



Pergamon

Accounting, Management & Information Technology 10 (2000) 53–79

ACCOUNTING  
MANAGEMENT AND  
INFORMATION  
TECHNOLOGIES

[www.elsevier.com/locate/amit](http://www.elsevier.com/locate/amit)

# Amethodical systems development: the deferred meaning of systems development methods

Duane Truex <sup>a,\*</sup>, Richard Baskerville <sup>b</sup>, Julie Travis <sup>c</sup>

<sup>a</sup> *Computer Information Systems, Georgia State University, Atlanta, GA 30303-3083, USA*

<sup>b</sup> *College of Business Administration, Georgia State University, Atlanta, GA 30303-3083, USA*

<sup>c</sup> *School of Information Systems, Curtin University of Technology, Bentley 6102, W.A., Australia*

---

## Abstract

This paper contributes a deeper understanding of the concept of methodical information systems development. The method concept is an assumption underlying much of the research into systems analysis, design and implementation. A postmodern deconstruction technique is used to discover a deferred concept: amethodical systems development. The methodical and amethodical views are developed in terms of their assumptions and their ideal characteristics. Our understanding of these two opposing views of systems development is important as a means to refocus our aims in research, practice and education in information systems development. © 2000 Elsevier Science Ltd. All rights reserved.

*Keywords:* Amethodical systems development; Methodical systems development; Information systems development; Emergent grammar; Emergent organization; Postmodern

---

## 1. Introduction

Do information systems development methods really describe ongoing systems development practice? Do they ever explain why information systems are developed in certain ways? Do they actually function as frameworks, formulas or templates for building successful information systems? Or are such methods merely unattainable ideals and hypothetical “straw men” that provide normative guidance to utopian development situations? Exploring these questions will help us to extract the essential lessons offered by the concept of information systems methods. We address these questions by considering the alternative descriptions and explanations that arise when

---

\* Corresponding author. Tel.: +1-404-651-3899; fax: +1-404-651-3842.

*E-mail address:* [dtruex@gsu.edu](mailto:dtruex@gsu.edu) (D. Truex).

a hypothetical “method-less” view of information systems development displaces our mainstream assumptions.

Methods for the building of information systems are clearly important elements in the information systems discipline. Yet there are gnawing problems about their practicability. Methods are often unsuitable for some individuals (Naur, 1993) and settings (Baskerville, Travis & Truex, 1992). Similar methods in similar settings yield distinctly different results (Turner, 1987). Developers may claim adherence to one method while ignoring this method in actual practice (Bansler & Bødker, 1993). While development methods research has essentially reified methods, it offers little fundamental understanding of what it means to be methodical and how methods are actually applied in the field (Wynekoop & Russo, 1993).

What does it mean for information systems development to be methodical? Both Oxford and Webster’s dictionaries primarily define the term “method” as meaning “the procedure for obtaining an object”. The secondary definitions fasten on such ideas as “orderly”, “systematic”, “regularity”, and “regimen”. The term “methodical” is the adjectival form. In the mainstream discourse of our field, method is the term used for an orderly, predictable and universal approach to information systems development. Method is clearly a concept of process rather than representation.

Methods are procedures which came to full flower in the validation of medical cures and in the development of pedagogical procedures and curriculum in the Middle Ages. Methods began as explanations but became procedures. (Ong, in Coyne, 1995, p. 210).

What does it mean for building information systems to be other than methodical? We use the term *amethodical* to refer to this concept. The term is purposefully a negative construct that connotes an open set of attributes that are essentially not methodical. Thus amethodical may reject structure, but does not imply anarchy or chaos. Amethodical systems building implies management and orchestration of systems development without a predefined sequence, control, rationality, or claims to universality. An amethodical development activity is so unique and unpredictable for each information systems requirement that even the criteria of contingent development methods are irrelevant.

The objective of this paper is to reconsider the concept of method; to think anew this key tenet in the information systems development liturgy. The purpose is not to wreak ruin on the concept of method, but to clarify, extend and refine its meaning.

We extend our understanding of the methodical development of information systems by exposing the alternative observations, descriptions and explanations of the systems-making process when considered from an amethodical viewpoint. This approach is necessary because the concept of method, as you will see in the following section, occupies an extremely privileged status in formal information systems development thought even though its origin is unstated. This privileged position is so taken for granted in mainstream discussions and research about information systems development that a technique was sought to allow an unfettered view of the method concept and alternative notions.

Postmodern analysis uses such techniques for decentering the author of a text. The terms “narrative” and “text” are used in a metaphorical sense to encompass human artifacts such as organizations, a stream of human discourse, or social interaction, as well as written artifacts. Under this analysis, text is defined by the act of being read or observed, rather than in being authored. The text’s meaning is impermanent since changing readers implies possible changes in the meaning. A narrative is a body of texts, perhaps with some historical dimension, for example, the “narrative of science”. A “metanarrative” regards a narrative about the narrative; for example, the philosophy of science is a metanarrative about the scientific narrative. “Local narratives” are bounded by a community of participants, for example, the local narrative of positivist science (Lyotard, 1987; Rosenau, 1992).

The paper will deconstruct method as a general and fundamental assumption to information systems development (Beath & Orlikowski, 1994; Boje & Dennehy, 1994; Coyne, 1995; Chapter 5 in Eisenberg & Goodall, 1993; Kilduff, 1993). Deconstruction is the process of disassembling socially constructed meanings that have been coded in texts (Eisenhardt, 1989). Deconstruction of assumptions in organizational texts reveals presumed structures in the reader’s thinking, exclusionary conceptual frameworks (like method) that conceal multiple meanings. We use the deconstruction technique to confront the reader with conflicting meanings whereby any resolution assumes only a temporary dominance until overturned by the continuous process of reinterpretation. (Appendix A briefly explains several related postmodern ideas for those readers less familiar with deconstruction.)

Essential to understanding our deconstruction is Derrida’s concept of *différance*. *Différance* simultaneously connotes the differences in word meanings and the deferral of meaning through the endless play of words. Texts tend to be centered on an idea that is called “privileged” or “dominant” (e.g. method in systems development texts). The opposing idea may not be named in the text, yet exists as a shadowy, tacit concept that lingers in the reader’s mind. This concept is referred to as the subtext, hidden text or marginalized text. The exact meaning of this marginalized text is deferred to the reader (e.g. describing a method suggests that the reader understands a method). This deferral sets up a cycle of redefinition that destabilizes the privileged text as a consequence of highly contextualized marginal texts (Cooper, 1989). *Différance* also implies that meanings are affected by the simultaneous understanding of the word and its marginalized anti-concept. Deconstructing a text involves seeking *différance*; to consciously raise the concepts not openly discovered by readers.

Any text always involves sets of privileged and marginalized concepts. Concepts can be marginalized through secondary streams of signification. These streams involve the constant recursion and renegotiation of word meanings without ever reaching a transcendental ideal (discussed further in Appendix A). Thus, any linguistic artifact may ignite not only a primary thought, but also an endless cycle of secondary references in a reader’s mind. *Différance* also points to the impossibility of precisely defining any term without implying a delineation of all concepts to which the term must not refer. Since such precision requires the exact definition of an infinite universe, which is impossible, all terms must be left to a certain degree, ambiguous.

A central problem in deconstructing information systems development is the lack

of any firm understanding of the marginalized concept of amethodical information systems development. At the broadest level, this involves illuminating the marginalized text in the information systems development metanarrative. Rather than deconstructing a single method as if it were a representative text of all methods, we will limit our scope to the discovery of method–amethod *différance* in our assumptions about the tradition of method that characterizes information systems development research.

The second section will explain how the method concept fills its privileged role in the systems development text. Section 3 deconstructs the privileged text of information systems development method and defers their meanings by giving voice to the marginalized amethod text. Section 4 continues the deferral, examining the cost of raising a formerly marginalized text (amethod) to the status of a privileged text. This amounts to a deconstruction of the arguments explored in Section 3. Section 5 concludes the paper by drawing the implications of the deconstruction for the important questions posed in the opening paragraph of the paper.

## 2. The privileged position of methodical information systems development

The methodical view is privileged because the modern concept of method has been so strongly impressed on our thinking about systems development, that the two concepts, *information systems development* and *information systems development method*, are completely merged in systems development literature. We can quickly make this merger evident by exposing method as the privileged text dominating our publications about the history, practice, culture and philosophy of information systems development.

### 2.1. History

Method is privileged because the history of information systems development is typically interpreted as the history of methods for systems development. For example, modern analysts characterize the progress of systems development in terms of historic generations of information systems development method beginning with Couger's (Couger, 1973) three-generation scheme—revised to five in Couger (1982)—that links generations of these methods to generations of computing hardware. Nolan (1979) examines six stages of organizational systems development progress in terms of the qualities and effectiveness of information systems development method. Yourdon and Constantine (1979); Yourdon (1981/1989) refer to three broad periods in the evolution of ideas and techniques in the field of information systems analysis by focusing on use and nature of method. Hirschheim and Klein (1992) delineate seven generations in terms of information systems development method.

This powerful historical alignment leads to problems in distinguishing method from information systems development altogether. Some analysts are compelled to redefine the term “method” beyond its original meaning, extending it to describe the discipline of building information systems rather than the way systems are developed.

These analysts conflate concepts like beliefs, values or material resources with method, often for the purpose of contrasting less methodical aspects of information systems development such as hermeneutics or emancipation within the context of systems development.

The predisposition to believe in the power of methodologies comes from Descartes who proposed that truth is more a matter of proper method than genial insight or divine inspiration. (Hirschheim, Klein & Lyytinen, 1995, p. 21)

From the basis of such extended definitions of method, it might be argued that the privileged text as presented here is only one of many views of method. However, this privileged text is clearly the mainstream, technical–rational view (Lyytinen, 1987). As a result, these other, broader views of method become marginalized along with the amethodical.

## 2.2. *Practice*

Method is the privileged text among the academic and practical communities that link to form the context for empirical research into information systems development. As a result, much of the instrumentation that researchers assemble for the purpose of gathering data about practical information systems development are only capable of detecting and measuring methodical concepts.

For example, one field study involved a survey instrument for determining how systems were developed (Necco, Gordon & Tsai, 1987). The respondent could report their systems development approach by selecting from a choice of alternatives that included the traditional/classical approach, the systems development life cycle approach, the structured approach, automated approaches, etc. A setting in which a unique systems development approach is invented for each project could not be measured by this instrument.

Even if the instrument included such an option, the after-the-fact rationality of developers would marginalize the amethodical. Developers are known to fake the rationality of information systems development in their attempts to make sense of the process (Bansler & Bødker, 1993; Parnas & Clements, 1986). Because of this faked rationality, empirical observations based solely in the reflections of practitioners would not detect or measure amethodical information systems development. For example, Sabherwal and Robey (1993, 1995) develop and analyze a rigorous inventory of systems development events that are solidly grounded on qualitative data from a broad set of interviews with information systems developers. But this data only captures the actors' rationalized reconstruction of the systems development events. An unplanned, on-the-fly collaboration that simply materialized in an uncontrolled setting might be characterized as a "committee" by an interview respondent in an attempt to verbalize a sensible account of their project. The interviewer would record that "a committee was formed", which might later be assumed to mean that "someone formed a committee" (Sabherwal & Robey, 1995, p. 311) by the analysts.

In such ways, our method paradigm prevents our research into information systems development from discovering any practices beyond the methodical.

### 2.3. *Culture and philosophy*

Method is privileged such that even philosophical and cultural analyses of the systems development milieu must be expressed in terms of information systems development method. For example, intensive studies of systems development using Burrell and Morgan's (Burrell & Morgan, 1979) framework inevitably characterize development paradigms according to the dominant information systems development method (cf. Hirschheim & Klein, 1989; Oliga, 1988). Information systems development cultures are expressed in the regionalization of some development methods such as MERISE (Calmes, Charbonnel & Dumas, 1991; Rochfeld & Tardieu, 1983) in French cultural dominions and Soft Systems (Checkland & Scholes, 1990; Checkland, 1984) in British and Scandinavian cultural dominions.

Even our attempts to break free of the privileged confinements of method are constrained to consider only "situated" information systems development methods. For example, contingent systems development methods consider only the correct selection of methods with a comparative analysis of alternative methods (cf. Avison & Fitzgerald, 1988; Davis, 1982; Jackson & Keys, 1984; McFarlan, 1981). Conceptualizations of idiographic approaches are limited to "one-shot" methods and meta-methods (Kumar & Welke, 1992) for developing perfected one-time information systems development methods. Almost entirely eluding the systems development literature is the thought of developing a system without any method at all (cf. Baskerville et al., 1992). Even more elusive is the possibility that amethodical development might be the normal way in which the building of these systems actually occurs in reality.

## 3. Methodical versus amethodical information systems development

In this section we illustrate the distinction between the privileged methodical information systems development text and the marginalized amethodical text by selecting examples of major characteristics of systems development that are common in the privileged text. Then we draw the *différence* contrast for each example, and show how this explains reasoned and observed anomalies in the information systems development process.

Different development methods require different techniques, sequences, steps and activities. Nevertheless, there are common assumptions in these approaches which may be delineated as a privileged narrative. To the extent that the privileged narrative is dominant, any examination and greater understanding of the marginalized narrative is deferred. This section explores several key components of both the privileged and marginalized narratives of method. Table 1 summarizes the assumptions and idealized characteristics discovered in each of these texts.

Too often, developers doggedly follow the privileged method without realizing

Table 1  
Assumptions and ideals of methodical and amethodical texts

Privileged methodical text	Marginalized amethodical text
1. Information systems development is a managed, controlled process idealizing logical decomposition reductionism	2. Information systems development is random, opportunistic process driven by accident idealizing holism creativity
3. Information systems development is a linear, sequential process idealizing temporal causal chain	4. Information systems development processes are simultaneous, overlapping and there are gaps idealizing fragmentation parallelism disconnectedness
5. Information systems development is a replicable, universal process idealizing generalization consistency formalisms	6. Information systems development occurs in completely unique and idiographic forms idealizing choice change ad hoc
7. Information systems development is a rational, determined, and goal-driven process idealizing goal predetermination process predetermination human cooperation	8. Information systems development is negotiated, compromised and capricious idealizing conflict social constructivism human independence

they were deep in the realm of subjectivity and ambiguity. One incident that illustrates the damaging practice of privileged method involved a complex systems analysis setting in which the users were having difficulty communicating their requirements to the designers. Two separate teams attempted the analysis by following established methods. The first team followed Canonical Database Analysis and Design. The resulting abstract design document was so totally removed from the reality of the problem setting that the client discarded the document and the team was dismissed. A second team attempted to follow a Business Systems Analysis approach. Within a fairly short period, the second team found themselves mired in senseless notation that could not be confirmed by the users. The second team resigned. A third design team attempted a prototyping approach, but just as quickly this approach began to fail. Only when this team began adapting the prototyping method, adjusting and inventing a unique approach to the problem setting, did users and analysts begin making sense out of the problems and the solution designs (Baskerville, 1993).

### *3.1. Privileged: information systems development is a managed, controlled process*

One of the major assumptions advanced through the methodical narrative is that information systems development is effectively managed and controlled by system developers. Methods provide universal mechanisms for achieving this management and control. Logical decomposition is one such mechanism; a “divide and conquer” technique which decomposes uncontrollable wholes into a series of smaller controllable processes. Methods lead the developer through this reductionist process until each subcomponent becomes conceptually manageable and the construction resources become determinable and effectively manageable. This decomposition further assumes that subsystems or components are interdependent slices of a system, with clear boundaries and with each being independently stable and robust (i.e. highly cohesive and loosely coupled). The boundaries and relationships of modules, objects, etc. are products of a rational design process. There is also an economic undercurrent in this assumption. Methods permit developers to “engineer” information flows in an effort to improve the economics of labor and other resources in the production and maintenance of information.

The history of systems development is dominated by methods that idealize and extend this “control by reduction” view of systems analysis, design and building activities (cf. Boland, 1979; Hirschheim, Klein & Lyytinen, 1996; Klein & Kraft, 1994; Kraft & Truex, 1994). For example, structured systems development (e.g. Yourdon & Constantine, 1979) shifted the focus of decomposition and boundaries from physical computing machinery to logical software processes. Information engineering (e.g. Finkelstein, 1989; Martin, 1984) broadened the scope of systems design to an organization-wide scale, expecting a stable collection of data and processing components that would be shared by all organizational activities. This set of components would only need rearrangement and extension in order to adapt and grow. Object-oriented design (e.g. Coad & Yourdon, 1991; Shlaer & Mellor, 1988) broadens the scope still further, seeking stability and development control through reusable and maintainable components across, as well as within, organizational boundaries.

Even methods that do not directly regard functional decomposition assume reduction as a control framework. For example, rapid prototyping (e.g. Carey & Mason, 1986; Naumann & Milton, 1982) introduced a new means for controlling and stabilizing user requirements by iteratively refining components for user experimentation. Prototyping assumes that each iteration will produce a reduced set of problems for which permanent solutions can be fixed. As the design progresses, the problem set is steadily reduced to a level where specifications can be fixed and the final system implemented in a manageable fashion.

### *3.2. Marginalized: information systems development is a random, opportunistic process driven by accident*

The privileged text elevates information systems development as a managed and controlled process. This marginalizes an alternative assumption that building systems

might be a natural outcome of a complex, multivariate setting affected by many uncontrolled events. The marginalized text would assign method a minor, perhaps irrelevant, role in the development process. In order for methods to actually control information systems development, developers must hold nearly perfect knowledge of a throng of interrelated factors. Many of these factors arise in the uncertain, confused arena of social behavior and autonomous human action, making predetermination impossible. Reductionist decomposition is problematic because the isolation of components is an imaginary process, and the resulting components do not reflect the rich interdependencies in organizational reality. Consequently the control is largely imaginary as well. The methodical actions join the mass of other irrational human activities that influence the systems development outcomes.

The marginalized text explains many anomalies of the systems development process, essentially an outcome of the routine failures of the reductionist control. Many methods resort to a “fix-it-up” stage where the developers are given license to holistically repair the irrational and unacceptable designs that proceeded from the application of the method. Examples reach across the history of methods. DeMarco’s (DeMarco, 1979) version of structured analysis included a holistic reconsideration and revision stage. Data base designs require “de-normalization” in order to convert fully normalized designs into feasible specifications (Loomis, 1987). Object-oriented designs need rather amethodical adjustments to combine and separate perfectly rational objects to meet myriad unexpected factors, such as the politics of object libraries (cf. Coad & Yourdon, 1991). Methodologists, couched in the privileged texts, view such repairs as momentary, pragmatic escapes from method. The marginalized, amethodical text adopts an opposing view. This rather amethodical reconsideration of the outcome of the process is necessary because the assembled, imaginary result simply doesn’t correspond with reality. The useful systems development work only happens in these repair stages, and in clandestine “extra-method” activities.

For example, Parnas and Clements (1986) noticed that information systems developers produce documentation which only makes it appear that they followed a rational method. The reality of the design process is a tortured discovery process, and the faked documentation disguises the way simple truths emerged. Much of the really useful information (like why alternatives were rejected) is not recorded:

We will never find a process that allows us to design software in a perfectly rational way. The good news is that we can fake it... The process is ‘faked’ by producing the documents that we would have produced if we had done this the ideal way. (Parnas & Clements, 1986, pp. 251, 256)

The marginalized text explains why this amethodical vision of systems development better describes reality than the privileged methodical view. Information systems development is an art and not a science. Like any other process of creation, it is open to serendipity and chance occurrences that cannot be predicted or managed. This assumption recognizes that systems develop at their own pace and react to

unexpected opportunities with constant and subtle adjustments that make the development project seem to have a life of its own.

For example, prototyping approaches expect user satisfaction to increase as an interactive prototype progresses. The set of user problems should decrease steadily. The privileged text assumes that the specifications steadily improve their alignment with user needs. These assumptions marginalize the possibility that users quickly weary of the process (Alavi, 1984), growing alienated or frustrated, and becoming less inclined to pursue prototype problems. This process also marginalizes the adaptation of the users to the prototype, a circumstance in which the users must force-fit their world onto an imaginary setting required by the method.

### *3.3. Privileged: information systems development is a linear, sequential process*

Another fundamental assumption that inhabits the privileged text involves the straightforward linearity of methodical systems development approaches. Most information systems development methods consist of ordered stages or grouped activities bounded by specific events or completed intermediate products. Indeed, the linear sequence of process (one step following another) provides much of the order and regularity that comprises the defining characteristic of the concept “method”.

The practical problems with such temporal sequences were recognized early. These are compensated with iteration, feedback and backward-loops that enable repetition of earlier stages and events, and the revision of intermediate products. Even the waterfall model was quickly revised so that, in the case of information systems development, water was sometimes allowed to fall “up” a little (cf. Yourdon, 1981/1989; Yourdon & Constantine, 1979).

In fact, this iterative allowance relaxes only slightly the linear assumptions about information systems development. The privileged assumptions about the causal nature of these sequences is largely undisturbed by iteration. Method assumes a progressive nature of the information systems development process that rarely permits information systems development to skip stages, events or intermediate products. Information systems development is a causal chain. The temporal importance of the sequence-progression of events remains firmly anchored (i.e. while some steps may need repeating none may be skipped or rearranged before its time). For example, step three (e.g. develop performance specifications) may be repeated several times, but never before step two (e.g. requirements determination) has been completed at least once. The intermediate products of a previous stage are usually assumed to be a precondition for initiating a following stage. Despite the presence of iteration, a tightly dependent and temporal causal chain is assumed by most methodical texts.

The staging sequence partly justifies a methodical approach. Information systems development methods serve to organize activities. By ordering phases or steps as the sequence of activities, developers improve efficiency by eliminating excessive rework and in-process changes. Further, the causal chain of information systems development methods eliminates irrational activities, such as completing a design element before other elements on which it is dependent. For example, a methodical approach might proscribe the designing of the physical file layout before the analysis

of the logical database elements, since the results of this analysis are likely to cause a revision of the file design.

#### *3.4. Marginalized: information systems development processes are simultaneous, overlapping and there are gaps*

If the information systems building process is random, opportunistic and driven by accident, it is unlikely that information systems development could be marshaled along as a linear, sequential process. Amethodical information systems development is an outcome of myriad development activities that emerge more or less independently. The process is disconnected and fragmented in that information systems developers become engaged in the systems development activities that arise as the project evolves. The events that determine the real systems development sequence of activities are extremely complex and opportunistic. Examples of far-ranging factors that could affect the sequence include the preliminary expertise of the information systems developers in the user problem domain, the availability of development machinery, the availability of users for interviews, and the technical expertise of the developers. Hence, development activities are fragmented and must proceed in parallel rather than in a strict sequence.

For example, Coad and Yourdon (1990) observed analysts and designers at MCC in Austin, Texas, and noted that these professionals tended to work opportunistically rather than methodically. The designers would work at a higher level of abstraction, then see a detailed area, dive into it, investigate it as long as it seemed interesting, and then return to a higher level of abstraction. In fact, the information systems development process is not a sequence at all. Many co-dependent activities continue simultaneously with varying degrees of communication. Stages get skipped when project schedules slip and hard deadlines approach. Some events are felt to be very critical and will be performed as soon as the opportunity arises rather than in the idealized sequence. A particular information systems development project, or some of its activities, may stop altogether for possibly lengthy periods. Interruptions and changing developer expertise lead to unexpected revisions and features. It is possible that the systems development process will produce an entirely different system with totally different purposes than those originally intended. The systems development process may also fade away without anyone taking particular notice that it failed to achieve any closure. Not only must the rationality of the development process must be faked, but the original intentions must also be revised to conform to whatever the outcomes must be. In retrospect, the work progressed in a manner that was partial, incomplete and indeed filled with missteps, dead-ends and backtracking.

The privileged text, with its methodical assumptions, obscures recognition of amethodical development. Any causal chain of dependent intermediate products in an information systems development project is largely imaginary. Any one of the chain of products can be created first, and all others constructed for compatibility. For example, if the project appears to the developers as primarily a data problem, they may choose to grapple immediately with the data analysis and design regardless of any methodical sequence. Afterwards, any “preceding” intermediate products are

created in a fashion that is compatible with the “subsequent”, “dependent” data design. Ideas about product chains and intermediacy are not very relevant to information systems development.

In fact, the assumption that exactly one activity must be first is methodical. The marginalized amethodical text assumes that several activities may begin simultaneously. Compatibility of activity outcomes is not really dependent on their sequence, but rather on cooperation, communication and negotiation. Many incompatibilities arise from this process of assumed and privileged sequentiality. (This is a frequent observation made by practitioners that is easily explained by the marginalized text, yet poorly illuminated by the privileged text.)

### *3.5. Privileged: information systems development is a replicable, universal process*

Methods have a surface appeal because they can seem completely mechanistic, taking highly objective descriptions of organizational requirements and inescapably manufacturing essentially identical successful systems. This text assumes that adherence to method allows greater replicability of a system and that the method can be simply applied in a variety of organizational settings. An important information systems development method is robust in the sense that it leads to dependable systems solutions in different organizational settings. A robust method can be generalized across a wide population of information systems development problems. The object-oriented metaphor of the cookie-cutter (method) and the cookie (a successful systems development outcome) may apply here.

As was the case with the temporal sequence assumption, the privileged text is usually intemperate in its assumptions about universal application. Methods must be adaptable to a degree that allows their use in a variety of settings. The exact process may have to be preconfigured according to the needs of the exact systems development setting. However, the basic methodical framework will remain as an outcome of this reconfiguration. The adapted method is always a recognizable descendant of the original. This retained framework is avoidable because, unless the methodology ignores the adaptability issue altogether, it will usually attempt a predetermination of the adaptation mechanisms. This privileged text assumes the pre-eminence of the methodical concept, thus a “meta-method” for adapting the method unavoidably emerges.

A typical mechanism for adapting methods arises from contingency theory. The exact information systems development method is contingent on certain factors. For example, McFarlan (1981) shows how the pre-existing information systems application portfolio and the political situation of the information systems organization might determine the general approach to systems development. In another example, Davis (1982) uses various organizational factors to determine which information systems development approach is best for a particular problem. Internally, many methods provide interchangeable components that can be selected contingently. For example, information systems developers using structured methods may choose a data-oriented or a process-oriented approach contingent on the nature of the problem

setting which involves a reorganization of the sequence (Yourdon, 1981/1989). Interchangeable notation is often contingent on existing organizational preferences. For example, object oriented methods may permit object behavior to be defined with either pseudocode, flowcharts, or state diagrams.

At a fundamental level, methods achieve a degree of universality by being more or less formal. There is an assumption that the method has been carefully determined and its use in many situations is known to be successful. Deviation from the method is dangerous because failure will follow unless the alterations are very carefully devised. Adherence to a method will lead to consistent, successful outcomes while ad hoc approaches are hit and miss.

The formality is similar to the objective formality of scientific positivism. Scientific methods permit a universal approach to a wide family of problem situations, reduce these problems to an abstract set of symbols (often a diagram), and usually permit elements of the problem space to be symbolically manipulated with a finite set of operations in order to deductively arrive at the solution (cf. Gause & Weinberg, 1989). At a functional level, the information systems development methods introduce structure into the systems development process and regulate activities. These methodical structures provide systems development with taxonomy of activities. This taxonomy reduces the complex set of systems development activities using a formal classification scheme. Methods help achieve economies by reducing redundant activities, which become apparent when similar activities are grouped. Further, the formal elements of information systems development methods assure that activities are comprehensive. Formal rules insure that all necessary work is completed without oversight and eliminate the expense and embarrassment of returning to the project for additional “repair” development after the project’s completion (cf. Yourdon, 1981/1989).

### *3.6. Marginalized: information systems development occurs in completely unique and idiographic forms*

Amethodical privileges local logics instead of global logics. That is, it assumes that singularity is more important than generality. System development is a local narrative, a form of situated action, whereas “method” is a metanarrative of generalized information systems development. Amethodical development assumes that every information system emerges in its own peculiar way. The impact of vastly different variables in the systems development setting demands a completely unique systems development process. These variables include organizational aspects like structures, cultures, resources, constraints, etc., plus individual aspects like talents, habits, predispositions, etc. Organizations are too inconsistent to permit replicable treatments. Each development project outcome arises in a revolutionary new way, not because a methodical framework has been adapted to the setting, but because an entirely new framework was invented for the setting.

One problem that inhibits our understanding of information systems development is our predisposition toward the privileged text of the methodical. Our rational, faked explanations of the information systems development process fail to note how the

predetermined methodical framework collapsed during strategic events and the activities progressed in a manner incompatible with the espoused methodical framework.

Rather than providing a replicable and universal process, methodical frameworks (if attempted at all) are effectively discarded early in the process. Consistency in an approach to systems development is characteristic of information systems developers who choose to mindlessly ignore the really difficult organizational issues that confront them. These developers absorb themselves in their own technology and rarely achieve more than mediocre solutions to the least significant aspects of the organization's needs. Amethodical information systems development is the outcome of the exercise of unprogrammed, independent human choice during every development project activity. Ad hoc approaches and methodical approaches are both hit and miss, but if the methodical approach is truly consistent, it consistently misses.

Fortunately, the predisposition to use "the same approach as last time" is typically abandoned early in a development process because the practical setting does not permit the processes to develop. Information systems development methods are not adapted using contingency techniques. Rather, new components are invented altogether on-the-fly, and the unique framework, while always present, may never even be identified. Information systems developers may claim that certain contingent components were adopted and the framework followed. However, close examination may reveal that the actual activities do not resemble any of the contingent alternatives, and they were highly inconsistent with the claimed framework.

This marginalized assumption explains some observed problems with information systems development methods. Bansler and Bødker (1993) report a triangulated research project in which they discovered that surveys reporting the heavy use of structured analysis in the workplace can be subsequently disproved by in-depth case studies. They first used questionnaire surveys to determine the information systems development method used by an organization. This was followed by a case study approach in which they observed the actual application of this method. They found that despite initial claims that a particular structured method was used, the more intensive investigation showed otherwise. Tools from the structured methods were adopted, but these are combined in unique ways with other tools. Importantly, the structured procedures are not actually used at all.

The amethodical viewpoint also assumes that few organizations undergoing redevelopment of their information systems will actually hold still during the process. Organizations are themselves emerging in a fluid process of internal change and environmental adaptation. Any lengthy systems development process must itself emerge and change or it will slowly become irrelevant to the needs of its client. The contingencies cannot be predetermined because the nature of the organization at the conclusion of the process will be substantially different than the nature of the organization at the outset. Consequently, many of the systems development activities, events and products have to be reinvented along the way. Perhaps even the overarching information systems development framework will transform completely.

This marginalized assumption explains why early systems development activities may be rejected during later phases. For example, Baskerville et al. (1992) presented

two case studies as evidence that long-term information systems development methods are hopelessly ineffective because the organization changes quickly enough to invalidate initial findings. Consequently, the later stages of information systems development become inflicted by largely artificial structures that interfere with the development process. In this study, systems development methods are either subvertly or overtly abandoned and more pragmatic, locally invented approaches are adopted to complete the information systems development project.

The commotion in the underlying organization makes formal aspects of information systems development methods problematic. Early intermediate products, like requirements specifications, are rendered obsolete by the inescapable lapse of time. Predefined budgets obviate the expensive rework. Besides, the time required for the rework may leave subsequent intermediate products equally obsolete. In the end, the systems development outcome convulses into a (perhaps successful) conclusion with every component, intermediate product and activity either discarded or simultaneously reworked. This is why the information systems analysis process is never completed, it just runs out of time (DeMarco, 1979).

In fact, the nature of problems being presented to information systems development have changed in the latest decades. Formal development methods have been used to cleanly solve problems that are amenable to formal methods, and over the years this has left a residue of “messy”, less amenable problems (Gause & Weinberg, 1989). The early successes of formal systems development methods has led to boundary extensions of the problem domain to include a new target population of problems. Now the problem space for formal methods has become one that is dominated by problems that are highly unsuited to such formalisms. These problems are fraught with volatility, exceptions, unstructured data and unpredictable processing requirements.

This marginalized assumption also helps explain why our understanding of methodical information systems development is blotted by conflicting research findings, disagreements about concepts, contradictory theories and glaring gaps in our body of knowledge concerning this activity. We have conflicting evidence as to whether systems development methods are ever used; or that they work successfully if indeed they are ever used (Wynekoop & Russo, 1993). Much of the existing research misses the fundamental questions:

In short, methodology knowledge is based on conceptual writings and studies of small systems in unrealistic contexts, augmented by surveys that often compare unspecified methodologies (e.g., comparing prototyping to SDLC use)... We do not understand how methodologies are selected or adapted or how they should be selected—or if they should be selected at all. We don’t know how methodologies are used or how effective they are. (Wynekoop & Russo, 1993, p. 186)

### *3.7. Privileged: information systems development is a rational, determined, and goal-driven process*

The fundamental sequence of events in many information systems development methods requires the determination of a set of goals prior to systems analysis and

implementation of actions to achieve these goals. There is a shared assumption in this privileged text about a three-stage rational sequence: (1) determine goals, (2) determine steps and events that lead to these goals, (3) follow the steps and generate the events. This assumption closely follows the other privileged assumptions like causal linearity, reductionism and universality, but it does take these ideas one step further by assuming that information systems developers will share the espoused goals and faithfully adhere to the plan by exercising their rational powers.

Thus, pre-identified goals serve as targets and motives for reasoned, well-considered and explicitly chosen sets of activities. For example, structured methods, information engineering, and prototyping methods situate information systems planning in the context of organizational goals and strategies. The decision about which systems are to be built and the order in which they are to be built is made on the basis of rational criteria which most logically aid the organization in meeting its given and explicitly stated goals.

This process is reminiscent of the “normal” scientific ideals that have been, through university training, inculcated in many information systems developers: (1) state the theory, (2) operationalize the hypothesized relationship between variables, (3) manipulate the independent variables to achieve the desired effects on the independent variables. Information systems practitioners replace scientific rigor with professional rigor. They acquire positive knowledge about their settings, and by rational adjustments to systems components, they exercise positive control over their social environment. As professional techniques, information systems development methods borrow epistemological assumptions from their scientific reference disciplines. Essentially they provide a structural framework for the acquisition of knowledge about the environment following the data collection procedures of the natural sciences (cf. Nagel, 1961; Neurath, 1939). These systems development methods are the paradigm within which professionals learn about their surroundings and project their control over these surroundings.

### *3.8. Marginalized: information systems development is negotiated, compromised and capricious*

The amethodical view makes rather different assumptions. First, the outset of a development project will find little agreement about a set of goals for the process of changing the system. In order to reach any espoused agreement the goals must be broad, unspecific, and ambiguous. The mechanism of ambiguity is used to permit human actors to interpret these vague goals differently. This tacit disagreement of individual interpretation leads to a systems development reality characterized by many conflicting sets of goals. It is extremely difficult to apprehend these conflicts for two reasons. First, the individuals operate within a dichotomy of theory-in-action contrasted with espoused theory (Argyris & Schön, 1978) which may prevent them from understanding and pronouncing their own set of goals in a predetermined way. That is, individual goals will be discovered as the information systems development process unfolds. A realistic set of unambiguous, achievable goals cannot be predetermined.

A second assumption holds that organizational reality is socially defined. The infor-

mation systems developers' knowledge about their environment is influenced by habituated behavior institutionalized values, politically reified observations, etc. (Berger & Luckmann, 1966). The epistemological basis of system developers' knowledge about their environment is interpretive rather than positive. The organizational reality is subject to revisions as the information systems development project unfolds. These revisions may even be violently twisted, giving the project a "through-the-looking-glass" setting.

Third, the human actors in the development process do not always behave predictably and rationally. They are sometimes spiteful, playful, stubborn, capricious, eccentric, etc. Some people will be creative and ambitious while others will be unimaginative and removed. This behavior frames the social context in which the information systems developers will negotiate the development project through the innately conflicting sets of goals. The negotiated aspect of the systems development outcome is significant because it may very well tie the final system characteristics to a very different set of purposes and actors than those that may have ostensibly predetermined an expected outcome. Individual charisma, persuasiveness, political power or tenacity may become a paramount influence.

Rather than being a rational, purposive, and goal-driven process, information systems development is subject to human whims, talents, and the personal goals of the actors involved. From the amethodical view, information systems development is a process by which some questionable observations are used to construct a set of ambiguous goals and disputable steps that are handed over to a group of independently minded sprites who will proceed to do whatever they want to anyway.

This marginalized text seems to explain some observations about information systems developments that are problematic for the privileged assumptions. For example, Brooks (1987) acknowledges the shortcomings of development methods because they cannot displace human inspiration and creativity; they cannot "inflare or inspire the drudge" (p. 19). For Brooks, the important component in information systems development is a great designer, a talented human being. Lee (1991) recognized that a preoccupation with positivist science as a criteria for our reference disciplines has led to systems development methods which ignore human meanings and human values as fundamental elements of systems designs. He praises the merits of non-methodical, "creative" professions, such as architecture, as more appropriate reference disciplines for the field of information systems development. Beath and Orlikowski (1994) deconstruct the relationship between the information systems professional and the user in the context of the information engineering method. They found an ideology of user involvement that contradicted a deeply embedded dichotomy between users and analysts that reinforced an ambivalence to user concerns. This dichotomy is itself privileged, with the analysts dominant and the users submissive. This dichotomy conflicts with the user responsibility for ultimately using the system. Because this dichotomy might be shared by most development methods, the issue has practical implications for information systems development in general:

We believe the findings discussed in this paper raise a central challenge for the field of IS. In general, both the research and practice of systems development have

tended to take for granted rather than question the structural distribution of power, authority, knowledge, control and resources that constitute the institutional context of systems development. (Beath & Orlikowski, 1994, p. 375)

These marginalized assumptions also help explain why the exact same design problem can engender completely different design solutions. For example, Turner (1987) studied the implications of wide variance in design solutions created by student designer teams. He suggested that many development methods disallow important design processes such as the “creative leap” by which conflicting choices are elegantly reconciled using an inspirational merger of opposing objective analyses. Naur (1993) studied the use of systems development methods from the perspective of their suitability for the variety of human beings that may be drawn into the systems professions. Naur found that designers will fundamentally disagree on how development methods were intended to be practised. He concludes that no method can hope to find universal practice because every method must be inherently unsuitable to some large portion of the population of professionals who are charged with systems design.

The amethodical view suggests that each development project must actually include activity (however informal) that defines its own unique approach to solving each development problem. The amethodical assumptions suggest that information systems development is a pastiche of activities, events and products. Some of these elements may be borrowed from various methods (including scientific methods and information systems development methods), some elements borrowed from elsewhere (like art, literature, or architecture), and some elements invented altogether. Each element of the development pastiche is very likely to violate, in some way or another, the structures of any particular information systems development method. Like an emergent grammar, information systems development is a collection of systems idioms, proverbs, clichés, formulas, jargon, favored technologies, habitualized interfaces, typical notation, and so on. Information systems development is pasted together “on-the-fly”, in a manner that is interactive and negotiated with the organizational actors and the rest of the environment of the systems development project.

#### **4. Implications for the goals of information systems development**

We admittedly bifurcate the concepts of method and amethod into opposing texts in each of the preceding sections. In one sense, this bifurcation did not raise a marginalized text from a privileged one, but instead created a second privileged text. The newly privileged text of amethodical was in effect presented as more important than the methodical counterpart. Deferral of meaning accepts that neither of these texts is the final word. The marginalized positions that we present are themselves imbued with cultural and intellectual assumptions and should be subject to a further deconstruction. The goal of deferral is not to displace one view with the other, nor is it to provide a kind of synthetic combination or compromise of the various views. Rather, one goal is to hold both views in mind simultaneously.

The danger in simply raising the marginalized views as a replacement for the privi-

leged view might be thought of as the “costs” of the marginalized view. For example, the marginalized view makes various stakeholders’ goals dominant over management’s goals. The amethodical text raises the importance of systems developers listening to many voices, serving many masters and eschewing any single, clear purpose. The cost of this shift is that of clarity of purpose. In the privileged, mainstream view, management’s goals were, for good or ill, clear, univocal and cheaper to fulfill from a transaction cost point of view.

There are other important examples of the costs of privileging the amethodical view. Embracing ambiguity and deferring the freezing of organizational and institutional structures surrenders the precision of the expected systems development outcomes and requirements definition. The amethodical view supports conflict over consensus. This view requires developers to attend different voices and interests. Political bargaining and negotiation lead to compromise solutions rather than assuming a consensus view of organizational settings. The cost is in the overhead of maintaining procedures for more or less constant negotiation and adjustment of ambiguous goals to changeable political decisions.

The amethodical view appreciates innovation and “organizational shake-ups” that lead to adaptation, experimentation and in turn to accidents and opportunism. There is a cost in allowing trial and error, and in adapting to accidents. The resultant solutions are as likely to represent a misfit as they are to represent a good fit. The mainstream view favors calculated organizational fits and matches such that the system mimics the expected characteristics of the organization.

By privileging the amethodical we ignore method’s nature as a systems development “coping” mechanism. Method provides a means for developers to cope with the essential fluidity of the social organization by ignoring continuous change and freezing a particular organizational view in time. By privileging the amethodical we ignore the familiar and pervasive aspects of method that guide and balance developers, even those who might be unknowingly engaged in amethodical development. Developers cope through the use of method as a parable rather than a procedure. Methods guide rather than direct the developer’s process. Method is a means to frame the domain of inquiry and establish an initial conceptual and intellectual boundary on the problem. Privileging only the amethodical ignores the way that experienced developers intuitively approach the development process as a professional language game. In these particular language games the initial rules (or professional grammars) include the information systems development method itself. Certainly the game and its rules evolve in use. This evolution regards the engagement between developers and their methods, and defines part of their professional discourse. This discourse describes the negotiation process through which the rules (method) are changed during development process. The amethodical view marginalizes the way this language game commences (e.g. the original set of rules or the method) in favor of privileging the way this game evolves.

By privileging the amethodical we lose the important role of methods as design metaphors that is often made by those who practice information systems development (Coyne, 1995). In creating their structured systems approach Yourdon and DeMarco were, after all, simply distilling their examples of best practice. Books presenting methodical approaches typically offer exemplars of successful development practice. Like

good poems that work as distilled symbolic representation of some aspect of the human condition, these exemplars are simply distilled metaphors for the ideal development effort.

By privileging the notion of organizational emergence we marginalize those aspects of organizational life which are regular and appear to have structure. Because humans engage in social organizing activities, there must be a tacit agreement to engage in socially constructive activities that, in turn, leads to organizational regularities. The amethodical marginalizes social agreements that persist long enough to become organizational features in need of systems development modeling.

By privileging the amethodical we also marginalize the history of successful methodical systems development in large-scale projects. Large-scale systems involve hundreds of developers and multiple development teams. These projects require, at minimum, an agreement on which organizational language (e.g. method) will be spoken as a starting point in the project discourse.

These counter-amethodical positions are themselves becoming privileged texts that involve the marginalization of other positions. These texts involve costs and invite further deconstruction. Let us illustrate with a further deconstruction of our recognition that methods provide the starting point for the rules in a language game that marginalizes the evolution of method. Once this initial rule-set consisted of structured methods; for many organizations this initial rule set has evolved into object-oriented methods. In this sense, systems development projects are continuous prototypes for the development of new methods. As a second example, our initial deconstruction privileges the amethodical perspective and marginalizes the psychological (individual and social), organizational, institutional reasons for method. Structuration theory illustrates these marginalized reasons by emphasizing the importance of structure on organizational learning and illustrating how the reification of practice creates structures, which in turn, serve to guide and constrain action and the definition of new structures. Our deconstruction subjugates structure in favor of action, thereby ignoring a justification for method because it depends on routinizing lessons learned from experience. More detailed examples of organizational, institutional and psychological reasons for method are discussed in Baskerville et al. (1992).

Method and amethodical come together in the tension of the developer's individual lived experience. As experience and theory (espoused method) intersect, they frequently contradict one another. Method privileges replicability, standardization, traceability, etc., whereas practice is about getting the job done. These examples illustrate why the deconstruction of method can be turned and turned. Method and amethod come together to form a language game of their own.

## **5. Conclusion**

Four questions were posed in the introduction to this article. We can draw insight into these issues from the deconstruction of the privileged method text. This insight also addresses the problems related to this text through important implications for research, practice and education.

First we asked whether information systems development methods really describe ongoing systems development practice and whether they explain why information systems are developed in certain ways. The strength of the amethodical reality suggests that we have a very limited understanding of how systems are developed. Methodical assumptions clearly define the paradigm of modern systems development thought so strongly that little research effort has been brought to bear on measuring amethodical phenomena. Notwithstanding the efforts directed at learning how to examine and analyze information systems development processes (Newman & Robey, 1992; Robey & Newman, 1996; Sabherwal & Robey 1993, 1995), it is doubtful that our instruments are presently capable of measuring such phenomena if the state of the disciplinary discourse does not admit them as possibilities. Einstein once described the confinements of such strong theoretical assumptions: “It is the theory that decides what we can observe” (Root Bernstein, 1989, p. 419). Methodical development so dominates our thinking that it has become a self-confirming construct, impossible to disprove in modern terms.

The marginalized text suggests that information systems development unfolds differently than we previously believed and that developers adapt methods to particular situations. Developers are successfully mixing and matching elements from seemingly contradictory systems methods, e.g. structured (cf. Connell & Shafer, 1989). We now have open research questions about these amethodical phenomena and the criteria for a successful amethodical systems development project. These implications include the very real need to develop instruments and research approaches that permit us to measure the degree to which information systems development is methodical and amethodical.

We also asked whether methods actually function as frameworks, formulas or templates of successful information systems development. This question raises the implications of the amethodical text for practice. Our improved understanding of the contrasting information systems development texts legitimizes idiographic, one-shot development schemes that arise “on-the-fly” as a negotiation between systems development actors and the systems development environment. That is, this understanding permits project managers to relieve system developers of the need to “pretend” that a method is followed absolutely. Project managers are free to be more aware of how the system is actually being constructed. These managers would be encouraged to focus on the control of, and the value in, the dissimilarities (the pastiche) of the actual development activities. If method becomes an obvious fiction in an information systems development project, project managers could learn to recognize and benefit from the adaptive flux of the amethodical aspect of the process.

In addition, the amethodical text suggests a dominance of notation over process in the selection of a systems development method. Information systems development methods often embrace a notation set as well as a process for developing the notation. Possibly the adoption of the notation is more critical for defining how the systems development process will unfold, since the notation defines what can and cannot be represented by the analysis and design. This criticality suggests that systems development practitioners faced with decisions regarding method adoption give their strongest consideration to the notation prescribed by the method, rather than the prescribed pro-

cedures or heuristics. Indeed, the most practical notation set might well be pasted together from various sources.

Finally, we asked if methods were unattainable ideals and hypothetical “straw men” that provide normative guidance anchored in utopian development situations? Here we find educational implications, since the presentation of information systems development methods also can be informed by the amethodical presence. Methods seem more like idealizations than prescriptions, and might better be presented as “cases” or “exemplars” rather than practical frameworks. This shift reveals the need to present a set of sound examples of how parts of various systems development methods can be mixed and matched (perhaps with other, newly invented parts); plus examples of how development approaches can be assembled “on-the-fly” by cannibalizing bits from the various ideals found in the textbooks.

A further need in our information systems development classrooms is for the concepts and criteria for innovating one-shot information systems development approaches and negotiating the development activities with the development environment. Because of our methodical paradigm, we don’t actually know how this happens “in the wild”. Students need guidelines for recognizing elements of the ideal settings for discrete parts of development methods. For example, how and when should developers separate the notation from the procedure?

The answer to the questions raised in the introduction do not reside in definitive solutions or prescriptions for action. Rather in the fact of the asking and answering they become part of the endless chain of signification, the différance and the deferral that keep these notions at play. By opening questions about systems development without method, we may best clarify what it means to use a method in developing an information system.

By adopting a single domineering concept of method all of our thinking about information systems development becomes imprisoned by this one concept. *The method* is not only our way of thinking about systems development, it is our way of thinking about “thinking about systems development”. Just as there is an intellectual cost incurred by privileging the marginal (amethod), we must remember that the price of one domineering concept (method) is a tightly defined boundary for our knowledge about certain activities. In information systems development, this price includes the perceptions that we forego by assuming an unwavering methodical viewpoint. When the idea of method frames all of our perceptions about systems development, then it becomes very difficult to grasp its non-methodical aspects. It limits not only our understanding, but our empirical observations of how the development of information systems will unfold in a human organization. In other words, our obsession with method can cause us to ignore activities that do not fit within a methodical frame.

## Acknowledgements

The authors acknowledge and appreciate the strong influence on this paper of discussions with Heinz K. Klein. We also gratefully acknowledge the comments of Keld Bødker, Paul Cule, Gordon Everest, Bernie Glasson, Mark Keil, Finn Kensing, John

King, Jungwoo Lee, Kalle Lyytinen, Eph McLean, Wanda Orlikowski, Dan Robey, Steve Sawyer and Jon Turner as well as the anonymous reviewers. This research was supported, in part, by a research grant from the College of Business Administration of Georgia State University.

## Appendix A

For readers less familiar with postmodernism, there are several key ideas that are assumed in the body of the paper. These ideas regard postmodern thinking as it relates to organizational settings and post-structural organizational thought. One aspect of postmodernism implies that important values and beliefs underpinning modern religion, philosophy, art and science have imposed limits on human thought and activity. Breaking free of these limits releases the discovery of unnoticed dimensions in our world. This emancipation not only involves the uprooting of fundamental modern assumptions, but the even more difficult task of finding and exposing these unquestioned beliefs. Postmodernism is not a philosophy as such. Rather it is a collection of ideas, a pastiche of related concepts which have been arising in numerous disciplines since the end of the First World War including sociology (Foucault, 1970) philosophy and literary criticism (Lyotard, 1987, #462) and linguistics (“Structure, sign and play in the discourse of human sciences”, pp. 278–293 in Derrida 1978, 1982). Such postmodern exploration is also found in organizational analysis (e.g. Cooper, 1989; Cooper & Burrell, 1988; Kilduff, 1993), organizational theory (e.g. Bergquist, 1993; Boje & Dennehy, 1994; Coyne, 1995) and in information systems development methodology (e.g. Coyne, 1995).

Another aspect arises in the decentering of the author of the text. This decentering means that the intentions of the reader become paramount. The author, like the subject, is contingent, temporary, situational and ephemeral. What an author *meant* to say is immaterial because it is never recoverable, is itself clad in chains of signification and is, anyway, the subject of interpretation involving other (the readers’) chains of signification. “Chains of signification” is an important semiotic concept in the work of structural linguists such as Saussure and Piaget. But there is an important twist recognized by Derrida and adopted by nonstructural linguists such as Paul Hopper. This twist acknowledges that chains of signification are a type of infinite recursion, always at play and never returning to an initial point of reference or the “transcendental ideal.” Literature, art and science depend on this concept. Should the ideal meaning of every utterance become known there would be no need for further discussion. Thus, one can never claim to have “captured” an author’s intention.

The marginalized texts of amethodical development benefit from a fundamental undercurrent in non-structural thinking about organizations, i.e. the concept of *emergence* (cf. Truex & Klein, 1991). Unlike structuration theory (Giddens 1979, 1984), which focuses on a structural duality, emergence theory is a complete alternative to the structure concept in science, grammar, and organizations. Emergence is the notion that human activity is dominated by change: “destructure”, not structure. Reality, knowledge, grammar, etc. are in motion: no “states” exist, just “flux”. The emergence

concept also suggests why structure seems to appear. Emergence points to a social “need to structure”, arising from language use and formation in discourse. Non structural linguists describe it as an attempt to achieve structure where:

...like speech itself, [grammar] must be viewed as a real-time, social phenomenon, and therefore is temporal; its structure is always deferred, always in a process but never arriving, and therefore emergent. (Hopper, 1987, pp. 141–142)

Emergence is not a sequence of succeeding structures, but a continual movement toward structure without every attaining a steady-state. Emergence is the postponement or deferral of structure, leaving grammars, organizations and science as provisional, negotiable, and epiphenomenal.

The denial of structure as a preexisting and required condition of grammars in linguistics supports a similar denial of structure as a preexisting and required condition in organizations, thus giving rise to the concept of emergent organizations. This emergent organizational concept regards adaptation and flexibility as a dominant feature of the organizational landscape which requires an information systems development approach that acknowledges and adapts to emergent organizational phenomena.

Thus, in the second instance or marginalized notion of the amethodical, we note that the organization undergoing modeling of its functions and data meanings is indeed a shadow of that which it is not. The organization may be described as a meaning-creating system and one simultaneously engaged in the deferral of meaning. Similarly the system being developed may be viewed as one that defers meaning. This may be illustrated as follows. Systems are derived from symbolic representations of organizational function, processes or data. The information systems is constructed from specifications that are themselves derived from the requirements of a thing in transition. The information systems is already a third or fourth order signification of the organizational element being modeled. There is a continuous chain of signification at each stage pointing backward to a fluid and transient (e.g. emergent) phenomena: A social organization.

## References

- Alavi, M. (1984). An assessment of the prototyping approach to information systems development. *Communications of the ACM*, 27, 556–563.
- Argyris, C., & Schön, D. (1978). *Organizational learning: a theory of action perspective*. Reading: Addison-Wesley.
- Avison, D. E., & Fitzgerald, G. (1988). *Information systems development: methodologies, techniques and tools*. London: Blackwell Scientific.
- Bansler, J., & Bødker, K. (1993). A reappraisal of structured analysis: design in an organizational context. *ACM Transactions on Information Systems*, 11 (2), 165–193.
- Baskerville, R. (1993). Semantic database prototypes. *Journal of Information Systems*, 3 (2), 119–144.
- Baskerville, R., Travis, J., & Truex, D. P. (1992). Systems without method: the impact of new technologies on information systems development projects. In K. E. Kendall, K. Lyytinen, & J. I. DeGross, *Transactions on the impact of computer supported technologies in information systems development* (pp. 241–260). Amsterdam: Elsevier Science.

- Beath, C. M., & Orlikowski, W. J. (1994). The contradictory structure of systems development methodologies: deconstructing the IS–user relationship in information engineering. *Information Systems Research*, 5 (4), 350–377.
- Berger, P. L., & Luckmann, T. (1966). *The social construction of reality: a treatise in the sociology of knowledge*. New York: Anchor Press.
- Bergquist, W. (1993). *The postmodern organization: mastering the art of irreversible change*. San Francisco, CA: Jossey-Bass.
- Boje, D. M., & Dennehy, R. F. (1994). *Managing in the postmodern world: America's revolution against exploitation*. (2nd ed.). Debuque, IA, USA: Kendall/Hunt.
- Boland, R. J. (1979). Control, causality and information system requirements. *Accounting, Organizations and Society*, 4 (4), 259–272.
- Brooks, F. P. (1987). No silver bullet: essence and accidents of software engineering. *Computer*, 20 (4), 10–19.
- Burrell, G., & Morgan, G. (1979). *Sociological paradigms and organisational analysis*. London: Heinemann.
- Calmes, F., Charbonnel, G., & Dumas, P. (1991). *Merise et Ossad: deu méthodologies à comparer in Autour et a L'Entour de Merise*, April, Sophia Antipolis.
- Carey, T., & Mason, R. (1986). Information system prototyping: techniques, tools and methodologies. In W. Agresti, *New paradigms for software development* (pp. 48–58). Washington, DC, USA: IEEE Press.
- Checkland, P., & Scholes, J. (1990). *Soft systems methodology in practice*. Chichester, UK: John Wiley.
- Checkland, P. B. (1984). Systems theory and information systems. In T. M. A. Bemelmans, *Beyond productivity: IS development* (pp. 9–23). Amsterdam: North-Holland.
- Coad, P., & Yourdon, E. (1990). *Object-oriented analysis*. Englewood Cliffs, NJ, USA: Prentice Hall.
- Coad, P., & Yourdon, E. (1991). *Object-oriented design*. Englewood Cliffs, NJ, USA: Yourdon Press.
- Connell, J., & Shafer, L. (1989). *Structured rapid prototyping: an evolutionary approach to software development*. Englewood Cliffs, NJ, USA: Prentice Hall.
- Cooper, R. (1989). Modernism, post modernism and organizational analysis 3: the contribution of Jacques Derrida. *Organization Studies*, 10 (4), 479–502.
- Cooper, R., & Burrell, G. (1988). Modernism, postmodernism and organizational analysis: an introduction. *Organization Studies*, 9 (1), 91–112.
- Couger, J. (1982). Evolution of system development techniques. In J. Couger, M. Colter, & R. Knapp, *Advanced system development/feasibility techniques* (pp. 6–13). New York: John Wiley.
- Couger, J. D. (1973). Evolution of business system analysis techniques. *Computing Surveys*, 5 (3), 167–198.
- Coyne, R. (1995). *Designing information technology in the postmodern age: from method to metaphor*. Cambridge, MA, USA: MIT Press.
- Davis, G. B. (1982). Strategies for information requirements determination. *IBM Systems Journal*, 21 (1), 4–30.
- DeMarco, T. (1979). *Structured analysis and systems specs*. New York: Yourdon Press.
- Derrida, J. (1978). Structure, sign and play in the discourse of human sciences, *Writing and difference* (pp. 278–293). Chicago, IL, USA: University of Chicago Press.
- Derrida, J. (1982). *Différance*. Chicago, IL, USA: University of Chicago Press.
- Eisenberg, E., & Goodall, H. L. (1993). *Organizational Communication: Balancing Creativity and Constraint*. Chapter 5: “Cultural studies of organizations and communications”. New York: St. Martin's Press.
- Eisenhardt, K. M. (1989). Agency theory: an assessment and review. *Academy of Management Review*, 14 (1), 57–74.
- Finkelstein, C. (1989). *An introduction to information engineering: from strategic planning to information systems*. New York: Addison-Wesley.
- Foucault, M. (1970). *The order of things: an archeology of the human sciences*. New York: Vintage Books.
- Gause, D., & Weinberg, G. (1989). *Exploring requirements: reality before design*. New York: Dorset House.
- Giddens, A. (1979). *Central problems in social theory: action, structure and contradiction in social analysis*. Berkeley, California: University of California Press.
- Giddens, A. (1984). *The constitution of society: outline of the theory of structuration*. Cambridge: Polity Press.
- Hirschheim, R., Klein, H., & Lyytinen, K. (1996). Exploring the intellectual structures of information sys-

- tems development: a social action theoretic analysis. *Accounting, Management and Information Technology*, 6(1–2), 1–64.
- Hirschheim, R., & Klein, H. K. (1989). Four paradigms of information systems development. *Communications of the ACM*, 12, 1199–1216.
- Hirschheim, R., & Klein, H. K. (1992). A research agenda for future information systems development methodologies. In W. W. Cotterman, & J. A. Senn, *Challenges and strategies for research in systems development*. New York: John Wiley and Sons.
- Hirschheim, R., Klein, H. K., & Lyytinen, K. (1995). *Information systems development and data modeling*. Cambridge, UK: Cambridge University Press.
- Hopper, P. (1987). *Emergent grammar*. *Berkeley Linguistics Society*, 13, 139–157.
- Jackson, M. C., & Keys, P. (1984). Towards a system of systems methodologies. *Journal of the Operational Research Society*, 35, 473–486.
- Kilduff, M. (1993). Deconstructing organizations. *Academy of Management Review*, 18 (1), 13–31.
- Klein, H. K., & Kraft, P. (1994). Social control and social contract in netWORKing. *Computer Supported Cooperative Work*, 2, 89–108.
- Kraft, P., & Truex, D. (1994). Postmodern management and information technology in the modern industrial corporation. In R. Baskerville, & S. Smithson, *Information technology and new emergent forms of organizations* (pp. 113–127). Amsterdam: Elsevier Science.
- Kumar, K., & Welke, R. J. (1992). Methodology engineering: a proposal for situation-specific methodology construction. In W. W. Cotterman, & J. A. Senn, *Challenges and strategies for research in systems development* (pp. 257–269). Chichester, UK: John Wiley and Sons.
- Lee, A. S. (1991). Architecture as a reference discipline for MIS. In H. E. Nissen, H. K. Klein, & R. A. Hirschheim, *Information systems research: contemporary approaches and emergent traditions* (pp. 573–592). Amsterdam: North-Holland.
- Loomis, M. (1987). *The database book*. New York: Macmillan.
- Liotard, J.-F. (1987). The postmodern condition. In K. Baynes, J. Bohman, & T. McCarthy, *After philosophy* (pp. 73–93). Cambridge, MA, USA: The MIT Press.
- Lyytinen, K. (1987). Different perspectives on information systems: problems and solutions. *ACM Computing Surveys*, 19 (1), 5–46.
- Martin, J. (1984). Information engineering. In L. Mihatov, *An information systems manifesto* (pp. 83–99).
- McFarlan, F. W. (1981). Portfolio approach to information systems. *Harvard Business Review*, 59, 142–150.
- Nagel, E. (1961). *The structure of science: problems in scientific explanation*. London: Routledge and Kegan Paul.
- Naumann, J. D. J., & Milton, A. (1982). Prototyping: the new paradigm for systems development. *MIS Quarterly*, 6 (3), 29–44.
- Naur, P. (1993). Understanding Turing's universal machine: personal style in program description. *The Computer Journal*, 36 (4), 351–372.
- Necco, C., Gordon, C., & Tsai, N. (1987). Systems analysis and design: current practices. *MIS Quarterly*, 11, 461–476.
- Neurath, O. (1939). Foundations of the social sciences. In O. Neurath, R. Carnap, & C. Morris, *Foundations of the unity of science* (pp. 1–53). Chicago, IL, USA: University of Chicago Press.
- Newman, M., & Robey, D. (1992). A social process model of user analyst relationships. *MIS Quarterly*, June, 249–266.
- Nolan, R. (1979). Managing the crisis in data processing. *Harvard Business Review*, March–April, 115–126.
- Oliga, J. (1988). Methodological foundations of systems methodologies. *System Practice*, 1, 87–112.
- Parnas, D. L., & Clements, P. C. (1986). A rational design process: how and why to fake it. *IEEE Transactions on Software Engineering SE*, 12, 251–257.
- Robey, D., & Newman, M. (1996). Sequential pattern and information systems development: an application of a social process model. *ACM Transactions on IS*, 14, 30–63.
- Rochfeld, A., & Tardieu, H. (1983). MERISE: an information system design and development methodology. *Information and Management*, 6 (3), 143–159.
- R. S. Root Bernstein. (1989). *Discovering: inventing and solving problems at the frontiers of scientific knowledge* (p. 419). New York: Harper and Row.

- Rosenau, P. M. (1992). *Post-modernism and the social sciences*. Princeton, NJ, USA: Princeton University Press.
- Sabherwal, R., & Robey, D. (1993). An empirical taxonomy of implementation based on sequences in information systems development. *Organization Science*, 4 (4), 548–576.
- Sabherwal, R., & Robey, D. (1995). Reconciling variance and process strategies for studying information systems development. *Information Systems Research*, 6 (4).
- Shlaer, S., & Mellor, S. (1988). *Object-oriented systems analysis: modeling the world in data*. Englewood Cliffs, NJ, USA: Yourdon Press.
- Truex, D. P., & Klein, H. K. (1991). A rejection of structure as a basis for information systems development. In R. K. Stamper, P. Kerola, R. Lee, & K. Lyytinen, *Collaborative work, social communications and information systems* (pp. 213–235). Amsterdam: Elsevier Science Publishers B.V. (North-Holland).
- Turner, J. (1987). Understanding the elements of system design. In R. J. Boland, & R. A. Hirschheim, *Critical issues in information systems research* (pp. 97–111). Chichester, UK: John Wiley and Sons.
- Wynekoop, J., & Russo, N. (1993). System development methodologies: unanswered questions and the research–practice gap. Paper presented at the the 14th International Conference on Information Systems, Orlando, FL.
- Yourdon, E. (1981/1989). *Modern structured analysis*. Englewood Cliffs, NJ, USA: Yourdon Press.
- Yourdon, E., & Constantine, L. (1979). *Structured design*. Englewood Cliffs, NJ, USA: Prentice-Hall.