2 ACCOMMODATING EMERGENT WORK PRACTICES: ETHNOGRAPHIC CHOICE OF METHOD FRAGMENTS

Richard Baskerville
Computer Information Systems Department
Georgia State University
35 Broad Street
Atlanta, Georgia 30302
U.S.A.

Jan Stage
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7
DK-9220 Aalborg East
Denmark

Abstract

Development methodology is a key issue for research in information system development. It is often assumed that methodologies and practice are closely related, but there are few attempts to justify this assumption. Much of the literature on development methodologies is normative and conceptual; empirical work into the efficacy of these methods is lacking. In fact, empirical evidence indicates that we should instead try to understand the system development process as being emergent, so even if methodologies appear to the observer as structure, they are only transient regularities in work practices that are constantly shifting form. Even if it is claimed that a project employs a certain methodology, it is usually not used as prescribed.

In order to realign research and practice, we must improve our understanding of, and means to support, the ways in which development is conducted in practice. This paper presents a framework for understanding how work practices are accom-

modated to the work setting. The framework defines work practice and method fragment, and it describes a sociological process for accommodation: selecting method fragments in emergent work practices.

1. INTRODUCTION

Development methodologies have for several decades been a key concern for research in information systems development (ISD). One avenue of work in this area has been to express research results and experiences in the form of an ISD methodology. Another avenue has been to see methodologies as generalized descriptions of the way in which work is conducted in development organizations. Both of these avenues share the assumption that methodologies and practice are closely related, so methodologies can serve as models of, and for, ISD in practice.

There are very few attempts to justify this assumption. Much of the literature on ISD methodologies is normative and conceptual; empirical work into the efficacy of these methods is lacking (Wynekoop and Russo 1997). Moreover, this assumption is challenged by experiences and empirical studies of ISD practice.

1.1 Limitations of ISD Methodologies

A common feature of ISD methodologies is to describe development as a completely rational process. Yet it has been argued that although developers produce documentation that makes it appear that they followed a rational design process, in reality that design process was a tortured discovery operation, and the faked documentation summarizes the simple truths that emerged (Parnas and Clements 1986). The system development process is emergent, so even if methodologies appear to the observer as structure, they are only transient regularities in work practices that are constantly shifting form (Truex et al. 2000).

The practical relevance of ISD methodologies is also limited by the application domains of these methodologies. Many methodologies employ a formalized language for expressing analysis results and design proposals. However, formal methods only provide clean solutions to problems that are amenable to formal methods, leaving a growing residue of "messy," less-amenable problems that cannot be resolved with method (Gause and Weinberg 1989). In a similar line of reasoning, it has been argued that the challenge of developing high-quality software cannot be handled through the use of methods that essentially require that software developers act the same way as machines. Development work instead should be understood as a form of theory building where intuition

and generation of ideas form the basis for expressing and refining how information technology may be useful in a given work setting (Naur 1985).

ISD methodologies, however, devote little attention to their users. It is unusual to see a methodology that presents explicit requirements to the developers that will be using it. Nevertheless, experienced developers have argued that good designers or good design teams are more critical to systems design than good design methodologies (Boehm and Papaccio 1988; Brooks 1987). This is supported by an experiment that discovered wide variance in design solutions by student teams despite duplicated methodologies (Turner 1987).

1.2 ISD Practice

ISD practitioners and organizations often claim that they employ a certain methodology in their development efforts. For example, many organizations stated 10 to 15 years ago that they were using structured methods. Today, similar statements are made about object-oriented methods. Even the leaders in structured approaches observed early on this mismatch between claims and actual practice. In 1986, for example, Yourdon observed that a vast number of development organizations claim the use of structured methods, but his own estimate was that only about 10% of the data processing organizations in North America practiced the basic ideas of the methodology in a disciplined fashion (Yourdon 1986).

More recent research on adoption of methodology supports this estimate. At the very least, one in four ISD practitioners claims to be following internally developed methods (Fitzgerald 1996). Thus development organizations seem to practice methodologies in a different fashion than the profession prescribes. This difference is illustrated by empirical findings identifying a divergence between method and practice at MCC in Austin, Texas. It was observed that even though the methodological guidelines suggested a different approach, analysts and designers tended to work at a higher level of abstraction, then see a detailed area, dive into it, investigate it, and return to a higher level of abstraction (Coad and Yourdon 1991). In two participative case studies, it was discovered that long-term methods are ineffective because the organization changes underneath them, and that methods artificially structure and interfere with the development process (Baskerville et al. 1992).

In a triangulated survey and case study, it was discovered that surveys could erroneously show heavy use of structured analysis and design in the work place. A follow-up detailed case study showed that only parts of the methodology were used, combined with parts of other methodologies, and that structured procedures were never used (Bansler and Bødker 1993). This finding was confirmed by an in-depth case study of eight organizations by which the study questioned the existence of formalized commercial methodologies "in the wild," since even

the organizations that claimed to use them did not in fact use them rigorously or in totality (Fitzgerald 1997). Indeed, an earlier, related survey indicated that only 14% of the developers who were interviewed claimed to be following published methodologies at all (Fitzgerald 1996).

The evidence and arguments above indicate that work practices in ISD are generally emergent, either being unstructured or embracing an adaptive structure that is emergent. Practitioners do not adopt methodologies; they adapt fragments of methods to a development situation. This is the case even if it is claimed that they employ a certain methodology. Evidence of this kind of emergence was confirmed in Fitzgerald's in-depth study:

> practitioners will not adopt formalized methodologies in their prescribed form and, indeed, that they may be modifying and omitting aspects of methodologies in a very pragmatic and knowledgeable fashion. In fact, there was considerable evidence that methodologies are tailored quite precisely to the exigencies of the development environment faced in organizations currently (Fitzgerald 1997, p. 211).

1.3 Understanding and Supporting Practice

The discussion above illustrates that emergent ISD settings drive practitioners to invent and adapt fragments of methods in unique ways rather than to wholly adopt a published methodology. This process appears to be a complex social-organizational phenomenon. Despite the known divergence between abstract methodology and concrete practice, we still have very little descriptive or prescriptive advice about how practitioners go about their day-to-day application of methodology.

Few published methodologies offer prescriptions for lifting their fragments for use elsewhere, or suggest ways to interject fragments from other methodologies. Most of the work done in this area is external to, and independent of, published methodologies. This work is called method engineering and the concept of a method fragment or method chunk is one of its central premises (Rolland and Prakash 1996). The method engineering approach is limited in its ability to consider the social and organizational aspects of ISD method adaptation. It has a tool orientation that brings focus to structural aspects of the methodology: notations, specifications, process definitions, etc. However, the approach lacks deep consideration of organizational culture, politics, social communication channels, etc. Fragment selection is assumed to be a technical rational debating process (cf. Harmsen et al. 1994; Oinas-Kukkonen 1996).

The dichotomy parallels the gulf between the positivist views of natural science and the interpretivist views of social science. Indeed, our preoccupation with positivist science as a criteria for reference disciplines prevents us from considering merits of non-methodical, "creative" professions as reference disciplines for ISD methods (Lee 1991). Instead of positivist-style method "engineering," interpretivist social science would seek an adaptive methodology that would open accommodation of soft, ill-structured issues like culture and politics.

The purpose of this paper is to demonstrate the potential of incorporating such advice into our body of socio-organizational ISD literature by describing a sociological process for accommodating methodology: adapting method fragments in emergent work practices. The description of the process will combine experiences and literature to define a socio-organizational process for selecting method fragments in order to design emergent work practices.

In the following section, we will define what we mean by work practice, which is the fundamental conception of real development processes. Then we proceed in section 3 to describe how we can see method fragments as opposed to complete methodologies as the essential elements we use to accommodate our work practices. The accommodation process is developed in section 4. Finally, section 5 concludes the discussion.

2. WORK PRACTICE

Work practice is the concept we will use to refer to the way in which a concrete development process is actually conducted in practice. The concept is often used in the ISD literature, but it is rarely defined explicitly. The principal concepts underlying ISD may not have changed very much over the years. These include the process model, prototyping, user participation, structured approach, information hiding, functional decomposition, cohesion and coupling, entity relationship diagrams, data-flow diagrams, data dictionaries and object-orientation (Fitzgerald 2000).

Our definition of the concept of work practice is based on a research project from the early 1980s. A key aim of this project was to describe how the situations that occur in a development project are influenced by interplay between the conditions of the project and its work practice. This is illustrated in Figure 1. The *conditions* and *work practice* influence the *situations* that occur (arrows marked 1 and 2), the situations may change conditions and work practices (arrows marked 3), and work practices may filter the influence of conditions on the situations that occur (arrow marked 4). The course of a project may then be described as a sequence of occurrences as indicated in the figure (Andersen et al. 1990; Lanzara and Mathiassen 1984).

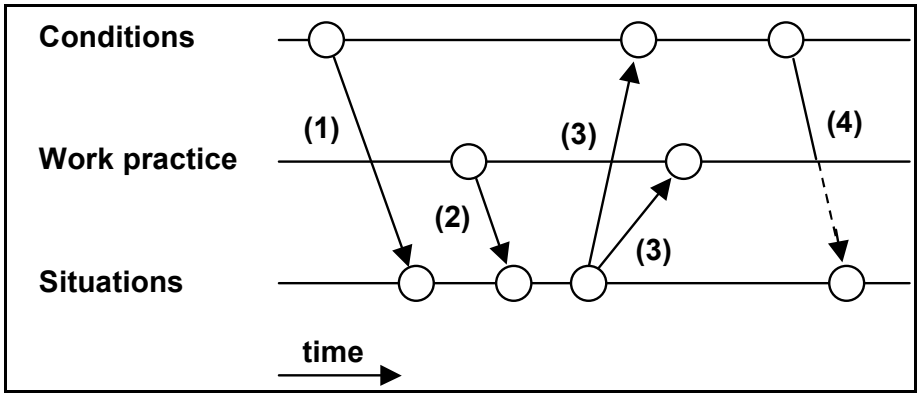

Figure 1. The Interplay Between Conditions and Work Practice

Based on a study of the empirical results from this project, descriptions of eight system development projects (MARS 1984a, 1984b, 1984c, 1984d), and an empirical study of system development from the same period (Borum and Enderud 1981), we define a *work practice* as being constituted by the following seven elements:

- *Organization* denotes the structural, organizational boundaries for the project team and its work. These boundaries define division of work and specialization, goal definition and solution space, distribution of power and responsibility, and distribution of efforts and rewards (Borum and Enderud 1981, pp. 62-63).
- *Management* is the continuous monitoring, control, and coordination of the organization of the project team and the way it carries out its work.
- *Strategy* denotes the overall approach that the development team employs to the total task of developing and introducing an information system. This element includes division into subtasks, conditions for carrying out individual activities, interplay between activities, and the course of action over time. For example, the strategy of a project may be based on a specific combination of prototyping and specifying.
- *Collaboration* includes both the mutual interaction between the members of the project team and the interaction between the project team and the other stakeholders that are involved in or influenced by the development project.
- *Techniques* denote the detailed way in which certain development activities are carried out (Mathiassen 1981, p. 100). For example, a project may employ techniques from a specific development methodology.
- *Tools* are the artifacts that are employed in certain development activities (Mathiassen 1981, p. 100). Some tools relate closely to specific techniques.

For example, this applies when the results of applying a technique are documented by means of a certain notation which then serves as a documentation tool. Other examples are fourth generation tools and development environments.

- *Evaluation* denotes the procedures and measures that are used to assess the quality of a product or the course of the project. For example, this may be a procedure and a set of criteria for measuring how an information system is adapted to a work situation.

3. METHODOLOGY AND METHOD FRAGMENT

In order to understand and support selection of method fragments, we need a precise definition of methodologies and their elements. An information system development *methodology* can be defined as an organized collection of concepts, beliefs, values, and normative principles that are supported by material resources (Lyytinen 1987). A description of a methodology can be divided into the following four elements:

- The *perspective* is the methodology's overall view on and understanding of activities in the user and development organizations. In many cases, the perspective is not described explicitly but it will, nevertheless, influence applications of the methodology.
- The *application domain* is the set of information system development projects toward which the methodology is oriented. It can be described by the part of a user organization that is in focus and the intended change of this organization. An example of this might be object-oriented development of planning systems.
- *Prerequisites* are the requirements to conditions and work practices that arise when the methodology is applied (Mathiassen 1981, 1997).
- *Activities* are the overall elements that organize the practical guidelines of a methodology. A methodology divides development work into a number of activities. Each activity is then described in terms of a number of elements including those emphasized above (Mathiassen et al. 1996). Activities usually include the following six elements:
  - — *Principles* are the overall ideas that control how an activity is conducted.
  - — *Concepts* are the terminology that is defined and employed in an activity.
  - — *Techniques* are the detailed guidelines for conducting and activity.
  - — *Notations* are the means of expression that are used to capture and maintain the results of an activity.
  - — *Products* are the outcome of an activity.

— *Criteria* are the measures that are used to evaluate the products of an activity.

A *method fragment* is any element of a methodology's guidelines for its activities that is separated from the methodology. It is a concept, notation, tool, technique, etc. that is lifted from the framework of an overall methodology and interjected in a specific work practice. Under this concept, each systems development project is a moving pastiche of miscellaneous parts: bits of external methodologies, internal methods, innovative, unique techniques invented on-the-fly, *etc.* (Baskerville et al. 1992; Truex et al. 1999).

4. FRAMEWORK FOR ACCOMMODATING WORK PRACTICES

In order to describe and understand ISD practice, we will provide a framework based on the idea that practitioners "accommodate" by selecting, inventing, and combining method fragments to fit their needs as developers in a concrete social and technical setting. This idea builds on the observation that practitioners cannot follow prescribed methodologies, nor can they entirely engineer their methods or adapt and modify their methods in entirely rational ways (Truex et al. 2000). The accommodation is an ill-structured process by which their approach to information systems development is made suitable and congruous with the people involved, e.g., developers and users, and the available technology. In so doing, the developers attempt to tailor their approach and bring it into agreement or concord with the gestalt of their systems development work practice.

The *framework* for accommodating work practices consists of three components, as illustrated in Figure 2. First there is an extensible set of generalized method fragments originating from published *methods* as well as *method fragments* that have been innovated through previous practice. These elements are described conceptually in section 3.

The second component is the set of determinants of fragment selection. These ill-structured determinants are the aspects of a *work practice* that are described in section 2, and they help in defining for the developer the fragments that might be relevant to consider for adaptation.

The third component entails a sociologic process for the on-going *accommodation:* selection, invention, and combination of method fragments that are appropriate in a given work setting. Accommodation of methodologies to practical settings is a rich and messy process, conducted in a complex social setting.

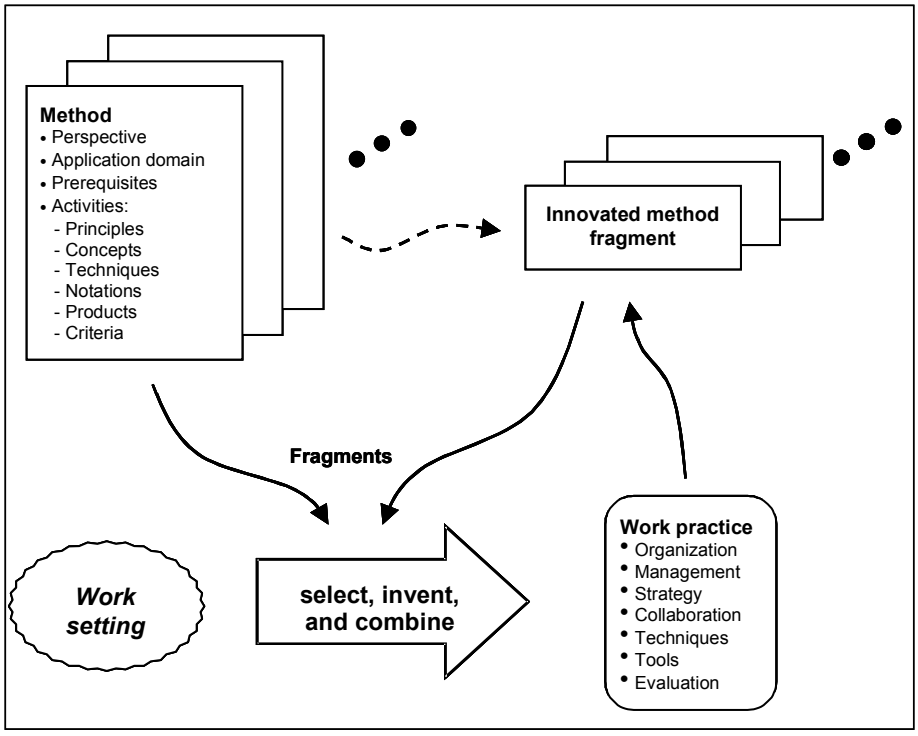

Figure 2. Components of a Social Process for Method Fragment Adaptation

Participative ISD methods, springing from the roots of socio-technical design, are partly a recognition of the cultural clash between system designers and users. Our aim is to extend the boundaries of this collision even further, and consider the interplay between designers, users, programmers, methodologists, and other people in their work setting. Given this extended boundary, the phenomenon under study is the everyday *modus operandi* system developers enact as they accommodate development, adapt methods, and select or discard method fragments. Such everyday social philosophy is called *ethnomethodology* (Gubrium 1990). In line with Lee's suggestions, it is consistent with this view of the process to consider anthropology as a potential reference discipline for processes of adapting methods (Lee 1991). Currently, ethnography is a key research method within anthropology.

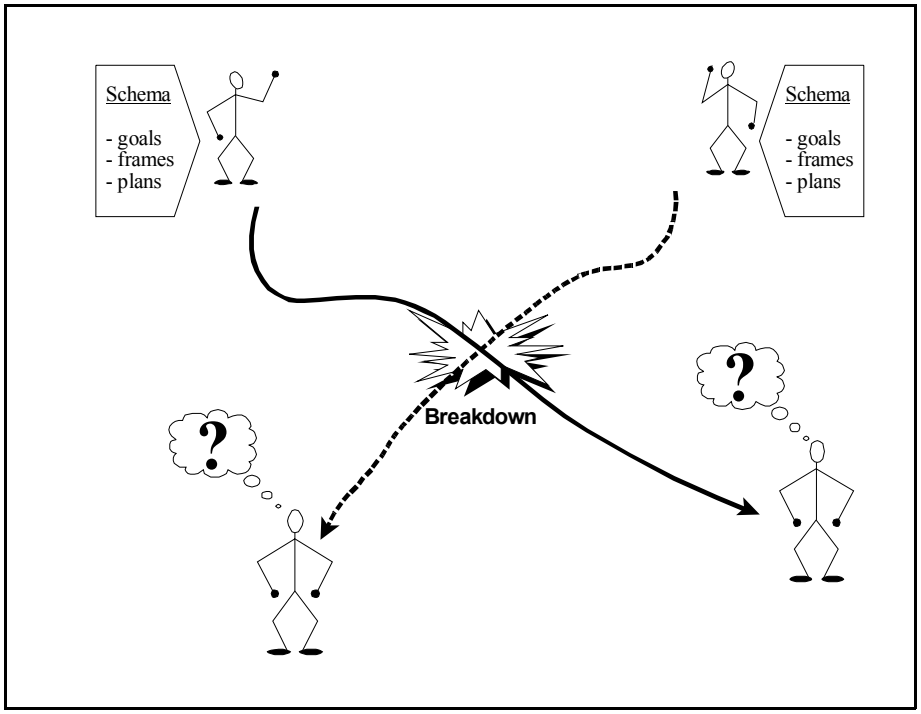

Figure 3. The Ethnographic Strip

4.1 Practical Ethnography

There are three types of ethnographic methods: structural, articulative, and practical. Structural ethnography is the study of the way people within a culture define and organize their ideas and perceptions of reality. Articulative ethnography is the study of how people enact these ideas and perceptions. Practical ethnography balances structural and articulative approaches, seeking to discover structure through behavior (Gubrium 1990). Michael Agar (1986) provides an exceptionally clear definition of a practical ethnography.

Agar's focus is on the ethnographic *encounter*. Such an encounter could include a meeting or exchange between a system developer and a user, or a developer and a project manager, or a developer and a customer, etc. Agar defines four major units of analysis within an encounter: schema, strip, breakdown, and resolution (see Figure 3).

The *schema* consists of goals, frames, and plans of the people involved in the encounter (including the developer). Goals represent the set of objectives

that each of the people is aiming to achieve. Frames represent the way they organize their beliefs, knowledge, affections, etc. Plans represent the way each person hopes to achieve their goals.

A *strip* is an observed social act. A strip is rather like a scenario or a story; it is the history of the encounter in terms of the behavior of the people during the encounter.

A *breakdown* occurs when people are unable to make sense of the encounter. Breakdowns occur with frightening regularity in many development contexts. Breakdowns do not mean that people are crazy. They mean that the people involved simply do not understand the schema within which their companions are formulating their behavior. One ethnographic goal is to recognize breakdowns for what they are, and to determine what they mean. Agar defines two types of breakdowns. Occasioned breakdowns occur naturally and unexpectedly during encounters. Mandated breakdowns are purposely created by developers in order to further study behavior discovered in an earlier breakdown.

Resolution occurs when people discover the reason for a breakdown. The difference between their own schemas and those of their companions is clarified and the reason for breakdown is discovered. There are also two types of resolution. An intervening resolution is a partial resolution that explains some of the weird behavior in the strip. A coherent resolution is achieved when an explanation, or a system of explanations, is discovered that explains all weird behavior in a strip. Coherence, or a coherent resolution, will very often clear up breakdowns in several strips.

4.2 Using Practical Ethnography to Understand Method Accommodation

Agar's general ethnography can be used to describe the fragment selection process and the way in which method accommodation unfolds. Method accommodation centers on strips. Strips describe encounters between project stakeholders, and a breakdown occurs when the actors in the strip are unable to make sense out of the encounter. Perhaps there was bizarre user or developer behavior or there were absurd specifications. Practical ethnography suggests that when breakdowns occur, these must be recognized and documented.

Breakdowns provide an indication that the methodology is somehow inconsistent with the schemas of the stakeholders. Different method fragments, or different adaptations of the fragments, could be needed that can capture and consider relevant goals, frames, and plans of the people involved in the project. In this way, the breakdown is an indicator that the methodology does not accommodate the development setting.

How do developers proceed following such an occasioned breakdown? One approach would be to seek discovery of how the goals, frames, and plans of the various stakeholders are inconsistent with the development approach. A means for this could be a mandated breakdown process, where breakdowns are designed and centered on the introduction of new or newly adapted method fragments. This discovery and breakdown process can be iterated until resolution is achieved. Resolution occurs when the strange behavior notable in the breakdowns becomes rational to all parties.

5. EXAMPLE

Practical ethnography is useful for describing and understanding the ill-structured mechanisms for accommodating a system development setting. We can demonstrate this with a simple example: the process of fragment selection. To illustrate this point, we will describe below a case setting in which different ethnographic encounters arise.

5.1 Case Description: IT Development in a Bank

This example regards a development process in the IT department of a bank. The bank has a headquarters and 40 branches dispersed across its region. Within this region, the bank is one of the major competitors. The IT department maintains and runs a large centralized transaction system that was introduced more than 20 years ago with a batch-oriented mode of operation. Since then, it has been transformed into a modern database system with on-line access from all branches. Five years ago, the terminals on all cashier desks were exchanged with PCs, and all clerks got their own PC at their desk. A decentralized local server was installed in each branch. Originally, the PCs were thought of as being just a new kind of terminal, but lately, it has been considered whether their processing capacity might be used for other purposes.

Each of the strips below will describe selected encounters between the system developers and their community. In each case, we will identify a breakdown and the accommodation of the social setting by adapting the systems development methodology.

5.2 Strip One—The Concepts: Account or Customer?

Top management had initiated a project to clarify the potentials by developing a prototype of a tool for supporting bank clerks in the granting of loans to customers. The project group consisted of a senior bank clerk from the

main department that represented the future users and five developers from the IT department. In order to avoid potential conflicts, the project group was asked by management to minimize their relation to all bank clerks, and they should never visit any of the branches.

When the original system was developed, the manual account file system was analyzed. The concept "account" was defined, and attributes of, and legal transactions on, an account were specified in detail. The developers believed this description was still valid because the previous changes of the information system had mainly been of a technical nature.

The project group based their analysis on the concept of an account. However, it quickly turned out to be of limited relevance. As the encounter between the developers and the senior clerk unfolded, a breakdown in the shared understanding materialized. Initially, the developers thought the clerk's reaction was bizarre. While the developers had framed their thinking as accounts, the senior clerk had a different frame. They explored the work processes related to granting of a loan, and accounts began to take on only subordinate importance.

As the developers sought a new frame, they learned more about the clerk's work processes, and it became clear that a description could not be based on the account concept. A loan is not granted because of the state of a certain account. It is more important to get an overall impression of the complete financial situation of the customer. The developers reframed their thinking and substituted related customer and account concepts in place of the simplified account concept as the basic concepts in analysis. Resolution of the breakdown was achieved by extending the concepts (a method fragment) to describe the users' work in a richer manner (adaptation).

5.3 Strip Two—The Perspective: System or Use?

After half a year, the project group presented a proposal for design of the system. The proposal was accepted by management and afterward was submitted to all employees for comments. While the project group waited for reactions, it started to define the more detailed design and to explore ways of implementing the design.

Four months later, the project group had not received any reactions. Management decided to call a meeting where all employees were invited to meet the project group. This meeting turned out to be a confusing and unpleasant encounter between the project group and the user community. The user community asserted that the description of the design proposal was far too technical. It only described the computer system and software, not the use of it. It was impossible to understand what the new system would be like in use. They

demanded a better description. In this breakdown, the mental frameworks of the users and the developers were too distinct for meaningful communication between the two cultures.

Management was itself very concerned about getting acceptance for this system, and they requested that a prototype be developed. The developers accepted the criticism from the employees and formed a new project group that included three bank clerks from the main department, one manager from the main department, the original five developers from the IT department, and an external consultant. Resolution required that the developers (and management) reframe their methodology from their rational delivery approach (a method fragment) to a participatory approach (an alternative method fragment) in order to make system use-in-context (as opposed to system design) the basic perspective for their design descriptions. In this way, the methodology was adapted to accommodate the social setting of system use. The tools, techniques, and frameworks of participatory development were adapted to accomplish that.

5.4 Strip Three—The Notation: Correct or Relevant?

The project group decided to start describing systematically the paperwork involved in granting a loan. The context of the encounter was a two-day retreat by an expanded project group. The retreat participants were the project group and nine bank clerks from three different branches. The first day of the retreat focussed on general issues. For example, the consultant presented other experiences with administrative systems, the manager presented how management had planned the use of IT in future years, and the project group presented the status of the project and the ideas for the new system. They emphasized that a description of the paperwork was very important. Finally, the consultant finished the day by presenting a method for structured analysis.

The second day of the retreat was used for making system descriptions. Three groups were formed (A, B, and C). Each group included a bank clerk from the project group, a bank clerk from each of the three branches and a developer. The consultant circulated between the groups. By the end of the day, the work of the three groups was presented, and the relevance of the description method was discussed.

The collaboration within the groups turned out to be the most important factor of the seminar. Importantly, the final description was more influenced by the dynamics within the groups than it was by the rules of the method. This was clearly illustrated in the final descriptions.

The bank clerks controlled group A as the developer was very modest and mainly acted as an advisor on the method. The description from this group

focussed on the relations between the work processes involved in granting of loans. In addition, it described the physical environment in the three branches that were represented by the participants. In this respect, it illustrated the actual organization of work. On the other hand, the group violated the rules and guidelines of the method. For example, they introduced a special signature for informal communication, and work was not described exactly by processing of information, but rather by the criteria that controlled decisions. Time-consuming activities were also described. Their product represented an attempt to describe human processing of information. It focussed more on the heart of the work than on the forms that were used.

The descriptions made by groups B and C were very different. In both of these groups, the developer dominated. As a result, the rules and guidelines of the method were more closely followed. In this respect, the resulting system descriptions were more correct, and thereby they provided a better foundation for the design team. But the bank clerks later acknowledged that they presented a very incoherent and fragmented description of the work processes.

The contrast in the notation and designs of group A *vis-à-vis* the others represents a clear ethnographic breakdown between the developers and the users. Driven by the need for clear technical design notation, the users in groups B and C were constrained to present a design that incorrectly described their work processes. Group A produced a more accurate, but unusual, representation of the real process.

An open discussion by all retreat participants followed. The differences began to clear as both developers and users began to resolve the breakdown. The bank clerks emphasized that all three descriptions provided a very poor picture of what granting of loans "really" is. They also agreed that of the three descriptions, the one made by group A provided the best picture.

The descriptions from groups B and C, although both more faithful to the standard notation, were quite distinct from each other. In group B, an experienced bank clerk had participated, and her understanding of the work was the main basis for their description. In group C, the members participated more equally, and the description gave a general picture of the common elements in the three branches that were represented. Comments on this description denoted it as the best product by the consultant. She considered it to be a good basis for design, and it was ultimately adopted.

In this strip, one of the groups invented their own notation for describing informal communication, and their description focussed on criteria for decision making rather than information flow. This reframing could be described as an example of intervening resolution. It represented an opportunity for the design team to accommodate the social setting by innovating the notation (a method fragment), but it became a lost opportunity. No coherent resolution was ever

achieved because the developers kept focussing on the correct notation that centralized the processing of data, whereas the users wanted to describe the decision-making process. In the end, the design proposal was based on the description that was most faithful to the method and least descriptive in terms of the core aspects of the work process.

6. CONCLUSION

This paper has presented a framework for accommodating emergent work practices from existing methodologies. The framework consists of three components: (1) a set of generalized concepts for describing method fragments, (2) a set of elements that are relevant for accommodating work practices through method selection, and (3) a sociologic process by which humans determine the method fragments that are appropriate in a given situation.

The components of the framework have been illustrated through an empirical example. In this illustration, the work practice that emerged in a development project is described and explained in terms of the components of the framework and the accommodation process.

The work in this paper is not without its limitations. The basis for our conclusions is a logical argument based on published research and illustrated *post hoc* in an empirical setting. We believe the framework and accommodation theory ought to be subjected to more rigorous field testing to determine if the general form will hold in settings other than the single case presented here. For example, action research would provide field validity for the prescriptive proposals.

Accommodation theory seems to be a concept that is complementary to the rational-technical models currently used in method engineering for fragment selection. This theory also elaborates some of the well-known theoretical propositions about information systems methodology, such as amethodical development, in the context of work practices. This elaboration is a significant advance and a useful contribution to further our understanding of one of the central arenas in information systems: development methods.

7. REFERENCES

Agar, M. *Speaking of Ethnography*, Newbury Park, CA: Sage, 1986.

Andersen, N. E., Kensing, F., Lassen, M., Lundin, J., Mathiassen, L., Munk-Madsen, A., and Sørgaard, P. *Professional System Development: Experiences, Possibilities, and Action*, Englewood Cliffs, NJ: Prentice-Hall, 1990.

Bansler, J., and Bødker, K. "A Reappraisal of Structured Analysis: Design in an Organizational Context," *ACM Transactions on Information Systems* (11:2), 1993, pp. 165-193.

Baskerville, R., Travis, J., and Truex, D. "Systems Without Method," in *The Impact of Computer Supported Technologies on Information Systems Development*, K. Kendall, K. Lyytinen, and J. I. DeGross (eds.), Amsterdam: North-Holland, 1992, pp. 241-270.

Boehm, B., and Papaccio, P. N. "Understanding and Controlling Software Costs," *IEEE Transactions on Software Engineering SE 12* (4:10), 198, pp. 1462-1477.

Borum, F., and Enderud, H. *Conflicts in Organizations: Illustrated Through Studies of System Development Work (in Danish)*, Copenhagen: Munksgaard, 1981.

Brooks, F. P. "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer* (20), 1987, pp. 10-19.

Coad, P., and Yourdon, E. *Object-Oriented Analysis* (2nd Edition), Englewood Cliffs, NJ: Yourdon, 1991.

Fitzgerald, B. "An Investigation of the Use of Systems Development Methodologies in Practice," in *Proceedings of the Fourth European Conference on Information Systems*, J. D. Coelho, T. Jelassi, W. König, H. Krcmar, R. O'Callaghan, M. Saaksjarvi (eds.), Lisbon: New University of Lisbon, 1996.

Fitzgerald, B. "The Use of System Development Methodologies in Practice: A Field Study," *Information Systems Journal* (7:3), 1997, pp. 201-212.

Fitzgerald, B. "Systems Development Methodologies: The Problem of Tenses," *Information Technology and People* (13:2), 2000, pp. 13-22.

Gause, D., and Weinberg, G. *Exploring Requirements: Quality Before Design*, New York: Dorset House, 1989.

Gubrium, J. *Analyzing Field Reality*, Newbury Park, CA: Sage, 1990.

Harmsen, A., Brinkkemper, J., and Oei, J. "Situation Method Engineering for Information System Project Approaches," in *Proceedings of the CRIS'94 Conference*, Twente, The Netherlands: University of Twente, Department of Computer Science, 1994, pp. 1-34.

Lanzara, G. F., and Mathiassen, L. *Mapping Situations Within a System Development Project: An Intervention Perspective on Organizational Change* (MARS Report 6), Aarhus, Denmark: Aarhus University, Department of Computer Science, 1984.

Lee, A. S. "Architecture as a Reference Discipline for MIS," in *Information Systems Research: Contemporary Approaches and Emergent Traditions*, H.-E. Nissen, H. K. Klein, and R. A. Hirschheim (eds.), Amsterdam: North-Holland, 19901, pp. 573-592.

Lyytinen, K. "A Taxanomic Perspective of Information Systems Development: Theoretical Constructs and Recommendations," in *Critical Issues in Information Systems Research*, R. Boland and R. Hirschheim (eds.), New York: Wiley, 1987.

MARS. *System Development in Practice: Jydsk Telefon (in Danish)* (MARS Report 2), Aarhus, Denmark: Aarhus University, Department of Computer Science, 1984a.

MARS. *System Development in Practice: Øk Data (in Danish)* (MARS Report 5), Aarhus, Denmark: Aarhus University, Department of Computer Science, 1984b.

MARS. *System Development in Practice: Regnecentralen Af 1979 (in Danish)* (MARS Report 3), Aarhus, Denmark: Aarhus University, Department of Computer Science, 1984c.

MARS. *System Development in Practice: Sparekassernes Datacenter (in Danish)* (MARS Report 4), Aarhus, Denmark: Aarhus University, Department of Computer Science, 1984d.

Mathiassen, L. *System Development and System Development Method (in Danish)* (Report PB-136), Aarhus, Denmark: Aarhus University, Department of Computer Science, 1981.

Mathiassen, L. *Reflective Systems Development* (Report), Aalborg, Denmark: Aalborg University, Department of Computer Science, 1997.

Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., and Stage, J. "Method Engineering: Who's the Customer?" in *Method Engineering. Principles of Method Construction and Tool Support*, S. Brinkkemper, K. Lyytinen, and R. Welke (eds.), London: Chapman and Hall, 1996, pp. 232-245.

Naur, P. "Intuition in Software Development," in *Formal Methods and Software Development* (Volume 2), H. Ehrig (ed.), Berlin: Springer-Verlag, 1985, pp. 60-79.

Oinas-Kukkonen, H. "Method Rationale in Method Engineering and Use," in *Method Engineering: Principles of Method Construction and Tool Support*, S. Brinkkkemper, K. Lyytinen, R. Welke (eds.), London: Chapman and Hall, 1996, pp. 87-93.

Parnas, D., and Clements, P. "A Rational Design Process: How and Why to Fake It," *IEEE Transactions on Software Engineering SE 12*, 1986, pp. 251-257.

Rolland, C., and Prakash, N. "A Proposal for Context-Specific Method Engineering," in *Method Engineering: Principles of Method Construction and Support*, S. Brinkkemper, K. Lyytinen, and R. Welke (eds.), London: Chapman and Hall, 1996, pp. 191-208.

Truex, D. P., Baskerville, R., and Klein, H. K. "Growing Systems in an Emergent Organization," *Communications of the ACM* (42:8), 1999, pp. 117-123.

Truex, D., Baskerville, R., and Travis, J. "Amethodical Systems Development: The Deferred Meaning of Systems Development Methods," *Accounting, Management and Information Technology* (10), 2000, pp. 53-79.

Turner, J. "Understanding the Elements of System Design," in *Critical Issues in Information Systems Research*, R. J. Boland and R. A. Hirschheim (eds.), Chichester, England: J. Wiley, 1987, pp. 97-111.

Wynekoop, J. L., and Russo, N. L. "Studying System Development Methodologies: An Examination of Research Methods," *Information Systems Journal* (7:1), 1997, pp. 47-65.

Yourdon, E. "What Ever Happened to Structured Analysis," *Datamation*, June 1986, pp. 133-138.

About the Authors

Richard Baskerville is a professor of information systems and chairman in the Department of Computer Information Systems, College of Business Administration, Georgia State University. He is an active author whose research specializes in security of information systems, methods of information systems design and development, research methods, and the interaction of information systems and organizations. Richard's practical and consulting experience includes advanced information system designs for the U.S. Defense and Energy Departments. He is former chair of the IFIP Working Group 8.2, and a Chartered Engineer under the British Engineering Council. He holds M.Sc. and Ph.D. degrees from the London School of Economics. He can be reached by e-mail at baskerville@acm.org.

Jan Stage is an associate professor in Computer Science at Aalborg University. His research interests include methods for analysis and design in software engineering, object-oriented analysis and design, and analysis and design patterns. His articles have appeared in *MIS Quarterly, Information and Software Technology, Information Technology and People*, and *The Scandinavian Journal of Information Systems*. He is a co-author of *Object-Oriented Analysis and Design* and holds a Ph.D. in Computer Science from Oslo University. Jan can be reached by e-mail at jans@intermedia.auc.dk.