CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Dynamic Models of Knowledge-Flow Dynamics

By

Mark Nissen and Raymond Levitt

CIFE Working Paper #76
November 2002

STANFORD UNIVERSITY

If you would like to contact the authors, please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

Dynamic Models of Knowledge-Flow Dynamics

Mark E. Nissen, Naval Postgraduate School
Raymond E. Levitt, Stanford University

14 November 2002

Abstract

Knowledge is unevenly distributed through most enterprises, so knowledge flow (e.g., across time, location, organization) is critical to organizational efficacy and performance under a knowledge-based view of the firm. Although *knowledge flow* is an inherently dynamic concept, however, the corresponding phenomenon remains poorly understood, and extant approaches to its modeling and description (e.g., natural language texts and figures) are fundamentally static and largely ambiguous. In this research, we build upon emerging theory for multidimensional conceptualization of the knowledge-flow phenomenon to develop *dynamic* models of knowledge-flow dynamics. Drawing from recent advances in computational organization theory, we describe a research approach and modeling environment that enables the dynamics of enterprise knowledge flows to be formalized through computational models. Particularly when compared to extant descriptive theory articulated through natural language, such formal models are considerably less ambiguous, more reliable and quite explicit. We illustrate this research approach and modeling environment through formal representation and simulation of several knowledge-flow processes from the domain of software development. When used in conjunction with current theory, the new insight into and understanding of knowledge-flow dynamics revealed through this research is compelling and represents a contribution to the information systems literature. The article closes with key new directions for the kind of research elucidated by this work.

Keywords: artificial intelligence, computational organization theory, knowledge flow, knowledge management, knowledge representation, organizational learning, simulation, software.

Acknowledgement. This research is sponsored in part by the Office of Naval Research, Young Investigator Program: N0001401WR20304.

INTRODUCTION

Knowledge represents a critical resource in the modern enterprise—so critical that it is now being conceptualized as central to competitive advantage in a knowledge-based view of the firm (Cole 1998, Grant 1996, Spender 1996). But knowledge is not evenly distributed through the enterprise. Capitalizing on this resource for enterprise performance depends upon its rapid and efficient transfer from one organization, location or time of application to another. From a technological perspective, such dynamic dependence points immediately to the design of information systems (IS)—along with corresponding organization and process characteristics (Leavitt 1965, Davenport 1993)—to enhance knowledge flow. But knowledge is distinct from information and data (e.g., it

enables direct, appropriate action; see Davenport et al. 1998, Teece 1998), and few extant IS even address *knowledge* as the focus or object of flow (Nissen 1999). Indeed in this light, the IS field does not have the benefit of strong theory on knowledge flow, as Alavi and Leidner (2001, p. 126) note, there exist "large gaps in the body of knowledge in this area."

So how does knowledge flow through the modern enterprise, and what kinds of managerial interventions (e.g., IS development, training, organizational change, workflow reconfiguration) can be made to enhance the flow of knowledge? A number of theoretical models have been developed to describe various aspects of the knowledge-flow phenomenon (e.g., Augier et al. 2001, Dixon 2000, King and Ko 2001, Markus 2001, O'Leary 2001, Schultze and Boland 2000, Swap et al. 2001), but few provide insight into the phenomenon itself; that is, a paucity of models have been developed to describe *how* knowledge flows through the enterprise. Until we can understand the phenomenon of knowledge flow, we are unlikely to design effective flow-enhancing interventions. By analogy, imagine trying to design flow-enhancing electronic devices (e.g., amplifiers, integrated circuits) without understanding the phenomenon of electrical flow, or trying to develop flight-worthy aircraft (e.g., designing jet engines, wings) without understanding how air flows at various velocities, angles, densities and temperatures. This is not limited to designing physical systems: consider the futility of designing an assembly line without understanding how manufacturing work (e.g., purchased materials, fabricated parts, subassemblies) flows through sequential and concurrent operations, or a logistical network without understanding how cargo flows through various topological configurations (e.g., hub and spoke, ring, point to point).

Von Hippel (1994) takes a notable step in the direction of understanding how knowledge flows, as he examines causal factors for the relative marginal costs—characterized by the term *stickiness*—associated with transferring tacit and explicit knowledge for technical-innovation problem solving. Szulanski (1996, 2000) goes further by tying his "stickiness" notion to four different stages of the knowledge-transfer process (i.e., initiation, implementation, ramp-up, integration). Numerous life cycle models (see Nissen et al. 2000) adopt a similar staged view of knowledge flow. Nonaka (1994) goes further still, as he introduces a model describing a "spiral" of

dynamic interaction between tacit and explicit knowledge along an epistemological dimension (p. 18), and he characterizes four processes (i.e., socialization, externalization, combination, integration) that enable individual knowledge to be "amplified" (p. 21) and effect organizational knowledge "crystallization" (p. 25) along the ontological dimension. A later and related work (Nonaka et al. 1996) identifies enabling "triggers" for and provides additional workplace examples of each knowledge-flow process (e.g., the "trigger" for socialization is building a "field" of interaction; p. 842).

Building upon these theoretical steps, Nissen (2002) integrates and extends the research above to develop a phenomenological model of enterprise knowledge flow. This model makes flow time explicit and supports a multidimensional representational framework that enables a new approach to analysis and visualization of diverse knowledge-flow patterns in the enterprise. However, even this rich characterization of knowledge-flow dynamics remains static in terms of its representational model; that is, like the other extant models of knowledge flow, it does not use a *dynamic* representation of knowledge-flow dynamics. This is much like trying to visualize three-dimensional motion through a static picture (e.g., camera snapshot). Moreover, throughout this line of research, important dynamic interactions between model elements remain obscured through descriptive models based upon natural language texts and figures.

The research described in this article continues the work above by addressing the knowledge-flow phenomenon, but it takes a qualitative leap by introducing dynamic representation of knowledge-flow dynamics. Drawing upon current research advances in computational organization theory, the resulting model provides much greater fidelity and insight into knowledge-flow dynamics than the extant literature does, and it enables the execution and performance of diverse knowledge-work processes to be simulated for analysis and comparison. Further, through commitment to a formal model of the knowledge-flow phenomenon, the resulting representation is considerably clearer and more precise than even the richest of those above, and the approach is broadly generalizable. This represents a substantial new contribution to the IS literature. And through a program of systematic research (e.g., controlled experimentation through repeated simulation of enterprise processes under varying conditions), dynamic models

of knowledge-flow dynamics can help us understand how to enhance the flow of knowledge through the modern enterprise.

To describe this approach, we begin by drawing from the literature to develop a rich but static, multidimensional representation of enterprise knowledge flow. Building upon this representation, a dynamic knowledge-flow model is specified and instantiated, and its use and utility are in turn demonstrated through simulation of important knowledge-work processes from the IS literature. The article concludes by summarizing key implications of this work and outlining an agenda for continued research along the lines of this investigation.

STATIC MODEL INTEGRATION

The Spiral Model described by Nonaka (1994) serves as the cornerstone of model integration in this section. Unlike the technical-transfer work of von Hippel and Szulanski above, this model describes the kinds of continuous and routine flows that comprise the bulk of organizational knowledge work. Figure 1 delineates the interaction between epistemological and ontological dimensions used by Nonaka as the principal means for describing knowledge as it flows through the enterprise. As noted above, this flow is characterized by four enterprise processes (and epistemological conversions): socialization (tacit to tacit), externalization (tacit to explicit), combination (explicit to explicit), and internalization (explicit to tacit). The related *trigger* concept (i.e., from Nonaka et al. 1996) is also integrated into the figure to show where each knowledge-conversion process is "induced" (p. 842) by one of four triggers: field building, dialog, linking explicit knowledge, and learning by doing.

Briefly, socialization denotes members of a team sharing experiences and perspectives, much as one anticipates through tightly knit workgroups and communities of practice; in terms of the trigger, a "field" of interaction is seen as facilitating this sharing. Externalization denotes the use of metaphors through dialog that leads to articulation of tacit knowledge and its subsequent formalization to make it concrete and explicit; such dialog or what Nonaka et al. refer to as "collective reflection" (p. 842) is described as inducing externalization. Combination denotes coordination between different groups in the organization—along with documentation of existing knowledge—to link and combine new intra-team concepts with other explicit knowledge in the

enterprise. Internalization denotes diverse members in the organization applying the combined knowledge from above—often through trial and error—and in turn translating such knowledge into tacit form at the organization level (e.g., through work practices and routines); the term *learning by doing* is used to describe the trigger for knowledge internalization. As suggested by the repeating pattern delineated in the figure, such interaction between "triggers" and conversions enables a continuous "spiral" of knowledge.

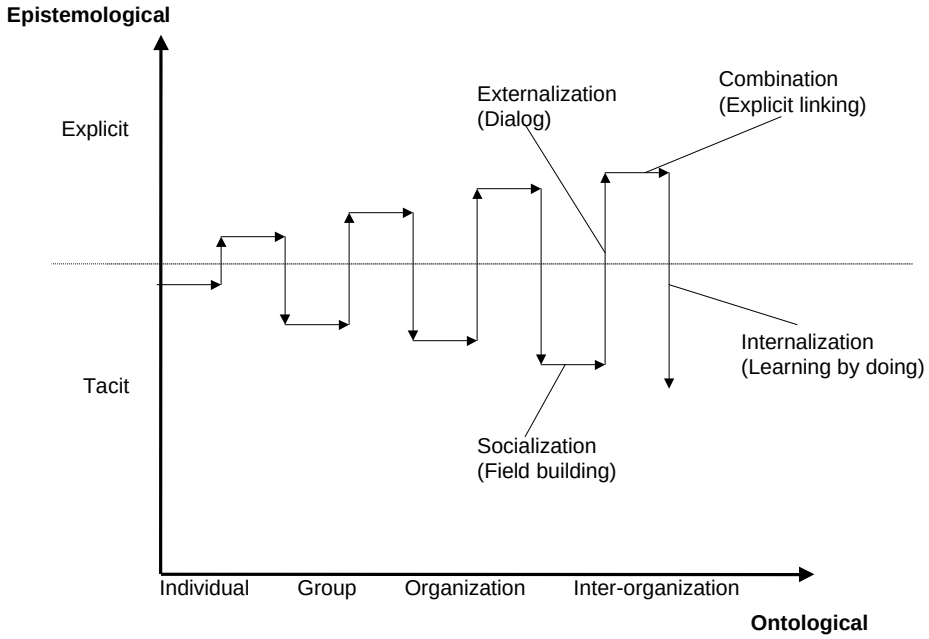

Figure 1 Spiral Model (adapted from Nonaka 1994, Nonaka et al. 1996)

We extend this Spiral Model by drawing from Nissen et al. (2000), who develop their Life Cycle Model by amalgamating several individual works (e.g., Despres and Chauvel 1999, Gartner Group 1999, Davenport and Prusak 1998, Nissen 1999). This amalgamated model describes a continuous cycle with six phases of knowledge flowing through the enterprise: 1) creation, 2) organization, 3) formalization, 4) distribution, 5) application, and 6) evolution. Briefly, the creation phase begins the life cycle, as new knowledge is generated within an enterprise; similar terms from other models include *capture* and *acquire*. The second phase pertains to the organization, mapping or bundling of knowledge, often employing systems such as taxonomies, ontologies and

repositories. Phase 3 addresses mechanisms for making knowledge formal or explicit; similar terms from other models include *store* and *codify.* The fourth phase concerns the ability to share or distribute knowledge in the enterprise; this also includes terms such as *transfer* and *access*. Knowledge use and application for problem solving or decision making in the organization constitutes Phase 5, and a sixth phase is included to cover knowledge refinement and evolution, which reflects organizational learning—and thus a return to knowledge creation—through time. It is important to note, as in the familiar life cycle models used in IS design (e.g., System Development Life Cycle or SDLC), progression through the various phases of this Life Cycle Model is generally iterative and involves feedback loops between stages; that is, all steps need not be taken in order, and the flow through this life cycle is not necessarily unidirectional.

Integrating the concepts above, the resulting model can be represented using three dimensions: *epistemological, ontological* and *life cycle*. Clearly, many additional dimensions could also be integrated into this model (e.g., declarative and procedural, see Nolan Norton 1998; practical and theoretical, see Spender 1996; know-what and know-how, see Ryle 1958; causal, conditional and relational, see Zack 1998; embodied, encoded, embrained, embedded and procedural, see Venzin et al. 1998). But using three dimensions strikes a balance between descriptiveness and parsimony, and the three dimensions selected for this model all derive from research focused specifically on knowledge flow. We use this three-dimensional representation to characterize in new ways the complex interactions between knowledge in alternative states as it flows through the enterprise. Yet we also preserve the descriptive and explanatory abilities of the individual models that underlie (and are subsumed by) this integrative work.

The three-dimensional representation also enables us to visualize a diversity of enterprise knowledge flows, as it defines a vector space and enables us to plot dynamic trajectories for each flow. Drawing from Nissen (2002), three notional knowledge-flow trajectories are plotted in Figure 2 for illustration. For instance, the simple linear flow labeled “P&P” depicts the manner in which most large enterprises inform and attempt to acculturate employees through the use of policies and procedures: explicit documents and guidelines that individuals in the organization are expected to memorize, refer to and observe. As another instance, the cyclical

flow of knowledge through a life cycle (labeled “KMLC”) reflects a more-complex dynamic than its simple linear counterpart. The “spiral” dynamic from above can also be delineated in this space by the curvilinear vector sequence **S-E-C-I** (i.e., corresponding to the processes of socialization, externalization, combination, and internalization, respectively). Further, drawing from the Life Cycle Model, we can append processes to represent knowledge creation and evolution (labeled as vectors **K** and **V,** respectively), which we in turn link together to depict a serpentine flow that spans the ranges of all three dimensions. Notice in this representation that we can explicitly depict flow times of various magnitudes by proportionately adjusting the thickness of arrows used to represent each knowledge-flow vector; this enables us to differentiate graphically between specific flow processes and states associated with “sticky” and “fluid” knowledge, for example. Notice also that different knowledge flows delineate discernable patterns in this vector-space representation (e.g., lines, cycles, spirals). Clearly, a great many flows and patterns can be depicted in this manner.

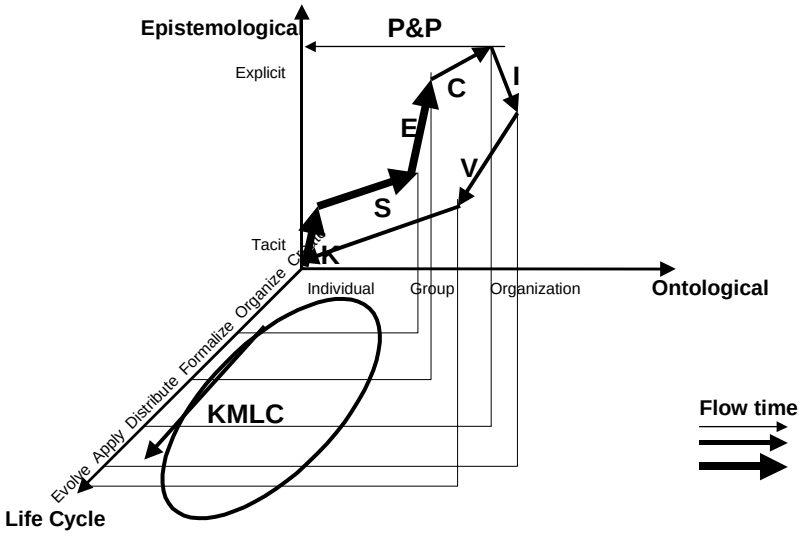

Figure 2 Notional Knowledge-Flow Trajectories (adapted from Nissen 2002)

DYNAMIC MODEL REPRESENTATION

Despite the richness of our multidimensional representation above, the representation itself remains static; that is, although we use it to describe the dynamics of knowledge-flow processes, the representation is limited to a static combination of natural language texts and diagrams. In this section, we draw upon current research advances in computational organization theory to describe a dynamic representational environment used for formal organizational modeling, and we employ this environment to describe a computational model of enterprise knowledge-flow processes.

Virtual Design Team Research

The Virtual Design Team (VDT) Research Program (VDT 2002) reflects the planned accumulation of collaborative research over two decades to develop rich theory-based models of organizational processes. Using an agent-based representation (Cohen 1992, Kunz et al. 1998), micro-level organizational behaviors have been researched and formalized to reflect well-accepted organization theory (Levitt et al. 1999), and extensive empirical validation projects (e.g., Christiansen 1993, Thomsen 1998) have demonstrated the representational fidelity and shown how the qualitative behaviors of VDT computational models correspond closely with a diversity of enterprise processes in practice.

The VDT research program continues with the goal of developing new micro-organization theory and embedding it in software tools that can be used to design organizations in the same way that engineers design bridges, semiconductors or airplanes—by modeling, analyzing and evaluating multiple virtual prototypes of the system to be designed in a computer. Clearly this represents a significant challenge. Micro-theory and analysis tools for designing bridges and airplanes rest on well-understood principles of physics (e.g., involving continuous numerical variables, describing materials whose properties are relatively easy to measure and calibrate), and analysis of such physical systems yields easily to differential equations and precise numerical computing.

In contrast, theories describing the behavior of organizations are characterized by nominal and ordinal variables, with poor measurement reproducibility, and verbal descriptions

reflecting significant ambiguity. Unlike the mathematically representable and analyzable micro-behaviors of physical systems, the dynamics of organizations are influenced by a variety of social, technical and cultural factors, are difficult to verify experimentally, and are not amenable to numerical representation, mathematical analysis or precise measurement. Moreover, quite distinct from physical systems, people and social interactions (i.e., not molecules and physical forces) drive the behavior of organizations; hence such behaviors are fundamentally non-deterministic and difficult to predict at the individual level. Thus, people, organizations and business processes are qualitatively different than bridges, semiconductors and airplanes, and it is irrational to expect the former to ever be as understandable, analyzable or predictable as the latter. This represents a fundamental limitation of the approach.

Within the constraints of this limitation, however, we can still take great strides beyond relying upon informal and ambiguous theoretical description of organizational behavior. For instance, the domain of organization theory is imbued with a rich, time-tested collection of micro-theories that lend themselves to qualitative representation and analysis. Examples include Galbraith's (1977) information processing abstraction, March and Simon's (1958) bounded rationality assumption and Thompson's (1967) task interdependence contingencies. Drawing from this theory base, we employ symbolic (i.e., non-numeric) representation and reasoning techniques from established research on artificial intelligence (AI) to develop computational models of theoretical phenomena. Once formalized through a computational model, the symbolic representation is "executable," meaning it can be developed to emulate the dynamics of organizational behaviors.

Even though the representation is qualitative (e.g., lacking the precision offered by numerical models), through commitment to computational modeling, it becomes formal (e.g., people viewing the model can agree on exactly what it describes), reliable (e.g., the same sets of organizational conditions and environmental factors generate the same sets of behaviors) and explicit (e.g., much ambiguity inherent in natural language is obviated). Particularly when used *in conjunction with* the descriptive natural language theory of our extant literature, this represents a substantial advance. Further, once a model has been validated to accurately emulate the

qualitative behaviors of the field organization it represents, it can be used to examine a multitude of cases (e.g., many more and diverse than observable in practice) under controlled conditions (e.g., repeating the same events multiple times, manipulating only one or a few variables at a time through repeated trials, stopping the action for interpretation). This alone offers great promise in terms of theory development.

Additionally, although organizations are inherently less understandable, analyzable and predictable than physical systems, and the behavior of people is non-deterministic and difficult to model at the individual level, it is well known that individual differences tend to average out when aggregated cross-sectionally and/or longitudinally. Thus, when modeling aggregations of people in the organizational context (e.g., work groups, departments, firms), one can augment the kind of symbolic model from above with certain aspects of numerical representation. For instance, the distribution of skill levels in an organization can be approximated—in aggregate—by a Bell Curve; the probability of a given task incurring exceptions and requiring rework can be specified—organization wide—by a distribution; and the unpredictable attention of a worker to any particular activity or event (e.g., new work task, communication, request for assistance) can be modeled—stochastically—to approximate collective behavior. As another instance, specific organizational behaviors can be simulated hundreds of times—such as through Monte Carlo techniques—to gain insight into which results are common and expected versus those that are rare and exceptional.

Of course, applying numerical simulation techniques to organizations is nothing new (e.g., see Law and Kelton 1991). But this approach enables us to *integrate* the kinds of dynamic, qualitative behaviors emulated by symbolic models with quantitative aggregate dynamics generated through discrete-event simulation. It is through such integration of qualitative and quantitative models—bolstered by strong reliance upon well-established theory and commitment to empirical validation—that our approach diverges most from extant research methods and offers new insight into the dynamics of organizational behavior. This summarizes the approach we take to modeling knowledge-flow dynamics.

VDT Modeling Environment

The VDT modeling environment has been developed directly from Galbraith's information processing view of organizations. This information processing view of organizations has two key implications (Jin and Levitt 1996). The first is ontological: we model knowledge work through interactions of *tasks* to be performed, *actors* communicating with one another and performing tasks, and an *organization structure* that defines actors' roles and constrains their behaviors. Figure 3 illustrates this view of tasks, actors and organization structure. As suggested by the figure, we model the organization structure as a network of reporting relations, which can capture micro-behaviors such as managerial attention, span of control and empowerment, and we represent the task structure as a separate network of activities, which can capture organizational attributes such as expected duration, complexity and required skills. Within the organization structure, we further model various *roles* (e.g., marketing analyst, design engineer, manager), which can capture organizational attributes such as skills possessed, level of experience and task familiarity. Within the task structure, we further model various sequencing constraints, interdependencies and quality/rework loops, which can capture considerable variety in terms of how knowledge work is organized and performed.

As also suggested by the figure, each actor within the intertwined organization and task structures has a queue of information tasks to be performed (e.g., assigned work activities, messages from other actors, meetings to attend) and a queue of information outputs (e.g., completed work products, communications to other actors, requests for assistance). Each actor also processes such tasks according to how well the actor's skill set matches those required for a given activity, the relative priority of the task, the actor's work backlog (i.e., queue length), and how many interruptions divert the actor's attention from the task at hand. Actors' collective task performance is further constrained by the number of individual actors assigned to each task, the magnitude of the task, and both scheduled (e.g., work breaks, ends of shifts, weekends and holidays) and unscheduled (e.g., awaiting managerial decisions, awaiting work or information inputs from others, performing rework) downtime.

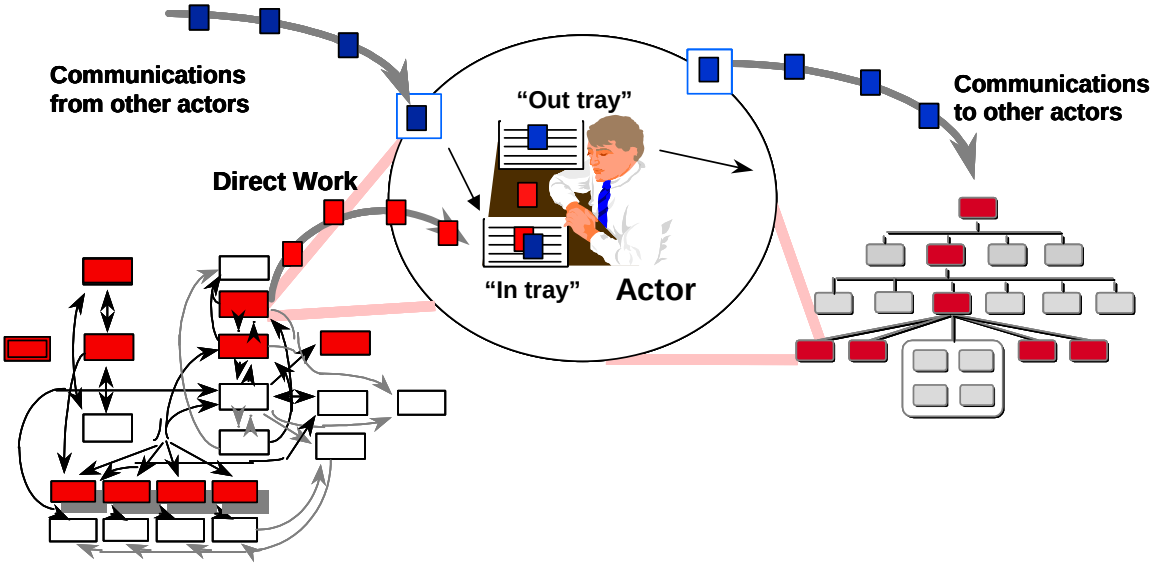

Figure 3 VDT Information Processing View of Knowledge Work

The second implication is computational: both primary work (e.g., planning, design, management) and coordination work (e.g., group tasks, meetings, joint problem solving) are modeled in terms of *work volume.* This construct is used to represent a unit of work (e.g., associated with a task, a meeting, a communication) within the task structure. In addition to symbolic execution of VDT models (e.g., qualitatively assessing skill mismatches, task-concurrency difficulties, decentralization effects) through micro-behaviors derived from organization theory, the discrete-event simulation engine enables (virtual) process performance to be assessed (e.g., quantitatively projecting task duration, cost, rework, process quality).

Clearly quantitative simulation places additional burden on the modeler in terms of validating the representation of a knowledge-work process, which generally requires fieldwork to study an organization in action. The VDT modeling environment benefits from extensive fieldwork in many diverse enterprise domains (e.g., power plant construction and offshore drilling, see Christiansen 1993; aerospace, see Thomsen 1998; software development, see Nogueira 2000; healthcare, see Cheng and Levitt 2001; others). And through the process of "backcasting"—predicting known organizational outcomes using only information that was available at the beginning of a project—VDT models of operational enterprises in practice have demonstrated

dozens of times that simulated organizational behaviors and results correspond qualitatively to their actual counterparts in the field (Kunz et al. 1998).

Viewing VDT as a validated model of project-oriented knowledge work, researchers have begun to use this dynamic modeling environment as a "virtual organizational testbench" to explore a variety of organizational questions, such as effects of distance on performance (Wong and Burton 2000) or to replicate classical empirical findings (Carroll and Burton 2000). Thus, the VDT modeling environment has been repeatedly and longitudinally validated as representative of both organization theory and enterprises in practice. This gives us considerable confidence in its results. However, because of its *information* processing view of the organization, the VDT modeling environment is not specifically designed to represent processes associated with the flow of *knowledge* through an enterprise. The computational knowledge-flow model developed below represents a new step in terms of VDT research.

Computational Knowledge-Flow Model

Here we employ the VDT modeling environment to represent knowledge-flow processes associated with software development. With its output the product of collaboration between people and computers, software development represents a relatively pure form of knowledge work, and the associated processes and tools have long been the focus of IS literature. This domain also helps elucidate our multidimensional knowledge-flow representation from above, and it highlights both differences and linkages between flows of work in the enterprise and the corresponding flows of knowledge. Building upon prior research in this domain (e.g., Nogueira 2000), we use the VDT modeling environment's *behavior file* for software development, which benefits from the kind of field research and validation discussed above.

Using a relatively simple work process for exposition, an overview of our VDT model for software development is illustrated through the screenshot presented in Figure 4. The organization structure is comprised of only three elements: 1) a knowledge worker (i.e., person icon at top of diagram labeled "S/W arch") with skills in software-architecture development, 2) a team of (10) software engineers (i.e., person icon at top of diagram labeled "S/W eng team1") with skills in software analysis, design and programming, and 3) a project lead (i.e., person icon

at top of diagram labeled “S/W PM”) with skills in project planning, supervision and software development. The task structure is comprised of only two work elements: 1) software architecture (i.e., rectangular icon at middle of diagram labeled “S/W arch”) and 2) software engineering (i.e., rectangular icon at middle of diagram labeled “S/W engr”); the four milestones shown (i.e., project start, architecture complete, application complete, project finish) serve as markers of progress against schedule but do not involve work.

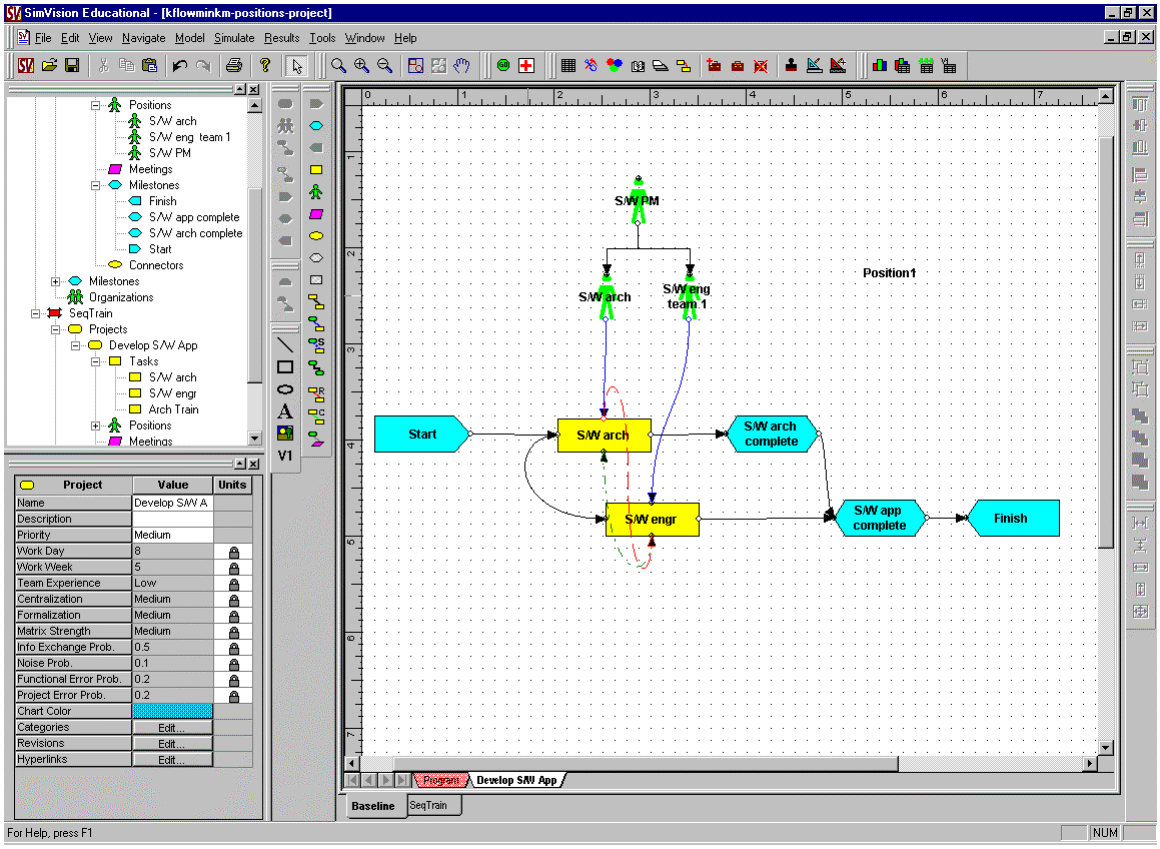

Figure 4 VDT Software Development Model – Baseline

Five types of connections are delineated between organization and task elements in this representation: 1) precedence connections (e.g., between “S/W arch” and “S/W engr” tasks, “S/W arch” task and “S/W arch complete” milestone; shown in black) link tasks and milestones according to the order in which they must be accomplished; 2) communication connections (e.g., between “S/W arch” and “S/W engr” tasks; shown in green) link tasks associated with reciprocal interdependence, which require mutual adjustment by actors (Thompson 1967); 3) rework

connections (e.g., between “S/W arch” and “S/W engr” tasks; shown in red) link tasks in which exceptions or failures “downstream” (e.g., in software-engineering activities) feedback to cause rework “upstream” (e.g., in software-architecture activities); 4) assignment connections (e.g., between “S/W eng team1” actors and “S/W engr” task; shown in blue) indicate which organizational actors are assigned responsibility for each work task; and 5) supervision connections (e.g., between “S/W PM” and “S/W arch” actors) indicate the formal organizational hierarchy.

The two windows at the left of this figure reveal a tree structure of the representation (top) and numerous model parameters (e.g., priority, centralization, team experience; bottom) used to instantiate a particular work process (e.g., a software development effort in the field). These parameters are all set to empirically determined “normal” values for a software development project such as our example, except for “team experience,” which we initially set to a relatively low level to represent a group of software engineers that do not have much experience working together.

The diagram in Figure 4 is representative of the kinds of models developed in the VDT environment. The diagram itself is clearly static, but by linking to both symbolic emulation and discrete-event simulation, the extended representation is dynamic, in that time varying states, conditions and results are projected. And because all objects, relations and parameters in this representation are explicit, this formal model of organizational behavior has little ambiguity. The process illustrated in the figure represents a baseline case, in which only the flows of work (e.g., “S/W arch” and “S/W engr” tasks) and information (e.g., between actors performing the “S/W arch” and “S/W engr” tasks) are modeled. Each actor is instantiated with a specific set of skills (e.g., “software architecture,” “software engineering”) and level of experience (e.g., “medium”), which remains constant throughout the simulated period of process performance; that is, *all knowledge flows are assumed to have completed before the simulated project begins*. This represents the point of departure for our present research.

DYNAMIC KNOWLEDGE-FLOW BEHAVIOR

In this section, the computational knowledge-flow model from above is used to simulate the behavior and performance of knowledge work associated with software development. We first simulate the baseline model instantiated above to establish a basis for comparison. We then develop some related models to illustrate our dynamic representation of knowledge flows, in addition to their workflow and information-flow counterparts.

Baseline Model Dynamics

Table 1 summarizes some of the key simulation performance projections after 100 simulated runs (i.e., using the Monte Carlo technique) for the baseline model (Column 2). This includes both the software-architecure and –engineering tasks. Detailed derivation and description of the various attributes (Column 1) can be found in Jin and Levitt (1996). As shown in this table, the baseline software development work is projected to require just over 11 months to complete and cost $1187K. The Schedule Growth Risk attribute compares the projected duration of the project with the duration that would be achievable through Critical Path Analysis; that is, if the task were to take only the amount of time scheduled (i.e., 260 days), this attribute would be zero. In the baseline case, coordination, rework and delay cause the simulated schedule for this task to grow by 80 days. Attributes such as these are quite common among contemporary simulation models.

Alternatively, Functional Quality Risk summarizes the amount of work that would be required to correct all functional defects associated with the software work; if all such defects were reworked and fully corrected, this functional quality risk attribute would be zero. In the baseline case, empirically determined factors such as the likelihood of errors (e.g., design errors, coding bugs) and probability of correction by organizational actors suggest the project effort would have to grow by 41% to accomplish all functional rework. This represents an implicit measure of quality, for so many latent errors are likely to adversely affect the performance of the software product released without such errors having been corrected. With insight into quality such as through this attribute, our dynamic VDT representation goes beyond most simulation models.

Table 1 Simulated Performance Comparison

Attribute	Baseline	Sequent Train NL	Sequent Train L	Concur Train L	Team Building
Duration (mo)	11.1	**13.6**	**12.4**	**14.6**	**11.4**
Cost ($K)	$1187	**$1212**	**$1041**	**$1125**	**$908**
Schedule Growth Risk (days)	80	89	**54**	**102**	**0**
Functional Quality Risk (%)	41%	41%	41%	41%	41%
Project Quality Risk (%)	41%	42%	41%	41%	42%
Rework Fraction (%)	35%	33%	**18%**	24%	**13%**
Max Backlog Arch (days)	13	13	**6**	**84**	5
Max Backlog Engr (days)	13	13	9	9	**31**

The Project Quality Risk factor is similar, except it measures risk at the project level (i.e., rework associated with coordination between tasks; e.g., software architecture and engineering) as opposed to the functional (i.e., task) level. In the baseline case, empirically determined factors such as the likelihood of inter-task errors (e.g., poor software specification, incorrect design document) and probability of correction by the organization as a whole (i.e., *team* of software engineers, project lead) suggest the project effort would have to grow by (coincidentally also) 41% to accomplish all project rework. As with functional quality above, this attribute also measures an aspect of quality. Although such levels (i.e., 41% for functional quality, 41% for project quality) would be cause for concern in domains such as manufacturing, chemical processing or construction, quality risk levels are generally higher for service industries than their tangible-product counterparts, and these levels for software development are normal (i.e., empirically determined). As indicated by the Rework Fraction, 35% of the cost associated with the software project is comprised of rework (e.g., correcting design errors, fixing programming bugs); again, this level is expected in the domain of software development.

The Maximum Backlog attributes indicate the architecture actor has nearly 13 days' work in its input queue, and coincidentally, the software-engineering team has roughly the same backlog. This dynamic measure is useful to reveal bottlenecks in a process. Such bottlenecks

often present project managers with tough choices between cost and schedule; that is, to ameliorate the effects of a bottleneck, additional resources (e.g., more software engineers, authorization of overtime work) can be added to speed the processing of work, but doing so often increases project cost. This level of backlog would normally be considered occasion for some concern but not alarm. Many other attributes are projected through simulation, but those summarized in the table represent the key ones for comparison with the knowledge-flow models below.

Knowledge-Flow Models

As noted above, the baseline VDT ontology includes constructs to represent flows of work and information, but the corresponding flow of knowledge represents a new thrust, and the VDT modeling environment does not have an express approach or the appropriate primitive objects to represent knowledge flows. Indeed, we are only beginning to develop concepts and techniques for representing the flow of knowledge through static models (e.g., Nissen 2002). However, a key concept from such static models provides sufficient conceptual linkage for us to proceed: the flow of knowledge is (nearly always) prerequisite to the performance of work. Hence the same kinds of precedence relations modeled through VDT using connections between work tasks offers an approach to extending this modeling environment to incorporate knowledge flows.

For instance, consider the software-engineering work task from the model above. This task is shown in Figure 4 as being sequentially dependent upon its software-architecture predecessor. This indicates the architecture must be substantially underway before the software-engineering task can fruitfully begin; indeed, modeling contemporary practice in our model, we specify the software-engineering task as beginning when the architecture is 50% complete. This represents the flow work from one task enabling the performance of the next task. We can represent some flows of knowledge in analogous fashion; that is, drawing from Nissen (2002), just as the flow of work is driven by a set of (workflow) processes in the enterprise, so too is the flow of knowledge driven by a different set of (knowledge-flow) processes. We illustrate this approach through four scenarios or cases.

Case 1 – Knowledge Flow through Formal Training. This approach is illustrated in Figure 5, which extends our model from above to incorporate a knowledge-flow task associated with formal training. From our theoretical framework above, formal training represents the flow of relatively explicit knowledge along the epistemological dimension), from an organization (e.g., a firm contracted to perform the training) to an individual (e.g., student) or group (e.g., class) along the ontological dimension, at the distribution phase of the life cycle. Notice in this figure the new task (i.e., labeled "arch train")—say the software-architecture actor requires three months' training to learn a new tool being adopted by the enterprise for specifying architectures—is represented in comparable fashion to the other tasks (e.g., predecessor, successor and assignment links), except we use a different fill color and texture in the figure to show some distinction. We assume the training starts at project inception and must be complete before the software-architecture task can begin, and predictably, the entire schedule for the software development project is extended by a period close to this three-month time span. Simulated performance results for this case are presented in Column 3 of the table (i.e., labeled "Sequent Train NL") above for comparison with the baseline model.

As would be expected for addition of a new sequential task, both the project duration (i.e., now 13.6 months) and cost (i.e., now $1212K) reflect appreciable change; these are highlighted in bold print for contrast. Some of the other values are slightly different due principally to random variations in the simulation runs and the addition of the training (knowledge-flow) task. Despite dynamically modeling the additional time and cost associated with this knowledge-flow process, however, the case does not reflect any *benefit* in terms of new knowledge acquired by the software-architecture actor; that is, this scenario does not show improvement in the actor's skill level. If this situation were to become manifest, then there would be little reason to invest in the training course.

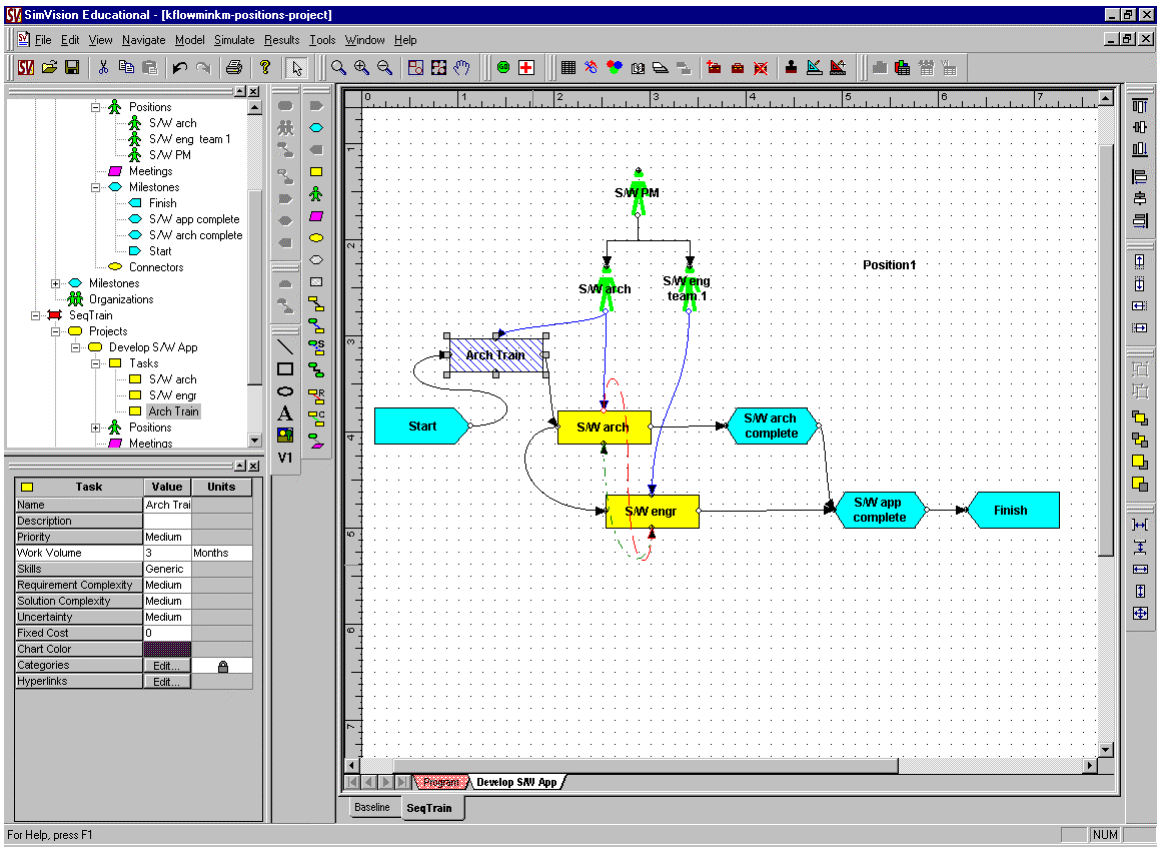

Figure 5 VDT Software Development Model – with Sequential Architecture Training

Case 2 – Knowledge Flow through Formal Training with Learning Effect. The fourth column in Table 1 (i.e., labeled “Sequent Train L”) includes the knowledge-flow benefit: after completing the training course, the software-architect actor has an increase in skill level (i.e., from “medium” to “high”) and performs better on the architecture task. Notice the knowledge-flow (i.e., learning) effect brings the schedule growth down to 54 days, and both project duration (i.e., 12.4 months) and cost (i.e., $1041K) also reflect this learning. Notice also the level of rework is down considerably (i.e., to 18%), as is the architecture actor’s backlog (i.e., to 6 days). These effects derive from the improved knowledge and corresponding work performance of the architect. Alternatively, this training course has no effect on the software-engineering team and its performance of the other project task; that is, the knowledge flow is restricted to an individual. Hence the overall simulation results do not show dramatic improvement. Indeed, even with the knowledge-flow effect, the project costs more and takes longer to complete with the training

course than in the baseline case summarized above. This represents another good point of comparison.

Case 3 – Knowledge Flow through Formal Training with Concurrent Knowledge and Work. Continuing with the architecture-training example from above to illustrate the incorporation of a knowledge-flow process, Figure 6 shows the same situation (i.e., actor takes the training course, actor's skills improve), except to "save time," the software-architect actor is expected to accomplish the training concurrently (e.g., via an online course accomplished in the office) with his assigned work task. The project leader figures this will reduce the project time by the three additional months represented by the architecture course. And given the knowledge flow associated with the course, this leader expects the project to even beat its original schedule.

From the static representation presented in the figure, this scenario reveals almost no difference with that in Figure 5. For instance, the reader can discern the architecture-training (knowledge-flow) task is no longer connected to the software-architecture (workflow) task; that is, the architecture work task no longer has to wait for the architecture training to be completed. Instead, architecture training (i.e., knowledge flow) and architecture development (i.e., workflow) are now performed concurrently, and both are now prerequisite to the software-engineering task. Other than this, the models are equivalent.

The simulated performance is summarized in Column 5 of the table (i.e., labeled "Concur Train L") above. Notice the project duration increases to 14.6 months, and project cost increases to $1125K. Contrary to the project leader's plan, requiring the software-architect actor to learn the new architecture-specification system while performing the architecture-development task itself requires more time and money than sending the person to the course for three months. This kind of result is not apparent from the static representation, and although the result is intuitive for many managers and organization scholars, the logic may not be immediately apparent. Notice the backlog for the software-architect actor has increased to 84 days. Consistent with practical experience, when a knowledge worker is required to accomplish two tasks in concurrence—as opposed to serially—a boundedly rational individual (e.g., limited cognitive capability) with constrained resources (e.g., a 40-hour workweek) may experience more difficulties *with both*

tasks and expend more time and effort overall. Such insights are nearly impossible to derive from static representations and verbal descriptions.

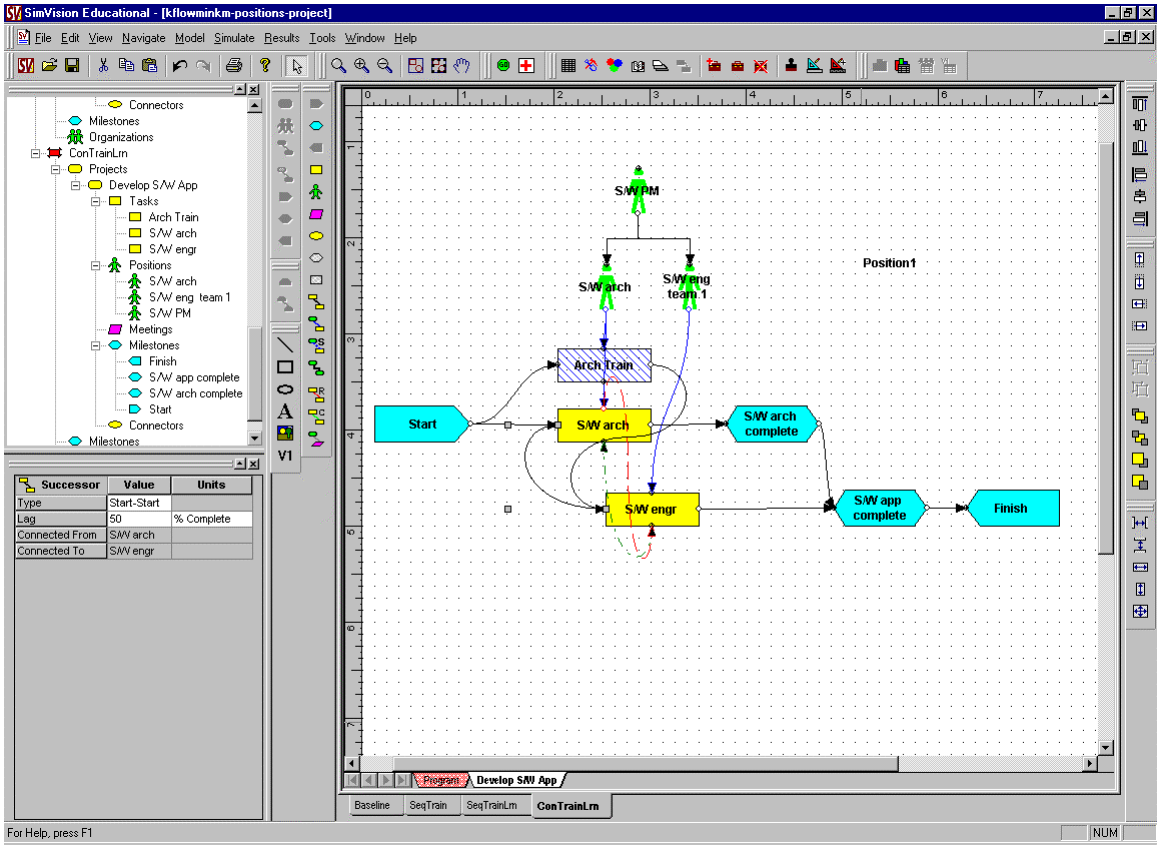

Figure 6 VDT Software Development Model – with Concurrent Architecture Training

Case 4 – Knowledge Flows through Formal Training and Team Building. Figure 7 displays our final scenario, in which the equivalent of one month's time is allocated for members of the software-engineering team to meet, share experiences and engage in the kinds of non-project activities often associated with a community of practice; this instantiates the socialization process from our theory above. As with the architecture-training course discussed above, this team-building activity pertains to the flow of knowledge as opposed to the flow of work—indeed, many managers consider such time to be wasted. Yet, as with the architecture training, some benefits in terms of team learning are expected. This represents a difficult scenario to model using the VDT environment, for we have no way to represent actors' skill levels changing during the course

of a task (e.g., software engineering). A current research goal includes extending the VDT modeling environment to address such situations.

To “trick” the current modeling environment, we separate the software-engineering task into two phases—one can think of these as separate *software releases*, a common practice in software development—and we represent two software-engineering teams: the first one (i.e., labeled “S/W eng team1”) is as before, with medium-level software-engineering skills and low team experience; the second team (i.e., labeled “S/W eng team Lrn”) is used to reflect the benefits this same group may enjoy from its investment in team building (e.g., high-level skills, higher team experience). The other elements delineated in Figure 7 remain the same as in the sequential-training case above (i.e., Case 2); that is, we discard the concurrent-training scenario (i.e., Case 3) as a bad idea.

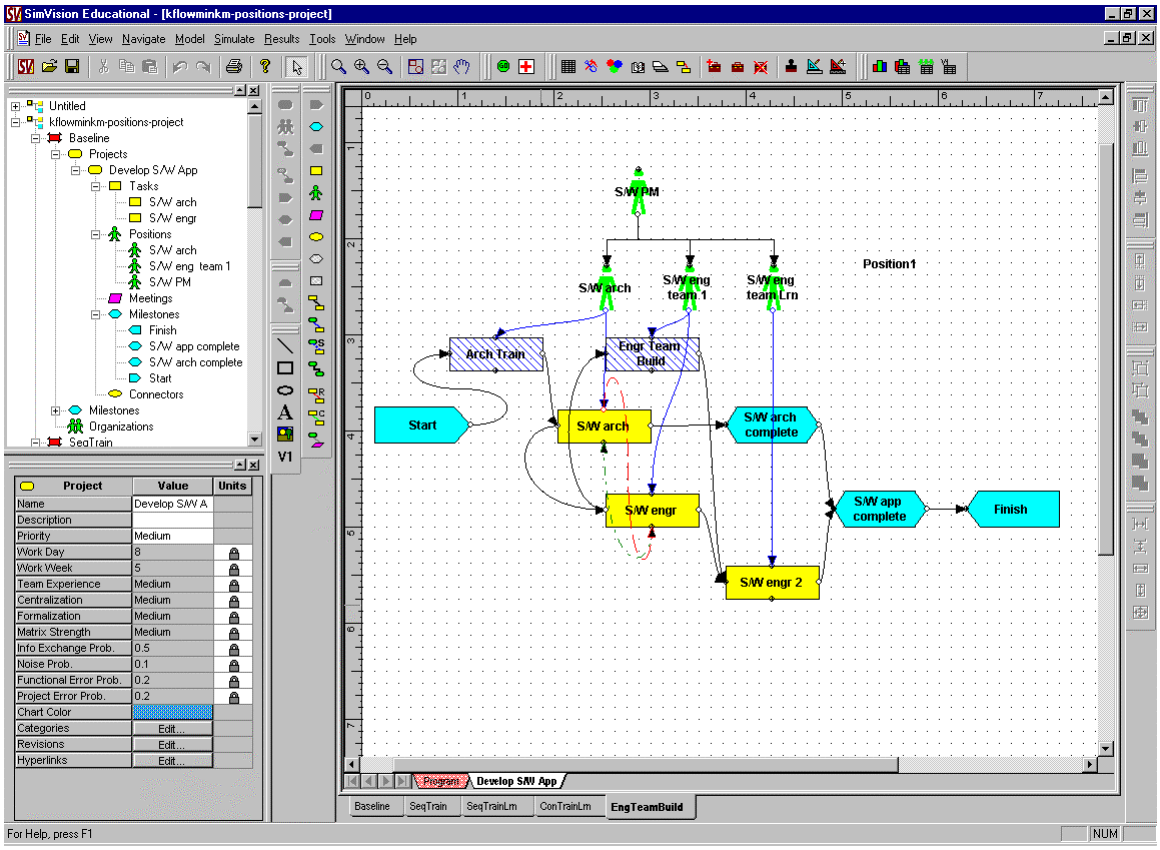

Figure 7 VDT Software Development Model – with Engineer Team Building

Notice the simulated results in Column 6 of Table 1 (i.e., labeled “Team Building”). Despite introducing some slack into the software engineers’ work schedule, the overall project duration contracts to 11.4 months; this is roughly equivalent to the original schedule that included no knowledge-flow processes (e.g., no architecture training, no team building). And because the knowledge and performance levels of both the software-architect actor and software-engineering team have increased, the project cost *is less than the baseline* (i.e., $908K). One key is the rework statistic, which has dropped to 13%. And notice the schedule growth of zero; this indicates the project has now converged onto the critical path: another improved result over the original project baseline. Alternatively, one can see the effect of the engineers’ investment in team building through its increased backlog (i.e., 31 days); this quantifies the intuitive notion of mounting project work that accumulates as team members are engaged in non-project (i.e., team-building) knowledge-flow activities.

CONCLUSION

Knowledge is unevenly distributed through most enterprises, so knowledge flow (e.g., across time, location, organization) is critical to organizational efficacy and performance under a knowledge-based view of the firm. Although *knowledge flow* is an inherently dynamic concept, however, the corresponding phenomenon remains poorly understood, and extant approaches to its modeling and description (e.g., natural language texts and figures) are fundamentally static and largely ambiguous. In this research, we build upon emerging theory for multidimensional conceptualization of the knowledge-flow phenomenon to develop *dynamic* models of knowledge-flow dynamics.

Drawing from recent advances in computational organization theory, we describe a research approach and modeling environment that enables the dynamics of enterprise knowledge flows to be formalized through computational models. Particularly when compared to extant descriptive theory articulated through natural language, such formal models are considerably less ambiguous, more reliable and quite explicit. We illustrate this research approach and modeling environment through formal representation and simulation of several knowledge-flow processes from the domain of software development. When used in conjunction with current theory, the new

insight into and understanding of knowledge-flow dynamics revealed through this research is compelling and represents a contribution to the information systems literature.

Moreover, many new directions for the kind of research elucidated by this work are now explicit. For instance, we note above how work on the VDT modeling environment is needed to improve its capability for representing dynamic knowledge-flow processes (e.g., team learning, on-the-job experience), and comparison of VDT with other approaches and tools from the computational organization theory field can help us identify the most promising techniques. As another instance, the dynamics of knowledge flow imply some change in knowledge is taking place, but it is unclear how such change relates to the literature on learning, either at an individual or organizational level.

Further, new research to map various rates of knowledge flow (e.g., through improvement theory) to alternative knowledge-flow processes (e.g., using our multidimensional model) may shed considerable light on a poorly understood dynamic phenomenon, and the kinds of tools discussed in this article provide the means for developing new micro-theory to identify and (virtually) test which kinds of managerial interventions (e.g., IS, training, organization changes, workflow reconfigurations) are most effective in specific enterprises, organizations and processes. Through the kind of intellectual exchange facilitated by this conference, we hope to help stimulate and guide a fruitful and focused program of continued collaborative research along these lines.

REFERENCES

Alavi, M. and Leidner, D.E., "*Review*: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly* 25:1 (2001), pp. 107-136.

Augier, M., Shariq, S.Z. and Vendelo, M.T., "Understanding context: Its Emergence, Transformation and Role in Tacit Knowledge Sharing," *Journal of Knowledge Management* 5:2 (2001), pp. 125-136.

Carroll, T. and Burton, R.M., "Organizations and Complexity: Searching for the Edge of Chaos," *Computational & Mathematical Organization Theory* 6:4 (December 2000), pp. 319-337.

Cheng, C.H.F., and Levitt, R.E., "Contextually changing behavior in medical organizations" *Proceedings of the 2001 Annual Symposium of the American Medical Informatics Association,* Washington, DC (Nov 3-7, 2001).

Christiansen, T.R., *Modeling Efficiency and Effectiveness of Coordination in Engineering Design Teams* doctoral dissertation, Department of Civil and Environmental Engineering, Stanford University (1993).

Cohen, G.P., *The Virtual Design Team: An Object-Oriented Model of Information Sharing in Project Teams* doctoral dissertation, Department of Civil Engineering, Stanford University (1992).

Cole, R.E. “Introduction,” *California Management Review* 45:3 (Spring 1998), pp. 15-21.

Davenport, T.H., *Process Innovation: Re-engineering Work through Information Technology* Boston, MA: Harvard University Press (1993).

Davenport, T.H., Jarvenpaa, S.L. and Beers, M.C. "Improving Knowledge Work Processes," *Sloan Management Review* (37)4 (1996), pp. 53-65.

Davenport, T.H., De Long, D.W. and Beers, M.C"Successful Knowledge Management Projects," *Sloan Management Review* (39)2 . (1998), pp. 43-57.

Davenport, T.H. and Prusak, L. *Working Knowledge: How Organizations Manage what they Know*. Harvard Business School Press: Boston, MA (1998).

Despres, C. and Chauvel, D. "Mastering Information Management: Part Six – Knowledge Management," *Financial Times* (8 March 1999), pp. 4-6.

Dixon, N.M., *Common Knowledge* Harvard Business School Press: Boston, MA (2000).

Galbraith, J.R., *Organization Design*, Addison-Wesley, Reading, Massachusetts (1977).

Gartner Group. "Knowledge Management Scenario," conference presentation, Stamford, CN, presentation label SYM8KnowMan1098Kharris (1998).

Grant, R.M., “Toward a Knowledge-Based Theory of the Firm,” *Strategic Management Journal*17 (1996), pp. 109-122.

Jin, Y. and Levitt, R.E., “The Virtual Design Team: A Computational Model of Project Organizations,” *Computational and Mathematical Organization Theory* 2:3 (1996), pp. 171-195.

King, W.R. and Ko, D.G., “Evaluating Knowledge Management and the Learning Organization: An Information/Knowledge Value Chain Approach,” *Communications of the Association for Information Systems* 5:14 (2001), pp. 1-27.

KPMG Management Consulting, *Knowledge Management Research Report* (1998), cited in Alavi and Leidner (2001).

Law, A.M. and Kelton, D., *Simulation Modeling and Analysis* (Second Edition) New York, NY: McGraw-Hill (1991).

Leavitt, H.J. "Applying organizational change in industry: structural, technological and humanistic approaches," in J. March (Ed.), *Handbook of Organizations* Chicago, IL: Rand McNally (1965).

Levitt, R.E., Thomsen, J., Christiansen, T.R., Junz, J.C., Jin, Y. and Nass, C., “Simulating Project Work Processes and Organizations: Toward a Mocro-Contingency Theory of Organizational Design,” *Management Science* 45:11 (1999), pp. 1479-1495.

Kunz, J.C., Levitt, R.E. and Jin, Y., “The Virtual Design Team: A Computational Simulation Model of Project Organizations,” *Communications of the Association for Computing Machinery* 41:11 (1998), pp. 84-92.

March, J.G. and Simon, H.A., *Organizations*, John Wiley, New York (1958).

Nissen, M.E. "Knowledge-Based Knowledge Management in the Re-engineering Domain," *Decision Support Systems* 27, Special Issue on Knowledge Management (1999).

Nissen, M.E., Kamel, M.N. and Sengupta, K.C. "Integrated Analysis and Design of Knowledge Systems and Processes," *Information Resources Management Journal* 13:1 (January-March 2000).

Nissen, M.E., "An Extended Model of Knowledge-Flow Dynamics," *Communications of the Association for Information Systems* 8 (2002), pp. 251-266.

Nogueira, J.C., *A Formal Model for Risk Assessment in Software Projects* doctoral dissertation, Department of Computer Science, Naval Postgraduate School (2000).

Nolan Norton Institute, “Putting the Knowing Organization to Value,” white paper (August 1998), cited in Alavi and Leidner (2001).

Nonaka, I. "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science* 5:1 (1994), pp. 14-37.

Nonaka, I., Takeuchi, H. and Umemoto, K., “A Theory of Organizational Knowledge Creation,” *International Journal of Technology Management, Special Issue on Unlearning and Learning for Technological Innovation* 11:7/8 (1996), pp. 833-845.

O’Leary, D.E., “How Knowledge Reuse Informs Effective System Design and Implementation,” *IEEE Intelligent Systems* (January/February 2001), pp. 44-49.

Polanyi, M. *The Tacit Dimension* Routledge and Keoan Paul: London (1967).

Ryle, G., *The Concept of Mind* London: Hutchinson (1958).

Schultze, U. and Boland, R.J., “Knowledge Management Technology and the Reproduction of Knowledge Work Practices,” *Strategic Information Systems* 9 (2000), pp. 193-212.

Spender, J.C. “Making Knowledge the Basis of a Dynamic Theory of the Firm,” *Strategic Management Journal* 17 (1996), pp. 45-62.

Swap, W., Leonard, D., Shields, M. and Abrams, L., “Using Mentoring and Storytelling to Transfer Knowledge in the Workplace,” *Journal of Management Information Systems* 18:1 (2001), pp. 95-114.

Szulanski, G., “Exploring Internal Stickiness: Impediments to the Transfer of Best Practice Within the Firm,” *Strategic Management Journal* 17 (1996), pp. 27-43.

Szulanski, G., “The Process of Knowledge Transfer: A Diachronic Analysis of Stickiness,” *Organizational Behavior and Human Decision Processes* 82:1 (2000), pp. 9-27.

Teece, D.J., "Research Directions for Knowledge Management, *California Management Review* (40)3 (Spring 1998), pp. 289-292.

Thomsen, J., *The Virtual Team Alliance (VTA): Modeling the Effects of Goal Incongruency in Semi-Routine, Fast-Paced Project Organizations* doctoral dissertation, Department of Civil and Environmental Engineering, Stanford University (1998).

Thompson, J. D., *Organizations in Action: Social Science Bases in Administrative Theory*, McGraw-Hill, New York (1967).

VDT. The Virtual Design Team Research Group website; URL: http://www.stanford.edu/group/CIFE/VDT/ (2002).

Venzin, M., von Krough, G. and Roos, J., "Future Research into Knowledge Management," in: G. von Krough, J. Roos and D. Kleine (Eds.), *Knowing in Firms: Understanding, Managing and Measuring Knowledge* London: Sage (1998), pp. 26-66.

Von Hippel, E., "'Sticky Information' and the Locus of Problem Solving: Implications for Innovation," *Management Science* 40:4 (1994), pp. 429-439.

Wong, S.S. and Burton, R.M., "Virtual Teams: What are their Characteristics, and Impact on Team Performance?" *Computational & Mathematical Organization Theory* 6:4 (December 2000), pp. 339-360.

Zack, M., "What Knowledge-Problems Can Information Technology Help to Solve," in: E. Hoadley and I. Benbasat (Eds.), *Proceedings* Americas Conference on Information Systems, Baltimore, MD (1998), pp. 644-646.