

The Many Views of a Process: Toward a Process Architecture Framework for Product Development Processes

Tyson R. Browning*

Neeley School of Business, Texas Christian University, TCU Box 298530, Fort Worth, TX 76129 USA

Received 31 October 2007; Revised 16 April 2008; Accepted 16 April 2008, after one or more revisions
Published online 4 September 2008 in Wiley InterScience (www.interscience.wiley.com)
DOI 10.1002/sys.20109

ABSTRACT

A product or system *development process* is a kind of complex system, arguably even more complex than the system it produces. Yet, the models and tools used by systems engineers and program managers to plan and manage technical work—such as process flowcharts, Gantt charts, work breakdown structures, and text-rendered procedures—are less sophisticated and capable than the ones used to design the product system. When used, the various process models are often challenged to incorporate and maintain synchronized program information—e.g., they may be created by different subgroups in a program and based on different assumptions, and they may diverge as a program proceeds. Recently, *architecture frameworks* (AFs) have been used to help manage the complexity in engineered systems. An AF provides a portfolio of *views* of a complex system, each of which describes it partially and in a format meaningful to its users and their particular needs. This paper proposes the application of AF concepts to the management of the work done to develop a complex system product. The pieces of work and their relationships constitute a complex process. A portfolio of integrated and synchronized views of a single process model would seem to be preferable to the current state—a number of disparate and uncoordinated management models. This paper introduces a new application of AFs to development processes and suggests this area as one for further research and development in the systems engineering community. © 2008 Wiley Periodicals, Inc. Syst Eng 12: 69–90, 2009

* E-mail: t.browning@tcu.edu

Key words: program management; project management; process modeling; architecture framework; views

1. INTRODUCTION

John Godfrey Saxe's famous 19th century poem, "The Blind Men and the Elephant," retells an ancient Asian story in which six blind men touch different parts of an elephant and proceed to argue about what the animal is like [Linton, 1878: 150–152]. The man who touched the tail thinks it is like a rope; the one who touched the tusk thinks it is like a spear; the man who touched a leg thinks it is like a tree; etc. The story has gained popularity in our contemporary age of complex systems, which are impossible to describe and understand completely from a single point of view.

The processes for developing complex systems are like this elephant. They entail thousands of activities, done by hundreds or even thousands of people, each producing results that enable other activities to occur. Because of their size, complexity, and uniqueness, large projects (programs) and their processes are difficult to manage. In attempts to manage complexity, project and program managers and systems engineers commonly use a variety of process models to plan and coordinate work, including process flowcharts, Gantt charts, work breakdown structures (WBSs), and formal procedures [e.g., PMI, 2004; Meredith and Mantel, 2006; Pinto, 2007]. However, these models present several difficulties. For one, they often contain overlapping information, meaning that a project using more than one of these models (as most projects do) is challenged to maintain consistency and synchronization among them. For example, a program might have one (evolving) list of activities in its schedules while its risk management plans contain somewhat different lists. While these lists might begin in tandem, without focused care they might diverge over the course of the program. Second, each model incorporates only a subset of the information about a project, omitting other information. For example, while the schedule would probably indicate the duration of activities, it probably would not note their contributions to reducing specific technical risks, which might be noted in the risk management plan. Third, in a large project or program, each model is often created or maintained by a different person or team, potentially based on different assumptions. Overall, each model is an abstraction of reality that provides a perspective on a project akin to that of a blind man touching an elephant. Hence, some participants may begin to ignore such models as anachronistic once a project gets underway, basing their decisions on hearsay and intuition

instead, while others may continue to use the models for decision support, despite their missing information and lack of integration.

To improve upon this common situation, some have advocated a single (virtual) repository for project information [Basu, Blanning, and Shtub, 1997; Bond, 1999; Presley et al., 2001], so that everyone "draws from the same well." A single source is conjectured to make projects more flexible, since there would be just one place to find information quickly and one place to record changes to plans. Process models provide one possibility for organizing a single source of project information [Browning, Fricke, and Negele, 2006]—perhaps even the best one [Crowston, 2003]. However, models that attempt to contain everything about a project have been cumbersome to build, maintain, understand, and use. Also, it has been noted that managers prefer simple models (that they understand and trust) to more realistic ones [Little, 1970]. Therefore, a new approach would seem to be needed that could simultaneously give managers completeness, integration, and synchronization on one hand and simplicity and focus on the other.

It is instructive to consider how this tension has been addressed in a related context, that of engineering a complex system. Developers of complex products have long faced a similar quandary with product design data. Each designer has their own, discipline-centric modeling tools for purposes of designing and evaluating particular aspects of a complex product (e.g., for an aircraft: aerodynamics, weight, structure, propulsion, etc.). Each such tool includes certain design parameters and conditions while ignoring or making assumptions about others. Hence, in some complex system engineering projects, a recent trend has been to consolidate many of the critical aspects of a design into an integrated set of models guided by an *architecture framework* (AF), wherein the various subsets of information useful for supporting particular design decisions are organized into assorted representations or views. Each such view provides a kind of portal through which the designers from varied disciplines can interact with a relatively simple portion of an otherwise rich and sophisticated model. That is, a view captures a subset of a model's attributes and provides a guideline for their presentation. However, to be useful, these views must align with various users' needs for decision support. Meanwhile, an AF serves to integrate and synchronize the views to

give a more complete and consistent description of the “elephant.”

This paper begins to explore how the AF approach to managing complexity might apply in the different but related context of models of a project’s process. A *project process* (hereinafter, simply “process”) is the set of related activities that accomplish a project (or program). That is, all of the work done on a project is part of its process (whether formalized or not). Since these various work segments depend on each other, processes are often modeled as an activity network. Information about a process may be organized and conveyed to different users (planners, managers, workers, etc.) through different models or views thereof. Better representations and views have been singled out in previous research as a key to improving product development project management [Browning and Ramasesh, 2007; Krishnan and Ulrich, 2001] and decision support systems in general [Basu, Blanning, and Shtub, 1997].

Specifically, this paper proposes the application of an AF approach to encourage the development of a *process AF* (PAF). An approach based on a PAF may simultaneously provide both simpler and more complete models for managers. After presenting a background on AFs and process models and views in the next two sections, the paper briefly reviews process model views and attributes based on literature reviews and case studies. The concluding discussion points towards how a PAF might be developed.

2. ARCHITECTURE FRAMEWORKS

2.1. Background

The structure and behavior of a complex system are impossible to understand fully from a single point of view. Using an example developed by Zachman [1987], even a relatively simple system like a house is not fully defined or determined by its blueprints. Additional views, such as elevations and a bill of materials, are also needed. The work of designing and building a house is split up among architects, contractors, draftsmen, surveyors, framers, electricians, plumbers, roofers, masons, carpenters, etc. There are very few “jacks of all trades,” and it is often impractical to pursue such a skill set. When dealing with much more complex systems, certainly no individual is able to grasp all of the important details, nor would such an individual have the time to do all the work. Therefore, complex system development is parsed into tasks undertaken by different groups who must make harmonious decisions [von Hippel, 1990]. Coordination among such groups becomes more challenging as their number and interdependencies

grow. Attempting to provide all of the detailed information about the system to each of these groups causes information overload and is often worse than not providing the information at all (because of the faulty assumption that communication occurred). Meanwhile, an alternative approach is to provide each group with an appropriate subset of information, in a format that facilitates the accomplishment of their tasks and supports their decision-making. This approach meshes with natural intelligence theory, where Minsky [2006] postulated that the human mind naturally maintains multiple models of a given system (e.g., physical, social, emotional, mnemonic, strategic, visual, and tactile) and rapidly switches between them depending on the current purpose. However, this approach requires identifying which subsets of information best support the purposes of each group.

To determine the information needed by various project participants, an inductive approach involves examining the models they use. Continuing the example of a house, electricians prefer wiring diagrams, with particular symbols representing outlets, switches, etc. and a labeling convention for wiring gauges, etc. Plumbers use pipe routing diagrams with a different set of symbols and labeling conventions. Contractors keep a bill of materials and a list of subcontractors. Collecting all such models yields a superset of information that collectively describes the system. If building a house requires multiple models, how many models are necessary to aid the perhaps thousands of workers involved in a much more complex process? An AF serves to integrate the various models of a complex system into a single, more complete model with multiple views. The views collectively represent and integrate the important attributes of a complex system model, as it is currently understood. In contrast, a deductive approach would entail prescribing certain attributes and views as important to a system, whether these were currently used and understood by the various workers or not. Both the inductive and deductive approaches have advantages, and both could contribute to the development of an AF.

2.2. Architecture Frameworks

The first so-called AF was the Zachman framework [Zachman, 1987; Sowa and Zachman, 1992]. While initially billed as an *information system* AF, it applies more broadly to any kind of complex product. Zachman based his framework on two key ideas:

1. A set of product architectural views is produced over the course of a project; these views represent the different perspectives of the different participants.

Table I. Example Distinctions among Architectural Views (Adapted from Zachman [1987])

View of a Building	Nature/Purpose (for a Building)	View of an Aircraft	View of an Information System	Generic View
Bubble charts	<ul style="list-style-type: none"> Basic concepts for building Gross sizing, shape, spatial relationships Architect/owner mutual understanding Initiate project 	Concepts	Scope/objectives	General view
Architect's drawings	<ul style="list-style-type: none"> Final building as seen by the owner Floor plans, cutaways, pictures Architect/owner agreement on building Establish contract 	Work Breakdown Structure (WBS)	Business model or description	Owner's view
Architect's plans	<ul style="list-style-type: none"> Final building as seen by the designer Translation of owner's view into a product Detailed drawings--16 categories Basis for negotiation with general contractor 	Engineering design, drawings, bill of materials (BOM)	Information system model or description	Designer's view
Contractor's plans	<ul style="list-style-type: none"> Final building as seen by the builder Architect's plans constrained by laws of nature and available technology "How to build it" description Directs construction activities 	Manufacturing engineering design, BOM	Technology-constrained model or description	Builder's view
Shop plans	<ul style="list-style-type: none"> Subcontractor's design of a part/section Detailed stand-alone model Specification of what is to be constructed Pattern 	Assembly and fabrication drawings	Detailed description	Out-of-context view
(n/a)	(n/a)	Numerical code programs	Machine language (or object code)	Machine language view
Building	<ul style="list-style-type: none"> Physical building 	Aircraft	Information system	Product

- The same product can be described in different ways for different purposes, resulting in different views.

That is, the various participants in a project—planners, managers, various designers, builders, subcontractors, etc.—each have a different perspective and require different pieces of information to perform their tasks. Zachman identified six different descriptions of a system—e.g., data (what), functional (how), spatial

(where), personal (who), temporal (when), and purposeful (why). Some of these views are listed in Table I, starting with the views of a building (along with their nature and purpose) and proceeding to the analogous views of an aircraft and an information system. The last column provides generic names for each of these views. Since Zachman was most interested in information systems, he also provided the insightful list in Table II, which demonstrates the different points of view of project participants.

As developers of ultra-complex systems, defense industries became early adopters of Zachman's ideas. In the 1990s, his ideas became manifest in the Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework [DoD, 1997], which looked at a broader swath of system types. As its scope expanded to cover systems of hardware, software, people, and other systems, the C4ISR AF evolved into the current U.S. Department of Defense AF (DoDAF) [DoD, 2007].¹ The DoDAF is used for a variety of system architectures, where the system is a developed product, such as an information system, satellite, aircraft, ship,

Table II. Different Architectural Views from Different Information Systems Functions (Adapted from Zachman [1987])

If you are:	Then you probably think an information systems architecture is:
A programmer	A structure chart
The database administrator	Data design
An analyst	A data flow diagram
A planner	Some combination of entity/relationship diagrams and/or functional flow diagrams
The communications manager	The business logistics infrastructure and/or the distributed systems architecture
An operations manager	The system architecture
A network administrator	The network architecture
A program support representative	Detailed data and program descriptions
A computer designer	Machine language
The president	Entity classes, process classes and/or a map

¹A similar framework, the Ministry of Defence Architectural Framework (MoDAF), has been developed in the UK [MoD, 2005].

or combination thereof (“system of systems”). The DoDAF has grown to include the 29 views shown in Table III. Each of these views presents a subset of the

information describing a complex product—a different blind man’s view of the elephant. The information in each view is not mutually exclusive; there is redun-

Table III. 29 Views of a Product Architecture, According to the DoDAF, Version 1.5 (adapted from DoD [2007])

View Type	Reference	View	General Description
Overall	AV-1	Overview and Summary Information	Scope, purpose, intended users, environment depicted, and analytical findings
	AV-2	Integrated Dictionary	Architecture data repository with definitions of all terms used in all products
Operational	OV-1	High-Level Operational Concept Graphic	High-level graphical/textual description of operational concept (high-level organizations, missions, geographic configurations, connectivity, etc.)
	OV-2	Operational Node Connectivity Description	Operational nodes, connectivity, and information flow
	OV-3	Operational Information Exchange Matrix	Information exchanged between nodes and the relevant attributes of that exchange (such as media, quality, quantity, and the level of interoperability required)
	OV-4	Organizational Relationships Charts	Organizational, role, or other relationships among organizations
	OV-5	Operational Activity Model	Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information
	OV-6a	Operational Rules Model	One of the three products used to describe operational activity (sequence and timing); identifies business rules that constrain operations
	OV-6b	Operational State Transition Description	One of the three products used to describe operational activity (sequence and timing); identifies business process responses to events
	OV-6c	Operational Event-Trace Description	One of three products used to describe operational activity (sequence and timing); traces actions in a scenario or sequence of events
	OV-7	Logical Data Model	Documentation of the system data requirements and structural business process rules of the Operational View
Systems and Services (S&S)	SV-1	S&S Interface Descriptions	Identification of systems nodes, services, and their interconnections
	SV-2	S&S Communications Descriptions	Systems nodes, services, and their related communications protocols
	SV-3	S&S Matrices	Relationships among and between systems and services; can be designed to show relationships of interest, e.g., system-type interfaces, planned vs. existing interfaces, etc.; three matrices: systems-to-systems, services-to-services, and systems-to-services
	SV-4a & b	S&S Functionality Descriptions	Functions performed by (a) systems and (b) services and the information flow among (a) system and (b) service functions
	SV-5a	Operational Activity to Systems Function Traceability Matrix	Mapping of system functions back to operational activities
	SV-5b	Operational Activity to Systems Traceability Matrix	Mapping of systems back to capabilities or operational activities
	SV-5c	Operational Activity to Services Traceability Matrix	Mapping of services back to capabilities or operational activities
	SV-6	S&S Data Exchange Matrices	Provides details of system or service data elements being exchanged between systems or services and the attributes of that exchange
	SV-7	S&S Performance Parameters Matrices	Performance characteristics of S&S View elements for the appropriate time frame(s)
	SV-8	S&S Evolution Descriptions	Planned incremental steps toward migrating a suite of systems or services to a more efficient suite, or toward evolving a current system to a future implementation
	SV-9	S&S Technology Forecasts	Emerging technologies and software/hardware products that are expected to be available in a given set of time frames and that will affect future development of the architecture
	SV-10a	S&S Rules Models	Describes S&S functionality; identifies constraints that are imposed on systems/services functionality due to some aspect of system design or implementation
	SV-10b	S&S State Transition Descriptions	Describes S&S functionality; identifies responses of a system/service to events
	SV-10c	S&S Event-Trace Descriptions	Describes S&S functionality; identifies system/service-specific refinements of critical sequences of events described in the Operational View
	SV-11	Physical Schema	Physical implementation of the Logical Data Model entities, e.g., message formats, file structures, physical schema
Technical Standards	TV-1	Technical Standards Profile	Listing of standards that apply to S&S View elements in a given architecture
	TV-2	Technical Standards Forecast	Description of emerging standards and potential impact on current S&S View elements, within a set of time frames

dancy. Parsimony is a secondary goal; the main goal is to provide views that are useful to various project participants and stakeholders while maintaining their consistency and synchronization. Any given project will probably not use all 29 views; it will select an appropriate subset of views based on its needs. The framework is also extendable, allowing researchers and practitioners to develop additional views to highlight important product characteristics and support additional design decisions. Thus, completeness is not a prerequisite to an AF's usefulness. On the contrary, an AF is likely to evolve to higher levels of usefulness over time.

2.3. Views and Their Organization

A view captures a subset of a system model's attributes and provides a guideline for their presentation.² Each view has its own techniques for describing attributes. For example, the DoDAF's System Technology Forecast view, SV-9, shows the relevant emerging technologies and organizes them according to which part(s) of the system they will influence. By distilling a subset of attributes, a view enables designers to focus on certain relationships. The structure of certain views may also facilitate particular analyses. For example, in the DoDAF's SV-3, the square matrix mapping the interfaces between system components lends itself to matrix analysis techniques, such as clustering product components for modularity [Browning, 2001]. However, when views are created "as needed" by product designers in the absence of an integrating framework, this may challenge the consistency of the information they contain in relation to pre-existing views [Peukert and Walter, 2007]. An AF therefore has the potential to serve as a kind of *periodic table* of views, organizing them in relation to each other based on the information they contain.

2.4. Status

An AF provides a valuable tool for structuring and integrating the various views of a system model. Before a complex product is actually built, its desired architecture can be modeled in terms of a collection of views. As the product's details are specified, the views can be updated to reflect this latest knowledge, potentially even automatically. These benefits have made AFs prominent in systems engineering [e.g., Richards et al., 2007a], and various AFs have been developed and are in use in contexts such as software development [the

"4+1 views" framework—Kruchten, 1995], enterprise information systems [TOG, 2006; Tang, Han, and Chen, 2004; Noran, 2003], enterprise architecting [Iyer and Gottlieb, 2004], and space system design [Richards et al., 2007b]. The emergence and popularity of AFs in a variety of contexts attests to their utility.

3. THE PROJECT PROCESS AND ITS VIEWS

So far, AFs have focused on two types of systems, products (especially information-technology-intensive systems) and enterprises. However, an AF approach may also benefit program managers and systems engineers who must deal with another kind of complex system, the process whereby a large, complex project (or program) is accomplished. A *process* is "an organized group of related activities that work together to create a result of value" [Hammer, 2001]. A system is "an integrated set of elements that accomplish a defined objective" [INCOSE, 2007]. A process is a kind of system, where the elements are typically *activities* (work to be done, decisions to be made, etc.) and the integrating relationships are the activities' *interdependencies* [Browning, Fricke, and Negele, 2006; Crowston, 2003]. Elements and relationships give rise to the emergent behaviors in complex systems [e.g., Axelrod and Cohen, 1999; Holland, 1998]. In attempts to better understand processes, researchers have developed numerous system-oriented models that treat the process as a network of interrelated activities [Browning and Ramasesh, 2007].

While the complexity of a system is challenging to measure, the *NK* model [Kauffman and Levin, 1987] has been widely used as a simple approach, where *N* is the number of system elements and *K* is the number of relationships between them. From this perspective, a process system can be at least as complex as the output it seeks to produce (e.g., a design for a product system), for at least three reasons: (1) for each detailed specification of a component or desired function in a product, at least one action or decision is required in the process (a one-to-many relationship); (2) anticipating and ameliorating the undesired, emergent behaviors of a complex product requires many additional activities (such as simulation and testing) in the process (another one-to-many relationship); and (3) this greater number of process activities than product components (due to both kinds of one-to-many relationships) is typically accompanied by a greater number of inter-activity connections. Despite this great complexity in processes, however, the decision support tools (e.g., models and simulations) aiding project managers in understanding, planning, and controlling such processes are often less

²IEEE further describes a *viewpoint* as "a specification of the conventions for constructing and using a view" [IEEE, 2000]. This reference also allows for a many-to-one mapping of models to views.

capable and mature than those used to design the outputs themselves. Project managers and participants are often left to rely on greatly simplified and disparate models, or even “mental models” [Senge, 1990], as they attempt to describe and control the “elephant.”

A model is an abstract representation of reality that is built, verified, analyzed, and manipulated to increase understanding. “All models are wrong, but some are useful” [Box, 1979]. A useful model is simple, robust, easy to control, adaptive, complete, and easy to communicate with [Little, 1970]. A process model includes the *attributes* of and the underlying *assumptions* about a process which are deemed sufficient to describe it for a particular purpose. If it aligns with the way its users think about a problem, even a model with subjective inputs can be extremely beneficial [Little, 1970]. That is, what determines a “good” or useful process model depends on the user and the decision to be supported, and thus a model fit for one use may not be appropriate for another [Browning, Fricke, and Negele, 2006; Crowston, 2003]. For example, a generic process model, for general use on all of a firm’s projects, will probably not contain sufficient details for managing each unique project. Including all such details would be inappropriate for the general model, even though details are needed to manage each project. Thus, *the fitness of a process model depends on the alignment of its content and structure with what is appropriate to support a particular decision, purpose, or use case.*

In contrast to a model, a *view* is a representation (an arrangement of symbols, a table, or other depiction) chosen to display a selected subset of a model’s attributes and assumptions [Browning and Ramasesh, 2007]. It is important to distinguish a model from a view. A process model, for example, contains information about a process, whereas a view presents some or all of that information in a chart, diagram, table, or other depiction. While many traditional models and views have a one-to-one correspondence, AFs may stipulate a one-to-many relationship between a model and its views, as the attributes captured in a rich model are accessed through multiple views. Views leverage the principles of information hiding [e.g., Parnas, 1972] to reduce complexity for decision makers.³ By reducing complexity and focusing on the specific needs of different constituencies, views can be a significant driver of innovation in system design [Alexander, 1964; Simon, 1981; Zachman, 1987; Schätz et al., 2002; Keller et al.,

2006] and product development decisions [Krishnan and Ulrich, 2001].

A Gantt chart (e.g., Fig. 1), which depicts activities and their temporal relationships, provides an example of a view of a process model. A basic Gantt chart shows the activity attributes of duration, start time, and finish time. Gantt charts may be augmented with additional information, such as the organizational unit responsible for an activity, activity resource requirements, precedence relationships (dependencies), activity percent complete, activity parent (or “roll up”) activity, etc. However, including too much information crowds the view, so all of the information in a richer model of the process is deliberately not included.

It is important to emphasize that, at a minimum, a process consists of both activities (work packages) and deliverables (work products). The deliverables flow in the input-output relationships between the activities. (A deliverable is a very general object representing any activity relationship, such as a transfer of information, data, knowledge, documents, estimates, prototypes, materials, etc.—even seemingly abstract items like the results of decisions.) Just as a system consists of both elements and relationships, a process, as a kind of system, consists of both activities and deliverables, although the deliverables tend to be deemphasized in many of the common views. (For example, Fig. 1 does not show the deliverables.) Also, as with any system, any process is part of (i.e., can be thought of as an activity within) a larger process, and each activity in a process may itself be viewed as a process (and further decomposed into lower-level activities). Thus, the terms “process” and “activity” are observer-dependent and often interchangeable. This paper will refer to the component work packages in a process as activities, only for the purpose of having distinct terms for the “parent” and “child” work packages. (Deliverables can also be decomposed and organized hierarchically, but this paper will not use different terms for parent and child deliverables.)

In terms of a physical product, architecture has been defined as “the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution” [IEEE, 2000]. Thus, the activities in a process and their relationships help determine the architecture of that process [Browning and Eppinger, 2002]. *Process architecture* refers to the structure of activities, their relationships, and the principles and guidelines governing their design and evolution.

In summary, then, a process consists of related activities. Most process models are therefore activity-based and (more or less) account for the activities’

³However, unlike in applications of information hiding to product modularity [e.g., Baldwin and Clark, 2000], a parsimony of views is not a primary goal. While a product component must physically reside in either one module or another, it is common and desirable for the information attributes in one view to co-exist in other views, as long as they are synchronized.

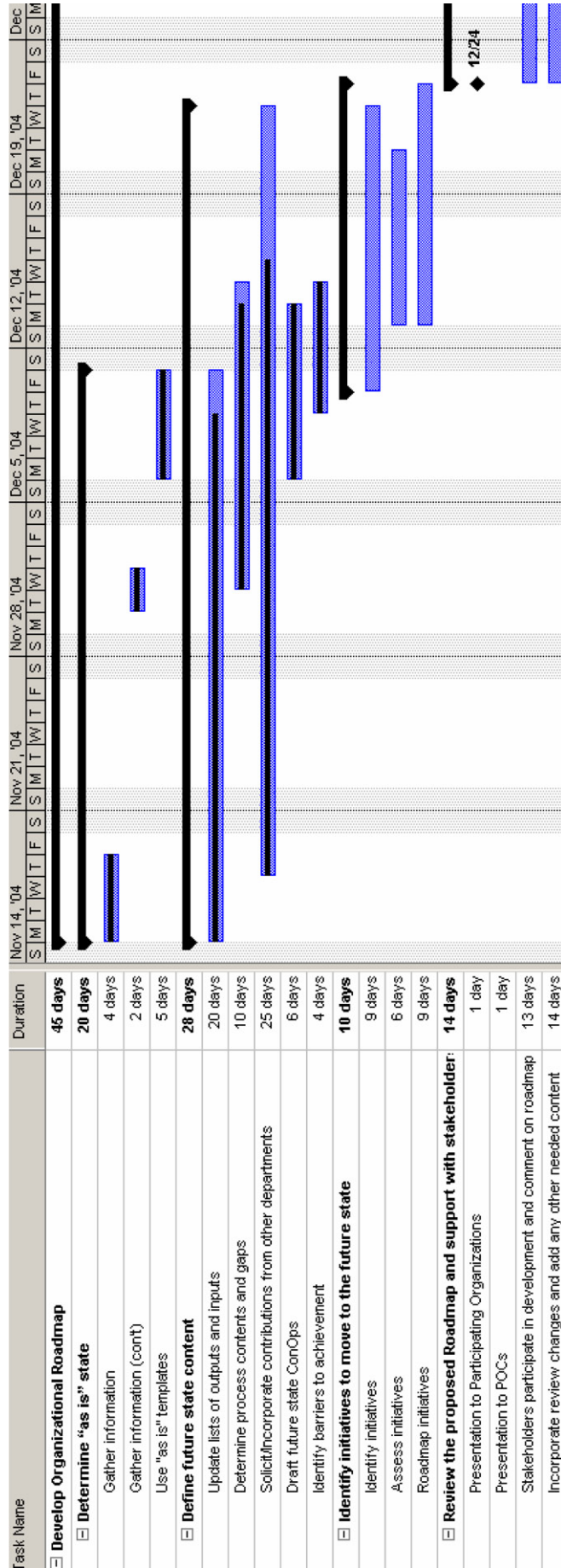


Figure 1. An example of a basic Gantt chart showing activity start, end, duration, and parent-child attributes [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

interrelationships. In contemporary practice, all of the information useful for planning and controlling the large numbers of varied activities and relationships in a complex process has not been captured in a single model. Instead, various models may capture certain subsets of the information, as guided by each model's purpose (e.g., a particular decision to support, etc.). AFs provide a potential avenue for synthesizing the information in multiple models into a single, rich, integrated, synchronized, and more complete model while maintaining the distinctiveness, simplicity, and usefulness of each view. Whereas Little [1970] noted a conflict between simplicity and completeness in managerial models, an AF offers the potential to achieve both simultaneously.

Finally, it is important not to confuse an AF approach applied to processes with prior approaches such as traditional process repositories or an organizational process handbook [Malone, Crowston, and Herman, 2003; Malone et al., 1999], even though all of these seek to organize process information. However, these approaches do so at different levels and to different degrees. Process handbooks and repositories seek to organize the various types of business processes across an enterprise, whereas the AF approach seeks to organize various subsets of the information (attributes and assumptions) pertaining to an individual project process into a catalog of multiple, useful views. The two approaches address different issues and are complementary.

4. LITERATURE REVIEW AND DATA ANALYSIS

4.1. Common Views of Process Information

Based on a literature review [Browning, 2007], Table IV presents many of the common ways to view a process as an activity network. Table V lists these views and a few others, along with the attributes they include (in their basic form) and example references (to which the reader is directed for further information on each view). These lists are representative rather than comprehensive and, in some cases, group closely related views into a single category.⁴ This review reveals that certain views display particular attributes of the activities

⁴In contrast to these views, researchers have also provided ways to view alternative process architectures or modes. For example, Malone et al. [1999; Malone, Crowston, and Herman, 2003] organize processes at various levels of abstraction to facilitate navigation of a process repository. Pentland [1995] provides a grammatical characterization of processes. Chung, Kwon, and Pentland [2002] emphasize the importance of visualizing a project's potential *process space*—the range of process scenarios that could unfold. However, Tables IV and V focus specifically on views of a single process as an activity network.

and/or deliverables in a process. Usually it would be possible to augment this basic set of attributes with additional attributes (such as choosing to show resource assignments in a Gantt chart), although the lists in Table V seek to present only the basic sets of attributes rather than all possible extensions.

4.2. Process Model Attributes

As discussed in Section 3, when conceived as a system, a process typically consists of two fundamental objects, activities (work packages) and their relationships, which can be defined in terms of input and output deliverables (work products). Activities and deliverables each have properties or attributes, some of which are listed in Table V. Based on the results from literature reviews [Browning, 2007; Browning, Fricke, and Negele, 2006] and data from several case studies,⁵ Table VI lists a collected superset of process attributes (which are defined in Tables VIII and IX in the Appendix). This list of attributes is not comprehensive; others could be added. However, this list more than spans the attributes of the views in Table V.⁶

4.3. The Attributes Shown in Particular Views

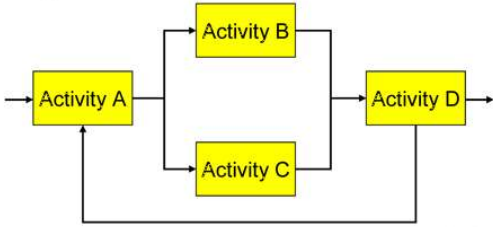
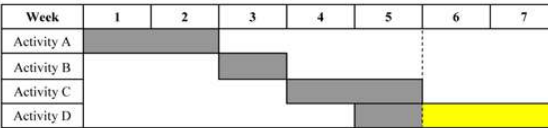
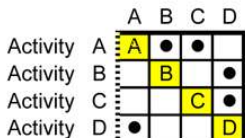
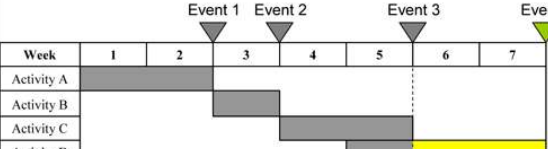
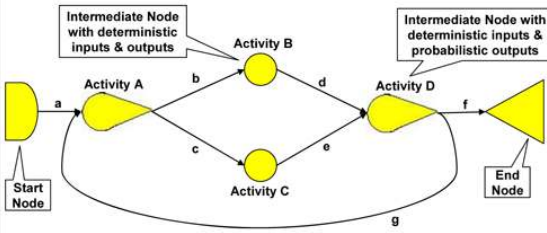
To provide further specifics regarding the process information contained in each view, Table VII maps the views (Table V) to the process attributes (Table VI) they include.⁷ Table VII uses a four-level relationship scale [0, 1, 1.5, 2], depending on whether the attribute is never, potentially, sometimes/partially, or always shown in a particular view. The assignment of these ratings is approximate, since many variations of par-

⁵Several case studies were conducted at a U.S. Fortune 100 company that develops complex, high-tech system products. The company is generally organized into functional organizations (e.g., engineering, manufacturing, program management office, marketing, etc.) and large projects (programs), each developing a particular complex system product and organized into a number of cross-functional teams. Primary data collection occurred in July–September 2006. To gain a diversity of perspectives within the company, the author interviewed 12 people from varied program and functional organizations. Through additional meetings and conversations, other individuals from assorted organizations in the company and with varied backgrounds also provided inputs.

⁶Note that a process model could contain additional objects besides activities and deliverables—such as organizational units, project states, and project events, each with their own attributes—although at some point such a model would cease to be merely a process model and become characteristic of an enterprise model (which represents a natural path for further research in this area but also adds many layers of complexity).

⁷Three views from Table V—High-Level “Life Cycle” Models, Activity-on-Arc Diagrams, and Stock-and-Flow Diagrams—were not carried forward in the analysis because they do not include many of the attributes. These three views are therefore excluded in Table VII.

Table IV. Brief Descriptions and Examples of Some Common Process Model Views (Adapted from Browning [2007])

<p>Flowchart (Network Diagram)</p> <p>The classical process representation; activities in boxes and relationships on arrows (i.e., activity-on-node [AON] or PERT^a diagram); sometimes shows branching nodes using diamonds; often augmented according to local preferences and conventions</p> 	<p>Gantt Chart</p> <p>The classical project management representation; depicts activities and their temporal relationships; may also indicate precedence relationships and activity status; sometimes augmented with additional activity attributes</p> 
<p>Design Structure Matrix (DSM)</p> <p>Square matrix of N activities on the diagonal, where marks in off-diagonal cells indicate an input/output relationship between activities; in the convention shown, feedback is shown below the diagonal</p> 	<p>Milestone Chart</p> <p>Similar to a Gantt chart with the addition of symbols representing major events (above or within the chart)</p> 
<p>GERT^b Diagram</p> <p>Extension to PERT that allows probabilistic branching between activities (nodes); arcs are lettered and have associated probabilities</p> 	<p>Textual Narrative</p> <p>Process documentation that explains in words what is to be done and how</p> <ul style="list-style-type: none"> • Process Name: Do Stuff • Process Owner: John Smith • Narrative: After collecting inputs from the preceding process, Activity A does x, y, and z. In so doing, it typically requires one designer and one mock-up person and takes one week. This activity must be done in accordance with the MST-3K standard and follow the design requirements provided. Additional work instructions are available at URL. Once complete, the results of Activity A are verified by a peer inspection. If no problems are found, then Activities B and C can each begin to work in parallel to do B-thing and C-thing. Activity B usually takes about a week, while Activity C usually takes two weeks. Each requires two designers. When both B and C are ready, their results enable the start of Activity D, which takes another week to synthesize B and C into D. Activity D requires three test engineers. If problems are found, then the process must return to Activity A to make adjustments, in which case Activities B and C will probably also have to be reworked, at least partially.

ticular views exist, some of which include additional attributes. In assigning a value to each cell, care was taken to assume a standard, basic version of each view. The “1” values were especially difficult to assign, because just how easy it would be to include an attribute (or not) depends on assumptions about a view’s practical capabilities and limitations. While the values in some individual cells could be argued, Table VII nevertheless provides a helpful overview and enables several observations.

First, the textual narrative has the capability to include almost any desired attribute, although it is up to its authors to do so. However, the textual process documentation at the case study company was inconsistently detailed. Moreover, when many attributes are included,

it becomes difficult to organize the narrative in a way that facilitates users finding a particular piece of information. Filtering subsets of information is difficult. A good search engine can get users to information quickly, but only if they know exactly what to look for. Therefore, simply determining whether or not a view can represent an attribute does not tell the whole story (but it is a start).

Second, most views emphasize the activities but not the deliverables [Browning, Fricke, and Negele, 2006; Browning and Ramasesh, 2007]. At best, some views name the deliverables without elaborating on them, while many views only treat the deliverables implicitly. The eEPC diagram provides the capability to emphasize deliverables, but, when it does so, the diagram becomes

Table IV (continued)

IDEF0^c Diagram

Emphasizes the input-output deliverables flowing among activities; activity boxes arranged diagonally on a single page; data inputs enter on the left of each box, control inputs enter from the top, mechanism inputs enter from the bottom; data outputs exit on the right of each box, while call outputs exit from the bottom; activity and deliverable hierarchies are also apparent

IDEF3 Diagram

Similar to flowchart, but with emphasis on flow junctions (And, Or, Xor; synchronous or asynchronous); activity identification numbers also shown

State Diagram

Most state diagrams merely show the possible states (nodes), connected by transition paths ("edges," using the terminology of directed graphs); in process modeling, they may also show the intervening activities as a different type of node

Activity-on-Arc Diagram

Circles represent events, such as start or end of a given activity; arrows (arcs) represent activities and are proportional in length to activity duration, which is given in parentheses after the activity name; dotted lines (dummy activities) connect dependent events where not implied by actual activities

CRUD^d Table

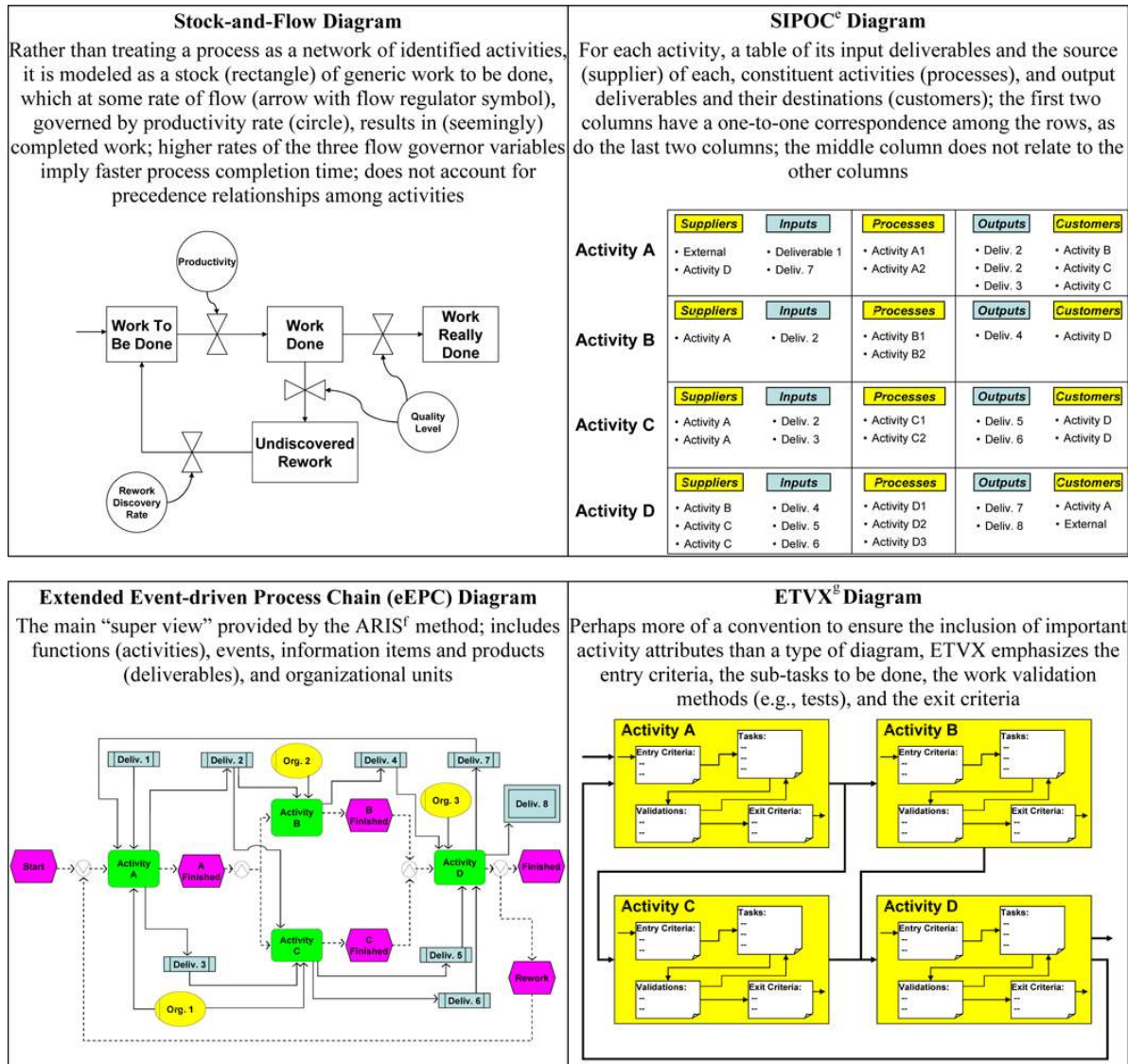
Shows activities' effects on deliverables; an activity can create, read only (use), update (modify), and/or delete a deliverable

	Deliverable 1	Deliverable 2	Deliverable 3	Deliverable 4	Deliverable 5	Deliverable 6	Deliverable 7	Deliverable 8
Activity A	R	C	C				U	
Activity B		R		C				
Activity C		R	R		C	C		
Activity D				R	U	R	C	C

Value Stream Map

Emphasizes activity cycle times (CT) and in-process times (IPT) and inter-activity wait times (WT); review activities shown as ovals instead of rectangles; intervening inventories shown as triangles; additional symbols also common

Table IV (continued)

^a Project Evaluation and Review Technique^b Graphical Evaluation and Review Technique^c IDEF0 stands for “Integrated Definition, Version 0.” There are also versions up to 14 for various niche applications, including IDEF3 for “process description capture.”^d Create-Read-Update-Delete^e Supplier-Input-Process-Output-Customer.^f The acronym is based on the German term for “Architecture of Integrated Information Systems.”^g Entry-Task-Validation-Exit.

cluttered. Therefore, additional, separate views might be needed to emphasize the characteristics of deliverables.

Third, the “WPS database record” view provides a means of potentially accessing a wide variety of information about the attributes of deliverables. This type of view—direct access to database objects (records) and

their attributes (fields)—seemed attractive to expert users such as process owners at the case study company. However, effective use of this type of view requires a higher level of prior knowledge about an object and its context.

Fourth, a number of attributes are not shown by any view, although some views could be expanded to show

Table V. References for and Attributes of Some Common Views of Process Models (Adapted from Browning [2007])

View	Example References	Primary Attributes			Comments
		Process	Activity	Deliverable	
Process flowchart – Network diagram – PERT chart – Activity-on-node (AON) diagram	[IBM 1969] [Moder <i>et al.</i> 1983] [Elmaghraby 1995]	Name, Children	Name, Suppliers, Customers		Often enhanced to include additional attributes, such as Responsible Organization (via “swim lanes”), etc.
Gantt chart – Milestone chart	[Gantt 1919]	Name, Children, Milestone Events	Name, Duration, Start Time, Finish Time		Often enhanced to include additional attributes, such as Percent Complete, Process/Activity hierarchy (Parent-Children), etc.
Design Structure Matrix (DSM)	[Browning 2001]	Name, Children	Name, Suppliers, Customers, Activity Sequence Vector	Supplier, Customer(s)	Often enhanced to include additional attributes such as: Duration, Responsible Organization, Rework Probability, Rework Impact, Process/Activity hierarchy (Parent-Children), etc.
GERT diagram	[Pritsker and Happ 1966]	Name, Children	Name, Suppliers, Customers	Flow Probability	
Textual narrative	[SPC 1996, pp. 50f; Olson 2006]	Name, Narrative Description			Various activity and deliverable attributes may be embedded depending on the guidelines followed
High-level “life cycle” models – Stage-gate/waterfall model – Spiral model – “Vec” model	[Unger and Eppinger 2002] [NASA 1995; Cooper 1994] [Boehm 2000] [Mooz and Forsberg 2006]	Name, Children, Planned Iterations, “Toll Gates”	Name, Suppliers, Customers Mode		Typically used only at a high level, to show the major phases of a large project
IDEF0 diagram	[NIST 1993; Feldmann 1998]	Name, Children	Name, Suppliers, Customers, Inputs, Outputs, Identification Number	Name, Supplier, Customer(s), Parent, Children	Distinguishes three types of inputs—data, controls, and mechanisms—and two types of outputs—data and calls
IDEF3 diagram	[Mayer <i>et al.</i> 1995]	Name, Children	Name, Suppliers, Customers	Flow Conditions	Emphasizes the synchronous or asynchronous AND, OR, or XOR flows among activities
State diagram – Event graph – Markov chain – Data flow diagram – Directed graph	[Harel 1987]	Name, Children, States ^a	Name, Suppliers, Customers	Name, Transition Conditions, Transition Probability	Activities are states; used by Petri Net and Unified Modeling Language (UML) models; project process applications require possibility of being in more than one state at a time
Activity-on-arc (AOA) diagram	[Elmaghraby 1995]	Name, Children, States	Name, Suppliers, Customers, Duration		Activities are the transitions between states; includes dummy arcs and events
CRUD Table			Name, Type of Use (for each deliverable)	Name	Often used to model database and information system architectures
Value Stream Map	[McManus 2005]	Name, Children, Cycle Time	Name, Suppliers, Customers, Cycle Time, In-Process Time	Wait Time	Recently adapted for modeling project processes; can also show additional features of process; emphasizes identifying sources of waste in processes
Stock-and-flow diagram	[Stermann 2000; Ford and Sterman 2003]	Stocks and flows of work; metrics			Does not identify discrete activities, deliverables, or precedence relationships; models completion of generic work

Table V (continued)

View	Example References	Primary Attributes			Comments
		Process	Activity	Deliverable	
SIPOC diagram – IPO diagram	[Browning <i>et al.</i> 2006]	Name, Children	Name, Suppliers, Inputs, Outputs, Customers, Children	Supplier, Customer(s)	More like a table than a diagram; IPO omits the Suppliers and Customers
Extended Event-driven Process Chain (eEPC)	[Scheer 1999; 1998]	Name, Children, Events, Organizational Units, Deliverables	Name, Inputs, Outputs, Suppliers, Customers, Responsible Organizational Unit	Name, Supplier, Customer(s), Flow Conditions, Parents, Children	Most comprehensive view out of the ones listed in this table, but can become unwieldy when all objects are included at once; can be used to filter out subsets of objects and show only those
ETVX	[Radice <i>et al.</i> 1985]		Entry Criteria, Children, Validations, Exit Criteria		
Responsibility Assignment Matrix (RAM)	[PMI 2004]	Name	Name, Roles		A table mapping activities to organizational units who fill roles on or have a responsibility for each activity
Work Product Standard (WPS) database record	Practitioner-developed at case study company			Many (see Table 7)	Shows a database record (or a report thereof on a spreadsheet) of many deliverable attributes

^aCould also be events or conditions.

them. However, doing so causes the views to become cluttered. Additional views might be needed to represent these attributes in a useful way.

5. TOWARDS A PROCESS ARCHITECTURE FRAMEWORK

This paper proposes the development of a *process architecture framework* (PAF) and presents some preliminary explorations and reviews that illuminate a path in this direction. Using a catalog of views, a PAF provides a seemingly simple while more complete, integrated,

and synchronized model of a complex process. The important decisions facing managers and systems engineers in large, complex projects require appropriate support from model views that filter, organize, and synchronize the relevant information. In contemporary projects, many of the views are based on disparate models, which omit (or make assumptions about) some information and include extraneous information [Browning, 2007]. A set of appropriate views would include only the right information (for making a certain decision, or conducting a certain analysis or evaluation) and exclude (or hide) other information—yet, this

Table VI. Attributes of Two Fundamental Objects in a Process Model (Adapted from Browning [2007])

Process/Activity (Work Package) Object Attributes		Deliverable (Work Product) Object Attributes	
<ul style="list-style-type: none"> Name Parent Constituents (“Children”) Mode Shadowing Deployment Version Number Brief Description Inputs Outputs Entry Criteria 	<ul style="list-style-type: none"> Standard Roles Deployed Roles Basis for Requirement Rules References Standard Risks Deployed Risks Narrative Description Tailoring Guidance System Identification Number WBS Element Association 	<ul style="list-style-type: none"> Name Parent Constituents (“Children”) Mode Shadowing Deployment Version Number Brief Description Suppliers Customers Key Criteria and Measures of Effectiveness Requirements Acceptance Criteria 	<ul style="list-style-type: none"> Standard Process Metrics Deployed Process Metrics Format Medium Artifact Rules References Narrative Description Tailoring Guidance System Identification Number WBS Element Association Change History Change Notifications
<ul style="list-style-type: none"> Exit Criteria Verifications Standard Process Metrics Deployed Process Metrics Tools 	<ul style="list-style-type: none"> Master Owner Standard Owner Deployed Owner Change History Change Notifications 		

would occur *within the context of the hidden information* rather than apart from it. That is, a PAF may serve as the organizing structure for a database of project and process information. Instead of the situation where each group participating in a project supports its preferred models and views by keeping its own, disparate databases, a PAF can provide a structure for collecting the various bits of system information into a central repository, thus ensuring that each group works with common information. Changes to the system could be immediately represented in all of its views. In the house-building example, moving a wall on the blueprint would immediately show up as a needed change on the wiring and plumbing diagrams. Hence, a PAF could also lay a foundation for artificial intelligence to detect project planning (scheduling, budgeting, technical risk, etc.) problems automatically. By having a framework to keep track of project changes and their implications, managers would have a better tool for avoiding nonharmonious decisions. Changes to the hidden information would have the potential to automatically trigger questions about the revealed information (because of the behind-the-scenes integration) rather than later in the process at important integration events.

An appropriate PAF would need to account for the attributes listed in Table VI and perhaps others as well. Various subsets of these attributes would be available to users through various views. The list of views in Table V is a start, but Table VII indicates a lack of attention to certain attributes by this set of views. It seems that additional views would need to be developed, perhaps by targeting specific users and their use cases. It is also important to mention that a “good” view does more than just include and exclude the right attributes; it also arranges those attributes in an elegant and intuitive way to accelerate good decision making. For example, the design structure matrix (DSM) view (shown in Part 1 of Table IV), highlights potential iteration and rework loops in a process, thereby focusing attention on the key drivers of cost and schedule risk in a project. If a particular aspect of a project is known or predicted to be a challenge, specific views could be customized to plan, monitor, and control that area. Future research could propose and verify new views as well as continue to gather the “home-grown” views currently used (because they are helpful) in various projects, companies, and industries.

New, customized views can be added to a PAF. In fact, a PAF may develop best if allowed to emerge rather than attempting to specify it completely *a priori*. However, such emergence would need to be guided by an appropriate set of simple rules and standards for storing and manipulating process attributes. The set of attributes listed in Table VI seems to provide a rich and

flexible platform for such development, although these lists would probably need to be expanded in different contexts.

While a catalog of views may ultimately populate a PAF, not all of the views will be equally useful for every project. Research will be needed not only to develop and verify new views, but to ascertain the contexts in which they are likely to be valuable. While experienced systems engineers and managers will have a firmer grasp of what they would like to know in order to plan and control a project, less experienced ones will require guidance in selecting the views most deserving of their attention on a particular kind of project.

ACKNOWLEDGMENTS

Nitin Joglekar, James Martin, four anonymous reviewers, and several individuals at company where the case studies occurred provided helpful comments on earlier versions of the paper. Marc Ortiz provided careful research assistance.

REFERENCES

- C. Alexander, Notes on the synthesis of form, Harvard University Press, Cambridge, MA, 1964.
- R. Axelrod and M. Cohen, Harnessing complexity, The Free Press/Simon & Schuster, New York, 1999.
- C.Y. Baldwin and K.B. Clark, Design rules: The power of modularity, Vol. 1, MIT Press, Cambridge, MA, 2000.
- A. Basu, R.W. Blanning and A. Shtub, Metagraphs in hierarchical modeling, *Management Sci* 43(5) (1997), 623–639.
- B. Boehm, Spiral development: Experience, principles, and refinements, CMU’s Software Engineering Institute, Pittsburgh, PA, 2000.
- T.C. Bond, Systems analysis and business process mapping: A symbiosis, *Bus Process Management J* 5(2) (1999), 164–177.
- G.E.P. Box, “Robustness in scientific model building,” *Robustness in statistics*, R.L. Launer and G.N. Wilkinson (Editors), Academic, New York, 1979, pp. 201–236.
- T.R. Browning, Applying the design structure matrix to system decomposition and integration problems: A review and new directions, *IEEE Trans Eng Management* 48(3) (2001), 292–306.
- T.R. Browning, Aligning the purposes and views of project activity network models, TCU Neeley School of Business Working Paper, Fort Worth, TX, 2007.
- T.R. Browning and S.D. Eppinger, Modeling impacts of process architecture on cost and schedule risk in product development, *IEEE Trans Eng Management* 49(4) (2002), 428–442.
- T.R. Browning and R.V. Ramasesh, A survey of activity network-based process models for managing product development projects, *Prod Oper Management* 16(2) (2007), 217–240.

- T.R. Browning, E. Fricke, and H. Negele, Key concepts in modeling product development processes, *Syst Eng* 9(2) (2006), 104–128.
- M.J. Chung, P. Kwon, and B.T. Pentland, Making process visible: A grammatical approach to managing design processes, *J Mech Des* 124(3) (2002), 364–374.
- R.G. Cooper, Perspective: Third-generation new product processes, *J Prod Innovation Management* 11(1) (1994), 3–14.
- K. Crowston, “Process as theory in information systems research,” *Organizing business knowledge*, T.W. Malone, K. Crowston and G.A. Herman (Editors), MIT Press, Cambridge, MA, 2003, pp. 177–190.
- DoD, C4ISR Architecture Framework, Version 2.0, U.S. Department of Defense, C4ISR Architecture Working Group, Washington, DC, 1997.
- DoD, DoD Architecture Framework, Version 1.5, U.S. Department of Defense, DoD Architecture Framework Working Group, Washington, DC, 2007.
- S.E. Elmaghraby, Activity nets: A guided tour through some recent developments, *Eur J Oper Res* 82(3) (1995), 383–408.
- C.G. Feldmann, *The practical guide to business process reengineering using IDEF0*, Dorset House, New York, 1998.
- D.N. Ford and J.D. Sterman, Overcoming the 90% syndrome: Iteration management in concurrent development projects, *Concurrent Eng Res Appl* 11(3) (2003), 177–186.
- H.L. Gantt, *Organizing for work*, Harcourt, Brace and Howe, New York, 1919.
- M. Hammer, Seven insights about processes, *Proc Conf Strategic Power of Process: From Ensuring Survival to Creating Competitive Advantage*, Boston, MA, 2001.
- D. Harel, Statecharts: A visual formalism for complex systems, *Sci Comput Program* 8(1) (1987), 231–274.
- J.H. Holland, *Emergence, Helix* (Addison-Wesley), Reading, MA, 1998.
- IBM, Flowcharting techniques, IBM Data Processing Techniques, International Business Machines, Yorktown Heights, NY, 1969.
- IEEE, IEEE recommended practice for architectural description of software-intensive systems, Institute of Electrical and Electronics Engineers Standards Association, 2000.
- INCOSE, *Systems engineering handbook: A guide for system life cycle processes and activities*, International Council on Systems Engineering, Seattle, WA, 2007.
- B. Iyer and R.M. Gottlieb, The four-domain architecture: An approach to support enterprise architecture design, *IBM Syst J* 43(3) (2004), 587–597.
- S.A. Kauffman and S. Levin, Towards a general theory of adaptive walks on rugged landscapes, *J Theoret Biol* 128(1) (1987), 11–45.
- R. Keller, T.L. Flanagan, C.M. Eckert, and P.J. Clarkson, Two sides of the story: Visualising products and processes in engineering design, *Proc 10th Int Conf Information Visualisation*, London, UK, 2006, pp. 362–367.
- V. Krishnan and K.T. Ulrich, Product development decisions: A review of the literature, *Management Sci* 47(1) (2001), 1–21.
- P. Kruchten, Architectural blueprints—the “4+1” view model of software architecture, *IEEE Software* 12(6) (1995), 42–50.
- W.J. Linton, *Poetry of America: Selections from one hundred American poets from 1776 to 1876*, Clowes, London, 1878.
- J.D.C. Little, Models and managers: The concept of a decision calculus, *Management Sci* 16(8) (1970), B466–B485.
- T.W. Malone, K. Crowston, and G.A. Herman (Editors), *Organizing business knowledge*, MIT Press, Cambridge, MA, 2003.
- T.W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C.S. Osborn, A. Bernstein, G. Herman, M. Klein, and E. O’Donnell, Tools for inventing organizations: Toward a handbook of organizational processes, *Management Sci* 45(3) (1999), 425–443.
- R.J. Mayer, C.P. Menzel, M.K. Painter, P.S. deWitte, T. Blinn, and B. Perakath, *Information Integration for Concurrent Engineering (IICE) IDEF3 process description capture method report*, Knowledge Based Systems, College Station, TX, 1995.
- H.L. McManus, *Product Development Value Stream Mapping (PDVSM) manual*, MIT Lean Aerospace Initiative, Cambridge, MA, 2005.
- J.R. Meredith and S.J. Mantel, *Project management*, 6th edition, Wiley, New York, 2006.
- M. Minsky, *The emotion machine*, Simon & Schuster, New York, 2006.
- MoD, Ministry of Defence architectural framework overview, Version 1.0, UK Ministry of Defence, London, 2005.
- J.J. Moder, C.R. Phillips, and E.W. Davis, *Project management with CPM, PERT and precedence diagramming*, Van Nostrand Reinhold, New York, 1983.
- H. Mooz and K. Forsberg, The dual vee—illuminating the management of complexity, *Proc 16th Annu Int Symp INCOSE*, Orlando, FL, 2006.
- NASA, *NASA systems engineering handbook*, NASA Headquarters, Code FT, SP-6105, Houston, TX, 1995.
- NIST, *Integration Definition for Function Modeling (IDEF0)*, National Technical Information Service, U.S. Department of Commerce, Springfield, VA, 1993.
- O. Noran, “A mapping of individual architecture frameworks (GRAI, PERA, C4ISR, CIMOSA, Zachman, ARIS) onto GERAM,” *Handbook on enterprise architecture*, P. Bernus, L. Nemes, and G. Schmidt (Editors), Springer, Berlin, 2003, pp. 65–210.
- T.G. Olson, Defining short and usable processes, *CrossTalk J Defense Software Eng* 19(6) (2006), 24–28.
- D.L. Parnas, On the criteria to be used in decomposing systems into modules, *Commun ACM* 15(12) (1972), 1053–1058.
- B.T. Pentland, Grammatical models of organizational processes, *Org Sci* 6(5) (1995), 541–556.
- A. Peukert and U. Walter, Integrating system views emerging from different engineering cultures, *INSIGHT* 10(1) (2007), 23–28.
- J.K. Pinto, *Project management*, Pearson Prentice Hall, Upper Saddle River, NJ, 2007.

- PMI, A guide to the project management body of knowledge, Project Management Institute, Newtown Square, PA, 2004.
- A. Presley, J. Sarkis, W. Barnett, and D. Liles, Engineering the virtual enterprise: An architecture-driven modeling approach, *Int J Flexible Manufacturing Syst* 13(2) (2001), 145–162.
- A.A.B. Pritsker and W.W. Happ, GERT: Graphical Evaluation and Review Technique: Part I. Fundamentals, *J Indust Eng* 17(5) (1966), 267–274.
- R.A. Radice, N.K. Roth, J. A.C. O'Hara, and W.A. Ciarfella, A programming process architecture, *IBM Syst J* 24(2) (1985), 79–90.
- M.G. Richards, N.B. Shah, D.E. Hastings, and D.H. Rhodes, Architecture frameworks in system design: Motivation, theory, and implementation, *Proc 17th Annu Int Symp INCOSE*, San Diego, CA, 2007a.
- M.G. Richards, N.B. Shah, D.E. Hastings, and D.H. Rhodes, Managing complexity with the Department of Defense architecture framework: development of a dynamic system architecture model, Massachusetts Institute of Technology, Engineering Systems Division, Cambridge, MA, 2007b.
- B. Schätz, A. Pretschner, F. Huber, and J. Philipps, Model-based Development, Institut für Informatik, Technische Universität München, Munich, 2002.
- A.-W. Scheer, ARIS—Business process frameworks, Springer, New York, 1998.
- A.-W. Scheer, ARIS—Business process modeling, Springer, New York, 1999.
- P.M. Senge, The fifth discipline: The art & practice of the learning organization, Currency Doubleday, New York, 1990.
- H.A. Simon, The sciences of the artificial, MIT Press, Cambridge, MA, 1981.
- J.F. Sowa and J.A. Zachman, Extending and formalizing the framework for information systems architecture, *IBM Syst J* 31(3) (1992), 590–616.
- SPC, Improving the software process through process definition and modeling, Software Productivity Consortium, International Thomson Computer Press, Boston, MA, 1996.
- J.D. Sterman, Business dynamics: Systems thinking and modeling for a complex world, McGraw-Hill, New York, 2000.
- A. Tang, J. Han, and P. Chen, A comparative analysis of architecture frameworks, School of Information Technology, Centre for Component Software and Enterprise Systems, Swinburne University of Technology, Melbourne, Australia, 2004.
- TOG, The Open Group Architecture Framework (TOGAF) 8.1.1, The Open Group, 2006.
- D. Unger and S.D. Eppinger, Product development process design: Planning design iterations for effective product development, MIT Center for Innovation in Product Development, Cambridge, MA, 2002.
- E. von Hippel, Task partitioning: An innovation process variable, *Res Policy* 19(5) (1990), 407–418.
- J.A. Zachman, A framework for information systems architecture, *IBM Syst J* 26(3) (1987), 276–292.



Tyson R. Browning is Assistant Professor of Enterprise Operations at the Neeley School of Business at Texas Christian University in Fort Worth, TX. He teaches Operations Management and Program Management in the MBA program and conducts research on managing complex enterprises, projects, and processes—the intersection of systems engineering and program management. He has served as a consultant for several companies including General Motors, Lockheed Martin, Northrop Grumman, Seagate, and Southern California Edison. He has previous work experience at Lockheed Martin Aeronautics Company, the Lean Aerospace Initiative, Honeywell Space Systems, and Los Alamos National Laboratory. Browning earned a B.S. in Engineering Physics from Abilene Christian University and two Master's degrees and a Ph.D. in Technology Management and Policy from the Massachusetts Institute of Technology. He has written numerous articles on engineering management, risk management, the design structure matrix, process modeling, and value measurement for conferences, books, and journals, including *ASME Journal of Mechanical Design*, *IEEE Transactions on Engineering Management*, *Journal of Operations Management*, *Production & Operations Management*, *Project Management Journal*, and *Systems Engineering*. He is a member of INCOSE, Institute for Operations Research and the Management Sciences (INFORMS), and Production and Operations Management Society (POMS).

APPENDIX: DESCRIPTIONS OF THE ACTIVITY AND DELIVERABLE ATTRIBUTES USED

Table VIII. Process and Activity (Work Package) Object Attributes and Descriptions

Attribute	Description
Name	A descriptive and distinctive name for the activity or process; should start with a verb (indicating an action) followed by a direct object; optionally, a modeler may also give the activity or process a unique identification number as a separate attribute
Parent	A link to the process object (or one of its modes) of which this activity is a part
Constituents ("Children")	If this activity is not at the lowest level of the activity breakdown structure, then this is a table containing a list of links to other activity objects that collectively describe the work done by this activity in greater detail; it is suggested to limit the number of constituent activities to 5-10 per level
Mode	A subtitle for the activity or process, for use in cases where more than one variant exists; can be used to represent activity crashing alternatives; e.g., four modes of software quality assurance are: desk check, peer review, walk-through, and Fagan inspection
Shadowing	A binary attribute; set to "Master" if the mode represents the master source; otherwise, set to "Shadow," where a shadow mode inherits all attributes of the master mode except where approved by tailoring
Deployment	A binary attribute; set to "Standard" if the mode represents a standard process; otherwise, set to "Deployed" and accompanied by the name of its instance—e.g., "ABC Project, XYZ area"; a deployed mode instance inherits all attributes of the standard mode except where approved by tailoring
Version Number	A user-specified number, incremented after any change to the mode, according to its significance; similar to the way software releases are numbered
Brief Description	An textual abstract of the activity mode, including its applicability (scope), general approach, and any other critical information
Inputs	A four-column table containing corresponding lists of (1) links to deliverable objects used by the activity mode, (2) the time when each deliverable is needed (expressed as a percentage of the activity completed—e.g., 0% indicates an input needed to begin the activity mode, while 50% indicates an input needed to do the second half of the activity mode), (3) the deliverable's required level of maturity or effectiveness, and (4) how the deliverable will be used (e.g., read only, update, delete)
Outputs	A three-column table containing corresponding lists of (1) links to the output deliverable objects produced by the activity mode, (2) the time when each deliverable is available (expressed as a percentage of the activity mode completed—e.g., 50% indicates an output available after the first half of the activity mode, while 100% indicates an output available only when the activity mode is complete), and (3) the deliverable's required level of maturity or effectiveness
Entry Criteria	A list of the events and/or conditions that should exist for the activity mode to begin execution; beginning without meeting these criteria implies additional risks; includes events that trigger or signal the beginning of the activity mode, such as authorizations, and the requisite conditions for beginning; note that the required maturity levels of inputs are also entry criteria, but these are associated with each input
Exit Criteria	A list of the events and/or conditions that should exist for the activity mode to stop execution; ending without meeting these criteria implies additional risks for downstream activities; includes events that trigger or signal the end of the activity mode, such as approvals, and the requisite conditions for ending; note that the required maturity levels of outputs are also exit criteria, but these are associated with each output
Verifications	A checklist of questions, tests, etc. used to confirm achievement of the exit criteria
Tools	A list of tools, templates, facilities, and any other non-standard equipment used to accomplish the activity mode; these items may be links to tool objects, where available
Standard Roles	A three-column table containing (1) a list of the roles to be filled to execute the standard activity mode, (2) the non-standard skills or training needed to fill each role, and (3) a link to the organization unit typically charged with providing the staff to fill each role
Deployed Roles	A three-column table containing (1) a list of the roles filled in this particular instance of the activity mode, (2) the non-standard skills or training possessed by those filling each role, and (3) a link to the organization unit filling each role
Standard Process Metrics	<p>A set of potential attributes of the standard activity mode:</p> <ul style="list-style-type: none"> • Duration (minimum, typical, and maximum) • Cost (minimum, typical, and maximum) • Repetition Discount (i.e., a learning curve effect) • Capability/Maturity Rating • Etc. (owner-defined) <p>Rather than being entered by a user, some of these metrics may be determined by other tools and imported.</p>

Table VIII. (continued)

Attribute	Description
Deployed Process Metrics	<p>A set of potential attributes of the <i>deployed instance</i> of the activity mode (e.g., as implemented on a project):</p> <ul style="list-style-type: none"> • Scheduled Duration (optimistic, planned, pessimistic)—i.e., “in this particular instance (e.g., on this project), we expect this activity to take this long” • Actual Duration (reported after execution is complete) • Budgeted Cost (optimistic, planned, pessimistic) • Actual Cost (reported after execution is complete) • Repetition Discount (i.e., a learning curve effect) • Schedule Criticality (the amount of slack or float in the activity; zero slack indicates that the activity is on the critical path) • Etc. (owner- and/or user-defined) <p>Rather than being entered by a user, some of these metrics may be determined by other tools and imported.</p>
Basis for Requirement	Lists any external standard such as AS9100 or CMMI that requires this activity mode and may provide a link to the section of the standard stating the requirement; or provides other rationale for the existence of the activity mode
Rules	A list of any non-obvious work policies, business rules, design rules, compliance requirements, sections of external standards, etc. affecting the planning or execution of this activity mode or instance thereof; may include links to the master source of information about each
References	A list of links to any references important to the accomplishment of the activity mode (e.g., manuals, instructions, guides, handbooks, etc.)
Standard Risks	A list of pitfalls, lessons learned, risks, failure modes (not to be confused with “activity mode”), or other potential problems encountered in past experiences with this activity mode or anticipated as possibilities; may also include guidance for avoidance or mitigation and links to further information
Deployed Risks	Additional risks foreseen for a deployed instance of the mode; over time, these risks can migrate to a standard risk list; deployed risks are often more specific and detailed than standard risks, which tend to be more generalized
Narrative Description	A longer, textual description of the activity mode, yet <i>as brief as possible</i> ; should provide a consistent level of detail throughout; where more detail is needed, two alternatives are suggested: (1) decompose the activity mode into constituent activities, wherein the additional detail can be discussed, or (2) provide a link to additional instructions (e.g., a manual); should avoid repeating information that is already addressed in (a) other attributes, (b) constituent activities, or (c) linked instructions; should not be cumbersome to read, write, or change
Tailoring Guidance	Brief instructions regarding tailoring, scaling, and sizing—e.g., restrictions, suggestions, etc.; where activity tailoring is likely, however, activities can be pre-tailored into modes
System Identification Number	A unique number, generated by and for use in the process database, to distinguish this version, mode, and instance from all other entries; not to be confused with any modeler-assigned number(s)
WBS Element Association	A list of all WBS elements for which this activity mode is used; assigned at the lowest-possible WBS level
Master Owner	A link to the lowest-level organizational unit(s) authorized to approve changes to the master mode and its instances
Standard Owner	A link to the lowest-level organizational unit(s) authorized to jointly approve changes to the standard and deployed instance(s) of the mode; if a master has no shadows, then this attribute is the same as Master Owner
Deployed Owner	A link to the lowest-level organizational unit(s) authorized to execute this instance of the mode and jointly approve any changes to it
Change History	A change log, mostly auto-generated by the process database system, containing user-entered reasons for any changes, dates of validity for each version, and links to old versions
Change Notifications	A list of links to organizational units who want to be informed of any changes to this mode

Table IX. Deliverable (Work Product) Object Attributes and Descriptions

Attribute	Description
Name	A descriptive and distinctive name for the deliverable; should include a noun and any necessary qualifiers (adjectives and adverbs); optionally, a modeler may also give the deliverable a number as a separate attribute
Parent	A link to the deliverable object (or one of its modes) of which this deliverable is a part
Constituents (“Children”)	If this deliverable is not at the lowest level of the deliverable breakdown structure, then this is a table containing a list of links to other deliverable objects that comprise this deliverable; it is suggested to limit the number of constituent deliverables to 5-10 per level
Mode	A subtitle for the deliverable, for use in cases where more than one variant exists; e.g., three modes of a deliverable could be: initial estimates, preliminary results, and final results
Shadowing	A binary attribute; set to “Master” if the mode represents the master source; otherwise, set to “Shadow”; a shadow mode instance inherits all attributes of the master mode except where approved by tailoring
Deployment	A binary attribute; set to “Standard” if the mode represents a generic deliverable in a standard process; otherwise, set to “Deployed” and accompanied by the name of its instance—e.g., “ABC Project, XYZ area”; a deployed mode instance inherits all attributes of the standard mode except where approved by tailoring
Version Number	A user-specified number, incremented after any change to the mode, according to its significance; similar to the way software releases are numbered
Brief Description	A textual abstract of the deliverable mode, including its general content and any other critical information
Suppliers	A table containing a list of links to supplier activity modes that produce the deliverable mode; generally, a deliverable mode should be provided by only one supplier, although in some cases there are options as to which source to use; such options would usually be eliminated in deployed instances, once selections have occurred
Customers	A two-column table containing (1) a list of links to the customer activity modes that receive the deliverable mode and (2) the deliverable mode’s required level of maturity or effectiveness in the eyes of each customer
Key Criteria and Measures of Effectiveness	A two-column table containing (1) a list of the criteria by which the customers judge the effectiveness, quality, and usability of the deliverable mode and (2) for each criterion, one or more measures of effectiveness or performance level
Requirements	For deployed instances only; a list of customer- and supplier-agreed performance levels for each measure of effectiveness, as well as any other stipulations on the quality and specifications of the deliverable; i.e., targets or goals
Acceptance Criteria	A checklist of questions, tests, etc. that will be used to confirm achievement of the requirements
Standard Process Metrics	<p>A set of potential attributes of the standard deliverable mode:</p> <ul style="list-style-type: none"> • Criticality (to risk reduction, etc.) • Volatility (propensity of deliverable to be modified after its initial creation) • Complexity • Etc. (owner-defined) <p>Rather than being entered by a user, some of these metrics may be determined by other tools and imported.</p>
Deployed Process Metrics	<p>A set of potential attributes of the <i>deployed instance</i> of the deliverable mode (e.g., as implemented on a project):</p> <ul style="list-style-type: none"> • Commitment Status (e.g., “requested,” “being negotiated,” “commitment in place”); listed separately for each supplier and customer • Scheduled Time of Availability (due date) • Actual Time of Availability (reported after deliverable is provided) • Actual Quality (reported in terms of each of the requirements after deliverable is provided) • Customer Satisfaction (reported after deliverable is provided) • Etc. (owner- and/or user-defined) <p>Rather than being entered by a user, some of these metrics may be determined by other tools and imported.</p>
Format	The format in which the deliverable mode will be provided, stored, etc.—e.g., file format, spreadsheet, word document, etc.
Medium	The medium of transfer of the deliverable mode—e.g., deposit in database or document management system, e-mail, phone, hand delivery, etc.
Artifact	A link to the actual deliverable artifact, if it exists in a computerized repository such as a product data management system
Rules	A list of work policies, business rules, design rules, compliance requirements, sections of external standards, etc. affecting the planning or execution of this deliverable mode or instance thereof; may include links to the master source of information about each
References	A list of links to any references pertaining to this deliverable mode—e.g., manuals, instructions, guides, etc.

Table IX. (continued)

Attribute	Description
Narrative Description	A longer, textual description of the deliverable mode, if necessary; should nevertheless be <i>as brief as possible</i> ; should provide a consistent level of detail throughout; where more detail is needed, it is suggested to decompose the deliverable mode into constituent deliverables, wherein the additional detail can be discussed; should avoid repeating information that is already addressed in other attributes or constituent deliverables; should not be cumbersome to read, write, or change
Tailoring Guidance	Brief instructions regarding tailoring, scaling, and sizing—e.g., restrictions, suggestions, etc.
System Identification Number	A unique number, generated by and for use in the process database, to distinguish this version, mode, and instance from all other entries; not to be confused with any modeler-assigned number(s)
WBS Element Association	A list of all WBS elements for which this deliverable is used; assigned at the lowest-possible WBS level
Change History	A change log, mostly auto-generated by the process database system, containing user-entered reasons for any changes, dates of validity for each version, and links to old versions
Change Notifications	A list of links to organizational units who want to be informed of any changes to this deliverable mode