Theoretical Reflections on AGILE DEVELOPMENT METHODOLOGIES

The traditional goal of optimization and control is making way for learning and innovation.

By Sridhar Nerur and VenuGopal Balijepally

If conflict and argumentation are indeed signs of progress, the field of software development must be advancing by leaps and bounds. In the past decade, there has been much debate among software developers over contradictory perspectives of the traditional plan-driven approaches vs. the agile philosophy of software development. Conspicuously absent, however, are the rich perspectives on similar shifts in patterns of thought in other disciplines and the articulation of underlying conceptual assumptions built into today's software designs. Here, we identify theoretical and conceptual support for the emerging epistemology—methods of searching for and validating basic knowledge—of software practice epitomized in the principles of agile methodologies. Such support is important [4]. Foremost, it enriches our understanding of the agile phenomenon by examining its theoretical roots. It also enables informed use of agile practices in appropriate situations; helps discover reasons for the success of software practice in complex social situations; and facilitates dissemination of agile concepts.

Software development is a complex undertaking beset with many problems, called "wicked problems" by Horst Rittel, an urban planner who pioneered the concept of issue-based information systems to facilitate the formulation and clarification of complex administrative decisions [2]. These problems tend to be unique and difficult to formulate, and solutions evolve continually as the designer gains a greater appreciation of what must be solved. The need to satisfy multiple conflicting viewpoints makes it difficult to devise a test to determine the effectiveness of solutions. "Argumentation," according to [2], is critical in solving such problems. It would appear that modes of inquiry in software development are beyond technical rationality based on observation and facts, or logical positivism [11]. Emerging practices (such as agile development) question the assumption that change and uncertainty can be controlled through a high degree of formalization. Proponents of agile methods have discovered inadequacies in formal design that follows systematic procedures dictated by rigid processes. These insights have produced a more incisive method of inquiry that departs from traditional approaches to software development. This conceptual shift and its resulting debates are hardly exceptional; similar evolutionary shifts in thought and methods of inquiry are also found in architecture and strategic management.

Our discussion here is based on the development of design concepts in architecture, as outlined in [2]. Briefly, Chris Alexander, a professor of architecture at the University of California, Berkeley, pioneered in the 1970s work on patterns in architectural design that inspired software design patterns, a conceptual breakthrough that revolutionized the way software is built. Design patterns are proven design experiences and best practices for solving similar problems across domains. Alexander's early model conceived design as an instrumental process dictated by technical rationality [2]. It aimed to identify the optimal means to a predictable end by employing scientific methods used in problem solving in the engineering disciplines [11]. This view reduced design to a linear sequence of well-defined steps, with analysis preceding synthesis [2]. Systematic problem decomposition is the key activity in analysis, a process reminiscent of hierarchical top-down functional decomposition that dominated thinking in structured software design in the years before object orientation found favor among developers.

As early as 1963, continuous feedback among phases, communication, and iterative cycles of analysis and synthesis were recognized as key aspects of design [2]. Inspired by work on wicked problems, argumentation involving stakeholders became an essential part of the design process. Interestingly, Alexander abandoned his early model in favor of a design process emphasizing "interaction between the source of knowledge and experience, and the decision maker" [2]. The idea of patterns is central to Alexander's subsequent way of approaching design.

While Rittel and Alexander both stressed the need to integrate multiple viewpoints, their modes did not explicitly include the concept of learning. In contrast, [2] viewed design as a learning process in which the solution evolves through repetitive cycles of problem formulation, solution evaluation, and documentation. Designers use their own personal expanding knowledge to continually reframe the problem and devise an appropriate solution. In the spirit of Donald Schon's reflection-in-action, the process involves repeated modifications of the practitioner's design through insights and knowledge gained from active interaction with the problem situation [11]. The improvisations and learning resulting from this dynamic interplay ("conversations," as Schon called it in [11]) between the designer and the problem lead to a succession of updated problem representations.

The development of problem-solving ideas in architecture reveals several insights into the nature of design:

Complexity. Problems are recognized as wicked, or not easily tractable;

Emergence. Design is conceived as a repetitive emergent process interlacing argumentation, introspective reflection, learning, and continual problem and solution refinement; and

Integrative. Integration of multiple perspectives is critical to problem resolution.

Strategic Thinking

The shift from a mechanistic perspective to a perspective that acknowledges the existence of environmental uncertainty and complexity is evident in today's strategic management thinking in organizational contexts. Early on, the dominant paradigm for strategy formulation was a formal process to identify the best fit between an organization and its environment [6]. This approach is predicated on the assumption of a foreseeable and unchanging world, which, of course, it never is. Moreover, this approach sought the best means to achieve a predictable end thorough analysis and reasoning, concordant with functional/technical rationality. According to Henry Mintzberg, a theorist in management and organizational research, the design, planning, and positioning schools of thought in strategic management reflect this view and tend to be highly prescriptive [7].

Consistent with early approaches to architectural design, strategy formulation through logical thinking and planning preceded implementation of strategic initiatives. Moreover, strategic management was the exclusive province of top management, with little or no involvement of the people who actually implement the related strategies. While these planned-approach schools historically exploited past experience, they lacked focus on learning through exploration and were inflexible and woefully ill-suited for agile response to rapid environmental change [6, 7].

The conceptual appeal and practicality of incremental learning in turbulent and complex organizational environments has led many strategists to reposition formulation closer to implementation [6]. In this view, strategy formulation (as an emergent process) is constantly influenced by learning, which occurs during implementation. Mintzberg, an early advocate of this approach, conceptualizes strategy formulation as a craft, likening strategists to craftspeople whose minds work in concert with their hands as they learn and improvise in shaping an artifact [7]. Effective strategies evolve through a learning process involving skills, experience, and insights gained through the dynamic interplay among formulation, implementation, and critical reflection. Thus, strategy is an emergent process in which strategy formulation and implementation are inseparable [7].

Mintzberg's "learning school" reflects this view of strategy making. In it, the strategist is like Schon's "reflective practitioners," learning from the context, "conversing" with it, and being influenced by their reflections on the actions taken in the situation [11].

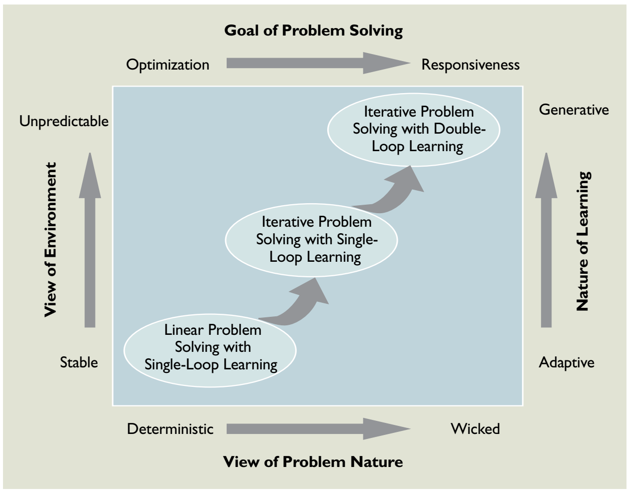

Figure 1. Evolutionary shifts in design thinking.

New Design Metaphor

Much can be learned about design efficacy by examining the patterns of inquiry in various disciplines. The progression of thought in software development parallels the maturation of design ideas in architecture and strategic management. The traditional mechanistic worldview is today being challenged by a newer agile perspective that accords primacy to uniqueness, ambiguity, complexity, and change, as opposed to prediction, verifiability, and control. The goal of optimization is being replaced by flexibility and responsiveness.

The new design metaphor incorporates learning and acknowledges the connectedness of knowing and doing (thought and action), the interwoven nature of means and ends, and the need to reconcile multiple worldviews. This approach is "substantially rational," facilitating learning and adaptation through the constant questioning of assumptions [9]. Constantly reexamining assumptions and evaluating discrepancies against existing norms and values fosters double-loop learning [1], which is "generative" by increasing both learning and the ability to innovate and use change to one's advantage [12]. In contrast, the traditional school of design is more in the spirit of single-loop learning [1], or "adaptive learning" [12], in which routine problem-solving parameters and preset rules, norms, and contrasts determine goal-seeking behavior [1, 8, 12].

The theoretical assumptions of the emerging practice of design—intertwining thought and action, critical reflection and learning after action, practicality, and satisfying human needs through continuing inquiry—may be traced to Action Learning Theory, Dewey's pragmatism, and phenomenology [8, 9]. A deterministic and rational approach based on law-like, scientific methods of inquiry is said to be "functionally rational" with roots in logical positivism—reasoning based on observation and fact—and scientific methods of inquiry [9, 11]. The table here outlines the distinctions between these two views of design; Figure 1 captures the shifts in design thinking in architecture, strategic management, and software development.

Several characteristics of "design thinking," according to [5], may be gleaned from the conceptually opposing strands of design in the table. Design thinking, according to [5], should be "synthetic" (integrating multiple worldviews), "adductive" (imagining and preparing for a preferred future state), "hypothesis-driven" (conceptually enacting "what-if" and "if-then" scenarios), "opportunistic" (alerting to evolving prospects), "dialectical" (resolving conflicting views through argumentation), and "inquiring and value-based" (questioning assumptions with openness while being sensitive to the values of all stakeholders).

In today's world of increasing social complexity software engineers need a broader interpretation of the metaphor of design than is generally accepted.

New Conceptualization of Software Development

The trend in management thinking, moving from a deterministic/mechanistic view of problem solving to a dynamic process, characterized by iterative cycles and the active involvement of all stakeholders, is reflected in software development as well. The "emergent metaphor of design" in the table is manifest in the agile methods in today's emerging software-development thinking. Agile methods are people-centric, recognizing the value competent people and their relationships bring to software development. In addition, it focuses on providing high customer satisfaction through three principles: quick delivery of quality software; active participation of concerned stakeholders; and creating and

	Traditional View of Design	Emergent Metaphor of Design
Design process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness
Problem-solving approach	Selection of best means to accomplish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution
View of the environment	Stable, predictable	Turbulent, difficult to predict
Type of learning	Single-loop/adaptive	Double-loop/generative
Key characteristics	Control and direction Avoids conflict Formalizes innovation Manager is controller Design precedes implementation	Collaboration and communication – integrates weltanschauungs, or worldviews Embraces conflict and dialectics Encourages exploration and creativity and is opportunistic Manager is facilitator Design and implementation are inseparable and evolve iteratively
Rationality	Technical/functional	Substantial
Theoretical and/or philosophical roots	Logical positivism, scientific method	Action learning theory, Dewey's pragmatism, phenomenology

Traditional and emerging perspectives of design.

leveraging change [3]. Big upfront designs/plans and extensive documentation are of little value to practitioners of agile methods. Important features of this approach include evolutionary delivery through short iterative cycles—of planning, action, reflection—intense collaboration, self-organizing teams, and a high degree of developer discretion.

Erstwhile software development practices reflected the principles of traditional design outlined in the table. The nature of the related problem solving involved analysis, prediction, verifiability, control, and optimization. Design was construed as a sequence of well-articulated steps aimed at choosing the best option to satisfy a predetermined end. Careful planning and early baseline definitions for a robust architecture were seen as ways to anticipate and control variations in processes, but it lacked the active participation of customers throughout the development process. These traditional practices reflected a hierarchy involving a command-and-control style of management with clear separation of roles.

The emerging agile philosophy heralds a new epistemology of software development. Its value depends largely on an organization's ability to nurture learning, teamwork, self-organization, and personal empowerment. Responsiveness and flexibility are achieved through a "heterarchy" characterized by self-organizing teams whose members collaborate, improvise according to problem context, and use their ingenuity to solve problems [8, 9]. The theoretical principles of holographic organizations hold considerable promise in explaining the conceptual underpinnings of agile practices that share many of the characteristics of the Collaborative Action Learning method described in [10].

Holographic Organization

The metaphor of the holographic organization draws its inspiration from the fact that every fragment or piece of a broken holographic film contains the information required to completely construct the image represented in the whole film [8, 9]. That is, what is in the whole hologram can be accessed through any of its parts, as each one is a reflection of the whole. Evidence suggests that the brain's memory, intelligence, and functionality are distributed across its parts such that damage to some of the parts does not result in complete loss of the overall functionality [9]. This redundancy of functions within parts, coupled with the capacity to learn, provides for flexibility and responsiveness—characteristics critical for survival in a complex and turbulent world.

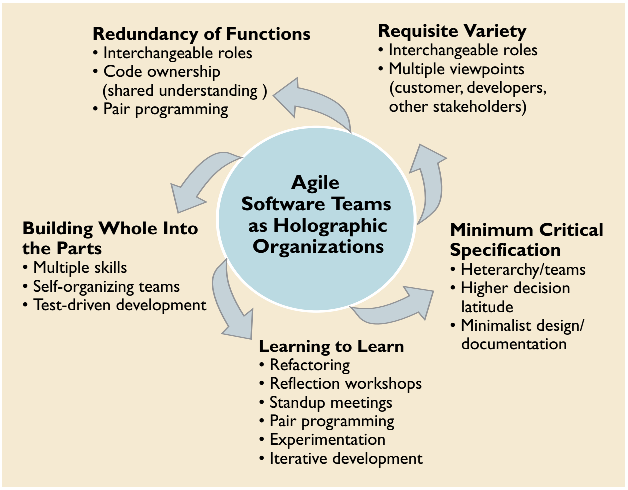
Redundancy of Functions
• Interchangeable roles
• Code ownership (shared understanding)
• Pair programming
Requisite Variety
• Interchangeable roles
• Multiple viewpoints (customer, developers, other stakeholders)
Agile Software Teams as Holographic Organizations
Building Whole Into the Parts
• Multiple skills
• Self-organizing teams
• Test-driven development
Minimum Critical Specification
• Heterarchy/teams
• Higher decision latitude
• Minimalist design/ documentation
Learning to Learn
• Refactoring
• Reflection workshops
• Standup meetings
• Pair programming
• Experimentation
• Iterative development

Figure 2. Alignment of agile practices with holographic principles (adapted from [8]).

The agile philosophy facilitates formation of holistic teams through a culture that encourages the interchangeability of roles or jobs based on autonomy. It builds both specialized and generalized skills among members so they become multifaceted and exhibit overlapping knowledge so they can self-organize in response to emergent requirements. The team's redundant skills enable it to function even when multiple members are missing. Such redundancy enhances the system's responsiveness to changing environments and complexity.

According to Ashby's law of requisite variety, a system's internal variety should at least match the variety and complexity of the environment with which it is confronted [4, 8]. The diversity (such as active stakeholder participation) and redundancy of skills (such as interchangeable roles) built into integrated self-organizing agile teams amplify the variety within the system. This variety enables agile teams to respond with minimal delay to the changing environment.

The notion of minimum critical specification refers to the identification of the smallest set of requirements to initiate a project. Defining all specifications at an early stage, some of which may be speculative, can constrain the inventiveness of team members to be locally responsive to opportunities that unfold in the problem-solving context. Minimum critical specification is therefore required to provide an organizational climate conducive to innovation and creativity. Agile team members are relatively diversified autonomous parts of a heterarchy, not subject to the rules and dictates that normally constrain actions in a hierarchical organization [9, 10]. Managers in agile teams lend focus to team members' actions, facilitating a thriving self-organizing environment. Managers who avoid extensive up-front planning and design while emphasizing design simplicity are consistent with the principle of minimum critical specification. Just as strategic intent guides thought and action throughout the process of strategy emergence, defining architectural intent may serve as a minimum critical specification.

Agile approaches encourage change and question assumptions, as reflected in, for example, the "speculate-collaborate-learn" cycle of Highsmith's Adaptive Software Development, stressing the importance of flexible/adaptive planning, extensive collaboration, and learning in achieving agility [3]. Practices include continuous code integration, refactoring to improve design and code, reflection workshops and stand-up meetings to determine what worked and what didn't, and instant feedback from participating stakeholders. These practices facilitate double-loop learning, supporting the principle of "learning to learn" [8]. The agile approach to software development appears to align itself with the principles of holographic organization theory (see Figure 2).

CONCLUSION

The tenets of agile methods depart from the traditional orthodoxy of software development. This shift in philosophy is not unusual, as similar patterns of intellectual evolution have emerged in other disciplines. A look at architecture and strategic management reveals that the progression of ideas in them is remarkably similar to conceptual pattern shifts in software design. Even a quick look broadens our horizons and enriches our inquiry into the evolutionary nature of software methodologies.

An expansive metaphor of design [5] and the theory of holographic organization [8, 9] offer a strong theoretical basis for the conceptual foundation of agile methods. Efforts to understand the theoretical roots of software development practices by examining the evolution of design ideas in architecture, strategic management, and other disciplines are even more relevant as system domains extend beyond simpler needs (such as technical functionality) to the complex social aspects of software applications (such as aesthetics, values, human judgment, morals, and ethics). C

REFERENCES

1. Argyris, C. and Schon, D. *Organizational Learning: A Theory of Action Perspective.* Addison-Wesley, Reading, MA, 1978.
2. Bazjanac, V. Architectural design theory: Models of the design process. In *Basic Questions of Design Theory,* W. Spillers, Ed. North-Holland Publishing Co., Amsterdam, 1974, 3–19.
3. Highsmith, J. *Agile Software Development Ecosystems.* Addison-Wesley, Boston, MA, 2002.
4. Jackson, M. *Systems Methodology for the Management Sciences.* Plenum Press, New York, 1991.
5. Liedtka, J. In defense of strategy as design. *California Management Review 42,* 3 (Spring 2000), 8–30.
6. Mintzberg, H., Ahlstrand, B., and Lampel, J. *Strategy Safari: A Guided Tour Through the Wilds of Strategic Management.* The Free Press, New York, 1998.
7. Mintzberg, H. Crafting strategy. *Harvard Business Review 65,* 4 (July-Aug. 1987), 66–75.
8. Morgan, G. *Images of Organization.* Berrett-Koehler Publishers, Inc., San Francisco, 1998.
9. Morgan, G. and Ramirez, R. Action learning: A holographic metaphor for guiding social change. *Human Relations 37,* 1 (Jan. 1984), 1–28.
10. Ngwenyama, O. Developing end users' systems development competence. *Information & Management 25,* 6 (Dec. 1993), 291–302.
11. Schon, D. *The Reflective Practitioner: How Professionals Think in Action.* Basic Books, New York, 1983.
12. Senge, P. The leader's new work: Building learning organizations. *Sloan Management Review 32,* 1 (Fall 1990), 7–23.

SRIDHAR NERUR (snerur@uta.edu) is an assistant professor of information systems in the Department of Information Systems and Operations Management in the College of Business Administration at the University of Texas at Arlington, Arlington, TX.
VENUGOPAL BALIJEPALLY (vebalijepally@pvamu.edu) is an assistant professor of information systems in the Department of Accounting, Finance & MIS in the College of Business at Prairie View A&M University, Prairie View, TX.