

**The Politics of Knowledge Work  
in a Software Development Group<sup>1</sup>**

Andrew Hargadon  
Graduate School of Management  
University of California, Davis  
abhargadon@ucdavis.edu  
530.752.2277

Beth A. Bechky  
Graduate School of Management  
University of California, Davis  
babechky@ucdavis.edu  
530.752.0911

2/28/03

Forthcoming in  
Advances in Qualitative Organization Research

---

<sup>1</sup> This work was done with the support of ICAIR

## Abstract

This paper describes the knowledge work of a group of software programmers and how that work is shaped by the ongoing social context of the work and the organization. Over time the knowledge of the organization, as captured in the written programs and the experiences of the programmers, evolves in subtle yet distinct directions. Evidence from an in-depth field study of this software group reveals three politicized activities — help-giving, help-seeking, and sanctioning — which enabled and directed this evolution. We first describe how the knowledge of this software group evolves across people and programs and the obstacles to sharing knowledge inherent in this work; we then explicate the role of help-seeking, help-giving, and sanctioning in this evolutionary process.

## **The Politics of Knowledge Work in a Software Development Group**

What are people doing when we say they are doing knowledge work?

Increasingly, organizational knowledge has become the central explanation for outcomes of organizational performance, learning, innovation and competitive advantage (c.f., Eisenhardt and Santos 2001). In this context, however, the role of the knowledge worker is underdeveloped. By most accounts, knowledge workers acquire, store, retrieve, combine, create, buy, sell, transfer, or share knowledge. Such language builds on traditional understandings of organizations as factories where knowledge is produced and markets where it is traded (Katz, Kahn, and Adams 1980; Morgan 1997). Knowledge workers engage in activities of production and exchange, yet these activities take place within the web of social interactions that give shape to work in organizations. We can better understand the notion of organizational knowledge by understanding the work that creates it and how that work is in turn a product of the social and political dynamics of the organization.

Traditional approaches to knowledge work draw from our established conceptions of production and exchange as the dominant activities of organizations. A focus on the production of new knowledge, for example, invokes images of “invention factories” and suggests such work involves a set of core processes that transform inputs (existing knowledge) to outputs (new knowledge).<sup>2</sup> More directly focused on knowledge work, Nonaka (1994) and Nonaka and Takeuchi (1995) describe how new knowledge is created in the conversion of existing knowledge between tacit and explicit. For example, an engineer at Matsushita designing a breadmaker converted the tacit knowledge of a baker into her own and then, combining it with her prior knowledge, made it explicit for the design team. More generally, the traditional stage models of the innovation and product development literatures also reflect assumptions about the sequence of operations necessary to produce innovative solutions (Wolfe 1994). Understandings of knowledge

---

<sup>2</sup> And while early images of modern organizations have relied on mechanistic imagery (Katz, Kahn & Adams, 1980; Morgan, 1997), the shift in attention from organizations producing goods to those providing services, and from production work to managerial work, has shifted the imagery from the greasy, gear-laden machines of the factory floor to the clean, digital information processing machines of the office (e.g., Galbraith, 1974; Scott, 1998).

work built on the factory metaphor allow us to see how knowledge, once it is externalized and objectified, can be shared and shaped.

Implicit in the factory perspective is that knowledge work can be understood and improved by studying these core processes apart from their surrounding social milieu. However, this perspective downplays the ways in which knowledge reflects the social and political dynamics of the organizations in which it resides. The social context of organizations can powerfully shape the process and product of knowledge work (Kogut and Zander 1996; Orlikowski 1992; Orlikowski and Gash 1994). Kogut and Zander (1996), for example, describe how the micro-social dynamics of organizations shape the processes of knowledge replication because those organizations, as social systems, provide individuals with identities that strongly shape their interpretations of the environment. These shared identities facilitate the exchange of predominant ideas while, at the same time, potentially undermine the ability to recognize alternative ones. As this emerging perspective suggests, production models of knowledge work neglect how the organizational context, as an ongoing social system, shapes both the process and outcome of this work.

Another perspective, the exchange metaphor, approaches knowledge work from existing models of markets and networks, in which goods and services move between participants. The exchange perspective has been most extensively mapped by transaction cost theories, which describe knowledge work as an internal market in which individuals buy, sell, and trade their knowledge (Coase 1937; Williamson 1975). This perspective both builds on and diverges from machine metaphors, as exemplified by Williamson's (1981: 552) description of the role of exchanges within the firm:

With a well-working interface, as with a well-working machine, these transfers occur smoothly. In mechanical systems we look for frictions: do the gears mesh, are the parts lubricated, is there needless slippage or other losses of energy? The economic counterpart of friction is transaction cost: do the parties to the exchange operate harmoniously, or are there frequent misunderstandings and conflicts that lead to delays, breakdowns, and other malfunctions?

This perspective, adapted in knowledge-based views of the firm, suggests that knowledge work can be understood by understanding how the organization's market characteristics shape the exchange of knowledge within the firm. Grant (1996: 112) argues, for example, that firms exist in "response to a fundamental asymmetry in the economics of

knowledge.” A similar perspective resides in social network analyses of organizational work and, in particular, in studies of the problem solving practices of professionals (Burt 1987, 1992a, b; Hansen 1999) and professional firms (Powell 1990; Powell and Brantley 1992). Hanson (1999) for example, explains the flow of knowledge across a large organization by considering the comparative costs and benefits of maintaining strong and weak ties to other project teams.

This work adopts a utilitarian and economic view of individual social ties and the transfer of resources between them. From this perspective, firms and the actors within them buy, sell, transfer, withhold, and in other ways manipulate the flow of knowledge as goods throughout the firm. An exchange perspective highlights the value of knowledge and its uneven distribution within (and across) firms, yet does so at the expense of understanding the social processes needed to produce and reproduce that knowledge. For example, Dougherty and Hardy (1996) and Bechky (2003) have described how functional boundaries and diverse occupational perspectives make it difficult to transfer ideas across the organization—not because of asymmetries in the distribution or gaps in the flow of knowledge but rather because the same information means one thing in one occupation and something wholly different in another. Efforts to facilitate the flow of knowledge across organizations informed by purely economic perspectives will founder on the political entanglements of such communities.

Understanding knowledge work requires acknowledging the micropolitical dynamics that, while neglected by models of production and exchange, nevertheless remain defining characteristics of organizations. Micropolitics refers to the local motivations, decisions, and actions that constitute managers’ pursuit of influence, status, and opportunity within organizations (Burns 1961). Such micropolitics not only shape the way knowledge is shared in organizations, but shape the knowledge itself. The construction of new knowledge and re-construction of old takes place within the micropolitical context of organizations. For example, research in the constructivist perspective (Berger and Luckman 1967; Meindl, Stubbart, and Porac 1994; Weick 1995) reminds us that knowledge does not exist outside of the social systems that produce and sustain it. Similarly, studies of the socially embedded nature of economic action demonstrate how the ends and means of individuals are shaped as much by cultural as

economic influences (Dacin 1997; Dacin, Ventresca, and Beal 1999; Granovetter 1973; Ventresca et al. 2000) And while such work has enriched discussion of epistemology or economic action, its value also extends to enriching our understandings of organizational work. In the same way that a perspective of social construction locates the production of knowledge within the context of larger societal systems, this view should help to ground the production of knowledge within the ongoing micropolitics of organizations. Similarly, while a social embeddedness perspective grounds economic actions within societies, this view may also help ground the exchange of knowledge across the firm within the ongoing life of an organization.

This paper uses a field study of work within a software group to explore two questions: how does the micropolitics of organizational life shape the evolution of knowledge in organizations? Existing research suggests that the acquisition or development of new knowledge — whether new products, work practices, or new shared understandings of organizational members — is as susceptible to the micropolitical dynamics of organizations as it is to the efficiencies of production and exchange (Orlikowski 1992; Orlikowski and Gash 1994). Capturing the effects of such dynamics requires the study of both the everyday work practices of organizational members and the fine-grained details, the nuts and bolts, of the technical work they produce. Thus this paper moves between descriptive accounts of the technical work of the software group and explanations of social and political dynamics that both shape and are shaped by this work. We first describe how the knowledge of this software group evolved across people and programs and the obstacles to sharing knowledge inherent in the process. We then identify three core activities of this evolutionary process—help-seeking, help-giving, and sanctioning—and explicate the micropolitical dynamics that drive individuals to participate in these activities.

## Methods

**Research Site: The MarketModels group.**<sup>3</sup> We gathered the data for this study in the winter and spring of 2000 at the Menlo Park, CA office of a mid-sized marketing consulting firm headquartered in Chicago. The firm’s work focuses on issues of sales

---

<sup>3</sup> A pseudonym.

strategy, product forecasting, and market research within the pharmaceutical industry. Because of our interest in determining how knowledge work unfolds within a community, we located a group within the organization whose work was relatively specialized and distinctive: the MarketModels group. This allowed us to neatly bound the group of informants and gather in-depth information about the knowledge sharing of every individual in the community.

MarketModels develops market forecasts of drug adoption curves and revenue projections for its clients. The group began in 1993 with the efforts of a single consultant who, in response to a client request, developed a revenue projection for a new drug the client was considering for introduction. The revenue projection was based on a complex model that accounted for the nature of the affected population, the diagnosis, prescription, and a number of other market variables that are relatively unique to each drug/disease pairing. The client's satisfaction with this original model led to the development of similar, increasingly refined models. Over time, additional consultants were transferred or hired to develop more such models. By the time of our study, this group was a distinct office of 15 forecast modelers located in Menlo Park. Now, the typical project team comprises two to four people, and project duration averages 6-8 weeks.

**Informants.** We interviewed all 15 members of the MarketModels group for this study. Every member was interviewed at least once, and we conducted repeated interviews with many informants over the course of 5 months. The informants ranged in status from two firm partners to two business analysts, with the bulk of the informants at the consultant and manager levels. While the two analysts were hired directly out of college, the remainder of the informants had graduate degrees; their educational backgrounds were quite varied, including a doctorate in biochemistry, a masters degree in statistics, and a number of MBAs. Several of the consultants had prior experience at other consulting firms.

**Interviews.** Interviews with informants were semi-structured and lasted between one and two hours. All interviews were face to face with the exception of several phone interviews with one informant who telecommuted, and thus was not available in the office. Both authors were present at the interviews, all but one of which were audiotaped.

The initial background questions centered on the type of work done at MarketModels and the structure and organization of project teams. Then we focused on questions intended to elicit data about how the social dynamics of the organization affected the evolution of knowledge. For instance, in order to gather information about how individuals provided one another with project-related knowledge, we had our informants describe their work experiences as members of project teams, concentrating on their preferred means of communication with others and the colleagues each chose to communicate with on a regular basis. These interviews were open-ended in that we encouraged informants to focus on the aspects of sharing knowledge they felt were most important to accomplishing their work, and we followed where they led.

The follow-up interviews emerged from what Spradley (1979) would call “task-related tour” questions, inquiries that focused on particular tasks that informants performed during the course of a project. Because we were interested in the way communication within projects unfolded, we were able to elicit more concrete data by directing our informants to describe particular projects on which they had worked. This entailed a discussion of the details of a particular project, in which each informant would pick a current or recent project, show us the models or outputs from the project, and talk us through how these developed over time. Because many of our informants worked together on projects, we gathered data from multiple informants about some of the projects, which allowed us to corroborate their understandings of project development. As a result of these task-focused interviews, we have also gathered documentary evidence of the work practices and outputs, including spreadsheet models, presentation overheads, website data, and even a book published by a member of the firm. Additionally, we unobtrusively observed the work patterns of the office during the duration of our visits there over the course of 5 months.

**Data analysis.** Both authors performed the data analysis, which followed an iterative process that compared data, existing literature, and emerging ideas. Analytical categories were developed through an initial joint coding of the data, and we individually compared these categories with the extant literature on knowledge sharing. We then jointly examined all the instances of each category; identification of important themes was based on discussion and agreement among both authors.

Stage One. After the first set of interviews, it became clear that the members of MarketModels thought of their projects as depending on a continuous evolution of the knowledge gained during previous projects. Therefore, we focused our initial data analysis on describing and analyzing this evolution, developing the category “drivers of model changes.” This category included various subcategories of influences on this evolution such as client behavior, manager influence, previous skills, and deadline pressures. We each combed the data for instances of these categories and jointly examined them to determine the prevalent themes.

Stage Two. Similarly, we jointly refined our analysis of the micropolitical influences on such evolution, further developing our theoretical categories by comparisons with the growing body of data and the literature. Here, we focused on the themes of helping, help seeking, and feedback. Our turn to the literature indicated that while there is an understanding of helping behavior, it often seemed to lack a context, while our data was indicating that helping behavior was triggered by help seeking. This helped us to refine our focus to investigate more thoroughly the contingencies involved in help seeking, and to determine how it related to help giving. The theme of sanctioning feedback also emerged from our data, as our informants frequently mentioned behavior of this type as either supporting or constraining offers to help and share knowledge.

In this paper, we first present an analysis of the nature of work done by the consultants at MarketModels, which will be described as a process of evolving models. These evolving models incorporate and build on the knowledge gained from previous models as well as develop new ideas to add to the current generation of models. However, this knowledge does not evolve in a vacuum. Instead it is embedded in the social and political relations of the consultants themselves. Therefore, the second half of our findings focuses on the interpersonal processes that contribute to the evolution of these models: the help giving, help seeking, and sanctioning behaviors that shape and constrain the development of models at MarketModels.

### **The Evolution of Knowledge at MarketModels**

For each client project, MarketModels develops a customized software application that accounts for the particulars of a disease, the medication, and the medical

environment and provides an interactive market and revenue projection that these companies can use in their strategic and financial planning decisions. MarketModels develops approximately 20-30 such software models per year. Each model is both new and old: new in that each model is customized to the particular needs of the client or drug, yet old in that each is constructed from the forecasting heuristics, assumptions, and sometimes even lines of programming code developed for past models.

The consultants at MarketModels represent the process of developing new models as one of evolving knowledge: using earlier models as a basis for new models was represented in the short-hand jargon of programmers as “open/save-as,” reflecting the software commands that sequentially open an existing file from a previous project and save it as a new file for the current project. One consultant described the set of questions that begin each new project, saying,

You can easily pick a base model to start from, a model that is as close as possible to what you want and use that as a base model—that works very well. Saying, ‘I’m going to start from Kim’s base model, or John’s base model or my old Pharma model’ or something like that. I’ll start with this and piece in other things that I need.

To this newly saved file, additional components are created or imported from other models. To this source model he might add other components that, for example, his manager is aware of and views as relevant, like a previous model that dealt with the same class of drugs or diseases. As one manager suggested:

For pharmaceutical companies I know what the therapy area is, I know what the product is. So, on a project that has particular demands and particular pieces of knowledge that need to be incorporated – it’s pretty easy for me to say, “Go talk to Kate.” I may not be able to tell you that Kate built the response curve that operates in this manner and integrates these types of things, but I know Kate built the response curve. I will say, “Kate can give you the details, make sure you sit down with her.”

Another means of identifying potentially valuable components from other models was through casual conversations with other consultants. As one consultant told us: “There’s a group of people who take the train down here from the city... they share methodology on the train.” As programmers encountered problems and as clients changed their specifications, the option to write original code or to draw from the work of others continuously presented itself. As another programmer described, “One of the most important decisions we have to make daily is, ‘all right, should I do this myself or should I try to gather somebody’s else’s work?’” Each completed software model, then,

consisted of a number of modules that were taken from previous models and used as is or updated to reflect changes in the coding scheme or structure of the new model.

The nature of MarketModels' products thus provides us with an opportunity to observe the evolution of organizational knowledge at MarketModels both in terms of knowledge embedded in software models and held by the organization's members. Each new model reflects its roots in previous models and its evolution beyond those past models. The use of the term evolution does not imply that each model represents an improvement over past models. Instead, the changes in each new model simply reflect adaptations to the particular conditions of each new project. This is an important point because the nature of evolutionary process is a critical window into the knowledge work that occurs at MarketModels.

### **The Politics of Evolution at MarketModels**

The evolution of knowledge at MarketModels can be seen in the use and re-use of modeling components and conventions. But it is the social and political dynamics between individuals at MarketModels that determine which parts of past models are adopted, thus shaping this evolutionary process. Looking closely at the work and interaction of the programmers reveals a number of obstacles that prevented the "latest thinking" from being incorporated into each new model.

First, much of the knowledge captured as code in the existing software models was not easily decipherable. "It's a little more difficult," a consultant emphasized, "to isolate a certain piece of somebody's model and pick that one piece and pull it into your model." There were instances, for example, when code was never understood but incorporated anyway (even through several generations of models) because the consultants were not sure what it did, but were afraid it might have an integral purpose so did not want delete it. As another consultant described,

We had remnants kind of floating around in the old models. Every time you take one worksheet from an old one and put it in yours, you get all the baggage. In the first model that I did, I had this adoption interface that I thought was really important for this forecasting – all these links, cells and all these columns – so I thought they were really important. I wound up using them somehow because I was taking it from someone else's model. But at some point, I don't know if it was from when I took it from there and used it or when that person took it from the other person before and used it, all the stuff that was there was no longer used at all.

So while it is possible to describe the knowledge of the organization as residing in the software models, this description must be tempered by the recognition that such knowledge is not easily released, but requires effort to fully decipher.

Second, because the knowledge captured by old models was often difficult to grasp, and because each new model needed to be tailored to the particulars of that project, determining *what* to borrow and from *which* past models was equally difficult. Further complicating the process, it was also difficult to determine *how* to borrow from existing work. One programmer explained how he understood and used the code of others:

It's a lot of learning what happened in the past, what's going on here. 'This module makes absolutely no sense to me. What's this code doing?' And cleaning up some of it, adding pieces from other people, and then hopefully finishing up in the next two weeks.

When trying to build on the knowledge of past models, consultants considered actions ranging from using existing code verbatim, to using only the ideas implicit in the code (the steps and procedures called for) but not the actual text, to using the experiences of the designers who wrote those earlier programs, to not using the old ideas at all.

On the one end, there was complete, almost unreflective incorporation of the ideas of past models, as in the following example:

I went to Jinny. I didn't ask her how to use it, I just said where can I get it, and then I added it and it kind of worked really quickly without me doing a lot to it. So I just – there it is. This was really the end of the project, so it was near the deadline, so I wasn't really caring how it worked. I was just caring that it worked.

Yet new models often had different variables than old models, and so programmers had to modify the old code before it could work seamlessly in their model. When these modifications appeared too cumbersome, programmers often used the existing code of a previous model just to guide their own version of the same functions, writing new code informed by their understanding of the old model. Finally, some programmers chose to write new code without using the old knowledge of the group.

Finally, the code only represented a particular solution to a programming problem and did not reveal whether that solution was any good. In order to maintain an exact copy of what the client now had in their possession, each model was kept in its original form. But programmers often talked about how they would have done a number of things differently based on their experiences and the client's reactions. In these cases, if asked about a particular model, the author might reply, "It was not a very elegant solution and

it's going to be really hard to disentangle this from the old answer that we incorporated it into," and then explain what he learned while doing the work. The consultants' experiences with their own models often provided insights which were not reflected in the original ideas and software—and consultants were very cognizant of the imperfections of past programs and how they might be improved, given more time or a more generous client, for example. As a result, the knowledge captured in the code of old models was only the knowledge of what worked for that particular time and place. The code did not incorporate knowledge about what did not work or what might have worked better.

As a result of these obstacles, the knowledge captured in the code, ideas, and experiences of each previous model did not flow easily into each new model. Instead, the re-use of the ideas and code of past models relied on the efforts of the individuals in the organization to share their past knowledge, because such sharing also entailed determining what could and should be shared and how it might best be shared. In the next section we address the question of how and why consultants made these efforts — how the social and political dynamics of MarketModels shaped the flow of knowledge across programs. The evidence identifies three activities that comprised the knowledge sharing process at MarketModels: help-giving, help-seeking, and sanctioning feedback.<sup>4</sup>

### **Help-Giving**

Helping behavior is the willingness of individuals to share what they know with others in the organization, especially when they are not required or expected to do so (Motowidlo, Borman, and Schmit 1997). As related to model-building at MarketModels, our data suggests that consultants helped each other in four ways: by sharing their knowledge as ideas, code, experiences, and references. Sharing ideas was an informal way of providing information about what a consultant was thinking when he created a piece of a model. Consultants shared knowledge more formally when they gave sections of code to one another. They sometimes combined these two modes of helping behavior, and added

---

<sup>4</sup> We present these three activities as distinct behaviors because it fits our data reasonably well and provides a simple and analytically useful way of summarizing these data. Nonetheless, the behaviors that we observed and the behaviors and motivations that were described to us could not always be cleanly distinguished: one person's help-giving behaviors, for example, were a way of sanctioning (reciprocating) help given previously to them.

experiences as well, explaining what they learned (about clients or drugs, for instance) by working on a particular model. Finally, they frequently helped by providing a referral, offering the name of a third party and suggesting what that person thought or experienced when working on a model. The following statement exemplifies help-giving activities:

So I'll ask, "Hey Eric, do you have a model where you do something like this?" And he'll say, "Oh yeah, you should check this model. And also e-mail Charles because he might have another example of how they did this," and then he'll mail me a model.

There appeared to be varying degrees of helping behavior, as related to model-building, within this organization. At one extreme, people would not help, despite knowing help was sought, by either not offering the "answers" they knew or, more subtly, by not taking the time to think about whether they knew something that might be relevant. People offered low amounts of help by reporting that a module or particular solution had been built before, and pointing out its location in the shared model database, leaving the programmer to overcome the obstacles inherent in pulling ideas from the code of past models. Moderate help involved, for example, offering suggestions about how a module they were familiar with might be revised or adapted to fit the new model and situation. Finally, high levels of helping behavior included such efforts as actually fitting an old component or module into the new model someone else was writing.

One constraint on helping behavior was the reputation effects that inhered in the code itself. Sometimes helping behavior might be inhibited by the unwillingness of helpers to share their code with others, either from embarrassment over their own previous work that they regarded as not well done, or because they felt that their work was wrong and that others could do a better job starting from scratch. Much of the original code was written under time constraints (i.e., on the weekend before the deadline), and consultants would often preface sharing their code with warnings that it was not elegantly written, nor perhaps easily adopted without adapting it to fit a new program. Some would even go so far as to add comments within the code apologizing for its poor quality and noting the deadline pressures they were under. All of our informants talked about models they created that were "hacked" together, and indicated that they were not so eager to share them. "If I had done something so hacked that I didn't want anybody to see it," one consultant said, "I'd say 'Oh, you know, I've got this, but do it yourself. Because you don't want to even look at my stuff.'"

Helping behavior was also shaped by the reputations of those seeking and giving help. For instance, one informant mentioned how he offered help more readily depending on the status of those seeking it. There was a line between helping others and doing their work for them, and consultants considered how their help would place demands on their own time. For instance, one consultant described a part of his most recent model:

This is all stuff Charles wrote that literally I just went and dropped it in. I think he actually dropped it in. I don't think I did anything... I think I said, "I'm going to the bathroom." He said it should take five minutes. While I was gone he came over to my computer and did it.

While this example was an easy fix that only demanded five minutes of another consultant's time, at other times, sharing old code might mean committing to more help down the line. As one consultant suggested:

I would say we have a tendency to say, "You know, I just recommend doing it yourself" because if you say, "Oh you should use mine" then that means they're probably going to come back at you in a couple of days and say, "My gosh, this isn't working at all, what's going on, can you look at this? Can you deal with this?" And so you sort of have to support it – if you give somebody a code of your own, if you give somebody something you've done, the goal is to make it as flexible as possible so that other people can use it.

Thus, help-giving was embedded in the social context of the group, and depended upon the considerations each participant made of the social and political context of their helping.

While helping is a critical behavior enabling models to reflect the latest work of the organization, informants spoke frequently of what triggered helping behavior — for example, specific and personal requests for their assistance — and what did not — entering solutions and experiences into a common database, or responding to company-wide emails. In part this may be an argument of individual efficiency. It is unclear that it is worthwhile entering ideas into a database or designing a module that can be reused until there is the specific need to reuse it. But there appeared to also be a tendency to want the credit for helping that comes from receiving personal appeals for help that respect someone's previous work and the current demands on their time. For example, as one consultant pointed out,

I'm happy to help people out, but they're probably going to want to call me first in order to get my attention, in order for me to send this, and I'm not just going to be compelled to put something on to the data base unless somebody's getting after me to do it. Then I'd say, "Oh yeah, yeah, you know what? That is really cool, I will post that and hopefully it will help some people."

So helping behavior was inextricably linked, at MarketModels, to the behaviors of help-seeking and sanctioning.

### **Help-seeking**

While the existing literature pays little attention to help-seeking behaviors, these actions seem to play the crucial triggering role in knowledge sharing, shaping helping behavior. The notion of anticipatory helping, without a clear problem or appreciative audience, is behavior that neither the organization seems to reward nor individuals seem to have time for, as the demands of their own projects always are greater than the time available. Consultants had varying opinions about whether they would seek help on a particular model. When programmers decided to gather somebody else's work, they engaged in a set of help-seeking activities which ranged from walking into the office next door and asking for help to sending an email to the whole office to searching through a CD-ROM library of past models. This central depository that houses the past models was often used, but only after discussions with the previous authors or past consultants who were familiar with those contexts and able to recognize and recommend useful aspects.

At MarketModels, whether and how a person sought help depended on several factors, including individual proclivity and social constraints. Some consultants tended to solve their problems themselves, preferring not to reuse the code of others. This sometimes stemmed from an enjoyment of coding and problem solving, and other times was a matter of not trusting others' work. Time constraints also played a role in the individual decision to seek help, as some consultants perceived the finding, understanding, and translating of other models to be time consuming and effortful. One consultant summarized this issue the following way:

You end up with people who are more willing to rewrite a model that's already been done because they want to have their own model that they're comfortable using and they want to – they're not interested in even using anybody else's stuff because they just don't like the way it's done... I'm perfectly content and know the structure of the models I've done in the past and I know exactly what it takes to adapt them for any issue with a client and so I would have been completely content to take my models, adapt it as need be, and never even have to deal with anybody else's code because I can do it all myself. I don't need to deal with emails back and forth, responses from different people, I can get it done. And

when I get it done I know it's going to work. I know the calculations work correctly. I'm not going to have to debug anything and I can just replicate an old model and produce it to the client.

When programmers did seek the help of others, this behavior was an unfolding social and political process. Someone first tried to create the model themselves, then looked at other models or asked someone nearby, then might seek help from a more distant person, if recommended, and might, even after that, get the advice that they build it from scratch rather than take the existing work. Consultants made judgments about who was most likely to be helpful, both from a social and technical perspective, as the following quote illustrates:

I'd most likely talk with somebody that I can just walk to their office. I also would think about the way that each of them program. Susan has her own way of doing some things and so I might be more likely to go ask Tony. Because Susan programs just using kind of different – I don't want to say – just different conventions.

As in help-giving, reputation also mattered in help-seeking. Help-seeking behavior was influenced by the status of those who were being sought— when appealing to a high status organization member, for example, consultants sought referrals from others first. The evidence suggests that help-seeking behavior tended to stimulate helping behavior in several ways. It demonstrated demand for someone's knowledge and experiences; it gave credit where credit was due for prior experiences; it established the context for new problems, making helping behavior more efficient and directed; it was a tangible and personal plea for assistance; and it demonstrated the amount of respect help-seekers held for helpers.

### **Sanctioning Feedback**

Sanctioning feedback describes the set of activities consultants in MarketModels engaged in that, either positively or negatively, valued the contributions that others made in sharing their knowledge. These activities played a signifying role in the knowledge work of MarketModels by casting judgment on the sharing behaviors of others in the organization. Many of these sanctions were directly reflected through help-seeking and help-giving behaviors, which incorporated the social understanding of consultants

regarding helping. When Mary helped Tony by giving him her past model, for example, and Tony incorporated part of the model, Mary would be likely to offer help again. In contrast, had Tony ignored the help, Mary would not be as quick to offer the next time. Additionally, if pointing out how old code could be re-used was likely to create more work for a consultant, that individual might be loath to mention it. In one instance, when Dave's offer of help expanded into more time than he thought appropriate, rather than stop, he simply decided to avoid future entanglements with that person. In general, consultants measured the value of their help in the extent to which others put it to use.

Therefore, activities such as the re-use of code either supported or suppressed future help. In addition, informal recognition of prior help served as a potent sanctioning behavior. For instance, the norm at MarketModels was for consultants to give credit to those who contributed while describing the project to others in the organization. This feedback loop was most notable in its absence; when one programmer felt he had contributed to a particular project and did not receive acknowledgement for it, he determined not to help that person in the future, subtly and silently eliminating connections between future models.

In addition to help-giving behaviors, help-seeking was often noticed and drew comment. Managers as well as consultants had a sense that there was a right time to seek help, and noted that people could be too quick to seek help, not taking the time to figure out whether the problem could be solved more easily on their own. Conversely, there were people who did *not* seek help when it was readily available and would be valuable for developing their models. MarketModels had a formal sanction for helping and help-seeking – feedback about helping was incorporated into the review forms that all consultants completed about working with one another. Informally, consultants could also develop a reputation for being someone who was too quick or slow to seek help. These informal and formal sanctions signaled to consultants that help should be sought, but in a socially and politically acceptable manner.

## **Discussion and Conclusion**

Work within MarketModels consisted of creating unique software solutions for a variety of clients. This work drew from the collective experiences of the organization's members which, in turn, depended on the efforts of individual members to share their own past experiences with others. The evidence suggests that these efforts were strongly influenced by an interdependent web of reputations, obligations, and individual identities; political influences that do not appear in extant models of knowledge work as production or exchange. We focus on the subset of the organization's knowledge that dwells within the code of software programs and the minds of individual programmers and which provides each new program with the ability to accomplish specific tasks such as calculating an adoption curve or reading variables from multiple databases. As individual programmers faced novel challenges, they drew from the ideas and experiences of the organization's past, using them as raw materials to create new knowledge. This new knowledge reflects the needs of the new situation, but also the process by which those old ideas and experiences came together from across the group. And this new knowledge, in turn, became more raw material for solving subsequent problems. The evolution of organizational knowledge can be seen, in this software group, in the study of sequential generations of programs and programmers. And the role of social interactions can be seen in how these past generations of people and programs come together or move apart.

We were interested in the circumstances within the organization that facilitate or impede the accessibility of past code, ideas, and experiences to the current authors of the forecasting models. This question is relevant to a broad range of organizations that seek to provide goods and services that are both customized for particular customers and circumstances and also build from the latest expertise and experiences of the organization. Such organizations include, for example, law firms that argue cases which build upon similar past cases but represent highly customized solutions in each generation; engineering design firms which build products and processes that are both unique to each application yet build upon an accumulated expertise within the firm; architectural and construction firms which build unique structures that depend upon the reuse of past designs; and perhaps even educators, whose curriculum designs are

constructed from their past experiences and may not integrate with or exploit the latest generations of curricula developed by their colleagues.

Perhaps the most practical implications of this study are the insights a political perspective might contribute to ensuring that each of the organization's products or services reflects the latest of the organization's knowledge. While the benefits of the reuse and incorporation of past knowledge are usually clear, the costs rarely are. Examining the political and social dynamics of knowledge sharing allows us to see why the success of such incorporation can be problematic.

The implications of the political process of evolution in MarketModels' knowledge was made clear to us by a series of events that unfolded during our study and threatened to disrupt this evolutionary process. A new consultant was hired who had substantially different (and more current) software training than the others. He instigated a series of changes that called for a fundamentally new way of structuring and programming models. Not only would the models be different, but these changes would also make obsolete much of the past knowledge that was held by the older members of the group. One consultant, who felt caught in the middle of this emerging dilemma, explained:

So we started out with this group of people who all had common knowledge of programming skills, common capabilities, and agreed on some conventions, built this practice area. Now we have more people coming in who might be better programmers than those people who started the thing, and they're doing the thing their own way, but it's kind of created somewhat of an issue in that we can't all share models together. We can't all share codes, because we don't understand what's going on in each other's codes...

Changing the structure of the software, at MarketModels, threatened to change the value of the knowledge people held of all the past models. Another consultant described:

Dave and Tony and Bill kind of came to an agreement about how they're going to do things, created a common model structure that has been followed in every model. And then we hired new people coming out of school who probably had more advanced programming classes, have a lot more knowledge of how to do things. They look at this stuff that was developed three or four years ago and say, "This isn't the best way to do things" and kind of add their ideas to it. And in doing so – when I look at their code I have no idea what's going on because I learned how to program according to the conventions that were made four years ago ... I learned by having Bill loan me one of his models. I took a day and went through it and asked him questions and that was sort of my training on the stuff.

This example illustrates how the practice of knowledge work was enmeshed in the social system of the organization: the new consultant's knowledge and skills, to be useful, had to mesh with the existing knowledge and skills of the others in the group. And the reluctance of older consultants to embrace this new knowledge reflected self-interested preservation of their role in keeping the organization's knowledge. But it also reflected their understanding that a shift to the new modeling techniques would make it extremely difficult to draw from the past models and experiences of programmers. Because each new model captured a small portion of the organization's overall knowledge and experience, any shift to a new standard threatened to only carry forward the knowledge captured in the few models making this shift. Lost would be the wealth of knowledge captured in the hundreds of past models, and in the experiences of those programmers who created them.

Additionally, those older consultants were quite aware of the turnover in junior programmers. So while they might appreciate the value and superiority of the recent programming languages and standards these new programmers brought with them, they were also acutely aware that these individuals were the most likely to leave the organization. The worst possible scenario would be to update the programming standards to reflect the best practices of the software field but obsolete much of MarketModels' programming expertise. As such, we might say that MarketModels faced a competency trap, in which the short-term costs of moving to a better process exceeded the benefits of remaining competent in an existing one.

The politics involved in deciding whose knowledge to seek out, who to share your own knowledge with, how to share it, and how the organization rewarded and punished such behaviors was not simply a social process that should be layered on top of production and exchange practices. These motivations and behaviors shaped the very knowledge that was being produced and transacted. So to describe what it is that knowledge workers are doing when we say they are doing knowledge work, we must by necessity embed this work within the organization as an ongoing social system. And looking forward, we should also explore how such a social process shapes the evolution of its output, knowledge itself.

## References

- Bechky, B. A. (2003) Creating shared meaning across occupational communities: The transformation of knowledge on a production floor. Forthcoming, *Organization Science*.
- Berger, P. L., and T. Luckman (1967), The Social Construction of Reality New York: Doubleday.
- Burns, T. (1961) "Micropolitics: Mechanisms of Institutional Change" *Administrative Science Quarterly*, 6, pp 257-281.
- Burt, R. S. (1987), Social Contagion and Innovation: Cohesion versus Structural Equivalence, *American Journal of Sociology*, 92, pp 1287-1335.
- \_\_\_\_\_ (1992a), The Social Structure of Competition. In *Networks and Organizations: Structure, Form, and Action*, ed. N. Nohria, R. G. Eccles. Boston: Harvard Business School Press
- \_\_\_\_\_ (1992b), Structural Holes: The Social Social Structure of Competition. Cambridge, MA: Harvard University Press.
- Coase (1937), The nature of the firm, *Economica*, 4, pp 386-405.
- Dacin, M. T. (1997), Isomorphism in context: the power and prescription of institutional norms, *Academy of Management Journal*, 40, 1, pp 46-82.
- Dacin, M. T., M. J. Ventresca, and B. D. Beal (1999), The embeddedness of organizations: dialogue & directions, *Journal of Management*, 25, 3, pp
- Dougherty, D., and C. Hardy (1996), Sustained product innovation in large, mature organizations: Overcoming innovation-to-organization problems, *Academy of Management Journal*, 39, 5, pp
- Eisenhardt, K. M., and F. M. Santos, eds. (2001), *Knowledge-Based View: A New Theory of Strategy?* Handbook of Strategy and Management. ed. A. M. Pettigrew, H. Thomas, R. Whittington. Thousand Oaks: Sage Publications.
- Granovetter, M. (1973), The Strength of Weak Ties, *American Journal of Sociology*, 6, pp 1360-1380.
- Grant, R. M. (1996), Toward a Knowledge-Based Theory of the Firm, *Strategic Management Journal*, 17, pp 109-122.
- Hansen, M. T. (1999), The search-transfer problem: the role of weak ties in sharing knowledge across organization subunits, *Administrative Science Quarterly*, 44, pp 82-111.
- Katz, D., R. L. Kahn, and J. S. Adams (1980), The study of organizations : findings from field and laboratory San Francisco, Calif.: Jossey-Bass.
- Kogut, B., and U. Zander (1996), What Firms Do? Coordination, Identity, and Learning, *Organization Science*, 7, 5, pp 502-518.
- Meindl, J. R., C. Stubbart, and J. F. A. Porac (1994), Cognition within and between organizations: Five Key Questions, *Organization Science*, 5, 3, pp 289-293.
- Morgan, G. (1997), Images of organization. 2nd ed. Thousand Oaks, Calif.: Sage Publications.
- Motowidlo, S. J., W. C. Borman, and M. J. Schmit (1997), A theory of individual differences in task and contextual performance, *Human Performance*, 10, 2, pp 71-83.

- Nonaka, I. (1994), A Dynamic Theory of Organizational Knowledge Creation, *Organization Science*, 5, 1, pp 14-37.
- Nonaka, I., and H. Takeuchi (1995), The knowledge-creating company : how Japanese companies create the dynamics of innovation New York: Oxford University Press.
- Orlikowski, W. J. (1992), The duality of technology: Rethinking the concept of technology in organizations, *Organization Science*, 3, pp 398-427.
- Orlikowski, W. J., and D. C. Gash (1994), Technological Frames: Making Sense of Information Technology in Organizations, *ACM Transactions on Information Systems*, 12, 2, pp 174-207.
- Powell, W. W. (1990), Neither Market nor Hierarchy: Network Forms of Organization. In *Research in Organizational Behavior*, ed. B. M. Staw, L. L. Cummings, vol. 12, pp. 295-336. Greenwich, CT: JAI Press
- Powell, W. W., and P. Brantley (1992), Competitive Cooperation in Biotechnology: Learning Through Networks. In *Networks and Organizations: Structure, Form, and Action*, ed. N. Nohria, R. G. Eccles. Boston: Harvard Business School Press
- Spradley, J. P. (1979), The ethnographic interview New York: Holt, Rinehart and Winston.
- Ventresca, M., M. Washington, D. Dialdin, and R. Lacey (2000), Form Entrepreneurship in Online Database Services, 1972-1992, *Working Paper, Kellogg Graduate School of Management*, pp
- Weick, K. E. (1995), Sensemaking in Organizations Thousand Oaks, CA: Sage Publications.
- Williamson, O. E. (1975), Markets and Hierarchies: Analysis and Anti-Trust Implications New York: Free Press.
- \_\_\_\_\_ (1981), The Modern Corporation: Origins, Evolution, Attributes, *Journal of Economic Literature*, XIX, December, pp 1537-1568.
- Wolfe, R. A. (1994), Organizational Innovation: Review, Critique, and Suggested Research Directions, *Journal of Management Studies*, 31, 3, pp