

2 Towards an Understanding of the Conceptual Underpinnings of Agile Development Methodologies

Sridhar Nerur, Alan Cannon, VenuGopal Balijepally, Philip Bond

Abstract: While the growing popularity of agile development methodologies is undeniable, there has been little systematic exploration of its intellectual foundation. Such an effort would be an important first step in understanding this paradigm's underlying premises. This understanding, in turn, would be invaluable in our assessment of current practices as well as in our efforts to advance the field of software engineering. Drawing on a variety of sources, both within and outside the discipline, we argue that the concepts underpinning agile development methodologies are by no means novel. In the tradition of General Systems Theory this paper advocates a transdisciplinary examination of agile development methodologies to extend the intellectual boundaries of software development. This is particularly important as the field moves beyond instrumental processes aimed at satisfying mere technical considerations.

2.1 Introduction

New approaches to managing software development continue to capture the attention of both practitioners and academic researchers. Across a variety of intellectual traditions, the “conversation” has focused on social, cultural and philosophical underpinnings of problem-solving approaches. Of late this dialogue has begun to be dominated by discussions of the Agile Development Methodology (ADM) (Boehm 2002; Boehm and Turner 2004; Cockburn and Highsmith 2001; Highsmith 2002; Larman 2004), a development approach emphasizing problem-framing and problem-solving techniques that are radical departures from earlier software methods.

While we see the rationale for the fundamental thrusts of the ADM conversation, we are also struck by the similarity of these elements with principles that were articulated long before ADM ever emerged. ADM's emphasis on individuals, for example, seems little removed from Sociotechnical Systems thinking (Cherns 1976; Cherns 1987) and roots that extend as far back as the 1950s. The

same can be seen in other conceptual foundations of ADM. For example, Larman (2003) mentions several approaches - including some software development practices - that used incremental iterations. In fact, the plan-do-study-act (PDSA), an iterative quality improvement cycle, may be traced back to the 1930s (Larman 2003).

We contend that the underlying assumptions of ADM are not novel in any sense. Concepts such as iterative development, learning, self-organization, reflective practice, self-directed teams and stakeholder participation, to name but a few, have evolved separately in other disciplines, and one or more of them were used in some form in software development as well. As Ludwig von Bertalanffy, the father of General System Theory (GST), points out, "Not only are general aspects and viewpoints alike in different fields of science; we find also formally identical or isomorphic laws in completely different fields." (1950, pg. 136). The goal of this paper is not to unravel the isomorphic principles that ADM shares with other fields, but to explore its conceptual underpinnings from several perspectives. In particular, our work here is an attempt to spell out some of the antecedents of ADM. Such a clarification is an important first step in understanding the underlying assumptions of this paradigm. This, in turn, would be invaluable in our assessment of current practices as well as in our efforts to advance the field of software engineering.

The paper is organized as follows. First, we explore how software development today is: a) more technically complex; b) more strategic; and c) brings to the fore the divergent viewpoints of a wider variety of stakeholders. This leads to the inescapable conclusion that software development, both today and in the future, is and will continue to be undertaken in a much more uncertain context. Second, we highlight the similarity of several ADM principles to long-established intellectual thrusts, both within and beyond the software development field. Third, we provide a brief description of Chernes's (1976; 1987) sociotechnical systems (STS) principles, with a view to focusing attention on their likeness to the core tenets of ADM. Fourth, the paper attempts to position ADM in the system of systems methodology (SOSM) framework presented by Jackson (2003). We then present the implications of our work for research and practice, followed by conclusions.

2.2 The Challenges of Contemporary Software Development

As information technology's role in the modern economy has grown in importance, software developers have found themselves confronted with challenges of unprecedented complexity. Some of this complexity can be attributed simply to technology that has itself grown in both scale and scope. When scholars and software professionals refer to today's software development as the solving of "wicked problems" (Nerur and Balijepally 2007; Poppendieck 2002), they are not simply considering technical issues. Rather, software development today forces

developers to interact with and consider the viewpoints of a wide variety of stakeholders, many of whom have conflicting views on the desirability of the software's features and functionality.

Make no mistake: Even when it does not involve navigating through these conflicting viewpoints, software development remains a formidable challenge. As software is developed for ever-more-novel applications, the fact that developers may have limited experience and incomplete information about what can and cannot be done (and what should or should not be tried) makes ADM's principle of iterative learning/adaptation all the more relevant. We argue, however, that two other factors – the “strategic” nature of software development and the increased heterogeneity among software development's stakeholders – interact with technical complexity to make the context of software development more uncertain than ever before. And it is this interaction and its resulting complexity, it stands to reason, that has stimulated the field's rush to adopt the ADM paradigm.

2.2.1 Strategic Nature of Software

The nature of problems addressed by software developers has changed significantly over the past few decades. Early software projects were geared towards increasing efficiencies by automating routine, structured tasks. Examples of these include transaction-processing systems (TPS) and early Management Information Systems (MIS). While the former focused on the collection and storage of data produced by business transactions, the latter used this internal data to generate a variety of reports (e.g., trends, exceptions, summary reports) to facilitate monitoring and control (e.g., Laudon and Laudon 2009). Stakeholders for these projects were few and of a shared mindset, and extensive involvement with them was neither expected nor necessary. Over the years, however, the evolution of development methodologies led to developers spending comparatively less of their time on such projects and more of their time on projects of a more strategic nature.

An Information System (IS) may be viewed as a sociotechnical system in which people, technology, business processes and the organization interact to gather, process, archive, and distribute information for the purposes of control, coordination, and decision making (e.g., Laudon and Laudon 2009, Piccoli 2008; O'Hara et al. 1999). Information Technology (IT) (i.e., hardware, software, telecommunications), a critical component of IS, plays a pivotal role in transforming businesses. O'Hara et al. (1999) provide an excellent conceptualization of three orders of change engendered by the introduction of new IT. Each level of change affects technology, business process/task, people, and organizational structure to varying degrees. First-order change, reminiscent of early IT endeavors, involved the automation of routine tasks, with technology and process being the primary concerns. Second-order change, referred to as “informate”, affected the work habits and established roles of people, thus emphasizing the social dimension of

change management as well as the need to understand the complex interaction among people, technology, and process. These systems did not merely automate, but provided valuable, timely and accurate information to recognize and leverage opportunities in the market place. Third-order change, called “transform”, has far-reaching implications for the organization, not just affecting people, technology, and process, but also the structure of the organization. Clearly, the higher the order of change entailed by IS, the greater its impact on organizational effectiveness and competitiveness, as well as on the strategic orientations of the firm. On the flip side, such high-order changes are fraught with uncertainties and are extraordinarily difficult to execute and manage.

What makes today’s projects more “strategic”? Theorists across a variety of disciplines emphasize three critical distinctions between efforts that are “strategic” and those that are not. First, strategic projects tend to be more “unstructured,” with no clear means-ends map to guide decision-making (Mintzberg et al. 1976). Second, as projects become more strategic they become more cross-functional, focused on solutions to challenges that cut across traditional functional boundaries such as marketing or operations (Day and Wensley 1983). This, combined with an unstructured setting, makes a “strategic” setting inherently more uncertain and complex. Finally, greater functional breadth raises the likelihood that multiple approaches to a desired end state are available (Galunic and Eisenhardt 1994); a strategic project likely has no “one best solution”.

As indicated earlier, in the last few years, development project emphasis has shifted to more unstructured domains. Developers are asked to develop tools that assist decision-making under uncertainty or generate business intelligence for the creation and sustenance of competitive advantage. The scope of projects has broadened from narrow and well-defined domains (e.g., payroll processing) to systems that affect many or even all aspects of an organization, both on an intra-enterprise (e.g., ERP) and extra-enterprise (e.g., virtual integration, supply-chain) basis. These are inherently “strategic” contexts, and they increase the uncertainty and complexity of the software development setting.

Further, software developers must now interact with a broader, more diverse set of stakeholders than ever before. These stakeholders may not be even in rough agreement as to the goals of the software, and their inputs into the software development process may be conflicting. Highly valued systems may not be optimal in a technical sense; rather, they may be attractive simply because they minimize the conflict that results from stakeholders having different wants and needs.

It seems clear, then, that mere technical prowess is no longer sufficient to ensure success in today’s software development environment. Further, organizational ideals of democratic workplaces that welcome a variety of viewpoints and encourage continuous learning increase the challenge of structuring software development efforts. Ultimately, today’s software developers must balance the often disparate needs of diverse stakeholders, a task far more challenging than merely fulfilling the functional requirements of a system. Thus, it is no surprise that the

agile paradigm of development – with its embracing of these emerging realities – is more than welcomed by the field.

2.3 What's New About Agile Development?

A recent article (Nerur and Balijepally 2007) on the theoretical roots of ADM suggests that its principles (Table 2.1) are consistent with problem-framing and problem-solving approaches that have evolved in disciplines such as architecture and strategic management. In this section we reiterate this view in asserting that the ideas and concepts behind the ADM movement have been around for decades. It is possible, we believe, that the field of computing in general – and software practitioners in particular – could have arrived at this level of maturity earlier had there been more openness to intellectual developments in allied fields such as Information Systems. Table 2.1 is by no means exhaustive. It does not, for example, include Iivari's PICO Model (Iivari and Koskela 1987), Boehm's Spiral Model (Boehm 1986), or the "collective resource approach" of Ehn and Kyng (1987).

ADM emphasizes the importance of social interaction in software development, stressing the importance of greater autonomy and decision-making discretion for developers. Within the ADM paradigm development is seen not as a means-end process, but rather as ongoing cooperation focused on, among others, the facilitation of learning, flexibility, communication and redundancy of functions. Many of these characteristics were seen in the theoretical schools of systems development that evolved in Europe – particularly in Scandinavia – where development practitioners were challenged early on with the complexity that results from interactions with a wide variety of interests and perspectives (Bansler 1989).

There is also substantial congruence between the field's move toward ADM and the strategic management field's movement from an emphasis on "linear" approaches toward more "adaptive" or "interpretive" understandings (Chaffee 1985). The linear view emphasizes rationality and logic, focusing on purposive planning and action toward clearly defined ends. In contrast, the adaptive view recognizes that an organization, much like an organism, must continually reconfigure and align itself to an ever changing environment (for example, see Morgan 2006). Finally, the interpretive view incorporates a social reality into strategy-making: The strategy chosen likely will reflect the competing worldviews of those who have a stake either in its formulation, implementation or results. Habermas (1984) frames these approaches similarly, pointing to the limitations of social processes grounded in instrumental reasoning. He suggests solutions that incorporate shared understanding and purpose as well as equitable and unconstrained participation of all stakeholders.

Table 2.1 Roots of Agile Principles

Agile Principle	Examples of Previous Developments
Emphasis on individuals	Sociotechnical systems [1976; as early as the 1950s – e.g., Trist(1981)] Soft Systems Methodology [Checkland 1979, 1981, 1988]
Emphasis on job and work design, quality of life, and accomplishing the work	Sociotechnical design [e.g., Cherns 1976; 1987; Olerup 1989]
Learning and adaptation – iterative development	Soft Systems Methodology (Checkland 1981) Interactive Planning (Ackoff 1974) Vickers’s Appreciative Systems (1968)
Participative development	Strategic Assumption Surfacing and Testing (SAST) (Mason and Mitroff 1981) Churchman’s Systems Approach (1968), Designing participatively (Mumford 1979)
Accepting and leveraging change	Soft Systems Methodology (1981); Interactive Planning (1974)
Self-organization	Complex Adaptive Systems, complexity theory (1970s)
Minimum Critical Specifications	Sociotechnical system design (e.g., Cherns 1976) Holographic principles (Morgan and Ramirez 1983)
Reflection in action (e.g., reflection work-shops)	Schon (1983) Collaborative Action Learning (Ngwenyama 1993) Reflective Systems Development (Mathiassen 1998)

Based on the preceding sections, one could easily argue that software development parallels patterns exhibited in a variety of disciplines – from architecture to product design to strategic management. The similarity of software development’s trajectory to that of other disciplines can be taken as evidence in support of Jantsch’s (1975) assertions that the dynamic process of knowledge acquisition and use remains relatively constant across a variety of diverse domains, particularly when these domains encompass such complexity as is present in social systems. Figure 2.1, based on Jantsch’s experience with diverse systems, highlights that inventive rather than purely analytical solutions are necessary to solve problems faced by purposeful human activity systems.

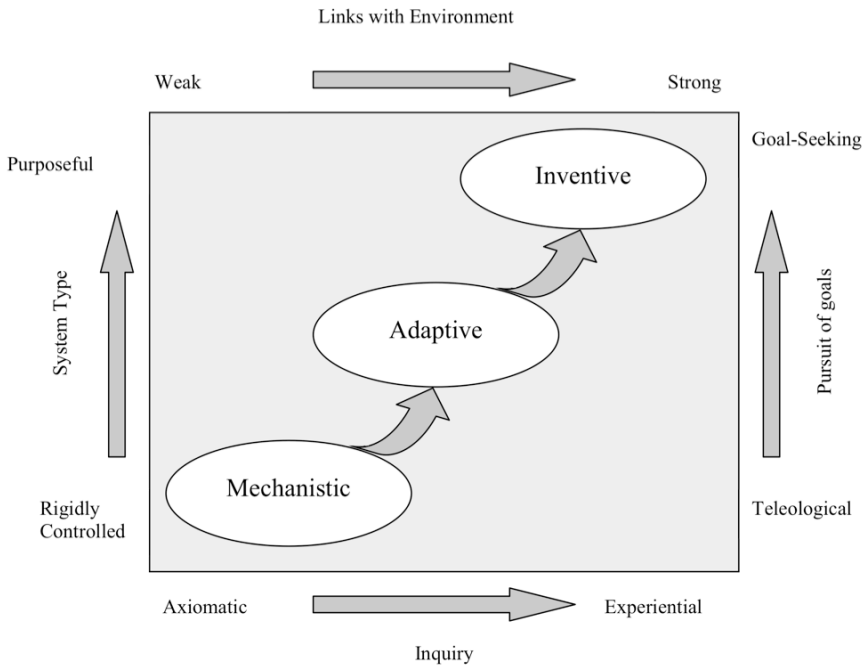


Fig. 2.1 Changing Dynamics of Systems. Adapted from: Jantsch (1975)

2.4 Principles of Sociotechnical Systems

ADM's principles and assertions also match up well with arguments made by proponents of Sociotechnical Systems (STS) theory. STS theory emphasizes (Cherns 1976; Cherns 1987; Olerup 1989):

Compatibility: Recognizing that design activity is characterized by conflict, Cherns (1987, pg. 154) advocates an approach to design that reconciles the means employed with the design objectives they aim to fulfill. This principle emphasizes a transparent design process that facilitates and encourages participation of all concerned. Concurrence of views is sought in an open organizational climate that is devoid of domination of any sort. Team members must have the latitude to evolve their own ways of solving problems in accordance with the requirements of the design. This is in stark contrast to constraining the actions of designers through rigid processes and rules. Design choices should address technical and social concerns at once rather than giving primacy to the former.

Minimum Critical Specifications: Removing bureaucratic restraints while providing minimal guidelines so that problem-solvers can exercise their ingenuity in solving problems. This principle stresses that only what is absolutely required

should be specified. This cautions against overly specifying what tasks need to be done, how they should be done, what roles designers should play in solving the problem, and so forth. Restricting the specifications to an essential minimum not only allows the designers to self-organize and improvise as the solution unfolds, but also gives them greater flexibility in exercising their design choices.

Variance Control: This was originally referred to as “the sociotechnical criterion” (Cherns 1976). Variance refers to any unanticipated occurrence (e.g., a software defect or a design flaw) that might have an adverse effect on outcomes. This principle suggests that such aberrations in quality or design flaws should be addressed as and when they arise (i.e., in close proximity to the cause of the problem). For example, a software design flaw that manifests itself when the final software product is being tested would be more expensive to fix and would require unnecessary coordination and communication between the testers and the designers. However, if the designers (i.e., software developers) had formally assessed their design prior to its implementation, they would have perhaps caught the error and gained additional design knowledge in the process. Further, when the designers detect and correct the error, they are probably accounting for the source or cause of the variance, which is unlikely to occur when the testers downstream find a problem with the software product.

Boundary Location: Encouraging and facilitating ample interaction between and among problem-solvers working on related activities, thereby enabling designers to have free interchange of ideas and information.

Information Flow: Ensuring that problem-solvers have easy and timely access to relevant information/knowledge.

Power and Authority: Designers are provided with resources needed to carry out their tasks. Further, they are given the power and authority to handle their resources. The designers, in turn, assume accountability for their performance and for the responsible use of the resources in their trust.

Multifunctional Perspectives: Developing and nurturing versatile personnel who can incorporate the views of a wider variety of stakeholders, thus enabling problem-solvers to accomplish more with less supervision (Cavaleri and Obloj 1993). Indeed, the goal of having self-organizing software teams that can expeditiously address unanticipated development problems can be realized only if members of such teams are multifaceted and equipped with a broader range of technical and leadership skills. This perspective is akin to the concept of *redundancy of functions* embodied in the holographic principles of management (Morgan 2006; Nerur and Balijepally 2007).

Support Congruence: This principle focuses on providing: a) a “constraint-free design mode” to the extent possible in order to reconcile the tension between control and influence that a team can exercise in its interactions with other related units; and b) a system that rewards and nurtures designers based on their awareness and grasp of the processes in which they are engaged.

Transitional Organization: This principle addresses the myriad challenges that face an organization when it is either starting up or being redesigned. The design

team and the process that it employs should not only reflect the new values, but must also disseminate these values to the rest of the organization. Cherns views the transitioning process as an opportunity for learning. Thus, the design team plays a pivotal role in managing and facilitating the change. For example, the extent to which the entire software development team learns, imbibes, and values agile principles would largely determine how successful an organization is in instituting process changes to accommodate agile practices.

Incompletion: Committing to ongoing problem-solving (i.e., planning, doing, reflecting, and acting) to encourage continuous refinement of solutions.

Some of the principles of STS outlined above could inform ADM practices. While ADM principles allude, at least in theory, to many of the STS tenets, their praxis in software development is somewhat limited and a more concerted effort may be required to reify these concepts. For example, agile methods encourage an open and collaborative development environment in which developers have the latitude to make decisions regarding the assignment of individual roles and responsibilities as well as the choice of problem-solving strategy. A closely related concept is the idea of self-organization where members of a software team organize themselves in response to emergent problems (Cockburn and Highsmith 2001). Implicit in this arrangement are the notions of self-management and shared leadership, both of which have considerable value in team-oriented knowledge work (Pearce and Manz 2005). However, these concepts can be implemented only if software developers expand their capabilities (e.g., better understanding of architectural and quality issues) and focus more on adding value to the team's primary objectives rather than to their job functions alone. Further, software developers need to expand their intellectual perspectives to appreciate the relevance of shared understanding, mutual trust, conflict resolution, and self-management leadership behaviors (e.g., self-reinforcement, self-goal-setting, self-evaluation, self-expectation, etc.) to self-organizing collaborative development (for example, see Manz and Sims 1987 and Pearce and Manz 2005).

It is apparent that STS principles such as *minimum critical specification* and *multifunctional perspectives* have the potential to influence ADM practices. Likewise, the rapidly increasing experience with ADM could be invaluable in extending and/or refining the design ideals of STS.

2.5 ADM and the System of Systems Methodologies (SOSM)

In his book *Systems Thinking: Creative Holism for Managers*, Jackson (2003) articulates the System of System Methodologies (SOSM) framework, which gives a broad classification of problem-solving approaches (akin to Weber's (1969) "ideal types"). The grouping of methods is based on the nature of the system (simple to complex) and the extent of convergence/divergence of the worldview and interests of the participants (a continuum from unitary to pluralistic to coercive). Thus, we

have six types – simple/unitary, simple/pluralistic, simple/coercive, complex/unitary, complex/pluralistic and complex/coercive – ranging from very structured problem-solving domains (e.g., payroll processing) to very complex unstructured ones (e.g., addressing global warming). Jackson (2003) subsequently links these approaches to the sociological paradigms of Burrell and Morgan (1979). Our aim in this section is to briefly explore how ADM's ideals relate to the SOSM.

Simple systems in which the participants (such as designers and stakeholders) have compatible worldviews and design goals present problems that are easily solved by hard systems thinking, exemplified by approaches such as Systems Analysis, Systems Engineering, and Operation Research. Efficiency, rather than effectiveness, is the primary aim of such methods. The extreme case involves complex problems involving stakeholders and decision-makers whose values, goals and interests are irreconcilable. Solutions to such situations cannot be evolved through an instrumental rational process, but are often coerced by other means, such as politics, power plays and domination by a majority. Postmodern thinking that strives for achieving diversity by bringing to the fore ignored or suppressed viewpoints is deemed to be influential in tackling such problems.

So, where does ADM figure in the SOSM framework? To answer this, we rely on Jackson's positioning of Soft Systems Methodology (SSM), an approach with which ADM shares similarities, in this framework. SSM involves multiple diverse stakeholders, emphasizes learning, and uses an iterative approach to gain a progressively greater appreciation of the system under development. Further, it may be applied to simple or complex systems. Therefore, it occupies the entire vertical column of the SOSM that corresponds to "Pluralist". ADM emphasizes: a) people and their collaborations; b) iterative development that actively engages customers; c) adaptive planning; d) flexibility of roles; e) self-organizing teams; f) a leadership style that focuses on facilitating and mentoring, rather than on controlling; g) active communication and feedback; h) the need to accept and leverage change; and i) continual reflection on practices. Thus, it leans towards the SOSM location in which Jackson places SSM.

While it would be illuminating to provide an analysis of ADM's assumptions from the perspective of Burrell and Morgan's (1979) sociological paradigms, such an exposition is beyond the scope of this paper. Traditional systems analysis and design approaches (particularly those that are heavily plan- and specification-driven) presume an objective world that is predictable and stable. Such methods further anticipate a specific end and therefore focus on arriving at that end in the most efficient manner as possible. The specifications, as well as the design models and diagrams derived from them, are believed to be objective and consistent with an independent reality in which processes can be directly observed, measured and verified (e.g., Hirschheim and Klein 1989). Thus, traditional approaches are largely functionalist (see Nerur and Balijepally 2007). As mentioned earlier, ADM recognizes that change is inevitable and advocates a "sense-making process" (Berger and Luckmann 1967 as cited in Hirschheim and Klein 1989) that involves adaptive planning, frequent iterations informed by feedback from stakeholders,

and reflective learning. Thus, it leans towards “social relativism” (Hirschheim and Klein 1989), or what Burrell and Morgan (1979) refer to as an “interpretive” sociological paradigm. The following observation by Hirschheim and Klein (1989, pg. 1205) seems to bear out this assessment:

“The mechanism of prototyping or evolutionary learning from interaction with partial implementations is the way technology becomes embedded in the social perception and sense-making process.”

2.6 Implications for Research and Practice

As information systems development has become more strategic, IT decision-makers and software development leaders have found themselves confronted with ever-more-diverse bases of constituents. This growth in the range or heterogeneity of viewpoints that must be considered increases the complexity of software development efforts independent of the technical challenges that are specific to particular development tasks. Development professionals confronted with such complexity find their task environments increasingly uncertain across multiple dimensions, and in response have seized on the agile paradigm to help overcome this uncertainty.

That this recently “emerged” paradigm can be interpreted as a re-combination of earlier viewpoints is not altogether surprising. Disciplines or sub-disciplines often go through protracted periods of incremental development, in which progress is both orderly and predictable. During such eras, broad agreement exists among discipline members regarding the challenges that should be addressed and the methods that should be used to address them, and the steady progression of knowledge serves to perpetuate this consensus. In many fields, however, there comes a time when the mass of unsolved puzzles or unexplained anomalies reaches some critical point. Then, boundaries that restrict problem-selection and problem-solving to narrow paths are cast aside, ushering in radically new approaches or mindsets.

While ADM has been proffered, in one form or another, for years if not decades, what has made this particular point in time ripe for a Kuhnian revolution with regard to development? We argue that a variety of facets of contemporary IT challenges underscore, more than ever before, the practical value of agility in software development. Further, the nature of software development in a post-industrial global economy is challenging development professionals to take on more roles – or to at least interact with a wider variety of constituents from other intellectual or professional spheres – than they ever have.

These developments have implications for both practice and research in software development. For development professionals, it is becoming increasingly important that their technical skill sets be augmented with social and political tools appropriate for interacting with a wide variety of stakeholders. The ability to re-

spond to and incorporate conflicting viewpoints will come to more readily characterize successful software development professionals.

Professionals also are being challenged to revisit their understanding of how much up-front planning is appropriate. Boehm (2002; also see Boehm and Turner 2004) argues that there is a delicate balance – what he refers to as the “sweet spot” – between being governed by up-front plans and being open to change as the result of new or conflicting information. Clearly, developers schooled in a rational approach will find themselves challenged by the need to find and maintain this balance, particularly as the potential sources of imbalance grow in number or diversity.

Scholars hoping to make sense of and contribute to contemporary software development also must be prepared for changes. The agile paradigm emphasizes the importance of intangibles such as personalities, personal development and reflection. Development efforts that are consistent with the agile paradigm might not conform neatly to “normal science” approaches, raising the risk that what scholars report or suggest is inaccurate or inappropriate. Researchers exploring development must themselves go “back to school” and look to theories or methods that are more consistent with the emerging environment.

2.7 Conclusion

Changes in the milieu of software development are demanding and will continue to challenge professionals for the foreseeable future. In this paper, we have attempted to point out the conceptual underpinnings of ADM. To this end, we outlined the principles of sociotechnical systems, mainly because of their resemblance to some aspects of ADM. There is definitely a potential here to extend this work and carry out a detailed examination of the similarities and differences between STS and ADM, along the lines of the comparative analysis of STS and Lean Production by Niepce and Molleman (1998). In addition, we used Jackson’s framework to further our understanding of the assumptions of ADM. One could easily extend the analysis to understand where ADM stands with regard to its ontological and epistemological orientations. The Burrell and Morgan (1979) framework, albeit controversial because of its adherence to Kuhn’s notions of paradigms and incommensurability (Deetz 1996), would still be a good starting point for such an analysis.

Not unlike other research studies, our paper has some limitations. First, our social/cultural/technical conditioning likely has influenced our analysis. Second, although there are many intellectual streams from which we could have analyzed the conceptual bases of ADM, we restricted our discussions to but a few. For example, it may be argued that an examination of ADM from some other perspective, say, Lean Production, Complex Evolving Systems (CES), Autopoiesis or Churchman’s Inquiring Systems, would have been more fruitful. Despite these

shortcomings, we hope our work offers early guidance to both scholars and practitioners alike as they continually seek solutions to contemporary development problems.

References

- Ackoff, R.L. (1974). *Redesigning the Future*. New York: John Wiley & Sons.
- Bansler, J. (1989). System Development Research, Scandinavia. *Scandinavian Journal of Information Systems*, 1, 3-20.
- Berger, P., Luckmann, T. (1967). *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. New York: Doubleday.
- Bertalanffy, von L. (1950). An Outline of General Systems Theory. *British Journal of the Philosophy of Science*, 1, 134-165.
- Boehm, B. (1986). A spiral model of software development and enhancement. *ACM Sigsoft Software Engineering Notes*, 11(4), 14-24.
- Boehm, B. (2002). Get Ready for Agile methods, with Care. *Computer*, 35(1), 64-69.
- Boehm and Turner (2004). *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley.
- Cavaleri, S. and Obloj, K. (1993). *Management Systems – A Global Perspective*. Belmont, CA: Wadsworth Publishing Company.
- Chaffee, E.E. (1985). Three Models of Strategy. *Academy of Management Review*, 10(1), 89-98.
- Checkland, P.B. (1979). Techniques in 'Soft' Systems Practice Part 1: Systems Diagrams – Some Tentative Guidelines. *Journal of Applied Systems Analysis*, Vol. 6, 33-40.
- Checkland, P. (1981). *Systems Thinking, Systems Practice*. Chichester: John Wiley.
- Checkland, P.B. (1988). Soft Systems Methodology: An Overview. *Journal of Applied Systems Analysis*, 15, 27-30.
- Cherns, A. (1976). The Principles of Socio-Technical Systems Design. *Human Relations*, 29(8), 783-792.
- Cherns, A. (1987). Principles of Socio-Technical Design Revisited. *Human Relations*, Volume 40, Number 3, 153-162.
- Churchman, C.W. (1968). *The Systems Approach*. New York: Dell.
- Cockburn, A., & Highsmith, J. (2001). Agile Software Development: The People Factor. *Computer*, 34(11), 131-133.
- Day, G.S., & Wensley, R. (1983). Marketing Theory with a Strategic Orientation. *Journal of Marketing*, 47(4), 79-89.
- Deetz, S. (1996). Describing Differences in Approaches to Organization Science: Rethinking Burrell and Morgan and Their Legacy. *Organization Science*, 7(2), 191-207.
- Ehn, P. and Kyng, M. (1987). The Collective Resource Approach to Systems Design. In G. Bjerknes, P. Ehn, and M. Kyng (eds.), *Computers and Democracy: A Scandinavian Challenge* (pp. 17-57), Aldershot, United Kingdom: Avebury.
- Galunic, D.C., & Eisenhardt, K. M. (1994). Renewing the Strategy-Structure-Performance Paradigm. In L.L. Cummings & B.M. Staw (Eds.), *Research in Organizational Behavior Vol. 16* (pp. 215-255), Greenwich, CT: JAI Press.
- Habermas, J. (1984). *The Theory of Communicative Action*. Boston, MA: Beacon Press.
- Highsmith, J. (2002). *Agile Software Development Ecosystems*. Boston, MA: Addison-Wesley.
- Hirschheim, R. and Klein, H.K. (1989). Four Paradigms of Information Systems Development. *Communications of the ACM*, 32(10), 1199-1216.
- Iivari, J. and Koskela, E. (1987). The PICO Model for IS Design. *MIS Quarterly*, 11(3), 401-419.

- Jackson, M.C. (2003). *Systems Thinking: Creative Holism for Managers*, Chichester: England, John Wiley & Sons, Ltd.
- Jantsch, E. (1975). *Design for Evolution*. New York, NY: George Braziller, Inc.
- Larman, C. (2004). *Agile & Iterative Development: A Manager's Guide*. Boston, MA: Addison-Wesley.
- Laudon, K.C. and Laudon, J.P. (2009). *Essentials of Management Information Systems* (Eighth Edition). Upper Saddle River, NJ: Pearson Prentice Hall.
- Manz, C.C. and Sims, H.P. (1987). Leading Workers to Lead Themselves: The External Leadership of Self-Managing Work Teams. *Administrative Science Quarterly*, 32, 106-128.
- Mason, R.O., & Mitroff, I.I. (1981). *Challenging Strategic Planning Assumptions : Theory, Cases, and Techniques*. New York: Wiley.
- Mathiassen, L. (1998). Reflective Systems Development. *Scandinavian Journal of Information Systems*, 10 (1 & 2), 67-118.
- Mintzberg, H., Raisinghani, D., & Theoret, A. (1976). The Structure of "Unstructured" Decision Processes. *Administrative Science Quarterly*, 21(2), 246-275.
- Morgan, G., & Ramirez, R. (1983). Action Learning: A Holographic Metaphor for Guiding Social Change. *Human Relations*, 37(1), 1-28.
- Morgan, G. (2006). *Images of Organization*. Thousand Oaks, CA: Sage Publications.
- Mumford, E., & Henshall, D. (1979). *Participative Approach to Computer Systems Design : A Case Study of the Introduction of a New Computer System*. London: Associated Business Press.
- Nerur, S., & Balijepally, V. (2007). Theoretical Reflections on Agile Development Methodologies. *Communications of the ACM*, 50(3), 79-83.
- Ngwenyama, O.K. (1993). Developing End-Users' Systems Development Competence. *Information & Management*, 25, 291-302.
- Niepc, W. and Molleman, E. (1998). Work Design Issues in Lean Production from a Sociotechnical Systems Perspective: Neo-Taylorism or the Next Step in Sociotechnical Design? *Human Relations*, 51(3), 259-287.
- O'Hara, M.T., Watson, R.T., & Kavan, B.C. (1999). Managing the Three Levels of Change. *Information Systems Management Journal*, 16(3), 63-70.
- Olerup, A. (1989). Socio-Technical Design of Computer-Assisted Work: A Discussion of the ETHICS and Tavistock Approaches. *Scandinavian Journal of Information Systems*, 1, 43-71.
- Pearce, C.L. and Manz, C.C. (2005). The New Silver Bullets of Leadership: The Importance of Self- and Shared Leadership in Knowledge Work. *Organizational Dynamics*, 34(2), 130-140.
- Piccoli, G. (2008). *Information Systems for Managers*. John Wiley & Sons, Inc.
- Poppendieck, M. (2002). Wicked Projects. *Software Development Magazine* [also available at <http://www.poppendieck.com/wicked.htm> - accessed on 11/9/2008]
- Schon, D. (1983). *The Reflective Practitioner: How Professionals Think in Action*. New York, NY: Basic Books.
- Trist, E.L. (1981). The Sociotechnical Perspective. The Evolution of Sociotechnical Systems as a Conceptual Framework and as an Action Research Program. In Van de Ven, A.H., & Joyce, W.F., (Eds.), *Perspectives in Organization Design and Behavior*, Wiley.
- Vickers, G. (1968). Science and the Appreciative System. *Human Relations*, 21(2), 99-119.
- Weber, M. (1969). *The Methodology of the Social Sciences*. New York, NY: Free Press.

Author Biographies

Sridhar Nerur is an associate professor of Information Systems at the University of Texas at Arlington. He holds an Engineering degree in Electronics from Bangalore University, a PGDM (MBA) from the Indian Institute of Management, Bangalore, India, and a Ph.D. in Business Administration from the University of Texas at Arlington. His publications include articles in leading journals such as *MIS Quarterly*, *the Strategic Management Journal*, *Communications of the ACM*, *Communications of the AIS*, *the DATA BASE for Advances in Information Systems*, *European Journal of Information Systems*, and *Information Systems Management*. He served as an Associate Editor of the *European Journal of Information Systems*. His research and teaching interests are in the areas of software design, adoption of software development methodologies, cognitive aspects of programming, dynamic IT capabilities, and agile software development.

Alan Cannon is an associate professor of operations management and statistics at the University of Texas at Arlington. His research on operations strategy and research methods has appeared in such journals as the *Journal of Operations Management*, *Organizational Research Methods*, *the International Journal of Production and Operations Management* and the *International Journal of Production Research*. His current research interests include capacity investment, service operations, performance measurement and hierarchical linear modeling.

VenuGopal Balijepally is an assistant professor of MIS in the College of Business at Prairie View A&M University, Texas. He received his Ph.D. in Information Systems from the University of Texas at Arlington and Post Graduate Diploma in Management (MBA), from the Management Development Institute, Gurgaon, India. His research interests include software development, social capital of IS teams, knowledge management and IT management. His research publications appear in *MIS Quarterly*, *Communications of the ACM*, *Communications of the AIS*, and various conference proceedings such as the Americas Conference on Information Systems, the Hawaii International Conference on System Sciences, and the Decision Sciences Institute.

Philip L. Bond holds the BA and MA in Economics from the University of Rhode Island, 1991 and 1993. He received the MS in Information Systems from the University of Texas at Arlington in 2009, and is a doctoral student at the University of Texas at Arlington. He has been an Instructor, Program Chair and Dean at ITT Technical Institute, and a Dean at Everest College.



<http://www.springer.com/978-3-642-12574-4>

Agile Software Development
Current Research and Future Directions
Dingsøyr, T.; Dybå, T.; Moe, N.B. (Eds.)
2010, XVII, 238 p., Hardcover
ISBN: 978-3-642-12574-4