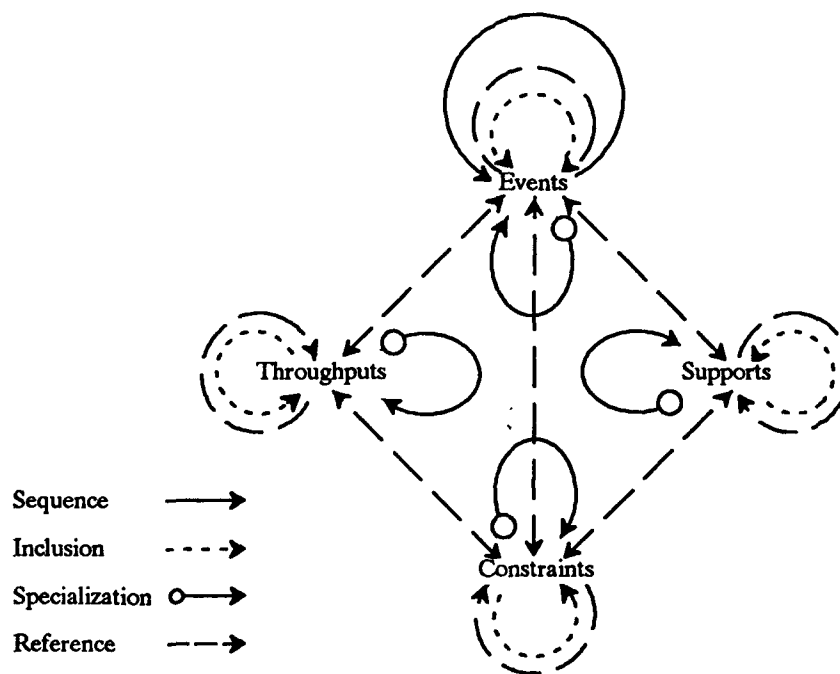
Figure 5-5. Addition of Sequence and Inclusion Relations

The other type of relation indicated in Figure 5-5 is the inclusion relation. Inclusion relations are shown by small looping, dotted-line arrows. As depicted, any object can include other objects of its own type, and no object can include objects of a different type. The inclusion relation is commonly used as a means for decomposing complex objects into simpler objects or for decomposing group objects into their components parts (each of which might further decompose into its parts, etc.).

Figure 5-5 shows all the objects and relations legally permissible using the template-based approach. (As stated throughout this guidebook, this set of process objects and relations is designed to be augmented or modified to suit site-specific objectives. This section, however, discusses the alternative notations entirely in terms of the basic set of four objects, and four relations.)

5.2 ALTERNATIVE REPRESENTATIONS

Discussed below are the following six alternative process representation and modeling notations:

- STDs

- ETVX

- SADT

- Statecharts

- Petri nets

- PASTA

- RINs

These notations were selected to accomplish a variety of objectives. STDs are discussed first because they offer one of the simplest approaches for modeling a process. Although STDs are quite limited when compared to other techniques, they can be quite useful for learning or teaching purposes.

ETVX and SADT were both selected because of their wide-spread acceptance by industry. Originally designed as techniques for representing software-based or software-oriented processes, they can be generalized for use in process definition and modeling.

Statecharts and Petri nets were selected because these notations offer opportunities for gaining insights not only into the static but also the dynamic characteristics of a process. PASTA is included because it was designed to support EBPM.

Finally, RINs are included as an example of a notation that readily translates into Petri nets and/or Statecharts.

5.2.1 STATE TRANSITION DIAGRAMS

STDs are often used for describing finite automata (finite state machines). Any process that can be described in terms of a finite automaton can be represented using an STD. Generally, a finite automaton accepts some series of input symbols and typically produces some series of output symbols (Kolman and Busby 1984). Each such output symbol is a function of the relevant input symbol, the current state of the automaton, or both. Additionally, as input symbols are received by the automaton, its state may change. With regard to process modeling, input symbols and output symbols typically represent milestones that occur in reality, and states within the machine represent different phenomena or activities within an overall process. States have a time-dimension, state transitions do not. Thus, finite state machines can be seen as one possible representation for modeling sequences of phenomena within some defined domain. When modeling the behavior of large or complex systems, it is often useful to discuss both states and sets of states. The term "mode" is optionally used to distinguish a state set (Sanden 1992a).

The dynamic behavior of a state transition machine is simply described. All state machines, at the outset, are set in their initial state. Whenever a legal input symbol is received, the machine always transitions to the next state, which may or may not be identical to the current state (this is the state transition).

From the data collection templates, it is state information captured by the templates that directly translates to STD models. It is often easier to represent events as arcs. The arcs among events can be throughput and state transitions; however, STDs are not good for representing supports and constraints. Also, the process engineer may want to consider an STD for throughputs (as they have state transitions also); but typically, not much insight would be gained from such a diagram (that is, it is likely to be just a chain of states, perhaps with a simple loop involved).

Because of scale-up problems, STDs are best used when a process (or part of a process) is not excessively large or complicated. Another advantage to STDs is that, due to their relative simplicity, the initial costs for training and the learning curve are likely to be lower than using other less familiar or more complex process models.

From the perspective of the objects and relationships used within the templates, STDs primarily capture event-event sequence relations. However, as shown in Figure 5-6, carefully constructed STDs can construct an intertwined model that not only shows events and their states, but also throughputs (products, for instance), and their evolution through a variety of states. Note that throughput state transitions can be used to drive the ordering of events, events can be used to determine the states of throughputs, or some combination of these two approaches can be captured—all using STDs.
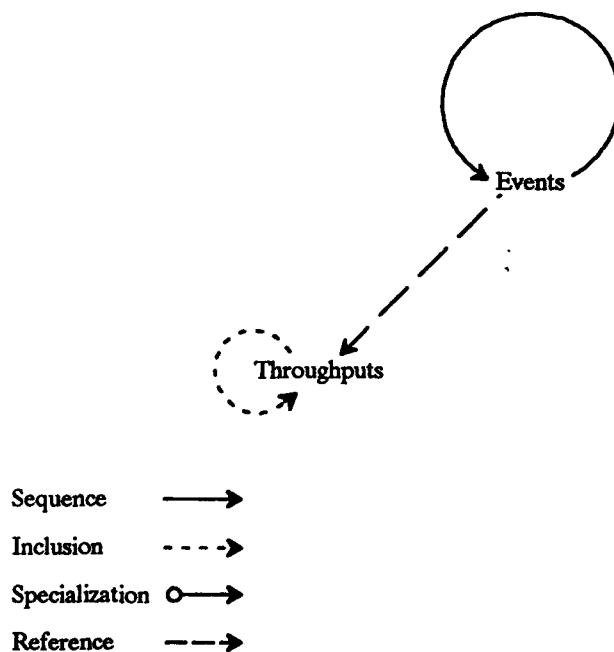
Figure 5-6. State Transition Diagram Relations

5.2.2 ENTRY-TASK-VALIDATION-EXIT

The premise behind the development of ETVX was the necessity to find a means to embed methods and tools into a common framework for intellectual and management control (Radice and Phillips 1988). ETVX (see Figure 5-7) is a quasi-diagrammatic representation of IBM's PPA. The authors of ETVX take the position that PPA is the highest representation of the software process and that

although it contains the necessary elements for representing software engineering environments and activities, it also is applicable across a much broader framework (Radice and Phillips 1988).
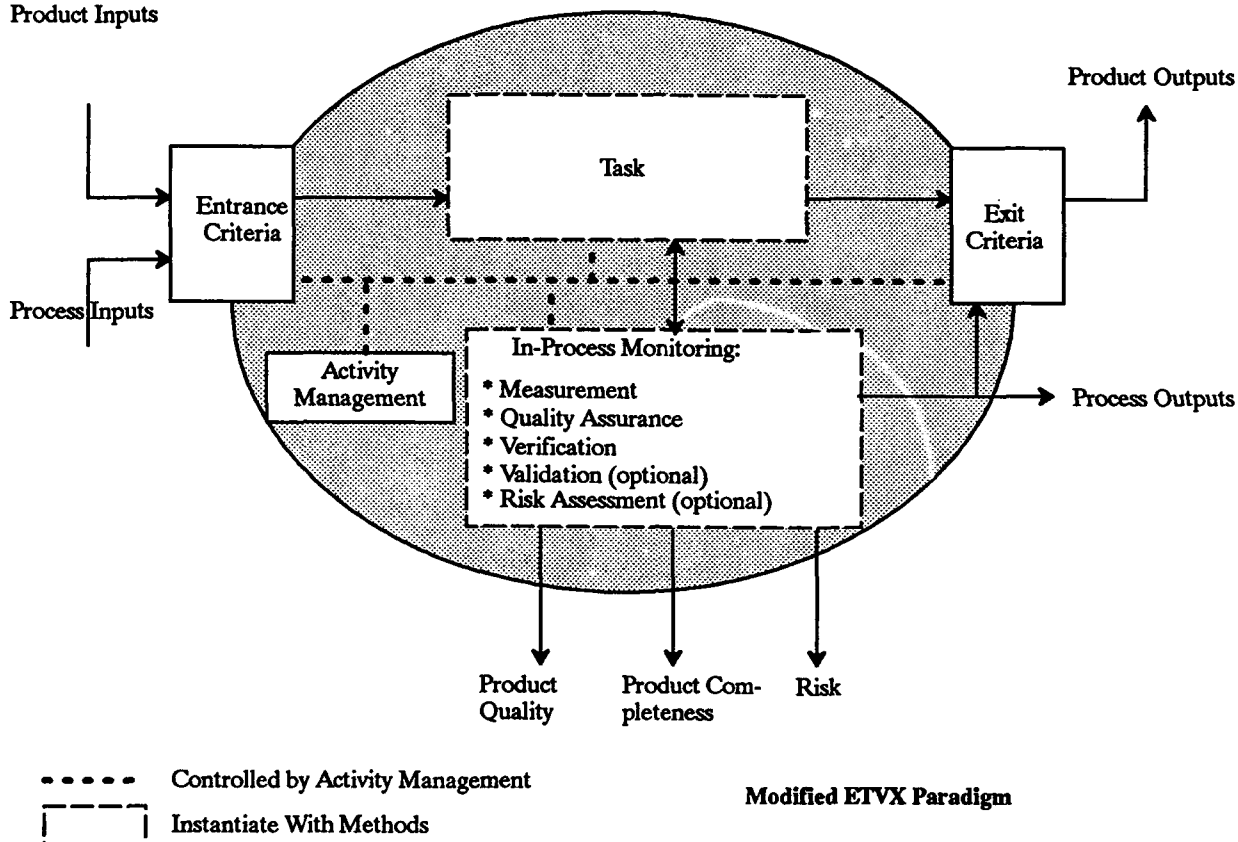
Figure 5-7. Entry-Task-Validation-Exit Diagram

Part of the motivation behind PPA was the belief that it is essential to have the ability to rigorously manage processes beyond the levels provided by individual tools and techniques. Also important was the need to provide some means whereby an evolving process could, by virtue of how that process was enacted and applied, influence or govern the requirements for new support tools.

Generally, PPA is intended to define a basis for beginning an orderly evolution in the way that software is engineered. Radice and Phillips (1988) claim that PPA:

- Ensures a repeatable and simple paradigm at all levels of the software process.

- Contains the means for self-improvement by basing itself on the need for statistical quality control.

- Requires a validation mechanism for any work item produced during the development cycle.

- Is based on what already exists in the software industry and draws only from the best proven alternatives.

- Addresses the complete life cycle of software production.

- Does not require a complete set of tools in its first iteration.

The ETVX paradigm is a procedural formalism for representing activities and relationships within PPA. An ETVX box represents the concept that at any level of abstraction a work activity must have entry (E) and exit (X) criteria, some task (T) to be done, and some means or collection of means for performing validation (V).

Hierarchical decomposition is achieved in ETVX by "exploding" the task (T) component of an activity and showing the subactivities (each in ETVX form) of which it is comprised. (Similarly, any subactivity can also be further decomposed, as necessary.) Additionally, the ETVX model does not imply all activities or tasks in succeeding stages wait for completion of preceding stages. Later stages may be functioning (that is, activities occurring) concurrent with preceding stages.

The only constraints governing the commencement of activities at a given stage are its entry criteria. Once the entry criteria are satisfied, the task related work of that stage may commence. After task related work is completed, validation may commence. Typically, the validation effort will yield information for use in evaluating the degree of compliance achieved with regard to the exit criteria. Formally, a stage is not complete until all of its exit criteria have been met. However, one or more exit criteria may be satisfied at any time during the task related work. In this way, entry criteria to other stages may become satisfied prior to complete satisfaction of exit criteria in prior stages, thereby leading to the previously mentioned parallelism and general asynchrony.

You should note that the ETVX approach does not yield an interconnected diagrammatic representation of the system being modeled. Instead, ETVX rigorously examines each of the four subcomponents that constitute an activity.

When using the templates, E and X are the **Entry Criteria** and **Exit Criteria** on the Event template. The parent/child relations on the template can be used to define ETVX decomposition. Validation is best captured by defining a set of events that are explicitly intended for performing validation, and then heuristically asserting that all events in a given model must, somewhere within their internal processing, invoke one or more events from the validation event-tree. **Internal Processing**, also on the Event template, maps to the tasks (T) in ETVX. The **Child Events**, from the Process template, explicitly describes the tasks in ETVX. Even though the templates contain all of the information for ETVX, it is important to remember that ETVX is not an operational representation. ETVX's usefulness comes from its conceptual presentation for process definition.

ETVX is useful for defining the high-level organizational processes since it has greater flexibility than the other approaches. This is especially useful in volatile environments where you may want to repeatedly change and update your representation as your understanding of the process grows. ETVX can also define a process in an environment where management is not willing to support the process control of a well defined process. ETVX is a good process representation when parts of the process, specifically some of the lower-level events, are defined in greater detail. This can provide a good compromise between high-level flexibility and low-level detail. Additionally, since ETVX has a hierarchical presentation mechanism, essential for representing large and complex processes, it represents hierarchical processes well but not as explicitly as other models.

From the perspective of the objects and relationships used within the templates, ETVX models primarily capture event-event, event-throughput, and event-constraint reference relations (Figure 5-8). The decomposition of events (shown by the inclusion relation) of events into subevents is also well supported. Notably absent from the ETVX approach are explicit conventions or techniques dedicated to representing the supports (roles and resources) required within a process. To capture such information,

the process modeler needs to rely on **Entry Criteria** and **Exit Criteria**, and he possibly needs to add adjunct documentation to detail the characteristics of various supports.

Sequence ⟶

Inclusion - - - ->

Specialization O⟶

Reference — — →
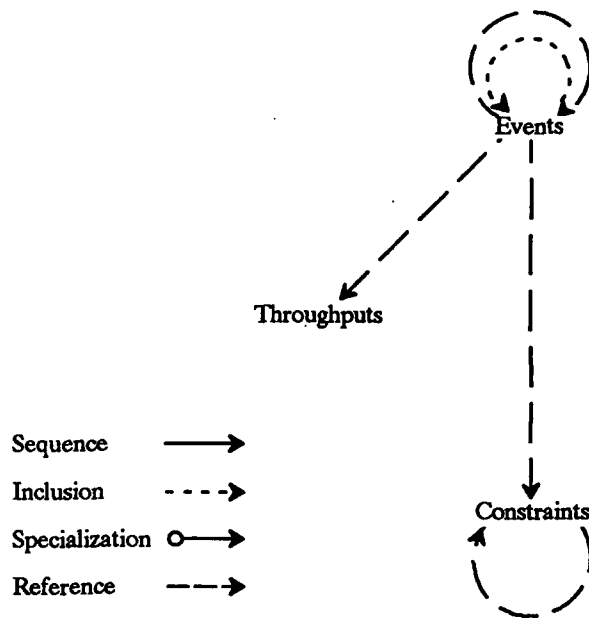
Figure 5-8. Entry-Task-Validation-Exit Relations

5.2.3 STRUCTURED ANALYSIS AND DESIGN TECHNIQUES

When applying the SADT to software systems, the overall approach consists of identifying activities, identifying the inputs and outputs of those activities, identifying factors that constrain the activities, and identifying resources or materials that support the activities (Marca and McGowan 1988).

As Figure 5-9 depicts, activities are represented diagrammatically as boxes. Inputs to an activity are labeled arrows arriving at the left side of the box. Outputs from an activity are labeled arrows departing from the right side of the box. Constraining influences are labeled arrows arriving at the top of the box, and enabling mechanisms are labeled arrows arriving at the bottom of the box.
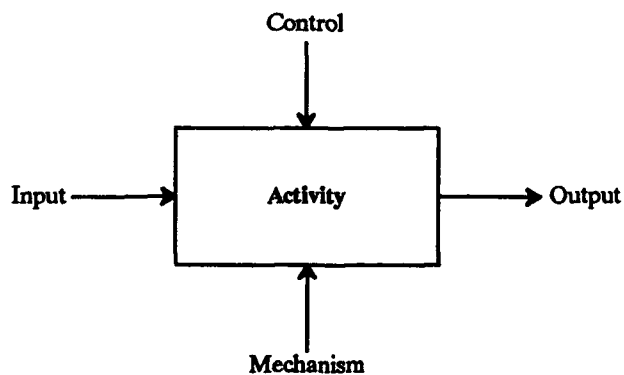
Figure 5-9. Structured Analysis and Design Technique Diagram

The outputs from one box may be the inputs, controls, or enabling mechanisms for any other box (including, in rare cases, itself). Boxes are all named and all arrows carry labels. Arrows are allowed

to be split into multiple branches or join to combine multiple branches into one. Any box can be decomposed into any number of subboxes. These, in turn, can be decomposed, and such decomposition can repeatedly continue until the necessary level of detail has been achieved. Inputs, outputs, and controls define the interfaces between boxes, and enabling mechanisms permit the controlled mixing of subjects. When a box is "exploded" to yield a new subordinate diagram, the box and diagram boundaries must match.

Figures 5-10 and 5-11 depict a small example of the top two layers of a process. Generally, and for diagrammatic clarity, a diagram is restricted to three to six boxes. This approach allows for a gradual progression in the presentation of details. Also, huge models are discouraged in favor of collections of many small, interrelated models. Each of these smaller models contributes meaning to and derives meaning from its interactions with the other small models. In principle, developing an understanding of each of the small models and it relationships to other small models will lead you to a clear understanding of otherwise very complex systems.

System Risk
Management
Plan

Software
Policy

TRW Ada
Process
Model
0

System
Requirements

Software
Product

Project Team
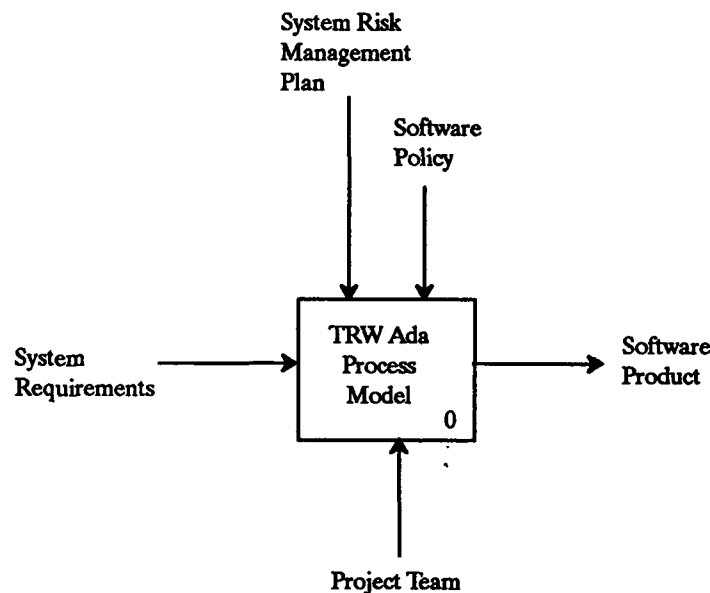
Figure 5-10. Structured Analysis and Design Technique Example 1

Commonly, it is throughputs that would be modeled in SADT as arrows arriving at the left side of a box and/or departing from the right side of a box. Constraints (especially external constraints) would be modeled as "control" arrows arriving at the top of an SADT box. The Support templates (Roles and Resources) can be used (from the event perspective) to represent the enabling mechanisms (arrows arriving at the bottom of an SADT box). Further, SADT permits capturing a considerable amount of text information; therefore, comment and description fields and other relevant information can be readily transferred from the templates to SADT, and vice-versa.

SADT is perhaps the most popular software PN being used to date. It is widely used for software system design, and there are a number of automated tools available on the market which support the technique. SADT is capable for large scale process definition; but SADT, like ETVX, is weak at capturing process dynamics. For example, an SADT link can only carry the syntactic structure of the process information but not the semantics of the enactment. For SADT to be enactable, a process engineer must define the type of link on the SADT diagram and then implement the semantics of the link. Along with the SADT diagram, the process engineer should also use the data dictionary for the implementation of
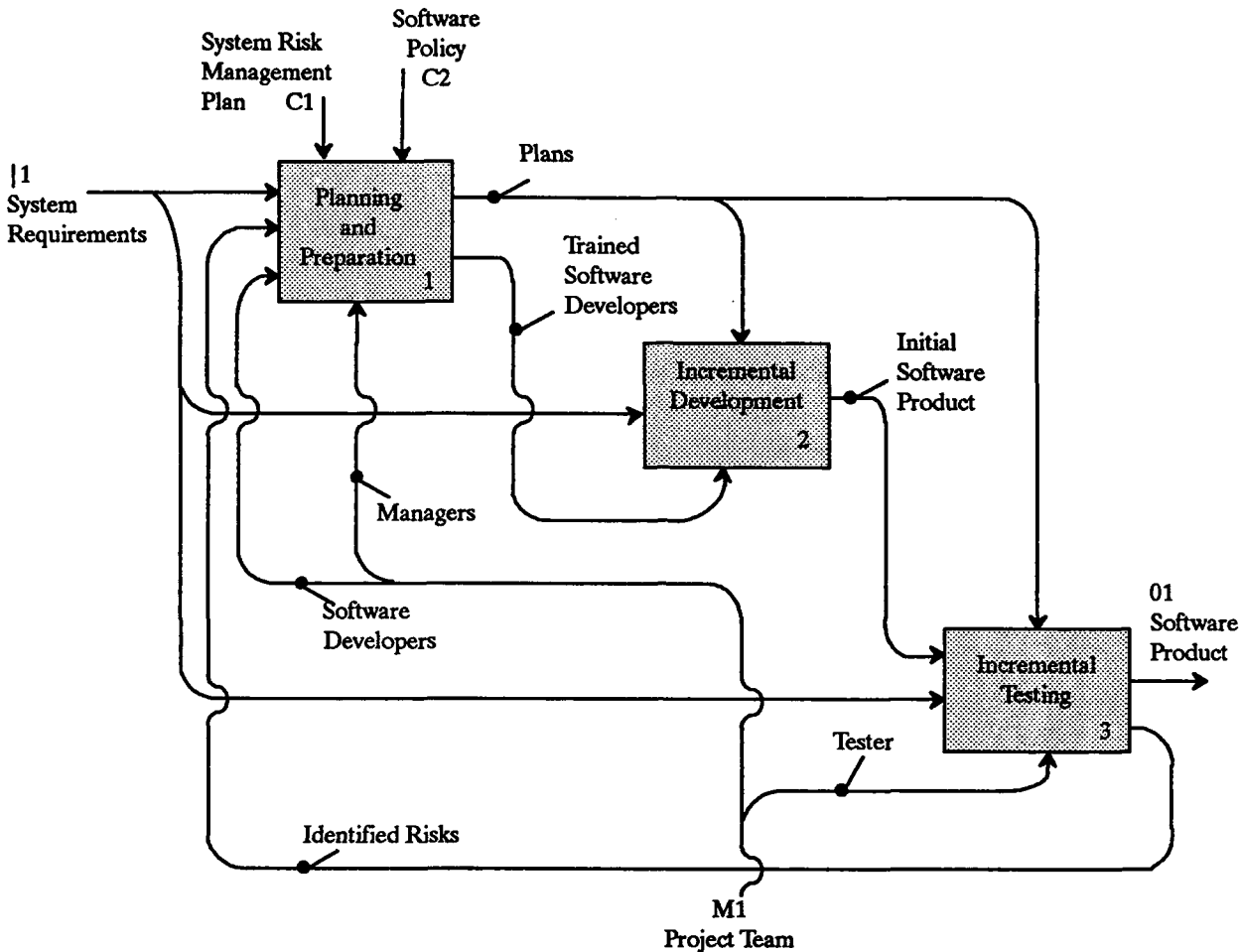
Figure 5-11. Structured Analysis and Design Technique Example 2

enactment. Other limitations with SADT include the difficulty representing the concept of roles and authority for a process. However, SADT can be relatively easy to use for defining large processes since it is capable of representing higher level abstraction and process decomposition structure.

As summarized in Figure 5-12, SADT models readily capture virtually all the reference relations between and among objects. Events can reference other events, throughputs, supports, and constraints. Similarly, throughputs and supports can each reference themselves. SADT also readily supports decomposition. The figure (Figure 5-12) shows inclusion (decomposition) relations existing on all four process objects. Although SADT primarily supports event decomposition, deliberate use by the process modeler will allow SADT models to show relative decompositions of throughputs, supports, etc. Finally, as highlighted by the event-event sequence relation in Figure 5-12, SADT does support capturing information about the sequence or order of events.

5.2.4 STATECHARTS

Statecharts are an extension of the basic notation used for finite state machines. Statecharts allow a finite automaton to be decomposed into a representation that models two or more interacting or communicating subsystems. Statecharts also support hierarchical decomposition of transition diagrams so that various levels of abstraction can be independently represented (Sanden 1992b). Figure 5-13
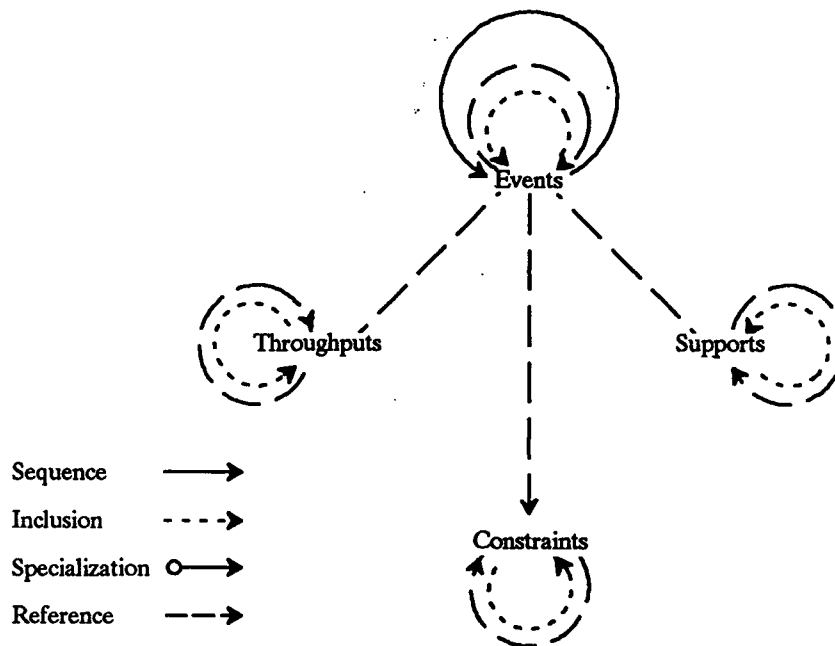
Figure 5-12. Structured Analysis and Design Technique Relations

and Figure 5-14 are examples of Statecharts adopted from Marc Kellner's SEI technical report (Kellner 1989). Figure 5-13 is an activity chart of Statemate that represents the functional perspective of process. Figure 5-14 is Statechart of Statemate that represents the behavior perspective of process.

Statecharts are commonly used to model concurrent, real-time system behaviors. Consequently, they are an intuitively attractive technique for general process modeling. Since statecharts were intended to model systems that are typically very complex, they contain a variety of conventions which both simplify and clarify the representations. In addition to the hierarchical decomposition already mentioned, there are techniques for representing superstates, conditional transitions, default states, history states, conditional entrance, selection-based entrance, and timeouts (Haral 1988; Coleman et al. 1990).

Hierarchical decomposition, in combination with the concept of superstates, allows gathering sets of states together that have common transitions. Typically, the superstates of a level are defined, then subordinate levels of substates are defined as an expansion of each superstate. Each such substate can, in turn, be viewed as a superstate and itself defined in terms of still lower substates. This iterative process can be repeated until the desired level of detail has been achieved.

Conditional transitions provide a simple means for extending the concept of some milestone-event causing a transition from one state to another state. Specifically, milestone-events can be coupled with conditions, and then only when the right combination of milestones and conditions occurs does the state transition happen. Using this technique, it is possible to represent situations where an activity causes a state transition only if some condition is simultaneously true or where an activity results in one type of transition given one condition and the same activity results in another type of transition given a different condition. Obviously, a condition may be replaced by an arbitrarily complex conditional expression, and thereby represents increasingly complex interactions. Conditional transitions are diagrammatically represented by following an arc's listed transition (or transitions) with a "/" character which in turn is succeeded by a simple or complex conditional expression.
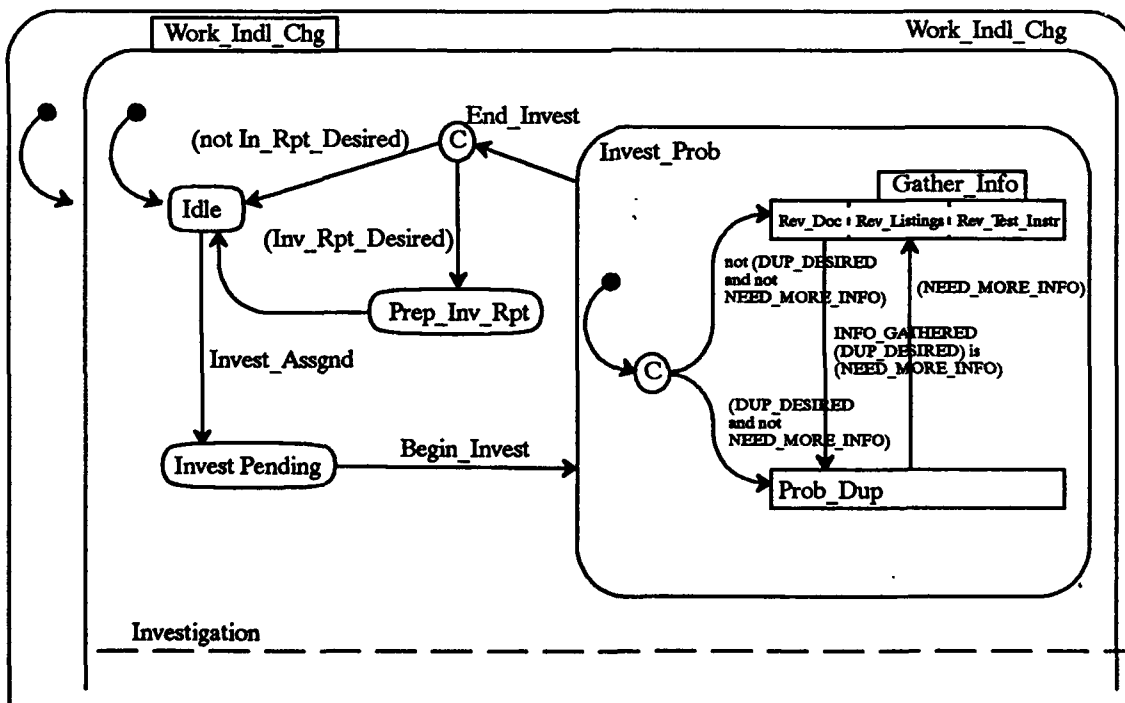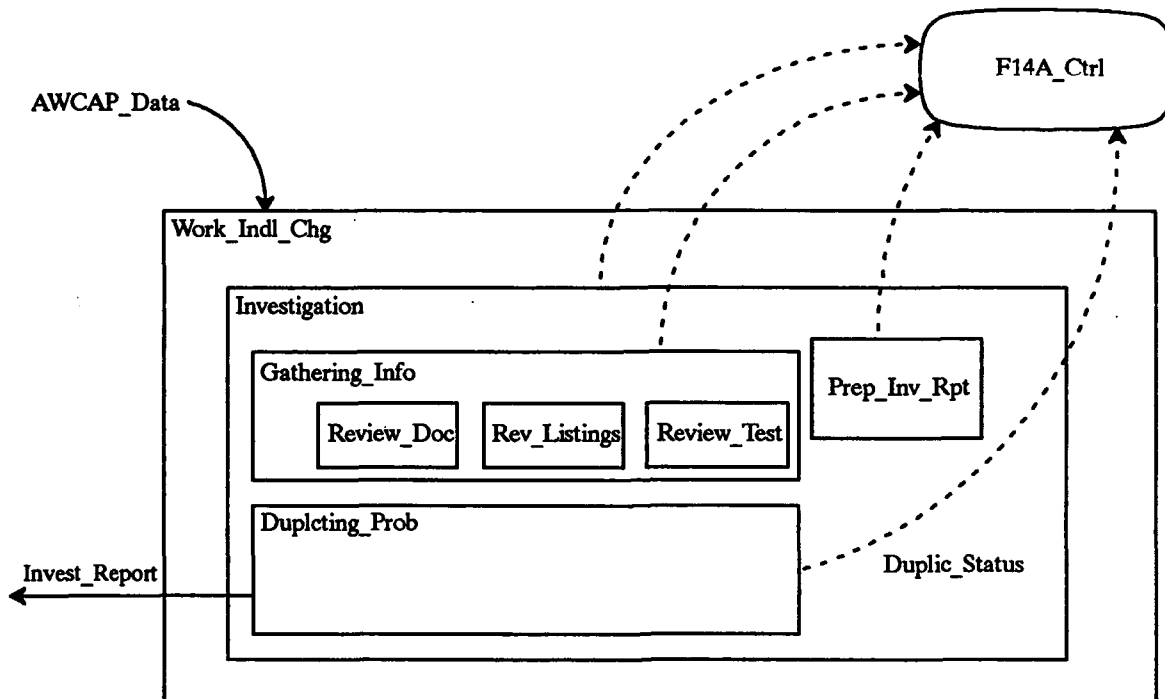
Figure 5-13. Statechart Example 1

Default states are a simple extension within statecharts that allows a single state within each orthogonal collection of states to be labeled as its default state. Independent of whether there are one or more orthogonal subsystems of state machines, any state machine can be marked to represent the default state of that machine. Furthermore, each subordinate refinement of superstates into substates can also include a marking that represents default states. By using this technique, it is possible to simply
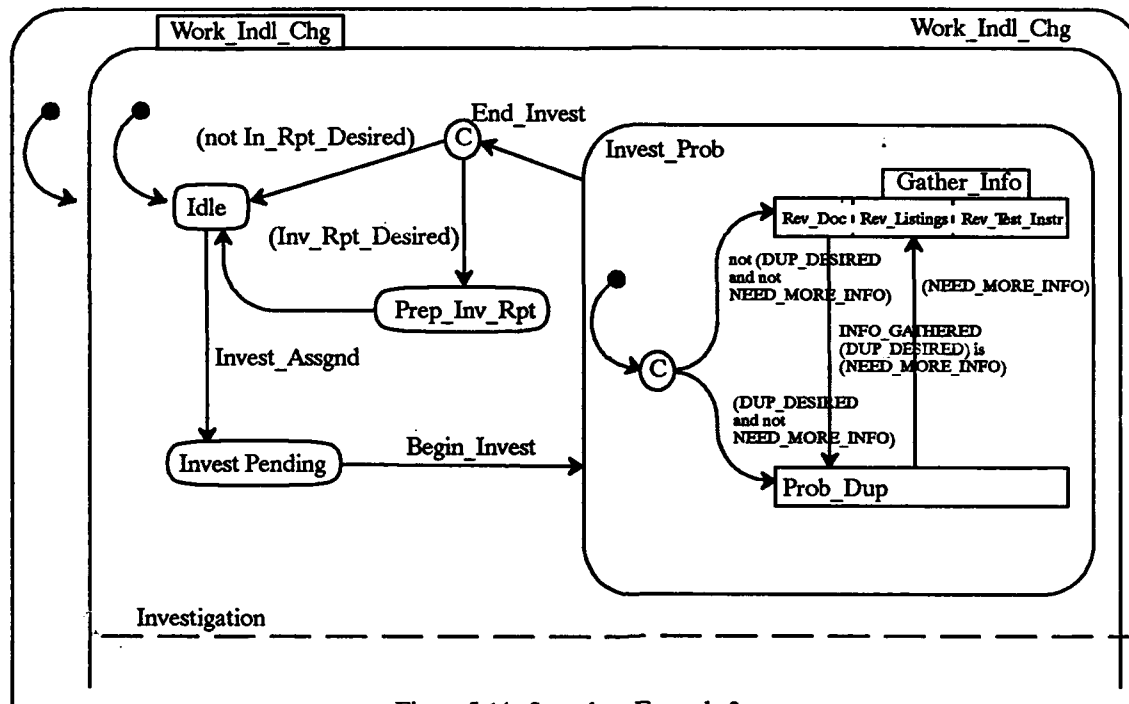
Figure 5-14. Statechart Example 2

and explicitly represent the behavior of a machine (or "submachine") upon its receipt of initial input. Default states are represented by an unlabeled arrow pointing to one of the states within a machine.

"History states" augment the concept of "default states." With default states, the machine always commences from the indicated default, regardless of any activity which may have happened internal to that machine during a prior activation. However, history states provide a mechanism for making the entry state to a machine a function of the state that the machine was left in when last activated. As implied by the name, with history states the history of activation directly affects the behavior of future activations. (With default states, a machine's prior activity is locally unimportant—the machine always commences from the same state.) The relevance of history within a given machine is indicated by placing an unlabeled arrow pointing to a small circle surrounding the letter "H" inside the boundary rectangle of an entire machine. Note that a machine might have both a default state and history states. In such circumstances, the default state is only relevant during the first activation of that machine. All subsequent activations would be influenced by history.

Conditional entrance is an abbreviated way to represent complex transitions from one state to a number of substates. If a state, for example, transitions into one of five different substates of another state, one diagrammatic representation would be to explicitly show the five conditional transitions, each bound to the relevant substate. Alternatively, conditional entrance can simplify the diagram in the following way. Instead of showing five condition-driven variations of the event causing the transition from the first state to the second state, only one line is drawn from the first state to the second state, and it is labeled "unconditionally" as the particular event. Then the decomposition of the second state shows the conditional information as follows. The event is shown as a single inbound event on a line pointing to a small circle surrounding the letter "C" (signifying the conditional entrance). From this encircled "C" radiates (in this example) five lines, one to each of the substates. Each of these lines is then labeled with the appropriate condition or conditional expression.

Selection-based entrance is similar to conditional entrance except that selection entrance is used when decomposing a higher state into its substates and when those substates have a clearly defined one-to-one correspondence with a range of discrete values generated by the superstate. Consequently, when showing, for example, 26 different substates (representing each letter of the alphabet, any one of which might be "selected" during the superstate), it is not necessary to show 26 inbound lines, each labeled with an activity representing the selection of the appropriate letter. Instead, these 26 labeled lines can be replaced by an unlabeled arrow pointing to a small circle surrounding the letter "S." Notationally, these two approaches are equivalent, but the latter notation eliminates considerable clutter and usually results in more readily interpreted and maintained diagrams.

Finally, there are occasions, particularly with real-time system representations, where it is necessary to ensure that the system or some subsystem does not linger unnecessarily in some state. This constraint is represented in statecharts by including a small, wavy line on the left side of the upper border of the state. Under this line would be a time value indicating when this state would automatically transition to the next (and possibly prior) state.

The time constraint concept includes a lower bound in addition to the upper bound. The lower bound provides a way to represent that once a state is entered, that system (or subsystem) remains there for at least the specified minimum time. In effect, all transition signals (or inputs), whether relevant or not, are ignored until the minimum time has passed. Neither, both, or either of these temporal constraints can be used in conjunction with a state (Haral 1988).

With these extensions, Statecharts are effectively a more robust and flexible implementation of STDs. Note that STDs can be represented using statecharts without any loss of behavioral meaning, but the converse is not true. Although relatively simple activities can often be easily diagrammed using STDs, such diagrams often become progressively more unwieldy and unmanageable as the complexity of activity increases. As a general rule, the options available in statecharts become increasingly valuable as the process being represented becomes increasingly complex.

As with STDs, the templates can be used to define a detailed state-driven model which converts almost directly into a Statechart model. Statecharts are especially good at capturing "parallel" execution of multiple state machines and can explicitly capture their interdependencies. Therefore, parallelism in events can be easily captured. However, it is important to note that if the process engineers want to eventually use Statecharts, they need to clearly note any parallelism and interdependencies that occur between steps within events.

Statecharts are good for representing critical processes where simulation and enactment are important. The tool, Statemate, will generate 'C' or Ada code for connecting to programming libraries where various process assets can be implemented. For a given event, if the set of tasks in a process definition can be limited to a set of key fundamental activities, they can be implemented to work with code generated by Statemate. Some potential problems with using Statecharts include that it typically requires low level detail. With Statechart, it is difficult to represent roles and resources (supports) and research. As with many notations, Statecharts can become quite difficult to scale-up for modeling highly complex, large processes. Since Statecharts capture dynamic process information, a process defined using Statecharts can be translated into a Petri net model. Dynamic models have the advantage of being subject to assessment and simulation (providing process engineers results include time-to-completion, dead-lock, completeness, consistency, correctness, race conditions, dead-locks, and similar dynamic phenomena).

As is summarized in Figure 5-15, Statecharts primarily capture events, events sequencing and coordination, and event decomposition. The execution of and interaction between events is highly constrained. When properly constructed, it yields an hierarchically decomposable executable model. Also shown in Figure 5-15 is the representation of throughputs in Statecharts. In practice, modeling throughputs can be difficult because they must also be described in terms of states and state transitions. However, this can be done with the resulting model showing the relationships between changes in event states and changes in throughput states: each (typically) highly constrained by the other.
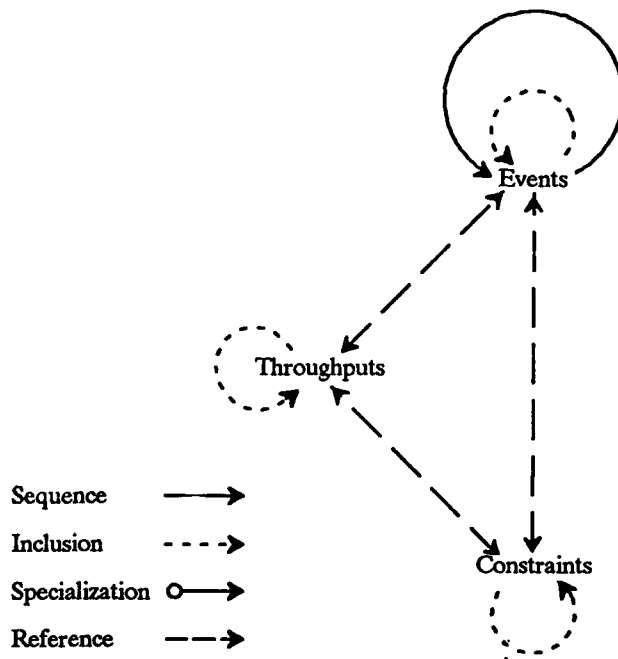
Figure 5-15. Statechart Relations

5.2.5 PETRI NETS

Petri nets are becoming progressively more widely used as a means for building a wide variety of process representations. Petri nets have been successfully used to model manufacturing processes, chemical processes, hard realtime embedded processes, etc. One of the most important characteristics of Petri nets is the fact that they capture the dynamic behavioral characteristics of the system being modeled (Figure 5-16). In effect, Petri nets can be executed.

In addition to the graphical notation, Petri nets also come with a significant body of mathematical formalism. By relying on the mathematical substructure of these diagrams, it is possible to do a static structural analysis of the system's dynamic behavioral characteristics without having to resort to actually running a simulation. This is of key importance, since it allows formal interpretation and analysis of a process model for both desirable and undesirable characteristics (Levis 1992).

Another key factor contributing to the facility of Petri nets is its graphical nature. The basic graphical representation principles are conceptually simple to learn and understand, yet they can be used to build detailed representations of complex systems or models.
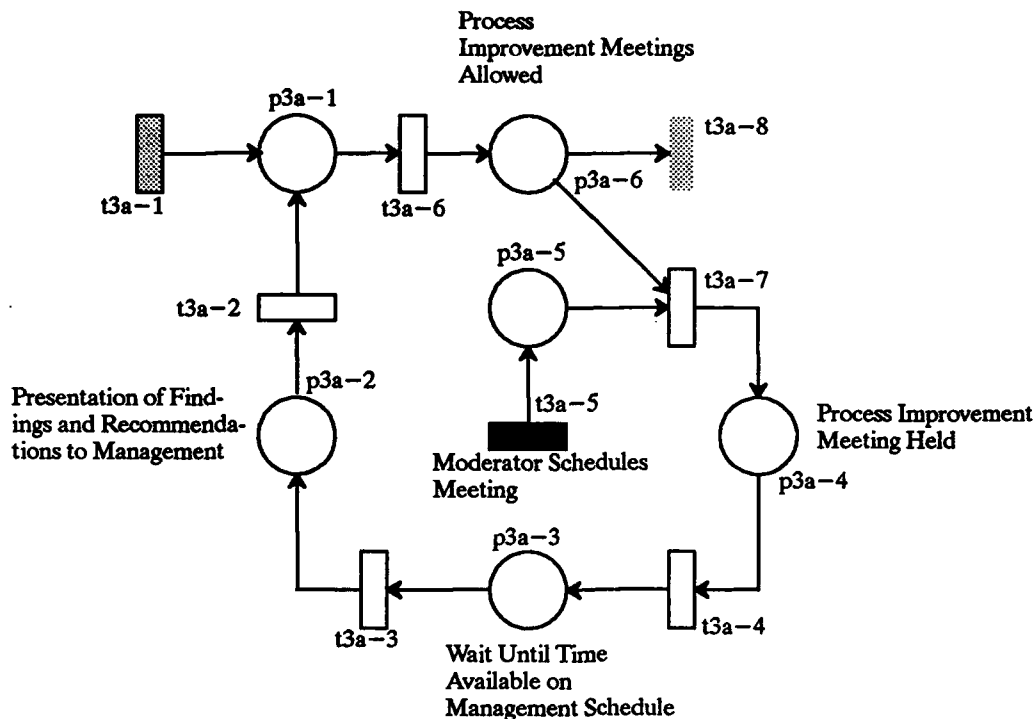
Figure 5-16. Petri-Type Net

Formally, a Petri net is a bipartite directed multigraph that includes an initial marking. The nets are comprised of two types of nodes: places and transitions that are connected by directed arcs. Arcs may connect either a place to a transition or a transition to a place. Graphically, transitions are usually depicted as bars, and places are depicted as circles.

The behavior of a Petri net is also simply described. A transition is said to be enabled if each input place to that transition carries at least as many tokens as are required by the "weight" of the connecting arc. If a transition is enabled, it may or may not fire (depending on the semantics of the model). However, if a transition does fire, the following two events occur. First, v tokens are removed from the input places (where v is the weight of the inbound connecting arc) and μ tokens are placed in each output place (where μ is the weight of the outbound connecting arc).

One simple extension to the Petri net theory includes the concept of a transition switch. When a switch is employed, only one of the output arcs fire after that transition is enabled. Consequently, instead of tokens appearing at all output places (as occurs with regular transitions) only one of the output places has a token appear. In the case of have only two output places, this behavior is exactly analogous to an "if, then, else" statement.

Other important extensions to basic Petri nets are the inclusion of timing and stochastic characteristics. Stochastic Petri nets are especially well suited for process modeling. These nets do not make the assumption that transitions are instantaneous (Henderson and Taylor 1991). They include, for instance, both an enabling time, and a firing time. This introduction of time almost invariably alters the execution characteristics of these nets, just as delays in real-world activities almost invariably alter the overall schedule governing some general process.

Petri nets, in their basic form, can easily capture the relationships between events from a coordination and execution perspective, and they do so entirely from the information contained in the entry and

exit conditions. More complex nets can be constructed that model availability of resources (represented by a token) and the nonviolation of constraints (represented by the **absence** of a token). However, the templates do not automatically request timing information, so if the eventual goal was, for example, timed stochastic Petri nets, then timing characteristics would need to be added to the information collected on the templates (and the easiest way to accommodate this is for the process engineer to modify the Foundation template and add a field or two that captures timing behavior).

Petri nets are useful for process simulation and dynamic process analysis. If process execution issues are critical, Petri nets can provide what might otherwise be obscure or limited insights into dynamic behavioral characteristics (i.e., the notation itself is executable). Since Petri nets are capable of low level models, such detailed models can be more labor intensive than higher level models using other PNs. In software engineering, many of the processes being defined are still ad hoc and volatile. For some organizations, it may be best to defer constructing detailed Petri net representations, because without automated tool support such representations may be difficult to update and maintain.

As is summarized in Figure 5-17, the primary strength of Petri nets is their ability to capture highly constrained (and especially, temporally constrained) information about a flow of events. Additionally, through a variety of techniques, Petri nets can be decomposed (sometimes called "unfolding") or recomposed ("folding"). Depending on the model being constructed, subnets might be developed that represent specialized forms of high-level Petri nets. For this reason, the following figure shows that both events and the constraints placed on those events can be involved in specialization/generalization relationships.
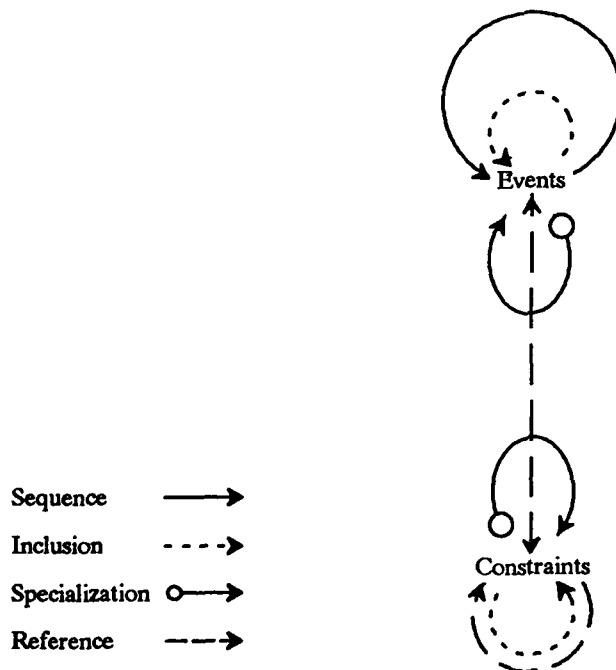
Figure 5-17. Petri Net Relations

5.2.6 PROCESS AND ARTIFACT STATE TRANSITION ABSTRACTION

The PASTA notation is designed to allow the process engineer to define a model for any development method or process that has defined artifacts and defined composition and dependency relations

among the artifacts. Defined artifacts and relations allow for explicit artifact states upon which process states and the overall process depend. Defined artifacts allow specification of the artifacts that record software development. Artifacts and relations support analysis of product completeness. For artifacts whose completeness cannot be formally specified, the line engineer can use technical reviews and issue tracking to specify completeness.

5.2.6.1 Structure of Formal Generic Process Model

Figure 5-18 shows all of the key terms used in defining a process model. Arrows depict the relationships used in defining the design model. In the bulleted list below, each term represents one relation between two artifacts.

- *Refer-to.* Analysis on the software process always refers to an artifact, A-state, and P-state.

- *Composed-of.* A P-state is composed of the operations that the technologist can currently perform and a role is composed of the activities that the role is able to perform.

- *Change.* An operation changes (promotes or demotes) an A-state.

- *Manipulate.* Operations manipulate artifacts.
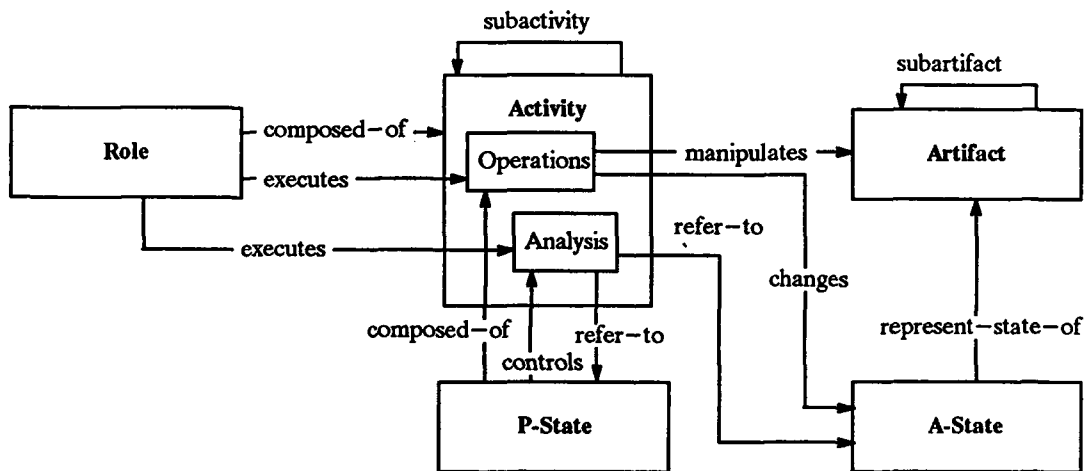
Figure 5-18. Relationships Defining the Design Model

5.2.6.2 Artifacts and Their States

Artifacts capture the decisions made during the software development process. Examples of artifacts are a description of the decomposition of a design into a set of components (such as modules, objects, packages, or subroutines) and a specification of the interface of one of the components. To characterize the state of a software development process, the engineer must characterize the state of the artifacts produced during the software process, e.g., whether or not the software decomposition is complete. However, merely characterizing the state of the artifacts is insufficient to describe a complete software process. The process modeler must also describe the activities that may be performed on artifacts, the conditions under which those activities are performed, and the roles of the people who may perform them.

5.2.6.3 Process State

The PASTA model has two levels. The lower level is based on the states of the artifacts produced during the software process; such states are called artifact states (A-states). A-states alone are insufficient to completely describe the software process, descriptions of activities, and operations on artifacts. The augmented states in the upper level state model are called process states (P-states). The two-level model allows the separation of the process description from the representation used for the artifacts. Whether an activity is performable or not depends on the state of the artifacts. At any point in time, the set of performable activities represent the choice of artifacts on which the line engineer may work. Those that are not performable represent the artifacts on which he may not work. The model prescribes a permissive ordering of activities (and on the work of the developer) by specifying which activities are performable and which are not performable at any point. Permissive means that the line engineer can choose from among the set of performable activities those to pursue. As with artifacts and activities, different processes will specify different roles, such as designer, reviewer, programmer, and manager. The logical grouping of activities in which people may participate defines the roles.

A P-state is represented as a rectangular box with two types, and only two types, of boxes that can intersect with it. Figure 5-19 shows an example. The two types of boxes are the entrance, and exit conditions. The difference is that the entrance condition has a P-state name subbox on the bottom of the box, but the exit condition box does not. The P-state name can be on any edge of the P-state boxes. This gives the process engineer a lot of geometric and topological freedom to draw. Figure 5-20 is an example of an artifact relation diagram. The process modeler can draw an arrow from an artifact class to an instance of an artifact. The name of the artifact has a dot in it to connect the class name and the name of the instance.

5.2.6.4 Translating From Process Templates to Process and Artifact State Transition Abstraction

The state information in the templates is certainly usable by PASTA, especially since different state-sets exist for events (processes) and throughputs (artifacts). A corporation can decide their formal migration path in different ways, e.g., from process templates to ETVX to PASTA or from the templates to PASTA directly. All of the process and activity templates can be translated to P-states. However, if some of the activity and task templates are supported by computer-assisted software engineering (CASE) tools, these activities and tasks will be mapped to operations in PASTA. Both the template and PASTA have role elements so the mapping is direct. A role in PASTA is defined as a collection of activities. The process engineer will need to collect the set of activities for the role. All of the product, research, and resource templates will be translated to artifacts in PASTA. The cross-reference will be used extensively to translate the collection of related information which is mapped differently to PASTA. This is especially true for the difficult translation of constraints, which do not exist in PASTA. In this case, a new artifact needs to be created for supporting constraints (e.g., checklists and signatures), defined for the artifact, and then referenced in the pre- or post-conditions for either P-states or Operations. Finally, all of the super-sub relationships captured in the templates can be translated to PASTA.

As is summarized in Figure 5-21, the primary strength of the PASTA approach is its ability to capture the relationships between events and products (throughputs) in a highly constrained manner. Note that PASTA models show flow of events as a function of state changes in underlying throughputs. Hence, it is the state of the throughputs that defines the state of the event and not the converse. Finally, you should note that PASTA presents what is fundamentally a two-level process model: the P-state level (events) and the A-state level (artifacts or throughputs).
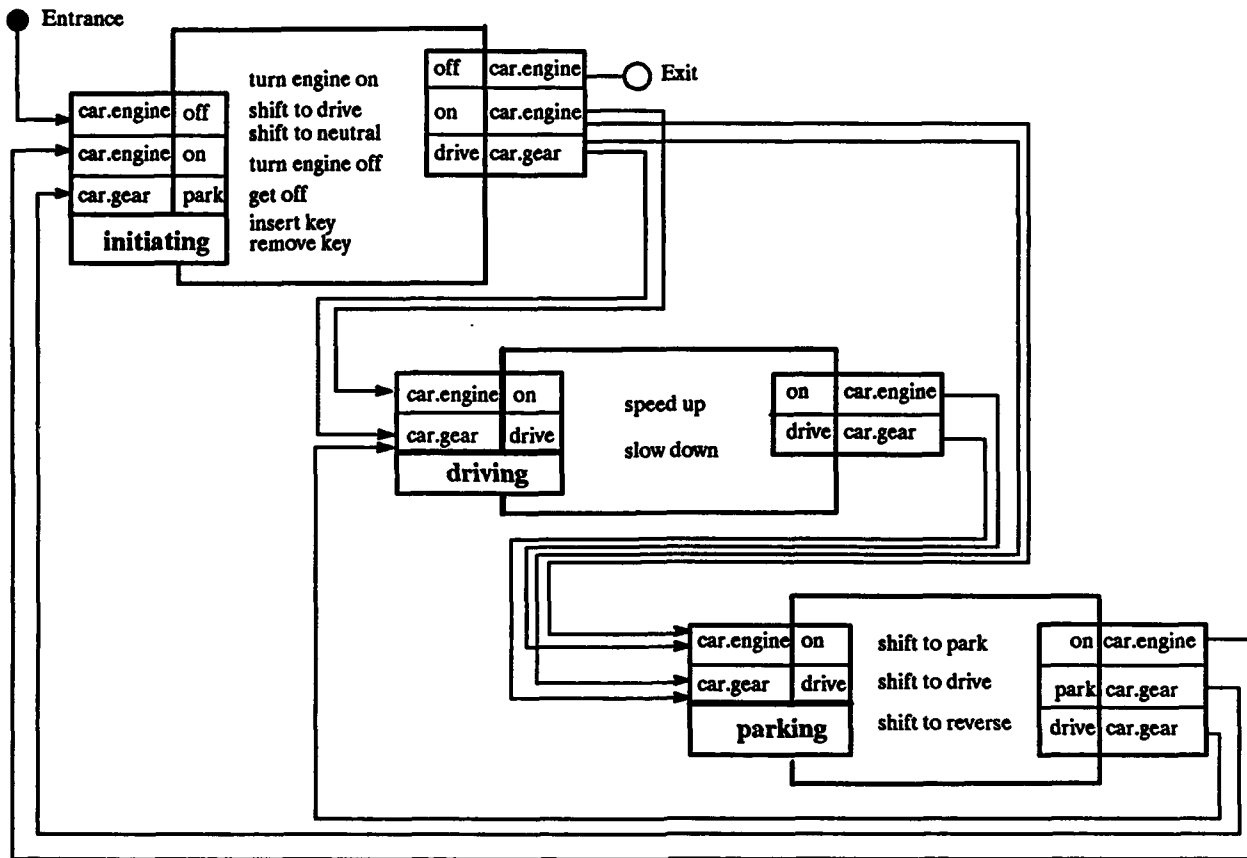
Figure 5-19. P-State Diagram Example
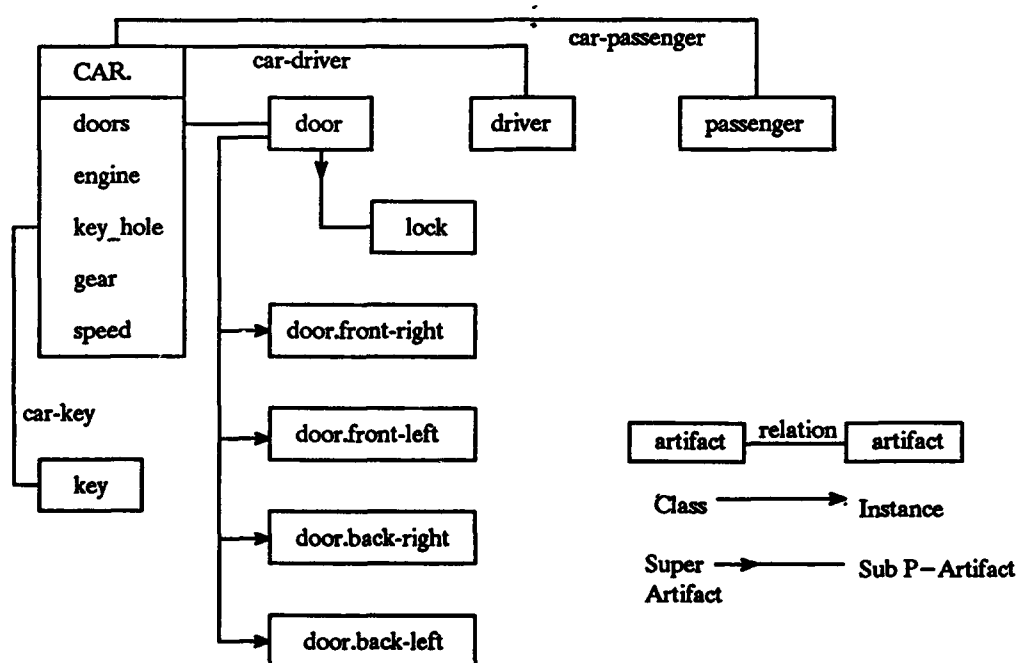
Figure 5-20. Artifact Relation Diagram Example
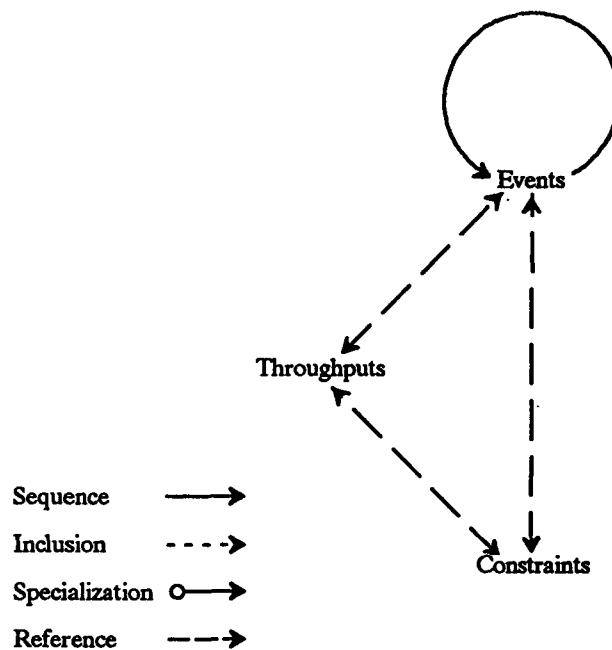
Figure 5-21. Process and Artifact State Transition Abstraction

5.2.7 ROLE INTERACTION NETS

An RIN is a visual formalism for the design, specification, and enactment of work processes that consist of activities ranging from informal to formal and from semi-automatic to fully automatic (Singh and Rein 1992). The RIN formalism describes processes as collections of organizational roles and their respective interactions. Role instances and their bindings to specific individuals are part of process instantiation.

A process is viewed as a collection of roles that communicate and interact to accomplish their goals. Each role in a process consists of a partially ordered set of tasks. Tasks are either solitary actions performed without the involvement of the other roles or joint actions performed with other roles. Joint tasks can be as simple as delivering a document or as complex as negotiating a contract.

An example of the visual formalism is seen in Figure 5-22. It depicts a simple process describing: the development of a document. The author, reviewer, and the repository form the set of interacting roles. Each role is represented by a column. Tasks appear as boxes. In the simple case shown here, tasks appear in the order of their execution: from top to bottom. Tasks requiring manual action are drawn as ⊠. Interactions between the roles are signified by connecting their respective task with a horizontal line. The interaction between roles often involves the transfer of an artifact, such as a document or report. In the notation, a transfer is indicated by curved flow lines from the source ⟋ ⟍ to its destination ⟋ ⟍ .

The complete notation includes many more features than are shown in this simple example. The order of tasks within a role form a partial order and not the simple linear order implied by the example. Alternation and iteration of tasks can be specified. Further, tasks can themselves be a process, thus providing a powerful abstraction/decomposition mechanism.
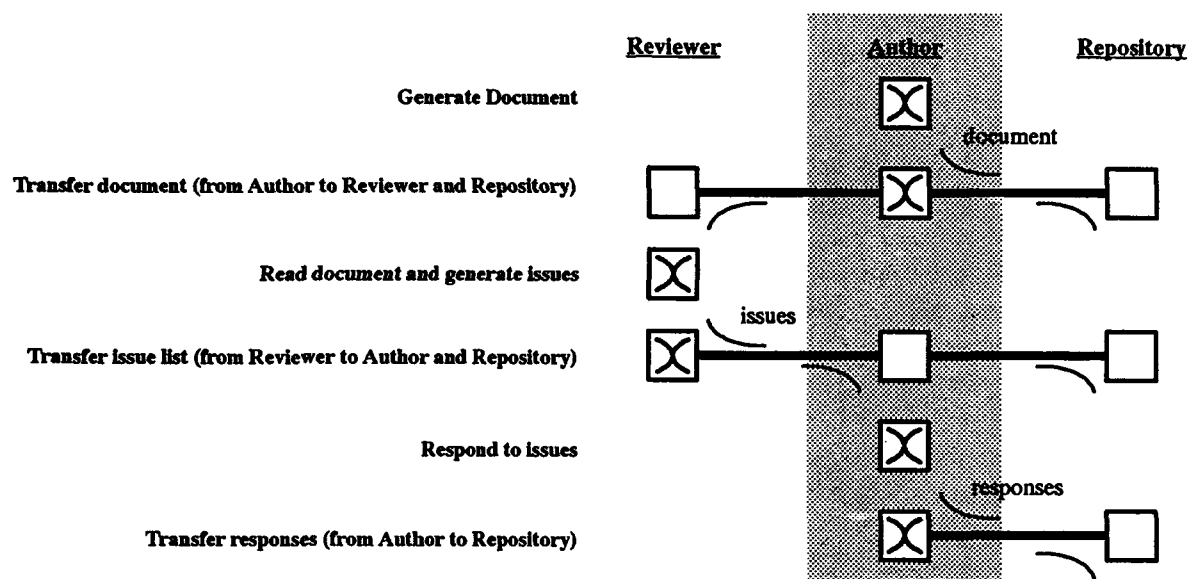
Figure 5-22. Role Interaction Net

Tasks have a number of attributes reminiscent of the ETVX formalism. These attributes serve as both description and additional support for process enactment. Figure 5-23 shows a task with its attributes expanded. The goal is a descriptive title for the task, e.g. "Generate Document." The objects attribute lists the artifacts (inputs and outputs) necessary for performing the task. The entry and exit conditions are the necessary conditions to begin and terminate a task. In addition to these conditions, the role occupant is required to initiate and terminate all manual tasks. The execution condition defines an invariant which must be true throughout a tasks execution. Failure of this condition causes the task to abort. The behavior describes the actions to be performed by the task. For automatic processes, this behavior is written in some machine-interpretable script language. Measurements define the metrics collected by the task. Finally, the permissions attribute defines the degree to which a role occupant can modify the task.
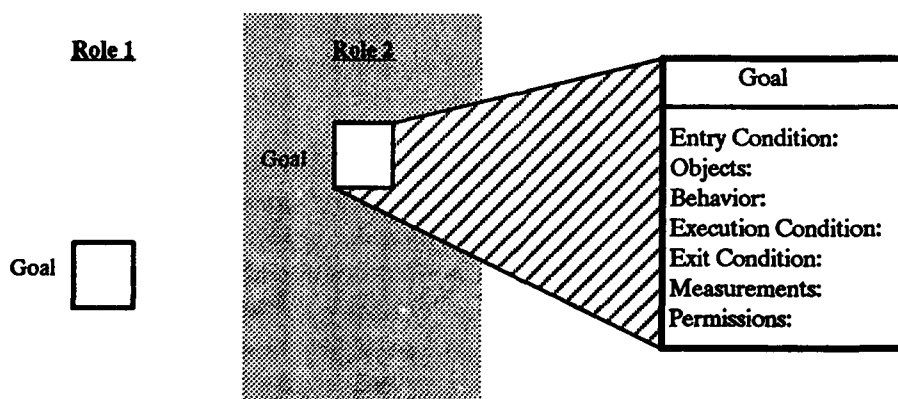
Figure 5-23. Role Interaction Net Templates

The RIN notation is supported by formally defined operational semantics. Both Singh and Rein (1992) and Singh (1992) provide formal semantics, in terms of Petri nets and STDs. These publications also provide further details and examples of the notation.

RINs seem to have a number of features that make it a desirable process modeling notation:

- Appealing visual notation

- Powerful abstraction/decomposition mechanism

- Well defined operational semantics

However, RINs are a relatively new notation and suffer from:

- Lack of industry wide exposure.

- Limited real-world application.

- Few supporting tools.

From the perspective of the objects and relationships defined within the templates (Figure 5-24), RINs capture all four of the meta-classes: events, throughputs, supports and constraints. Although inheritance is not an explicit part of the model, decomposition (or inclusion relation) is strongly supported through role decomposition.
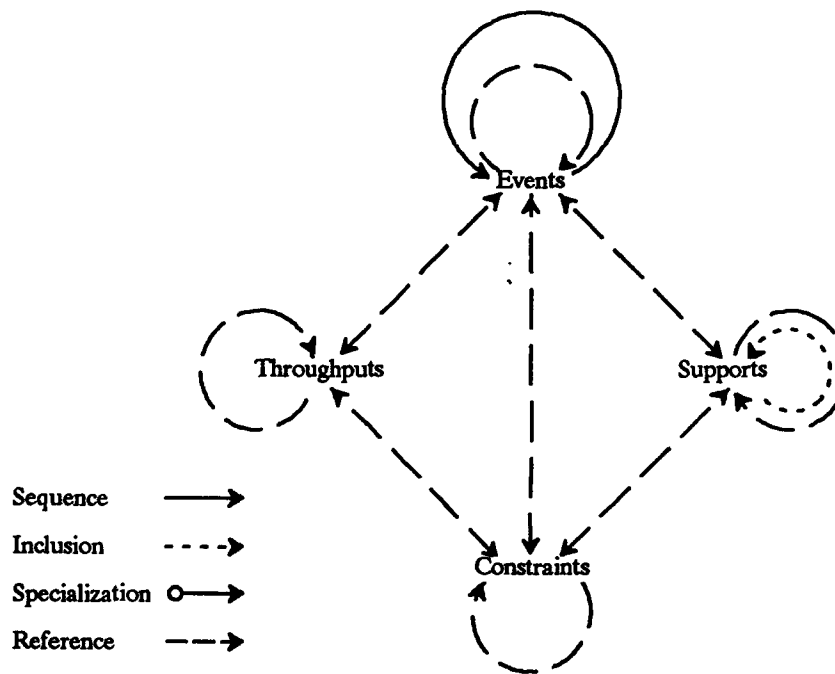
Figure 5-24. Role Interaction Net Relations

5.3 SUMMARY

This section has described a small but representative selection of other notations that may be used in the performance of process definition and modeling. As stated previously, template-based models can be used as a preliminary effort before constructing process models based on one or more of the other PNs, or the templates may be used to extend or augment information that has already been captured using an alternative notation.

It should again be stressed that there is no approach that is universally better or universally worse for all organizations under all circumstances. Instead, the choice of a notation is a highly site-specific choice and should involve evaluation and consideration of issues discussed in both this section and Section 4.3 (representative power), Section 2.4 (common notational characteristics), Section 2.5 (choosing a notation), and Section 2.6 (benefits to the template-based approach).

Finally, if you remain uncertain on which notation to use for process definition and modeling, the Consortium recommends you start with the templates and supporting techniques proposed in this guidebook. The primary advantage of this approach is that it is designed to allow you to adapt, tailor, and even fundamentally alter the set of templates (and supporting graphical notation) to fit the specific needs of your environment. Once you are using a relatively stable set of templates, you can examine the templates and their characteristics and compare and contrast them with the characteristics of alternative notations (such as those presented in this section). As related in Section 2, the "better" notation will be, of course, the one that most completely matches your definition and modeling requirements.

6. PROCESS REPRESENTATION PROGRAMS

The material in this section (Figure 6-1) discusses issues, approaches, and options relevant to commencing and continuing a program for process representation. This discussion examines how you can transition your organization or group into process definition and modeling, and then it briefly discusses where such efforts can eventually lead.

Section 6.1 presents material on introducing process definition and modeling into your organization. Section 6.2 introduces the subject of metrics and discusses how process definition can support a program of process and product measurement. Example metrics are shown, and discussion is provided correlating various metrics to different levels of SEI process maturity. Section 6.3 provides an overview description of the Evolutionary Spiral Process from the perspective of process representation. Section 6.4 discusses the use of process representation as a tool for process management, and Section 6.5 looks beyond conventional process management approaches and discusses the support process representation gives to automated process management within highly integrated environments. Section 6.6 briefly summarizes this material.
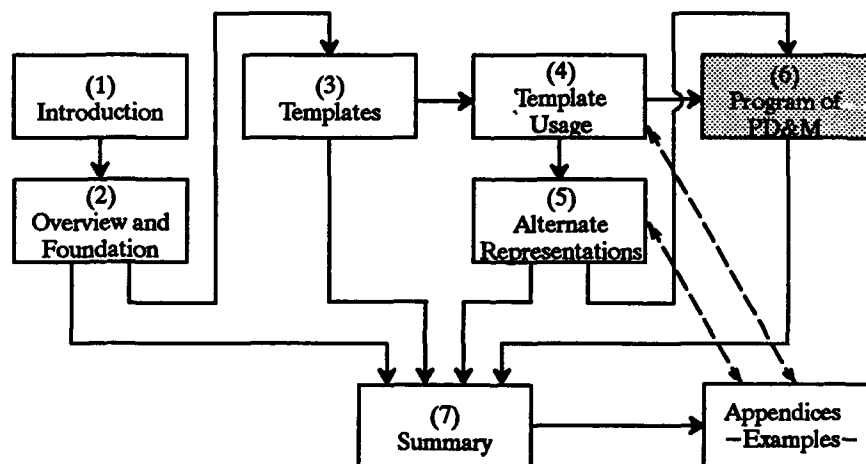
Figure 6-1. Guidebook Organization View 6

6.1 INTRODUCING PROCESS DEFINITION AND MODELING INTO AN ORGANIZATION

Institutionalizing process definition involves a variety of factors. First you need to be motivated. Why should you attempt to develop process models? What are the costs? What are the benefits? Second, there are logistical aspects. Who should perform in what roles? Third, there are practical issues. How does one get started? If the initial steps yield promising results, what should be attempted next? These are the issues examined in the following material.

6.1.1 MOTIVATION

As stated in Section 1, the techniques described in this guidebook can be used:

- As a common foundation for process analysis, design, development, and documentation.

- To facilitate the development of process-oriented guidebooks.

- To improve the usability of process-oriented guidebooks.

- To facilitate process training and education.

- To reduce the cost of developing process-oriented guidebooks.

- To create a defined process from a repeatable process.

- To create a repeatable process from the analysis of independently successful process activities.

- To facilitate process management.

- To facilitate process measurement.

- To facilitate process improvement.

- To facilitate process automation.

- To design and develop process models.

- As a migration path to and from process models based on existing notations (SADT, ETVX, STDs, etc.).

Once instituted, process definition and modeling provides the following benefits to its users:

- It can be used to develop precise, unambiguous process descriptions.

- It provides a basis for building and integrating tools that automate parts of the process.

- It facilitates enforcing standard practices.

- It enables all concerned parties (technologists, developers, managers) to agree on a standardized, documented process.

- It provides a basis for process improvement through improved process analysis and potentially faster *process evolution*.

- It enables management to potentially improve process efficiency by facilitating the establishment and collection of process metrics and related status information.

All of the above can be summarized in the following statement: the key advantage to process definition and modeling is increased clarity, increased understanding, increased control, and increased flexibility with regard to processes being practiced or considered. Clarity is increased because models are

intended to abstract out unimportant information (with regard to a specific objective) in order to better emphasize or highlight pertinent information. Understanding not only increases with clarity but also because process details are collected and organized in a way that allows ready access, study, and modification. Control is increased through instantiating process models to derive project-level process models which offer sufficient detail for managers to track a project's evolution (as guided by the overall process). Finally, flexibility is increased because an organization can experiment with alternative process models, as opposed to alternative processes.

Through the use of models, tentative or proposed process changes can be explored through constructing and examining alternative models instead of dedicating real resources and funds toward some type of pilot project to evaluate proposed process changes. While the model cannot provide the same types of insight a pilot project can, it is certainly true that any process problems corrected through examining the model are less costly than correcting those problems during actual performance of the process.

6.1.2 PROCESS DEFINITION AND MODELING STAFF

When introducing or expanding a process definition and modeling effort you must determine which roles are necessary for supporting the work and what types of skills are desirable for people performing those roles. Arguably, three to five matrixed engineers and one matrixed manager are necessary for establishing a process representation (definition and modeling) program. Because some training is involved, there is too much risk in training only one person and then having the entire program depend on the efforts and availability of that person. In principle, it is better to have more people contributing smaller amounts of their time than to have fewer people contributing significant amounts of time. Once a program is well underway, it is ideal to have one or more people work their way into essentially full-time responsibilities as "process engineers."

The preferred candidates for such work are those that already have a process-oriented background. This group includes people who have had experience in systems analysis and design. These individuals have already had considerable exposure to *process control* at a systemic level. Also included are those individuals who may have received exposure to systemic process issues through involvement with total quality management, process assessment efforts, and similar activities.

It should be noted that because the templates and the techniques for their use are sufficiently easy, the learning curve involved in becoming proficient in their application is comparatively nominal. From this perspective, the benefits of having several people trained and potentially available (as matrixed resources) for process representation efforts is relatively low-cost and low-risk. This is especially true when compared to the value derived from the increased likelihood of greater variety of insights and experience.

As discussed in Section 5, the templates can be used as a point of departure for alternative process representation notations such as ETVX, SADT, and Petri Nets. If such notations are one of the eventual goals of the process representation effort, it is highly advantageous to select process engineers partially as a function of whether they have a background in the alternative notation of choice. In all cases, using alternative notations for constructing process models is consistent with their use in constructing software models. Consequently, notation-specific skills previously acquired by process engineers can be applied to process representations using those notations.

6.1.3 GETTING STARTED

Once you have identified who will be working as process engineers, the next step is to identify the first target domain to be represented. The Consortium suggests that process definition and modeling, as

an organizational process, be the first representation constructed. Then performing subsequent process representation effort, the process engineers will be following the model they constructed during their first effort.

Before actually performing process representation, you must train the process engineers (and if possible, those who will be managing the efforts). Again, the templates and their use is no complex; therefore, training can be limited to two days or less. However, this depends on the number of related topics introduced during training (such as metrics, process maturity levels, and alternative process notations).

There are several example approaches in this guidebook; when you first commence a program of process representation, you can use these example approaches as suggested guidelines. There is a usage scenario presented in Section 4, and further examples exist in the appendices. As indicated in these examples, the suggested approach is that you initially construct indented lists, then construct graphical depictions, then fill in the templates using information derived from the graphical model and the indented lists, and finally, establish bindings or relationships between the templates.

Once you have a sufficiently complete process representation, it can be analyzed for areas of risk and for opportunities of improvement. Often, simply having an entire process detailed in an easily accessible and understandable format allows you to think about and discover risks and opportunities. A system of metrics can be constructed to ease model analysis by using more objective criteria. For example, as discussed in Section 4, the templates can be examined with regard to the average number of entry criteria and average number of exit criteria specified for events. Then, events whose criteria significantly exceed the average can be re-evaluated to determine whether such excessive binding to other events constitutes unusually high risk.

The eventual goal of analyzing models for risks and opportunities is to derive new and improved models. The new models can then be carefully analyzed for consistency, completeness, and potential problems. When management is confident that the proposed process is well understood and has sufficient potential benefits, they may decide to institute an exploratory or pilot project based upon the proposed process.

6.1.4 ONGOING IMPROVEMENTS IN PROCESS REPRESENTATION

Once you have used the templates to construct one or more process models, you will have insight into how the templates can be improved and explicitly tailored toward your needs and the nature of the models being constructed. As discussed in Section 4, you can add new fields, develop new classes of templates, include new meta-classes, and introduce even further levels of abstraction. In all cases, it is important to monitor the benefits when contrasted against the additional effort and complexity introduced by such extensions.

Limited space is one of the principal problems with hard-copy renditions of the templates. Although the one-page format presented in this guidebook makes presentation, discussion, and initial use of the templates easier, the single page format quickly becomes short of space when attempting to model large-scale complex processes. One suggested approach to alleviate this problem is to represent each template as a three page packet. The top page would contain foundation fields only and would be identical in format on all packets. The second page would contain meta-class fields only. The third page could contain only those fields unique to that specific class. This approach both preserves the advantages of the current template's composition and considerably expands the space available for use.

A better, if more ambitious approach, would be to automate support for process representation and to work with electronic versions of the templates. This not only allows for virtually unlimited field sizes

(constrained only by limitations in a supporting database, for instance), it also introduces the possibility of active links between the templates. Relatively trivial automated support, such as through hypertext, allows you to easily follow relationships between templates. This greatly improves the ease with which templates can be developed, analyzed, and maintained.

Once you are familiar with the development and use of process representations within your organization, you can look for ways in which process modeling can support other organizational endeavors, and vice versa. One major candidate for potential mutual support is with an organizational metrics and measures. Although many organizations have established measurement programs without using process representations or models, you should note that these programs can readily facilitate each other. Hence, you may want to tailor your templates to directly support a program of process measurement. As discussed in the next section, metrics are a excellent means by which insights are gained into dynamic process characteristics.

6.2 METRICS

The purpose of metrics is to increase objectivity by providing a common measure with which to compare different phenomena or the same phenomenon at different times. However, in addition to the considerable benefit derived from using metrics, there is also the risk of mismeasurement, or misinterpretation. The following discussion covers important characteristics of metrics: both from the perspective of benefits and from the perspective of associated risk. A set of inspection-related example metrics are shown as an example. Material at the end of this subsection shows information extracted from the Consortium's *Software Measurement Guidebook* (Version 2.0) (Software Productivity Consortium 1992b).

6.2.1 IMPORTANT CHARACTERISTICS

There are many important aspects to metrics, but four of the most important are:

- Metrics must correlate to the phenomenon being measured.

- Metrics must measure something of interest.

- Metrics must reflect something that can be influenced.

- Metrics must support making predictions.

From one perspective, metrics are intended to facilitate insight into some phenomenon in the real world. The intent of the metric is to change in some manner that is consistent with changes in the phenomenon being measured. If a metric does not correlate to what it should measure, it is useless. The more closely changes in the value of a measurement parallel change in the characteristics being measured, the greater the usefulness of that metric.

In addition to correlations, metrics must also measure something that is interesting. Interesting is invariably subjective, but the concept is crucial nevertheless. It is too easy to define a large collection of metrics and spend disproportionate amounts of time and effort measuring characteristics that yield little, if any, valuable insight. From this perspective, interesting metrics are those that provide you with insights that affect your decisions or your actions. Clearly, when you find that you largely ignore a particular metric and it does not influence your thinking, that metric has ceased to be interesting for you.

Another key characteristic of metrics is that the measure relates to something that can be influenced. There is little value gained by measuring something that has no relation to anything that can be influenced. Curiosity may be satisfied, but if the metric monitors something completely outside your sphere of control and if nothing related to that metric can be influenced by you, the metric does not allow adjusting a process to achieve changes in the data reflected by the metric. If the metric can in no way provide you with any useful guidance on how to alter the phenomenon being measured, or your actions with respect to that phenomenon, then the metric is essentially meaningless.

Finally, it should be stressed that one of the primary benefits to metrics is their use in validating predictions. For example, when proposing changes to a process model, you should attempt to predict what effect the changes will have on the metrics monitoring that process. When proposing a new or modified process, such proposals should be accompanied by claims to the effect that metric-a will go up by x units and metric-b will be reduced by about y percent, etc. One value to such predictions is that they more precisely communicate the expected benefit from instituting a particular change.

Of even greater value is the fact that this approach can contribute to verifying whether the actual results of the new process match the predicted results. If results are generally as predicted, there can be a higher degree of confidence that the predicted benefits will likewise be manifested. If changes in the metrics are running opposite to their predicted direction, it may be time to consider reinstituting the original process and re-examining the new process.

Please note, however, that it is not always the case that when the predictions are wrong, the process is wrong. It could be that the metrics are not measuring what they seem to be measuring or that there are confounding influences affecting the measurement that are not being considered in the interpretation. Simply put, the problem may not be what is being examined, but the way it is being examined. Fortunately, using metrics to make predictions is a self-correcting problem. Either the metrics are high-quality and our ability to make predictions improves, or the metrics are not high-quality and the lack of *predictability* causes you to re-examine and improve the metrics. In any event, you gain insight and make progress.

6.2.2 METRIC SUPPORT FOR PROCESS TRACKING AND COST MODELING

This material describes activity-based cost models and how you can apply them in starting your process modeling effort. To implement process models, you need to accumulate software process and product information. Many existing accounting systems maintain costs in terms of activity-based models.

This section presents methods now used to work with the software development processes in your organization. The methods express each major process in terms of an **activity.** The methods are used by management for **activity-based** cost estimation. **Activity-based models** use a bottom-up approach to software development cost estimation based on an analysis of the costs of the individual activities that compose the software development process. Activity-based models are especially effective in an environment in which you have established a software experience database and where you use that database to feed back information about the process to improve the process.

The cost estimation methods usually base estimates on a function of the software product size. In turn, the development schedule is estimated as a function of the cost estimates. The size, cost, and schedule parameters are closely related.

The methods presented in this section roughly correspond to a progression through the SEI process maturity levels. If your software development organization is at process maturity level 1, the initial

level, you probably have no experience data in a database. By the time your software development organization is at SEI process maturity level 2, the repeatable process level, you will have established a software experience database and accumulated sufficient data so that you can calculate unit costs for the main activities of software development (e.g., requirements definition, design, code and test, integration and test) and apply them on a project basis.

An activity-based model is built by assembling and ordering the activities that compose the development process to be used to produce the intended software product. The activities that form your development process may be from a previously used process, or they may come from a modified version of a previous process with some activities removed and other activities added. Alternatively, they may be a selected subset of activities from a "menu" of activities. An activity-based model enables you to begin by using the resource consumption (cost) for each of the activities in the development cycle, such as requirements analysis, preliminary and detailed design, code and unit test, computer software component (CSC) integration test, and computer software configuration item (CSCI) system test. Your project may not use all of the "repertoire" of possible activities. For example, if you are developing a new version of an existing system, you might not have any preliminary design in your development process. Define your software development cycle in terms of known and measurable activities based on your organization's experience as contained in your experience database.

You may use the activities in Table 6-1 as a menu from which to select activities, if appropriate, to form your development process. The ordering of the activities in Table 6-1 is a natural order for presentation but not necessarily the expected order of development.

Table 6-1. A Basic Activity-Based Development Model

"Process" or Activity	Subactivity/Product (DOD-STD-1521B and DOD-STD-2167A)	(LM/KSLOC)	
		New	Reused
Requirements analysis	System/segment design document	0.31	0.020
	Software development plan	0.13	0.010
	Preliminary software requirements specification	0.25	0.020
	Preliminary interface requirements specification	0.04	0.002
	Software requirements specification	0.39	0.030
	Interface requirements specification	0.04	0.002
Preliminary design	Software design document—preliminary design including design reviews	0.52	0.030
	Preliminary interface design document	0.07	0.005
	Software test plan	0.13	0.005
	CSC test requirements	0.01	0.000
Detailed design	Software design document—detailed design including design reviews	0.82	0.050
	Interface design document	0.07	0.005
	CSU test requirements and test cases	0.01	0.000
	CSC test cases	0.02	0.000
	Contents of CSU and CSC software development files	0.00	0.000
	Software test description—test cases	0.04	0.006

Table 6-1, continued

Coding and CSU testing	Implement source code including code inspections	1.48	0.070
	CSU test procedures	0.03	0.000
	CSU testing	0.73	0.050
	CSC test procedures	0.25	0.030
	Contents of CSU and CSC software development files	0.00	0.000
CSC integration and testing	CSC integration testing	0.74	0.200
	Software test description—formal test procedures	0.10	0.020
	Updated source code—error correction	0.10	0.010
	Contents of updated software development files	0.00	0.000
CSCI testing	CSCI testing including acceptance testing	0.73	0.150
	Software test report	0.01	0.000
	Updated source code—error correction	0.05	0.010

Table 6-1 contains activities from DOD-STD-2167A (Department of Defense 1988) and DOD-STD-1521B (Department of Defense 1985) and typical unit costs derived from actual experience in developing embedded software development software for development of 25 large (over 500 KSLOC [thousand source lines of code]) real-time command and control software systems. The unit costs shown in this table are for guidance.

The activity-based model assigns a unit cost to each activity and estimates costs in labor months (LM) or labor hours (LH) by multiplying the size of the software product (KSLOC or SLOC [source lines of code]) by the assigned unit cost (LM/KSLOC or LH/SLOC). The general form of the activity-based model is based on the operations of adding, modifying, reusing, and removing code. The general form of the model is:

$$TLM = \sum_{i=1}^{n}(LM/KSLOC)_{i,added} \cdot KSLOC_{added} + \sum_{i=1}^{n}(LM/KSLOC)_{i,modified} \cdot KSLOC_{modified}$$

$$+ \sum_{i=1}^{n}(LM/KSLOC)_{i,reused} \cdot KSLOC_{reused} + \sum_{i=1}^{n}(LM/KSLOC)_{i,removed} \cdot KSLOC_{removed}$$

where TLM indicates the total effort in LM for all n activities and $(LM/KSLOC)_{i,j}$ indicates a unit cost in LM/KSLOC for activity i and code category j.

You should view the unit costs in Table 6-1 as an example or as guidance based on specific experience in the aerospace industry with developing real-time command and control software. The same methodology would be used for estimating the costs of developing software for different industries. Your organization's experience coupled with your judgment as a cost estimator may lead to a modification of some or all of these unit costs—up or down. For example, in a situation where your organization wants to integrate many CSCs, you may want to increase the unit cost for CSC integration test to account for additional complexity. The same is true of the functional testing of the software system where you integrate many CSCIs in the CSCI test.

You should not view these unit costs as fixed quantities that must be applied with no modification. They represent **guidance** to the cost estimator and should be modified to represent the software

development environment in question. For example, if the unit cost for implementing source code as shown in Table 6-1 were to be modified, in the judgment of the estimator, to reflect the effects of a programming staff inexperienced in the language to be used, which would cost 20 percent more than the given unit cost, and of a strong management team, which would cost 10 percent less that the given unit cost, then the modified unit cost for the coding activity would be 1.48(1+.20−.10)=1.63 LM/KSLOC.

If a set of standard or guideline unit costs based on such a database, as suggested above, is not available, then you can use the Constructive Cost Model (COCOMO) detailed model, specifically the modern programming practices (MODP) effort multipliers shown in Boehm (1981). Using this scheme, you can modify the baseline unit cost for each activity from requirements through integration and test by a multiplier factor to adjust for the degree of adherence to modern programming practices. Boehm (1981) gives a set of factors to be used in this case. These factors can be used to develop worksheets to support cost calculations (see Table 6-2).

Table 6-2. Worksheet Cost Calculations for an Activity-Based Model

Project Name	EXAMPLE		Date	11/3/92
CSCI/Product Name	CSCI 19		Language	JOVIAL
Unit Cost Measurement (LM/KSLOC, LH/SLOC, etc.)			LM/KSLOC	
New Code Size	450	Reused Code Size		710
	New Code			
Activity	Base Unit Cost	Modifier	Estimator Unit Cost	Cost (LM)
Preliminary design (incl. documentation)	0.59	1.20	0.71	319.5
Detailed design (incl. documentation)	0.89	1.20	1.07	481.5
Code and CSU test	2.21	1.10	2.43	1,093.5
CSC integration test	0.74	1.00	0.74	333.0
Error correction (for CSC test)	0.41	1.00	0.41	184.5
CSCI test	0.73	1.00	0.73	328.5
Error correction (for CSCI test)	0.28	1.00	0.28	126.0
Total	5.85		6.37	2,866.5
	Reused Code			
Activity	Base Unit Cost	Modifier	Estimator Unit Cost	Cost (LM)
Preliminary design (incl. documentation)	0.035	1.00	0.035	24.8
Detailed design (incl. documentation)	0.055	1.00	0.055	39.0
Code and CSU test	0.120	0.00	0.000	0.0
CSC integration test	0.200	1.00	0.200	142.0
Error correction (for CSC test)	0.060	1.00	0.060	42.6
CSCI test	0.150	1.00	0.150	106.5
Error correction (for CSCI test)	0.040	1.00	0.040	28.4
Total	0.660		0.540	383.3
Overall Total LM				3,249.8

Unit costs must be periodically recalculated since they will change through time as more experience is recorded and as the development process is improved. Unit costs will likely decrease in the long term as the cumulative effects of your process improvement efforts change take effect.

Table 6-3 presents the unit costs of an Ada language development model (Cruickshank and Gaffney 1992) and compares it with the unit costs shown in Table 6-1. Industry experience shows that in general the costs of developing software using the Ada language is more costly than developing software using standard, more established languages. The higher cost for Ada is due to many factors including the increased functionality that Ada provides and industry inexperience with Ada (the latter of which, of course, decreases as Ada experience is accumulated within industry).

Table 6-3. Ada Development Model

Activity	Ada Model		Basic Activity-Based Model	
	LM/KSLOC	Percent	LM/KSLOC	Percent
Requirements analysis	0.74	7.4	1.16	16.4
Preliminary design	1.67	16.7	0.73	10.3
Detailed design	2.22	22.2	0.96	13.6
Code and unit test	2.22	22.2	2.49	35.2
CSC integration test	1.60	16.0	0.94	13.3
CSCI test	1.55	15.5	0.79	11.2
Total	10.00	100.0	7.07	100.0

Often, a CASE tool is seen as a potential approach for reducing project cost (Table 6-4). When considering the use of a CASE tool, you must determine (or estimate) its effect on specific activities in the software development process. This consideration should be tempered by a general recognition of the fact that some aspects of an activity are subject to automation while others can only be done by a person. For each activity, you should determine the impact of the CASE tool application on:

- The reduction in the unit cost of doing the activity.

- The additional cost, if any, on this and any other activity of applying the case tool.

- The inputs to and the outputs from the activity.

- The determination of how and when the activity is completed.

- The quality of the activity.

- The sequence of activities.

You must not only consider the potential impacts of the application of the CASE tool on each activity but also the possible interactions of the applications of several CASE tools. If investment (in the tools or the process) is the same for each application, you must recognize the possibility of a decreasing return on investment.

Table 6-4. Top-Down Cost Estimating Example

Program Cost-Driver	Proportion	Cost (LM)
Software development	0.30	2,500
Computer hardware development	0.00	0
Systems engineering	0.16	1,333
Test and evaluation	0.16	1,333
Manufacturing	0.00	0
Product support	0.06	500
CM, DM, QA	0.06	500
Program management	0.26	2,167
Total	1.00	8,333

Finally, you cannot assess the (potential) impacts of CASE tools without the detailed knowledge of the process that an activity-based model, backed up by an extensive experience database, provides. As an example, suppose that your development process is like the Ada model shown in Table 6-3; and suppose that you estimate that the application of the CASE tool results in a 30 percent reduction in the detailed design costs. Then the unit cost of detailed design will become $(2.22)(0.70)=1.55$ LM/KSLOC, all other activities being unaffected. This reduces the overall unit costs from 10.00 to 9.33 LM/KSLOC. Such calculations are impossible without a fundamental understanding of the activities that comprise your process, a history of metrics associated with those activities, and related products, resources, etc.

6.3 REPRESENTING THE EVOLUTIONARY SPIRAL PROCESS

A robust notation for process definition and modeling must be able to capably represent not only standard processes, but also emerging or progressive processes that reflect the latest research and experience from government, industry, and academia. The Consortium has developed and documented a new process approach based on the work of Barry Boehm. The following is excerpted material from the *Evolutionary Spiral Process Guidebook* and technical report, and has been augmented to include discussion on building process models to support ESP. For further information and details on the benefits and usage of ESP, please refer to the Consortium's *Evolutionary Spiral Process Guidebook* (Software Productivity Consortium 1991) and the *Process Engineering With the Evolutionary Spiral Process Model* (Software Productivity Consortium 1992a).

The ESP model conveys that particular patterns of activities can be beneficial in software process and project management. One key distinguishing characteristic to this approach is the importance given to ongoing risk management. In the most general sense, the sequence of activities are classified through four quadrants (modified from the Spiral Model by Boehm [1986]), as shown in Figure 6-2. In addition to the four quadrants, the ESP model identifies and defines key activities essential for successfully using the ESP approach, also shown in Figure 6-2. Most of the activities explicitly included in the ESP model are the key management, review, and commitment activities missing in traditional process models, such as the Waterfall Process Model.
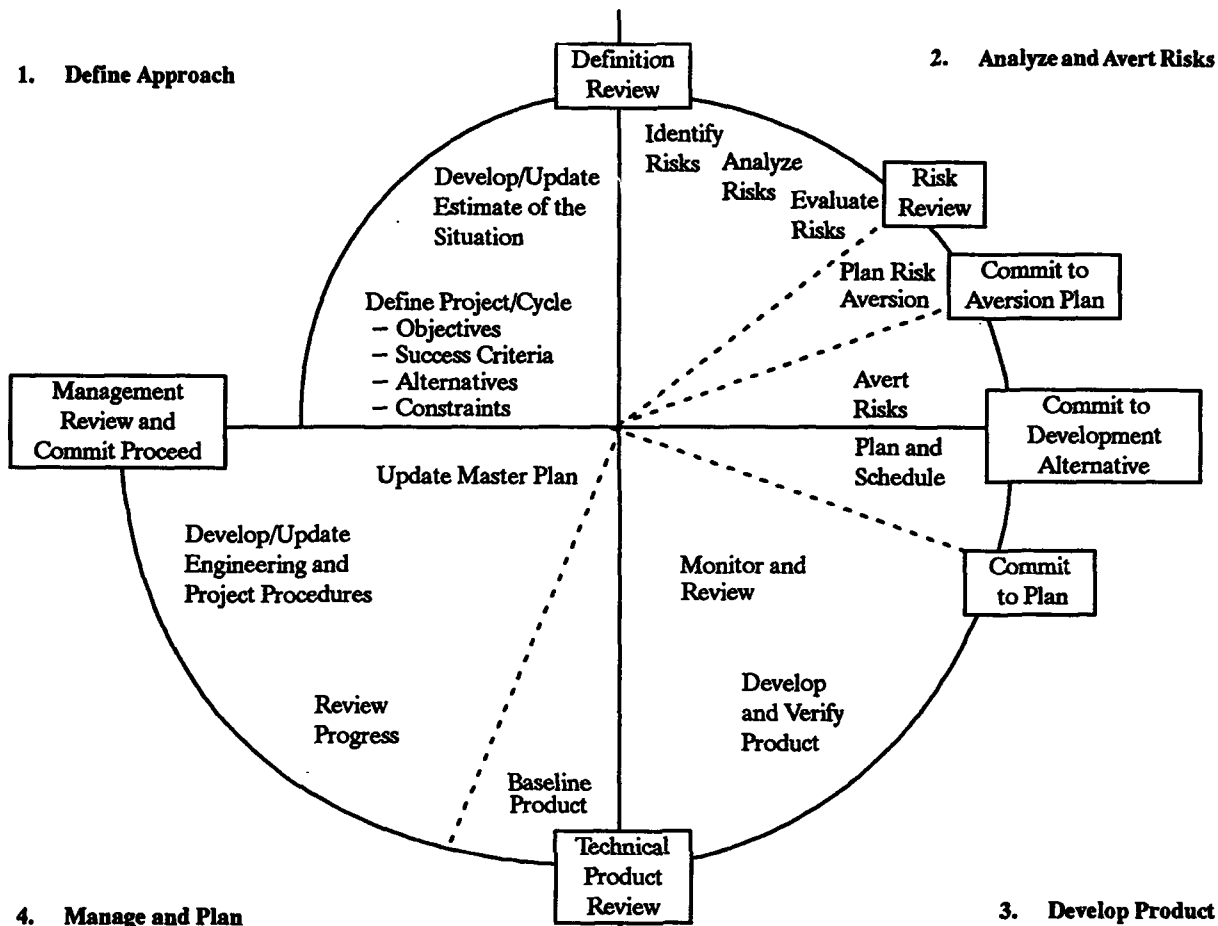
Figure 6-2. The Evolutionary Spiral Process Model

When constructing a model of an ESP-based process, it is important to note that a project traverses all four quadrants of the ESP model a cycle at a time. A cycle is a complete rotation of all four quadrants of the spiral model which, when completed, matures the product by the amount defined in cycle-specific objectives and success criteria. A spiral is one or more cycles which, when combined, create a specific accomplishment, such as a deliverable or key artifact. An example spiral consisting of four possible cycles is shown in Figure 6-3.

The amount of resources dedicated to a particular activity can, and typically will, vary from one cycle to the next. Additionally, in virtually all cases the completion level of throughputs (or products) will increase with each successive cycle. Typically, early cycles concentrate on creation of a development plan and risk aversion activities. Later cycles concentrate on product development activities. Template-based process representations can capture this cyclic evolution by having multiple sets of templates—each set dedicated to a given cycle. This facilitates not only indicating changes in levels and types of support, but it can also clearly distinguish between throughputs or products at different stages of completion. Alternatively, a single series of event templates can be used to represent the cyclic activity, and iteration can be captured through the use of state-sensitive throughput descriptions and logical relations.

1. Define Approach

2. Analyze and Avert Risk

Definition
Review

Identify, Analyze,
Evaluate Cycle Risks

Cycle Estimate of the Situation

Identify, Analyze,
Evaluate Project
Risks

Plan Cycle Risk Aversion

Project Estimate
of the Situation

Plan Project
Risk Aversion

Prototyping
Modeling

Management
Review and
Commit to
Proceed

4   3   2   1

Simulation

Commit to
Development
Alternative

Cycle Planning   *Benchmarks*
and Scheduling   *Other*

Update SDP, Budgets, EoS

Update SDP, Budgets, EoS,
and other planning documents

Develop Engineering
and Project Procedures

Project
Plan

Update Engineering and Project Procedures

Software
Requirements

Detailed
Design

Preliminary
Design

Review Progress (Actuals versus Estimates,
Objectives, Success Criteria, Alternatives,
Constraints)

Code and
Unit Test

B
A
S
E
L
I
N
E
P
R
O
D
U
C
T

Software
Integration
and Test

Acceptance
Testing

Formal
Reviews

4. Manage and Plan

3. Develop Product

Figure 6-3. An Example Spiral

The ESP model specifically recognizes the first cycle of any spiral as a project-level planning cycle; that is, defining the high-level project process definition, and documenting high-level objectives, success criteria, alternatives, constraints, risks, plans, and oversight strategies for the entire project. Subsequent cycles focus specifically on the work to be accomplished for that particular cycle or the detailed process definition in terms of the current cycle objectives, success criteria, alternatives, constraints, risks, plans, and oversight strategies. Knowledge gained as a result of working through a current cycle is used to review and update the project plans and process definitions initially developed in the first cycle.

The following subsections briefly discuss how a project traverses each of the four quadrants and performs the activities within each quadrant, and presents considerations for using the templates within each of these quadrants.

6.3.1 THE FIRST QUADRANT: DEFINE APPROACH

A project's approach definition establishes the ground rules for measuring progress at a high-level for the project or at a more detailed level for the current cycle.

During the first cycle, the project develops an *Estimate of the Situation* (EoS) which documents project level process drivers, including objectives, success criteria, alternatives, constraints, internal and external factors, and other information. In an environment which has predefined process architectures, and predefined process assets, these resources would be included in the EoS. The definition review at the end of the first quadrant authenticates and establishes consensus on the project level information contained within the EoS. During subsequent cycles, the project updates the EoS to include cycle-specific objectives, success criteria, alternatives, constraints, internal and external factors, and other information which will drive the process for that specific cycle. The definition review serves to authenticate and establish consensus on the cycle specific information.

The specific concepts and activities of the first quadrant are graphically depicted (using the SADT notation described in Section 5) in Figure 6-4.
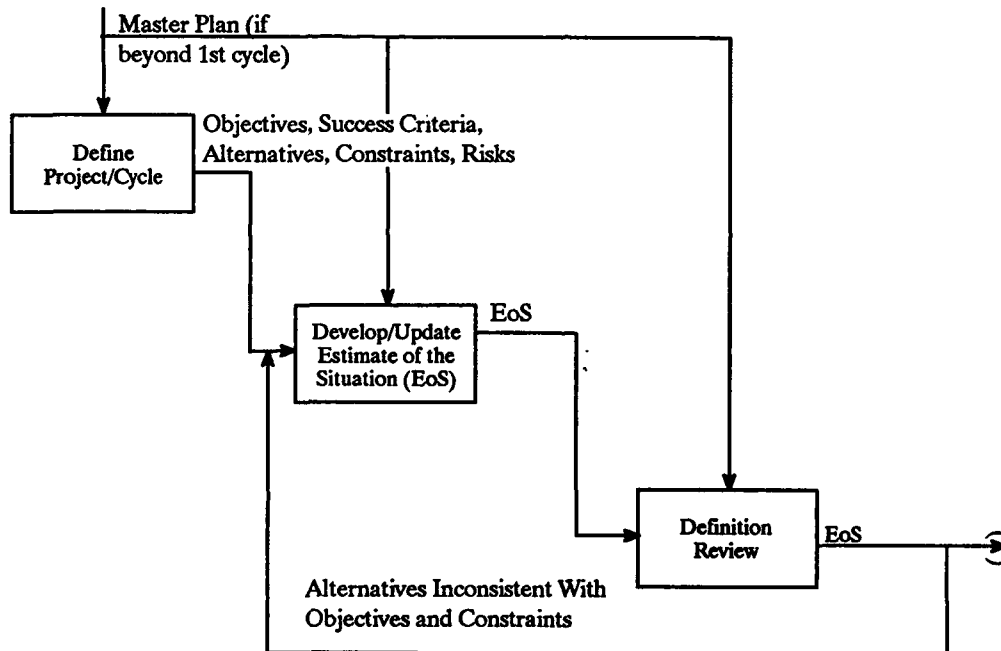
Figure 6-4. Define Approach Activities

6.3.2 THE SECOND QUADRANT: ANALYZE AND AVERT RISK

A project performs risk analysis and aversion to help "identify, address, and eliminate risk items before they become either threats to successful software operation or major sources of rework" (Boehm 1989). Risks are the potential undesired outcomes or missed opportunities which affect the project's defined objectives and success criteria. Risk analysis and aversion is a defensive management approach that focuses on project uncertainty and attempts to control project outcome.

During the second quadrant, a project concentrates on the factors, or risks, that may oppose project or cycle success. Most software development projects inherit or introduce some degree of risk. A project

specifically addresses these risks in the second quadrant of the ESP model by evaluating the alternatives defined in the first cycle in terms of the project constraints. The primary result of this evaluation is a quantified list of risks for the project and/or current cycle. Risks can be quantified by risk exposure, which is the product of the probability of failure (the chance of an unfavorable occurrence or the risk "bet") and the cost of failure (what happens if the risk "bet" is lost).

Also, during the second quadrant, a project should select one of the alternatives determined in the first quadrant, but only after its risks have been analyzed and averted to the extent possible. The selection and risk classification of alternative processes can be facilitated through having a variety of viable process models. The risks of any given approach can be analyzed and determined as a function of the characteristics of that specific cycle of the process and the "best" process selected as a function of risk mitigation. It is not necessary (or even possible) for a project to eliminate all risks during product development. The intention is to limit the risk exposure so that the success of the project is not in jeopardy. If a project finds itself in a zero risk situation, then the ESP model can become equivalent to other process models such as the Waterfall, Evolutionary, or Transform Process Models, as mentioned in (Boehm 1988).

Availability of the following resources may facilitate the second quadrant risk analysis and aversion activities :

- *Experienced Risk Analyst.* The ESP model depends on risk management to determine appropriate software development activities and their sequence. Although the project manager can function as the risk analyst, a project may find it desirable to have someone properly trained in risk analysis and management techniques. When building ESP-based process models, an explicit role of risk analyst should be predefined and bound to the activities of the second quadrant, regardless of the cycle.

- *Budget for Risk Activities.* DoD *contracts* specifying DOD-STD-2167a require some form of risk management (Department of Defense 1988, paragraph 4.1.4). As a general rule, expect to allocate the following portion of a project's software development budget for risk management activities (Charette 1991):

    - One to three percent for low-risk projects.

    - Three to five percent for medium-risk projects.

    - Five to seven percent for high-risk projects.

The reader should note the importance of metrics (as discussed in Section 6.2) for building organization history of project risk profiles. What types of projects introduce which types of risk, and how much? Which products increase risk, and does that risk increase or decrease over time? Your process definition and process models can explicitly represent risk management heuristics, but deliberate measurements must be the basis from from which those heuristics are constructed.

The specific activities of the second quadrant are:

- *Identify Risks.* Determine and categorize risks for the project and/or current cycle.

- *Analyze Risks.* Determine the likelihood of occurrence and consequence of each risk, and rank risks.

- *Evaluate Risks.* Identify and evaluate possible risk aversion strategies.

- *Risk Review.* Review by project team the risk identification, analysis, and evaluation activities.

- *Plan Risk Aversion.* Select and plan for the most appropriate risk aversion strategy.

- *Commit to Aversion Plan.* Formal recognition commitment to the selected risk aversion plan by all stakeholders.

- *Avert Risks.* Perform to risk aversion strategy.

The second quadrant risk analysis and aversion activities are graphically depicted in Figure 6-5.
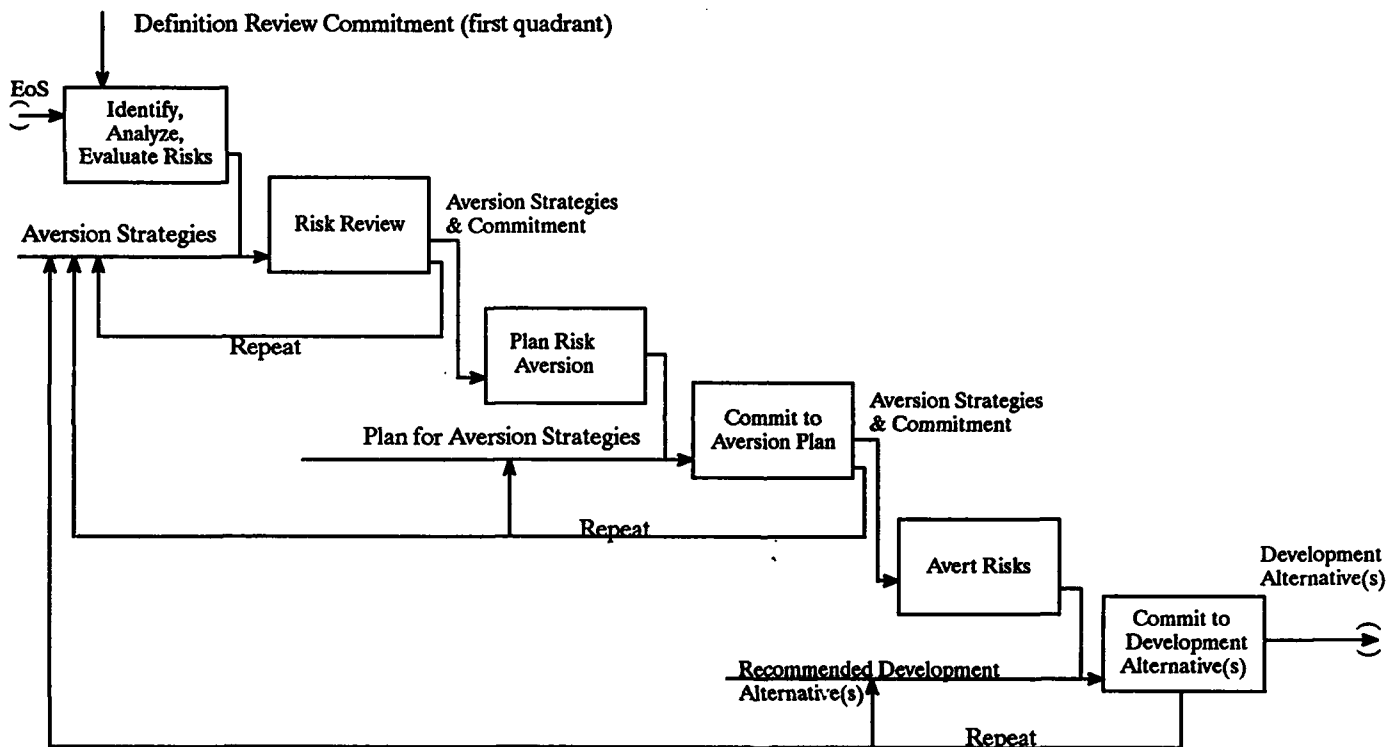
Figure 6-5. Risk Analysis and Aversion Activities

6.3.3 THE THIRD QUADRANT: DEVELOP PRODUCT

A project plans, schedules, and performs the technical activities for the cycle in the third quadrant. The activities should be driven from the development alternatives committed to in the second quadrant. The results of performing the activities should directly support the cycle objectives and success criteria defined in the first quadrant.

The specific activities that a project performs in the third quadrant are:

- *Plan and Schedule.* Identify, organize, schedule, and bind technical activities after any risks associated with development alternatives for the cycle have been averted.

- *Commit to Plan.* Review and commit to the cycle plans.

- *Develop and Verify Product.* Perform the activities for the cycle to produce artifacts or advance product maturity, and verify that the artifacts or advanced product maturity meet requirements, as well as cycle objectives, and success criteria.

- *Monitor and Review.* Monitor and review the technical development activities as they are being performed.

- *Technical Product Review.* Review product or part of product developed to ensure that cycle objectives and success criteria were met.

Generally, the percentage of templates used to detail the events, throughputs, supports, and constraints that apply to this quadrant of the cycle will exceed that of the other three quadrants combined. This is particularly true as the cycle advances from planning-intensive and prototyping activities and evolves into production-intensive cycles. Depending on the size of the project being managed, it can be advantageous on larger projects to have templates (and graphical models) dedicated to each cycle of the ESP. Although initially more labor-intensive, if your organization tends to repeatedly develop the same types of products or use the same types of processes, there is excellent opportunity to reuse process architectures from prior applications of ESP. For new projects, the general events, resources, throughputs, and constraints would typically have the same relationships as they had in prior project instantiations. Much of the work in the new project can then be given to project- or product-specific issues as opposed to process-specific.

Each of the third quadrant product development activities are graphically depicted in Figure 6-6.

6.3.4 The Fourth Quadrant: Manage and Plan

During the fourth and last quadrant, a project takes stock of progress based on the outcome and lessons learned during the cycle, compares actual results against the cycle objectives, re-evaluates and updates Master Planning documents and decides what to do next. Again, the potential contribution of a regular and uniform program of measurement can be seen. The planning and management activities serve to validate the work performed in the cycle and to adjust the project strategy based on the newly available information. The measurable objectives, or the success criteria defined during the first quadrant and monitored during the third quadrant, are critical to this set of activities in order to judge whether the cycle is complete or needs to be iterated or if an alternative should be modified or abandoned.

The specific activities of the fourth quadrant are:

- *Baseline Product.* Place the product(s) or part of the product produced as a result of executing the third quadrant development activities under configuration control.

- *Review Progress.* Evaluate cycle plan actuals versus estimates, success criteria, and lessons learned, and update process drivers, including including project objectives, success criteria, alternatives, constraints, risks, estimates, and other information.

- *Develop/Update Engineering and Project Procedures.* Update engineering and project procedures, if necessary, based on lessons learned during the cycle.

- *Update Master Plan.* Update all planning documents, as necessary, to record actual progress, reflect lessons learned, update estimates based on actual data, and update process drivers.

- *Management Review and Commit to Proceed.* Review updates to the Master Plan and commit to proceed with next cycle.
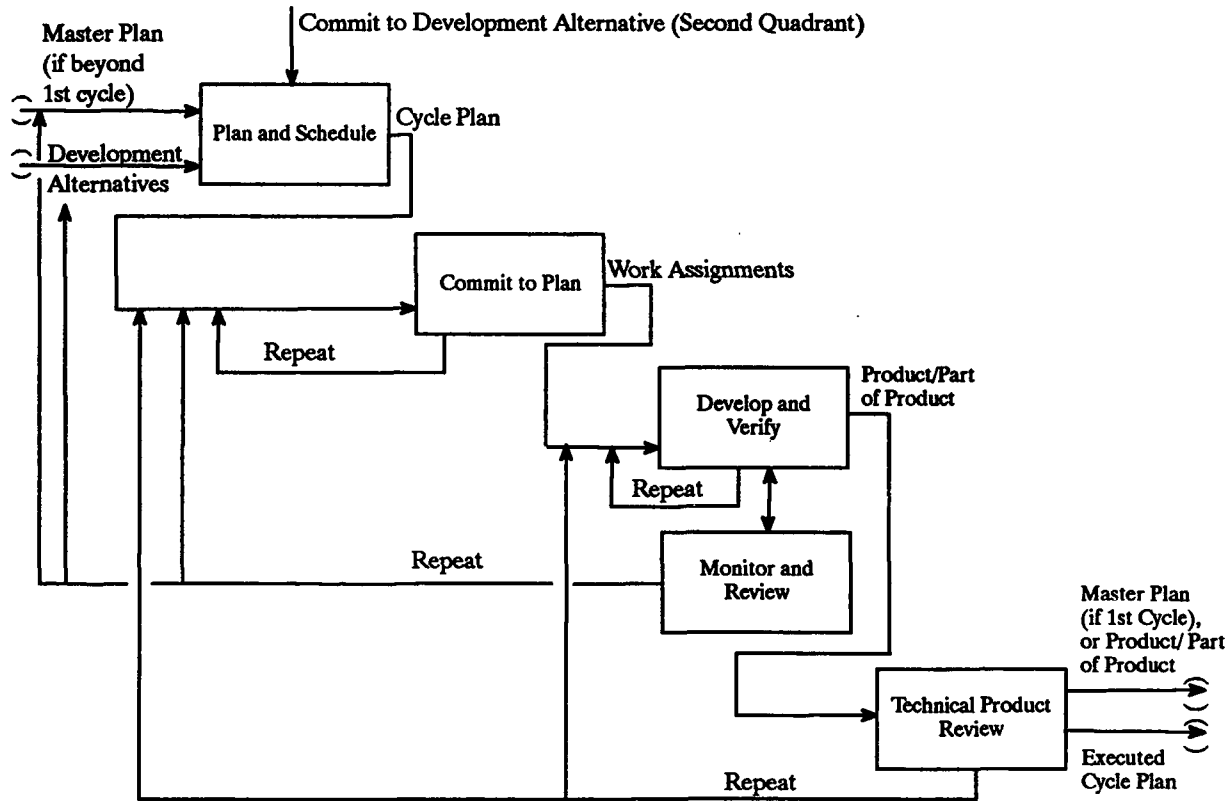
Figure 6-6. Develop Product Activities

It should be noted that with the ESP approach, detailed plans of later cycles may not be known or defined at the time of project initiation. This is contrary to traditional approaches which assert that the entire plan must be completely in place before any work can commence. The ESP approach includes the premise that the critical insights and information necessary for completing the project simply are not available at the time of project commencement. This is increasingly true as the duration of a project becomes increasingly longer.

As a compromise to not having committed to future project details, a process model can be used to indicate the general constraints that will exist on the later-cycle activities to be performed. For example, the project model might require that each activity include a task for performing verification and validation. Therefore, although within the ESP a project manager defers making decisions until sufficient information is available to support those decisions, the process model can constrain that manager's future decisions such that the manager can not, in this example, choose to save time by eliminating the validation tasks.

Each of the fourth quadrant plan and manage activities are graphically depicted in Figure 6-7.

6.3.5 EVOLUTIONARY SPIRAL PROCESS AND DYNAMIC PROCESS IMPROVEMENT

An integral feature of the ESP model is the ability to engineer the best possible process for addressing project needs. When following the ESP model, a project engineers its process dynamically by continuously documenting, instantiating, enacting, and evolving its software development process definition in an evolutionary fashion as shown in Figure 6-8. Regardless of the notation used to define your process (ETVX, SADT, the templates, etc.), the ESP model specifically provides for an orderly and controlled evolution of a baselined project process representation.
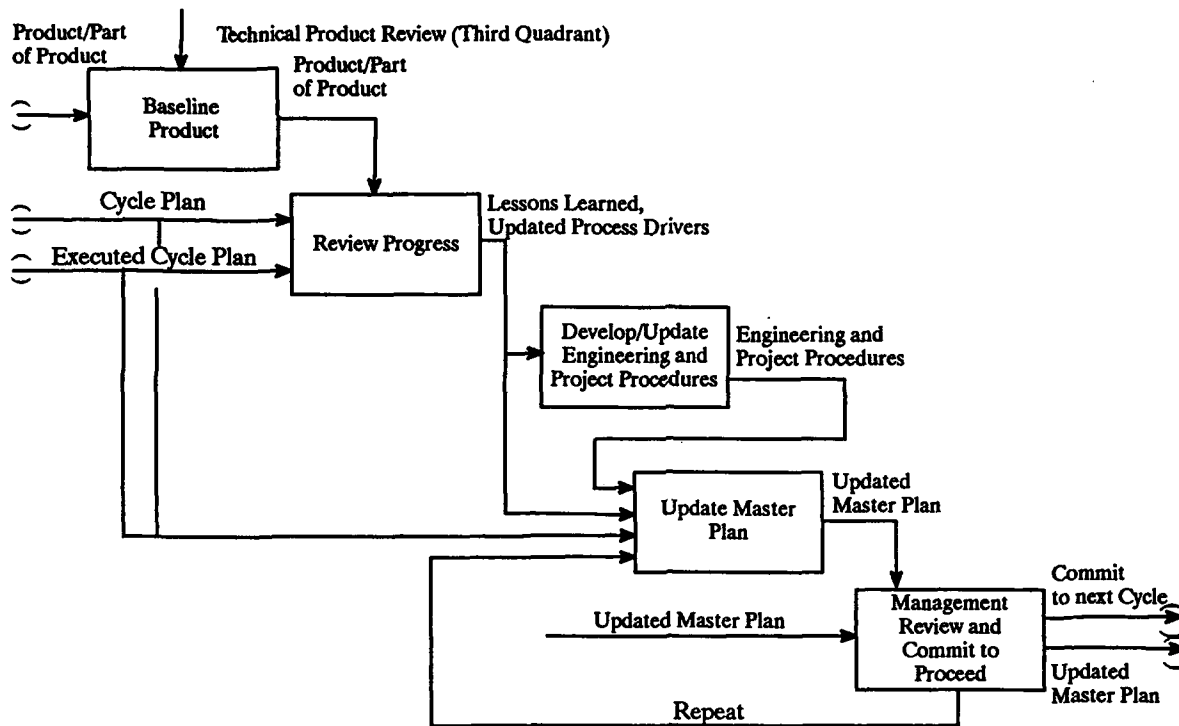
Figure 6-7. Manage and Plan Activities
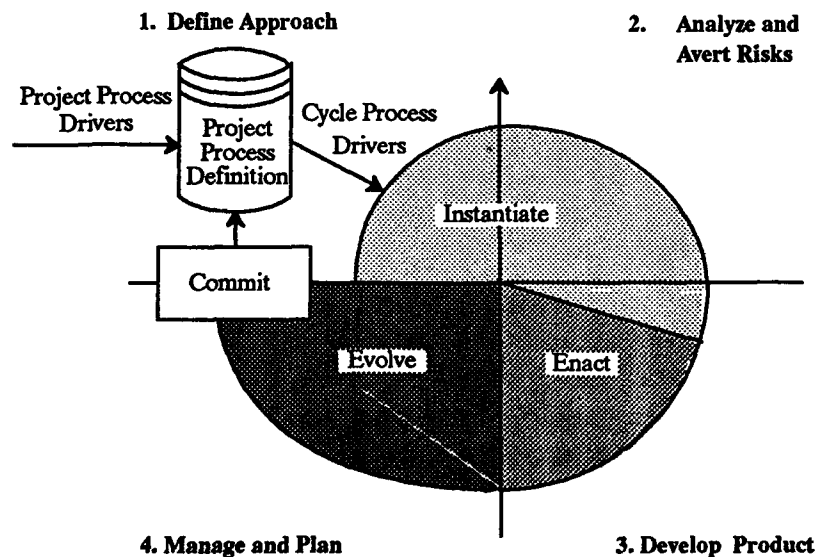
Figure 6-8. Evolutionary Project-Level Process Engineering

Evolutionary process engineering emphasizes the identification and continuous evaluation of key project and cycle characteristics with the potential for driving the process in some significant way. In the first cycle, a project's process definition is documented by defining an overall process from scratch or by tailoring an organizational process definition to the level of detail appropriate to the project's current understanding and its unique process drivers. In subsequent cycles, a project instantiates its project-level process definition by defining it to an enactable level of detail, enacts or performs the cycle process,

and evolves the process definition for the remainder of the project based on information gained, lessons learned, progress to date, and early strategies and mitigation decisions.

A project can follow the ESP model to engineer its software development process regardless of the level of organizational (CMM) process maturity. Although time and resource intensive, a project can develop its project process definition without having an organizational process definition to use as a foundation. The project-specific process model, activity specifications, and method descriptions engineered as a result of following the ESP model can be abstracted, and used by organizations at an ad hoc or repeatable process maturity level as input to developing an organizational process definition.

Experience shows that developing comprehensive software process definitions can be very expensive and time consuming, and it may be more desirable to develop general purpose process definitions together with techniques for reusing, tailoring, and enhancing them (Feiler and Humphrey 1992). Projects that are a part of an organization at the defined level of the CMM or above will likely spend less time and effort engineering its process because of the availability of:

- A standard organizational process definition.

- Tools, methods, and techniques, and supporting technology transfer mechanisms.

- An advanced and/or automated organizational software engineering environment.

- Historical technical and management data.

A project at a higher level of process maturity will generally tailor and instantiate the available organizational process definition and historical data into a project process definition as required by unique process drivers.

The ESP model encompasses process engineering actions at the project-level by:

- Documenting and baselining a project process definition in the first cycle.

- Instantiating the process for a cycle based on cycle process drivers.

- Evolving a project process definition as a project progresses and the process it enacted.

As organizations reach the highest levels of process maturity, it may be possible to optimize traditional process engineering activities by developing specific tailoring and instantiation guidance.

Some type of process representation is crucial. As discussed in Section 2, the "best" representation for you will be driven by a variety of factors that not only differ between organizations but which also change, in time, within the same organization. Regardless of your choice, it is clear that some representation beyond free-text based descriptions is critical for documenting, analyzing, and communicating about your process. In closing, the ESP approach is highly dynamic and makes change management an explicit, accepted, and deliberate part of the process. Consequently, to depict ESP-based processes the representation should likewise remain flexible, maintainable, and easily tailorable—all deliberate characteristics of the template-based notation proposed by this guidebook.

6.4 MANUAL PROCESS MANAGEMENT

Process descriptions, process models, programs of measurement, all can be used—and should be used—to support the actual management of projects. Once you have defined a model and have instantiated a project plan from the model, you can facilitate process management by using a variety of techniques or tools that monitor the actual evolution of the project. Sections 6.4.1 and 6.4.2 are two examples of ways you can use process representations to support process management. The first example discusses the value of using process checklists. The second example assumes at least some level of automation is available.

6.4.1 PROCESS CHECKLISTS

At an abstract level, the purpose of using a checklist is to allow a person to confirm compliance with a set of rules. These rules may require that certain items be available (such as in an inventory-type checklist) or that certain events or steps occur. A process checklist provides a simple method for guiding a person through a series of events; for verifying that certain events have occurred; and for verifying that throughputs, supports, etc., are available, generated, or participating as designed in the overall process.

The fields on the templates are used to develop first-draft process checklists. Successive versions of the checklists can add information not reflected on the templates. For example, an event template representing the final coding activity (before requesting an inspection) might state in its "exit criteria" field that the software code artifact (defined elsewhere using a product template) must be in a state of "ready_for_inspection."

For checklists, these state-sensitive requirements need to be changed to a list of descriptive statements elaborating on the criteria characteristic of that state. Continuing with this example, "ready_for_inspection" may mean that (1) the code compiles cleanly, (2) the program listing has both page and line numbers, (3) each module includes a copyright statement, (4) each module has a descriptive prologue, etc. Providing programmers with this type of checklist not only helps them to stay in compliance with the process, but it also provides a very easy way to keep them informed of exactly what the process is.

Additionally, checklists can include fields for a person to place their initials and the date and time, signifying that a given item on the checklist has been completed or confirmed. This provides an easy method for management to collect time-based information such as total hours spent within specific activities, total hours interval between different activities, etc. From this perspective, checklists can be a powerful tool as a medium for binding process representations and metrics.

6.4.2 ELECTRONIC MAIL-BASED PROCESS

The advantages to managing and handling checklists through electronic mail are considerable. One of the primary advantages is the ability to automatically notify individuals involved in a process that their input is needed on various checklists. In a process of any size, having checklists to monitor the entry and exit criteria of every event within that process quickly leads to an avalanche of checklists in circulation. The moment that the collected data becomes inconsistent, skipped, or randomly estimated, the value of using the checklists drops abruptly. These scale-up considerations can largely be mitigated through the use of electronic mail-based checklist management.

With the exception of inter-activity defect tracking, you should also note that comparative analysis of the fundamental relationships between the checklist line items is a relatively trivial automation effort. If a reasonably structured format is followed, it is relatively simple to write a parser that interprets the checklists, extract criteria compliance, and time-stamp values (as discussed above) and loads those values into a process metric database. Reports within the database environment are then developed that perform basic comparative analysis, perhaps provide a variety of summarized reports to management, and possibly recommend ways in which process quality can be improved and productivity increased.

6.5 AUTOMATED PROCESS MANAGEMENT VIA INTEGRATED ENVIRONMENTS

Although electronic mail is probably the most widely used tool for automated support of teams and groups of interacting people, considerable research continues to be done in advancing the state of the art for supporting teams of people. There is no doubt that a team of individuals working in a cooperative, cohesive, and coordinated manner can accomplish greater objectives, more efficiently, and more safely than people working individually and in isolation. The same is equally true of the processes and tools such a team uses.

The Information Engineering industry has amassed a vast collection of separate, stand-alone tools and techniques. Various research efforts are investigating alternatives for achieving synergistic advantages through integrating these into cohesive environments. More recently, considerable research has gone into processes in general: process maturity, process assets (i.e., reusable subprocesses), process asset libraries, etc. Virtually any effort to improve process coordination and management can be greatly supported by automating aspects of process management and enacting them within an integrated environment. Sometimes referred to as Integrated Process Support Environments (IPSEs), these environments strive to integrate not just the tools, but also the methods and techniques by which the combination of tools can be used. The resulting environments provide an ideal foundation from which to pursue EBPM.

One of the more traditional approaches to constructing environments supporting specific processes is to build a meta-service under which a variety of existing tools can be run. Ideally, such a layer also handles nominal file and data conversion efforts so that underlying tools can pass data through and between each other with minimum guidance from the user.

One example of this type of approach is the Software Life Cycle Support Environment (SLCSE): a VAX/VMS-based software development environment with a common user interface behind which "an environment framework that can create a variety of environments, each tailored to the needs of a particular software development project" (Strelich 1988). Regrettably, meta-layer tool approaches often encounter incompatible underlying data formats in the tools they attempt to integrate.

To partially address this problem, there are several significant ongoing efforts toward standardization. Aside from the relatively widespread practice of companies attempting to define and adhere to their own in-house standards, there have been fairly ambitious efforts toward developing industry-wide, national, and international standards. A major effort undertaken by the European community is the Public Common Tool Interface or PCTE. Although somewhat less true today, the basic persistent problem is that, "Most software engineering tools that are presently available result from individual efforts and tend to constitute a collection of vaguely related products, each filling a particular function, but without much consideration for the software development process as a whole" (Campbell 1986). Note the reference not just to the tools of the software industry, but to the process that, ideally, must accompany these tools. It is evident that robust integrations are achieved by combining both components and processes.

To date, the preferred approach to achieving cost-effective IPSEs has been to focus on abstract machine implementations, and to seek methods and techniques that provide users an environment supporting sometimes radically diverse roles. Additionally, this needs to be achieved under a reasonably consistent interface, and must allow consistent enforcement of the engineering process. However, support from an environment can consist of distinctly different types. Many environments are passive IPSEs and, typically, exhibit only the two most basic types of integration (data and user-interface). These systems are passive in the sense that they support rather than participate in the development process (Rodden and Sommerville 1988). Active support environments differ from passive environments in that they layer process and methodology integration onto user-interface and data integration.

The integration of process into IPSEs is well demonstrated by ISTAR. ISTAR uses a "contractual" approach for managing various work activities. Specifically, there is a hierarchical decomposition of work units into progressively smaller units until sufficient detail is achieved for management purposes. Effectively, each task then becomes an individual contract; subtasks are translated into subcontracts and so on down the hierarchy. In all cases, the contracts specify in a precise and sufficiently detailed manner the task to be performed and the deliverables required (Stenning 1986).

One of the most crucial characteristics exhibited by any IPSE is its ability to **evolve** and adapt to the changing requirements of both its environment and the users it supports. Particular emphasis has been given to this issue by the Arcadia project. As described in Taylor et al. (1988), "The Arcadia research project is investigating the construction of software environments that are tightly integrated, yet flexible and extensible enough to support experimentation with alternative software processes and tools."

This is a particularly challenging goal in that integrating a process into an environment is in many ways orthogonal from simply integrating another tool. Typically, a tool need only concern itself with one type of data format. A process, however, may attempt to usher a component through several different stages in the evolution of that artifact; stages which may, at times, embody considerably different representations of the component. Furthermore, tools rarely need to be state sensitive. Conversely, a process must not only be state sensitive but, practically speaking, it is essentially state driven. Even a comparatively simple process layered onto an IPSE can yield extensive collections of artifacts evolving through the state transitions managed by the IPSE.

In addition to process support, another characteristic of some EBPM implementations is actual process enforcement. An example of this approach is Darwin. An interesting characteristic of Darwin is that the law which governs the operations on basic objects within the system is also used to govern meta-messages, and thereby simultaneously constrain the evolution of the system (Minsky and Rozenshtein 1988). Not only are object services and access enforced, but the process itself (and the methods by which that process may change and evolve) is also enforced. Arguably, it could be stated that Darwin is one of the first IPSEs to acknowledge that management of the process itself is of higher priority than the individual functionality expressed by underlying tools.

6.6 SUMMARY

The material in this section has provided discussion into a variety of areas related to the establishment and support of a process representation program. Discussion has ranged from a manually managed process to integrated, automated process support; from resources required to start a program to an example of a new process management paradigm; from simple support tools such as checklists to complex programs involving detailed measurements and process related metrics. However, throughout

this section, and repeatedly throughout the guidebook, one theme often serves as the hub around which other discussions evolve: process maturity.

As you may recall, the highest level of process maturity is Level 5, the optimizing level. This level is characterized by improvement insights that directly translate into process changes. These changes include efforts toward defect prevention, improved development environments, and an overall trend toward automating the software process (as discussed in Section 6.5).

From this perspective, planning and managing process improvement translates to progressive advancement to higher stages of process maturity. Well defined and documented process descriptions and models are one means of beginning to work at the third level of process maturity (the "defined" process level).

Although metrics are also crucial to higher levels of process maturity, they can contribute to process improvement from any level. From the perspective of process definition and modeling, the Consortium highly recommends that after developing one or more (possibly template-based) process models, a set of metrics should be defined that allow you to gain insight into the processes instantiated from the model. By collecting data relevant to these metrics, comparative project histories will begin to develop. These histories can then be used as the foundation for process optimization.

In Section 4 there is a discussion on how template-based process models can support a program of process improvement. The method involved performing **static** analysis of the characteristics of the template-based description and deriving an improved process as a result of insights gained from the static architecture. Metrics allow for collecting information on the **dynamic** or behavioral characteristics of processes, and allows for insights to be gained into the temporal aspects of a process model.

Whether from static or dynamic analytical techniques, process improvement can be facilitated by proposing and constructing new or alternative process models and by analyzing those models for desirable characteristics. This can be a cyclic process repeatedly involving the use of static analytical techniques to gain insights into potential process improvements.

At some point, however, it becomes necessary to evaluate the proposed approach by deriving an instantiation of a pilot or shadow project from the model and then monitoring the evolution or enactment of that process. As discussed above, it is highly recommended—almost imperative—that some set of metrics is employed to measure the process. Actual values must be regularly compared with predicted values. In support of this effort, it may be useful to construct a series of checklists (as discussed above) that simplify and otherwise facilitate the collection of metric-related data. (If an automated, integrated process support environment exists, metric data collection can be almost entirely automatic.)

These efforts will result in a new set of history data being collected: data that reflects the characteristics of the experimental or new process. If the results are generally positive, other projects can be instantiated from the model yielding increased benefit to the organization. The additional projects will provide still further data which in turn can be used to guide the exploration of further improvements to the process models. Ideally, the organization settles into a regular cycle of improved models, improved organizational processes, improved pilot projects, improved actual projects, improved process insights based on metrics, which completes the cycle by contributing once again to improved process models and descriptions.

7. SUMMARY

As you can see in Figure 7-1, this section can be reached by a variety of routes. Any of Sections 2, 3, 5, and 7 serve as an immediate predecessor to this section. Depending on the number of sections read prior to reading this section, the material contained in this summary will seem more or less familiar.
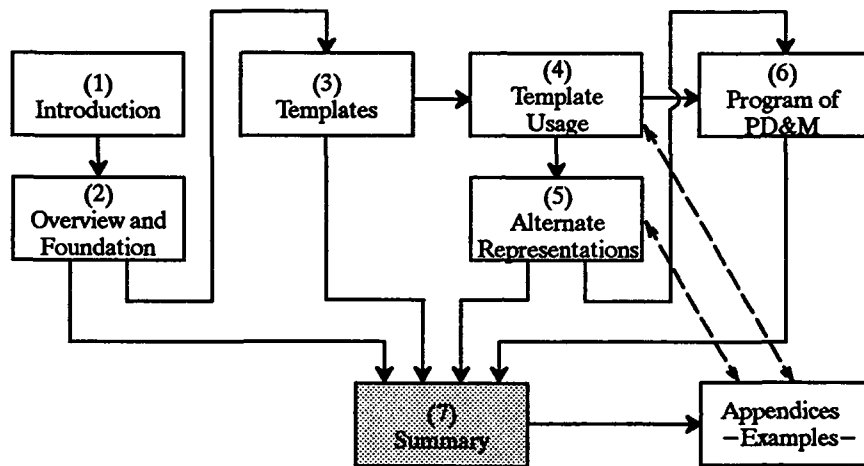
Figure 7-1. Guidebook Organization View 7

7.1 REVIEW

One of the chief contributions of this guidebook is the set of templates and graphical conventions proposed in Section 3. These templates and graphical techniques are completely adequate for use as is, but they are also designed to be easily tailored to site-specific objectives. Event templates (Managerial and Production), Throughput templates (Products and Research), Support templates (Roles and Resources), and the Internal and External Constraint templates all allow virtually any characteristic of a process to be defined and modeled.

This guidebook also presented several alternative notations (such as SADT and Petri nets) and showed how the templates can be used as a direct source of information for constructing process representations based on these alternative notations, and how the templates can be used to extend or augment the information already captured using a different representation.

Section 6 provided discussion, insights, and recommendations on how an organization can proceed with establishing a process representation program. This section also presented techniques for using process representations as a means to support manual process management or automated process management via integrated environments. Metrics were discussed, especially from the perspective of how metrics (and process representations, templates, and checklists) can be part of a larger program that contributes toward overall process improvement.

In the appendices you will find a variety of process representation examples. This includes an extensive example which provides step-by-step guidance in developing a template-based representation of a hypothetical "SWAT" formal inspection process. This example not only discusses an approach to performing process analysis and representation, it also presents an example and explanation of graphically rendering process information by constructing a Petri-net based model (note that this is distinctly different from using the graphical conventions described in Section 3).

Throughout the guidebook, the principle focus has been on templates and template-based graphical techniques. The primary reason for this orientation is that the advantages to using the templates are considerable. The templates are easy to learn and very easy to tailor. Furthermore, they can be easily and efficiently used to capture and maintain representations of even volatile or unusual processes, or to derive and analyze new models of proposed processes. This is especially true if initial renditions are done partially or entirely using graphical depictions.

It cannot be overstressed that this entire area of process analysis, process design, process representation, and process optimization is best approached cyclically: working from smaller to larger models, from tightly defined to broadly defined domains, from simple constructions of process architectures to more complex constructions and architectures, from generic templates to explicitly tailored or domain-sensitive templates, etc.

Although alternative notations can be advantageously used (especially if residual in-house experience is available), it is worth emphasizing that the templates are capable of capturing more information than is typically gathered using any one alternative notations. The templates can also be used (especially if the suggested or tailored, default state sets are employed) to bind such information more formally, and in more ways, than is often possible using alternative notations.

An overall goal of this guidebook has been to present a means whereby a company or organization can simply, efficiently, and cost effectively use a variety of process representation tools and techniques to support improved process analysis, design, and optimization. From this foundation, an organization is better positioned to readily answer questions such as:

- Where are the new opportunities for improving process quality?

- How can process risk be further reduced?

- How can process efficiency be increased?

- How can resources be used more efficiently and effectively?

- How can product quality be increased as a function of a higher quality process?

Finding answers to all these questions can be facilitated by analyzing process representations, by collecting and investigating process-related metrics and data, and by developing progressively more accurate depictions of the current and future processes that characterize the organization. While it is true that process definition and modeling is just one step to an overall program of continuous process optimization, it is a significant step and, as proposed within this guidebook, relatively simple, low-risk, and therefore potentially quite cost-effective.

7.2 FUTURE WORK

At the time of publication, there are three major new additions planned for the next release of the Process Definition and Modeling Guidebook. These are:

- Addition of a new meta-class template (Measurement) and at least two supporting class templates (Quality Measures and Productivity Measures).

- Extension of the graphical notation to highlight dynamic process characteristics.

- Development and elaboration of a mathematical representation and supporting mathematical techniques for increased formality in process model construction, analysis, and verification.

Other material already in the guidebook, but planned for expansion, are the areas of metrics and the use of process representations to facilitate the construction of project management plans and to facilitate ongoing project monitoring and control. Potentially scheduled for the next release is expanded material on the use of process representations to support integrated, EBPM.

This page intentionally left blank.
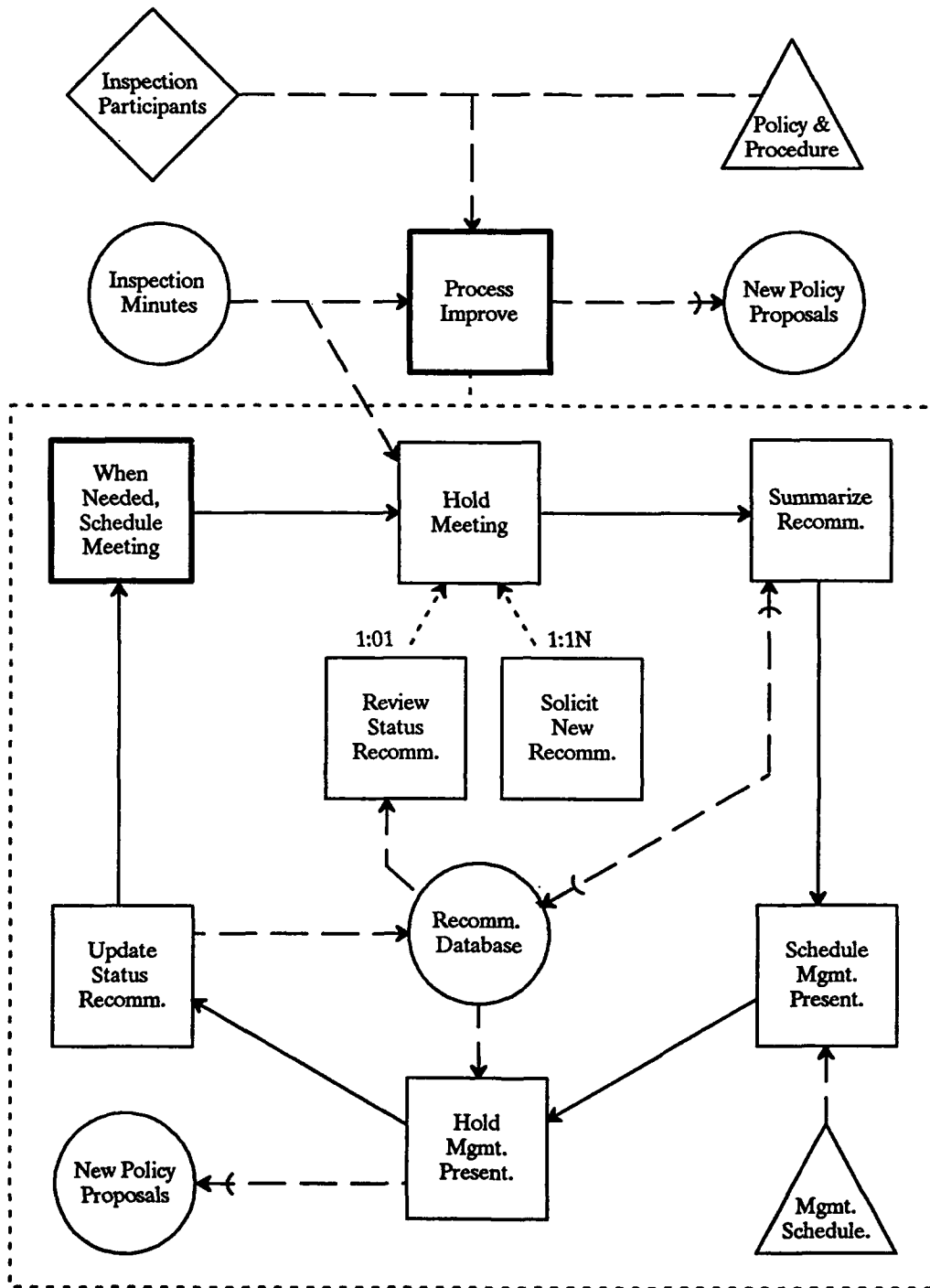
Figure 3-40. SWAT—D3

APPENDIX A. EXAMPLE TEMPLATE USAGE

This appendix describes a hypothetical process. It provides a detailed example of how you can analyze that process and how you construct a template-based process model.

The example presents a hypothetical business, Company-X, with a process analyst who is contracted to work on site and construct a process model.

A.1 DEFINITION AND MODELING APPROACH

Assume the following. A process analyst has been hired as a process consultant to define and model a process for Company-X. Company-X is interested in achieving Level-3 in the Software Engineering Institute's (SEI) assessment ranking. As a step toward that goal, they have introduced an "Inspection SWAT Team" which they intend to use to rapidly expand and improve their application of Formal Software Inspections. Company-X has contracted the analyst to build a template-based representation of their Inspection SWAT Team Process (ISTP) and then develop a Petri net representation from the templates.

The approach the analyst plans to take involves the following steps:

1.   Perform initial data collection

2.   Construct high level event templates

3.   Construct external constraint templates

4.   Construct role templates

5.   Construct product templates

6.   Bind external/role/product templates to event templates

7.   Construct low level event templates

8.   Extend existing templates (external, roles, products)

9.   Construct resource templates

10. Construct research templates

11. Construct internal constraint templates

12. Bind all throughput/support/constraint templates to event templates

13.   Evaluate template set and improve clarity/simplicity

14.   Derive Petri net representation of process dynamics

A.2 FORMAL INSPECTION PROCESS EXAMPLE

Beginning with the first step, the process analyst ascertains the following information.

Step 1. Perform Initial Data Collection. The analyst performs this step by examining company policies and procedures, and by interviewing key personnel. During this effort the analyst keeps notes on general process-related information regarding the Company's ISTP.

After the analyst summarizes his notes, he finds the following general information:

- There are seven divisions that can request the services of the SWAT team; however, the four divisions that develop real-time Ada systems have priority.

- The ISTP occurs in six stages:

    - Planning

    - Overview

    - Preparation

    - Inspection meeting

    - Rework

    - Follow-up

- Two process improvement events also occur:

    - Causal analysis

    - Inspection process improvement analysis

- There are a variety of roles that support the inspection process:

    - Super moderator

    - Moderator

    - Author

    - Key inspector

    - Inspector

    - Scribe

    - Reader

- Although there are plans to expand the program, the ISTP is currently used to inspect Ada packages and such technical documentation as:

    - End-user guides

    - System administration guides

    - Installation guides

- Inspect Ada packages against the Software Productivity Consortium's *Ada Quality and Style: Guidelines for Professional Programmers*. Inspect technical documentation against Company-X's Guidelines for Technical Documents.

- Due to the number and frequency of inspections, a dedicated meeting room has been set aside exclusively for inspections.

- There is a pool of approximately 20 employees to serve as SWAT members. The super moderator has overall responsibility for the entire program, and five others are authorized to be moderators. Those five plus another four are authorized as key inspectors and readers. All can serve, if necessary, as inspectors and scribes. Since all but the super moderator have principle engineering responsibilities, anyone may occasionally serve as the role of author.

- The company is compiling a series of checklists. These checklists specify entry and exit conditions for various inspection related events.

A.3 TEMPLATE USAGE

The remaining steps of the analyst's original plan involve using the default set of templates (as presented in Section 3).

Step 2. Construct High Level Event Templates. Using the above information (augmented by notes, phone calls, or conversations with participants), the analyst makes an initial indented list.

```
Inspection SWAT Team Process
    Inspection Activity
        Planning
        Overview
        Preparation
        Inspection Meeting
        Follow-up
    Causal Analysis Activity
    Inspection Process Improvement Activity
```

Next, using the indented list as a general guideline, the analyst constructs high level event templates. He then details the internal processing and basic event relationships (entry and exit criteria). (See the following pages.)

For reference purposes, the analyst also makes note of the following event-specific states (recommended as the default set of states in the guidebook).

Default Set of Process States:
    Pre-Enabled
    Enabled
    In-Progress
    Disabled
    Suspended
    Cancelled
    Completed

Level # 1	Management Template Template Type	Version # and Date 01a_070493

Name
   SWAT_Process
	Unique Identifier
	   SWAT

Purpose
   Inspection SWAT Team Process

Comments

Additional States

Entry Criteria	Internal Processing	Exit Criteria

VP_Request	In Parallel Do...
	    SWAT state is In_Progress
	    Perform SWAT_EA_INSPECT
	    Perform SWAT_ET_CAUSAL
	    Perform SWAT_EA_PROC−IMP
	Until VP Requests Cessation
		VP_Request

Throughputs	Supports

Parent Event(s)

Child Events

   SWAT_EA_INSPECT
   SWAT_ET_CAUSAL
   SWAT_EA_PROC−IMP

Internal Constraints	External Constraints

Revision History

   01a_070493 − Initial Version; 1st high level pass

Level # 2	Management Template Template Type	Version # and Date 01a_070493

Name
  Inspection
	Unique Identifier
	  SWAT_EA_INSP

Purpose
  This is the overall inspection activity and contains, as shown below, several stages.

Comments

Additional States

Entry Criteria	Internal Processing	Exit Criteria

SWAT::In_Progress	SWAT_EA_INSP state is In_Progress
	1) Perform SWAT_ET_INSP_PLAN
	   Perform SWAT_ET_INSP_OVER
	   Perform SWAT_ET_INSP_PREP
	   Perform SWAT_ET_INSP_I–MTG
	   Perform SWAT_ET_INSP_REWORK
	   Perform SWAT_ET_INSP_FOLLOW
	   If Reinspection Needed
	        Go To (1)
	   SWAT_EA_INSP state is Complete
		SWAT_EA_INSP::Complete

Throughputs	Supports		Internal Constraints	External Constraints
		Parent Event(s) SWAT		

		Child Events
		  SWAT_ET_INSP_PLAN
		  SWAT_ET_INSP_OVER
		  SWAT_ET_INSP_PREP
		  SWAT_ET_INSP_I–MTG
		  SWAT_ET_INSP_REWORK
		  SWAT_ET_INSP_FOLLOW		

Revision History

  01a_070493 — Initial Version; 1st high level pass

Level # 3	Production Template Template Type	Version # and Date 01a_070493

Name Planning	Unique Identifier SWAT_ET_INSP_PLAN

Purpose
This represents the planning stage of the inspection activity.

Comments

Entry Criteria	Internal Processing	Exit Criteria
SWAT_EA_INSP::In_Progress	SWAT_ET_INSP_PLAN state is In_Progress Evaluate the Item to be Inspected Estimate the expected inspection rate Construct a schedule Assign personnel to the schedule Confirm personnel availability Adjust/Redo schedule as necessary SWAT_ET_INSP_PLAN state is Complete	SWAT_ET_INSP_PLAN:: Complete

Products	Supports		Constraints
		Parent Event(s) SWAT_EA_INSP	
Research		**Child Events**	

Revision History

01a_070493 — Initial Version; 1st high level pass

Level # 3	Production Template Template Type	Version # and Date 01a_070493

Name Overview		Unique Identifier SWAT_ET_INSP_OVER

Purpose
> This represents the overview stage of the inspection activity.

Comments

Entry Criteria	Internal Processing	Exit Criteria
SWAT_EA_INSP::In_Progress and SWAT_EA_INSP_PLAN:: Complete	SWAT_ET_INSP_OVER state is In_Progress Introduce Objective Introduce Members Have Reader present the item to be inspected Review preparation time and schedule Dismiss meeting SWAT_ET_INSP_OVER state is Complete	SWAT_ET_INSP_OVER:: Complete

Products	Supports		Constraints
		Parent Event(s) SWAT_EA_INSP	
		Child Events	
Research			

Revision History

> 01a_070493 — Initial Version; 1st high level pass

Level # 3	Production Template ————————— Template Type	Version # and Date 01a_070493

Name
    Preparation
	Unique Identifier
	    SWAT_ET_INSP_PREP

Purpose

    This represents the preparation stage of the inspection activity.

Comments

    The preparation stage is characterized by all inspectors (both key and otherwise) reviewing the material for defects and completing trivial error logs and preparation error logs (both of which are brought to the actual inspection meeting).

Entry Criteria	Internal Processing	Exit Criteria

SWAT_EA_INSP::In_Progress
and
SWAT_EA_INSP_OVER::
Complete
	SWAT_ET_INSP_PREP state is
	     In_Progress
	Disperse material to be inspected and
	     associated support forms
	Each Inspector Does:
	     Review of Relevant Standards
	     Personal Inspection of Item
	     Completes Trivial Error Log
	     Completes Preparation Log
	     Completes Preparation_Exit
	          Form (Checklist)
	END [Each Inspector Does]
	Moderator Collects/Verifies Checklists
	Moderator distributes Inspection
	     Invitation letter
	SWAT_ET_INSP_PREP state is
	     Complete
		SWAT_ET_INSP_PREP::
		     Complete

Products	Supports		Constraints
		Parent Event(s) SWAT_EA_INSP	
Research		**Child Events**	

Revision History

    01a_070493 – Initial Version; 1st high level pass

Level # 3	Production Template Template Type	Version # and Date 01a_070493

Name
    Inspection_Meeting
	Unique Identifier
	    SWAT_ET_INSP_I−MTG

Purpose
    This represents the actual inspection (by the group) of the item to be inspected.

Comments

Entry Criteria
 	
SWAT_EA_INSP::In_Progress
and
SWAT_EA_INSP_PREP::
Complete
	Internal Processing
	
	SWAT_ET_INSP_I−MTG state is
	    In_Progress
	Take Attendance
	When (Everyone has arrived OR its
	        past start time and key
	        inspectors have arrived)
	    Review Purpose
	    Collect Trivial Error Logs
	    Perform Product Inspection
	    Review Findings
	    Determine if reinspection req'd
	END [When]
	Complete/Distribute Inspection exit memo
	SWAT_ET_INSP_I−MTG state is
	    Complete
		Exit Criteria
		
		SWAT_ET_INSP_I−MTG::
		    Complete

Products	Supports		Constraints
		Parent Event(s) SWAT_EA_INSP	
		Child Events	
Research			∴

Revision History

    01a_070493 − Initial Version; 1st high level pass

Level # 3	Production Template _____ Template Type	Version # and Date 01a_070493

Name
  Rework
		Unique Identifier
		  SWAT_ET_INSP_REWORK

Purpose
  Perform the necessary changes and updates to remove defects detected during inspection.

Comments

Entry Criteria
 	
SWAT_EA_INSP::In_Progress
and
SWAT_EA_INSP_I–MTG::
Complete
	Internal Processing
	
	SWAT_ET_INSP_REWORK state is
	  In_Progress
	Each Producer does:
	  Evaluates pertinent defects
	  Alters product only to correct
	    the defect
	  Notes location and scope of
	    changes
	  If Req'd: makes before and after
	    listings
	until all defects addressed
		Exit Criteria
		
		SWAT_ET_INSP_REWORK::
		  Complete

Products	Supports	Moderator notified that
		  all changes are complete
		SWAT_ET_INSP_REWORK state is
		  Complete
			Constraints

		Parent Event(s) SWAT_EA_INSP	
		Child Events	
Research			

Revision History

  01a_070493 — Initial Version; 1st high level pass

Level # 3		Production Template Template Type	Version # and Date 01a_070493

Name Follow-up		Unique Identifier SWAT_ET_INSP_FOLLOW

Purpose
   Verify that the authors/producers have addressed all defects detected by inspectors.

Comments
   Occasionally, the nature of the changes to the inspected item are such that the moderator
   may elect to call a reinspection (even though originally it might not have seemed necessary).
   If no reinspection is necessary, this is the last (regular) stage of the inspection process.

Entry Criteria	Internal Processing	Exit Criteria
SWAT_EA_INSP::In_Progress and SWAT_EA_INSP_I–MTG:: Complete	SWAT_ET_INSP_I–FOLLOW state is In_Progress Meet with the author(s) Discuss nature and scope of changes Evaluate each change Confirm only authorized changes made Determine whether a reinspection needs to be conducted Moderator completes the follow-up checklist SWAT_ET_INSP_FOLLOW state is Complete	SWAT_ET_INSP_FOLLOW:: Complete

Products	Supports		Constraints
		Parent Event(s) SWAT_EA_INSP	
		Child Events	
Research			

Revision History

   01a_070493 – Initial Version; 1st high level pass

Level # 2	Production Template Template Type	Version # and Date 01a_070493

Name Causal Analysis	Unique Identifier SWAT_ET_INSP_CAUSAL

Purpose
Attempt to identify common problems and suggest process changes to originate their cause.

Comments

Entry Criteria SWAT::In_Progress	Internal Processing SWAT_ET_CAUSAL state is In_Progress Take attendance Review outstanding issues from prior causal analysis meetings Report results of trend analysis efforts Solicit perceived problem areas from attendees Prioritize problem areas Discuss/determine process changes to reduce/remove root causes At end of meeting; review findings Dismiss meeting Compile Causal Analysis Report SWAT_ET_CAUSAL state is Complete	Exit Criteria SWAT_ET_CAUSAL:: Complete

Products	Supports		Constraints
		Parent Event(s) SWAT	
Research		**Child Events**	

Revision History

01a_070493 — Initial Version; 1st high level pass

Level # 2	Production Template Template Type	Version # and Date 01a_070493

Name Inspection_Process_Improvement		Unique Identifier SWAT_ET_PROC-IMP

Purpose
Improve the way inspections are actually conducted.

Comments

Inspection process improvement efforts are not constrained to any particular inspection subprocess. Inspections, causal analysis, and even this process are all potentially subject to process improvement efforts.

Entry Criteria
 	
SWAT::In_Progress
	Internal Processing
	SWAT_ET_PROC-IMP state is
	    In_Progress
	Take attendance
	Review outstanding issues from prior
	    process improvement meetings
	Solicit suggestions from attendees
	Organize suggestions by domain
	Evaluate cost/benefit
	Prioritize by estimated cost/benefit
	Investigate alternatives
	Compile final recommendations
	At end of meeting; review findings
		Exit Criteria
		
		SWAT_ET_PROC-IMP::
		    Complete

Products	Supports	Dismiss meeting
		Compile Inspection Process Improvement
		    Report
		SWAT_ET_PROC-IMP state is
		    Complete
			Constraints

Parent Event(s)   SWAT

Child Events

Research

Revision History

01a_070493 — Initial Version; 1st high level pass

Step 3. Construct External Constraint Templates. The process analyst reviews his work to date and decides it is accurate enough to procced with establishing and binding a few other template classes. As indicated in the initial work plan, the first nonevent class of templates tc be included are the external constraint.

Using his notes, the analyst builds the following simple indented list:

> Vice-President; Quality Assurance
> Policies, Procedures, and Guidelines
>> SPC Ada Quality and Style Guide
>> Company-X Technical Document Guidelines

The analyst also builds a list of the default set of states recommended for this class.

Default Set of External Constraint States:
> Compliance_Unknown <EXPORT>
> Compliance_Under_Evaluation
> Compliance_Achieved
> Compliance_Failure
> Compliance_Waived

Finally, the analyst creates the following four instances (one for each line on the indented list) of external constraints.

Level # 1	External Constraint Template Template Type	Version # and Date 01a_070593

Name
    VP Quality Assurance
	Unique Identifier
	    SWAT_CE_VP

Purpose

    Monitor and determine whether the SWAT inspection effort should continue or cease.

Comments

    The VP of Quality Assurance has overall control and responsibility for the SWAT Inspection process.

Special Form Of
(parent)
 	
[None]
	Additional States/Description
	
	Require_Cessation
	    This state is used to represent the VP declaring the SWAT program be ceased.
	Approve_Cessation
	    This state represents VP approval of a super moderator request to cease the SWAT inspection process.

General Form Of (list children) [None]	Constrained Events SWAT	Constrained Throughputs [None]
		Constrained Supports [None]

Revision History

    01a_070593 — Initial Version; 1st high level pass

Level # 1	External Constraint Template Template Type	Version # and Date 01a_070593

Name Policies, Procedures, and Guidelines	Unique Identifier SWAT_CE_PPG

Purpose

Provide written reference standards applicable to conducting the SWAT inspection process.

Comments

Under this group of constraints are all the applicable documented standards necessary for performing the SWAT inspection process.

Special Form Of (parent)
 	
[None]
	Additional States/Description
	
	Require_Cessation
	    This state is used to rep· :sent the VP declaring the SWAT program be ceased.
	Approve_Cessation
	    This state represents VP approval of a super moderator request to cease the SWAT inspection process.	

General Form Of (list children)
 	
SWAT_CE_PPG_
    ADA
 	
SWAT_CE_PPG_
    X−TECHDOC
	Constrained Events
	
	SWAT
		Constrained Throughputs
		
		[All products subject to or participating in the SWAT formal inspection process.]

		Constrained Supports [Any roles or resources which are governed by one or more sections in any applicable documentation.]

Revision History

01a_070593 − Initial Version; 1st high level pass

Level # 2	External Constraint Template Template Type	Version # and Date 01a_070593

Name
    SPC Ada Quality and Style Guide
	Unique Identifier
	    SWAT_CE_PPG_ADA

Purpose
    Provide Ada-specific reference standards applicable to Ada throughputs/products.

Comments

Special Form Of
(parent)
 	
    SWAT_CE_PPG
	Additional States/Description
	
	    [None]

General Form Of
(list children)
 	
    [None]
	Constrained Events
	
	    SWAT
		Constrained Throughputs
		
		    [All Ada-oriented software products subject to or participating in the SWAT formal inspection process.]

		Constrained Supports [None]

Revision History

    01a_070593 — Initial Version; 1st high level pass

Level # 2	External Constraint Template	Version # and Date 01a_070593
	Template Type	

Name	Unique Identifier
Company-X Technical Documentation Guidelines	SWAT_CE_PPG_X−TECHDOC

Purpose

    Provide employees with guidelines, examples, and forms to support technical document development.

Comments

Special Form Of (parent)	Additional States/Description
SWAT_CE_PPG	[None]

General Form Of (list children)	Constrained Events	Constrained Throughputs
[None]	SWAT	[All technical documentation subject to or participating in the SWAT formal inspection process.]
		Constrained Supports [None]

Revision History

    01a_070593 − Initial Version; 1st high level pass

Step 4. Construct Role Templates. The process analyst references his notes and some existing documentation collected. He constructs the following indented list for representing roles:

Super Moderator
Inspection SWAT Team
       Moderator
       Producer
       Inspector
              Key Inspector
              Regular Inspector
       Scribe
       Reader

The analyst then builds a list of the default set of role states (to be referenced when considering the addition of new role states):

Available_Exclusively
Available_Shared
Not_Available
Disabled
Suspended

| Level #

1 | Role Template

Template Type | Version # and Date

01a_070593 |
| --- | --- | --- |

| Name

Super Moderator | | Unique Identifier

SWAT_SP_S−MOD |

Purpose

Provides overall management of, and expert participation in, the SWAT Inspection Process.

Comments

Generally, the super moderator has overall responsibility to develop, manage, and steadily expand the SWAT formal inspection process. This person also provides regular status reports to management and incorporates their recommendations. May participate in any SWAT role.

Additional States/Descriptions

[None]

**Associated Authority
(via Internal Constraints)
and Applicable Events**

[The super moderator can exercise any of the authority granted by the internal constraints except those which are specifically mapped to upper management.]

Part Of

[None]

Supported Events

[TBD]

Composed Of

[None]

External Constraints

SWAT_CE_PPG

Revision History

01a_070593 − Initial Version; 1st high level pass

Level # 1	Role Template —————————————————— Template Type	Version # and Date 01a_070593

Name
    Inspection Team
	Unique Identifier
	    SWAT_SP_TEAM

Purpose
    Participates, as a group, in various events of the SWAT inspection process.

Comments
    As indicated below, the team has certain minimum and maximum constraints on the number of people participating in the various roles of a specific team.

Additional States/Descriptions

[None]

**Associated Authority
(via Internal Constraints)
and Applicable Events**

[TBD]

Part Of

[None]

Supported Events

[TBD]

Composed Of

SWAT_SP_TEAM
        MOD (1)
        READ (1)
        PROD (1..N)
        INSP (4..6)
                KEY (2..4)
                REG (0..4)
        SCRIBE (1)

[Note: only a key inspector can serve in the role of a reader.]

External Constraints

SWAT_CE_PPG

Revision History

01a_070593 — Initial Version; 1st high level pass

Level # 2	Role Template Template Type	Version # and Date 01a_070593

Name Moderator	Unique Identifier SWAT_SP_TEAM_MOD

Purpose

Manages the inspection of an artifact throughout the stages of the SWAT inspection.

Comments

The moderator works closely and coordinates with the super moderator. Moderators are key inspectors that have received additional training to qualify them for moderating inspections. Moderators are chosen (by the super moderator) on an inspection by inspection basis.

Additional States/Descriptions

[None]

**Associated Authority
(via Internal Constraints)
and Applicable Events**

[TBD]

Part Of

SWAT_SP_TEAM

Supported Events

[TBD]

Composed Of

[None]

External Constraints

SWAT_CE_PPG

Revision History

01a_070593 — Initial Version; 1st high level pass

Level # 2	Role Template Template Type	Version # and Date 01a_070593

Name Reader	Unique Identifier SWAT_SP_TEAM_READ

Purpose

Presents the artifact to be inspected both during the overview and during the inspection meeting.

Comments

For any item being inspected, the reader/presenter must also have been one of the key inspectors.

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [TBD]

Part Of SWAT_SP_TEAM	Supported Events SWAT_ET_INSP_OVER SWAT_ET_INSP_I-MTG	
Composed Of [None]		External Constraints SWAT_CE_PPG

Revision History

01a_070593 — Initial Version; 1st high level pass

Level # 2	Role Template _____ Template Type	Version # and Date 01a_070593

Name Producer	Unique Identifier SWAT_SP_TEAM_PROD

Purpose

    Generally assists the inspector and makes the necessary defect corrections.

Comments

    If there are more than three producers for a given artifact, then one or two "key" producers should be selected to represent the artifact, and then coordinate, perform, or otherwise take responsibility for assuring that defects are properly removed from the inspected item.

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [TBD]

Part Of SWAT_SP_TEAM	Supported Events [TBD]	
Composed Of [None]		
		External Constraints SWAT_CE_PPG

Revision History

    01a_070593 — Initial Version; 1st high level pass

Level # 2	Role Template _____ Template Type	Version # and Date 01a_070593

Name Inspector	Unique Identifier SWAT_SP_TEAM_INSP

Purpose

　　Performs detailed analysis of inspection artifact and provides documented findings of defects.

Comments

　　Note on the supporting documentation that there are two types of inspectors.

Additional States/Descriptions

　　　　　　[None]

**Associated Authority
(via Internal Constraints)
and Applicable Events**

　　　　　　[TBD]

Part Of

　　SWAT_SP_TEAM

Supported Events

　　　　[TBD]

Composed Of

SWAT_SP_ TEAM_
　INSP_KEY

SWAT_SP_ TEAM_
　INSP_REG

External Constraints

　　SWAT_CE_PPG

Revision History

　　01a_070593 — Initial Version; 1st high level pass

| Level #

3 | Role Template

Template Type | Version # and Date

01a_070593 |
|---|---|---|

| Name
   Key Inspector | Unique Identifier
   SWAT_SP_TEAM_INSP_KEY |
|---|---|

Purpose
   Performs expert detailed analysis of inspection artifact and provides documented findings of defects.

Comments

   May also be asked to perform in the role of a reader. (A key inspector that has performed all nonmoderator roles at a formal inspection is eligible for training as a moderator.)

Additional States/Descriptions

      [None]

**Associated Authority
(via Internal Constraints)
and Applicable Events**

      [TBD]

Part Of

   SWAT_SP_TEAM_INSP

Supported Events

      [TBD]

Composed Of

      [None]

External Constraints

   SWAT_CE_PPG

Revision History

   01a_070593 — Initial Version; 1st high level pass

Level # 3	Role Template _____ Template Type	Version # and Date 01a_070593

Name
    Regular Inspector
	Unique Identifier
	    SWAT_SP_TEAM_INSP_REG

Purpose
    Performs analysis of inspection artifacts, and provides documented findings of defects.

Comments

Additional States/Descriptions

      [None]

**Associated Authority
(via Internal Constraints)
and Applicable Events**

      [TBD]

Part Of

    SWAT_SP_TEAM_INSP

Supported Events

      [TBD]

Composed Of

      [None]

External Constraints

    SWAT_CE_PPG

Revision History

    01a_070593 — Initial Version; 1st high level pass

Level # 2	Role Template _____ Template Type	Version # and Date 01a_070593

Name Scribe	Unique Identifier SWAT_SP_TEAM_SCRIBE

Purpose

Takes detailed notes, using an inspection log, of all defects discussed at the inspection meeting.

Comments

This role is NOT to be performed by clerical, administrative, secretarial, or support staff. This is to be a technical person performing in a technical capacity.

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [None]
Part Of SWAT_SP_TEAM	**Supported Events** SWAT_ET_INSP_I−MTG
Composed Of [None]	
	External Constraints SWAT_CE_PPG

Revision History

01a_070593 − Initial Version; 1st high level pass

Step 5. Construct Product Templates. The process analyst references his work schedule and sees that the schedule calls for product templates to be constructed next. Having had no special problems with the proceeding steps, the analyst decides that there is no reason to deviate from that plan at this point and proceeds with establishing the Product class templates.

As with the other steps, the analyst begins by establishing an indented list of the main products that are created by, or otherwise manipulated by, the SWAT inspection process. The initial indented list is as follows:

Inspectable Artifacts
    Ada Software
        Ada Package Specifications
        Ada Package Specifications and Package Bodies
    Technical Guides
        End-User Guides
        System Administration Guides
        Installation Guides
        Miscellaneous Technical Documents
Required Inspection Documentation
    Error Logs
        Trivial Error Log
        Preparation Error Log
    Inspection Summary Report
        Software Inspection Summary Report
        Technical Documentation Inspection Summary Report
Inspection Memos
    Inspection Invitation Memo
    Inspection Exit Memo
    Inspection Close Memo

As before, the analyst takes a moment to note the recommended set of default states pertaining to products in general:

Default Set of Product States:
    Unauthorized
    Authorized
    In_Progress
    In_Rework
    Disabled
    Suspended
    Cancelled
    Completed

Level # 1	Product Template Template Type	Version # and Date 01a_070693

Name Inspectable Artifact		Unique Identifier SWAT_TP_ART

Purpose

These products are characterized by all being susceptible to the SWAT inspection process.

Comments

Currently, the SWAT inspection process focuses upon Ada artifacts and technical documentation artifacts. However, standards, etc., exist to support a wider variety of items. Expansion of the SWAT program will occur as more people are trained in this process.

Additional States Needs Inspection Released	Descriptions Product has not yet passed inspection. Product has passed inspection and can be returned into main-stream processing.	Part Of [None]
		Composed Of SWAT_TP_ART_ADA SWAT_TP_ART_GUIDE
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

Level # 2	Product Template <hr>Template Type	Version # and Date 01a_070693
Name Inspectable Ada Artifact		**Unique Identifier** SWAT_TP_ART_ADA
Purpose Software for delivered systems.		
Comments These products are all Ada code modules: package specs, bodies, or both.		

Additional States [None]	Descriptions [None]	**Part Of** SWAT_TP_ART
		Composed Of SWAT_TP_ART_ ADA_SPEC SWAT_TP_ART_ ADA_BODY
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_SPC

Revision History

01a_070693 — Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693

Name Ada Package Specification	Unique Identifier SWAT_TP_ART_ADA_SPEC

Purpose

    Ada Package specifications primarily assist with verifying the systems design.

Comments

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_ADA
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_SPC

Revision History

    01a_070693 — Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693
Name Ada Package Body		**Unique Identifier** SWAT_TP_ART_ADA_BODY
Purpose Ada Package Body (with the specification implied or explicit) for product software support.		
Comments		

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_ADA
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_SPC

Revision History

01a_070693 − Initial Version; 1st high level pass

Level # 2	Product Template Template Type	Version # and Date 01a_070693

Name
    Technical Guides
	Unique Identifier
	    SWAT_TP_ART_GUIDE

Purpose

    Explain and support all aspects of installing, using, and maintaining the Ada software product.

Comments

    Expanding the types of guides being inspected is an ongoing area of growth. When necessary, new types of documentation can be handled as "miscellaneous" technical guides until more formal designations and support can be developed.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_ART

		Composed Of
		SWAT_TP_ART_GUIDE_
		    USER
		
		SWAT_TP_ART_GUIDE_
		    INSTALL
		
		SWAT_TP_ART_GUIDE_
		    SA
		
		SWAT_TP_ART_GUIDE_
		    MISC

Evolves From Events	**State Transitions (Event/Step)**	
[TBD]	[TBD]	**External Constraints** SWAT_CE_PPG SWAT_CE_PPG_ X−TECHDOC

Revision History

    01a_070693 − Initial Version; 1st high level pass

Level # 3	Product Template _____ Template Type	Version # and Date 01a_070693

Name
    Technical User Guide
	Unique Identifier
	    SWAT_TP_ART_GUIDE_USER

Purpose
    Explains and supports all aspects of using the Ada software product.

Comments

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_GUIDE

		Composed Of [None]

Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	

		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X–TECHDOC

Revision History

    01a_070693 – Initial Version; 1st high level pass

Level # 3	Product Template _____ Template Type	Version # and Date 01a_070693

Name
    System's Administrator Guide
	Unique Identifier
	    SWAT_TP_ART_GUIDE_SA

Purpose
    Explains and supports all aspects of supporting the Ada software product.

Comments

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_GUIDE
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X–TECHDOC

Revision History

    01a_070693 — Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693
Name Installation Guide		**Unique Identifier** SWAT_TP_ART_GUIDE_INSTALL
Purpose Explains all aspects of installing (and removing) the Ada software product.		
Comments		

Additional States [None]	**Descriptions** [None]	**Part Of** SWAT_TP_ART_GUIDE
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X−TECHDOC

Revision History 01a_070693 − Initial Version; 1st high level pass

Level # 3	Product Template _____ Template Type	Version # and Date 01a_070693

Name Miscellaneous Technical Guide	Unique Identifier SWAT_TP_ART_GUIDE_MISC

Purpose

Explains any essential information not already provided by other technical documentation.

Comments

This is essentially a catchall designation. If a technical guide does not fit under any of the other categories, it can be designated and inspected as a miscellaneous technical guide.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_GUIDE
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X–TECHDOC

Revision History

01a_070693 – Initial Version; 1st high level pass

Level # 1	Product Template	Version # and Date
	Template Type	01a_070693

Name	Unique Identifier
Required Inspection Documentation	SWAT_TP_DOC

Purpose

Provides a paper audit trail of the essential functions and events of the inspection process.

Comments

All of the documents represented as "children" of this class (regardless how many generations removed) must be produced by some event in the inspection process. These are mandatory, and if the model does not capture where they are produced, then re-examine the event templates and correct.

Additional States	Descriptions	Part Of
[None]	[None]	[None]
		Composed Of
		SWAT_TP_ DOC_LOG
Evolves From Events	**State Transitions (Event/Step)**	SWAT_TP_ DOC_RPT
[TBD]	[TBD]	
		External Constraints
		SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

Level # 2	Product Template Template Type	Version # and Date 01a_070693

Name Error Logs	Unique Identifier SWAT_TP_DOC_LOG

Purpose

Log defects, by location, type, and severity, of inspected items.

Comments

Currently, standardized error logs are used; however, customized or artifact-specific logs are being considered.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC
		Composed Of SWAT_TP_DOC_ LOG_TRIV SWAT_TP_DOC_ LOG_PREP
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693

Name Trivial Error Log	Unique Identifier SWAT_TP_DOC_LOG_TRIV

Purpose

Logs trivial defects by location, type, and severity of inspected items.

Comments

These defects are not discussed during the inspection meeting. Instead, the logs are turned in at the inspection meeting by the inspectors. They are delivered to the producers for use in correcting the item under inspection. These logs also become a part of permanent records.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC_LOG
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693

Name
    Preparation Error Log
	Unique Identifier
	    SWAT_TP_DOC_LOG_PREP

Purpose

    Logs nontrivial defects by location, type, and severity found during inspection.

Comments

    Preparation error logs contain all nontrivial defects during inspection of the artifact. These logs
    are used (by the inspector who created them) during the inspection meeting. Defects are noted by
    the scribe (during the meeting) and the compiled results given to the producer(s).

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC_LOG
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

    01a_070693 — Initial Version; 1st high level pass

Level # 2	Product Template Template Type	Version # and Date 01a_070693

Name Inspection Summary Reports	Unique Identifier SWAT_TP_DOC_RPT

Purpose

Provide a variety of useful, summarized information for management and for archival.

Comments

Typically, summary reports are tailored to the artifacts subject to inspection. Currently, however, only two such tailored report formats exist (shown below). Work is being done on considering more highly tailored summary reports.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_DOC
		Composed Of SWAT_TP_DOC_ RPT_S−WARE
Evolves From Events	**State Transitions (Event/Step)**	SWAT_TP_DOC_ RPT_T−DOC
[TBD]	[TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 − Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693

Name Inspection Software Summary Report	Unique Identifier SWAT_TP_DOC_RPT_S—WARE

Purpose

Provides a variety of useful, summarized information for management and for archival.

Comments

This summary report is focused specifically on collecting data which supports the metrics developed for monitoring not only product quality but also for monitoring various characteristics of the SWAT formal inspection process itself.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC_RPT
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693

Name Inspection Technical Document Summary Report	Unique Identifier SWAT_TP_DOC_RPT_T−DOC

Purpose

   Provides a variety of useful, summarized information for management and for archival.

Comments

   This summary report is focused specifically on collecting data which supports the metrics developed
   for monitoring not only product quality but also for monitoring various characteristics of the SWAT
   formal inspection process itself.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC_RPT
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

   01a_070693 − Initial Version; 1st high level pass

Level #	Product Template	Version # and Date
1	Template Type	01a_070693

Name	Unique Identifier
Inspection Memorandums	SWAT_TP_MEMO

Purpose

Provide written announcements of event transitions within the SWAT inspection process.

Comments

Memorandums all have basic "boilerplate" versions used to create the memos. The memos are used to provide details to formalize announcing what is about to happen and to formalize acknowledging what has happened. As such, these serve as excellent audit trails of transpiring events.

Additional States	Descriptions	Part Of
[None]	[None]	[None]

Composed Of

SWAT_TP_MEMO
I–INV

SWAT_TP_MEMO
I–EXIT

SWAT_TP_MEMO
I–DONE

Evolves From Events	State Transitions (Event/Step)
[TBD]	[TBD]

External Constraints

SWAT_CE_PPG

Revision History

01a_070693 – Initial Version; 1st high level pass

Level # 2	**Product Template** Template Type	Version # and Date 01a_070693

Name Inspection Invitation Memorandum	Unique Identifier SWAT_TP_MEMO_I−INV

Purpose

    Provides all participants with a written request to attend an inspection meeting.

Comments

    These memorandums detail such information as the name of the inspected artifact, the date and time of the inspection, where the inspection will be conducted, and who to contact for further information.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_MEMO
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

    01a_070693 − Initial Version; 1st high level pass

Level # 2	Product Template Template Type	Version # and Date 01a_070693

Name Inspection Exit Memorandum	Unique Identifier SWAT_TP_MEMO_I–EXIT

Purpose

Provides all participants with a written acknowledgment of the end of an inspection meeting.

Comments

Included on this memorandum are any action items decided during the meeting and a reiteration of what is expected from whom. Also noted are any relevant details regarding (possible) reinspections.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_MEMO
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 – Initial Version; 1st high level pass

Level # 2	Product Template Template Type	Version # and Date 01a_070693
Name Inspection Done Memorandum		**Unique Identifier** SWAT_TP_MEMO_I–DONE

Purpose

Provides written acknowledgement that a product was released from the inspection process.

Comments

This memorandum not only goes to participants, but it is also forwarded to all interested or relevant managers. Most important, this memorandum details the release status of the inspected artifact (i.e., it passed inspection or failed inspection, etc.).

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_MEMO

		Composed Of
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	[None]
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

Step 6. Bind External/Role/Product Templates with Event Templates. At this point, the analyst feels he has a reasonable "firstcut" representation of a high level view of the SWAT inspection process. Before proceeding with developing additional templates at the class level, the analyst prefers to bind the existing templates and to evaluate the consistency of the model to date.

The following pages show the updated templates. Note that the revision section of the template are a simple indicator of which templates were updated.

Level # 1	Management Template Template Type	Version # and Date 01b_070793

Name
    SWAT_Process
	Unique Identifier
	    SWAT

Purpose
    Inspect SWAT Team Process.

Comments

Additional States

Entry Criteria	Internal Processing	Exit Criteria

(SWAT_CE_VP::
        Compliance_Achieved
 	
OR
 	
SWAT_CE_VP::
        Compliance_Waived)
	In Parallel Do...
	    SWAT state is In_Progress
	    Perform SWAT_EA_INSPECT
	    Perform SWAT_ET_CAUSAL
	    Perform SWAT_EA_PROC−IMP
	Until
	    (SWAT_CE_VP::
	        Require_Cessation
	            OR
	    [Moderator asks to cancel])
	SWAT state is Cancelled
		(SWAT_CE_VP::
		        Require_Cessation
		
		OR
		
		SWAT_CE_VP::
		        Approve_Cessation)

Throughputs	Supports		Internal Constraints	External Constraints
[None]	SWAT_SP_ S−MOD			SWAT_CE_ VP

Parent Event(s)

Child Events

    SWAT_EA_INSP
    SWAT_ET_CAUSAL
    SWAT_EA_PROC−IMP

Revision History

    01a_070493 − Initial Version; 1st high level pass
    01b_070793 − Initial Version; Binding templates

Level # 2	Management Template Template Type	Version # and Date 01b_070793

Name Inspection	Unique Identifier SWAT_EA_INSP

Purpose

This is the overall inspection activity and contains, as shown below, several stages.

Comments

[None]

Additional States

[None]

Entry Criteria	Internal Processing	Exit Criteria

SWAT::In_Progress
 	
SWAT_TP_ART::
  Needs_Inspection
	SWAT_EA_INSP state is In_Progress
	1) Perform SWAT_ET_INSP_PLAN
	  Perform SWAT_ET_INSP_OVER
	  Perform SWAT_ET_INSP_PREP
	  Perform SWAT_ET_INSP_I-MTG
	  Perform SWAT_ET_INSP_REWORK
	  Perform SWAT_ET_INSP_FOLLOW
	  If Reinspection Needed
	    Go To (1)
	  SWAT_EA_INSP state is Complete
		SWAT_EA_INSP::Complete
		
		SWAT_TP_ART::
		  Released

Throughputs	Supports		Internal Constraints	External Constraints

SWAT_TP_
  ART
SWAT_TP_
  DOC
 	
SWAT_TP_
  MEMO_
  I-INV
  I-EXIT
  I-ONE
	SWAT_SP_
	  TEAM_
	  MOD
	
	SWAT_SP_
	  TEAM
			[TBD]	SWAT_CE_
				  PPG

Parent Event(s)
SWAT

Child Events
SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_ET_INSP_I-MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW

Revision History

01a_070493 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level # 3	Production Template Template Type	Version # and Date 01b_070793

Name	Unique Identifier
Planning	SWAT_ET_INSP_PLAN

Purpose

    This represents the planning stage of the inspection activity.

Comments

Entry Criteria	Internal Processing	Exit Criteria

SWAT_EA_INSP::In_Progress	SWAT_ET_INSP_PLAN state is
	    In_Progress
	Evaluate the Item to be Inspected
	Estimate the expected inspection rate
	Construct a schedule
	Assign personnel to the schedule
	Confirm personnel availability
	Adjust/Redo schedule as necessary
	SWAT_ET_INSP_PLAN state is
	    Complete
		SWAT_ET_INSP_PLAN::
		    Complete

Products	Supports		Constraints
SWAT_TP_ ART	SWAT_SP_ S−MOD		
	SWAT_SP_ TEAM_ MOD	**Parent Event(s)** SWAT_EA_INSP	
		Child Events	
Research			

Revision History

    01a_070493 − Initial Version; 1st high level pass
    01b_070793 − Initial Version; Binding templates

Level # 3	Production Template Template Type	Version # and Date 01b_070793

Name Overview	Unique Identifier SWAT_ET_INSP_OVER

Purpose

This represents the overview stage of the inspection activity.

Comments

Although the Supports section shows that producers participate in this activity, this is optional (in practice, however, producers almost always attend). Note that the reader presents the overview.

Entry Criteria	Internal Processing	Exit Criteria
SWAT_EA_INSP::In_Progress and SWAT_EA_INSP_PLAN:: Complete	SWAT_ET_INSP_OVER state is In_Progress Introduce Objective Introduce Members Have Reader present the item to be inspected Review preparation time and schedule Dismiss meeting SWAT_ET_INSP_OVER state is Complete	SWAT_ET_INSP_OVER:: Complete

Products	Supports		Constraints
SWAT_TP_ ART	SWAT_SP_ TEAM_ MOD		
	SWAT_SP_ TEAM_ READ	**Parent Event(s)** SWAT_EA_INSP	
	SWAT_SP_ TEAM_ PROD	**Child Events**	
Research	SWAT_SP_ TEAM_ INSP		

Revision History

01a_070493 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level # 3	Production Template Template Type	Version # and Date 01b_070793

Name Preparation	Unique Identifier SWAT_ET_INSP_PREP

Purpose

This represents the preparation stage of the inspection activity.

Comments

The preparation stage is characterized by all inspectors (both key and otherwise) reviewing the material for defects and completing trivial error logs and preparation error logs (both of which are brought to the actual inspection meeting).

Entry Criteria SWAT_EA_INSP::In_Progress and SWAT_EA_INSP_OVER:: Complete	Internal Processing SWAT_ET_INSP_PREP state is In_Progress Disperse material to be inspected and associated support forms Each Inspector Does: Review of Relevant Standards Personal Inspection of Item Completes Trivial Error Log Completes Preparation Log Completes Preparation_Exit Form (Checklist) END [Each Inspector Does] Moderator Collects/Verifies Checklists Moderator distributes Inspection Invitation letter SWAT_ET_INSP_PREP state is Complete	Exit Criteria SWAT_ET_INSP_PREP:: Complete

Products SWAT_TP_ ART SWAT_TP_ DOC_LOG_ TRIV PREP	Supports SWAT_SP_ TEAM_ MOD		Constraints SWAT_CE_PPG
		Parent Event(s) SWAT_EA_INSP	
Research		Child Events	

Revision History

01a_070493 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level # 3	Production Template _____ Template Type	Version # and Date 01b_070793

Name
 	
    Inspection_Meeting
	Unique Identifier
	
	    SWAT_ET_INSP_I−MTG

Purpose

    This represents the actual inspection (by the group) of the item to be inspected.

Comments

Entry Criteria

SWAT_EA_INSP::In_Progress
       and
SWAT_EA_INSP_PREP::
       Complete

Internal Processing

Produce SWAT_TP_MEMO_I−INV
SWAT_ET_INSP_I−MTG state is
    In_Progress
Take Attendance
When (sufficient attendance)
    Review Purpose
    Collect SWAT_TP_DOC_
        LOG_TRIV
    Perform Product Inspection
    Review Findings
    Determine if reinspection req'd
END [When]
Produce SWAT_TP_MEMO_I−EXIT
SWAT_ET_INSP_I−MTG state is
    Complete

Exit Criteria

SWAT_ET_INSP_I−MTG::
    Complete

Products
SWAT_TP_ART
SWAT_TP_
  DOC_LOG_
  TRIV_
  PREP
SWAT_TP_
  DOC_RPT
SWAT_TP_
  MEMO_
  I−INV
  I−EXIT
	Supports
	
	SWAT_SP_
	TEAM_
	(all)

Parent Event(s)
    SWAT_EA_INSP

Child Events

Constraints

    SWAT_CE_PPG

Research

Revision History

    01a_070493 − Initial Version; 1st high level pass

    01b_070793 − Initial Version; Binding templates

Level # 3	Production Template Template Type	Version # and Date 01b_070793

Name Rework	Unique Identifier SWAT_ET_INSP_REWORK

Purpose
Perform the necessary changes and updates to remove defects detected during inspection.

Comments

Entry Criteria
 	
SWAT_EA_INSP::In_Progress
and
SWAT_EA_INSP_I−MTG::
Complete
	Internal Processing
	
	SWAT_ET_INSP_REWORK state is
	   In_Progress
	Each SWAT_SP_TEAM_PROD:
	   Evaluates pertinent defects
	   Alters product only to correct
	      the defect
	   Notes location and scope of
	      changes
	   If Req'd: makes before and after
	      listings until all defects
	      addressed ˙
		Exit Criteria
		
		SWAT_ET_INSP_REWORK::
		Complete

Products
SWAT_TP_
 ART
 	
SWAT_TP_
 DOC_LOG_
 TRIV
 PREP
 	
SWAT_TP_
 MEMO_
 I−DONE
	Supports
	
	SWAT_SP_
	TEAM_
	MOD
	
	SWAT_SP_
	TEAM_
	PROD
		SWAT_SP_TEAM_MOD notified that
		   all changes are complete
		SWAT_ET_INSP_REWORK state is
		   Complete
			Constraints

		Parent Event(s) SWAT_EA_INSP	

Research		Child Events	

Revision History

   01a_070493 − Initial Version; 1st high level pass

   01b_070793 − Initial Version; Binding templates

Level #	Production Template	Version # and Date
3	_Template Type_	01b_070793

Name	Unique Identifier
Follow-up	SWAT_ET_INSP_FOLLOW

Purpose

Verify that the authors/producers have addressed all defects detected by inspectors.

Comments

Occasionally, the nature of the changes to the inspected item are such that the moderator may elect to call a reinspection (even though originally it might not have seemed necessary). If no reinspection is necessary, this is the last (regular) stage of the inspection process.

Entry Criteria

SWAT_EA_INSP::In_Progress
and
SWAT_EA_INSP_I−MTG::
Complete

Internal Processing

SWAT_ET_INSP_I−FOLLOW state is In_Progress
Meet with the author(s)
Discuss nature and scope of changes
Evaluate each change
Confirm only authorized changes made
Determine whether a reinspection needs to be conducted
Moderator completes the follow-up checklist
Produce SWAT_TP_MEMO_I−DONE
SWAT_ET_INSP_FOLLOW state is Complete

Exit Criteria

SWAT_ET_INSP_FOLLOW::
Complete

Products

$WAT_TP_
ART

SWAT_TP_
DOC_LOG_
TRIV
PREP

SWAT_TP_
MEMO_
I−DONE

Research

Supports

SWAT_SP_
TEAM_
MOD

SWAT_SP_
TEAM_
PROD

Parent Event(s)

SWAT_EA_INSP

Child Events

Constraints

Revision History

01a_070493 − Initial Version; 1st high level pass

01b_070793 − Initial Version; Binding templates

Level # 2	Production Template Template Type	Version # and Date 01b_070793

Name Causal Analysis	Unique Identifier SWAT_ET_CAUSAL

Purpose
Attempt to identify common problems and suggest process changes to originate their cause.

Comments

Entry Criteria SWAT::In_Progress	Internal Processing SWAT_ET_CAUSAL state is In_Progress Take attendance Review outstanding issues from prior causal analysis meetings Report results of trend analysis efforts Solicit perceived problem areas from attendees Prioritize problem areas Discuss/determine process changes to reduce/remove root causes At end of meeting; review findings Dismiss meeting Compile Causal Analysis Report SWAT_ET_CAUSAL state is Complete	Exit Criteria SWAT_ET_CAUSAL:: Complete

Products SWAT_TP_ DOC_RPT	Supports SWAT_SP_ S-MOD SWAT_SP_ TEAM_ (any)		Constraints
		Parent Event(s) SWAT	
Research		**Child Events**	

Revision History

01a_070493 – Initial Version; 1st high level pass

01b_070793 – Initial Version; Binding templates

Level # 2	Production Template _____ Template Type	Version # and Date 01b_070793

Name Inspection_Process_Improvement	Unique Identifier SWAT_ET_PROC-IMP

Purpose

Improve the way in which inspections are actually conducted.

Comments

Inspection process improvement efforts are not constrained to any particular inspection subprocess. Inspections, causal analysis, and even this process are all potentially subject to process improvement efforts.

Entry Criteria	Internal Processing	Exit Criteria

SWAT::In_Progress	SWAT_ET_PROC-IMP state is
	   In_Progress
	Take attendance
	Review outstanding issues from prior
	   process improvement meetings
	Solicit suggestions from attendees
	Organize suggestions by domain
	Evaluate cost/benefit
	Prioritize by estimated cost/benefit
	Investigate alternatives
	Compile final recommendations
	At end of meeting; review findings
		SWAT_ET_PROC-IMP::
		   Complete

Products	Supports	Dismiss meeting
		Compile Inspection Process Improvement
		   Report
		SWAT_ET_PROC-IMP state is
		   Complete
			Constraints
			
			SWAT_CE_PPG

SWAT_TP_
  DOC_RPT
	SWAT_SP_
	  S-MOD
		Parent Event(s)
		   SWAT	

	SWAT_SP_ TEAM_ (any)	**Child Events**	
Re earch			

Revision History

   01a_070493 — Initial Version; 1st high level pass
   01b_070793 — Initial Version; Binding templates

Level # 1	External Constraint Template Template Type	Version # and Date 01a_070593

Name VP Quality Assurance		Unique Identifier SWAT_CE_VP

Purpose

Monitors and determines whether the SWAT inspection effort should continue or cease.

Comments

The VP of Quality Assurance has overall control and responsibility for the SWAT Inspection process.

Special Form Of (parent) [None]	Additional States/Description Require_Cessation This state is used to represent the VP declaring the SWAT program be ceased. Approve_Cessation This state represents VP approval of a super moderator request to cease the SWAT inspection process.	

General Form Of (list children) [None]	Constrained Events SWAT	Constrained Throughputs [None]
		Constrained Supports [None]

Revision History

01a_070593 — Initial Version; 1st high level pass

Level # 1	External Constraint Template Template Type	Version # and Date 01b_070793

Name	Unique Identifier
Policies, Procedures, and Guidelines	SWAT_CE_PPG

Purpose

Provide written reference standards applicable to conducting the SWAT inspection process.

Comments

All the applicable documented standards necessary for performing the SWAT inspection process are under this group of constraints.

Special Form Of (parent)
 	
[None]
	Additional States/Description
	
	Require_Cessation
	    This state is used to represent the VP declaring the SWAT program be ceased.
	Approve_Cessation
	    This state represents VP approval of a super moderator request to cease the SWAT inspection process.	

General Form Of (list children)
 	
SWAT_CE_PPG_
      ADA
 	
SWAT_CE_PPG_
      X-TECH-
DOC
	Constrained Events
	
	SWAT
	
	SWAT_EA_INSP
	
	SWAT_ET_INSP_PREP
	
	SWAT_ET_INSP_I-MTG
	
	SWAT_ET_INSP_PROC-IMP
		Constrained Throughputs
		
		SWAT_TP_ART_ADA_(all)
		
		SWAT_TP_ART_GUIDE_(all)
		
		SWAT_TP_DOC_LOG_(all)
		
		SWAT_TP_DOC_RPT_(all)
		
		SWAT_TP_MEM_(all)

		Constrained Supports
		
		SWAT_SP_S-MOD
		
		SWAT_SP_TEAM_
		      MOD
		      READ
		      PROD
		      INSP_(all)
		      SCRIBE
		[Note; Ask Human Resources for further details on job descriptions.]

Revision History

01a_070593 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level # 2	External Constraint Template Template Type	Version # and Date 01b_070793

Name
    SPC Ada Quality and Style Guide
		Unique Identifier
		    SWAT_CE_PPG_ADA

Purpose
    Provides Ada-specific reference standards applicable to Ada throughputs/products.

Comments

Special Form Of
(parent)
 	
    SWAT_CE_PPG
	Additional States/Description
	
	            [None]	

General Form Of
(list children)
 	
    [None]
	Constrained Events
	
	            SWAT
		Constrained Throughputs
		
		    SWAT_TP_ART_ADA
		
		    SWAT_TP_ART_ADA_SPEC
		
		    SWAT_TP_ART_ADA_BODY

		Constrained Supports [None]

Revision History

    01a_070593 — Initial Version; 1st high level pass
    01b_070793 — Initial Version; Binding templates

Level # 2	External Constraint Template Template Type	Version # and Date 01b_070793

Name Company-X Technical Documentation Guidelines	Unique Identifier SWAT_CE_PPG_X–TECHDOC

Purpose
Provide employees with guidelines, examples, and forms to support technical document development.

Comments

Special Form Of (parent) SWAT_CE_PPG	Additional States/Description [None]

General Form Of (list children) [None]	Constrained Events SWAT	Constrained Throughputs SWAT_TP_ART_GUIDE SWAT_TP_ART_GUIDE_USER SWAT_TP_ART_GUIDE_SA SWAT_TP_ART_GUIDE_INSTALL SWAT_TP_ART_GUIDE_MISC
		Constrained Supports [None]

Revision History

01a_070593 – Initial Version; 1st high level pass

01b_070793 – Initial Version; Binding templates

| Level #

1 | Role Template

Template Type | Version # and Date

01b_070793 |
|---|---|---|

| **Name**

Super Moderator | **Unique Identifier**

SWAT_SP_S−MOD |
|---|---|

Purpose

Provides overall management of, and expert participation in, the SWAT Inspection Process.

Comments

Generally, the super moderator has overall responsibility to develop, manage, and steadily expand the SWAT formal inspection process. This person also provides regular status reports to management and incorporates their recommendations. May participate in any SWAT role.

Additional States/Descriptions

[None]

**Associated Authority
(via Internal Constraints)
and Applicable Events**

[The super moderator can exercise any of the authority granted by the internal constraints except those which are specifically mapped to upper-management.]

Part Of

[None]

Supported Events

SWAT

SWAT_ET_INSP_PLAN

SWAT_ET_CAUSAL

SWAT_ET_PROC−IMP

Composed Of

[None]

External Constraints

SWAT_CE_PPG

Revision History

01a_070593 − Initial Version; 1st high level pass
01b_070793 − Initial Version; Binding templates

Level # 1	Role Template Template Type	Version # and Date 01b_070793

Name Inspection Team	Unique Identifier SWAT_SP_TEAM

Purpose

Participates, as a group, in various events of the SWAT inspection process.

Comments

As indicated below, the team has certain minimum and maximum constraints on the number of people participating in the various roles of a specific team.

Additional States/Descriptions

[None]

Associated Authority (via Internal Constraints) and Applicable Events

[TBD]

Part Of

[None]

Supported Events

SWAT_EA_INSP

SWAT_ET_INSP_I—MTG

SWAT_ET_CAUSAL :

SWAT_ET_PROC—IMP

Composed Of

SWAT_SP_
    MOD (1)
    READ (1)
    PROD (1..N)
    INSP (4..6)
        KEY (2..4)
        REG (0..4)
    SCRIBE (1)

[Note: Only a key inspector can serve in the role of a reader.]

External Constraints

SWAT_CE_PPG

Revision History

01a_070593 — Initial Version; 1st high level pass
01b_070793 — Initial Version; Binding templates

Level # 2	Role Template Template Type	Version # and Date 01b_070793

Name Moderator	Unique Identifier SWAT_SP_TEAM_MOD

Purpose

Manages the inspection of some artifact throughout the stages of the SWAT inspection.

Comments

Works closely and coordinates with the super moderator. Moderators are key inspectors that have received additional training to quality them for moderating inspections. Moderators are chosen (by the super moderator) on an inspection by inspection basis.

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [TBD]

Part Of SWAT_SP_TEAM	Supported Events SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER	
Composed Of [None]	SWAT_ET_INSP_PREP SWAT_ET_INSP_I-MTG SWAT_ET_INSP_FOLLOW SWAT_ET_INSP_CAUSAL SWAT_ET_INSP_PROC-IMP	
		External Constraints SWAT_CE_PPG

Revision History

01a_070593 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level # 2	Role Template Template Type	Version # and Date 01b_070793

Name Reader	Unique Identifier SWAT_SP_TEAM_READ

Purpose

Presents the artifact to be inspected both during the overview and during the inspection meeting.

Comments

For any item being inspected, the reader/presenter must also have been one of the key inspectors.

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [TBD]

Part Of SWAT_SP_TEAM	Supported Events SWAT_ET_INSP_OVER SWAT_ET_INSP_I–MTG SWAT_ET_INSP_CAUSAL SWAT_ET_INSP_PROC–IMP	
Composed Of [None]		External Constraints SWAT_CE_PPG

Revision History

01a_070593 – Initial Version; 1st high level pass

01b_070793 – Initial Version; Binding templates

Level # 2	Role Template _____ Template Type	Version # and Date 01b_070793

Name
 	
    Producer
	Unique Identifier
	
	    SWAT_SP_TEAM_PROD

Purpose

    Generally assists with the inspection and makes the necessary defect corrections.

Comments

    If there are more than three producers for a given artifact, one or two key producers should be selected to represent the artifact. He then coordinates, performs, or otherwise takes responsibility for assuring that defects are properly removed from the inspected item.

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [TBD]

Part Of
 	
    SWAT_SP_TEAM
	Supported Events
	
	    SWAT_ET_INSP_OVER
	        (optional)
	
	    SWAT_ET_INSP_I–MTG
	
	    SWAT_ET_INSP_REWORK
	
	    SWAT_ET_INSP_FOLLOW
	
	    SWAT_ET_INSP_CAUSAL
	
	    SWAT_ET_INSP_PROC–IMP	

Composed Of [None]		
		External Constraints · SWAT_CE_PPG

Revision History

    01a_070593 – Initial Version; 1st high level pass
    01b_070793 – Initial Version; Binding templates

Level # 2	Role Template ─────────────────── Template Type	Version # and Date 01b_070793

Name
    Inspector
	Unique Identifier
	    SWAT_SP_TEAM_INSP

Purpose
    Performs detailed analysis of inspection artifact and provides documented findings of defects.

Comments
    Note on the supporting documentation that there are two types of inspectors.

Additional States/Descriptions
 	
        [None]
	Associated Authority
	(via Internal Constraints)
	and Applicable Events
	
	    [TBD]

Part Of
 	
   SWAT_SP_TEAM
	Supported Events
	
	  SWAT_EA_INSP
	
	  SWAT_ET_INSP_OVER
	
	  SWAT_ET_INSP_PREP
	
	  SWAT_ET_INSP_I-MTG
	
	  SWAT_ET_INSP_FOLLOW
	
	  SWAT_ET_INSP_CAUSAL
	
	  SWAT_ET_INSP_PROC-IMP	

Composed Of SWAT_SP_ INSP_KEY SWAT_SP_ INSP_REG		**External Constraints** SWAT_CE_PPG

Revision History

   01a_070593 — Initial Version; 1st high level pass
   01b_070793 — Initial Version; Binding templates

Level # 3	Role Template ———————————— Template Type	Version # and Date 01b_070793

Name Key Inspector	Unique Identifier SWAT_SP_TEAM_INSP_KEY

Purpose

Performs expert detailed analysis of inspection artifact and provides documented findings of defects.

Comments

May also be asked to perform in the role of a reader. (Also, a key inspector that has performed all nonmoderator roles at formal inspections is eligible for training as a moderator.)

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [TBD]

Part Of SWAT_SP_TEAM_INSP	**Supported Events** SWAT_EA_INSP SWAT_ET_INSP_OVER (required)	

Composed Of
 	
[None]
	SWAT_ET_INSP_PREP
	    (required)
	
	SWAT_ET_INSP_I–MTG
	    (required)
	
	SWAT_ET_INSP_FOLLOW
	    (optional)
	
	SWAT_ET_INSP_CAUSAL
	    (required)
	
	SWAT_ET_INSP_PROC–IMP
	    (required)
		External Constraints
		
		SWAT_CE_PPG

Revision History

01a_070593 – Initial Version; 1st high level pass
01b_070793 – Initial Version; Binding templates

Level # 3	Role Template _____ Template Type	Version # and Date 01b_070793

Name Regular Inspector	**Unique Identifier** SWAT_SP_TEAM_INSP_REG

Purpose
Performs analysis of inspection artifacts and provides documented findings of defects.

Comments

Additional States/Descriptions [None]	**Associated Authority (via Internal Constraints) and Applicable Events** [TBD]

Part Of SWAT_SP_TEAM_INSP	**Supported Events** SWAT_EA_INSP SWAT_ET_INSP_OVER (optional) SWAT_ET_INSP_PREP (optional) SWAT_ET_INSP_I-MTG (optional) SWAT_ET_INSP_FOLLOW (optional) SWAT_ET_INSP_CAUSAL (optional) SWAT_ET_INSP_PROC-IMP (optional)	
Composed Of [None]		
		External Constraints SWAT_CE_PPG

Revision History

01a_070593 — Initial Version; 1st high level pass
01b_070793 — Initial Version; Binding templates

Level # 2	Role Template Template Type	Version # and Date 01b_070793

Name Scribe	Unique Identifier SWAT_SP_TEAM_SCRIBE

Purpose

Takes detailed notes, using an inspection log, of all defects discussed at the inspection meeting.

Comments

This role is NOT to be performed by clerical, administrative, secretarial, or support staff. This is to be a technical person performing in a technical capacity.

Additional States/Descriptions

[None]

Associated Authority (via Internal Constraints) and Applicable Events

[None]

Part Of

SWAT_SP_TEAM

Supported Events

SWAT_ET_INSP_I−MTG

SWAT_ET_CAUSAL

SWAT_ET_PROC−IMP

Composed Of

[None]

External Constraints

SWAT_CE_PPG

Revision History

01a_070593 − Initial Version; 1st high level pass

01b_070793 − Initial Version; Binding templates

Level # 1	Product Template Template Type	Version # and Date 01b_070793

Name Inspectable Artifact	Unique Identifier SWAT_TP_ART

Purpose

These products are characterized by all being susceptible to the SWAT inspection process.

Comments

Currently, the SWAT inspection process is focused upon Ada artifacts and technical documentation artifacts. However, standards, etc., exist to support a wider variety of items. Expansion of the SWAT program will occur as more people become trained in this process.

Additional States	Descriptions	Part Of
Needs Inspection	Product has not yet passed inspection.	[None]
Released	Product has passed inspection and can be returned into mainstream processing.	

Composed Of

SWAT_TP_ART_ADA

SWAT_TP_ART_GUIDE

Evolves From Events	State Transitions (Event/Step)
SWAT_EA_INSP	
SWAT_ET_INSP_PLAN	
SWAT_ET_INSP_OVER	[TBD]
SWAT_ET_INSP_PREP	
SWAT_ET_INSP_I–MTG	
SWAT_ET_INSP_FOLLOW	

External Constraints

SWAT_CE_PPG

Revision History

01a_070693 – Initial Version; 1st high level pass

01b_070793 – Initial Version; Binding templates

Level # 2	Product Template Template Type	Version # and Date 01b_070793

Name Inspectable Ada Artifact	Unique Identifier SWAT_TP_ART_ADA

Purpose

Software for delivered systems.

Comments

These products are all Ada code modules; package specifications, bodies, or both.

Additional States Compiles Clean	Descriptions Indicates that the producers have confirmed that there are not any compile-time errors in their product.	Part Of SWAT_TP_ART
		Composed Of SWAT_TP_ART_ ADA_SPEC SWAT_TP_ART_ ADA_BODY
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_SPC

Revision History

01a_070693 — Initial Version; 1st high level pass
01b_070793 — Initial Version; Binding templates

Level # 3	Product Template _____ Template Type	Version # and Date 01a_070693

Name
   Ada Package Specification
	Unique Identifier
	   SWAT_TP_ART_ADA_SPEC

Purpose
   Ada Package specifications primarily assist with verifying the systems design.

Comments

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_ADA
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_SPC

Revision History

   01a_070693 — Initial Version; 1st high level pass

Level # 3	**Product Template** Template Type	**Version # and Date** 01a_070693
Name Ada Package Body		**Unique Identifier** SWAT_TP_ART_ADA_BODY

Purpose
Ada Package Body (with the specification implied or explicit) for product software support.

Comments

Additional States [None]	**Descriptions** [None]	**Part Of** SWAT_TP_ART_ADA
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_SPC

Revision History
01a_070693 — Initial Version; 1st high level pass

Level # 2	Product Template _____ Template Type	Version # and Date 01b_070793

Name
    Technical Guides
	Unique Identifier
	    SWAT_TP_ART_GUIDE

Purpose

    Explain and support all aspects of installing, using, and maintaining the Ada software product.

Comments

    Expanding the types of guides inspected is an ongoing area of growth. When necessary, new types of documentation can be handled as miscellaneous technical guides until more formal designations and support can be developed.

Additional States	Descriptions	Part Of
Spell−Checking Done	Indicates that errors detected by a spelling checker have been removed.	SWAT_TP_ART

		Composed Of
		
		SWAT_TP_ART_GUIDE_
		    USER
		
		SWAT_TP_ART_GUIDE_
		    INSTALL
		
		SWAT_TP_ART_GUIDE_
		    SA
		
		SWAT_TP_ART_GUIDE_
		    MISC

Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X−TECHDOC

Revision History

    01a_070693 − Initial Version; 1st high level pass

    01b_070793 − Initial Version; Binding templates

Level # 3	Product Template _____ Template Type	Version # and Date 01a_070693
Name Technical User Guide		**Unique Identifier** SWAT_TP_ART_GUIDE_USER
Purpose Explains and supports all aspects of using the Ada software product.		
Comments		

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_GUIDE
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X–TECHDOC

Revision History

01a_070693 – Initial Version; 1st high level pass

Level # 3	**Product Template** Template Type	Version # and Date 01a_070693

Name System's Administrator Guide	**Unique Identifier** SWAT_TP_ART_GUIDE_SA

Purpose

Explains and supports all aspects of supporting the Ada software product.

Comments

Additional States [None]	**Descriptions** [None]	**Part Of** SWAT_TP_ART_GUIDE
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X–TECHDOC

Revision History

01a_070693 – Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01b_070793

Name Installation Guide	Unique Identifier SWAT_TP_ART_GUIDE_INSTALL

Purpose

Explains all aspects of installing (and removing) the Ada software product.

Comments

Additional States Approved_By_Human_Factors_Grp	Descriptions Indicates that human−factors related issues have already been addressed.	Part Of SWAT_TP_ART_GUIDE

Composed Of

[None]

Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]

External Constraints

SWAT_CE_PPG

SWAT_CE_PPG_
    X−TECHDOC

Revision History

01a_070693 − Initial Version; 1st high level pass
01b_070793 − Initial Version; Binding templates

Level # 3	Product Template Template Type	Version # and Date 01a_070693

Name Miscellaneous Technical Guide.	Unique Identifier SWAT_TP_ART_GUIDE_MISC

Purpose

Explains any essential information not already provided by other technical documentation.

Comments

This is essentially a catchall designation. If a technical guide does not fit under any of the other categories, it can be designated and inspected as a miscellaneous technical guide.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_ART_GUIDE
		Composed Of [None]
Evolves From Events [TBD]	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG SWAT_CE_PPG_ X–TECHDOC

Revision History

01a_070693 — Initial Version; 1st high level pass

Level # 1	Product Template Template Type	Version # and Date 01b_070793

Name Required Inspection Documentation	Unique Identifier SWAT_TP_DOC

Purpose

Provides a paper audit trail of the essential functions and events of the inspection process.

Comments

All of the documents represented as "children" of this class (regardless of how many generations removed) must be produced by some event in the inspection process. These are mandatory, and if the model does not capture where they are produced, then reexamine the event templates and correct.

Additional States [None]	Descriptions [None]	Part Of [None]
		Composed Of SWAT_TP_ DOC_LOG
Evolves From Events SWAT_EA_INSP	**State Transitions (Event/Step)** [TBD]	SWAT_TP_ DOC_RPT
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level # 2	Product Template Template Type	Version # and Date 01a_070693

Name
    Error Logs
	Unique Identifier
	    SWAT_TP_DOC_LOG

Purpose

    Log defects by location, type, and severity of inspected items.

Comments

    Currently, standardized error logs are used; however, customized or artifact-specific logs are being considered.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC

Evolves From Events
 	
[TBD]
	State Transitions (Event/Step)
	
	[TBD]
		Composed Of
		
		SWAT_TP_DOC_
		    LOG_TRIV
		
		SWAT_TP_DOC_
		    LOG_PREP
		
		External Constraints
		
		SWAT_CE_PPG

Revision History

    01a_070693 — Initial Version; 1st high level pass

Level # 3	Product Template _____ Template Type	Version # and Date 01b_070793

Name Trivial Error Log	Unique Identifier SWAT_TP_DOC_LOG_TRIV

Purpose

Logs trivial defects by location, type, and severity of inspected items.

Comments

These defects are not discussed during the inspection meeting. Instead, the logs are turned in at the inspection meeting by the inspectors. They are delivered to the producers for use in correcting the item under inspection. These logs also become a part of permanent records.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC_LOG
		Composed Of [None]
Evolves From Events SWAT_ET_INSP_PREP SWAT_ET_INSP_I−MTG SWAT_ET_INSP_FOLLOW	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 − Initial Version; 1st high level pass

01b_070793 − Initial Version; Binding templates

Level # 3	Product Template Template Type	Version # and Date 01b_070793

Name Preparation Error Log	Unique Identifier SWAT_TP_DOC_LOG_PREP

Purpose

Logs nontrivial defects by location, type, and severity found during inspection.

Comments

Preparation error logs contain all nontriv.al defects during the artifact's inspection. These logs are used (by the inspector who created them) during the inspection meeting. Defects are noted by the scribe (during the meeting) and the compiled results given to the producer(s).

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_DOC_LOG
		Composed Of [None]
Evolves From Events SWAT_ET_INSP_PREP SWAT_ET_INSP_I–MTG SWAT_ET_INSP_FOLLOW	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level # 2	Product Template Template Type	Version # and Date 01b_070793
Name Inspection Summary Reports		**Unique Identifier** SWAT_TP_DOC_RPT

Purpose

Provide a variety of useful, summarized information for management and for archival.

Comments

Typically, summary reports are tailored to the artifacts subject to inspection. Currently, however, only two such tailored report formats exist (shown below). Work is done on considerably more highly tailored summary reports.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_DOC

		Composed Of SWAT_TP_DOC_ RPT_S-WARE SWAT_TP_DOC_ RPT_T-DOC

Evolves From Events	**State Transitions (Event/Step)**	
SWAT_ET_INSP_I-MTG SWAT_ET_CAUSAL SWAT_ET_PROC-IMP	[TBD]	**External Constraints** SWAT_CE_PPG

Revision History

01a_070693 — Initial Version; 1st high level pass

01b_070793 — Initial Version; Binding templates

Level #	Product Template	Version # and Date
3	_Template Type_	01a_070693

Name	Unique Identifier
Inspection Software Summary Report	SWAT_TP_DOC_RPT_S–WARE

Purpose

Provides a variety of useful, summarized information for management and for archival.

Comments

This summary report is focused specifically on collecting data which supports the metrics developed for monitoring not only product quality but also for monitoring various characteristics of the SWAT formal inspection process itself.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_DOC_RPT

		Composed Of
		[None]

Evolves From Events	State Transitions (Event/Step)	
[TBD]	[TBD]	

		External Constraints
		SWAT_CE_PPG

Revision History

01a_070693 – Initial Version; 1st high level pass

Level # 3	Product Template Template Type	Version # and Date 01a_070693
Name Inspection Technical Document Summary Report		**Unique Identifier** SWAT_TP_DOC_RPT_T–DOC
Purpose Provides a variety of useful, summarized information for management, and for archival.		
Comments This summary report is focused specifically on collecting data which supports the metrics developed for monitoring not only product quality but also for monitoring various characteristics of the SWAT formal inspection process itself.		

Additional States [None]	**Descriptions** [None]	**Part Of** SWAT_TP_DOC_RPT
		Composed Of [None]
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History 01a_070693 – Initial Version; 1st high level pass

Level # 1	Product Template Template Type	Version # and Date 01a_070693

Name Inspection Memorandums		Unique Identifier SWAT_TP_MEMO

Purpose

Provide written announcements of event transitions within the SWAT inspection process.

Comments

Memorandums all have basic "boilerplate" versions used to create the memos. The memos provide details to formalize announcing what is about to happen and to formalize acknowledging what has happened. As such, these serve as excellent audit trails of transpiring events.

Additional States [None]	Descriptions [None]	Part Of [None]
		Composed Of SWAT_TP_MEMO I–INV SWAT_TP_MEMO I–EXIT SWAT_TP_MEMO I–DONE
Evolves From Events [TBD]	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 – Initial Version; 1st high level pass

Level # 2	Product Template Template Type	Version # and Date 01b_070793

Name Inspection Invitation Memorandum	Unique Identifier SWAT_TP_MEMO_I−INV

Purpose

Provides all participants with a written request to attend an inspection meeting.

Comments

These memorandums detail such information as the name of the inspected artifact, the date and time of the inspection, where the inspection will be conducted, and who to contact for further information.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_MEMO
		Composed Of [None]
Evolves From Events SWAT_EA_INSP SWAT_ET_INSP_I−MTG	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 − Initial Version; 1st high level pass

01b_070793 − Initial Version; Binding templates

Level # 2	Product Template Template Type	Version # and Date 01b_070793

Name Inspection Exit Memorandum	Unique Identifier SWAT_TP_MEMO_I–EXIT

Purpose

Provides all participants with a written acknowledgment of the end of an inspection meeting.

Comments

Included on this memorandum are any action items decided during the meeting and a reiteration of what is expected from whom. Also noted are any relevant details regarding (possible) reinspections.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_MEMO
		Composed Of [None]
Evolves From Events SWAT_EA_INSP SWAT_ET_INSP_I–MTG	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 – Initial Version; 1st high level pass
01b_070793 – Initial Version; Binding templates

Level # 2	Product Template Template Type	Version # and Date 01b_070793

Name Inspection Done Memorandum	Unique Identifier SWAT_TP_MEMO_I−DONE

Purpose

Provides written acknowledgement that a product was released from the inspection process.

Comments

This memorandum not only goes to participants, but it is also forwarded to all interested or relevant managers. Most important, this memorandum details the release status of the inspected artifact (i.e., it passed inspection and/or failed inspection).

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_MEMO
		Composed Of [None]
Evolves From Events SWAT_EA_INSP SWAT_ET_INSP_FOLLOW	State Transitions (Event/Step) [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01a_070693 − Initial Version; 1st high level pass

01b_070793 − Initial Version; Binding templates

Step 7. Construct Low Level Event Templates. According to the original plan, the analyst re-evaluates the process being defined and determines whether there are any advantages to introducing more detail to the process model. By doing so, some events that had previously been represented by production templates might be promoted to activities represented by management templates (with one or more supporting activities or tasks). The analyst proceeds by expanding upon the original indented event list and yielding the new list below:

Inspection SWAT Team Process
 Inspection Activity
  Planning
  Overview
  Preparation
  Inspection Meeting
   Review of Purpose
   Collection of Trivial Logs
   Product Inspection
   Review of Findings
   Reinspection Determination
  Rework
  Follow-up
 Causal Analysis Activity
 Inspection Process Improvement Activity
  Consensus Recommendations
  Presentation to Management

Default Set of Process States:
 Pre-Enabled
 Enabled
 In-Progress
 Disabled
 Suspended
 Cancelled
 Completed

As a result of expanding the indented list, the following two templates are altered:

- SW_ET_INSP-I_MTG which had been a production template, is altered to SW_EA_INSP_I-MTG, a management template.

- SW_ET_PROC-IMP which had been a production template, is altered to SW_EA_PROC-IMP, a management template.

Level # 3	Management Template _Template Type_	Version # and Date 01c_070993

Name Inspection_Meeting	Unique Identifier SWAT_EA_INSP_I–MTG

Purpose
This represents the actual inspection (by the group) of the item to be inspected.

Comments

Additional States

Entry Criteria
 	
SWAT_EA_INSP::In_Progress
and
SWAT_EA_INSP_PREP::
Complete
	Internal Processing
	
	Produce SWAT_TP_MEMO_I–INV
	SWAT_EA_INSP_I–MTG state is
	    In_Progress
	Take Attendance
	When (sufficient attendance)
	  Do SWAT_ET_INSP_I–MTG_PURP
	  Do SWAT_ET_INSP_I–MTG_TLOG
	  Do SWAT_ET_INSP_I–MTG_INSP
	  Do SWAT_ET_INSP_I–MTG_FIND
	  Do SWAT_ET_INSP_I–MTG_REIN
	END [When]
	Produce SWAT_TP_MEMO_I–EXIT
	SWAT_EA_INSP_I–MTG state is
	    Complete
		Exit Criteria
		
		SWAT_EA_INSP_I–MTG::
		Complete

Throughputs
 	
SWAT_TP_
  ART
SWAT_TP_
  DOC_LOG_
  TRIV
  PREP
SWAT_TP_
  DOC_RPT
SWAT_TP_
  MEMO_
  I–INV
  I–EXIT
	Supports
	
	SWAT_SP_
	TEAM_
	(all)
		Parent Event(s)
		    SWAT_EA_INSP
		
		Child Events
		    SWAT_ET_INSP_I–MTG_
		      PURP
		      TLOG
		      INSP
		      FIND
		      REIN
			**Internal
			Constraints**
				**External
				Constraints**
				
				SWAT_CE_
				PPG

Revision History

01a_070493 – Initial Version; 1st high level pass

01b_070793 – Initial Version; Binding templates

01c_070993 – 2nd Pass

Level # 2	Management Template Template Type	Version # and Date 01c_070993

Name Inspection_Process_Improvement	Unique Identifier SWAT_EA_PROC-IMP

Purpose

Improves the way in which inspections are actually conducted.

Comments

Inspection process improvement efforts are not constrained to any particular inspection subprocess. Inspections, causal analysis, and even this process are all potentially subject to process improvement efforts.

Additional States

Entry Criteria
 	
SWAT::In_Progress
	Internal Processing
	SWAT_EA_PROC-IMP state is
	    In_Progress
	Take attendance
	Review outstanding issues from prior
	    process improvement meetings
	Solicit suggestions from attendees
	Organize suggestions by domain
	Evaluate cost/benefit
	Prioritize by estimated cost/benefit
	Investigate alternatives
	Compile final recommendations
	At end of meeting; review findings
		Exit Criteria
		
		SWAT_EA_PROC-IMP::
		    Complete

Throughputs	Supports	Dismiss meeting
		Compile Inspection Process Improvement
		    Report
		SWAT_EA_PROC-IMP state is
		    Complete
			Internal
			Constraints
				External
				Constraints

	SWAT_SP_ S-MOD	**Parent Event(s)** SWAT		SWAT_CE_ PPG
SWAT_TP_ DOC_RPT	SWAT_SP_ TEAM_ (any)	**Child Events** SWAT_ET_PROC-IMP_RECOM SWAT_ET_PROC-IMP_PRESENT		

Revision History

01a_070493 — Initial Version; 1st high level pass
01b_070793 — Initial Version; Binding templates
01c_070993 — 2nd Pass

As shown below, the following new task templates are added:

      SW_ET_INSP_I-MTG_PURP
      SW_ET_INSP_I-MTG_TLOG
      SW_ET_INSP_I-MTG_INSP
      SW_ET_INSP_I-MTG_FIND
      SW_ET_INSP_I-MTG_REIN

and

      SW_ET_PROC-IMP_RECOM
      SW_ET_PROC-IMP_PRESENT

Level # 4	Production Template Template Type	Version # and Date 01c_070993

Name
    Inspection Meeting Purpose
	Unique Identifier
	    SWAT_ET_INSP_I—MTG_PURP

Purpose
    With all in attendance, review the specific goals of this inspection meeting.

Comments

Entry Criteria	Internal Processing	Exit Criteria

SWAT_EA_INSP_I—MTG::
    In_Progress
	SWAT_ET_INSP_I—MTG_PURP
	    state is In_Progress
	Commence when
	    Start_Time has Passed AND
	    Sufficient Key Inspectors Available
	else
	    Wait until sufficient key inspection
	    OR
	    Moderator decides to cancel
	If SWAT_SP_TEAM_MOD has NOT
	    decided to cancel
	    Briefly overview item
	    Note applicable SWAT_CE_PPG
	    State agenda
	SWAT_ET_INSP_I—MTG_PURP
	    state is Complete
		SWAT_ET_INSP_I—MTG_
		    PURP::Complete
		
		OR
		
		SWAT_ET_INSP_I—MTG_
		    PURP::Cancel

Products	Supports		Constraints
SWAT_TP_ ART	SWAT_SP_ TEAM_ MOD PROD READ INSP_ (all) SCRIBE		SWAT_CE_PPG

Parent Event(s)
SWAT_EA_INSP_I—MTG

Child Events

Research

Revision History

    01c_070993 — Extending Model (2nd pass)

Level # 4	Production Template Template Type	Version # and Date 01c_070993

Name Inspection Meeting Log Collection	Unique Identifier SWAT_ET_INSP_I—MTG_TLOG

Purpose

Collects all trivial error logs.

Comments

Trivial error logs document those items that do not need to be discussed before the entire inspection team.

Entry Criteria	Internal Processing	Exit Criteria

SWAT_EA_INSP_I—MTG::
          In_Progress
SWAT_ET_INSP_I—MTG_
     PURP::Complete
	SWAT_ET_INSP_I—MTG_TLOG
	          state is In_Progress
	For all (SWAT_SP_TEAM_INSP_KEY and
	          SWAT_SP_TEAM_INSP_REG)
	     Collect SWAT_TP_DOC_LOG_
	          TRIV
	     Confirm primary fields all have
	          values (insist on metrics,
	          i.e., hours spent, etc.)
	     Have Inspector make any
	          necessary corrections
		SWAT_ET_INSP_I—MTG_
		          TLOG:Complete

Products	Supports	SWAT_ET_INSP_I—MTG_TLOG state is Complete	Constraints

SWAT_TP_
   DOC_LOG_
   TRIV
	SWAT_SP_
	   TEAM_
	   MOD
	   PROD
	   READ
	   INSP_
	   (all)
	   SCRIBE		

		Parent Event(s) SWAT_EA_INSP_I—MTG	
Research		**Child Events**	

Revision History

01c_070993 — Extending Model (2nd pass)

Level # 4	Production Template Template Type	Version # and Date 01c_070993

Name Inspection Meeting – Actual	Unique Identifier SWAT_ET_INSP_I–MTG_INSP

Purpose
Performs a formal inspection as a group.

Comments

Entry Criteria

SWAT_EA_INSP_I–
MTG::
     In_Progress

SWAT_ET_INSP_I–
MTG_
     TLOG:Complete

Internal Processing

SWAT_ET_INSP_I–MTG_INSP
    state is In_Progress
While not done
  SWAT_SP_TEAM_READ presents the
    material by advancing through it
    section by section
  SWAT_SP_TEAM_INSP_(all) provides
    feedback and comments based on
    their notes and on the current
    examination
  SWAT_SP_TEAM_SCRIBE notes all
End while
SWAT_ET_INSP_I–MTG_INSP
    state is Complete

Exit Criteria

SWAT_ET_INSP_I–
MTG_
    INSP:Complete

Products

SWAT_TP_
  ART

SWAT_TP_
  DOC_LOG_
  TRIV
  PREP

SWAT_TP_
  DOC_RPT

Research

Supports

SWAT_SP_
  TEAM_
  MOD
  PROD
  READ
  INSP_
  (all)
  SCRIBE

Parent Event(s)
  SWAT_EA_INSP_I–MTG

Child Events

Constraints

SWAT_CE_PPG

Revision History

01c_070993 – Extending Model (2nd pass)

Level # 4	.	Production Template Template Type	Version # and Date 01c_070993

Name
    Findings Review .
		Unique Identifier
		    SWAT_ET_INSP_I–MTG_FIND

Purpose
    Review with all in attendance the composite findings.

Comments

Entry Criteria
 	
SWAT_EA_INSP_I–
MTG::
         In_Progress
 	
SWAT_ET_INSP_I–
MTG_
         INSP:Complete
	Internal Processing
	
	SWAT_ET_INSP_I–MTG_FIND
	         state is In_Progress
	Review all findings with those in
	         attendance
	While presenting the material, remind the
	         inspectors to follow their prep logs
	         to confirm nothing missed
	Update findings (a composite version of
	         SWAT_TP_DOC_LOG_PREP)
	SWAT_ET_INSP_I–MTG_FIND
	         state is Complete
		Exit Criteria
		
		SWAT_ET_INSP_I–
		MTG_
		         FIND:Complete

Products
 	
SWAT_TP_
    ART
 	
SWAT_TP_
    DOC_LOG_
    TRIV
    PREP
	Supports
	
	SWAT_SP_
	    TEAM_
	       MOD
	       PROD
	       READ
	       INSP_
	       (all)
	       SCRIBE
			Constraints
			
			SWAT_CE_PPG

		Parent Event(s) SWAT_EA_INSP_I–MTG	
Research		**Child Events**	

Revision History

    01c_070993 – Extending Model (2nd pass)

Level #	Production Template	Version # and Date
4	Template Type	01c_070993

Name	Unique Identifier
Inspection Meeting Purpose	SWAT_ET_INSP_I—MTG_REIN

Purpose

Review with all in attendance the specific goals of this inspection meeting.

Comments

Entry Criteria	Internal Processing	Exit Criteria
SWAT_EA_INSP_I—MTG:: In_Progress SWAT_ET_INSP_I—MTG_ FIND:Complete	SWAT_ET_INSP_I—MTG_REIN state is In_Progress Remind team of reinspection criteria Remind team of reinspection voting process (i.e, 1 yet vote = reinspection or majority, etc.) Take vote Note on SWAT_TP_DOC_RPT whether reinspection will be held If reinspection not expected Remind group that Moderator can optionally call for reinspection Dismiss meeting SWAT_ET_INSP_I—MTG_REIN state is Complete	SWAT_ET_INSP_I—MTG_ REIN:Complete

Products	Supports		
SWAT_TP_ ART SWAT_TP_ DOC_LOG_ TRIV PREP SWAT_TP_ DOC_RPT	SWAT_SP_ TEAM_ MOD PROD READ INSP_ (all) SCRIBE		Constraints SWAT_CE_PPG
Research		**Parent Event(s)** SWAT_EA_INSP_I—MTG	
		Child Events	

Revision History

01c_070993 — Extending Model (2nd pass)

Level # 3	Production Template _____ Template Type	Version # and Date 01c_070993

Name Inspection_Process_Improvement_Recommendations	Unique Identifier SWAT_ET_PROC–IMP_RECOM

Purpose

Identify primary inspection process problems and to identify alternative solutions.

Comments

Entry Criteria	Internal Processing	Exit Criteria
SWAT_ET_PROC–IMP: :In_Progress	SWAT_ET_PROC–IMP_RECOM state is In_Progress SWAT_SP_S–MOD presents general trend data HAVE EACH IN ATTENDANCE Present top 2 "Opportunities" Note on board/paper Prioritize entire list In priority order Discuss problem domain Discuss solution domain Note top 2 or 3 most promising solutions Compile composite report of findings and recommendations SWAT_ET_PROC–IMP_RECOM state is Complete	SWAT_ET_PROC–IMP_ RECOM::Complete

Products SWAT_TP_ DOC_RPT	Supports SWAT_SP_ S–MOD SWAT_SP_ TEAM_ (any)		Constraints
		Parent Event(s) SWAT_EA_PROC–IMP	
Research		**Child Events**	

Revision History

01c_070993 – 2nd Pass

Level # 3	Production Template Template Type	Version # and Date 01c_070993

Name Present Process Improvement Recommendations	Unique Identifier SWAT_ET_PROC–IMP_PRESENT

Purpose

Present group-consensus-based recommendations on process improvement to management.

Comments

The specific domain of the process improvement recommendations is that of the inspection process. Although many other areas impact the inspection process, typically such external areas are harder to change. This activity typically proposes changes local to the inspection process.

Entry Criteria	Internal Processing	Exit Criteria
SWAT_ET_PROC–IMP: :In_Progress SWAT_ET_PROC–IMP_ RECOM::Complete	SWAT_ET_PROC–IMP_PRESENT state is In_Progress SWAT_SP_S–MOD: Presents general trend data Presents overview of perceived problem areas Presents targeted problem areas Presents overview of proposed solutions (and a handout with details, if necessary) Responds to questions (with assistance from any SWAT_SP_TEAM_MOD in attendance) SWAT_ET_PROC–IMP_PRESENT state is Complete	SWAT_ET_PROC–IMP_ PRESENT::Complete

Products	Supports		Constraints
SWAT_TP_ DOC_RPT			
	SWAT_SP_ S–MOD	**Parent Event(s)** SWAT_EA_PROC–IMP	
	SWAT_SP_ TEAM_ MOD	**Child Events**	
Research			

Revision History

01c_070993 — 2nd Pass

Step 8. Extend/Merge Existing Templates (External Constraints, Roles, Products). Before the analyst creates templates from the remaining classes, he needs (originally discussed with Company-X) to extend the initial set of templates (on the assumption that having performed more detailed analysis, other template candidates would likely become apparent).

As shown below, only one new Role template and one new External Constraint template are added. However, it became apparent to the analyst that considerably more products, or more accurately by-products, were produced by the SWAT inspection process. The analyst found that a considerable number of checklists exist that detail Entry and Exit Criteria and that aid those involved in the formal inspection process by reminding them **what** types of things or actions are needed and **when** during the process. A total of 16 entry and exit condition checklists are used in the process. *Note:* The standards for checklists are defined as an external constraint, but the checklists themselves are modeled as products.

Additionally, the analyst was told that two new memorandums and a management request form were added to the inspections process. The analyst created the first version of each template as shown on the following pages.

Default Set of External Constraint States:
    Compliance_Unknown <EXPORT>
    Compliance_Under_Evaluation
    Compliance_Achieved
    Compliance_Failure
    Compliance_Waived

Default Set of Role States:
    Available_Exclusively
    Available_Shared
    Not_Available
    Disabled
    Suspended

Default Set of Product States:
    Unauthorized
    Authorized
    In_Progress
    In_Rework
    Disabled
    Suspended
    Cancelled
    Completed

Following is a list of the new templates:

- Role Template: Inspection Support Technician

- External Constraint Template: Company-X Checklist Reference Manual

- Product Template: Causal Analysis Invitation Memorandum

- Product Template: Inspection Process Improvement Invitation Memorandum

- Product Template: (Parent and 1 child) Presentation to Management – Request Form

- Product Template: Entry/Exit Criteria Forms (parent)

  - Product Template: Planning Entry Criteria Checklist

  - Product Template: Planning Exit Criteria Checklist

  - Product Template: Overview Entry Criteria Checklist

  - Product Template: Overview Exit Criteria Checklist

  - Product Template: Preparation Entry Criteria Checklist

  - Product Template: Preparation Exit Criteria Checklist

  - Product Template: Inspection Meeting Entry Criteria Checklist

  - Product Template: Inspection Meeting Exit Criteria Checklist

  - Product Template: Rework Entry Criteria Checklist

  - Product Template: Rework Exit Criteria Checklist

  - Product Template: Follow-up Entry Criteria Checklist

  - Product Template: Follow-up Exit Criteria Checklist

  - Product Template: Causal Analysis Entry Criteria Checklist

  - Product Template: Causal Analysis Exit Criteria Checklist

  - Product Template: Inspection Process Improvement Entry Criteria Checklist

  - Product Template: Inspection Process Improvement Exit Criteria Checklist

Level # 2	External Constraint Template Template Type	Version # and Date 01c_070993

Name Company-X Process Support Checklists	Unique Identifier SWAT_CE_PPG_X–CHKLST

Purpose

Provide employees with process-specific checklists.

Comments

Typically, these checklists are of two types: entry criteria checklists and exit criteria checklist. In all cases, the objective is to see that certain minimum process constraints have been followed. Additionally, the checklists are used as an audit trail of events.

Special Form Of (parent) SWAT_CE_PPG	Additional States/Description [None]	
General Form Of **(list children)** [None]	**Constrained Events** SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_ET_INSP_I–MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC–IMP	**Constrained Throughputs** [None] **Constrained Supports** [None]

Revision History

01c_070993 – 2nd Pass

Level # 2	Role Template _____ Template Type	Version # and Date 01c_070993

Name Inspection Support Technician	Unique Identifier SWAT_SP_S-TECH

Purpose

 Provides administrative and limited technical support to the Super Moderator.

Comments

 The support technician should be thoroughly familiar with the inspection process; and may, on occasion, attend (in a passive or background capacity) any inspection meeting.

Additional States/Descriptions [None]	Associated Authority (via Internal Constraints) and Applicable Events [None]

Part Of [None]	Supported Events SWAT SWAT_ET_INSP_PLAN SWAT_ET_CAUSAL SWAT_ET_PROC-IMP	
Composed Of [None]		
		External Constraints SWAT_CE_PPG

Revision History

 01c_070993 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_070993

Name Causal Analysis Invitation Memorandum	Unique Identifier SWAT_TP_MEMO_CAUSAL

Purpose

Provides written invitation to a causal analysis meeting.

Comments

Typically will be sent to all qualified persons.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_MEMO

		Composed Of [None]
Evolves From Events SWAT	**State Transitions (Event/Step)** [TBD]	
		External Constraints SWAT_CE_PPG

Revision History

01c_070993 – 2nd Pass

Level # 2	Product Template _____ Template Type	Version # and Date 01c_070993

Name	Unique Identifier
Process Improvement Invitation Memorandum	SWAT_TP_MEMO_PROC−IMP

Purpose

Provides written invitation to a SWAT inspection process improvement meeting.

Comments

Typically will be sent to all qualified persons.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_MEMO

Composed Of

[None]

Evolves From Events	State Transitions (Event/Step)
SWAT	[TBD]

External Constraints

SWAT_CE_PPG

Revision History

01c_070993 − 2nd Pass

Level # 1	Product Template	Version # and Date
	Template Type	01c_071293

Name	Unique Identifier
Management Inspection Forms	SWAT_TP_MGMT

Purpose

Provide a collection of forms to facilitate communication with management.

Comments

Additional States	Descriptions	Part Of
[None]	[None]	[None]

		Composed Of

Evolves From Events	State Transitions (Event/Step)	SWAT_TP_MGMT_ REQUEST
SWAT	[None]	

		External Constraints
		SWAT_CE_PPG

Revision History

01c_071293 — 2nd Pass

Level #	Product Template	Version # and Date
2	Template Type	01c_071293

Name	Unique Identifier
Management Presentation Request	SWAT_TP_MGMT_PRES−REQ

Purpose

Informs management that a set of process improvement findings are ready to be presented.

Comments

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_MGMT

		Composed Of
		[None]

Evolves From Events	State Transitions (Event/Step)	
SWAT SWAT_ET_PROC−IMP_ RECOM	[None]	

		External Constraints
		SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

| Level #

1 | Product Template

Template Type | Version # and Date

01c_071293 |
|---|---|---|

| Name

Inspection Entry/Exit Forms | Unique Identifier

SWAT_TP_EE |
|---|---|

Purpose

Provide a collection of checklists which define entry and exit conditions.

Comments

To date, there are no checklists that are either support or throughput specific. In all cases, the current set of checklists address events. If this changes in the future, consider a second tier distinction (i.e., ...EE_E_, ...EE_S, ...EE_T for events, support, and throughput checklists).

| Additional States

[None] | Descriptions

[None] | Part Of

[None] |
|---|---|---|
| | | **Composed Of**
SWAT_TP_EE_

PLAN–IN, PLAN–OUT
OVER–IN, OVER–OUT,
PREP–IN, PREP–OUT,
I–MTG–IN,
I–MTG–OUT,
REWORK–IN,
REWORK–OUT
FOLLOW–IN
FOLLOW–OUT,
CAUSAL–IN,
CAUSAL–OUT,
PROC–IMP–IN
PROC–IMP–OUT |
| **Evolves From Events**

SWAT_EA_INSP | **State Transitions (Event/Step)**

[None] | **External Constraints**

SWAT_CE_PPG |

Revision History

01c_071293 – 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Planning Entry Form	Unique Identifier SWAT_TP_EE_PLAN−IN

Purpose

Provides a checklist that determines whether planning entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE

		Composed Of [None]

Evolves From Events SWAT_EA_INSP	State Transitions (Event/Step) [None]	

		External Constraints SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Planning Exit Form	Unique Identifier SWAT_TP_EE_PLAN−OUT

Purpose

Provides a checklist that determines whether planning exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_ET_INSP_PLAN	**State Transitions (Event/Step)** [None]	
		External Constraints SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name
    Inspection Overview Entry Form
	Unique Identifier
	    SWAT_TP_EE_OVER–IN

Purpose

    Provides a checklist that determines whether overview entry criteria have been met.

Comments

    All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_EA_INSP	**State Transitions (Event/Step)** [None]	
		External Constraints SWAT_CE_X–CHKLST

Revision History

    01c_071293 – 2nd Pass

Level #	Product Template	Version # and Date
2	Template Type	01c_071293

Name	Unique Identifier
Inspection Overview Exit Form	SWAT_TP_EE_OVER−OUT

Purpose

Provides a checklist that determines whether overview exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_EE
		Composed Of
Evolves From Events	**State Transitions (Event/Step)**	[None]
SWAT_ET_INSP_OVER	[None]	
		External Constraints
		SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Preparation Entry Form	Unique Identifier SWAT_TP_EE_PREP-IN

Purpose

Provides a checklist that determines whether preparation entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
Evolves From Events SWAT_EA_INSP	**State Transitions (Event/Step)** [None]	**Composed Of** [None]
		External Constraints SWAT_CE_X-CHKLST

Revision History

01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Preparation Exit Form	Unique Identifier SWAT_TP_EE_PREP−OUT

Purpose

Provides a checklist that determines whether preparation exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_EE
		Composed Of
Evolves From Events	**State Transitions (Event/Step)**	[None]
SWAT_ET_INSP_PREP	[None]	
		External Constraints SWAT_CE_X−CHKLST

Revision History

01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Meeting Entry Form	Unique Identifier SWAT_TP_EE_I−MTG−IN

Purpose

Provides a checklist that determines whether inspection meeting entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_EA_INSP	State Transitions (Event/Step) [None]	
		External Constraints SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Meeting Exit Form		Unique Identifier SWAT_TP_EE_I-MTG-OUT

Purpose

Provides a checklist that determines whether inspection meeting exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_ET_INSP_I-MTG	State Transitions (Event/Step) [None]	
		External Constraints SWAT_CE_X-CHKLST

Revision History

01c_071293 -- 2nd Pass

Level # 2	Product Template _____ Template Type	Version # and Date 01c_071293

Name Inspection Rework Entry Form	Unique Identifier SWAT_TP_EE_I-REWORK-IN

Purpose

Provides a checklist that determines whether rework entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_EA_INSP	**State Transitions (Event/Step)** [None]	
		External Constraints SWAT_CE_X-CHKLST

Revision History

01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name
   Inspection Rework Exit Form
	Unique Identifier
	   SWAT_TP_EE_I-REWORK-OUT

Purpose

   Provides a checklist that determines whether rework exit criteria have been met.

Comments

   All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_ET_INSP_REWORK	**State Transitions (Event/Step)** [None]	
		External Constraints SWAT_CE_X-CHKLST

Revision History

   01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Follow-up Entry Form	Unique Identifier SWAT_TP_EE_I-FOLLOW-IN

Purpose

Provides a checklist that determines whether follow-up entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of
Evolves From Events SWAT_EA_INSP	**State Transitions (Event/Step)** [None]	[None]
		External Constraints SWAT_CE_X-CHKLST

Revision History

01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Follow-up Exit Form	Unique Identifier SWAT_TP_EE_I-FOLLOW-OUT

Purpose

Provides a checklist that determines whether follow-up exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE

		Composed Of [None]

Evolves From Events SWAT_ET_INSP_FOLLOW	State Transitions (Event/Step) [None]	

		External Constraints SWAT_CE_X-CHKLST

Revision History

01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293
Name Inspection Follow-up Entry Form		**Unique Identifier** SWAT_TP_EE_I-FOLLOW-IN

Purpose

Provides a checklist that determines whether follow-up entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
Evolves From Events SWAT_EA_INSP	**State Transitions (Event/Step)** [None]	**Composed Of** [None] **External Constraints** SWAT_CE_X-CHKLST

Revision History

01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name *Inspection Follow-up Exit Form*	Unique Identifier SWAT_TP_EE_I–FOLLOW–OUT

Purpose

Provides a checklist that determines whether follow-up exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_ET_INSP_FOLLOW	State Transitions (Event/Step) [None]	
		External Constraints SWAT_CE_X–CHKLST

Revision History

01c_071293 – 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Causal Analysis Meeting Entry Form	Unique Identifier SWAT_TP_EE_I−CAUSAL−IN

Purpose

Provides a checklist that determines whether causal analysis entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT	State Transitions (Event/Step) [None]	
		External Constraints SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

Level # 2	Product Template ——————————————— Template Type	Version # and Date 01c_071293

Name Inspection Causal Analysis Exit Form ·	Unique Identifier SWAT_TP_EE_I–CAUSAL–OUT

Purpose

Provides a checklist that determines whether follow-up exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of [None]
Evolves From Events SWAT_ET_CAUSAL	State Transitions (Event/Step) [None]	
		External Constraints SWAT_CE_X–CHKLST

Revision History

01c_071293 — 2nd Pass

Level # 2	Product Template Template Type	Version # and Date 01c_071293

Name Inspection Process Improvement Meeting Entry Form	Unique Identifier SWAT_TP_EE_I−PROC−IMP−IN

Purpose

Provides a checklist for determining whether process improvement entry criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States [None]	Descriptions [None]	Part Of SWAT_TP_EE
		Composed Of
Evolves From Events SWAT	State Transitions (Event/Step) [None]	[None]
		External Constraints SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

Level #	Product Template	Version # and Date
2	Template Type	01c_071293

Name	Unique Identifier
Inspection Process Improvement Exit Form	SWAT_TP_EE_I−PROC−IMP−OUT

Purpose

· Provides a checklist for determining whether process improvement exit criteria have been met.

Comments

All checklists, after completion and usage, are to be forwarded through the moderator to the super moderator to facilitate process management and metric data collection, respectively.

Additional States	Descriptions	Part Of
[None]	[None]	SWAT_TP_EE
		Composed Of
Evolves From Events	**State Transitions (Event/Step)**	
SWAT_EA_PROC−IMP	[None]	[None]
		External Constraints
		SWAT_CE_X−CHKLST

Revision History

01c_071293 − 2nd Pass

Step 9. Construct Resource Templates. The analyst found that relatively few resources are needed to support Company-X's current approach to the SWAT inspection process. One meeting room has been designated for the SWAT inspection process. Aside from that, an inspection metrics database is the only other dedicated resource.

Default Set of Resource States:
    Available_Exclusively
    Available_Shared
    Not_Available
    Disabled
    Suspended

Resource Template: Inspection Meeting Room

Resource Template: Inspection Metrics Database

Level # 1	Resource Template _____ Template Type	Version # and Date 01a_071393

Name
  Inspection Meeting Room
	Unique Identifier
	  SWAT_SR_MTG−ROOM

Purpose

  Provides a room dedicated for SWAT inspection work.

Comments

  SWAT inspection scheduling takes first priority in this room − any other use can be over-ridden
  if needed by the SWAT team. Typically, nonSWAT inspections also use the room on a regular
  (but generally nonintrusive) basis.

Additional States/Descriptions

Schedule_Conflict:	Indicates a state of overlapping reservations.
Reserved_SWAT:	Held for use by the SWAT inspection team.
Reserved_Inspection:	Held for inspection use, but not SWAT.
Reserved_Other:	Held for noninspection use.

Operational Guidelines/Limits

  Maximum Seating: 22

Part Of

  [None]

Supported Events

  SWAT

  SWAT_EA_INSP

  SWAT_EA_INSP_I−MTG

  SWAT_ET_CAUSAL

  SWAT_EA_PROC−IMP

Required Supplies/Materials

  (2) Overhead Projectors

  (1) Flipchart (Paper)

  [Note: File cabinet is to
  remain in room for dedi-
  cated inspection use.]

Composed Of

  [None]

External Constraints

  [None]

Revision History

  01a_071393 − 2nd Pass; new details

Level # 1	**Resource Template** Template Type	Version # and Date 01a_071393

Name Inspection Metrics Database	Unique Identifier SWAT_SR_DB–METRIC

Purpose

Provides a common repository for all inspection related metric data.

Comments

The database collects information from the entire inspection. However, SWAT–specific data is designated as such.

Additional States/Descriptions [None]		Operational Guidelines/Limits [Currently, only the super moderator, moderators, the inspection technical support person, and managers have access to this repository.]
Part Of [None]	**Supported Events** SWAT_ET_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_EA_PROC–IMP	**Required Supplies/Materials** [N/A]
Composed Of [None]		**External Constraints** ["Company-X Guidelines on Security and Access Privileges" (not yet modelled as an external constraint).]

Revision History

01a_071393 — 2nd Pass; new details

Step 10. Construct Research Templates. Virtually every aspect of Company-X's SWAT inspection process is governed by the manipulation of paper artifacts (such as memorandums, checklists, etc.) Consequently, the analyst has essentially been able to model all significant work represented by a product.

However, one area not currently covered by this approach occurs during the rework stage. Specifically, the producers can solicit the advice of inspectors for alternative ways to address different defects. The analyst considers this too important to overlook and decides to capture this part of the process by using a Research template.

Default Set of Research States:
Unauthorized
Authorized
In_Progress
In_Rework
Disabled
Suspended
Cancelled
Completed

Research Template: Inspector Recommendations

Level #	Research Template	Version # and Date
1	_Template Type_	01a_071393

Name	Unique Identifier
Inspector Recommendations	SWAT_SR_INS−REC

Purpose

Provide producers with insights and suggestions.

Comments

During the inspection meeting, discussion of fixes or corrections is specifically prohibited. However, an inspector can offer that they have one or more suggestions. In such cases, the producer(s) optionally can solicit such advice during the rework stage.

Additional States	Descriptions	Part Of
[None]		[None]

		Composed Of
		[None]

Evolves From Events	State Transitions (Event/Step)	
SWAT_EA_INSP SWAT_EA_INSP_I−MTG SWAT_ET_INSP_REWORK	[None]	

		External Constraints
		[None]

Revision History

01a_071393 − 2nd Pass; new details

Step 11. Construct Internal Constraint Templates.

Default Set of Internal Constraint States:
   (NONE)

Following is an indented list of new internal constraint templates:

- Internal Constraint Template: Standard Constraint Templates

    - Internal Constraint Template: Commencement permission

    - Internal Constraint Template: Suspension permission

    - Internal Constraint Template: Recommencement permission

    - Internal Constraint Template: Cancellation permission

    - Internal Constraint Template: Resurrection permission

    - Internal Constraint Template: Completion permission

    - Internal Constraint Template: Override permission

    - Internal Constraint Template: Executive permission

- Internal Constraint Template: Inspection Constraint Templates

    - Internal Constraint Template: Reinspection permission

    - Internal Constraint Template: Causal Meeting Initiation permission

    - Internal Constraint Template: Process Improvement Meeting Initiation permission

Level # 1	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name	Unique Identifier
Standard Internal Constraint Set	SWAT_CI_STD

Purpose

Provides a set of standard constraints consistent with the standard event state set.

Comments

This set of constraints is intended to allow those with sufficient authority the permission to "declare" state changes (typically) in events and (rarely) in throughputs and supports. The SWAT process currently limits this authority to event−based state change declarations only.

Special Form Of (parent)	Constrained Throughputs	Constrained Supports
[None]	[None]	[None]

General Form Of (list children)	Constrained Events	Associated Roles
SWAT_CI_STD_ COMM SUSP RECO CANC RESU COMP OVER EXEC	SWAT [All permissions are potentially available on all events occurring within the SWAT process.]	SWAT_SP_S−MOD

Revision History

01a_071493 − 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name
    Standard Commence Permission
	Unique Identifier
	    SWAT_CI_STD_COMM

Purpose
    Provides authority to designated roles allowing them to declare "commencement" state transitions.

Comments
    As reflected below, this state change declaration authority is applicable to events only.

Special Form Of (parent) SWAT_CI_STD	Constrained Throughputs [None]	Constrained Supports [None]
General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I-MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC-IMP_RECOM	Associated Roles SWAT_SP_S-MOD SWAT_SP_TEAM_MOD

Revision History

    01a_071493 — 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Standard Suspension Permission	Unique Identifier SWAT_CI_STD_SUSP

Purpose

    Provides authority to designated roles allowing them to declare "suspend" state transitions.

Comments

    As reflected below, this state change declaration authority is applicable to events only.

Special Form Of (parent) SWAT_CI_STD	Constrained Throughputs [None]	Constrained Supports [None]
General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I−MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC−IMP_RECOM	Associated Roles SWAT_SP_S−MOD SWAT_SP_TEAM_MOD

Revision History

    01a_071493 — 2nd Pass; new details

Level #	Internal Constraint Template	Version # and Date
2	Template Type	01a_071493

Name	Unique Identifier
Standard Recommence Permission	SWAT_CI_STD_RECO

Purpose

Provides authority to designated roles allowing them to declare "recommencement" state transitions.

Comments

As reflected below, this state change declaration authority is applicable to events only.

Special Form Of (parent)	Constrained Throughputs	Constrained Supports
SWAT_CI_STD	[None]	[None]

General Form Of (list children)	Constrained Events	Associated Roles
[None]	SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I-MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC-IMP_RECOM	SWAT_SP_S-MOD SWAT_SP_TEAM_MOD

Revision History

01a_071493 — 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Standard Cancelation Permission		Unique Identifier SWAT_CI_STD_CANC

Purpose

Provides authority to designated roles allowing them to declare "cancel" state transitions.

Comments

As reflected below, this state change declaration authority is applicable to events only.

Special Form Of (parent) SWAT_CI_STD	Constrained Throughputs [None]	Constrained Supports [None]
General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I—MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC—IMP_RECOM	Associated Roles SWAT_SP_S—MOD SWAT_SP_TEAM_MOD

Revision History

01a_071493 — 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Standard Resurrection Permission		Unique Identifier SWAT_CI_STD_RESU

Purpose

    Provides authority to designated roles allowing them to declare "resurrect" state transitions.

Comments

    As reflected below, this state change declaration authority is applicable to events only.

Special Form Of (parent) SWAT_CI_STD	Constrained Throughputs [None]	Constrained Supports [None]
General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I−MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC−IMP_RECOM	Associated Roles SWAT_SP_S−MOD SWAT_SP_TEAM_MOD

Revision History

    01a_071493 — 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Standard Completion Permission	Unique Identifier SWAT_CI_STD_COMP

Purpose

Provides authority to designated roles allowing them to declare "complete" state transitions.

Comments

As reflected below, this state change declaration authority is applicable to events only.

Special Form Of (parent) SWAT_CI_STD	Constrained Throughputs [None]	Constrained Supports [None]

General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I—MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC—IMP_RECOM	Associated Roles SWAT_SP_S—MOD SWAT_SP_TEAM_MOD

Revision History

01a_071493 — 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Standard Override Permission		Unique Identifier SWAT_CI_STD_OVER

Purpose

    Provides authority to temporarily override other constraints so as to allow events to proceed.

Comments

    This authority allows for overriding entry or exit conditions that are unsatisfied, and allows events to either start or "finish" is spite of (unsatisfied) constraints that would normally prevent them from doing so.

Special Form Of (parent) SWAT_CI_STD	Constrained Throughputs [None]	Constrained Supports [None]
General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I—MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC—IMP_RECOM	Associated Roles SWAT_SP_S—MOD

Revision History

    01a_071493 — 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Standard Executive Permission	Unique Identifier SWAT_CI_STD_EXEC

Purpose

Provides authority to permanently override constraints.

Comments

Whereas override authority allows for essentially ignoring certain constraints, executive authority allows those constraints to be permanently declared "satisfied."

Special Form Of (parent)	Constrained Throughputs	Constrained Supports
SWAT_CI_STD	[None]	[None]

General Form Of (list children)	Constrained Events	Associated Roles
[None]	SWAT_EA_INSP SWAT_ET_INSP_PLAN SWAT_ET_INSP_OVER SWAT_ET_INSP_PREP SWAT_EA_INSP_I—MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL SWAT_ET_PROC—IMP_RECOM	SWAT_SP_S—MOD

Revision History

01a_071493 — 2nd Pass; new details

| Level #

1 | Internal Constraint Template
_____
Template Type | Version # and Date

01a_071493 |
|---|---|---|

| Name
    Inspection Extended Constraint Set | | Unique Identifier
    SWAT_CI_INS |
|---|---|---|

Purpose
    Provides a set of extended constraints for mapping inspection-specific authorities.

Comments
    This set of constraints provides a variety of authorities which (when coupled with roles) represent
    various discretionary decisions that can be made.

| Special Form Of
(parent)

[None] | Constrained Throughputs

[None] | Constrained Supports

[None] |
|---|---|---|
| General Form Of
(list children)

SWAT_CI_INS_

    REIN

    CAUS

    PRIM | Constrained Events

    SWAT

    SWAT_ET_CAUSAL

    SWAT_EA_PROC−IMP | Associated Roles

    SWAT_SP_S−MOD |

Revision History

    01a_071493 − 2nd Pass; new details

Level # 2	**Internal Constraint Template** Template Type	Version # and Date 01a_071493

Name
    Reinspection Declaration Authority
	Unique Identifier
	    SWAT_CI_INS_REIN	

Purpose
    Provides authority to declare that a reinspection is necessary.

Comments

Special Form Of (parent) SWAT_CI_INS	Constrained Throughputs [None]	Constrained Supports [None]

General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_EA_INSP_I-MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW	Associated Roles SWAT_SP_S-MOD SWAT_SP_TEAM_MOD SWAT_SP_TEAM_INSP_KEY SWAT_SP_TEAM_INSP_REG SWAT_SP_TEAM_INSP_READ SWAT_SP_TEAM_INSP_PROD

Revision History

    01a_071493 — 2nd Pass; new details

Level # 1	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Inspection Extended Constraint Set	Unique Identifier SWAT_CI_INS

Purpose

Provides a set of extended constraints for mapping inspection-specific authorities.

Comments

This set of constraints provides a variety of authorities which (when coupled with roles) represent various discretionary decisions that can be made.

Special Form Of (parent) [None]	Constrained Throughputs [None]	Constrained Supports [None]

General Form Of (list children) SWAT_CI_INS_ REIN CAUS PRIM	Constrained Events SWAT SWAT_ET_CAUSAL SWAT_EA_PROC-IMP	Associated Roles SWAT_SP_S-MOD

Revision History

01a_071493 — 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Causal Meeting Declaration Authority	Unique Identifier SWAT_CI_INS_CAUS

Purpose

Provides authority to declare that a causal analysis meeting is necessary.

Comments

Special Form Of (parent) SWAT_CI_INS	Constrained Throughputs [None]	Constrained Supports [None]

General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_EA_INSP_I-MTG SWAT_ET_INSP_REWORK SWAT_ET_INSP_FOLLOW SWAT_ET_CAUSAL	Associated Roles SWAT_SP_S-MOD SWAT_SP_TEAM_MOD SWAT_SP_TEAM_INSP_KEY

Revision History

01a_071493 — 2nd Pass; new details

Level # 1	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name Inspection Extended Constraint Set	Unique Identifier SWAT_CI_INS

Purpose

Provides a set of extended constraints for mapping inspection-specific authorities.

Comments

This set of constraints provides a variety of authorities which (when coupled with roles) represent various discretionary decisions that can be made.

Special Form Of (parent) [None]	Constrained Throughputs [None]	Constrained Supports [None]

General Form Of (list children) SWAT_CI_INS_ REIN CAUS PRIM	Constrained Events SWAT SWAT_ET_CAUSAL SWAT_EA_PROC-IMP	Associated Roles SWAT_SP_S-MOD

Revision History

01a_071493 -- 2nd Pass; new details

Level # 2	Internal Constraint Template Template Type	Version # and Date 01a_071493

Name	Unique Identifier
Inspection Process Improvement Meeting Declaration Authority	SWAT_CI_INS_PRIM

Purpose

Provides authority to declare that a SWAT inspection process improvement meeting is necessary.

Comments

Note: The current policy states that you need three key inspectors to agree that a process improvement meeting should be scheduled. Such meetings can also occur at the request of either the super moderator or any of the moderators.

Special Form Of (parent) SWAT_CI_INS	Constrained Throughputs [None]	Constrained Supports [None]

General Form Of (list children) [None]	Constrained Events SWAT_EA_INSP SWAT_EA_INSP_I-MTG SWAT_EA_PROC-IMP	Associated Roles SWAT_SP_S-MOD SWAT_SP_TEAM_MOD SWAT_SP_TEAM_INSP_KEY

Revision History

01a_071493 — 2nd Pass; new details

Step 12. Bind All Throughput/Support/Constraint Templates to Event Templates. Binding the templates would proceed as it did in Step 6 when external constraints, roles, and products were bound to the events. To avoid reproducing over 100 example templates, explicit examples of this second-pass binding are not shown. However, the resulting templates would be very similar to the current set: the exception would be references to the extended product, role, and external constraints and to the new resources, research, and internal constraint templates.·

Step 13. Evaluate Template Set and Improve Clarity/Simplicity. At this step, the process analyst reviews the templates and edits them to improve clarity and simplicity. Clarity is improved by assuring deliberate and consistent use of naming conventions, indentation conventions, and logical flow constructs and, as discussed in Section 3, by reducing redundancy.

You can often improve simplicity by removing excess information. In many cases, this results in removing excessive information from within fields. In some cases, however, a reviewing of a collection of templates may yield several templates that can be removed altogether (with a corresponding removal of references to those templates from other templates). For example, the process analyst might elect to remove the two "resource" templates since they do not make a substantive contribution to the general process model.

Step 14. Derive Petri Net Representation of Process Dynamics. As the final step in this example, the process analyst uses the templates to facilitate the construction of a Petri net model which represents the SWAT inspection process. The following Petri net involves a few extensions to facilitate decomposition of one otherwise rather complex diagram into a collection of diagrams that are easier to depict and discuss. (If you are unfamiliar with the principles of Petri nets, review the Petri net material near the end of Section 5.)

- **●** *Token:* Used to mark a place.

- **○** *Place:* A location where tokens may reside. In this example, places often represent events transpiring. Places are connected by arcs to transitions.

- **▯** *Transition:* Flow control and coordination mechanism. Connected by arcs to places.

- **⦵** *Subnet:* Represents an underlying set of places and transitions connected by arcs.

- **▮** *Rule-Driven Transition:* A transition that has no input places, but instead fires as a result of some event occurring, permission being granted, etc.

- **▤** *Subnet Entry Connector:* Fires whenever a token arrives at a subnet place (see above).

- **▨** *Subnet Exit Connector:* Becomes enabled whenever the subnet place is empty of all tokens (and the exit connector is otherwise enabled).

- **▧** *Switched Transition:* When enabled, only one of the output places receives a token. (These transitions always have two or more output arcs.)

- **→** *Arc:* Connects places to transitions, and vice versa.

Since Petri nets are especially useful for capturing relationships between events, this example is derived almost entirely from the event templates constructed above. Using those templates as reference, the following net can be interpreted as follows. First, in Figure A-1 note that place p1−2 has a token, signifying that the necessary material to support the SWAT inspection program is available. Since p1−1 does not have a token, t1−3 is not enabled. However, when t1−1 fires (signifying that the vice president (VP) has given permission for the SWAT inspection process to commence), a token appears at p1−1. Since all the input places to t1−3 have tokens, t1−3 is enabled and fires, placing a single token in each of its output places. In this case, p1−3 is the only output place. While a token exists at p1−3, the subnet it represents (essentially, the entire SWAT program) is active. Note that t1−4 remains disabled until t1−2 fires (signifying that the VP has requested the SWAT inspection process cease). When t1−4 fires, the token disappears from p1−3 and a token appears at p1−4 (signifying the SWAT process has halted). The t1−5 can then fire, and a token appears at p1−2, indicating that once again, SWAT inspection support material is ready, but that VP permission is needed before the process can (re)execute.

The next diagram (Figure A-2) is a simple subnet that is itself composed to two subnets and models the task template labeled SWAT (version number 01a_070493). When a token arrives at place p1−3 (above), the convention in this example is that it causes t2−1 to fire, thereby causing tokens to appear at p2−1, p2−2, and p2−3. Note that even while tokens remain in these places, t2−2, because it is a subnet exit, is not enabled unless (as described in the key above) p1−3 is empty of tokens. In other words, so long as VP permission has not been rescinded, the process improvement, causal analysis, and inspection activities can all continue in parallel. However, when VP permission is removed, t2−2 fires, and tokens are removed from p2−1, p2−2, and p2−3.
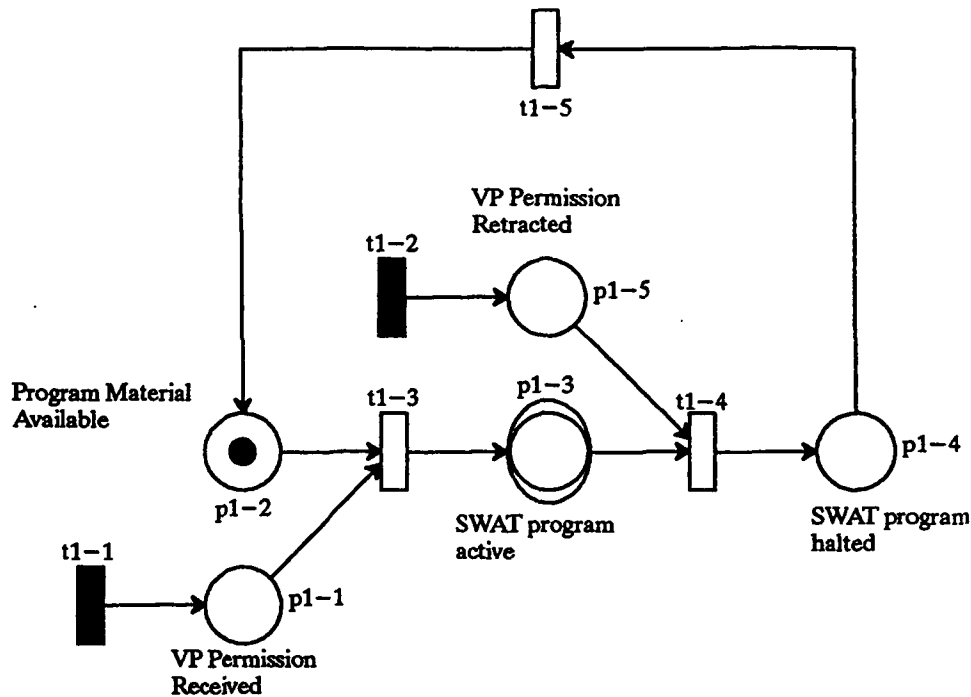
Figure A-1. SWAT Inspection Process
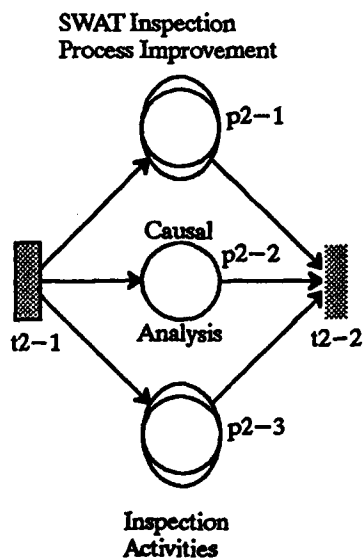
Figure A-2. SWAT Program Active

The subnet p2−1 is expanded in Figure A-3 and depicts a rendition of the events detailed on templates SWAT_EA_PROC_IMP (version 01c_070993), SWAT_ET_PROC_IMP_RECOM (version 01c_070993), and SWAT_ET_PROC_IMP_PRESENT (version 01c_070993). Again, when a token arrives at p2−1, the subnet causes transition t3a−1 to fire and a token to appear at p3a−1. The t3a−6 fires and a token appears at p3a−6 (and disappears from p3a−1). As long as t3a−8 does not become enabled by a token departing from subnet place p2−1 (above), then only the firing of t3a−7 (indicating that a meeting is being held) can cause the token to be removed from p3a−6. However, t3a−7 cannot fire until t3a−5 fires and places a token in p3a−5 (indicating that a moderator has scheduled the meeting). The firing of t3a−7 causes a token to appear at p3a−4 indicating the meeting is being

held. The t3a−4 fires and the resulting token at p3a−3 indicates that there is typically an interval between when the process improvement meeting has been held and when the results can be reported to management. A token at p3a−2 (after the firing of t3a−3) represents the meeting in which findings are presented to management. t3a−2, and then t3a−6 fire, resulting in a token appearing at p3a−6. In this "marking," the net is once again ready for the moderator to schedule a meeting, and this activity can continue as long as a token remains in place p2−1.
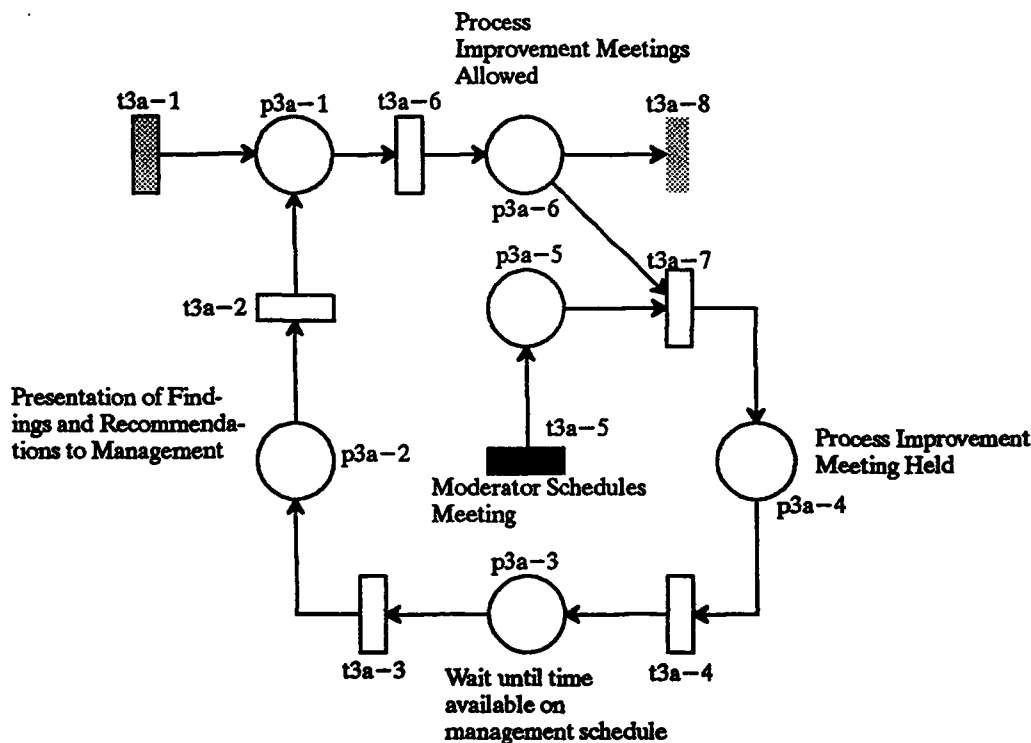
Figure A-3. SWAT Inspection Process Improvement

The subnet p2−3 (Figure A-4) represents the inspection meeting activity depicted by the following templates:

- SWAT_EA_INSP (version 01b_070793)

- SWAT_ET_INSP_ PLAN (version 01b_070793)

- SWAT_ET_INSP_ OVER (version 01b_070793)

- SWAT_ET_INSP_ PREP(version 01b_070793)

- SWAT_EA_INSP_ I−MTG (version 01c_070993)

- SWAT_ET_INSP_ I−MTG_REIN (version 01c_070993)

- SWAT_ET_INSP_ REWORK (version 01b_070793)

- SWAT_ET_INSP_ FOLLOW (version 01b_070793)

The expanded representation of that net is shown in Figure A-4. When a token arrives at p2−3, t3b−2 fires, leaving just one of three input places to t3b−4 without a token (p3b−2). The firing of t3b−1 (representing the arrival of something to be inspected) causes a token to appear in p3b−2, thus enabling t3b−4. After this transition fires, a token appears at p3b−4 to represent the planning task, which is then followed by the overview task.

Note that when t3b−6 fires, tokens appear at five separate places. These places represent five different people performing the preparation task prior to attending the inspection meeting. When these inspectors are done, t3b−7 fires and a token arrives at p3b−11 which represents the occurrence of the inspection meeting. Next t3b−8 fires, and a token appears at p3b−12. This represents the original producer(s) correcting defects detected and reported by the inspection process. When rework is complete, t3b−9 fires and places a token at p3b−13 indicating follow-up work by the moderator. At this point, the moderator makes a final decision whether or not a reinspection meeting is needed. As represented by the switch transition, t3b−10, a token will either appear back at p3b−11 or, if a reinspection is not needed, at p3b−1. As with the other subnets, this subnet continues to remain ready and executing until the subnet exit connector (t3a−8) fires and removes the token from p3b−1 and (in this example) permanently prevents further execution of this subnet.

Throughout this example, discussion of time has been deferred in the interests of reduced complexity. However, it is comparatively easy to construct timed Petri nets. Time can be associated with either transitions or with places but never (for mathematical reasons) with both. Time associated with transitions causes transitions to wait, after being enabled, for the defined amount of time before they fire. When associating time with places (as would likely be done in an expanded version of this example) tokens carry an attribute that determines whether they are available or unavailable. Unavailable tokens cannot enable transitions. When a token appears at a place, it is always tagged as unavailable. It remains unavailable until the determined amount of time for that place has transpired, at that point the token becomes available. In this example, p3b−6 might have an associated time of four hours (indicating it is expected to take that inspector four hours to complete their preparation). The p3b−7 might have a time of 8 hours; p3b−8, 6 hours; p3b−9, 15 hours; and p3b−10, 12 hours. If this were the case, 15 hours after t3b−6 fired, t3b−7 would fire. The firing of t3b−7 would always occur after the maximum of the times associated with its input places, as that would be the amount of time required for the last token (in this case, at p3b−9) to change from unavailable to available, thereby enabling t3b−7.

Although a thorough discussion of Petri nets is well beyond the scope of this Guidebook, it is important to note that one of the key advantages to Petri net models is their ability to model the dynamic or behavioral characteristics of the processes they represent. The ability to gain dynamic insights is typically completely absent from static process representations (i.e., any representation that cannot actually be executed). If dynamic process insights are important, it is highly recommended that the field of Petri nets be given further study.
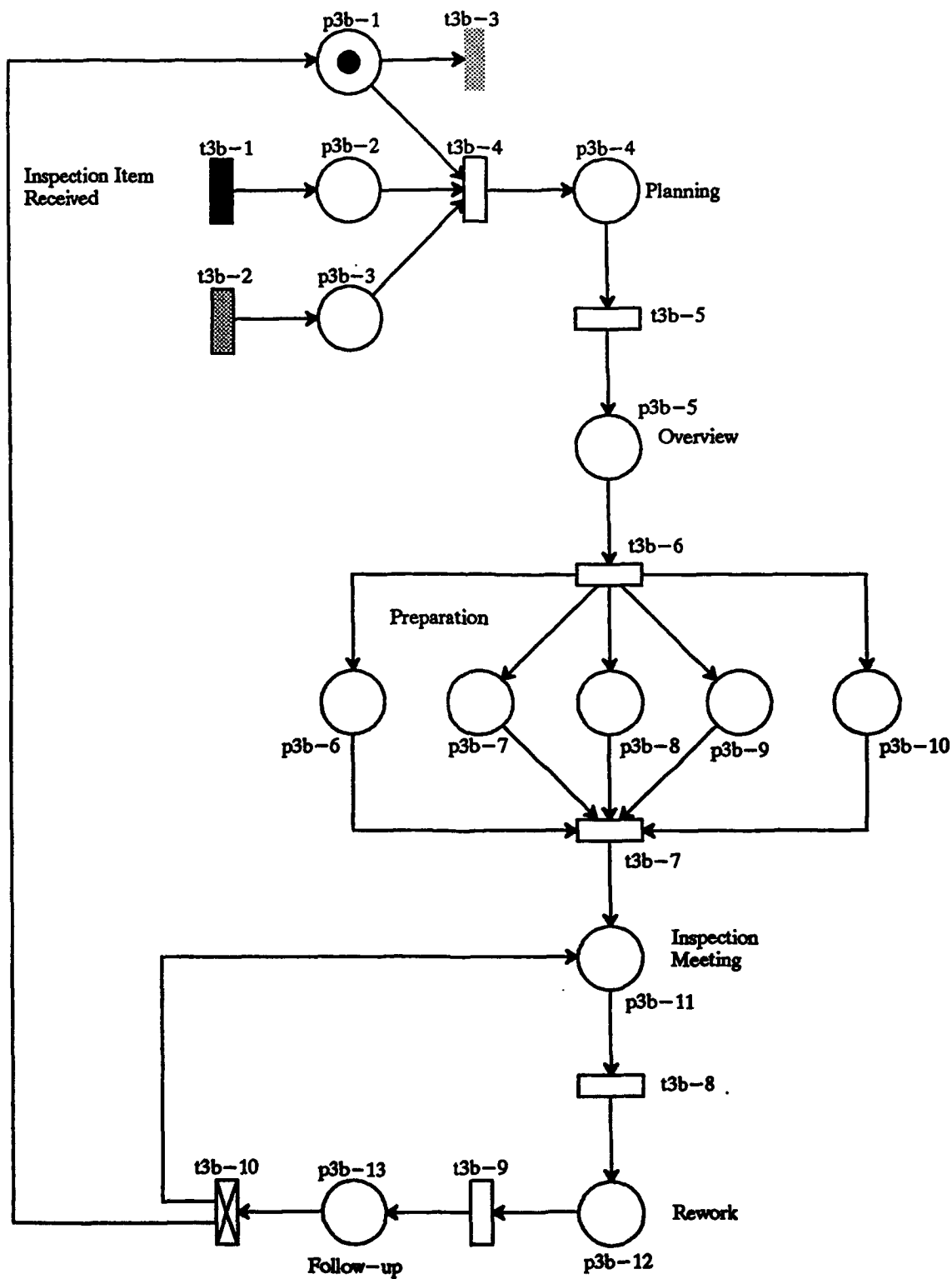
Figure A-4. Inspection Activities

APPENDIX B. ETVX EXAMPLE

The ETVX example is based on the following sample. The ETVX paradigm is a procedural formalism for representing activities and relationships within PPA. An ETVX box represents the concept that at any level of abstraction a work activity must have entry (E) and exit (X) criteria, some task (T) to be done, and some means or collection of means for performing validation (V).
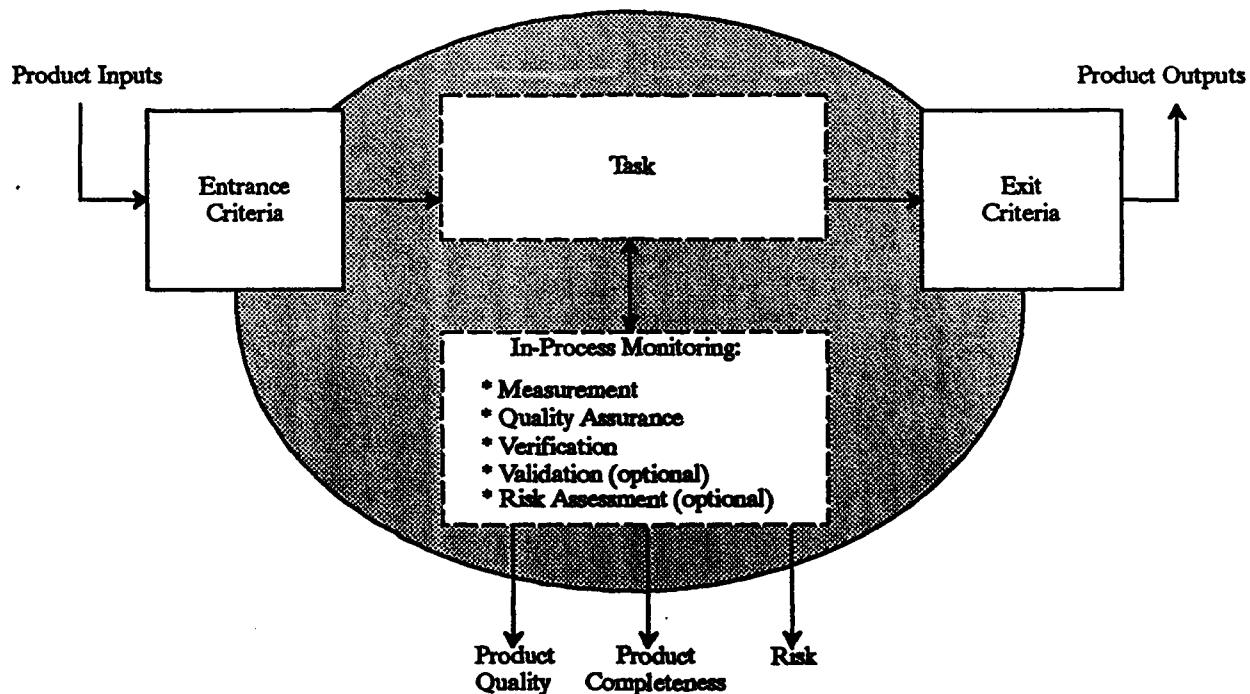
Figure B-1. ETVX Diagram

You archive hierarchical decomposition in ETVX by "exploding" the task (T) component of an activity and showing the subactivities (each in ETVX form) of which it is comprised. (Similarly, any subactivity can also be further decomposed, as necessary.) Additionally, the ETVX model does not imply that all activities or tasks in succeeding stages wait for completion of the preceding stages. Later stages may function (that is, activities occurring) concurrently with preceding stages.

You should note that the ETVX approach does not yield an interconnected diagrammatic representation of the system being modeled. Instead, ETVX depicts each of the four subcomponents that constitutes an activity. In the comparative example section, multiple ETVX activities are shown within a loose network of interconnecting lines. These directed lines should improve your interpretation of this particular example, but such a diagram is not typical of the traditional ETVX model.

When using the templates, E is the Entry Criteria and X is the Exit Criteria on the Event template. Use the parent/child relations on the template to define ETVX decomposition. Validation can best

be captured by defining a set of events that are explicitly intended for performing validation, and then heuristically asserting that all events in a given model must, within their internal processing, invoke one or more events from the validation event tree. Internal Processing, also on the Event template, maps to the tasks (T) in ETVX, and the Supporting Events from the Process template explicitly describes the tasks in ETVX. Even though the templates contain all of the information for ETVX, it is important to remember that ETVX is not an operational representation; ETVX's usefulness comes from its conceptual presentation for the process definition.

ETVX is good for defining the high level organizational process since it is less formal than other models which results in greater flexibility for change as your understanding of the process grows. ETVX can also be used for defining a process in an environment where management is not willing to support the process control of a well defined process. ETVX is a good process representation when parts of the process (especially low level events defined in more detail) are intended to be automated. Because ETVX has a hierarchical presentation mechanism essential for representing large and complex processes, it well represents such processes, but lacks the formality of other models.

B.1 APPROACHES

The general approach for generating an ETVX diagram from the process definition template set is to run down the event meta-class template and collect the templates. Try to form the event structure tree from the event related indented list. Then use the tree as a map to fill in the ETVX diagrams one by one. To fill each ETVX, use the cross reference fields to get the information for validation represented as event and constraints that use a checklist to validate the inspection. Overall, there are 17 ETVX diagrams shown in the next section. The sequence follows the number on the left of the event's name in Figure B-2.
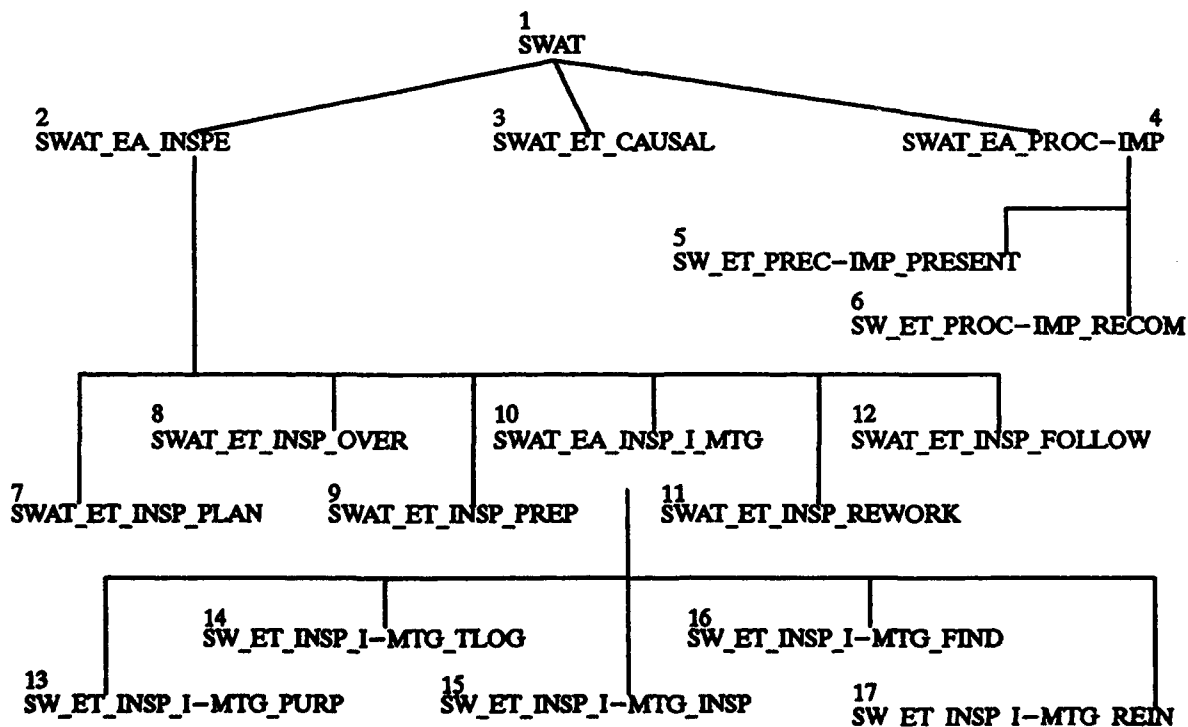
Figure B-2. Parent-Child Event Tree Structure

B.2 ETVX PRESENTATION OF INSPECTION PROCESS

Figure B-3. ETVX Diagram 1: SWAT

Figure B-4. ETVX Diagram 2: SWAT_EA_INSP

SWAT_TP_DOC_RPT

SWAT_TP_EECAUSAL--OUT

SWAT_TP_EECAUSAL-IN

SWAT::
In_Progress

Take attendance
Review outstanding issues from prior
    causal analysis meetings
Report results of trend analysis efforts
Solicit perceived problem areas from
    attendees
Prioritize problem areas
Discuss/determine process changes to
    reduce/remove root causes
At end of meeting; review findings
Dismiss meeting
Compile Causal Analysis Report

SWAT_ET_CAUSAL::
Complete

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_TP_EE

Figure B-5. ETVX Diagram 3: SWAT_ET_CAUSAL

SWAT_TP_
DOC_RPT

SWAT::
In_Progress

Take attendance
Review outstanding issues from prior
    process improvement meetings
Solicit suggestions from attendees
Organize suggestions by domain
Evaluate cost/benefit
Prioritize by estimated cost/benefit
Investigate alternatives
Compile final recommendations
At end of meeting; review findings
Dismiss meeting
Compile Insp. Proc. Improvement Report

SWAT_EA_PROC-IMP::
Complete

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_TP_EE_PROC-IMP_IN
* SWAT_TP_EE_PROC-IMP_OUT
* SWAT_CE_VP

Figure B-6. ETVX Diagram 4: SWAT_EA_PROC-IMP

SWAT_TP_
DOC_RPT

SWAT_ET_PROC-IMP:
In_Progress

SWAT_ET_PROC-IMP_
RECOM::Complete

Presents general trend data
Presents overview of perceived
  problem areas
Presents targeted problem areas
Presents overview of proposed
  solutions (and a handout
  with details, if necessary)
Responds to questions (with
  assistance from any
  SWAT_SP_TEAM_MOD
  in attendance)

SWAT_ET_PROC-IMP_
PRESENT::Complete

In-Process Monitoring:
* SWAT_CE_PPG
* SWAT_CI_STD
* SWAT_TP_INS

Figure B-7. ETVX Diagram 5: SW_ET_PREC-IMP_PRESENT

SWAT_TP_
DOC_RPT

SWAT_ET_PROC-IMP:
:In_Progress

SWAT_SP_S-MOD presents general
  trend data
HAVE EACH IN ATTENDANCE
  Present top 2 "Opportunities"
  Note on board/paper
Prioritize entire list
In priority order
  Discuss problem domain
  Discuss solution domain
  Note top 2 or 3 most promising
    solutions
Compile composite report of findings
  and recommendations

SWAT_ET_PROC-IMP_
RECOM::Complete

In-Process Monitoring:
* SWAT_CE_PPG
* SWAT_CI_STD
* SWAT_TP_INS

Figure B-8. ETVX Diagram 6: SW_ET_PROC-IMP_RECOM

SWAT_TP_
ART

MEMOI−INV

SWAT_EA_INSP::
In_Progress

Evaluate the Item to be Inspected
Estimate the expected inspection rate
Construct a schedule
Assign personnel to the schedule
Confirm personnel availability
Adjust/Redo schedule as necessary

SWAT_ET_INSP_PLAN::
Complete

In-Process Monitoring:

* SWAT_CI_INS

* SWAT_CI_STD

* SWAT_TP_EE_PLAN_IN

* SWAT_TP_EE_PLAN_OUT

Figure B-9. ETVX Diagram 7: SWAT_ET_INSP_PLAN

SWAT_TP_
ART

SWAT_TP_
ART

SWAT_EA_INSP_PLAN::
Complete

SWAT_EA_INSP::
In_Progress

SWAT_ET_INSP_OVER state is
In_Progress
Introduce Objective
Introduce Members
Have Reader present the item to
be inspected
Review preparation time and schedule
Dismiss meeting
SWAT_ET_INSP_OVER state is
Complete

SWAT_ET_INSP_OVER::
Complete

In-Process Monitoring:

* SWAT_CI_INS

* SWAT_CI_STD

* SWAT_TP_EE_OVER_IN

* SWAT_TP_EE_OVER_OUT

Figure B-10. ETVX Diagram 8: SWAT_ET_INSP_OVER

SWAT_TP_
ART

Disperse material to be inspected and
associated support forms
Each Inspector Does:
Review of Relevant Standards
Personal Inspection of Item
Completes Trivial Error Log
Completes Preparation Log
Completes Preparation_Exit
Form (Checklist)
END [Each Inspector Does]
Moderator Collects/Verifies Checklists
Moderator distributes Inspection
Invitation letter

SWAT_EA_INSP::
In_Progress
SWAT_EA_INSP_OVER::
Complete

SWAT_TP_
DOC_LOG_
TRIV
PREP

SWAT_ET_INSP_PREP::
Complete

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_CE_PPG
* SWAT_TP_EE_PREP_IN
* SWAT_TP_EE_PREP_OUT

Figure B-11. ETVX Diagram 9: SWAT_ET_INSP_PREP

SWAT_TP_
ART

SWAT_TP_
DOC_LOG_
TRIV
PREP

SWAT_EA_INSP::
In_Progress
SWAT_EA_INSP_PREP::
Complete

Take Attendance
When (sufficient attendance)
Do SWAT_ET_INSP_I-MTG_PURP
Do SWAT_ET_INSP_I-MTG_TLOG
Do SWAT_ET_INSP_I-MTG_INSP
Do SWAT_ET_INSP_I-MTG_FIND
Do SWAT_ET_INSP_I-MTG_REIN
END [When]
Produce SWAT_TP_MEMO_I-EXIT

SWAT_TP_
DOC_RPT

SWAT_TP_
MEMO_
I-INV
I-EXIT

SWAT_EA_INSP_I-MTG::
Complete

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_CE_PPG
* SWAT_TP_EE_I-MTG_IN
* SWAT_TP_EE_I-MTG_OUT

Figure B-12. ETVX Diagram 10: SWAT_EA_INSP_I_MTG

SWAT_TP_
ART

SWAT_TP_
DOC_LOG_
TRIV
PREP

SWAT_EA_INSP::
In_Progress

SWAT_EA_INSP_I–MTG::
Complete

Each SWAT_SP_TEAM_PROD:
   Evaluates pertinent defects
   Alters product only to correct
     the defect
   Notes location and scope of
     changes
   If Req'd: makes before and after
     listings until all
     defects addressed
SWAT_SP_TEAM_MOD notified that
   all changes are complete

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_CE_PPG
* SWAT_TP_EE_REWORK_IN
* SWAT_TP_EE_REWORK_OUT

SWAT_ET_INSP_REWORK::
Complete

SWAT_TP_
MEMO_
I–DONE

Figure B-13. ETVX Diagram 11: SWAT_ET_INSP_REWORK

SWAT_TP_
ART

SWAT_TP_
DOC_LOG_
TRIV
PREP

SWAT_EA_INSP::
In_Progress

SWAT_EA_INSP_I–MTG::
Complete

Meet with the author(s)
Discuss nature and scope of changes
Evaluate each change
Confirm only authorized changes made
Determine whether a reinspection needs
   to be conducted
Moderator completes the followup
   checklist
Produce SWAT_TP_MEMO_I–DONE

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_CE_PPG
* SWAT_TP_EE_FOLLOW_IN
* SWAT_TP_EE_FOLLOW_OUT

SWAT_ET_INSP_FOLLOW::
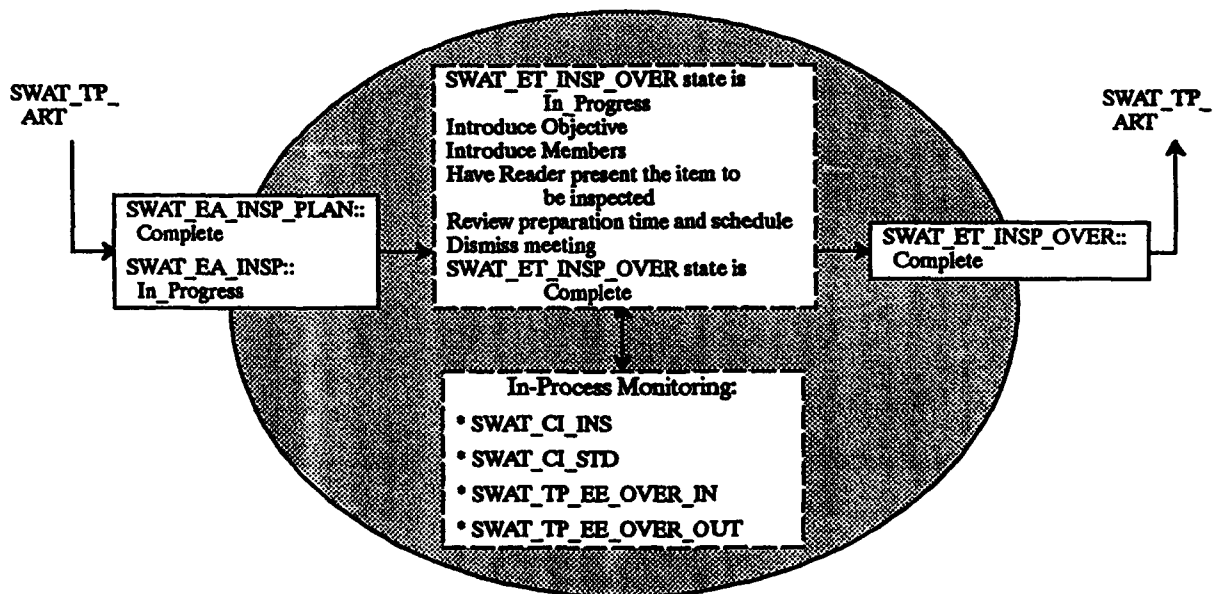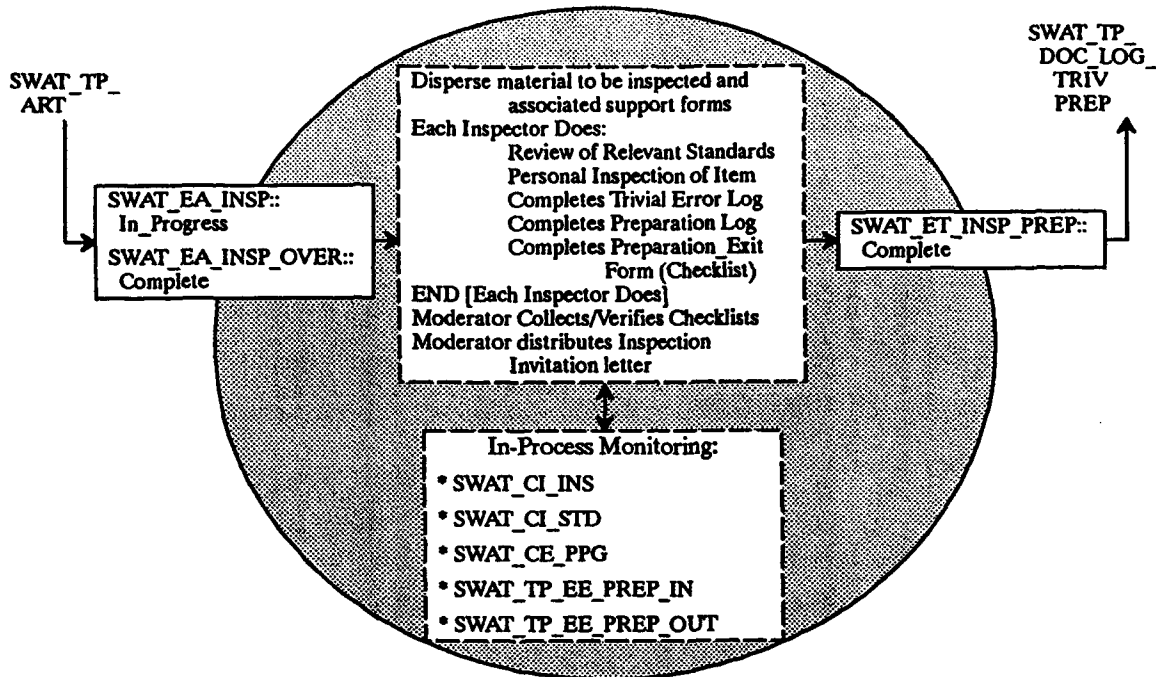Complete

SWAT_TP_
MEMO_
I–DONE
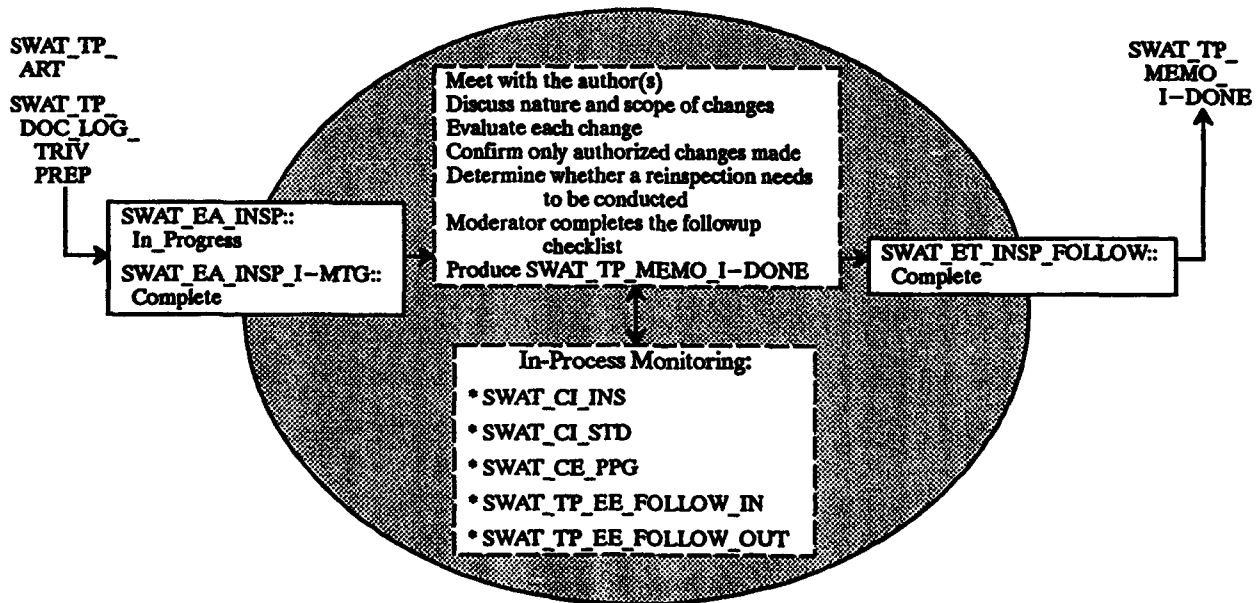
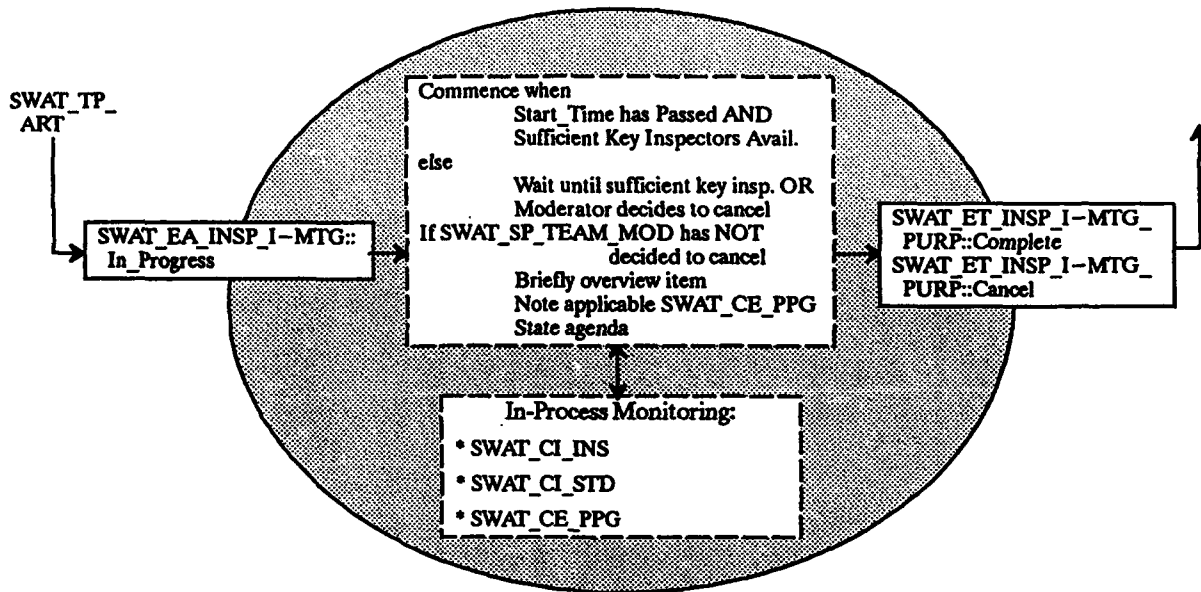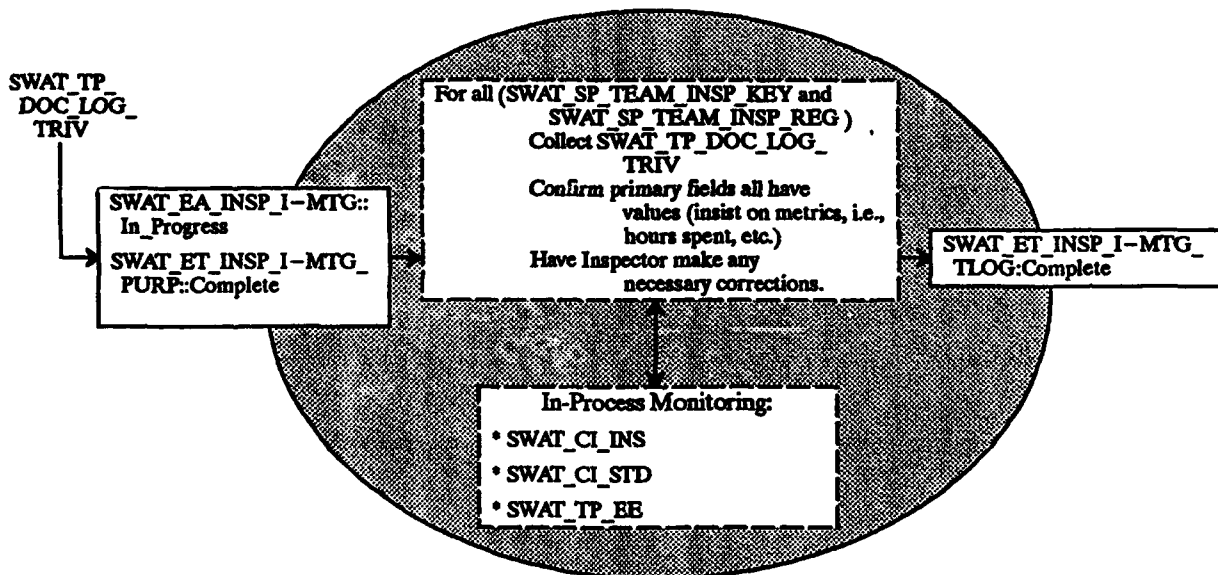Figure B-14. ETVX Diagram 12: SWAT_ET_INSP_FOLLOW

Figure B-15. ETVX Diagram 13: SW_ET_INSP_I—MTG_PURP

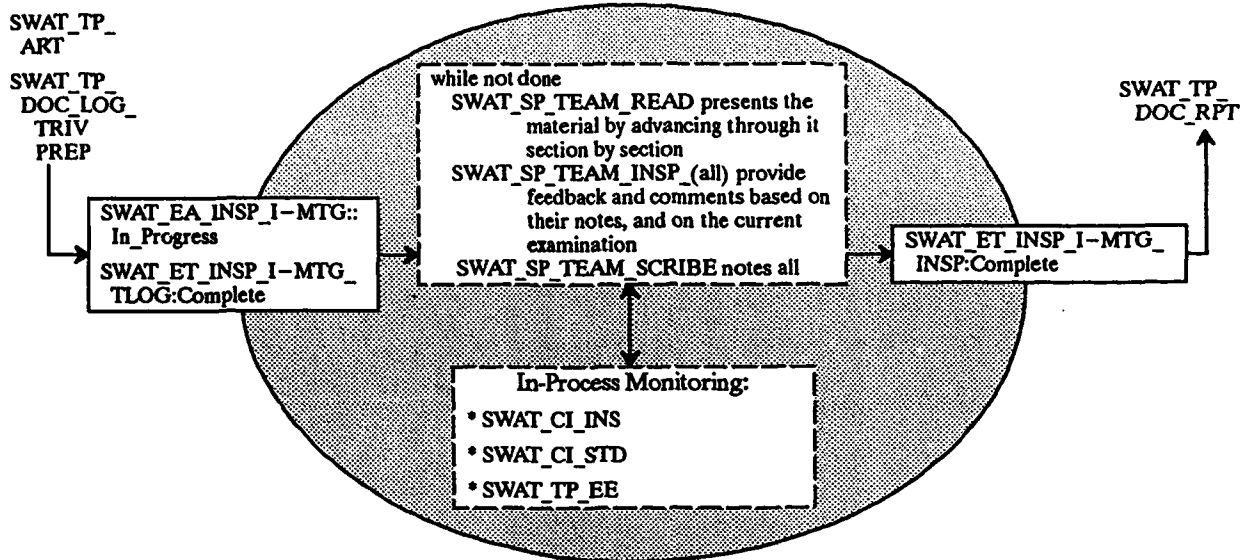Figure B-16. ETVX Diagram 14: SW_ET_INSP_I—MTG_TLOG

SWAT_TP_
ART

SWAT_TP_
DOC_LOG_
TRIV
PREP

SWAT_EA_INSP_I-MTG::
In_Progress

SWAT_ET_INSP_I-MTG_
TLOG:Complete

while not done
SWAT_SP_TEAM_READ presents the
material by advancing through it
section by section
SWAT_SP_TEAM_INSP_(all) provide
feedback and comments based on
their notes, and on the current
examination
SWAT_SP_TEAM_SCRIBE notes all

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_TP_EE

SWAT_TP_
DOC_RPT

SWAT_ET_INSP_I-MTG_
INSP:Complete

Figure B-17. ETVX Diagram 15: SW_ET_INSP_I-MTG_INSP

SWAT_TP_
ART

SWAT_EA_INSP_I-MTG::
In_Progress

SWAT_ET_INSP_I-MTG_
INSP:Complete

Review all findings with those in
attendance
While presenting the material, remind the
inspectors to follow their prep logs
to confirm nothing missed
Update findings (a composite version of
SWAT_TP_DOC_LOG_PREP)

In-Process Monitoring:
* SWAT_CI_STD
* SWAT_CE_PPG

SWAT_TP_
DOC_LOG_
TRIV
PREP

SWAT_ET_INSP_I-MTG_
FIND:Complete

Figure B-18. ETVX Diagram 16: SW_ET_INSP_I-MTG_FIND

SWAT_TP_
  ART

SWAT_TP_
  DOC_LOG_
    TRIV
    PREP

SWAT_EA_INSP_I—MTG::
  In_Progress

SWAT_ET_INSP_I—MTG_
  FIND:Complete

Remind team of reinspection criteria
Remind team of reinspection voting
                process (i.e, 1 yet vote = reinsp.
                or majority, etc.)
Take vote
Note on SWAT_TP_DOC_RPT whether
                reinspection will be held
If reinspection not expected
                Remind group that Moderator can
                optionally call for reinspection
Dismiss meeting

In-Process Monitoring:
* SWAT_CI_INS
* SWAT_CI_STD
* SWAT_TP_EE

SWAT_TP_
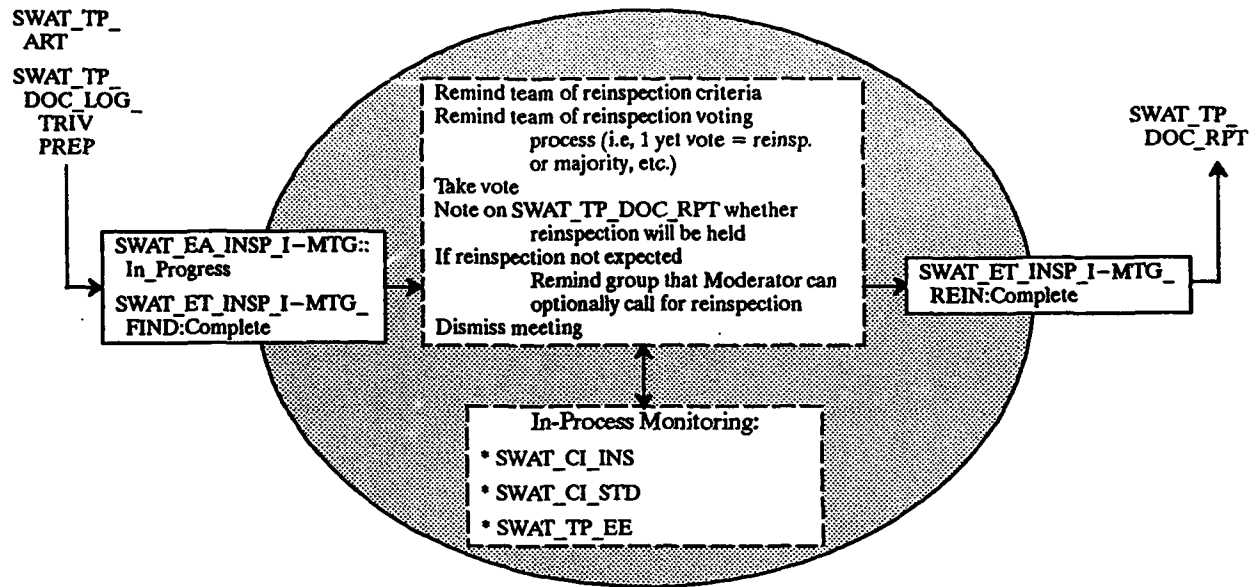 DOC_RPT

SWAT_ET_INSP_I—MTG_
  REIN:Complete

Figure B-19. ETVX Diagram 17: SW_ET_INSP_I—MTG_REIN

This page intentionally left blank.

APPENDIX C. SADT EXAMPLE OF SWAT PROCESS

When applying the SADT to software systems, the overall approach consists of:

- Identifying activities.

- Identifying the inputs and outputs of those activities.

- Identifying factors that constrain the activities.

- Identifying resources or materials that support the activities (Marca and McGowan 1988).

As Figure C-1 depicts, an activity is represented diagrammatically as a box. Inputs to an activity are labeled arrows arriving at the left side of the box. Outputs from an activity are labeled arrows departing from the right side of the box. Constraining influences are labeled arrows arriving at the top of the box, and enabling mechanisms are labeled arrows arriving at the bottom of the box.
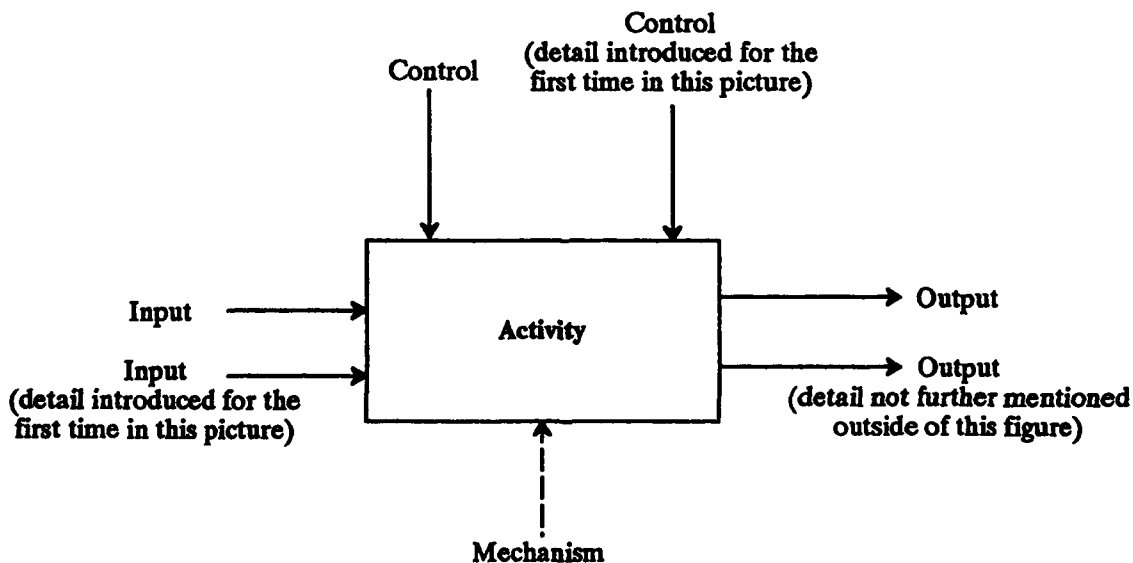

Figure C-1. SADT Diagram

The outputs from one box may be the inputs, controls, or enabling mechanisms for any other box (including, in rare cases, itself). Boxes are all named, and all arrows carry labels. Arrows are allowed to be split into multiple branches or join to combine multiple branches into one. Any box can be decomposed into any number of subboxes. These, in turn, can be decomposed. You can continue such decomposition until the necessary level of detail has been achieved. Inputs, outputs, and controls

define the interfaces between boxes, and enabling mechanisms allow controlled mixing of subjects. When a box is "exploded" to yield a new subordinate diagram, the box and diagram boundaries must match.

Commonly, it is the throughputs from the templates that are modeled in SADT as arrows arriving at the left side of a box and departing from the right side of a box. Constraints (especially external constraints) are modeled as "control" arrows arriving at the top of an SADT box. The support templates (roles and resources) can be used (from the event perspective) to represent the enabling mechanisms. Further, SADT also allows for capturing a considerable amount of text information, so the comment field and other relevant information can also be ported across.

SADT is perhaps the most popular process notation being used to date. It has been widely used for software system design, and there are a number of automated tools available on the market which support the technique. SADT is capable of being used for large scale process definition, but SADT is still not formal enough. For example, an SADT link can only carry the syntactic structure of the process information, but it cannot carry the semantics of the enactment. For SADT to be enactable, a process engineer must define the type of link on the SADT diagram then implement the semantics of the link. Along with the SADT diagram, the process engineer should also use the data dictionary for implementing enactment. Other SADT limitations include the number of blocks in a diagram along with the difficulty of representing the concept of role play for a process. However, SADT can be relatively easy to use for defining large processes, since it is capable of representing higher level abstraction and process decomposition structure.

C.1 SADT EXAMPLE

The SWAT process has been modeled as a four level (0–3) event tree in the process template in Appendix A. As shown in Figure C-2, the SWAT is the root process; the templates are translated to SADT according to the approach listed in Section 5. In the SADT example, there are five SADT diagrams, as they are marked in the Figure C-2. Figure C-3, is the SADT Diagram0 in the event tree. It is the top level of the event tree. Figure C-4 SADT Diagram1, Figure C-5, is the SADT Diagram2, Figure C-6 is the SADT Diagram3, and Figure C-7 is the SADT Diagram4.
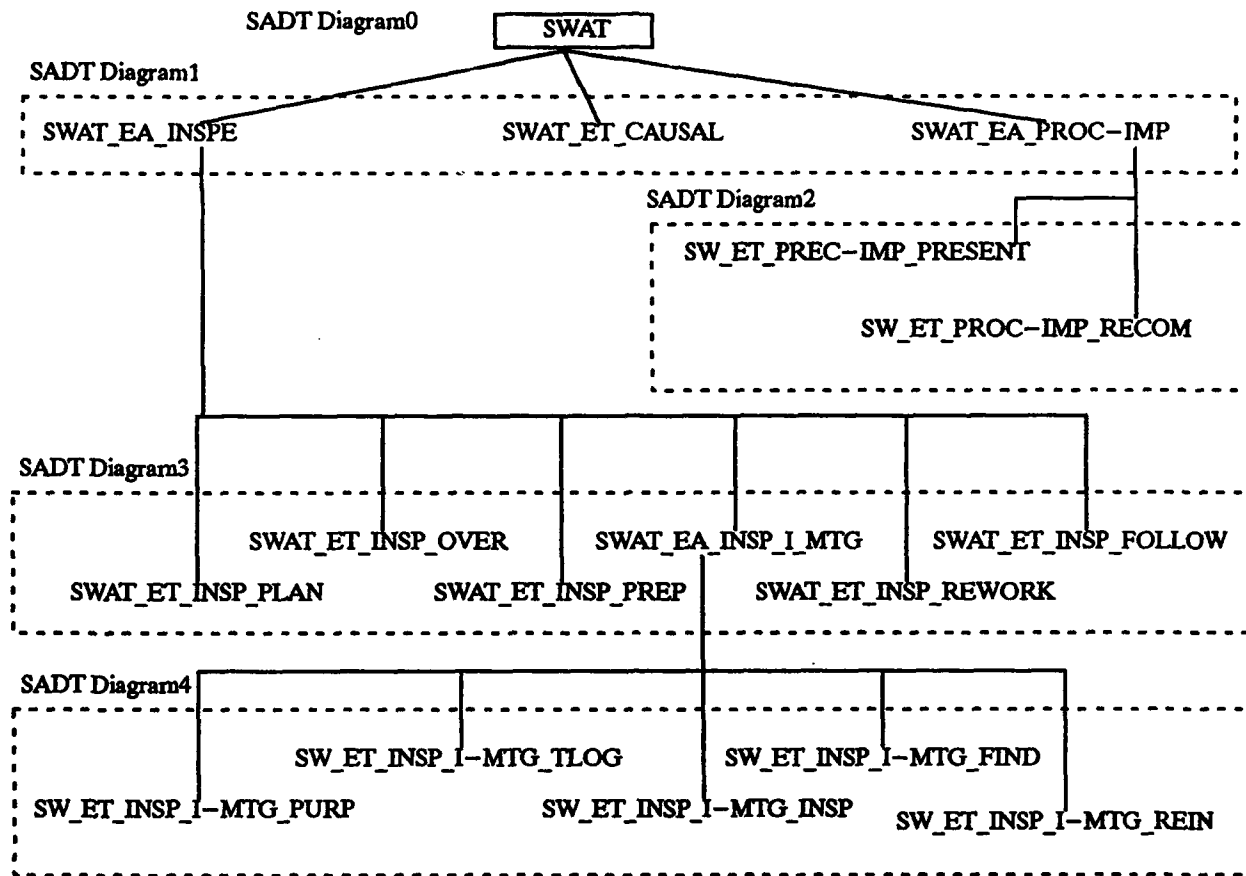
SADT Diagram0 — SWAT

SADT Diagram1

SWAT_EA_INSPE   SWAT_ET_CAUSAL   SWAT_EA_PROC−IMP

SADT Diagram2

SW_ET_PREC−IMP_PRESENT

SW_ET_PROC−IMP_RECOM

SADT Diagram3

SWAT_ET_INSP_OVER   SWAT_EA_INSP_I_MTG   SWAT_ET_INSP_FOLLOW

SWAT_ET_INSP_PLAN   SWAT_ET_INSP_PREP   SWAT_ET_INSP_REWORK

SADT Diagram4

SW_ET_INSP_I−MTG_TLOG   SW_ET_INSP_I−MTG_FIND

SW_ET_INSP_I−MTG_PURP   SW_ET_INSP_I−MTG_INSP   SW_ET_INSP_I−MTG_REIN

Figure C-2. Parent-Child Event Tree Structure

SWAT_CI_STD   SWAT_CE_VP

SWAT_CE_PPG   SWAT_CI_INS

SWAT_TP_ART ——→   ——→ SWAT_TP_MEMO

SWAT_TP_EE ——→   SWAT   ——→ SWAT_TP_DOC

                                       ——→ SWAT_TP_EE

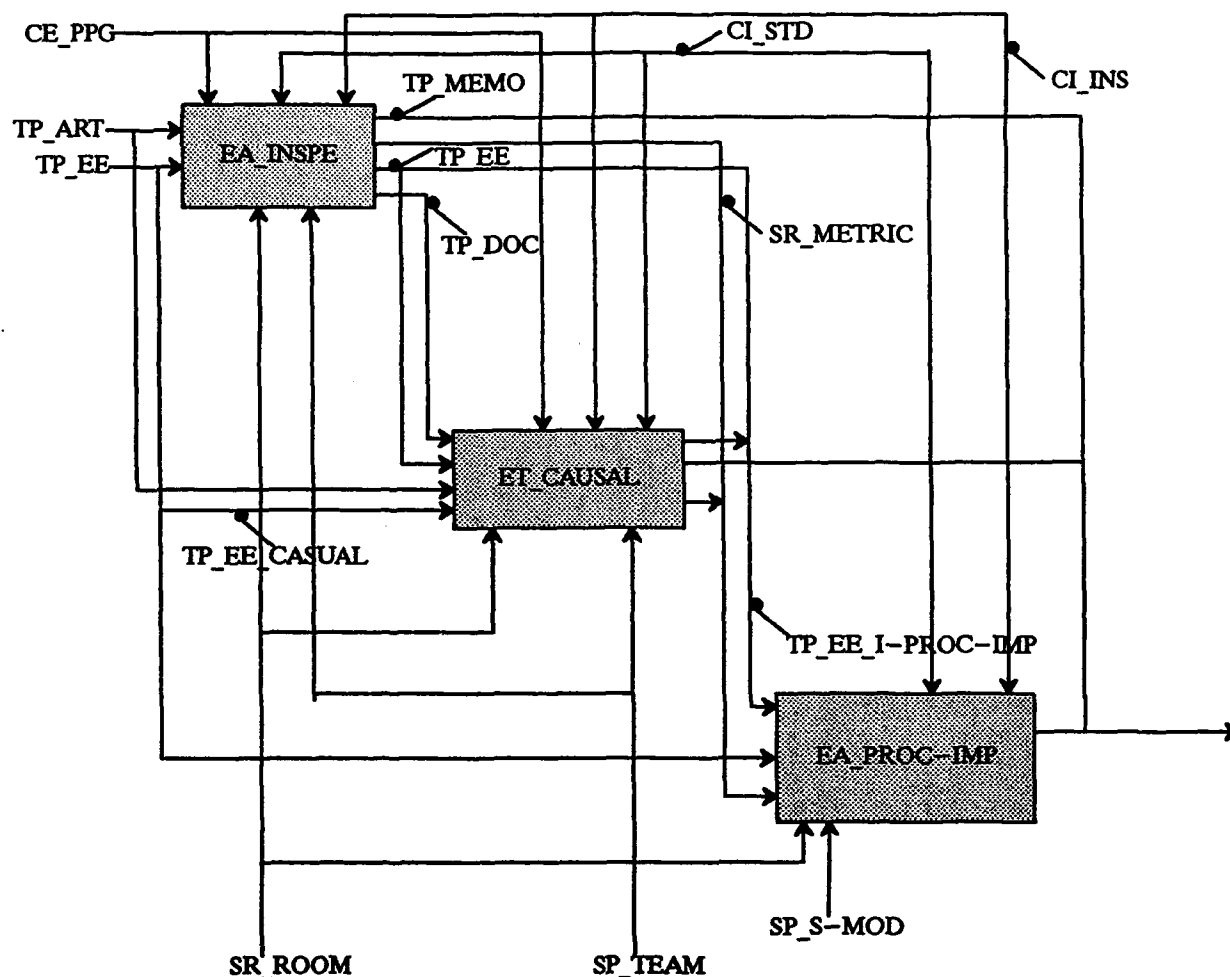SWAT_SR   SWAT_SP_TEAM
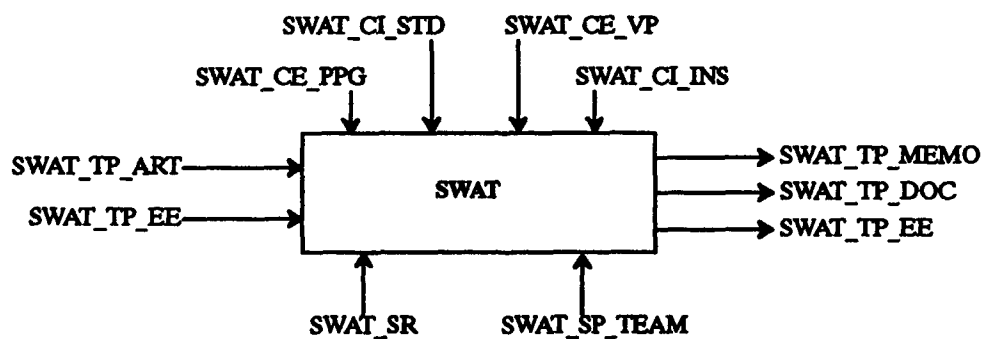
Figure C-3. SADT Diagram 0: SWAT

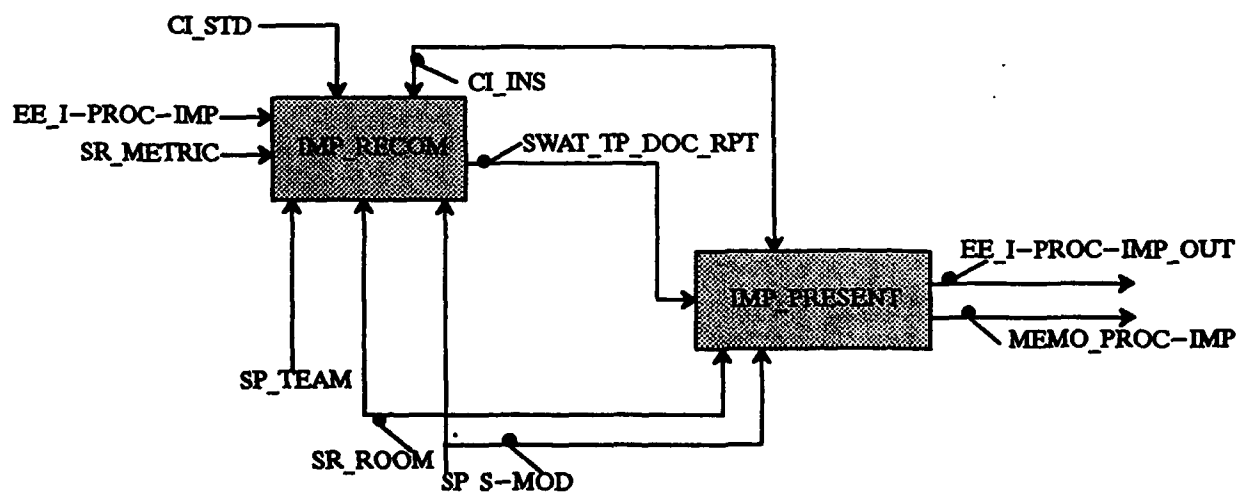Figure C-4. SADT Diagram 1: SWAT in zoom out

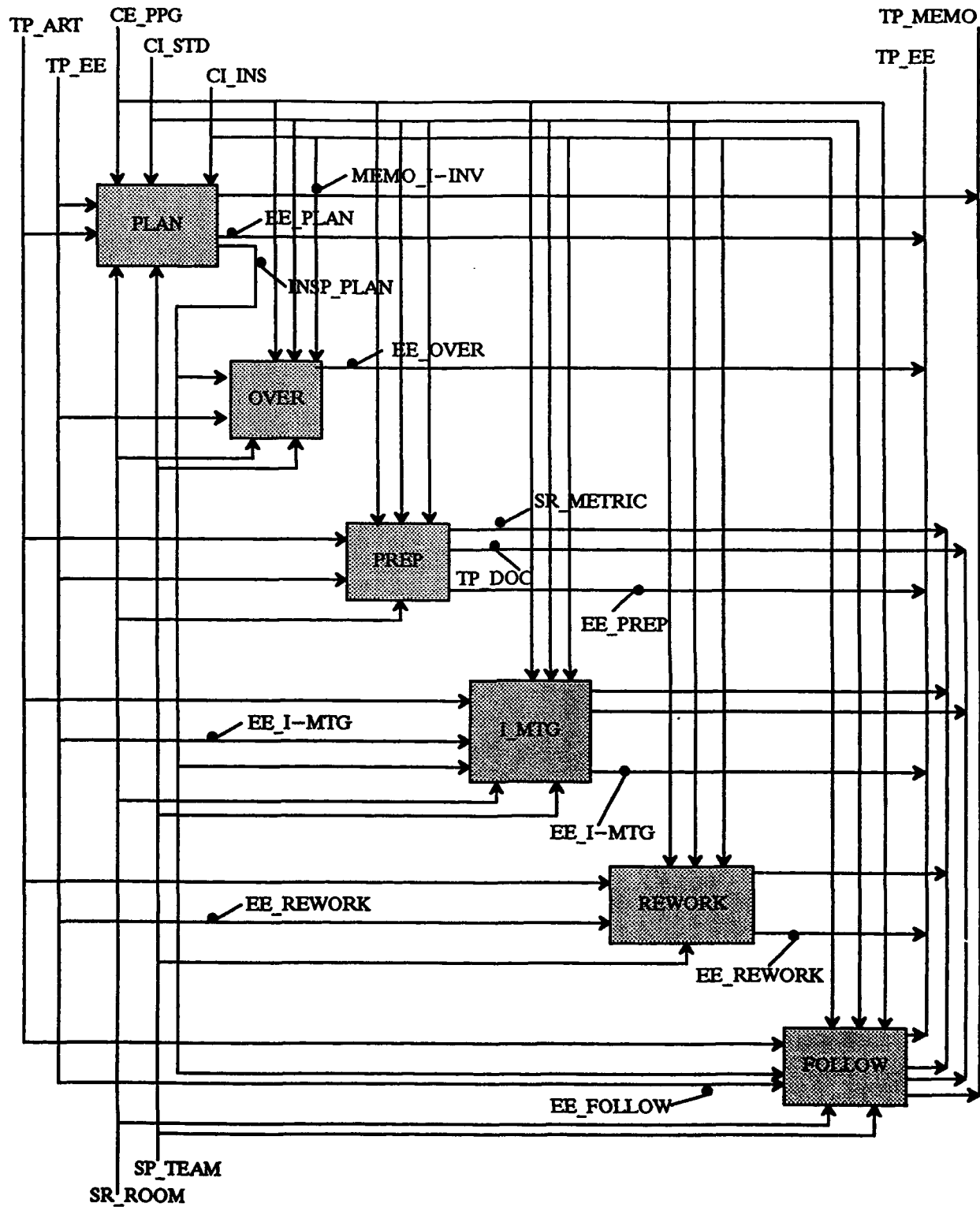Figure C-5. SADT Diagram 2: SWAT_EA_PROC—IMP

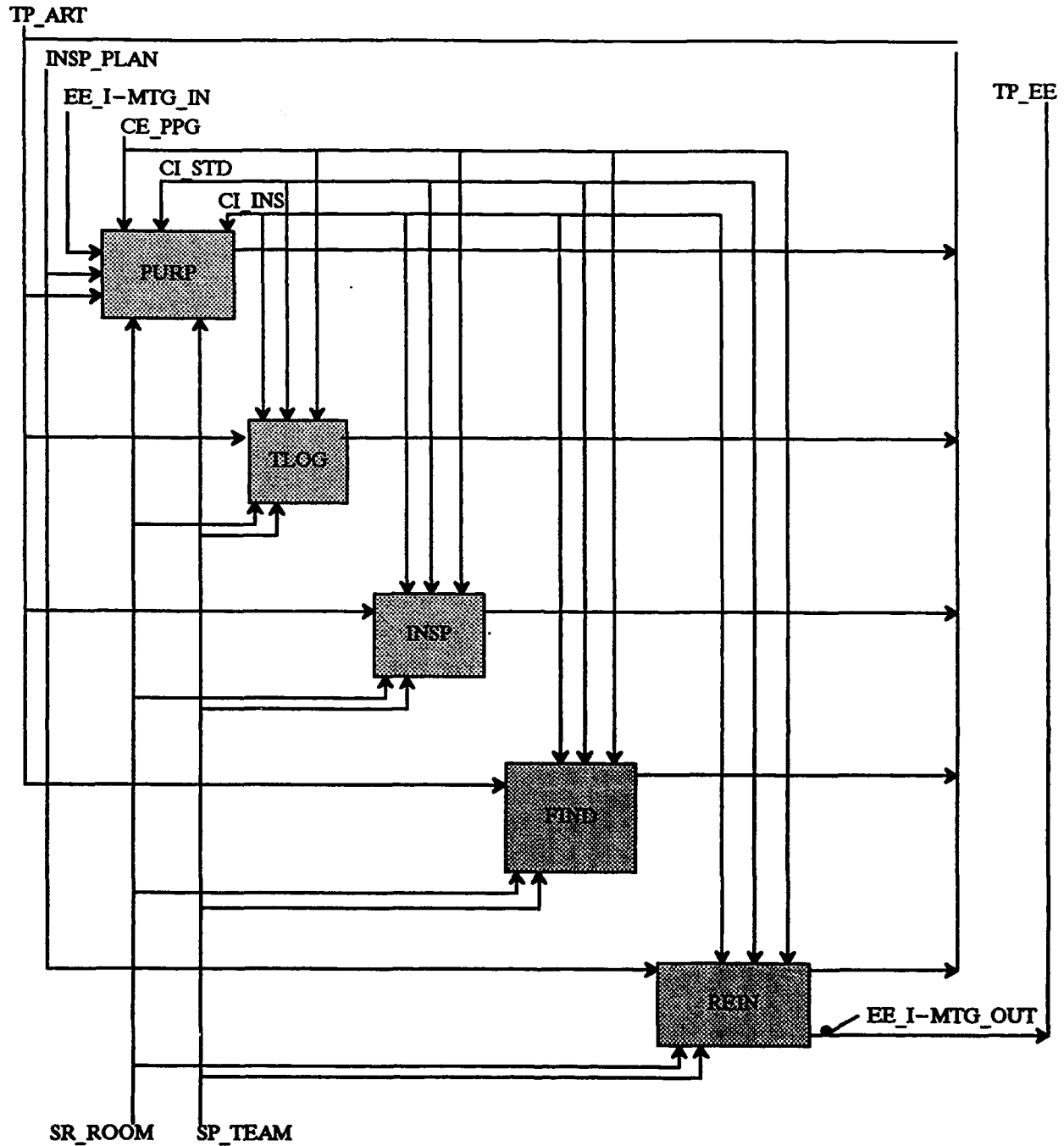Figure C-6. SADT Diagram 3: SWAT_EA_INSPE

Figure C-7. SADT Diagram 4: SWAT_EA_INSP_I_MTG

APPENDIX D. STATE TRANSITION DIAGRAMS

Shown below is a simple example of the use of state transition diagrams (STDs) to support representations of processes. STDs have a very limited set of conventions that can be used in process modeling. Although this has advantages (they are, for instance, relatively easy to learn) there are limitations to traditional state diagrams. These limitations include the difficulty of modeling parallel or simultaneous state machines that execute cooperatively. Furthermore, state machine models quickly become complex, and interpretation of such models becomes correspondingly more difficult. Virtually all the significant limitations encountered using STDs have been addressed by a superset notation called Statecharts (see Section 6 for more information on both STDs and Statecharts).

Nevertheless, STDs are still useful for presenting small models of phenomena or objects whose behavior or characteristics are describable using a relatively limited number of state transitions. The example below is quite small, so as to emphasize the potential readability of this type of diagram.

In the example scenerio from Section 7 (on which the following diagram is based), all products that are submitted for inspection transition through the same general set of states. To make this diagram more interesting, the state set for items to be inspected is expanded to include the following possible states:

- Pre-Inspection
- Pre-Planning
- Pre-Preparation
- In-Preparation
- Post-Preparation
- In-Inspection
- In-Rework
- Passed-Inspection
- Released

In Figure D-1, states are depicted in italicized, bold face type and events that cause state transitions are shown in standard typeface. States are shown as circles, and the directed arcs which connect the circles represent various events. (Further information on interpreting STDs can be found near the end of Section 5.)
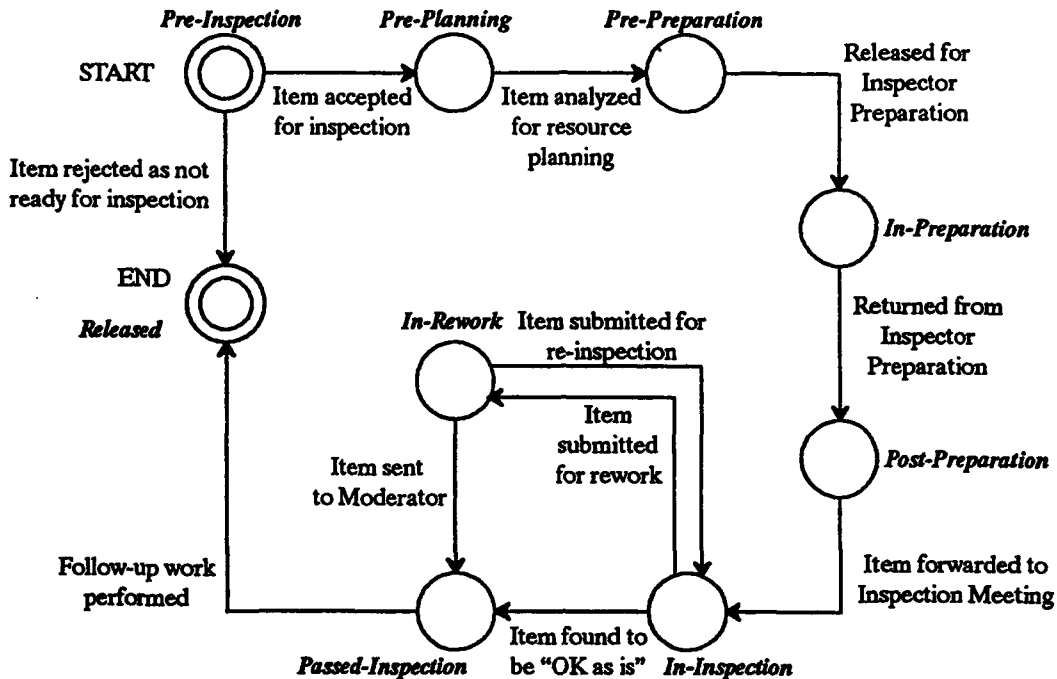
Figure D-1. State Transition Diagram

As discussed in Figure D-1, STDs are particularly useful for rendering simple diagrams of comparatively simple phenomena.

It should also be stressed that this diagram differs in perspective from the examples rendered in other notations. Specifically, this diagram does not attempt to depict the SWAT inspection process; instead, it depicts the evolution of an inspectable item through that process. However, the same notational conventions shown above can also be used to develop an STD from the perspective of the process itself.

GLOSSARY

Constraint	Process constraints describe the limiting conditions associated with the activation, performance, or cessation of an event. Whereas supports can be viewed as those things required to enable or make the right things happen, constraints can be viewed as those things required to disable or prevent the wrong things from happening. In this guidebook, constraints have been divided into two general types: internal constraints and external constraints.
Contract	A formal agreement on a set of externally observable process enactment states and results. This agreement is a process constraint, whose violation has legal consequences (which are outside the process space unless that aspect is modeled as well).
Cycle	A traversal of all four quadrants of the spiral model, which donates that some aspect of the product has matured by a specific amount.
Estimate of the situation (EoS)	A document that identifies the project's goals, strategies, product and process assumptions, and the assets available for performing a project.
Events	Events are processes, activities, or tasks.
External Constraints	External constraints include all factors that may limit or constrain how an event proceeds, that are not directly attributable to local authority (which are modeled as internal constraints). Examples of external influences that may constrain an event include quality requirements, corporate standards, division policies, engineering procedures, process guidelines, and management directives. External constraints differ from internal constraints in that they are typically not subject to discretionary use—they are intended to be, and expected to be, explicitly followed, regardless of project-specific issues.

Guidance	The use of a process definition (constraint) by an observer or process agent to provide the enacting process agent with the legal set of process step options at any point of the enactment of the observed process. This may involve process cues, process interaction, or process management.
Internal Constraints	Internal process constraints are typically managerial in nature and usually take the form of authority and permission. Examples of internal constraints include management authority or permission required before an event can commence. Internal constraints also convey authority to roles to suspend events, cancel events, re-commence events, cease events, etc. In all cases, internal constraints are always coupled with a role (typically a role signifying lead or managerial responsibility, but in all cases a role signifying—by definition—some form of authority). As a rule, internal constraints are those constraints that you have authority to change, countermand, enforce, etc.
Policy	A guiding principle; a process constraint, usually at a high level, that focuses on certain aspects of a process and influences the enactment of that process.
Precision	The degree to which the process definition completely specifies all the actions needed to produce accurate results. That is, a precisely defined process, executed with fidelity, produces an accurate result.
Predictability	An indication that either the process is intended to terminate and does terminate, or that the process is intended to be nonstop and that it does continue until terminated by a control process (or its agent).
Process	A series of actions or operations conducing to an end. A series of actions intended to reach a goal, possibly resulting in products.
Process architecture	A conceptual framework for incorporating process elements in consistent ways (or for signaling that the process element is incompatible with the architecture). A framework within which project-specific processes are defined.

Process control	The external influence over process enactment by other enacting processes. This influence may be driven by process evaluation and may be through control of the process enactment state, reassignment of resources, or change of process goals through process evolution.
Process definition	An instantiation of a process design for a specific project team or individual. It consists of a partially ordered set of process steps that is enactable. Each process step may be further refined into more detailed process steps. A process definition may consist of (sub)process definitions that can be concurrently enacted. Process definitions, when enactable by humans, are referred to as process scripts. Process definitions for nonhuman enactment are referred to as process programs.
Process evolution	The evolution of process definitions (static) as well as the evolution of enacting processes (dynamic), e.g., nonstop processes. Both static and dynamic change must be managed to ensure stability of the process and control over the process results.
Process model	A possibly partial process definition for the purpose of modeling certain characteristics of an actual process. Process models can be analyzed, validated, and, if enactable, simulates the modeled process. Process models may model at the process architecture, design, or definition level. Process models are at times used to predict process behavior. Process models themselves have an architecture, a design, and a definition.
Process modeling	Process modeling both extends and constrains process definition by requiring that the process model adheres to a predefined set of objects, relationships, methods, and structural conventions, the latter of which are often rendered graphically.
Process representation	A general term referring to the combined or sequential efforts of jointly performing process definition and process modeling.

Process supports	A process support is any non-throughput item that is needed by an event for the event to be performed. Events need throughputs, as that typically is the purpose of events: to accept one or more throughputs, modify, manipulate, inspect, and possibly create one or more new throughputs, and pass those along to other events. However, more is needed by an event than just the throughputs. These non-throughput items are all modeled as supports. Two common types of support include roles and resources.
Products	Product artifacts are tangible throughputs within a process. This type of throughput typically represents the vast majority of artifacts that pass through a process. Examples include code modules, end-user guidebooks, circuit boards, and anything else tangibly produced by a process. Products can be decomposed into subproducts, sub−subproducts, etc. This decomposition is captured within a model by the inclusion relation.
Project	An enactable or enacting process whose architecture has control processes (project management) and enacting processes performing the project tasks.
Project management	An enactable or enacting process whose goal is to create project plans and, when authorized, instantiate them, monitor them, and control their enactment. These responsibilities are commonly known as project planning (i.e., development of process plans) and project control (i.e., process evaluation of plan information and process control to make adjustments, if necessary).
Project manager	A human agent enacting the control process responsible for the execution of a project.
Redundancy	A process task or step that is not required by an error-free enactment. Redundancy thus compensates for human or other errors in process enactment.

Research	Research is a by-product of a process; but it differs from products in that research is considered intangible. If for instance, the research leads to a technical paper, that technical paper is modeled as a product. However, if experiments or investigations are being performed within one or more events, but nothing tangible is available as evidence of the work, the throughput can still be explicitly modeled as a research (intangible) artifact. As with products, research can be decomposed into subresearch, sub-subresearch, etc. This decomposition is captured within a model by the inclusion relation.
Resources	Resources are nonhuman items needed to support an event. Examples include equipment, office space, supplies, funding, etc. All items that might be required to support an event can be modeled as resources. Resources can be decomposed (using the inclusion relation) so that while one level of event abstraction shows that the training building is required, at a lower or more detailed level of abstraction the support might show that only a small classroom is actually required.
Robustness	The degree to which the process rejects unauthorized process control and/or modification (intrusion).
Role	Roles commonly represent either individual humans or humans working in concert toward a common goal or set or goals. Consequently, "programmer," "manager," "clerk," etc., all define roles that can be assumed by individuals. However, "programming team," "inspection department," and "quality assurance division" also define roles. In the latter case, the roles are essentially organizational roles as opposed to individual. For process definition, roles can be defined at all levels of abstractions.
Spiral	One or more cycles.
Task	A process (step), typically enacted by a human, requiring process planning and control.

Throughput

A throughput is either a tangible or intangible artifact. Throughputs usually refer to intermediate and final (sub)products of the software development process. They can be a physical artifact, usually modeled as a product (such as a module, a document, or a schedule) or an intangible artifact, such as the knowledge gained from having performed research.

REFERENCES

Boehm, Barry W. 1981	*Software Engineering Economics*. Englewood Cliffs, New Jersey: Prentice-Hall.
1986	A Spiral Model of Software Development and Enhancement. *ACM Software Engineering Notes* 11:22–42.
1988	A Spiral Model of Software Development and Enhancement. *IEEE Computer* 21:61–72.
1989	*Tutorial: Software Risk Management*. Washington, D.C.: IEEE Computer Society Press.
Campbell, I. 1986	*PCTE Proposal for a Public Common Tool Interface*. Software Engineering Environments, IEEE Computing Series 7.
Charette, Robert N. 1991	*Risk Management Seminar*. Herndon, Virginia: Software Productivity Consortium.
Coleman, Glenn L., Charles P. Ellison, Gentry P. Gardner, Daniel L. Sandini, and John W. Brackett 1990	*Experience in Modeling a Concurrent Software System Using STATEMATE*. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering, 104–108.
Cruickshank, R.D., and J.E. Gaffney, Jr. 1992	A Software Cost Model of Reuse Within a Single System. *Conference on Analytical Methods in Software Engineering Economics*. McLean, Virginia: MITRE Corp.
Department of Defense 1985	*Technical Reviews and Audits for Systems, Requirements, and Computer Programs*, DOD-STD-1521B. Washington, D.C.: Department of Defense.
1988	*Military Standard: Defense System Software Development*, DOD-STD-2167A. Washington, D.C.: Department of Defense.
Feiler, Peter H., and Watts S. Humphrey 1992	*Software Process Development and Enactment: Concepts and Definitions*, CMU/SEI-02-TR-4. Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University.

Haral, David
1988

Statecharts: A Visual Formalism for Complex Systems. Department of Applied Mathematics, The Weizmann Institute of Science, Rehovat, Israel. 1986. (Eventually published under the same title in 1987 in *Science of Computer Programming*, 8(1987):231–274.)

Henderson, W., and P. Taylor
1991

Embedded Processes in Stochastic Petri Nets. *IEEE Transactions on Software Engineering* 17, 2.

Humphrey, Watts S.
1990

Managing the Software Process, SEI Series in Software Engineering.

Kellner, M.
1989

Representation Formalisms for Software Process Modeling. *Proc. 4th International Software Process Workshop.*

Kolman, Bernard, and
Robert C. Busby
1984

Discrete Mathematical Structures for Computer Science. Englewood Cliffs, New Jersey: Prentice-Hall Inc.

Levis, Alexander
1992

Class lecture notes during ECE590/SYST659 Spring 1992 course at George Mason University.

Marca, David A., and
Clement L. McGowan
1988

SADT: Structured Analysis Design Technique. McGraw-Hill Book Company.

Minsky, N., and
D. Rozenshtein
1988

A Software Development Environment for Law-Governed Systems, Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.

Radice, Ronald A., and
Richard W. Phillips
1988

Software Engineering: An Industrial Approach. Englewood Cliffs, New Jersey: Prentice-Hall Inc.

Rodden, T., P. Saywer, and
I. Sommerville
1988

Interacting with an Active, Integrated Environment, Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.

Sanden, Bo
1992a

Software Systems Construction: Sequential and Concurrent Designs Implemented in Ada. Pre-Published Textbook used at George Mason University for Spring 1992. Material is the property of Bo Sanden and Prentice Hall.

1992b

3.0 Statecharts. Supplemental Handout #2 for INFT821. Spring 1992: George Mason University.

Singh, Baldev, and Gail L. Rein
1992

Role Interaction Nets (RINs): A Process Description Formalism, CT-083-92. MCC Technical Report.

Singh, Baldev
1992

Interaction Roles: A Coordination Model, CT-084-92. MCC Technical Report.

Software Productivity
Consortium
1991

Evolutionary Spiral Process Guidebook, SPC-91076-MC. Herndon, Virginia: Software Productivity Consortium.

1992a

Process Engineering With the Evolutionary Spiral Process Model, SPC-92079-CMC. Herndon, Virginia: Software Productivity Consortium.

1992b

Software Measurement Guidebook, SPC-91060-MC. Herndon, Virginia: Software Productivity Consortium.

Stenning, V.
1986

An Introduction to ISTAR, Software Engineering Environments, IEEE Computing Series 7.

Strelich, T.
1988

The Software Life Cycle Support Environment (SLCSE); A Computer Based Framework for Developing Software Systems. Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.

Taylor, R. et al.
1988

Foundations for the Arcadia Environment Architecture. Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.

This page intentionally left blank.

BIBLIOGRAPHY

Britton, K.H., R.A. Parker, and D.L. Parnas. "A Procedure for Designing Abstract Interfaces for Device Interface Modules," *Proc. 5ICSE*. 195-204, 1981.

Clements, P.C., R.A. Parker, D.L. Parnas, J.E. Shore, and K.H. Britton. *A Standard Organization for Specifying Abstract Interfaces*. NRL Report 8815, June 14, 1984.

Curtis, B. *Modeling, Measuring, & Managing Software Development Process. The M3 Life Boat for Software Tarpits*. Tutorial #4, The 13th Internal Conference on Software Engineering, 1991.

Dijkstra, E.W. "Co-operating Sequential Processes." *Programming Languages*, Edited by F. Genuys. New York: Academic Press, 43-112.

Feiler, Peter, and Watts Humphrey. *Software Process Definitions Draft Document*. Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie-Mellon University, 1991.

Guindon, R. *A Framework for Building Software Development Environments: System Design as Ill-structured Problems and as an Opportunistic Process*. MCC Technical Report STP-298-88, 1988.

Kirby, J. Jr., R.C.T. Lai, and D.M. Weiss. "A Formalization of a Design Process." *Proc. 1990 Pacific Northwest Software Quality Conference*. Oct. 29-31, 1990:93-114.

Osterweil, L. "Software Processes Are Software Too." *Proc. 9ICSE*. March 1987.

Parnas, D.L. "On the Criteria to be Used in Decomposing a System into Modules." *Communications of the ACM* 15,12 (1972):1053-1058.

Parnas, D.L. and P.C. Clements. "A Rational Design Process: How and Why to Fake It." *IEEE Transactions on Software Engineering*, February 1986.

Potts, C. "A Generic Model for Representing Design Methods." *Proc. 11ICSE*, 1989.

Potts, C., and G. Bruns. "Recording the Reasons for Design Decisions." *Proc. 10ICSE*, April 1988.

Rendes, Barry, and Ralph M. Stair, Jr. *Quantitative Analysis for Management*. Third. Allyn and Bacon Inc., 1988.

This page intentionally left blank.