Software Process Development and Enactment: Concepts and Definitions

Peter H. Feiler and Watts S. Humphrey

Carnegie Mellon University
Software Engineering Institute

ABSTRACT

The scientific treatment of the software process is relatively new and, as with any new field, the initial terminology is often confusing. When terms can have a diversity of meanings, technical communication is more difficult and technological progress is constrained. This paper defines a core set of concepts about the software process. These concepts are intended to facilitate communications and to provide a framework for further definitions. The definitions focus on essential concepts; however, they do not represent a comprehensive glossary of common software process terms. Following an initial overview, this paper outlines the basic process concepts which underlie the definitions. The definitions are then grouped in four sets: a framework for process definition, an engineering of process, an enactment of process, and process properties. This is followed by illustrations of the use of these concepts in several domains. The paper concludes with some observations on the definition process.

1. Overview

This report includes descriptions of some basic "core" software process terms. Its purpose is to provide a common communication framework for the software process and to reflect the views and findings of leading software process researchers. With the growing scientific focus on the software process, there is a need for common terms and definitions. It is hoped that this paper will facilitate communication and foster continued development of software process technology. It is also hoped that this basic definition set will serve as a foundation for further definitions and ultimately a comprehensive glossary of software process terminology.

This effort was initiated at the 6th International Software Process Workshop (ISPW6). Several workshop participants noted the need for a document to describe and define the terms commonly used by the software process community. A small group headed by Peter Feiler and Watts Humphrey volunteered to develop these definitions. The plan was to focus on a core set of approximately 25 to 30 terms that covered the basic set of process concepts. With input from many sources, the authors have established a core set of definitions and documented them in this report. To limit the number of terms, terms whose definitions can be easily mapped to the abstract concepts defined here have been omitted. To constrain the scope of this work, many terms with applicable existing definitions have also been omitted. Thus, this document does not represent a comprehensive glossary of terms in the software process domain.

An appreciation of the importance of definitions can be seen from the work of Simon and others. They have shown that people retain information in chunks [Simon 81]. People typically think of these chunks as units, and they have widely varying chunk "vocabularies" with which they are fluent. There is also evidence that expert knowledge is built by the accumulation of an expanding store of such "chunks." It would thus appear that one's ability to think and to communicate can be substantially enhanced with a precisely defined set of rich abstractions. As a broad community of professionals arrives at a common understanding of terms to describe common abstractions, it is better able to communicate succinctly and precisely. In advanced technical work, communication permits new work to be built on prior achievements. Improved communication thus facilitates more rapid technological advancement and more rapid application of that technology to the betterment of humankind.

Process concepts are being applied to software with increasing success ([Humphrey 91], [Kolkhorst 88]) but the rate of application of these concepts is limited both by the relatively primitive state of knowledge in this new field and by the lack of a common and precise basis for technical communication. The first requirement for precise communication is an agreed-upon core of terminology. Without such agreement, people are less able to understand others'

work and to build upon it. As software evolves from a craft to an engineering discipline, technical advances must draw on a broader context than can be reached through personal experience. The basic reason for this report is to propose an initial foundation for communication on software process.

Section 2 of this paper provides the conceptual context for the definitions, followed, in Section 3, by the software process definitions. Many of the terms are accompanied by explanatory comments. Section 4 illustrates the use of these abstract concept definitions to describe some common process terms. Section 5 contains a brief summary of the authors' views on the state of process technology and what can be expected in the future.

2. The Software Process Context

The meanings of terms are often dependent on the contexts in which they are used. The definitions in this paper were developed within the context of the authors' views and opinions on the software process. Rather than require the reader to infer the context for these definitions, the paper starts with a brief discussion of this context. It is the authors' view, however, that as other fields apply process principles, many of these definitions will be found applicable.

A definition framework has been found to be useful for thinking about the software process and the most critical terms needed to define and discuss it. It has helped to detect gaps and has clarified the relationships of the various terms. The selected structure is shown in Figure 2-1, which identifies the overall relationships among software and process activities and indicates the activities to which the various definitions relate. This overall structure has been helpful in this definition work and it may also be useful to the reader.

Another context question concerns the scope of process issues to be covered. While there is considerable interest and activity on the broad subjects of process management and process improvement, the scope of these definitions has been limited intentionally to definition, modeling, and enactment issues. This does not imply that the broader management issues are not important, but that their inclusion would extend the scope of this work far beyond the authors' original intent. There is also a growing literature on process management topics that provides at least some definitional guidance ([Humphrey 89], [Paulk 91]).

Finally, there is no simple way to limit the scope of a definition document. In a subject as new as the software process, many ordinary terms can have subtle meanings. The selection of terms to include in these definitions was thus based on two criteria: the term represents an essential process concept that cannot easily be derived from other concepts, or the term is currently being used inconsistently. As a result, terms such as model, definition, and fidelity
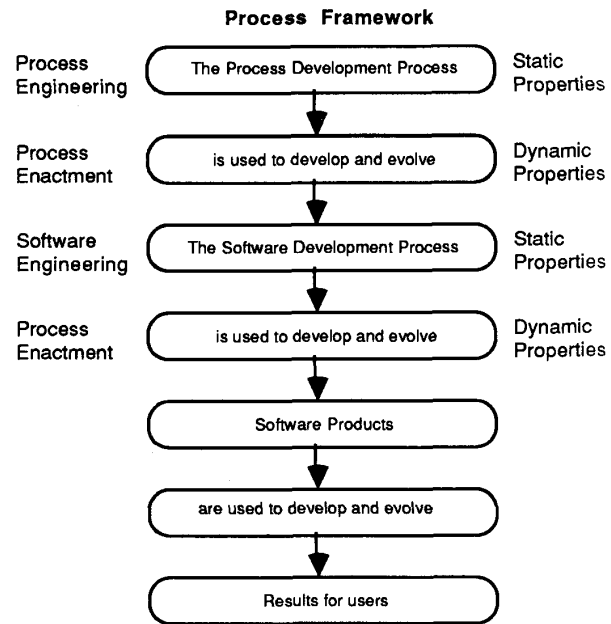
Process Framework

Figure 2-1 Structure of Process Concepts

have been included while subprocess, template, or cue have not.

2.1: A Framework for Process Definition

Software development organizations have found that by defining their processes they improve their effectiveness ([Humphrey 91], [Kolkhorst 88]). To the extent that software process definitions[1] make high-quality software easier and more economical to produce, they will become widely valued and used. This means that software process definitions must be both useful to the practitioners and reasonably economical to produce. Experience to date, however, demonstrates that the development of a comprehensive software process definition can be very expensive and time consuming. Thus, there is a premium on widely applicable means for developing general purpose process definitions together with techniques for reusing, tailoring, and enhancing them. Just as with software, this implies that large scale software processes should be carefully designed and constructed.

The concept of a software process architecture can be best described by examining how organizations are likely

---

1. The reader should note a potential confusion with terminology. This paper is about the definition of process terms. One such term, as used in this paragraph, is "process definition," which is later defined as an implementation of a process design in the form of a partially ordered set of process steps that is enactable.

to use process definitions. Rather than having a single monolithic process that all projects must use, they will likely find that different projects will have differing needs. For example, the development project for enhancing a large, widely-used product will likely require some different process activities from a project to develop a new program for a single user. Since process development is expensive, there is considerable motivation for process commonality and sharing. This is enhanced by the fact that different projects in the same organization will likely have many common activities. Large organizations will have many process definitions that they wish to share. The more logical and explicit the relationship among these definitions, the more likely it is that elements of the various project processes can be shared.

One way to address the need to share process definitions is to develop a set of general purpose, reusable process elements. Generality, however, requires interfaces and structural standards; a complete set of such interfaces and standards comprises a process architecture. A properly conceived process architecture should permit its member process elements to be more widely used and thus to be more economically viable.

Another question concerns the degree of process refinement. A complex software process can be viewed as a nested set of abstractions. Each process is composed of subprocesses, each of which in turn may also have smaller elements. While there is no clear technical limit to the level of refinement for a software process, there are practical concerns, including the scale of the projects for which the process is designed, the degree to which the work is partitioned among implementors, the resources and time available for process definition, the level of capability or understanding of the process users, and the scalability of the process itself.

2.2: Engineering of Processes

The software process can be viewed in much the same way as software. It has many of the same artifacts and requires quite similar disciplines and methods [Osterweil 87]. It is useful to think about the software process development life cycle in software development terms. For example, one should start with clear requirements followed (or, more properly, accompanied) by architecture and design. Processes must be validated against users' needs, and limited prototypes may be needed before full scale development is undertaken. Special testing is required, as is planning, instantiation, migration from the prior process, and operational support.

Regardless of the degree of testing, all process bugs are not generally found before general process instantiation and enactment. Because software processes typically require at least partial human enactment, there is a major requirement and usability problem and it is much more difficult to do effective testing without end-user involvement. It is likely that only a small fraction of the total num-

ber of process "bugs" will be found in early laboratory testing. It is thus advisable to follow process development testing with early user prototype testing. Even then, as the projects evolve and the software professionals gain experience with the process, there will be many ideas for further improvement. It is thus important for organizations to establish mechanisms to obtain continual user feedback to guide process repair and evolution.

The software development community is still learning that timely and comprehensive user feedback is crucial to a quality development process. In developing the software process, it takes considerable experience before process designers can produce processes that are directly usable without major modification and evolutionary improvement. Process usability is a function of process design, user experience, the project domain, and many other factors. Because user needs vary widely even within a single organization, and because the needs of each user will change with experience, they must be thoroughly and regularly monitored. Direct, continuous, and comprehensive user involvement is thus essential for effective process development and evolution. Such involvement will generate improvement suggestions that must be handled. This in turn will require process support facilities for tracking, recording, handling, and installing process fixes and enhancements.

2.3: Enactment of Processes

It is desirable to define software processes with sufficient precision so that many of the routine enactment tasks can be automated. Software engineering remains a highly creative process, and for the foreseeable future, the opportunities for process automation will likely be limited to the most routine activities and tasks. As a consequence, software process enactment issues must consider human agents as well as automation through machine agents. The use of human agents raises issues of planning, controlling, monitoring, enforcing, and training. The software process must also be monitored, measured, and repaired and it must relate to other processes and activities within the organization. There are even questions of unauthorized intrusion, improper process modification, and process recovery.

In short, processes have the full range of enactment properties seen with data processing systems. At one extreme are small, largely autonomous, single string processes; at the other are highly structured networks of interacting parallel processes. The issues of designing, planning, monitoring, and controlling must also vary considerably across this spectrum.

2.4: Process Properties

Software processes must be evaluated. What constitutes a good process and how can one tell if a particular process fits a specific user need? The first basic requirement is that the properties of a specific process should fit

30

the needs of the project using them. While a comprehensive discussion of process assessment and evaluation is beyond the scope of this paper, there is a growing body of literature on these subjects. Software process assessments have been widely used by U.S. software organizations for several years [Humphrey 89] and there are now a number of organizations that conduct such assessments as a business. The U.S. Department of Defense has also adopted an SEI-developed capability evaluation method for determining the most effective software contractors from among several offerers. The SEI is also developing a Capability Maturity Model which is a comprehensive listing of those practices that are appropriate for various levels of software process capability [Paulk 91].

The current state of software process technology is such that process evaluation is currently limited to a rudimentary examination of the presence or absence of various activities. As this field evolves, more comprehensive evaluations will be practical. Such evaluations will likely examine the process structure, its behavior under various conditions, and its adaptability. At this time, it is premature to attempt quantitative measures of process quality. However, there are several available qualitative measures. These relate to how difficult the process is to use and the quality of the results it produces. Other properties concern the degree to which the process has persistence or whether it merely executes short tasks in response to an external agent.

3. Software Process Definitions

This chapter presents the selected "core" process terms together with their definitions and explanatory comments. The definitions are formatted as follows: the term being defined is in **boldface**, the definition for the term is in *italics*, and any rationale or explanatory information is in plain text following the definition. Unless otherwise noted, these terms are all defined in the context of the software process. However, since we have attempted to abstract the terms from the particular domain of software process, they may have broader applicability. Where this is confusing, it is suggested that the reader add the word "process" before any term. For example, "development" equates to "process development."

The first definition is the term process, which is defined at a highly abstract level. It is followed by two terms that refer to pieces of a process.

> **Process:** *A set of partially ordered steps intended to reach a goal.* While the term process is used in many different contexts, the context for this definition is software. For software development, the goal is production or enhancement of software products, or the provision of services. Other examples are the software maintenance process, the acceptance testing process, or the process development process.

Process Step: *An atomic action of a process that has no externally visible substructure. The process step is the basic process abstraction.* A process step is a discrete, bounded activity of finite duration with a level of abstraction that depends on the enacting context. For example, a process step, as used by a programmer and included in a process script, would generally require substantial elaboration to be suitable for a process program.

Process Element: *A component of a process.* Process elements range from individual process steps to very large parts of processes. They may be templates to be filled in, fragments to be completed, or abstractions to be refined.

Process Script: *A process definition that is suitably designed and instantiated for enactment by a human.* Process scripts are designed to adapt to the particular user's needs and often must be modified as users gain experience and facility.

Process Program: *A process definition that is suitably designed and instantiated for enactment by machine.* Process programs must be designed to fit the particular computing environmental needs for format and detail and generally be tested, debugged, and modified much like computer programs.

The remainder of the definitions are organized into several groups. First, the Framework for Process Definition defines foundation terms that are used in the subsequent definitions. Next, the terms in Engineering of Process elaborate on the process of developing process definitions. Concepts concerning the enactment and management of defined processes are described in Enactment of Processes. Finally, terms defining properties of defined processes are discussed in Process Properties.

Figure 3-1 illustrates the relationship among many of the process entities and the actions on them. Here, the boxes represent various process elements and the arrows refer to actions. Starting, for example, with an initial process architecture, a process design is developed. From there, one or more process definitions can be developed. This design and development activity may uncover architectural defects or desirable enhancements that are fed back to evolve the process architecture. Further, existing process architectures, designs, and definitions may be tailored to fit changing project needs. With a complete process definition, a plan is developed for its project use. This involves analysis and adjustment through a control process and generally involves external constraints. An effective control process will also utilize measurements through process traces. Once an appropriate plan is established, the process definition is converted to enactable form (instantiated). When

31

this enactable process is sent to the initial enactment state and a suitable agent is provided, an enacting process can be initiated. During enactment, the control process monitors performance and makes appropriate adjustments. This dynamic control phase may involve reference to the process plan, the process definition, and the process trace.

1. Tailor
2. Develop
3. Evolve
4. Instantiate
5. Initiate
6. Interact
7. Plan
8. Enact
9. Monitor
10. Analyze
11. Adjust

Figure 3-1 Process Entities and Actions

3.1: Framework for Process Definition

This section defines the basic process artifacts. The next section introduces terms that represent actions on these objects. For example, this section defines process designs as the results of designing processes.

The structure depicted here differs somewhat from current general usage in the process community. Here, we have chosen to establish a design hierarchy somewhat analogous to that used in system engineering. With systems, one starts with an overall architecture and proceeds to a design, then an implementation, and then to an operable unit. With processes, the flow is the same only we replace the

term implementation with process definition and operable unit with enactable process. In prior process parlance, the terms process program and process model have been used more or less interchangeably to refer to what are here called defined processes. This new terminology is introduced because we found that more precise terms are required. The terms model and program continue to be used in a form that is closer to traditional usage in software and computer science.

Process Architecture: *A conceptual framework for consistently incorporating, relating, and tailoring process elements into enactable processes.* An architecture provides a space of process designs. A process architecture is often needed when a process must relate to other existing or future processes. Examples of such needs are process element reuse, process enhancement, and process tailoring. An essential characteristic of a process architecture is its ability to indicate whether a process element is or is not compatible with the architecture.

Process Design (noun): *An embodiment of a process architecture that establishes the architectural options and parameters, the existing elements to be reused, the structure and behavior of the new elements, and the relationships among these elements.* A process design may be for a specific project, an entire organization, or possibly for larger classes of projects or organizations. A process design is produced to meet specified goals. The completed design includes the process definition and instantiation standards and interfaces, the overall process structure, and the functions and relationships of the process elements. This may include reusable process definitions and partially or fully populated process elements. The design specifies the selection choices to be made during process development.

Process Definition: *An implementation of a process design in the form of a partially ordered set of process steps that is enactable.* At a lower level of abstraction, each process step may be further refined into more detailed process steps. A process definition may consist of (sub)process definitions that can be enacted concurrently. Process definitions whose levels of abstraction are refined fully are referred to as complete or fit for enactment. Completeness, however, depends on context since a definition that is complete for one process agent may not be for another. A process definition may be for a class of projects, a specific project team, or an individual professional.
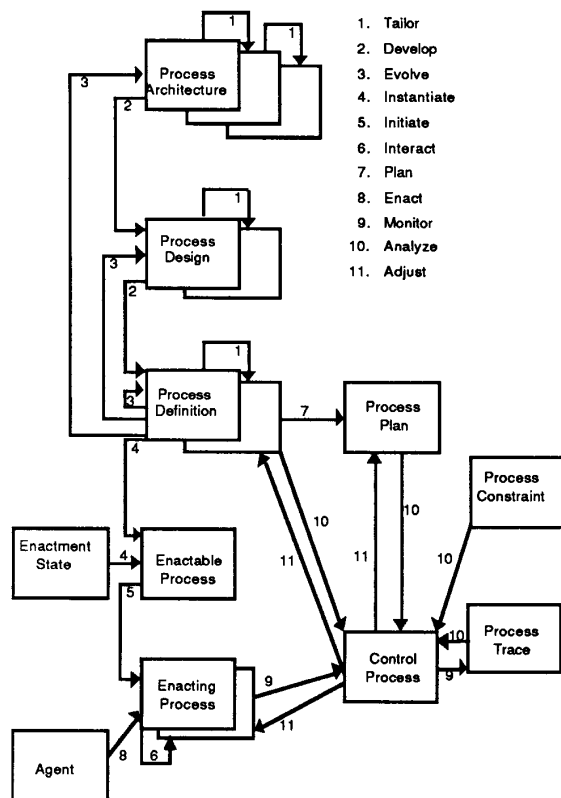
32

Process Plan: *A specification of the resources necessary for the enactment of a process definition, the relationships of these resources to process steps, the products produced by these steps, and any constraints on enactment or resources. Process plans guide the instantiation and use of processes while project plans guide the design, development, evolution, and tailoring of processes (or products).* Process plans are created with respect to a process definition containing process steps to be planned and managed. Resources include human process agents, computer resources, time, and budgets. Relationships refer to the estimation or assignment of resources to process steps in order to meet project objectives. The more common term of project plan typically contains work packages, i.e., a process definition at a certain level of abstraction, together with one or more process plans. The distinction between the project plan and the process plan is important because effective project planning is facilitated by the existence of a defined process. This often requires that the process plan be established and implemented in parallel with or even before the project plan. Because of the time and resources required, the process design and development must be done in advance of the project need.

Enactable Process: *An instance of a process definition that includes all the elements required for enactment.* An enactable process consists of a process definition, required process inputs, assigned enactment agents and resources, an initial enactment state, an initiation agent, and continuation and termination capabilities. A process that lacks any of these conditions is not enactable. It should be noted that while enactable processes may not actually terminate, at least for long periods, they must have a termination capability so they can be stopped in an orderly way when necessary.

Process Model: *An abstract representation of a process architecture, design, or definition.* Process models are process elements at the architectural, design, and definitions level, whose abstraction captures those aspects of a process relevant to the modeling. Any representation of the process is a process model. Process models are used where use of the complete process is undesirable or impractical. A process model can be analyzed,

validated, and, if enactable, it can simulate the modeled process. Process models may be used to assist in process analysis, to aid in process understanding, or to predict process behavior.

3.2: Engineering of Processes

This section includes discussions of concepts related to the engineering of processes. The engineering of a process is a process that itself can be defined and engineered.

Development: *The act of creating enactable processes. It may include planning, architecture, design, instantiation, and validation.* If the process development activity is for a complete process, all or most of these activities should be conducted. If, however, the development effort is to repair or enhance an existing process, an abbreviated set of activities might be used.

Tailoring: *The act of adapting process designs and process definitions to support the enactment of a process for a particular purpose.* Process tailoring may involve the use of process templates and may result in specialized process definitions.

Planning: *The act of developing a process plan for the enactment of a process definition.* Process planning should typically precede process instantiation and enactment. If the process being planned, for example, was process development, then the process plan would be part of the process development project plan.

Instantiation: *The act of creating enactable processes from process definitions and process plans.* Process instantiation may be static or dynamic. In static instantiation, the complete process definition is instantiated before enactment. In the dynamic case, the process development, the instantiation of individual steps of that process, and the enactment of these steps may be interleaved. The dynamic case permits enactment of processes to be started before their definition is complete. Process instantiation may involve the packaging of various process options and tailorings into a set of process scripts or the assignment of resources to planned process steps. In this latter case, instantiation must be accompanied by both a process definition and a process plan. In some cases, it may be desirable to simultaneously plan multiple instantiations to reduce the amount of planning and improve the efficiency of instantiation.

33

Evolution: *The act of changing existing process definitions to meet new needs.* Process evolution is typically intended to correct known problems or to evolve the process to meet new needs. Process evolution should be planned and carefully managed to insure the continuing integrity and quality of the resulting products. Process definitions may evolve without affecting existing enacting instances, or they may evolve together with their enacting instances. This latter case is needed for non-stop or long-running processes.

3.3: Enactment of Processes

Terms dealing with enactment of processes are grouped into four subgroups: Process Enactment, Process Control, Process Authority, and Process Assurance. Process Enactment introduces terms concerning the mechanics of enacting a process. Process Control addresses the monitoring, analysis, and adjustment of a process to direct or to improve its behavior. The Process Authority terms deal with authorization, appraisal, delegation, and intrusion. Process Assurance deals with the methods of adapting a process definition to address unexpected situations, and with the means for ensuring proper enactment of the established process definition.

Process Enactment

Process Enactment: *The execution of a process by a process agent according to a process definition.* The enactment of an actual process, also referred to as process performance, may be according to an implicit process definition. A process may be represented in process models. Enactment of a process model allows the performance of a process to be monitored and guided.

Agent: *An entity that enacts a process definition.* This entity may be a person following a process script or a machine executing a process program. The process agent interprets the enactable process.

Process Constraint: *A defined condition that an enacting process must satisfy.* Process constraints typically relate the enacting process to external processes or agents. Examples of process constraints are product requirements, authorizations, standards, and procedures. Such constraints may be included in process designs, process programs, process scripts, process architectures, or process plans.

Enactment State: *The state of a particular enactable process. The enactment state consists of:*

- *enactment flow pointers that refer to the steps in the process definition currently being enacted*
- *state information reflecting the satisfaction of process constraints*
- *the satisfaction record of conditions and dependencies*
- *current resource utilization records*
- *the status of the work products being produced*

The process enactment state is changed through enactment of a process step, through interaction with a cooperating process, or through initiation, suspension, resumption, or termination by a controlling process. The specification of the process state must also consider the states of other related entities, whether precisely defined or not. Examples of such related entities are the process plan, the associated product, the project, and the organization. The process enactment state is associated with an instance of an enactable process. The enactable process, in turn, identifies the specific version of the process definition being enacted. Since the process steps may have various levels of abstraction, their process states must also have equivalent levels of abstraction.

Enacting Process: *An active enactable process, i.e., an enactable process whose agent or agents are executing the process definition.* An enacting process may be in a suspended state if an assigned process agent is not available or other process constraints are not satisfied.

Interaction: *The interchange between two cooperating enactable processes.* This interchange may be for the purpose of communication (interchange of data) or coordination (interchange of control). The interchange may be synchronous or asynchronous, and its state data may be accessible or hidden. That is, one process may have access to state data for other processes while they have no access to its state data. Andrews discusses a set of process interaction paradigms in distributed systems, many of which can also be applied to the software process [Andrews 91].

34

Automation: *The use of machine process agents in process enactment.* Here, the use of a machine agent is facilitated by a fully developed process definition embodied in a suitable process program.

Process Control

Control Process: *A process that exists separate from an enacting process, but has access to its state data and can affect its enactment.* A control process, for example, may initiate, monitor, adjust, or terminate its controlled process or processes. The form of process control is a key process design decision. It may be fully centralized or distributed. In the distributed case, for example, one can visualize each process step behaving like a program procedure that invokes other steps. This would permit the design of recursive processes where steps may call themselves.

Monitoring: *The observation of the enactment of a process.* Monitoring is performed by a separate observer or control process that monitors the enactment state of the observed process. This observation is performed concurrently and typically with minimum interference in the behavior of the process being observed. Process monitoring may be performed by the same process agent that is enacting the process or by a separate agent. The purpose of process monitoring is to gather data on the enactment state and process resource utilization. This data is then used to provide a process measurement database for use in process planning, analysis, control, and adjustment.

Process Trace: *A sequence of snapshots of the enactment state reflecting the observed enactment history.* Tracing is one means of monitoring and measuring a process. A trace may be of resource utilization, enaction time, error occurrences, or any other desired parameters.

Analysis: *The use of process traces, process definitions, and process plans to assess process behavior.* Process analysis may be performed to determine if an enacting process is conforming to process constraints, if it is meeting plan objectives, or if process plan adjustments or process definition changes are required. Process analysis is also an important part of process evolution.

Adjustment: *The influence of a control process over the enactment of another process.* This influence may be guided by analysis of observed process data and exercised through changes to the process enactment state, through reassignment of resources, or through changes to the process definition.

Process Authority

Approval: *The granting of the right to enact a planned process or process step.* Approval is given in accordance with the established process definition and process plan by an authorized process agent. Lack of approval from a designated responsible authority (controlling process) constitutes a violation of process constraints. For example, Mr. Jones may approve a specific process change. His signature is the approval.

Authorization: *The granting of the right to give approval.* The establishment of explicit authorities and responsibilities is an essential part of process planning and instantiation. For example, Mrs. Smith authorizes Mr. Jones to approve process changes.

Delegation: *The authorizing of other agents to enact process steps that the delegating process (agent) is authorized to perform.* Delegation possibly involves refinement of the process definition before the responsibility to enact is transferred. For example, Mr. Jones may delegate his approval authority to Ms. Doe.

Intrusion: *Unauthorized process monitoring, adjustment, or repair.* One example of intrusion could be a case where a project-specific tailoring improperly changes the baseline process.

Process Assurance

Process assurance focuses on conformance of process enactment. The first three terms address non-conformance and its correction by modifying the enacting process, while the second two items refer to activities that focus on assuring that the process enactment follows the process definition.

Incident: *A failure in process enactment.* Incidents can be due to a process intrusion, the occurrence of an event that was not anticipated by the process definition, or a process definition defect.

35

Repair: *The temporary correction or adjustment of a non-conforming process to meet an immediate need.* Repairs are typically required when there is insufficient time for process evolution or when the adjustment is a unique event. Process repair actions may result in process improvement proposals for process evolution.

Recovery: *The adjustments required to allow continuing enactment after a process incident.* Recovery requires the detection of process incidents and involves the restoration of the process to a known enactment state in order to allow continuing enactment

Enforcement: *The activities used to insure that process enactment conforms to process constraints.* Enforcement may or may not be effective, depending on the capabilities of the process agent and the enforcement methods used. This is particularly the case with human process agents. Enforcement may also include the initiation of (possibly predefined) consequences in case of constraint violation.

Guidance: *The activity of providing the enacting process agent with assistance regarding the legal steps at any point during process enactment.* Guidance may be provided by a control process and may involve process cues, process interaction, or process management.

3.4: Process Properties

The process property terms relate to the properties of entire processes as well as to the properties of their elements. The process property terms are divided into two categories, static and dynamic. Static properties concern the characteristics of process definitions, enactable process, and process results. The dynamic properties concern the enactment behavior of processes.

Static Process Properties

Accuracy: *The degree to which the product produced by the process matches the intended result.* Note that neither the product produced by an accurate process nor the product intended by that process may match the actual need.

Fidelity: *The faithfulness with which a defined process is followed.* Fidelity concerns the degree with which the human or machine agents performing the process exactly follow the defined actions. Fidelity is related to

enforcement. Effective enforcement guarantees high levels of fidelity, although high levels of fidelity may occur without stringent enforcement.

Fitness: *The degree to which the people or machines enacting the process can faithfully follow actions it specifies.* A fit process definition is thus designed so the enacting process agent can faithfully follow it, while an unfit process definition may be so poorly represented as to be impractical, inconvenient, or unintelligible. Note that process fitness may be achieved through other means than process design and definition. Examples are training and technical support.

Precision: *The degree to which the process definition specifies all the actions needed to produce accurate results.* That is, a precisely defined process executed with fidelity produces an accurate result.

Redundancy: *A process step is redundant if its removal would not alter the results of the process, given that the process is executed with fidelity.* Redundancy may be added to compensate for human or other errors in process enactment, i.e., low process fidelity.

Scalability: *The breadth of activities for which the process definition is designed.* This might include the ranges in numbers of people, size of product, time duration, product life cycle, or development environment for which the process is fit and precise. A highly scalable process is thus likely to be of more general value than one that is less scalable.

Maintainability: *The degree to which the process is designed to readily permit static or dynamic process evolution.* Maintainability is achieved through localization of information, a cleanly structured design, and well architected interfaces. The purpose of such design approaches is to limit the impact of process changes and thus simplify the process change process.

Dynamic Process Properties

Dynamic process properties concern the singular behavior of a process and the nature of its interactions with other processes. The first three properties concern singular process behavior, while the latter two deal with the relationships among processes.

Lifeness: *The degree to which an enacting process that contains concurrent interacting sub-processes deals with deadlock, starvation, and termination.* The degree of lifeness indicates whether a process is intended to

36

terminate and does terminate, or whether it is intended not to stop and does not stop. Deadlock refers to the situation when two or more interacting processes wait for each other. Starvation results from improper process design or from excessive diversion of critical enactment resources, not allowing the process to progress according to plan.

Robustness: *The degree to which the process rejects intrusion.* A robust process, for example, would typically be maintained under configuration control, thus rejecting unauthorized changes to the process definition.

Fault Tolerance: *The degree to which the process, once an intrusion has occurred, either continues to produce accurate results or recognizes the inaccuracy of its results and initiates corrective action.* Here, for example, a disciplined change management system would insure that project-specific changes were isolated from the baseline process.

Autonomy: *The degree to which an enacting process operates independently of other processes.* The degree of autonomy indicates the extent to which the process and all its subprocesses are independent of other processes.

Responsiveness: *The degree to which an enacting process proactively initiates and controls other processes, or reactively responds to events from other processes.* A highly responsive process thus may be highly incremental and frequently in an enactable condition waiting for interactions to trigger further enactment. A high degree of responsiveness may entail a reduction in resource utilization efficiency.

4. Domain-Specific Use of Process Concept Definitions

The following are some examples of process terms in several domains that are expressed using this core set of terms. The purpose of these examples is to illustrate the use of the core set of terms for defining additional process terms and to provide some concrete examples of these concepts and definitions. We have chosen the domains of project management, operating systems, and process engineering.

4.1: Project Management Domain

Role: *The responsibility for enacting a process or subprocess definition.* An agent, when enacting a process, is referred to as assuming the process role. In this role, the agent is limited to the set of operations reflected in the steps of that process. These operations are specified in the script for a human agent or in the program for a machine agent. A process may have more than one participating agent.

Task: *A process step typically enacted by a human, requiring process planning and control.* This definition describes the term task as it is typically used in project management or for managing employee work assignments.

Contract: *A formal agreement on a process plan and a set of process enactment states and results.* This agreement constitutes a process constraint, whose violation has (possibly legal) consequences.

Policy: *The guiding principles for process development and/or enactment.* They often result in process constraints, usually at a high level, that focus on certain aspects of a process and influence the enactment of that process.

Project: *An enactable process or enacting process, whose architecture has control processes for managing the project and enacting processes for performing the project tasks.* A project could be to develop a product or to develop a process definition.

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, project control, and process control. When properly performed, the project management function is typically highly interactive in reviewing and approving process plans, in using process analysis results, and in making process adjustments.

Project Plan: *The project plan consists of a process definition at a level of abstraction such that its process steps have to be managed in the context of one or more plans (i.e., resources, responsibilities, and schedules).* When engineering of the process is involved, project plans should explicitly incorporate appropriate process plans.

Project Manager: *A human agent with overall responsibility for enacting the control process.* With a properly designed process, this includes all aspects of controlling and managing project execution. Project managers are also frequently involved in such related

37

activities as requirements determination, goal setting, resource allocation, contract negotiation, systems partitioning, coordination, and post contract evaluation.

4.2: Operating System Domain

In an *ACM Computing Surveys* article, Horning and Randall have surveyed and discussed operating systems concepts as they relate to process support [Horning 73]. The terms used by the authors reflect the vocabulary used by the research community at the time. The table in Figure 4-1 lists some of the most prevalent terms and relates them to the definitions in this paper. The reader may also notice that, in the context of operating systems, there is less emphasis on a priori planning, while increased emphasis is placed on dynamic scheduling and different forms of process interaction. The terms abstraction and refinement have been implicitly introduced in the context of the definition of process steps in this paper, while they have been spelled out explicitly by Horning and Randall.

4.3: Process Engineering Domain

This domain concerns the process for developing processes. Such a process development process can be developed, tailored, planned, instantiated, evolved, and enacted. Further, since large organizations would generally need many different kinds of processes and since many of these processes will be of different magnitudes and in different evolutionary stages, the organization would also generally need more than one process development process. This process family, or system of process development processes, would likely include a process architecture, a library of reusable process definitions, and various standards, templates, and forms.

The process architecture would relate all members of this process family. It might have the following sections:
- definitions of the major process elements and their functions
- naming conventions and standards
- standard process templates or formats
- interface specifications
- composition and tailoring rules

The library of reusable process elements would be used as common building blocks to construct multiple process definitions. These reusable elements would be developed consistently with the architecture and would be usable with any process definition that met the same standards. Some examples of such reusable elements are Process Development Planning, Process Requirements Definition, and Process Development Estimation. This last example would be one of the lower level reusable elements referenced by Process Development Planning.

Operating system term	Process concept	Explanation
Process	Process	
Process abstraction	Not explicitly defined	A process step in form of subroutine or process embodying the abstraction
Process refinement	Not explicitly defined	The act of defining the substructure of a process step
Process state	Enactment state	State of execution of a process
Computation	Process trace	
Processor, machine	Agent	Typically refers to machine agent
Exact realization	Precision	
Process interaction	Interaction	Three forms: cooperation, competition, interference
Interpretation	Enactment	
Image	Enactable process	Executable load image of a program
Program	Process definition	
Language description	Not explicitly defined	Notation to express process definition
Executive	Control process giving approval for enactment	Focus on adjustment state
Debugger	Control process performing repair	Focus on monitoring, analysis, and adjustment of enactment with respect to desired behavior of process definition

Figure 4-1 Process Term Sample in the Operating System Domain

Examples of the standards needed by the process development process are Process Review Standards, Process Requirements Standards, and Process Naming Standards. Typical templates would show the formats for process development plans, the contents of a process development es-

38

timate, and the structure and format for a process development architecture. Typical forms would be the Process Development Estimate and the Process Improvement Proposal.

Examples of end-user **process development** scripts are the following:

1. The process for developing a new family of processes would include a process architecture, a reusable element library, and end-user scripts. An example of such a process family would be the complete set of software development and maintenance processes needed by a large software organization.

2. The process for developing a new end-user script that conformed to an existing process architecture would include new scripts and possibly some new or modified reusable process elements. An example of such a process might be the maintenance and support process for a newly developed software product.

3. The process for enhancing an existing process definition to meet some new need would likely include modifications to the existing process script with possibly some new reusable element definitions or modifications. An example of such a process enhancement would be the modification of a software development process to conform to a newly defined customer acceptance testing requirement. In the case of more substantia process modifications, changes or additions to the process architecture might be needed as well.

5. Conclusions

This core set of process definitions and concepts is intended both to facilitate the broader application of these concepts in the software community and to serve as a foundation for further definitional work. Since software process technology is in an early state of development, it is assumed that these core terms will undergo evolutionary change and that many more terms will be added. It is, however, hoped that this core set will be sufficient to facilitate communication on the software process and various process related domains.

The development of these definitions has been a surprisingly rewarding process. Not only has the work of listing and defining these core concepts and terms clarified and sharpened our views, but we have learned a great deal from interacting with many of our associates. We have also found that there is enormous interest in this subject. Software process technology is new and developing rapidly, and is just beginning to be structured and codified. It is an important and useful field for practitioners, and it is a rich and largely unexplored field for research.

As we delve into this subject, it is clear that there is a richness and substance to the technology that is barely discernible on the surface. In principle, we are talking about the design of processes that will permit fallible humans, with the aid of machines, to produce infallible products. To do this economically and to responsively meet society's needs is a challenge of the first order. The challenge of software process research is thus to find economic and effective ways to have many people cooperatively perform complex and precise intellectual tasks. Success will be measured both by the effectiveness with which these processes meet users' needs and by the degree to which they contribute to making software engineering a rewarding and fulfilling profession. As this field evolves, the technology it develops will undoubtedly be of value to many other human activities.

6. Acknowledgments

Table 1 Index of Process Definitions

Term	Section	Term	Section
Accuracy	3.4	Monitoring	3.3
Adjustment	3.3	Planning	3.2
Agent	3.3	Precision	3.4
Analysis	3.3	Process	3
Approval	3.3	Process Architecture	3.1
Authorization	3.3	Process Assurance	3.3
Automation	3.3	Process Authority	3.3
Autonomy	3.4	Process Constraint	3.3
Control Process	3.3	Process Control	3.3
Delegation	3.3	Process Design	3.1
Development	3.2	Process Definition	3.1
Enactable Process	3.1	Process Element	3
Enacting Process	3.3	Process Enactment	3.3
Enactment State	3.3	Process Model	3.1
Enforcement	3.3	Process Plan	3.1
Evolution	3.2	Process Program	3
Fault Tolerance	3.4	Process Script	3
Fidelity	3.4	Process Step	3
Fitness	3.4	Process Trace	3.3
Guidance	3.3	Recovery	3.3
Incident	3.3	Redundancy	3.4
Instantiation	3.2	Repair	3.3
Interaction	3.3	Responsiveness	3.4
Intrusion	3.3	Robustness	3.4
Lifeness	3.4	Scalability	3.4
Maintainability	3.4	Tailoring	3.2

References

[Andrews 91] Andrews, G.R., "Paradigms for Process Interaction in Distributed Systems," *ACM Computing Surveys*, March 1991, pp. 49-90.

[Hornig 73] Horning, J.J. & Randell, B., "Process Structuring," *ACM Computing Surveys*, March 1973, pp. 5-24.

[Humphrey 91] Humphrey, W.S., Snyder, T.R. & Willis, R.R., "Software Process Improvement at Hughes Aircraft," *IEEE Software*, July 1991, pp. 11-23.

[Humphrey 89] Humphrey, W.S., *Software Process Management*, Addison-Wesley, Reading, MA, 1989.

[Kolkhorst 88] Kolkhorst, B.G. & Macina, A.J. "Developing Error-Free Software," *Proceedings of Computer Assurance COMPASS* '88, NIST, IEEE, July 1988.

[Osterweil 87] Osterweil, L.J., "Software Processes are Software Too," *Proceedings of 9th International Conference on Software Engineering (ICSE9)*, IEEE Computer Society Press, April 1987.

[Paulk 91] Paulk, M.C., Curtis, B. & Chrissis, M.B., et al., *Capability Maturity Model for Software*, Software Engineering Institute Technical Report CMU/SEI-91- TR-24, DTIC ADA240603, Carnegie Mellon University, Pittsburgh, PA, August 1991.

[Simon 81] Simon, H.A., "Studying Human Intelligence by Creating Artificial Intelligence," *American Scientist*, vol. 69(3), May-June 1981, pp. 300-309.