Regular Paper

Key Concepts in Modeling Product Development Processes*

Tyson R. Browning,[1] Ernst Fricke,[2] and Herbert Negele[2]

[1]M.J. Neeley School of Business, Texas Christian University, TCU Box 298530, Fort Worth, TX 76129

[2]BMW Group, Knorrstrasse 147, 80788 Munich, Germany

Received 27 July 2005; Revised 11 November 2005; Accepted 13 November 2005, after one or more revisions
Published online in Wiley InterScience (www.interscience.wiley.com).
DOI 10.1002/sys.20047

ABSTRACT

This paper provides a foundation for modeling the set of activities and their relationships by which systems are engineered, or, more broadly, by which products and services are developed. It provides background, motivations, and formal definitions for process modeling in this specialized environment. We treat the process itself as a kind of system that can be engineered. However, while *product* systems must be created, the *process* systems for developing complex products must, to a greater extent, be discovered and induced. Then, they tend to be reused, either formally as standard processes, or informally by the workforce. We distinguish and clarify several important concepts in modeling processes, including: product development versus repetitive business processes, descriptive versus prescriptive processes, activities as actions versus deliverables as interactions, standard versus deployed processes, centralized versus decentralized process modeling, "as is" versus "to be" process modeling, and multiple phases in product development. We also present a basically simple yet highly extendable and generalized framework for modeling product development processes. The framework enables building a single model to support a variety of purposes, including project planning (scheduling, budgeting, resource loading, and risk management) and control, and it provides the scaffolding for knowledge management and organizational learning, among numerous other uses. © 2006 Wiley Periodicals, Inc. Syst Eng 9: 104–128, 2006

Key words: process modeling; product development; systems engineering; system design process; engineering management; project management

*This paper would not be possible without the benefits of uncounted interactions with professional and academic colleagues over the past decade. We thank the many participants in our tutorials at the 2001–2003 International Symposia of INCOSE for their helpful comments and feedback. We also thank Andy Sage for his comments and advice and three anonymous reviewers for their helpful suggestions, all of which improved the paper.

†Authors' e-mail addresses: T.Browning@tcu.edu; Ernst.Fricke@bmw.de; Herbert.Negele@bmw.de

1. INTRODUCTION

Processes are a key aspect of contemporary systems engineering (SE) theory and practice. In addition to their use in the engineering of systems, they are at the heart of approaches such as project management, Total Quality Management (TQM), Lean, Six Sigma, reengineering, ISO 9000,[1] CMMI^{SM},[2] etc. However, the term "process" and the tasks most properly associated with process definition, compliance, and improvement are some of the most "differently understood" concepts around. The problem is exacerbated by the fact that "process" is such a simple word: Everyone thinks they know what it means, but varied opinions about its implications have caused uncounted man-years of meetings, debates, and waste in many industries. And while many have tried to settle the debate by providing process modeling standards and tools, most of them have fallen short in important areas, particularly when it comes to treating processes as systems and using an SE approach in their development.

In this paper, we hope to clarify some of these "dif-understandings" by discussing the motivations for and key concepts of processes as used for systems development, or, more generally, *product* (and service) *development* (PD). These perspectives and approaches stem from our variety of experiences working in the international automotive and aerospace industries and our backgrounds of research on many aspects of the subject. The content is based on a number of interactive presentations to experienced practitioners, including the first part of a full-day tutorial offered thrice at the International Council for Systems Engineering (INCOSE) 2001–2003 International Symposia. The other parts of the tutorial addressed three other important areas, as shown in Figure 1. Due to space constraints, this paper does not address the content in parts two through four: Rather, it sets the foundation for them.

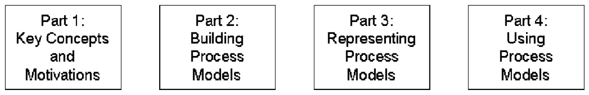

Part 1: Key Concepts and Motivations
Part 2: Building Process Models
Part 3: Representing Process Models
Part 4: Using Process Models

Figure 1. Content of tutorial on product development process modeling.

Before going further, we need clear definitions of the terms PD, process, and model. PD is an endeavor comprised of the myriad, multi-functional activities done between defining a technology or market opportunity and starting production.[3] The goal of PD is to create a "*recipe*" for producing a product [Reinertsen, 1999]. This recipe must conform to requirements stemming from customer or market needs. It includes the product's "ingredients" (bill of materials) and "preparation directions" (manufacturing, supply, distribution, and support systems). While many business processes seek an identical result, repeatedly, PD seeks to do something new, once.[4] Hence, PD involves creativity and innovation and is nonlinear and iterative [Kline, 1985]. Although certain activities may repeat, the desired overall result is unique (unless one speaks in terms of a generic result like "success"). Actually, this is the case for any *project* or *program*, as these terms are defined in the project management literature: For example, "A project is a temporary endeavor undertaken to create a unique product, service, or result" [PMI, 2004, p. 5]. Hence, the concepts presented in this paper apply to any project or program.

A *model* is an abstract representation of reality that is built, verified, analyzed, and manipulated to increase understanding of that reality. Models can reside in the mind (mental models) or be codified. "All models are wrong, but some are useful" [Box, 1979]. A useful model is helpful for making predictions and testing hypotheses about the effects of contemplated actions in the real world, where such actions would be too disruptive or costly to try. A useful model also provides insights otherwise available only through (sometimes painful) experience. While scientists focus on *descriptive* models, engineers and managers furthermore want *predictive* models, for which validation and estimation of modeling error are practically impossible [Hazelrigg, 1999]. Here, we are interested in models that help represent, understand, engineer, manage, and improve PD processes.

A *process* is "an organized group of related activities that work together to create a result of value" [Hammer, 2001] or "a network of customer-supplier relationships and commitments that drive activities to produce results of value."[5] Thus, one can think of the work on any project or program as a large process. Process models are typically activity network models. Ambiguities, uncertainties, and interdependencies of activities, their results, their assigned people, and their tools make PD processes extremely complex and challenging to

[1]International Organization for Standardization.

[2]Capability Maturity Model®—Integrated. Capability Maturity Model® and CMMI^{SM} are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

[3]There may not necessarily be a clean break between development and production: some test and evaluation units may require parts of the production process prior to the "official" start of production, and some development work may continue past production start.

[4]Of course, PD includes a spectrum of intensities of "newness," from the truly novel to the minor upgrade of an existing system. Most of our discussion is oriented towards the novel end of the spectrum.

[5]Gabriel Pall, personal communication; based on Pall [1999].

model. We must clearly distinguish between reality (the way work really gets done) and a model (an abstract description of the way work can or should get done). Processes exist in both places. Especially in industry, the term "process" is used in both contexts without clear qualification—causing many to mistake the map for the territory. Any work (even creativity and innovation) done to produce a result has a process, though perhaps not a process model. In other words, *every enterprise has processes* (ways to get results), even though they may not be modeled, documented, consistent, effective, or efficient.

The literature on process modeling is extensive, although only a relatively small subset of this literature addresses PD processes specifically. Most of the literature is aimed at business and manufacturing processes, which, as we discuss in Section 3.1, differ from PD processes in significant ways. Several reviews of PD process models are available [Browning and Ramasesh, 2005; O'Donovan et al., 2005; Smith and Morrow, 1999], so we will not repeat them here. Instead, we summarize what we recognize as some fundamental propositions that form the basis of PD process modeling theory:

i. *The process of invention/innovation cannot be fully mechanized.* Despite continued advances in artificial intelligence, meta-modeling, and techniques such as TRIZ [e.g., Savransky, 2000; Clausing and Fey, 2004], we cannot prescribe a completely mechanistic approach to creativity [Dougherty, 2001].

ii. Nevertheless, *the PD process has some repeatable structure*. This proposition stems from the engineering design and systems engineering literature, where design is something of an art but with many consistent patterns. That is, while PD seeks to do something new, once, an individual or organization tends to follow a similar approach in each instance and learns and adapts (more or less) through successive occurrences.

iii. The project management task is to plan and control the project, determine and schedule activities, manage commitments, etc. *Project management is facilitated by a structured approach*, especially one supported by models of what work can and should be done when, and what information can and should be created when—i.e., process models.

iv. Processes can be regarded and treated as systems that should be engineered purposefully and intelligently, facilitated by useful models [Negele, Fricke, and Igenbergs, 1997; Pajerek, 2000].

v. In many aspects, complex process behaviors can be better understood by examining their relatively simpler, constituent parts (actions) and those parts' endogenous and exogenous relationships (interactions). We refer to this as the *decomposition paradigm* in process modeling. While it is also at times referred to as the reductionist paradigm, we resist that description as misleading: We do not mean to imply that a system can be fully understood by reducing it to a mere set of elements and relationships. However, holonic decomposition can provide an effective means of organizing a useful model.

vi. *There is always a gap between the real system and a model of it.* The size of this gap is determined by the model's richness, fidelity, accuracy, "realism," etc. In modeling, verification and validation are used in an effort to close this gap. However, many models can be quite "useful" despite large gaps, if these gaps are chosen appropriately.

vii. *Process models are built for a purpose*, such as to document the way work is done, to estimate the duration of a project, etc. [Browning and Ramasesh, 2005]. Process models built for one purpose may not be useful for other purposes, although this type of misuse is common in industry. We discuss this issue further in Section 2.2.

These propositions form some of the theoretical basis for PD process modeling. They have enabled the construction and use of a large number of PD process models. But they have not ensured success. For example, process modelers may assume that too much of the PD process structure should persist from one project to the next.

At this early point, we must also distinguish a process model from a process Capability Maturity Model (CMM). For example, the CMMI [SEI, 2002] contains 25 required and expected process areas—such as Requirements Management, Risk Management, etc.—which altogether include around 185 practices. However, the CMMI does not fully define the interfaces, interactions, or flow between the process areas or practices, and it does not cover the specifics of engineering design for particular products. Thus, CMMs and other such standards may show what kinds of activities to include in a process model, but not how to string them together at the level of execution. A CMM is a model of a process's capability and maturity, not a model of the process itself.

With this introduction to processes and process models, the rest of this paper is organized as follows. In Section 2, we discuss the motivations for building proc-

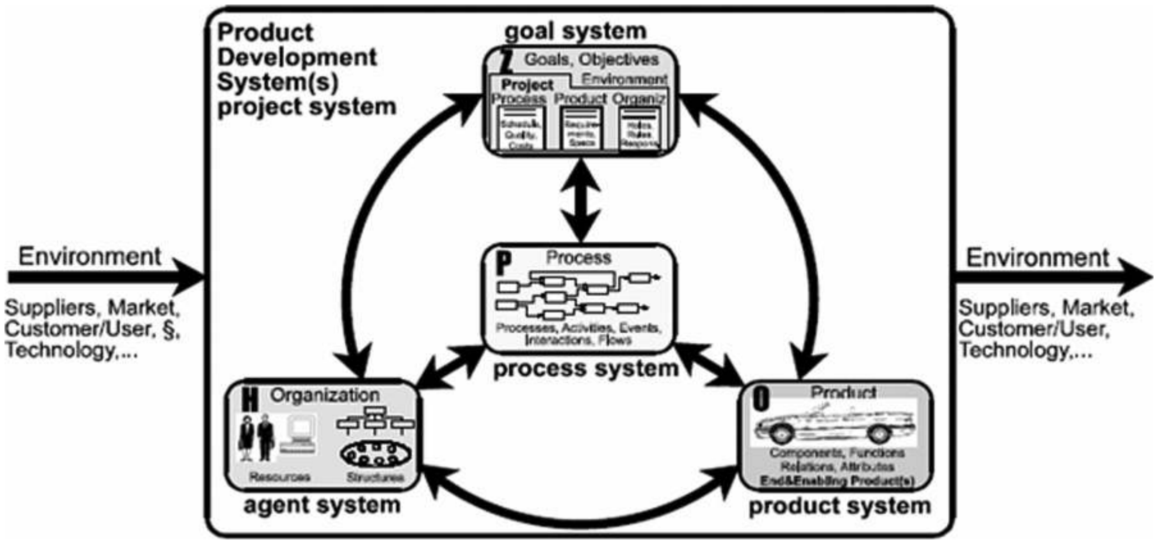

Product Development System(s) project system
goal system
Z Goals, Objectives
Project Environment
Process Product Organiz.
P Process
Processes, Activities, Events, Interactions, Flows
process system
H Organization
Resources Structures
agent system
O Product
Components, Functions Relations, Attributes End&Enabling Product(s)
product system
Environment
Suppliers, Market, Customer/User, §, Technology,...
Environment
Suppliers, Market, Customer/User, Technology,...

Figure 2. ZOPH model [from Negele, Fricke, and Igenbergs, 1997; Negele, 1998; Negele and Wenzel, 2000, with permission from Dr. Negele].

ess models and treating them as systems. Section 3 addresses some key concepts, and Section 4 presents our proposal for a generalized framework for modeling PD processes. Section 5 provides our concluding remarks.

2. MOTIVATION

2.1. Why Treat Processes as Systems?

While most SE literature focuses on physical (hardware and software) systems—i.e., products—and while some addresses political, societal, and economic systems, a process is also a kind of system—one that has received much less attention as such. Comparing the definitions of a process given in the previous section to the following definitions of a system should make this obvious. A system is:

- A regularly interacting or interdependent group of items forming a unified whole
- A group of devices or artificial objects or an organization forming a network especially for distributing something or serving a common purpose
- An organized or established procedure[6]
- An integrated set of elements that accomplish a defined objective.[7]

[6]The first three definitions are from the Merriam Webster online dictionary (www.m-w.com).

[7]The fourth definition is from INCOSE [2004].

In fact, the product and process systems are just two of several important systems in a project or program. In the ZOPH[8] model (Fig. 2), Negele, Fricke, and Igenbergs [1997] identify these and two other important systems, the agent[9] and goal systems. Based on other work [Eppinger and Salminen, 2001; Browning, 2001], Figure 3 shows an organization system and a tool system explicitly, yielding five systems. Each of the five systems is related to the others, is composed of elements with relationships, and thus can be discussed in terms of its network structure and architecture. The product system is the desired result of the project—i.e., in the case of PD, the product "recipe" described previously. Here, the system consists of designed physical (hardware, software, and/or people) components that may be related via a variety of types and degrees of interactions. The process system is the work done and interim results achieved to produce the product system. The process system consists of related activities. The organization system consists of people assigned to do the work to produce the product system—i.e., individuals, groups, teams, or other organizational units, related to each other by communication, reporting, etc. The tool system represents the technologies used by the people to do the work to get the product. While some tools, like drafting boards and pencils, may be essentially unrelated in terms of a PD system model, many contempo-

[8]ZOPH is the German acronym for Zielsystem (goal system), Objektsystem (product system), Prozeßsystem (process system), and Handlungssystem (agent system).

[9]Negele, Fricke, and Igenbergs's [1997] conception of the agent system includes organization, relevant technologies, resources, methods, tools, etc.

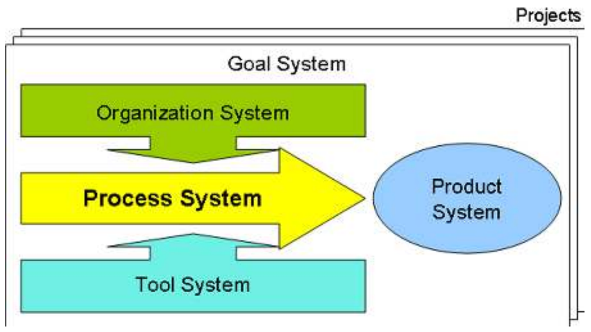


Figure 3. Five systems in a project.

rary software tools must interact as a direct result of the relationships between activities and the people doing them. The software applications and information technologies (IT) used by an organization to accomplish work constitute a significant portion of the tool system. The product, process, organization, and tool systems operate in the context of requirements or goals, which may themselves be related: e.g., making it easier to meet one requirement may make it more difficult to meet another. Hence, the fifth system is the goal system. Thus, each of the five systems is related to and both enables and constrains the others, even though Figure 3 does not show all such relationships; Figure 2 does a better job in this regard. An enterprise typically has multiple projects going on at once (represented by the layers in Fig. 3), and there are strong incentives to achieve commonality in these five systems across projects.

According to the propositions advanced above for process modeling, models of these five systems and their interactions would be extremely valuable aids for designing, managing, and improving them [Danilovic and Browning, 2006]. While models of the product, process, and organization systems typically exist to some extent in most projects, these models are seldom integrated or used to verify each other in industry. This is a tragedy, because a tremendous amount of waste could be eliminated, and much more informed technical and managerial decisions could be made, if an integrated model spanned these systems [Negele, 1998; Negele and Wenzel, 2000]. For example, Figure 4 stylistically depicts how information from an integrated model could be filtered to show individual systems and their interactions—e.g., people, teams, departments, and other organization units assigned to various processes; tools used by particular organizational units or for particular activities; and requirements bearing upon particular product components, process activities, organizational units, or tools. However, as Figures 2 and 3 emphasize, the process system is in many ways the nexus of the systems' interactions: If the project were a sentence, the process would be the action verb. Hence, in our estimation, *a useful process model is the basis for and key to the effective integration of the project system models and the effective management of projects*.

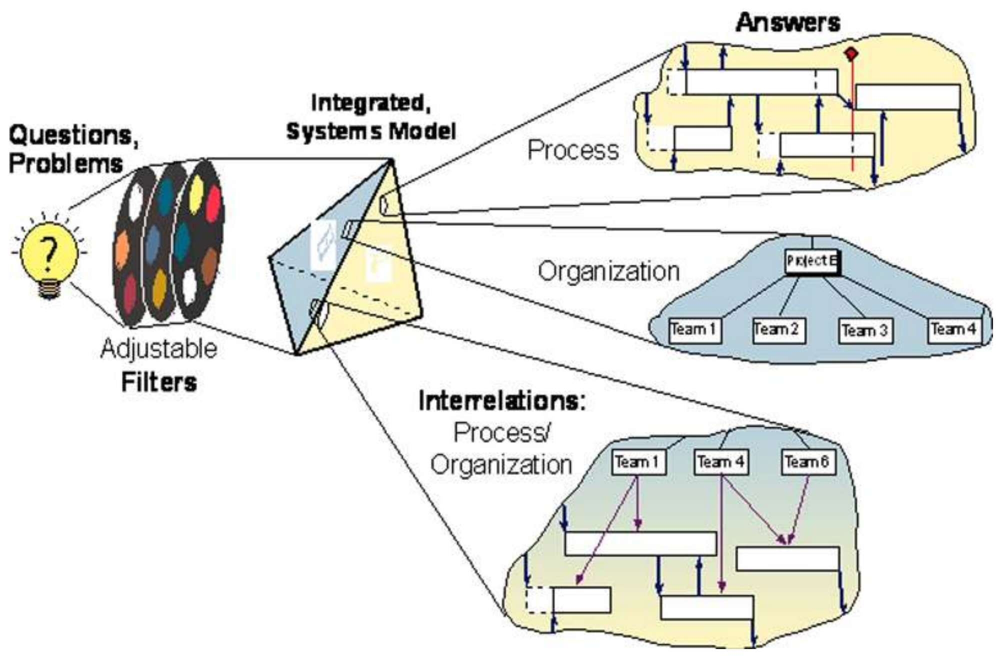


Figure 4. Project analysis and decision support enabled by a multisystem model [from Negele et al., 1997, with permission from Dr. Negele].

Thus, a process is a system, and, because of its vital relationships to the other systems in a project, a model of the process system is an excellent basis for integrating models of the other project systems. Unfortunately, however, many process models are poor representations of processes as systems. To drive this point, consider two of Rechtin's [1991, p. 29] salient heuristics for systems architecting:

- "Relationships among elements are what give systems their added value."
- "The greatest leverage in systems architecting is at the interfaces."

Restating these heuristics in terms of processes, we get:

- Relationships among activities are what give processes their added value.
- The greatest leverage in process architecting is at the interfaces.

Hence, a good process model should pay special attention to the interfaces among activities. However, this is not the case with many process models [Browning, 2002]. For example, many flowcharts include a nominal number of arrows—just enough to connect the boxes—and label the boxes but not the arrows. Other (mental and codified) process models have led to the common expression of processes as chains, whereas in reality, especially in PD, processes are in fact dense networks or webs of interdependencies [Negele, 1998; Negele, Fricke, and Igenbergs, 1997], as shown in Figure 5. Therefore, a rich, holistic, integrated model of the PD process system is necessary, and this model must integrate with models of the other systems in PD projects.

2.2. Why Build Process Models?

Process models can be built for a variety of reasons. Traditionally, process models have provided a basis for planning and managing projects. By listing the activities to be done (often resulting from a work breakdown structure [WBS]) and their dependencies, planners can get an idea of a project's critical path, duration, etc. From there, they can explore opportunities to reduce duration (e.g., through "crashing"). Thus, the temporal aspect of projects has been the traditional focus, and this application of process models has spread widely through the project management community—q.v., the Project Management Institute's *Guide to the Project Management Body of Knowledge* (PMBOK) [PMI, 2004].

More recently, higher product complexity, increased competition, customer expectations for customization, shortened reaction times, and larger numbers of activities and amounts of information to coordinate have increased the need for a systematic approach to managing projects, especially large ones (i.e., programs). Concurrent Engineering and Integrated Product and Process Development (IPPD) have increased the overlap among PD activities, dramatically increasing the coordination challenge. Standards such as CMMI have mandated additional activities to include in projects. Increased coordination with partners and suppliers (made possible by technology, but not necessarily regulated by that technology) has also contributed to increased complexity in contemporary PD projects. The complexity of the PD process correlates with the complexity of the product being developed. Very rarely, if ever, can a single person or small team effectively, efficiently, and consistently identify and coordinate all the required interactions. It is no wonder, then, that so many projects and programs fail to meet their stakeholders' expectations, not only for technical reasons, but also for so-called "managerial reasons"—i.e., failure to meet technical goals within expected cost and/or schedule constraints.

Also, in the last 20 years, process descriptions have become the basis for codifying and communicating certain organizational knowledge about what work to do and how to do it. Many companies' internal policies, and external standards such as the ISO 9000 series, mandate process documentation. While the noble idea is to provide the workforce with a common description of work methods and a basis for coordination, at least four significant barriers have prevented this from happening in most cases.

First, the models are usually too abstract and ambiguous for most workers' day-to-day needs. In fact, companies have some incentive to keep the models *purposefully ambiguous* so that when a process conformance auditor shows up, any actions by the workforce will in high likelihood be found to fall under the large and loose umbrella of a vague process description. Thus, subtle pressure often exists to build a kind of a process model that is not really useful for day-to-day

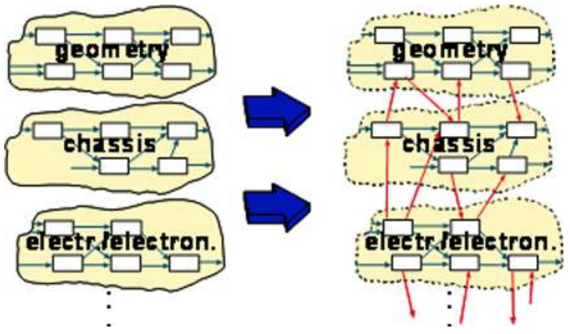


Figure 5. Processes are networks, not chains [from Negele, 1998, with permission from Dr. Negele].

Table I. Some Uses of Process Models

- To focus on value-adding activities, not on figuring out where to get inputs and where to provide outputs
- To provide transparency and situation visibility to the workforce, so each worker is empowered to see their part in the whole enterprise
- To ensure good decision-making by the right people at the right time using the right information
- To meet commitments in a predictable, repeatable, consistent way
- To support understanding of and learning about complex processes
- To benefit from captured best knowledge on how to do certain things
- To provide a "scaffolding" or skeleton for organizing knowledge about work and interactions (knowledge management)
- To plan and manage work with some accuracy and confidence
- To help avoid failure modes that beset previous, similar processes
- To have a common vocabulary for discussing work and results
- To codify an approach to accomplishing a project, against which each participant can compare their own mental model and either alter it to align with the group or trigger a discussion about potential problems
- To provide a baseline way of doing something, against which improvements can be measured
- To use a common, best approach when people do similar things on different projects or programs
- To enable "what if" analyses of potential process changes
- To enable innovative process improvements (reengineering, lean, etc.)
- To convince customers that a reasoned, proven approach is being used to meet their needs
- To convince auditors that work is done to certain standards (e.g., ISO 9001)

management and coordination. This unfortunate circumstance has caused many debates, misunderstandings, and wastes of time and money in industry, not to mention a failure to realize the advantages provided by a really useful process model.

Second, the common flowcharting approach (and many other approaches) to process modeling fails to capture most of the relationships between the activities in a project. The resulting models do not help the workforce to self-coordinate; they are used primarily for scheduling. In contemporary practice, the process flowchart has often been replaced by the even less informative Gantt chart view. Even for scheduling purposes only, the process models in use are often woefully inadequate. Schedules are just a partial view of a process model. They are one scenario for how activities might work together. They do not address exactly what information must flow or what state it must be in. And they are often notoriously wrong, because they are based on undocumented, unshared, and often false assumptions. Something more useful is needed to support the development of complex systems.

Third, building a process model takes time, and the most elaborate process models have taken a very long time to build. Aside from the obvious question about justifying the return on such investments, the long development time exacerbates the "wish-was" problem, where process modelers tend to document either the way they wish a process would occur or else the way it occurred some time ago.[10] Often, a centralized group of "process modelers" is pulled away from "real work" to document other peoples' work, resulting in a model that those doing the "real work" fail to embrace. The information contained in the process model is often documented in user-unfriendly prose ("process documentation" or "procedures"), which is extremely cumbersome to maintain, thereby compounding the process model's anachronism.

Fourth, and worst of all, the policies of many companies have forced employees to work in a constant state of cognitive dissonance, where they must pretend to follow a documented process while doing "what really needs to be done." These and other shortcomings of most process models in industry have led to indifference, dismissal, and sometimes hostility from the workforce in regard to process modeling. In essence, most of the problems with and shortcomings of the process models used in industry can be traced to a fundamental lack of knowledge and education about what a process model is and what it can be used for. We begin to address this shortcoming in this paper.

Process models can be built for a variety of uses beyond project scheduling and compliance with external standards. Drawing from lists by Fricke et al. [1998] and Browning [2002], Table I provides a list of potential purposes for modeling processes. Browning and

[10]As best as we can recall, credit for the term "wish-was" goes to Steven D. Eppinger.

Ramasesh [2005] provide the taxonomy of purposes shown in Table II. Models built for one of these purposes are often not very useful for the others—not that this need be the case; it just often is. For example, process models built for the purpose of compliance with standards may not be good for day-to-day project coordination. Models built to facilitate day-to-day coordination are often not the ones useful for communicating with executives and customers. Unfortunately, attempting to force-fit a model developed for one purpose in order to suit another is common practice. Different users have different needs and require different information, emphases, and views—e.g., those managing a schedule may need a Gantt chart. Of course, models should be built with a purpose in mind. Sometimes, modelers lose sight of this and continue well past the point of satisfying a need—modeling for the sake of modeling. Modeling is a means, not an end.

A modeling *framework* is a generic approach which may be applied to modeling any situation within its scope, but which in itself provides only general insights. Some example process modeling frameworks are PERT/CPM, DSM, and IDEF0.[11] Within a framework, a modeling *instance* is a model created to provide specific guidance. Using a metaphor familiar to children, a framework is a sandbox and a model (instance) is a sandcastle. Just as the properties of sand limit the forms that may be created, a framework constrains the features of models that may be built within it [O'Donovan et al., 2005]. The PD process modeling literature includes a variety of frameworks, including the list of 18 compiled in Table III (which does not address all process modeling frameworks, only those applied in the literature to modeling the PD process). While other structured modeling frameworks, such as Unified Modeling Language (UML) and functional flow block diagrams, are widely used in the SE field, these applications are almost exclusively to modeling *product* system processes, functionality, and behaviors—not the activities undertaken by the engineers themselves.

The variety of users and needs for process models is part of the reason why there are so many modeling frameworks in academia and so many disparate models in industrial organizations. On a single, large project, for example, perhaps one group manages schedule with a model of activities and relationships. Another group manages budget with a list of activities and cost accounts (often failing to account for the relationships among activities). Yet another group manages risk, tying projected risk reductions to certain events, activities, or milestones. Each of these groups has its own models, yet the lists of activities—upon which each of the models are based—are often unsynchronized at best or very different at worst.

Table II. Taxonomy of Purposes for Process Models [adapted from Browning and Ramamesh, 2005]

1. PD project visualization
   a. Actions, interactions, and commitments
   b. Customized "views"
2. PD project planning
   a. Making commitments
   b. Choosing activities
   c. Structuring the process
   d. Estimating, optimizing, and improving key variables (time, cost, etc.)
   e. Allocating resources
3. PD project execution and control
   a. Monitoring commitments
   b. Assessing progress
   c. Re-directing
   d. Re-planning
4. PD project development
   a. Continuous improvement
   b. Organizational learning and knowledge management
   c. Training
   d. Metrics
   e. Compliance

A *generalized framework* for process modeling could provide a basis for integrating the disparate models in use across an organization. Information unneeded by one group of users for their particular purposes could be hidden, yet all users would be "drawing from the same well." Assumptions made by one group would automatically be validated against the latest, actual information possessed by another group. With an integrated process model—one that accounted for activity durations, costs, relationships, effects on technical performance criteria, effects on risk measures, pitfalls (failure modes) and lessons learned, etc.—an organization would benefit by being able to explore "what if" scenarios that traded off cost, schedule, technical performance, and risk. In Section 4, we propose that a generalized framework for process modeling can indeed suit the variety of purposes in Tables I and II.

We hope to convey the power of a process model to support dynamic, innovative PD processes. The point of process modeling is not to meticulously model the details of an anachronistic process or hamper creativity. Rather, we propose that a process model serve to empower the workforce by getting the information they need (and especially the information they do not know they need) to them at the right time, thereby freeing

[11]PERT: project evaluation and review technique; CPM: critical path method; DSM: design structure matrix; IDEF: integration definition, of which versions "0" (IDEF0) and "3" are most widely used for PD process models.

Table III. 18 PD Process Modeling Frameworks[a, b]

Framework	Example References	Viewpoint on PD Process	Distinguishing Assumptions	Typical Purposes	Key Variables/Attributes
Phase/Stage-Based Models	[Boehm 2000] [Cooper 2001]	A set of sequential or iterative phases/stages, usually with criteria for stage transition	• Other, lower-level models will manage within stages • Review gates between stages	• High-level characterization of PD process • Evaluation of resource allocation among stages • Stage tradeoffs and risks	• Stages iterative or not? • Gate impedance • Key decision points • Key milestones
Activity/Phase Overlapping	[Krishnan *et al.* 1997] [Terwiesch and Loch 1999]	A dyad of coupled activities, which can be overlapped to some extent	• Working with preliminary information increases chances of rework • Activities vary in sensitivity • Activities evolve outputs	• Optimize degree of activity overlapping with respect to time, cost, and/or performance	• Information evolution rate • Activity input sensitivity • Rework costs • Information exchange frequency
Activity Networks (Flowcharts, PERT/CPM)	[Elmaghraby 1995]	A forward flow of activities with nominal, acyclical dependencies	• Activity finish-to-start (F-to-S) relationships • Independent activity times • CP duration = project duration	• Completion time estimates • Slack/Float time calculation • Time-cost tradeoffs (crashing) • Cost or time minimization • Process mapping	• Stochastic activity durations • Time-cost elasticities • Slack/Float times • Early & late starts and finishes
GPRs	[Elmaghraby 1995]	Activity network with greater complexity and variety of interfaces	• Activity relationships besides F-to-S: F-to-F, S-to-S, and S-to-F	• Network analysis • Activity criticality	• Activity relationship type • Early start and before finish timings
GERT	[Moore and Taylor 1977] [Neumann 1990]	Activity network with cyclical dependencies (looping)	• Activity dependencies can be contingent and cyclical	• Network analysis	• Probabilistic branching and looping • AND, OR, XOR nodes
Queuing Models	[Taylor and Moore 1980] [Adler *et al.* 1995]	A network of activities executed and constrained by limited processors (resources)	• Resource contention and constraints	• Resource-constrained project scheduling • Resource allocation	• Activity resource requirements • Resource priorities and assignments
IDEF (IDEF0, IDEF3), SADT	[NIST 1993] [Marca and McGowan 1993] [Menzel and Mayer 1998]	A hierarchy of decomposed levels of activities (functions) with several types of input-output relationships	• Processes are hierarchies of networks • Inputs and outputs deserve explicit attention • Timing is often ignored	• Sophisticated mapping of processes	• Activity inputs, controls, outputs, mechanisms (ICOMs)
Design Structure Matrices	[Steward 1981] [Eppinger 2001] [Browning 2001]	A structured network of activities with substantial and cyclical dependencies	• Dependency network is relatively dense • Feedback dependencies cause iteration/rework	• Activity sequencing • Iteration management • Concise representation of process	• Dependency strengths • Activity sequence • Network structure

Framework	Example References	Viewpoint on PD Process	Distinguishing Assumptions	Typical Purposes	Key Variables/Attributes
Control Theory Models	[Joglekar and Ford 2005] [Lee *et al.* 2004]	A set of concurrent activities that generate work for each other	• Process structure drives convergence rate • Closed-loop optimal control	• Resource allocation policies • Completion time estimates • Characterize process stability	• Resource allocation and work transformation matrices
Petri Nets	[van der Aalst and Hee 2004] [Tacconi and Lewis 1997]	A discrete event system of activities (transformations) and states	• Pre-conditions for transformations to occur • Often assume steady-state conditions	• Process verification for reachability, deadlocks, efficiency, etc. • Workflow modeling • Manufacturing process modeling	• States, transformations, weighted arcs • Event graphs • Circuits and loops
Markov Models	[Smith and Eppinger 1997] [Ahmadi and Wang 1999]	A set of states achievable through the execution of activities	• Activity completion triggers state transition • Rework states	• Project time characterization and minimization • Activity sequencing	• Transition probabilities • Activity durations
System Dynamics	[Ford and Sterman 1998] [Ford and Sterman 2003]	A few major phases that create amounts of work and rework for each other	• Each phase modeled using a generic structure • Work can be measured as generic "tasks to be done"	• Process behavioral analysis • Iteration management policy	• Degree of concurrency • Work release policies • Rework discovery rate
IPO (SIPOC, ETVX)	[Fricke *et al.* 2000] [Negele *et al.* 1999] [Radice *et al.* 1985]	A set of activities, each requiring inputs and producing outputs	• Timing is often ignored • I/O links must agree	• Process description and documentation	• Inputs, suppliers • Outputs, customers • Sub-processes
Signposting	[Clarkson and Hamilton 2000] [O'Donovan *et al.* 2003]	A set of activities with alternative input requirements and output capabilities depending on information confidence and maturity	• I/O varies in quality/maturity and governs activity activation	• Design process analysis • Design confidence mapping	• I/O confidence levels • Activity "possibility"
Business Process Modeling	[Arkin 2002] [Scheer 1998b] [Kamath *et al.* 2003] [Sivaraman and Kamath 2003]	A business process with discrete, event-driven activities, interfaces, and timing issues	• Generalized, object-oriented modeling methods • Model storage in database • Model reuse	• Reengineering • Enterprise engineering • IT integration • Workflow management	• Activities as objects with myriad potential attributes • Other types of objects—e.g., documents
Network of Commitments	[Pall 1999]	A "network of commitments" implied by dependencies among activities	• Activity owners must commit to dependencies	• Process management and adaptation	• Communication acts • State of commitments
Value Stream Mapping	[McManus 2004]	A set of value-adding and non-value-adding activities	• Emphasize determination of value-adding activities	• Process mapping • Application of Lean principles • Waste reduction	• Value-adding activities • Information flows
Process Grammars/ Languages	[Pentland 1995] [Malone *et al.* 1999] [Chung *et al.* 2002] [Chou 2002] [Jun and Suh 2002]	Activities and their arrangements can be specified with a standard language	• Repositories of standard processes are valuable • Process "unfolds" as a function of information requirements and results	• Knowledge management • Process data interchange • *Process space* description • Process planning	• Activities • Data deliverables • Production rules

[a]Other process modeling frameworks (notably for business process modeling) exist which have not been applied to PD in the literature. These are not all included in the table. Most of the included frameworks have also been applied outside of PD. GPR: generalized precedence relation; GERT: graphical evaluation and review technique; SADT: structured analysis and design technique; IPO: input-process-output; SIPOC: supplier-IPO-customer; ETVX: entry-task-validations-exit.

[b]Here we consider only a small subset of system dynamics (see System Dynamics row) modeling instances that decompose the PD process into a network of discrete activities.

them to focus their creativity and innovation on value-adding activities instead of on circumventing dissonant systems, searching for information, and cleaning up problems caused by poor assumptions. Thus, counter-intuitively for some, process models actually *enable* creativity and innovation to flourish: an appropriate amount of structure is needed to focus innovation on the most valuable problems instead of letting it "reinvent the wheel." Processes are the nexus of the five project systems in Figure 3. A process model provides a powerful fulcrum for leveraging a project, a program, or an enterprise.

Finally, if we buy an argument that many managers do not know as much as they would like about what they are doing, and, furthermore, that they are afraid to admit this, then we find another key reason for a lack of support for process modeling. A model lays out the key assumptions and mental models of the modelers, paving the way for others to compare with their own mental models and disagree. Some managers prefer to manage only on the basis of their hidden mental models, thereby not truly empowering their team. If this is common, then we again see an important gap in managerial education and perception. Process models provide the key to sharing assumptions, understanding the areas of project uncertainty and ambiguity, managing commitments and accountability, and completing the project on schedule, within budget, to specifications, and with minimal surprises—every manager's goals. Many enlightened managers realize this, which is why we remain optimistic.

3. KEY CONCEPTS

Having established the importance of building useful process models that treat the process as a system, we now turn to discuss several key concepts that provide an important background and foundation for PD process modeling.

3.1. Distinguishing PD Processes from Many Other Business Processes

Business and manufacturing process modeling are large fields with substantial literature on modeling frameworks and techniques. However, project processes in general and PD processes in particular are unlike these conventional processes in a number of significant ways. Innumerable modeling efforts have failed or fallen far short of their potential because of failing to realize this. First, as we mentioned in the Introduction, most business processes, such as manufacturing, order processing, and purchasing, strive to do the same thing repeatedly, whereas projects such as PD seek to do something new, once. Second, the outputs of activities in most business processes can be verified immediately (e.g., the part has been made to specifications or not), whereas the outputs of many PD activities, such as information, cannot be verified until much later. Third, while some business processes are largely functional (e.g., manufacturing or sales), PD is most properly a multidisciplinary endeavor, one that spawns many interdependencies among activities. In fact, PD processes are better thought of as networks or webs instead of as chains [Negele, Fricke, and Igenbergs, 1997]—popular parlance (e.g., "Critical Chain" [Goldratt, 1997]) notwithstanding. Fourth, conventional business processes tend to be more sequential, driven by firm dependencies on specific materials and data, whereas PD processes tend to be more parallel (think Concurrent Engineering), since dependencies on information are "softer" in the sense that they can be replaced (for better or worse) by assumptions and early estimates. Fifth, the dependencies in conventional business processes tend to be clearer than those in PD processes; communication tends to be directed to known customers and suppliers. This is less true in PD, because a number of assumptions and interactions tend to be undocumented, and because of greater ambiguity in the required actions and interactions. Because of this "softness," PD processes need to be more flexible and agile than their relatively fixed counterparts.[12] Since creativity and innovation are not linear, PD processes are constantly changing based on the state of the project—adding activities and interactions, eliminating them, and iterating. Meanwhile, conventional business processes are less ambiguous (easier to define), more rigid, and easier to "reengineer" towards optimality in some dimension. Uncertainty, ambiguity, and risk are higher in PD [e.g., Schrader, Riggs, and Smith, 1993; De Meyer, Loch, and Pich, 2002], and most organizations tolerate higher risk in PD than they do in manufacturing or customer-service processes. (Nevertheless, companies take many unnecessary risks in PD; a useful process model can decrease them.) All of these differences between conventional business processes and PD processes have important implications for the modeling framework used and the approach taken to building the model.[13]

[12]However, the most flexible and agile project does not have "no process." While too much structure can be constraining, too little structure causes project participants to spend their time creating it, "reinventing wheels" and otherwise wasting effort on non-value-adding activity. Too much process discipline is bad, as is too little—despite some affectionately calling such a state "agility" [Boehm and Turner, 2003]. We hypothesize an optimal amount of process structure, above or below which the project will lose value. We also hypothesize that this point will move according to a project's *uncertainty profile* [De Meyer, Loch, and Pich, 2002].

[13]Fricke et al. [1998] catalog some additional differences between PD and other business processes.

3.2. Descriptive Versus Prescriptive Process Models

A process model can be descriptive, prescriptive, or have aspects of both. A *descriptive* process model attempts to capture tacit knowledge about how work is really done. It tries to describe key features of the "as is" reality. It is built inductively. On the other hand, a *prescriptive* process model tells people what work to do and perhaps also how to do it. It is built deductively, perhaps drawing from an external standard and/or documentation from other projects. A prescriptive process is a standard process or procedure accompanied by a mandate to follow it exactly. Prescriptive processes become more appropriate as the work becomes more repetitive—e.g., as we move from PD projects to the conventional business processes described above. Many process models share some descriptive and prescriptive characteristics.

Of course, prescribing a process in the wrong environment is dangerous. Before becoming prescriptive, a process model should accumulate enough information, learning, and accuracy to ensure its feasibility and effectiveness. Even then, the PD environment is dynamic enough that no process model will become or remain complete and accurate in all aspects. In this paper, we focus on building descriptive process models that will help us understand what and how work is done—but that *may not have to be followed exactly* on a project. Having a shared, agreed-to representation of an approach and a network of commitments (discussed below) that is known to have worked in a somewhat similar situation is an invaluable aid to project planning and execution. The process details are also what empower the workforce to manage themselves. It is healthier for an organization to gradually evolve portions of a descriptive process model into a prescriptive one rather than to pull a prescriptive process out of the air (or an industry standard).

Are canonical process models such as the Spiral development model [Boehm, 2000] and the common SE "Vee" model [e.g., Forsberg, Mooz, and Cotterman, 2000] descriptive or prescriptive? They contain aspects of both. They are high-level, general descriptions of PD processes, and many enterprises use them as guidelines for their prescriptive processes. Enterprises may develop tiered process models, where the high levels are prescriptive and the lower levels are descriptive. Unfortunately, some companies fail to realize the distinction, thereby causing internal misunderstandings and debates about what process documentation is formal, auditable, etc. The resulting waste in time and effort in large organizations should not be underestimated.

Toyota seems to have a mature perspective on process models, treating them as the company's repository of state-of-the-art knowledge about how to do work, but constantly subjecting them to the scientific method [Spear and Bowen, 1999]. Each process is described in great detail, and each activity and interaction description is *hypothesized* to be the best way of getting the desired result. Hence, it is descriptive *and* prescriptive. The workforce has a common understanding of what is planned to be done and spends minimal time wondering how to do it. Clearly, a lot of effort is invested in planning: Rather than hoping that "we will figure out a way to do that when we get there," they explicitly develop a proposed approach, a hypothesis. Given this huge investment in detailing processes, one might think that Toyota would be highly resistant to change. Paradoxically, just the opposite is true. Each hypothesis is viewed as waiting to be disproved. A better way is always welcome, and there is a clear and quick mechanism for changing a process, which then becomes the new hypothesis for the best way to do the work. This approach promotes consistency; it also provides a ready baseline against which to compare a proposed change. Moreover, since they do not have to think so much about what work to do, the employees are freed to think about better ways to do the work. While used primarily in the Toyota Production System, this philosophy is also used in *projects* at Toyota.[14] It provides the basis for a true "learning organization."

As former U.S. President Dwight G. Eisenhower said, "The plan is nothing; planning is everything." It is impossible to pre-specify all actions and interactions in a PD project—i.e., to have a perfect plan. On the other hand, it is very inappropriate to make no attempt to do so—i.e., not to do enough planning. Planning involves systematic learning about what is known and unknown, what is certain and uncertain [de Geus 1988; De Meyer, Loch, and Pich, 2002]. More than just "documenting," modeling is the act of sorting out what is known and unknown and can lead to discovery of the "unknown unknowns." Many of the so-called "unk unks" in a

[14]Spear and Bowen's [1999] four rules of the Toyota Production System are highly instructive for (but not necessarily applicable to the same extent in) PD process modeling, and we will allude to these rules in other discussions:

1. All work shall be highly specified as to content, sequence, timing, and outcome.
2. Every customer-supplier connection must be direct, and there must be an unambiguous yes-or-no way to send requests and receive responses.
3. The pathway for every product and service must be simple and direct.
4. Any improvement must be made in accordance with the scientific method, under the guidance of a teacher, at the lowest possible level in the organization.

project are actually known by *someone*, but they nevertheless surprise project management because they were not accounted for in the project plans. Building a shared process model helps expose such latent information. *Having a prescribed way of doing something* needs to be reconsidered in the context of PD as meaning *having a hypothesized way of doing something*. This implies that "changing the plan" should be easier than it often is in many organizations.

3.3. Activity Dependencies and Information Flow

What causes one activity to depend on another? Where do the connectors on a flowchart or PERT chart come from? In many modeling and scheduling situations, activity dependencies are defined casually, as an individual or group sits down and thinks, "This should happen before that does." Even if the approach is more systematic, often only a nominal set of dependencies are recorded—just enough to connect all the boxes on the flowchart or Gantt chart and perhaps find a critical path. Thereafter, however, all of the dependencies so derived may be treated as completely firm! But where do dependencies among activities in PD projects really come from?

In PD, activities require information and perhaps other inputs to do their work and produce satisfactory results. If one actually asks the people who do them, one will find that most PD activities use a *set* of inputs rather than a single one.[15] Similarly, one will find that activities produce much more than their primary deliverable: they produce preliminary outputs, status reports, confirmations and/or rejections of other activities' propositions, etc. PD and its activities can be viewed as a process of information collection, creation, interpretation, transformation, and transfer.[16] PD activities require and produce information. The result of many PD activities is just information. Information is what flows; it is the life-blood of projects. Thus, to find the dependencies between activities, we need to find what information and other deliverables they require to do their work. Unfortunately, most process modeling fails to capture the full information flow, and the noted interactions are often poorly understood—e.g., when the boxes on a flowchart are labeled but the connectors are not. Activities also require quality inputs to produce quality outputs. A 100% value-adding activity (if such exists) will still produce garbage if fed garbage [Browning, 2002].

In information-intensive processes like PD, where activities depend on a number of inputs and provide a number of outputs, a casual approach to process modeling is insufficient. Since poor flow is the source of many process problems, the dependencies that establish the flow patterns must be recognized and managed. At the outset of building a descriptive process model, one is unlikely to capture all of the dependencies. Over the course of a dynamic project, dependencies may appear, change, and disappear. However, it is interesting to note that the dependencies tend to be the *more stable* part of a process model. That is, while the way an activity is done may vary from project to project, the activity's need for particular inputs (information, approvals, results of decisions, etc.) is more consistent. (For this reason, high-level program plans will often focus on the deliverables desired at certain milestones.)

Several authors [e.g., Winograd and Flores, 1986; Pall, 1999] consider processes as "networks of commitments," where each dependency represents a deliverable that must be agreed and committed to. The process model then provides the basis for accountability throughout an organization. (These commitments can change if necessary, but without a baseline set of commitments, there is nothing to change.) With this perspective, dependencies take on an even greater importance, and managers realize the impetus to manage *interactions* vice actions.[17]

In discussing the flow of information and other deliverables that create dependencies, we distinguish information from knowledge. Knowledge is required to do non-automated work, but information is what flows between activities done by different people (Fig. 6). There can be problems turning information into the right knowledge and turning knowledge into the right information. An individual's mental models play a role in both transformations, and a process model is useful when it helps align peoples' mental models and facilitates communication.

Note: As we will discuss below, we distinguish inputs from resources, although the latter are technically a kind of input. By *resources*, we refer to time, money, energy, enthusiasm, and other consumables used in the course of executing the activity. Within the category of resources, we also prefer to distinguish consumables from tools and equipment, which include

[15]Many PD activities can proceed based on assumptions about their inputs when the actual inputs are unavailable. This flexibility is a double-edged sword and makes a process model even more valuable for managing a PD project. Even the activities' participants may not recognize the assumptions they make for what they are: proxy inputs.

[16]This perspective is supported by an extensive literature on organizations [e.g., March and Simon, 1993; Burns and Stalker, 1961; Galbraith, 1977; Tushman and Nadler, 1978; Clark and Fujimoto, 1991] and design activities [e.g., Pahl and Beitz, 1995] as information processors.

[17]Some managers refer to this as managing the "white space" between activities or organizational units.

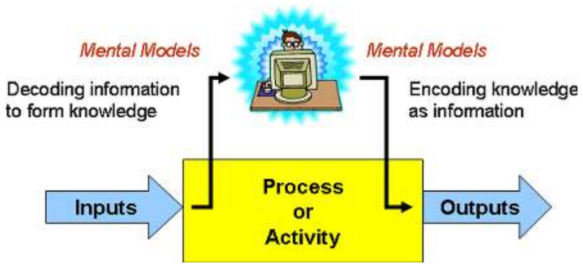

Mental Models
Mental Models
Decoding information to form knowledge
Encoding knowledge as information
Inputs
Process or Activity
Outputs

Figure 6. Distinguishing knowledge and information.

machines, hardware, software, templates, etc. required to perform the activity. It can also be helpful to distinguish facilities or work areas in this category.

3.4. General Objectives for PD Process Modeling

The discussion so far points to some general objectives for a PD process model, as given in Table IV. Unfortunately, the text-based, narrative process documentation found in many companies does not meet these objectives; in fact, it turns out to be very unhelpful except for passing audits. Nor do mere process flowcharts or maps fit the bill entirely. A more capable process modeling framework is needed, one that is simple but extensible to capture rich content where appropriate.

3.5. Foundations of a Generalized Framework for Modeling Processes: Activities and Deliverables

These objectives point to the need for a generic, flexible basis for structuring a process model. Such a framework would begin with two fundamental objects: *activities* (variously referred to as processes, process elements, subprocesses, tasks, steps, operations, functions, etc.) and *deliverables* (also known as inputs, outputs, results, work products, services, information, outcomes, artifacts, items, etc.). Activities are the constituent elements of a process system. They are packages of work to be done to produce results, the "boxes" on a flowchart. They require deliverables as inputs and produce them as outputs. Hence, they are both a customer and a supplier. They consume or use resources: time, money, people, tools, facilities, etc. Deliverables are the connectors on a flowchart that represent any information, data, result, material, etc. produced or required by an activity. On the basis of these two fundamental objects, we propose a generalized framework in Section 4.

Table IV. General Objectives for a PD Process Model

- *Elements*: represent the variety of activity attributes required to support the spectrum of model purposes
- *Relationships*: represent meaningful and varied relationships between activities
- *Maintenance*: be quick and easy to change and update, where appropriate, by almost everyone in the workforce (thus implying appropriate security features)
- *Computerization*: enable computer-based model building, storage, analysis, and presentation
- *Views*: enable visualization and comprehension by varied users from different perspectives; support varied but related views
- *Consistency*: provide a consistent representation of all relevant information in a formal structure
- *Planning*: support project planning, including process tailoring and activity selection, staffing, resource loading, budgeting, and scheduling
- *Empowerment*: enable project visualization, communication, and informed decision making at all levels
- *Adaptation*: support process agility and adaptation
- *Integration*: integrate easily with other process models in other parts of the organization and with other activity-based cost, schedule, and risk models in the organization
- *Simplicity* and *Expandability*: built of simple elements that can collectively model more complex processes; object-oriented; holonic
- *Improvement*: include allowances for improvement, particularly in the form of an improvement loop
- *Error detection*: automatically check for and flag integration problems and missing information, or provide assistance in this regard

3.6. Standard and Deployed Processes

The multiproject company struggles with the issue of commonality across projects, both in terms of organization structure (to facilitate personnel transfer and career development), tools (to leverage investments, training, and infrastructure compatibility), product architectures, and, not in the least, processes. Some organizations define a *standard process* to be used by all projects—i.e., a standard set of activities and deliverables, usually at a high level, although sometimes also at lower levels in particular areas. Typically, this standard process requires some amount of tailoring and/or scaling before it is helpful for planning and controlling a particular project instance. Such a modification of a standard process is termed a *deployed process*.

We provided a fuller discussion of the use of standard and deployed processes in part four of our tutorial (q.v., Fig. 1). Here, we will only note some of the difficulties these pose for modeling. First, because they are not used verbatim on any project, many organizations' standard processes tend to be detached from the way work is actually done. Many of those doing so-called "real work" may see the standard process as irrelevant, too generic to be helpful. And process modelers struggle with the right level of standardization when faced with clear differences in projects' ways of working. Second, company policy may attempt to mandate the use of a standard process in a prescriptive way. Third, and also because of its prescriptive nature, there is often pressure for it to be purposefully ambiguous, so that any activity done on a project will conveniently fall under its umbrella whenever, for example, an ISO auditor questions the workforce. These difficulties challenge the establishment and maintenance of standard and deployed processes. Thus, even though standard and deployed processes must be developed progressively and iteratively, as part of an organization's learning cycle, we generally recommend that an organization with existing projects build *deployed* process models *first*, based on the ways work is actually done on current projects. Then, a central organization can distill a standard process and strive to make it increasingly common (and improving) over time. This in turn facilitates efforts towards commonality in the tools system, where companies prefer to drive their various projects towards common information technology platforms, software applications, templates, etc. (Driving towards a common tool set should *follow* process standardization, however, rather than lead it, because the point is for the chosen tools to support the valued activities, rather than the other way around.)

3.7. Centralized Versus Decentralized Process Modeling

The discussion of standard and deployed processes leads to the question of *who* in a company should build a process model. It is important that the people who currently do the work be involved as much as possible since they tend to have the most reliable knowledge of it. On the other hand, having a centralized organization or a team get together in a conference room to "hammer it out" has some advantages. For one thing, these tend to be process modeling experts who can build sophisticated models. Yet, they suffer the major disadvantage of being removed from where the work is done—and the accompanying "wish-was" problem mentioned above. Furthermore, collecting and verifying the data comprising the process model will overwhelm a small group. Thankfully, the generalized framework for modeling processes that we present in Section 4 lends itself to distributing the modeling effort throughout the organization. Each person can contribute a little of the information in their area of specialization, and a centralized group can oversee its integration and verification. The centralized group will often specify the "trunk" of the enterprise's process "tree" structure but allow those currently doing the work to specify the twigs and leaves. To facilitate a decentralized approach to building PD process models, Fricke et al. [1998] and Negele et al. [1999] developed an input-process-output (IPO) database and integration tool called TIPO ("Tool for IPO"). Sabbaghian, Eppinger, and Murman [1998] also developed a Web-based tool to facilitate distributed process modeling.

3.8. "As Is" Versus "To Be" Process Modeling

As people build a process model, what should they describe—the way work is done now or the way they think it should be? Should they describe the present or prescribe the future? If there is an "obvious" way to do things better, is it a waste of time to document the current way? In practice, process modeling typically proceeds as a combination of both "as is" and "to be." For a process of significant size, it will take some period of time to build the process model, and during that time the enterprise will evolve and the process will change. The point of building the process model is to establish a baseline to the extent possible. Of course, one of the key requirements of a useful process model is that it be amenable to quick and easy maintenance and improvement.

Having the people who currently do the work contributing to building the model usually provides the best mix of current and desired practice in the baseline model. Major improvements (deviations from the current process) are best left for a second, proposed, "to be" version of the model, the value of which can be analyzed and compared to the "as is" state.

3.9. Dealing with Multiple Phases

PD occurs over several phases, such as conceptual design, preliminary design, and detailed design. As illustrated in spiral models [e.g., Boehm, 2000], each of these phases is somewhat similar, though successively more detailed. Therefore, does each phase need to be modeled separately? Can some of the modeling structure from one phase be reused for another?

We have found that, while the structure of each phase is similar at a very high level, the nature of the specific activities performed and the information required differs, especially in its attributes (such as duration, cost, maturity, entry and exit criteria, etc.). Hence, we recommend that each phase be approached separately at first, maximizing the inductive nature of the model building. Thereafter, efforts can and should be made to consolidate and standardize across phases to a reasonable extent, perhaps using varied *activity modes* (defined below). Having separate models of each phase to start with provides a basis for comparison, contrast, and validation in the model building process. Best practices from one phase can be used in the others, instead of "who-knows-how-good" practices from one phase being foisted on the others.

It is important to emphasize the reason for multiple phases or stages in uncertain, ambiguous, and risky projects like PD. By breaking a large, risky project or program down into smaller, chronological chunks, resource commitments can be delayed and reevaluated at phase-gate reviews, thereby reducing the overall risk by providing opportunities to terminate the project early if unsatisfactory scenarios materialize. The early phases are typified by fewer resources and much higher ambiguity. Activities in these phases are geared towards generating and discovering information and reducing ambiguity and risk, and the activities are much more likely to change drastically based on the results that appear. The downstream, detailed design phases obviously have a different emphasis; what needs to be done is much clearer by that point. Hence, the different phases of PD are best addressed by separate (but integratable) process models, and care should be taken to distinguish the similar activities in each phase accordingly.

3.10. Dealing with Unforeseen Uncertainty and Ambiguity

In their survey of PD process models, Browning and Ramasesh [2005] note the following two themes and call for further research to address them. First, most process models assume that all PD activities are known *a priori*. De Meyer, Loch, and Pich [2002] classify projects by their type and amount of uncertainty—from "variation," "foreseen uncertainty," and "unforeseen uncertainty" to "chaos"—and suggest that the approach to project management vary accordingly. The approach to process *modeling* should also vary [Lillrank, 2002]. While the current models are most applicable in cases of "variation" and "foreseen uncertainty"—where over 90% of activities and deliverables can be anticipated from one project to the next [Austin et al., 2000]—research is needed on process models to support project planning in cases of "unforeseen uncertainty" and "chaos." In these latter cases, the process should explicitly include activities to *reduce risks* and *discover opportunities*. Natural, adaptive, or emergent processes [Highsmith, 2000; Lévárdy and Browning, 2005] such as *bazaar-style development* [Raymond, 2001] offer promising avenues for further research.

Second, as standard processes become more widely used as a basis for project planning and organizational learning, research on tailoring and scaling a standard process for a particular project takes on greater importance. The existing frameworks and models do not capture the difference between process description and prescription, nor do they provide much guidance for process tailoring and scaling. When is the full, standard process really appropriate, and when might a fast-track, albeit higher-risk PD process make more sense? While pure, mechanistic design of innovative organizations does not work [Dougherty, 2001], an appropriate amount of planned process structure yields efficiency [Tatikonda and Rosenthal, 2000; Spear and Bowen, 1999] by enabling workers to focus their creativity on value-adding actions and coordinate their interactions. Austin et al. [2001] show that some basic process structure is appropriate even for conceptual design. Therefore, what is the right balance between process prescription and innovation [Benner and Tushman, 2003]? As flexible, adaptive processes become more attractive in climates of high uncertainty and ambiguity, how are these to be distinguished from what some managers say is the most adaptable process of all—"no process"?

These are important questions that bear much further scrutiny. However, it seems to be clear that both extremes—assuming one knows all activities and deliverables *a priori*, on one hand, and thinking that trying to model and plan what one does anticipate is a total waste of time, on the other—are problematic. The generalized framework that we propose in this paper is especially useful between these extremes. While some may attempt to use it in the first extreme, it is important to realize that all proposed activities, deliverables, and commitments represent a hypothesized baseline, a plan, and that it is the act of creating and updating them that is valuable, much more so than seeking to use the single, original version to keep a project "on track" (as discussed in Section 3.2). As a key benefit, the generalized framework enables (and even admonishes) capturing much of the information needed to re-plan rapidly and effectively.

3.11. Some "People" Issues in Process Modeling

"Here is Edward Bear, coming downstairs now, bump, bump, bump, on the back of his head, behind Christopher Robin. It is, as far as he knows, the only way of coming downstairs, but sometimes he feels that there really is another way, if only he could stop bumping for a moment and think of it."

—A.A. Milne, *Winnie-the-Pooh*, opening lines

Some people resist process modeling because they:

- Have witnessed a lack of benefit from previous process modeling efforts;
- Perceive a lack of resources;
- See current crises as more urgent and important;
- Do not understand the uses, value, and benefits of process models;
- Anticipate and fear control, over-prescription, over-systemization, reduction of creativity, and stifling of innovation;
- Are reluctant to share knowledge and collaborate;
- Have grown accustomed to "hiding" in bureaucracy and are averse to transparency;
- Prefer to work "harder" instead of "smarter"; or
- A combination of these reasons.

Arnold [2004] discusses some other inhibitors to process modeling efforts. In response to these problems, we offer the following as potential solutions:

- Starting small, demonstrating the benefits, and then expanding;
- Reprioritizing and reallocating existing resources when it is not possible to add resources;
- Education and training on the motivations for and benefits of process modeling;
- Communication that processes do not eliminate innovation but channel it towards critical problems and collaborations and the results customers value;
- Admonition to view process models as hypotheses of the best courses of action [Spear and Bowen, 1999], ready to be rejected when a better solution can be found[18];
- Incentive systems for sharing knowledge, collaborating, and improving; and
- Executive support!

Notwithstanding the last item, the most important of these is education. Unless people understand "why" and appreciate the value, they will act as barriers instead of enablers to process modeling success. Often, the critics of process modeling can point to one or two particular experiences to justify their opposition. For instance, perhaps, in an effort to "get things done," their organization isolated a cadre of individuals to work on a process model (cf. the centralized approach to modeling discussed in Section 3.7). When presented to the rest of the organization, the results of this work were not actively accepted, as they did not understand why the model was built the way it was. (This "short cut" [or "short circuit"] approach to process modeling and its less than stellar results is a major contributor to the poor reception of process models in many circles.) Or, perhaps they think that their conception of a process model (perhaps as a long narrative) would not help them do their work. Whatever the issues, it is extremely important to find the most vocal opponents of process modeling and strive to educate them about its benefits. Removing a barrier to success is often much more effective than increasing the drivers of success (because the barriers will also increase their resistance proportionally), which is why we favor education even over executive support. As an excellent explanation of the value of spending precious company resources on process improvement and learning vice the "crisis *de jour*," we recommend a paper by Repenning and Sterman [2001]. Finally, it is crucial that everyone involved in process modeling receive some benefit commensurate with the contribution they make.

3.12. Definitions

We conclude this section on key concepts by providing our recommended definitions for some process modeling terms in Table V.

[18] Sometimes lazy, creative people find helpful short cuts, but often they must be kept to following a best practice. Give them the opportunity to find a better way, but make them prove it is better, not just in terms of efficiency, but in terms of effectiveness and consistency and in light of all the potential failure modes that the baseline process is geared towards preventing.

Table V. Definitions of Some Process Modeling Terms[a]

Process	(1) An organized group of related **activities** that work together to create a result of value (2) A network of customer-supplier relationships and commitments that drive work **activities** to produce results of value Named using a verb and a direct object: e.g., "Design the Wing"
Activity (or Task, Step, Sub-Process, etc.)	A unit of work defined by its inputs, outputs, resources used, and potentially other attributes as listed in Table 6; an element of a **process** (but from another perspective a process itself) Often, the terms "process" and "activity" are used relative to each other, in a hierarchical sense, where "process" refers to a large unit of work and "activity" refers to the larger unit's constituent units. Named using a verb and a direct object: e.g., "Design the Wing"
Process Element	A generic reference to a unit of work; any **process**, **activity**, task, etc.; it can be made up of other process elements, and it can be part of a larger process element
Input	A **deliverable** from a supplier **process element** used by a consumer **process element**
Output	A **deliverable** resulting from a **process element**'s work; should provide value to the consumer
Deliverable	Any **input** or an **output** to a **process element**; includes products, services, authorizations, parts, assemblies, information, software, equipment, drawings, tools, plans, consulting, support, installations, deliveries, repairs, results of decisions, etc. Named using a noun and any necessary qualifiers (adjectives, adverbs): e.g., "Avionics Requirements"
Flow	The pattern of **input-output** relationships (transmission of **deliverables**) and work performed among **process elements** that specifies their execution sequence
Activity Mode	A particular version of an activity with different attributes (**inputs**, **outputs**, **entry criteria**, cost, duration, etc.)
Entry Criteria	The required performance level or maturity of the **inputs** in order to perform the activity (in a given mode)
Exit Criteria	The required performance level or maturity of the **outputs** in order to finish the activity successfully (in a given mode)
Parent Process	The next-larger **process element** of which this **process element** is a part (or **child** in the hierarchy)
Child Process	The next-smaller **process element** of which this **process element** is a **parent**
Organizational Unit or Element	A generic reference to a human resource or grouping thereof that performs work (executes a **process element**); any person, group, team, organization, council, board, committee, etc.
Process Description, Map, or Model	A symbolic or abstract representation (textual, graphical, symbolical, etc.) of a **process**; includes a statement of both the constituent **process elements** and their relationships

[a]Notes for Process item: (1) Hammer [2001]; (2) Gabriel Pall, personal communication; based on Pall [1999]

4. A GENERALIZED FRAMEWORK FOR MODELING PD PROCESSES

The PD process modeling literature seems to struggle with regard to the right amount of standardization. (Recall Table III, which lists 18 PD process modeling frameworks.) While a diversity of frameworks places no reins on innovation, reinvention seems rampant in the literature, as extended models in one framework merely incorporate attributes that have already been accounted for in another framework. This diversity also causes problems in practice, as models for different purposes are not easily integrated or understood across functions. Our observations of the uses of PD process models and the key concepts discussed above suggest possibilities for and advantages of a more general approach. Hence, we propose a synthesis in the form of a generalized framework (GF) for modeling PD processes. A GF would be helpful and useful, without significant drawbacks, and is validated by similar efforts outside of PD.

As discussed in Section 3.5, two primary *objects*,[19] actions and interactions—i.e., *activities* and *deliver-*

[19]Here, we discuss process modeling using some terminology from the object-oriented (o-o) modeling literature. o-o modeling is a robust modeling paradigm with high structural correspondence to peoples' mental models of reality [e.g., Kilov, Rumpe, and Simmonds, 1999]. o-o modeling and its popular Unified Modeling Language (UML) are used extensively for designing product systems [e.g., Crisp et al., 2000; Senin, Wallace, and Borland, 2003].

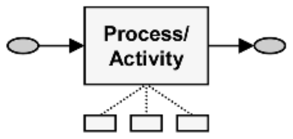
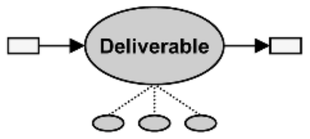
ables—provide the foundation for a PD process model. Each of these objects can have a myriad of attributes; the 18 frameworks in Table III each account for and emphasize certain ones. For example, traditional activity network models emphasize the activity attributes of duration, cost, and predecessors, but they do not address the attributes of the deliverables. From the "purpose perspective" [Browning and Ramasesh, 2005], while no single purpose requires a model with all of the attributes, each purpose requires a subset of them and often can benefit from accounting for more of them. (Again, research has often moved in the direction of enriching models for one purpose by accounting for additional attributes that have already been emphasized in models for other purposes.)

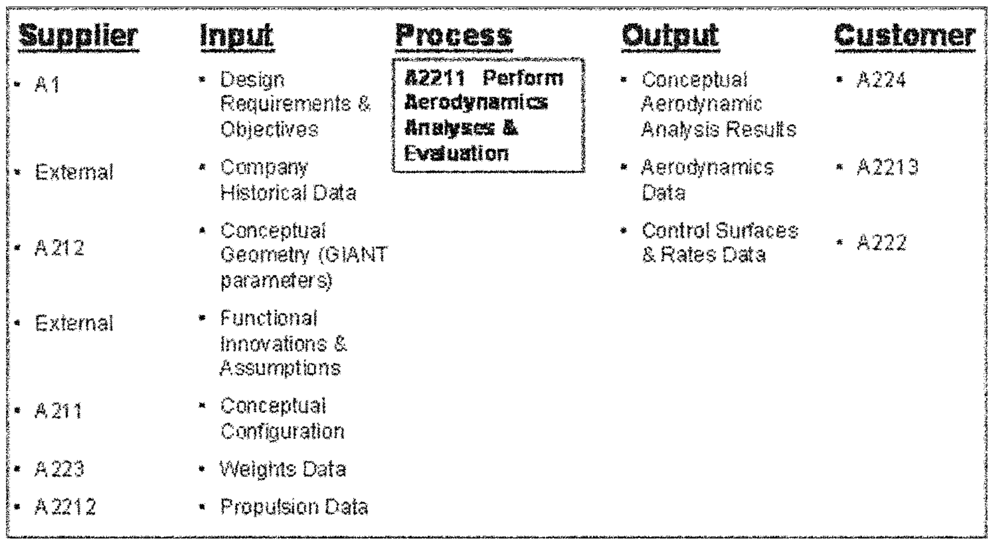
Activities and deliverables provide the kernels or basic "Lego® blocks" from which increasingly complex process structures can be built. Table VI shows each with a rich (but not exhaustive) superset of their attributes. These are used to "tag" relevant information (or links to information) to the object, especially in three ways: (i) An activity's input and output attributes provide one or more pointers to unique deliverable objects (and vice-versa); (ii) an activity's (deliverable's) parent and child attributes provide pointers to unique activity (deliverable) objects, thereby specifying hierarchical (actually, holonic) work and deliverable breakdown structures; (iii) other attributes can optionally point to *other kinds of objects*—such as organization units (e.g., individuals, teams), events, issues, policies, goals, milestones, "toll gates," resources, and tools—representing elements of the other four project systems in Figure 3. No single model need utilize all of the attributes, but having a structure to accommodate them facilitates process model integration. This potentially very rich model is stored in a database, independent of the many views that could be used to represent aspects of it. A single view is inadequate to convey all of its objects and attributes in a visually appealing way, and hence a variety of views should be used, as discussed in relation to Figure 4 and in Browning and Ramamesh [2005].

At a minimum, to form the most basic process model, each activity must list its inputs and outputs and thus be linked to the intervening deliverable objects. Hence, process modeling sometimes begins with an input-process-output (IPO) or supplier-IPO-customer (SIPOC) representation, such as the example in Figure 7. However, the SIPOC diagram covers only the bare minimum of activity attributes needed to get a process model started.

Beyond the two fundamental objects, a process model can include other objects such as:

Table VI. Fundamental Building Blocks of PD Process Models and Some of Their Attributes (Adapted from Browning [2002], Fricke et al. [1998], and Negele et al. [1999])

	
Process/Activity Object Attributes	**Deliverable Object Attributes**
• Name	• Name
• Description	• Description
• Mode	• Mode
• Inputs (and sources, timing, sensitivity)	• Supplier(s), with agreement and commitment from each
• Outputs (and sinks, timing)	• Consumer(s), with agreement and commitment from each
• Parent process	• Parent deliverable
• Constituent activities ("children")	• Constituent deliverables ("children")
• Metrics (duration, cost, uncertainty, risk, criticality, switching cost, budget, deadline, capability, robustness, flexibility, productivity, percent complete, improvement curve, etc.)	• Requirements (what is agreed and committed to)
• Resources (tools, facilities, etc.)—pointers to tool system	• Constraints (regulations, policies, etc.)
• Roles (training, skills, assignments, skill gaps)—pointers to organization system	• Verification and Validation criteria
• Organizational owner; assigned resources	• Metrics (status, confidence, maturity, quality, value added, due date, time until ready, lateness penalty, complexity, time last modified, percent complete, etc.)
• Entry criteria	• Format
• Exit criteria	• Medium of transmission
• Work policies, business rules, design rules [Baldwin and Clark 2000]	• Pointer to actual artifact/data (perhaps in product system)
• etc.	• etc.



Supplier	Input	Process	Output	Customer
• A1	• Design Requirements & Objectives	A2211 Perform Aerodynamics Analyses & Evaluation	• Conceptual Aerodynamic Analysis Results	• A224
• External	• Company Historical Data		• Aerodynamics Data	• A2213
• A212	• Conceptual Geometry (GIANT parameters)		• Control Surfaces & Rates Data	• A222
• External	• Functional Innovations & Assumptions			
• A211	• Conceptual Configuration			
• A223	• Weights Data			
• A2212	• Propulsion Data			

Figure 7. Example SIPOC diagram.[21]

- Organizational units (person, team, company, etc.)
- Tools (facility, template, computer system, software application, etc.)
- Product elements (sub-system, component, etc.)
- Goals (requirement, objective, policy, etc.)

Each of these objects would have its own set of attributes. Collectively, these objects and their attributes provide a sufficient basis to model and link the five project systems shown in Figure 3. Other objects and attributes may be added to model other systems or aspects thereof.

Whenever an attribute of one of these objects refers to another object (such as when an activity has a list of input deliverables as one of its attributes), this reference is captured simply as a pointer or link to the referent object itself, which stores the actual information. The objects are holonic and can share other convenient properties of object-oriented modeling frameworks, such as full or selective inheritance. In addition, a parent process element with constituent activities can have one or more attributes (meta-data) to represent holistic behaviors not captured collectively by the constituent activities. Thus, the model can represent much more than a reductionist perspective.

As the marketing literature models a product as a vector of attributes, Table VI enables processes to be treated the same way. The GF can be tailored for a particular use via a Quality Function Deployment (QFD)-like mapping of purposes (“whats”) to attributes (“hows”). Applications to new purposes would require the addition of new attributes. Thus, the GF theory enables new research to identify the attributes of process models that will make consistent and valid descriptions of reality and have predictive value for a given purpose.

A GF offers many other benefits. It raises attentiveness to attributes that would be ignored from the perspective of one framework, such as the importance of deliverables’ attributes. Thus, one would expect the GF to improve model validity, accuracy, resolution, and ability to represent causality [Hazelrigg, 1999]. By accounting for work *states* as well as workflows, the GF facilitates modeling adaptive processes and their emergent characteristics [Highsmith, 2000]. As new activities are discovered in an ambiguous project, they are easily inserted into the GF model of the network, which adapts quickly to their presence and indicates any changes and their effects.[20] The object-oriented view of activities and their mapping to organizational objects also facilitates agent-based modeling and analysis approaches.

Moving towards a GF provides further advantages, especially for practitioners. In practice, models built for a specific purpose but using different frameworks, such

[20]Like the integration of a new information technology system into an existing company infrastructure, the effective integration of a new activity into a project can cost more than the activity itself. The way to break this paradigm is to “develop systems and process that are self-integrating—that are able to “understand” their own capabilities, limitations, input needs, and output abilities; and which are able to “negotiate” their relationship with the rest of the enterprise’s systems when they are plugged in and given their performance specifications” [IMTI, 2000: 27].

[21]The data for this example come from an unmanned combat aerial vehicle (UCAV) PD process at The Boeing Company, as described in Browning [1998].

as a value stream map [e.g., McManus and Millard, 2002] and a signposting model [Clarkson and Hamilton, 2000], could be integrated into a single, rich model, which could furthermore be abstracted for any particular user or purpose by filtering out the irrelevant attributes in a customized view. Meanwhile, each user and view would benefit from querying the full, common, rich model instead of a single, disparate, shallow model. While a model built in one of the 18 frameworks would make implicit assumptions about attributes not normally accounted for by the framework, in a GF these assumptions are explicit or replaced by real information. Improved process model integration enables increased integration of real processes, which in turn enables improved organizational, tool, and product integration. Furthermore, by including attributes for "lessons learned" and "potential pitfalls" in the activity objects, and by generally capturing information about what and how work is done, the process model becomes a skeleton for organizational knowledge. By enabling an organization to structure and reuse their knowledge about what and how work is done, a GF facilitates process-model-based project planning and tailoring. Chiefly, following a GF makes a process model more broadly applicable, integratable, adaptable, usable, reusable, maintainable, and easy to store. Individually, each of these mean real money saved in practice; combined, they can enable a revolutionary approach to enterprise planning and management. A compelling motivation for a GF is provided by trends in other areas, including the broader business process modeling and information technology literature [e.g., Tissot and Crump, 1998; Presley et al., 2001; Scheer, 1998a; Zachman, 1987], efforts to structure process specification languages [e.g., Schlenoff et al., 2000] for manufacturing processes, and the modeling of complex product architectures as a set of views drawn from a complex database of information [DoD 2001; Yu, Harding, and Popplewell, 2000]. These areas portend the needs and desires of industry.

Using a GF also benefits the research community by allowing the buildup of a shared library of integratable and reusable process models upon which to test future analysis techniques. With a generic scaffolding, models and databases of process information captured in one research project, using a particular framework (such as Petri nets), could be more readily integrated, reused, reanalyzed, and verified in another project using another framework (such as IDEF3), and any missing data would be highlighted. Thus, it is also possible to view Table VI as the basis for a data interchange format between PD process models. Since PD projects are more complex than manufacturing operations and unique rather than repetitive, the models are potentially richer and more varied. Thus, having a GF is of even greater benefit for PD process modeling. Since collecting the rich data set necessary to build a PD process model is a significant barrier to researchers, the capability to share models in a standard format should accelerate the pace of research.

5. CONCLUSION

The methods and tools used to design and manage complex projects and programs and their work processes have evolved considerably over the last 50 years [Cleland, 2004], from early work on phased program planning and systems engineering in the U.S. Department of Defense in the 1950s, to Concurrent Engineering [Smith, 1997] or IPPD via Integrated Product Teams (IPTs) in the 1980s, to the wider incorporation of decision support tools such as QFD [e.g., Hauser and Clausing, 1988; Akao, 1990], concept selection [e.g., Pugh, 1991], and decision analysis [e.g., Clemen, 1996] in the 1990s. We have also witnessed growth in international standards such as the ISO 9000 series and the CMMI process guidance. However, the advent and use of all of these has not necessarily increased program success in terms of being able to meet goals within a schedule and a budget. For one thing, the variety of and detail within many of today's methods and tools have arguably increased specialization and the division of labor in organizations. Also, the time-compression desired to reach markets and customers more quickly has led to increased concurrency and multi-disciplinary activity, all of which dramatically increases the coordination and integration challenges facing managers and systems engineers. Furthermore, the individuals who have lived through most of the evolution in techniques over the last fifty years have now retired or are retiring, which contributes to "organizational forgetting" [e.g., de Holan, Phillips, and Lawrence, 2004]. Thus, with all of these methods and tools and their disparate users comes an even greater need for managerial and technical leadership support in terms of a systematic approach to integrating and coordinating the work required to achieve a unique result. We see process models as the enabler for such support, as well as providing the scaffolding for knowledge management. While the outlook in this direction is optimistic, much work remains to be done to guide this development in the most effective and efficient ways.

In this paper, we have laid a foundation for process modeling in a systems engineering context. The SE context, or, more broadly, the PD context, differs from that of typical business processes in significant ways.

Hence, process modeling should be tailored to that environment. This paper seeks to begin to fill that gap.

Education and training on the purposes of and approaches to process modeling is an essential starting point. Many process modelers have attempted to "leap into modeling" just as software engineers are notorious for wanting to "leap into coding." It is important to know "why" before doing, which is the reason for a paper focused on the background and motivation for process modeling. Even so, many of the powerful capabilities of process models and the competitive advantages they provide must be left for discussion elsewhere. Such important topics, not addressed in this paper, include building, representing, and using PD process models (q.v., Fig. 1). Within these broad areas, important subtopics include process-model-based management, the process model life cycle, project and program planning and scheduling, process analysis, process improvement, process *model* improvement, roadmapping the evolution of dynamic processes, process tailoring and scaling, measures and metrics, leading indicators, iteration management, and risk management.

REFERENCES

P.S. Adler, A. Mandelbaum, V. Nguyen and E. Schwerer, From project to process management: An empirically-based framework for analyzing product development time, Management Sci 41(3) (1995), 458–484.

R. Ahmadi and R.H. Wang, Managing development risk in product design processes, Oper Res 47(2) (1999), 235–246.

Y. Akao (Editor), Quality function deployment: Integrating customer requirements into product design, Productivity Press, Cambridge, MA, 1990.

A. Arkin, Business Process Modeling Language, BPMI, 2002, www.bpmi.org.

E. Arnold, 10 pitfalls of process development with avoidance solutions, Proc 14th Annu Symp INCOSE, Toulouse, France, 2004.

S. Austin, A. Baldwin, B. Li, and P. Waskett, Integrating design in the project process, Proc ICE: Civil Eng 138(4) (2000), 177–182.

S. Austin, J. Steele, S. Macmillan, P. Kirby, and R. Spence, Mapping the conceptual design activity of interdisciplinary teams, Des Stud 22(3) (2001), 211–232.

C.Y. Baldwin and K.B. Clark, Design rules: The power of modularity, MIT Press, Cambridge, MA, 2000, Vol. 1.

M.J. Benner and M.L. Tushman, Exploitation, exploration, and process management: The productivity dilemma revisited, Acad Management Rev 28(2) (2003), 238–256.

B. Boehm, Spiral development: Experience, principles, and refinements, CMU's Software Engineering Institute, Pittsburgh, PA, 2000.

B. Boehm and R. Turner, Balancing agility and discipline: A guide for the perplexed, Addison-Wesley, New York, 2003.

G.E.P. Box, "Robustness in scientific model building," Robustness in statistics, R.L. Launer and G.N. Wilkinson (Editors), Academic Press, New York, 1979, pp. 201–236.

T.R. Browning, Modeling and analyzing cost, schedule, and performance in complex system product development, Ph.D. Thesis (TMP), Massachusetts Institute of Technology, Cambridge, MA, 1998.

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, Process integration using the design structure matrix, Syst Eng 5(3) (2002), 180–193.

T.R. Browning and R.V. Ramasesh, Modeling the product development process: A survey of the literature, TCU M.J. Neeley School of Business, Working Paper, Fort Worth, TX, Oct. 2005.

T. Burns and G.M. Stalker, The management of innovation, Tavistock, London, 1961.

S.-C. Chou, ProActNet: Modeling processes through activity networks, Int J Software Eng Knowledge Eng 12(5) (2002), 545–580.

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.

K.B. Clark and T. Fujimoto, Product development performance: Strategy, organization, and management in the world auto industry, Harvard Business School Press, Boston, 1991.

P.J. Clarkson and J.R. Hamilton, "Signposting," A parameter-driven task-based model of the design process, Res Eng Des 12(1) (2000), 18–38.

D. Clausing and V. Fey, Effective innovation: The development of winning technologies, ASME Press, New York, 2004.

D.I. Cleland, The evolution of project management, IEEE Trans Eng Management 51(4) (2004), 396–397.

R.T. Clemen, Making hard decisions: An introduction to decision analysis, PWS-Kent, Boston, 1996.

R.G. Cooper, Winning at new products: Accelerating the process from idea to launch, Perseus, Reading, MA, 2001.

H.E. Crisp, N.T. Hoang, C.M. Nguyen, N.E. Karangelen, and D. Britton, An integrated information representation schema for complex human centric systems, Proc 10th Annu Int Symp INCOSE, Minneapolis, 2000, pp. 551–557.

M. Danilovic and T.R. Browning, Managing complex product development projects: Insights from comparing design structure matrices and domain mapping matrices, Jönköping International Business School, Working Paper, Jönköping, Sweden, Jan. 2006.

A.P. de Geus, Planning as learning, Harvard Bus Rev 66 (1988), 70–74.

P.M. de Holan, N. Phillips, and T.B. Lawrence, Managing organizational forgetting, MIT Sloan Management Rev 45(2) (2004), 45–51.

A. De Meyer, C.H. Loch, and M.T. Pich, Managing project uncertainty: from variation to chaos, MIT Sloan Management Rev 43(2) (2002), 60–67.

DoD, DoD Architecture Framework Version 1.0 (Draft), U.S. Department of Defense, Washington, DC, 2001.

D. Dougherty, Reimagining the differentiation and integration of work for sustained product innovation, Org Sci 12(5) (2001), 612–631.

S.E. Elmaghraby, Activity nets: A guided tour through some recent developments, European J Oper Res 82(3) (1995), 383–408.

S.D. Eppinger, Innovation at the speed of information, Harvard Bus Rev 79 (2001), 149–158.

S.D. Eppinger and V. Salminen, Patterns of product development interactions, Proc Int Conf Eng Des (ICED), Glasgow, 2001.

D.N. Ford and J.D. Sterman, Dynamic modeling of product development processes, Syst Dyn Rev 14(1) (1998), 31–68.

D.N. Ford and J.D. Sterman, Overcoming the 90% syndrome: Iteration management in concurrent development projects, 2003.

K. Forsberg, H. Mooz, and H. Cotterman, Visualizing project management, Wiley, New York, 2000.

E. Fricke, H. Negele, L. Schrepfer, A. Dick, B. Gebhard, and N. Härtlein, Modeling of concurrent engineering processes for integrated systems development, Proc 17th Digital Avionics Syst Conf, Bellevue, WA, 1998.

E. Fricke, A. Schulz, P. Wehlitz, and H. Negele, A generic approach to implement information-based systems development, Proc 10th Annu Int Symp INCOSE, Minneapolis, 2000, pp. 263–270.

J.R. Galbraith, Organization design, Addison-Wesley, Reading, MA, 1977.

E.M. Goldratt, Critical chain, The North River Press, Great Barrington, MA, 1997.

M. Hammer, Seven insights about processes, Proc Strategic Power Process Ensuring Survival Creating Competitive Advantage, Boston, 2001.

J.R. Hauser and D. Clausing, The house of quality, Harvard Bus Rev 66 (1988), 63–73.

G.A. Hazelrigg, On the role and use of mathematical models in engineering design, J Mech Des 121(3) (1999), 336–341.

J.A. Highsmith, Adaptive software development: A collaborative approach to managing complex systems, Dorset House, New York, 2000.

IMTI, Integrated Manufacturing Technology Roadmapping project: An overview of the IMTR roadmaps, Integrated Manufacturing Technology Initiative, Oak Ridge, TN, 2000.

INCOSE, Systems engineering handbook: A "what to" guide for all SE practitioners, International Council on Systems Engineering, 2004.

N.R. Joglekar and D.N. Ford, Product development resource allocation with foresight, European J Oper Res 160(1) (2005), 72–87.

H.B. Jun and H.-W. Suh, The Hierarchical Frame of Enterprise Activity Modeling (HF-EAM), IEEE Trans Eng Management 49(4) (2002), 459–478.

M. Kamath, N.P. Dalal, A. Chaugule, E. Sivaraman, and W.J. Kolarik, "A review of enterprise process modelling techniques," Scalable enterprise systems: An introduction to recent advances, V. Prabhu, S. Kumara, and M. Kamath (Editors), Kluwer Academic, New York, 2003.

H. Kilov, B. Rumpe, and I. Simmonds (Editors), Behavioral specifications of businesses and systems, Kluwer Academic, Boston, 1999, Vol. 523.

S.J. Kline, Innovation is not a linear process, Res Management 26(2) (Jul.-Aug. 1985), 36–45.

V. Krishnan, S.D. Eppinger, and D.E. Whitney, A model-based framework to overlap product development activities, Management Sci 43(4) (1997), 437–451.

S.G. Lee, K.L. Ong, and L.P. Khoo, Control and monitoring of concurrent design tasks in a dynamic environment, Concurrent Eng Res Appl 12(1) (2004), 59–66.

V. Lévárdy and T.R. Browning, Adaptive test process—designing a project plan that adapts to the state of a project, Proc 15th Annu Int Symp INCOSE, Rochester, NY, 2005.

P. Lillrank, The broom and nonroutine processes: A metaphor for understanding variability in organizations, Knowledge Process Management 9(3) (2002), 143–148.

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.

D.A. Marca and C.L. McGowan, IDEF0/SADT business process and enterprise modeling, Eclectic Solutions Corporation, San Diego, 1993.

J.G. March and H.A. Simon, Organizations, Blackwell Business, Cambridge, MA, 1993.

H. McManus, Product development value stream mapping manual, MIT Lean Aerospace Initiative, Cambridge, MA, 2004.

H.L. McManus and R.L. Millard, Value stream analysis and mapping for product development, Proc Int Council Aeronaut Sci (ICAS) Congress, Toronto, 2002.

C. Menzel and R.J. Mayer, "The IDEF family of languages," Handbook on architectures of information systems, P. Bernus, K. Mertins and G. Schmidt (Editors), Springer, Berlin, 1998, pp. 209–241.

L.J. Moore and B.W. Taylor, Multiteam, multiproject research and development planning with GERT, Management Sci 24(4) (1977), 401–410.

H. Negele, Systemtechnische Methodik zur ganzheitlichen Modellierung am Beispiel der integrierten Produktentwicklung (A systems engineering methodology for comprehensive modeling applied to integrated product development), Ph.D. Thesis (Institute of Astronautics), Technische Universität München, Munich, Germany, 1998.

H. Negele and S. Wenzel, Systems engineering meta-tools for complex product development, Proc 10th Annu Int Symp INCOSE, Minneapolis, 2000, pp. 465–472.

H. Negele, E. Fricke, and E. Igenbergs, ZOPH—a systemic approach to the modeling of product development systems, Proc 7th Annu Int Symp INCOSE, Los Angeles, 1997, pp. 773–780.

H. Negele, E. Fricke, L. Schrepfer, and N. Härtlein, Modeling of integrated product development processes, Proc 9th Annu Int Symp INCOSE, Brighton, UK, 1999, pp. 1253–1261.

K. Neumann, Stochastic project networks: Temporal analysis, scheduling and cost minimization, Springer, Berlin, 1990, Vol. 344.

NIST, Integration Definition for Function Modeling (IDEF0), National Technical Information Service, U.S. Department of Commerce, Springfield, VA, 1993.

B. O'Donovan, T.R. Browning, C.M. Eckert, and P.J. Clarkson, "Design planning and modelling," Design process improvement: A review of current practice, P.J. Clarkson and C.M. Eckert (Editors), Springer, 2005, pp. 60–87.

B.D. O'Donovan, P.J. Clarkson, and C.M. Eckert, Signposting: Modelling uncertainty in design processes, Proc Int Conf Eng Des (ICED), Stockholm, 2003.

G. Pahl and W. Beitz, Engineering design, The Design Council, London, 1995.

L. Pajerek, Processes and organizations as systems: When the processors are people, not pentiums, Syst Eng 3(2) (2000), 103–111.

G.A. Pall, The process-centered enterprise: The power of commitments, St. Lucie Press, New York, 1999.

B.T. Pentland, Grammatical models of organizational processes, Org Sci 6(5) (1995), 541–556.

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.

S. Pugh, Total design: Integrated methods for successful product engineering, Addison-Wesley, Reading, MA, 1991.

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.

E.S. Raymond, The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary, O'Reilly, 2001.

E. Rechtin, Systems architecting: Creating & building complex systems, PTR Prentice Hall, Englewood Cliffs, NJ, 1991.

D. Reinertsen, Lean thinking isn't so simple, Electron Des 47 (1999), 48.

N.P. Repenning and J.D. Sterman, Nobody ever gets credit for fixing problems that never happened: Creating and sustaining process improvement, California Management Rev 43(4) (2001), 64–88.

N. Sabbaghian, S. Eppinger, and E. Murman, Product development process capture & display using Web-based technologies, Proc IEEE Int Conf Syst Man Cybernet, San Diego, CA, 1998, pp. 2664–2669.

S.D. Savransky, Engineering of creativity: Introduction to TRIZ methodology of inventive problem solving, CRC Press, Boca Raton, FL, 2000.

A.-W. Scheer, "ARIS," Handbook on architectures of information systems, P. Bernus, K. Mertins, and G. Schmidt (Editors), Springer, Berlin, 1998a, pp. 541–564.

A.-W. Scheer, ARIS—business process frameworks, Springer, New York, 1998b.

C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, and J. Lee, The Process Specification Language (PSL) overview and Version 1.0 specification, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2000, p. 83.

S. Schrader, W.M. Riggs, and R.P. Smith, Choice over uncertainty and ambiguity in technical problem solving, J Eng Technol Management 10(1) (1993), 73–99.

SEI, Capability Maturity Model® Integration (CMMI^{SM}), Version 1.1, Carnegie Mellon University Software Engineering Institute, Pittsburgh, PA, 2002.

N. Senin, D.R. Wallace, and N. Borland, Distributed object-based modeling in design simulation marketplace, J Mech Des 125(1) (2003), 2–13.

E. Sivaraman and M. Kamath, Business process modeling and workflow management: Current status and research issues, Stillwater, OK, 2003.

R.P. Smith, The historical roots of concurrent engineering fundamentals, IEEE Trans Eng Management 44(1) (1997), 67–78.

R.P. Smith and S.D. Eppinger, A predictive model of sequential iteration in engineering design, Management Sci 43(8) (1997), 1104–1120.

R.P. Smith and J.A. Morrow, Product development process modeling, Des Stud 20(3) (1999), 237–261.

S. Spear and H.K. Bowen, Decoding the DNA of the Toyota production system, Harvard Bus Rev 77 (1999), 97–106.

D.V. Steward, The design structure system: A method for managing the design of complex systems, IEEE Trans Eng Management 28(3) (1981), 1–74.

D.A. Tacconi and F.L. Lewis, A new matrix model for discrete event systems: Application to simulation, IEEE Control Syst Mag 17 (1997), 62–71.

M.V. Tatikonda and S.R. Rosenthal, Successful execution of product development projects: Balancing firmness and flexibility in the innovation process, J Oper Management 18(4) (2000), 401–425.

B.W. Taylor and L.J. Moore, R&D project planning with Q-GERT network modeling and simulation, Management Sci 26(1) (1980), 44–59.

C. Terwiesch and C.H. Loch, Measuring the effectiveness of overlapping development activities, Management Sci 45(4) (1999), 455–465.

F. Tissot and W. Crump, "An integrated enterprise modeling environment," Handbook on architectures of information systems, P. Bernus, K. Mertins, and G. Schmidt (Editors), Springer, Berlin, 1998, pp. 481–507.

M.L. Tushman and D.A. Nadler, Information processing as an integrating concept in organizational design, Acad Management Rev 3(3) (1978), 613–624.

W. van der Aalst and K.v. Hee, Workflow management: Models, methods, and systems, MIT Press, Cambridge, MA, 2004.

T. Winograd and F. Flores, Understanding computers and cognition: A new foundation for design, Addison-Wesley, New York, 1986.

B. Yu, J.A. Harding, and K. Popplewell, Supporting enterprise design through multiple views, Int J Agile Management Syst 2(1) (2000), 71–82.

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 in the M.J. Neeley School of Business at Texas Christian University, Fort Worth, Texas, USA. He teaches MBA courses in project, program, and operations management and conducts research on enterprise operations, process modeling, product development, project management, engineering management, and systems engineering. Prior to joining TCU, Tyson was a Senior Project Manager in Integrated Company Operations at Lockheed Martin Aeronautics Company in Fort Worth, where he was the technical lead and chief integrator of the enterprise process architecture and author of company policies and processes driving the transition to a process-based company. Before joining Lockheed Martin, he worked with the Lean Aerospace Initiative at the Massachusetts Institute of Technology (MIT), conducting on-site research at Boeing and six other automotive and aerospace companies. He has also worked for Honeywell Space Systems and Los Alamos National Laboratory. Tyson 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 MIT. He has authored over 20 papers, publishing in *IEEE Transactions in Engineering Management Project Management, Journal*, *Systems Engineering*, *Technology Management Handbook*, and others. He is a member of INCOSE and the Institute for Operations Research and the Management Sciences (INFORMS).

Ernst Fricke is project manager in systems engineering, being responsible for the implementation of a more distinctive systems engineering approach in BMW's future vehicle topic projects. Before, he worked two years as a Senior Consultant in BMW's In-House Consulting Department for Development, Production and Purchase Processes. Until 2002 he was the Head of Systems Engineering and Support at CargoLifter Development GmbH, an aeronautic start-up company. The years before he had worked as "wissenschaftlicher Mitarbeiter" at the Institute of Astronautics, Technische Universität München on systems engineering issues in several industrial projects with partners mainly from aerospace and automotive industry. Ernst received a Master's Degree in aerospace engineering from Technische Universität München (TUM) in 1994 and his Ph.D. in Mechanical Engineering from TUM in January 1999. He is a graduate of the International Space University SSP '97, Houston, TX, is co-founder of the German Chapter of INCOSE, and received the Outstanding Paper Award for the best *Systems Engineering* journal paper of the years 1998–2003. Ernst is Co-Editor of the German Systems Engineering book series in the Herbert Utz Verlag.

Herbert Negele is Project Manager for the implementation of systems engineering in Driver Assistance Systems at BMW Group. Prior to this assignment, he was leading major project, risk, and process management initiatives within the electrics/electronics system development and at the overall vehicle level. Until November 1999 he worked as an Assistant Professor at the Institute of Astronautics in the field of systems engineering with special focus on systems modeling and simulation, integrated product and process development, and project management. He received a Master's Degree in aerospace engineering in 1993 and his Ph.D. in systems engineering in 1998 from the Technische Universität München. Herbert is a co-founder of the German Chapter of INCOSE (Gesellschaft für Systems Engineering, GfSE e.V.) and was its first Vice President in 1997 and 1998. He received the Outstanding Paper Award from *Systems Engineering* for the best journal paper in the years 1998–2003. Herbert is Co-Editor of the German Systems Engineering book series in the Herbert Utz Verlag and a member of the INCOSE Commercial Steering Board.