Seeking a new Paradigm for Software Project Management – is it Agile, Lean or a Model of Concurrent Perception

Roy Morien
roym@nu.ac.th
Department of Computer Science and Information Technology, Faculty of Science
and Faculty of Management & Information Sciences
Naresuan University, Phitsanulok, Thailand

1. Abstract

Since the late 1970s the practice of Software Project Management seems to have been quite constrained within the basic strictures of the Waterfall Model of development and project management. The literature abounds with suggestions of how to improve this way of thinking. Rigor in planning, estimating and following the plan has remained the centre points of such suggestions. This has not proved to be successful.

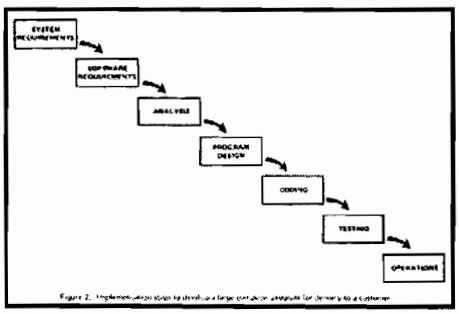
This paper will draw together concepts and practices of 'agile development', 'lean product development', 'chaordic systems' and 'the Model of Concurrent Perception' to suggest a new Paradigm of Software Development and Project Management.

2. Introduction

The true nature of the software development process and a clear definition of its characteristics is missing from the literature. The names in use clearly imply an activity akin to civil engineering and construction projects. Principle among these is the set of practices termed Software Engineering. In Object Oriented development, the Pattern oriented approach includes a Pattern of 'software factory'. We see discussion about the role of the Database Architect, a primarily construction role and activity. Project planners are exhorted to 'plan the work and work the plan', and requirements documents are referred to as blueprints. A professional project management activity is the development of a Gantt chart, and students of Project Management inevitably are taught to use some form of project management tool, such as Microsoft Project(R).

The Waterfall Model of development clearly seeks to achieve certainty in the future project activities and certainty in outcomes. Contracts are drawn in such a way as to give clients the confidence of certainty in what they can expect to receive at a certain time in the future, to be developed at a certain cost, with certain functionality (that they themselves must provide, of course). The contracts that are drawn up seemingly provide the supplier with certainty of income, certainty of scope, therefore certainty of project activity, undertaken with certainty according to a plan.

Nonetheless, adherents of the Waterfall Approach have frequently modified the Waterfall Model to include feedback loops, necessary to overcome the initial model of a system without feedback mechanisms. The model has been enhanced by the adoption of Change Management practices, which fundamentally acknowledges that things do change with the passing of time. This, however, is given restrictive form in favour of the principle of adhering to the plan. Moving off the plan, failing to achieve critical path outcomes as planned, including different matters in the plan, are all seen to be undesirable. The epithet 'scope creep' is applied to this wayward practice, and is always considered undesirable and a sign of poor project management.



To ensure that this discussion is understood, here is a diagram of The Waterfall Model, as published in Royce [1]. Interestingly, Royce seems to have meant that these are the activities involved in software development, even though he labelled them 'the implementation steps'. What appears to be an unfortunate use of words is what seems to have given rise to the adoption of this Model of linear or serially phased development, which was not what Royce intended.

This paper will explore an alternative view of software development projects, and propose a new way of thinking about the activity of software development, and software project management. The discussion will draw upon management literature normally considered outside the scope of software project practice, and will explore the nature of the software development activity, and consider a new approach to software project management based upon the different characteristics identified as being the more correct nature of that activity, not the engineering oriented nature that has hitherto been considered the received way.

The Four Dimensions of Software Projects

In the book *Rapid Development*, by Steve McConnell [2], it says that a system development activity includes four things that are important: These are:

- People - Tools - Process - Product

A development activity is done by PEOPLE (for PEOPLE), using development TOOLS, while following a PROCESS, to produce a PRODUCT.

Clearly, all four must be included and considered in any discussion of the successful project. However, McConnell states that

... we now know with certainty that peopleware issues have more impact on software productivity and software quality than any other factor (at p.12)

... it is now crystal clear that any organization that's serious about improving productivity should look first to the peopleware issues of motivation, teamwork, staff selection and training. (at p.13)

Many other authors have explored the importance of 'the people' in organisations and organizational activities. Senge [3], quoting Kazuo Inamori, founder and president of Kyocera, a world's leading company in advanced ceramics technology. *'whether it is research and development, company management, or any other aspect of the business, the active force is 'people'.*

Champy, in discussing reengineering management[4] states '*... responsibility and authority are so widely distributed throughout the organization that virtually everyone becomes a manager, if only of his or her own work'* (at p.70), and *'...our core values, our company culture based on trust and respect for individuals. It's about empowering people at the lowest level of the organization to run with their ideas. That freedom fosters a lot of creativity and enthusiasm'.* (at p.70), *'We just started giving out assignments and trusting people to carry them through.'* (at p.70) and *'We are all in this to build a sustained competitive advantage. You are not here solely to perform your function. You are here to add value.'* (at p.73). Finally, Champy discussed *' ... enabling (people); redesigning work so that people can exercise their skills and capabilities to the fullest extent possible – then stepping back and letting it happen.'* (at p.115). de Geus [5] states *'... organizations' true nature is that of a community of humans.'*

It is possible to find many more references in the literature to aspects of human behaviour in organizations, references to the ability of people to work independently without close supervision, to be creative, to be self-managed, and the central role that 'people' play in the success or failure of organizational processes. Developing software is a human and business activity. It is also a technical activity, but first, it is a 'human-centric' activity. Therefore, any development process will only be effective if it enables the people to perform at their best, and most creative. The development process must enable learning, and enhance the capabilities of the developers. Project management must be people-management, first and foremost, and acknowledge that the people are well-trained, competent professionals, or are at least eager and willing to learn and become so. Developers will contribute the technical knowledge to the development project, and the client, or client representative, must contribute the business knowledge. This means that the developers, and the clients, must work closely together, collaboratively, and must recognise each others' abilities and knowledge.

3. Managers Must Manage, Damaging as that is.

One of the problems with having managers is that they always seem to want to manage things. Unfortunately, this activity is usually manifested as a 'command and control' approach to directing the lives and work of subordinates. There is a supposition, not always correct, that because the manager is the manager, and has subordinates, then the manager knows better than anyone what should be done, and, more particularly, how it should be done and how long it should take. This has too often proven incorrect and even damaging to the prospects of the business unit, and the business as a whole. In their book *The Wisdom of Teams: creating the high-performance organization*, Katzenbach & Smith [5] recount the experience of a major railroad company which, in the early 1980's, was facing huge competitive challenges following deregulation of transport in the US. A highly successful team was opposed by senior managers. The management culture (fundamentally an hierarchical, command and control culture) had not changed for 100 years, and the fact that managers continued to manage, imposing a command and control management regime, was leading the company to oblivion. It was only after this management culture was overcome that the company's fortunes were substantially improved. Modern 'management' demands leadership, not control.

This command and control approach to project management has been the main management style on most software projects which have operated under the phased, linear development model as described in The Waterfall Model (now a generic name for the style of project management that has linear or serially phased activities). It must be so, because that development model demands up-front plans, up-front requirements determination, adherence to the plan, and does not allow for adaptation of plans or the evolution of requirements. Nor does it have regular and frequent feedback cycles, relying on the project manager's estimates and plans to be correct *ab initio*. The project manager must maintain strict control over all activities to ensure their estimates are met.

4. Chaordic Systems & Ecosystems

In the landmark book Birth of the Chaordic Age [6], Dee Hock coined the term 'chaordic' to describe '*the*

behavior of any self-governing organism, organization or system which harmoniously blends characteristics of order and chaos'. It is suggested here that the activity of software development, as a system, manifests those characteristics. Latterly, the terminology of 'the digital ecosystem' has also indicated that *'The digital ecosystem is defined as an open, loosely coupled, demand-driven, domain clustered, agent-based self organised collaborative environment ... for a specific purpose or goal, and everyone is proactive and responsive for their own benefit or profit. The essence of digital ecosystems is creating value by making connections through collective intelligence. Digital Ecosystems promote collaboration instead of unbridled competition and ICT based catalyst effect in a number of domains to produce networked enriched communities.'* It is suggested that a software development project is better described as an ecosystem, with many characteristics of being a digital ecosystem; *'promoting collaboration', 'a collaborative ... for a specific purpose or goal* (the development of a system)*', ' creating value by making connections through collective intelligence* (see The Wisdom of Teams)*'*.

Can these be at least some of the characteristics of the software development activity? Is it, or can it be, chaordic, inasmuch as it can be a self-governing organism, or an ecosystem that promotes collaboration and achieve a specified purpose or goal. It is suggested that this is indeed the case.

Therefore, as such, it defies a management 'command and control' style. As Champy (op.cit.[4] states, variously, *'You must have a culture that encourages qualities like relentless pursuit... bottomless resources of imagination ... and both smooth teamwork and individual autonomy'* (and therefore) *'... You cannot have a culture of obedience to chains of command and the job slot. It just won't work.'* and *'the best approach to such a system is '... enabling (people); redesigning work so that people can exercise their skills and capabilities to the fullest extent possible – then stepping back and letting it happen.'* That is, let the ecosystem work, let the harmonious blend of order and chaos occur, and create a collaborative environment.

5. Project Success Criteria

In the traditional Waterfall Approach, every attempt is made, at the start, to impose order on the process. Has this been successful? Can it be successful? Many research projects have identified the unfortunate statistics of failure, such as only 2% of systems were used as delivered, and 28% of systems that were paid for but were never delivered, and 47% of delivered systems were never used. *'...53% of projects overrun cost estimates by 189% or more (at a cost of US$59 billion per year in the US alone)'* (Standish Group, [7]). This research is supported by a US government study on software development projects, which revealed that 60% of projects were behind schedule and 50% were over cost (cited in Garmus & Herron, [8]). The study also showed that 45% of delivered projects were unusable, a dismal comment on the state of software development projects. The experience in Australia of the Australian Customs Department is instructive. Initial estimates of $35 million, and 3 years, blew out to $250 million and 5 years, with huge and damaging outcomes when the system failed to perform properly when 'switched on'.

An important question is - Why is this? Perhaps it is because '(project management activities) *are undertaken in the mistaken belief that what sometimes works in independent manufacturing processes will succeed in software development'*. This view of software development as a manufacturing process, or an engineering process, has been significantly deprecated by many authors, even though it seems to have been a central philosophy upheld by many others - cf: the terminology of 'software engineering'. For example, *'...someone grabbed hold of the construction-manufacturing paradigm which suggests that we can layout an architecture, design the system, and construct it. Experience has shown that this is a painful and expensive way to develop dinosaurs.'* and *'The construction paradigm is the major reason that so many customers are dissatisfied...'* and *'...software doesn't build, it grows and evolves'* (Arthur, [9]).

There is an interesting implication to be drawn from these statements, which can be stated in these terms - project success is measured against three standard criteria (within cost budget, within time schedule, within defined scope), yet two of those criteria are not met in at least half of projects, or, projects are evaluated in terms of how well they meet estimated cost budgets and schedule estimates, yet at least half of these projects fail to meet these estimates by a substantial margin. As for the third criteria; within scope, the Standish Group (op.cit [7]) reports that only 7% of systems delivered 100% of required features (46% of systems delivered more than 75% of required features, and overall, on average, systems delivered only 61% of required features). An interesting further consideration is that research has shown that a substantial proportion, up to 60%, of features delivered in many systems are never or rarely used by the users of the system.

6. The Model of Concurrent Perception

Can there be an harmonious blend of order and chaos? Rubinstein et al [10] proffers a model of decision making behaviour that describes most compellingly the characteristics of chaordic systems. It is this model; *The Model of Concurrent Perception*, that seems entirely appropriate to the activity of software development. The Model of Concurrent Perception 'moves us from questions to answers, from divergent perceptions to convergent perceptions, from individual creativity to team implementation, from abstract thinking to concrete action, from quick experimentation to quality results, from deliberate chaos to emergent order' and 'chaos

should be deliberately created up front'. By 'chaos' they mean that the situation be thrown open to participation and discussion by all interested stakeholders, and a rich mix of views, opinions, suggestions, expertise and ideas be aroused, thus creating a 'chaotic' situation from which order will emerge. *'Questions need to be raised from the outset. When you start out with divergent questions, you will end up with convergent answers. When you start out with chaos, you will end up with order.'*

To create 'certainty' is to imply that the future can be controlled, which is a fallacy. Uncertainty is the hallmark of the future. Ours is not to know the future, but just to plan for it, including whatever contingencies can be foreseen. The Model of Concurrent Perception says this in this way; *'This* (start with chaos, end up with order) *is far preferable to the scenario where everyone coasts through a seemingly structured and orderly project and the end result is chaos'*. This last phrase seems to almost perfectly describe the traditional phased software development approaches where every effort is made, by the creation of a detailed and 'frozen' requirements specification, and a detailed plan that is rigorously held to, to have 'a structured and orderly project'. 'Plan the work and work the Plan' is seemingly the motto of the 'successful' project manager. Research has shown that the end result of such an approach seems too often to end in chaos, characterized by disappointment, rejection and refusal to use the resultant system. Or the delivery of a system that is less than useful, and has low business value. These statistics clearly indicate a descent into chaos from a starting point of imposed order. Assuming that these systems were developed using a traditional phased approach, which is not an unreasonable assumption, we can see relevance and correctness of the situation of 'seemingly structured and orderly project' where the 'end result is chaos'.

There are many examples of highly successful projects that seem to be well described by the Model of Concurrent Perception including the development by the Boeing Corporation of the 777 airliner, and the development of the Lexus luxury motor vehicle by the Toyota Company.

In the development of the 777 airliner, for example, the project manager 'created more than 200 design/build teams with members from design, manufacturing, suppliers and customer airlines – everyone from pilots to baggage handlers'. All project teams and members were urged to 'share early and share often'. The project scenario being painted here is clearly the *'...start out with chaos'* situation, which, in this case resulted in the creation of a highly successful airliner.

In the development of the Lexus motor vehicle, as described in Liker [11] it is stated that '(in vehicle design) *Effectiveness starts with what is popularly being called the 'fuzzy front-end'*. The project leader stated *'The end result was not just my effort alone, but all the people along the way who originally opposed what I was doing, and who all came around and were able to achieve all these targets that I had set in the first place'*. The Lexus motor vehicle is a very popular model in the marketplace. Various aspects of the Model of Concurrent Perception were clearly able to be seen here. *'Questions need to be raised from the outset'*; indeed many questions were raised about design issues, even about the need for the model. *'When you start out with divergent questions, you will end up with convergent answers'* was demonstrated by the people 'who all came around' and achieved the design targets. *'When you start out with chaos, you will end up with order'*.

A much- performed activity in many projects, including software development projects, is that of brain storming. Brain storming sessions are governed by some quite strict rules, including 'At the start, every idea is accepted without argument or disagreement'. Successful brain storming meetings create the initial chaos of many ideas and suggestions, and then allow the order to emerge as decisions agreed upon. SWAT analysis can be seen also as a chaos-creating activity from which order emerges. Estimating methods such as the Wide Band Delphi [12] have an emphasis on 'brain storming', or shared decision making.

7. The Learning Organization

Elsewhere we can go to the literature about a management discipline outside IS and Computer Science to seek insight into the best way to develop software systems. In this case, to view the software development function in terms of being a Learning Organization.

The concept and practice of the learning organization is amply discussed in Senge (op.cit.[3]). Peter Drucker defined a learning organization being necessary because *'The function of the society of post-capitalist organisations ... is to put knowledge to work ... it must be organised for constant change'*.

The Core Capabilities of a Learning Organization are summarized as (1) Creative orientation, (2) Generative discussion, and (3) Systems perspective (Maani & Cavana, [13] at p138.). These concepts are elaborated to mean:

- Creative orientation : The source of a genuine desire to excel. .. The source of an intrinsic motivation and drive to achieve … favors the common good over personal gains.
- Generative discussion: A deep and meaningful dialogue to create unity of thought and action
- Systems perspective: The ability to see things holistically by understanding the connectedness between parts.

Although Senge published nine years before Rubinstein & Firstenberg ([10], op.cit.) there are many similarities in their discussion. In Team Learning, Senge states (at p.236) *'team learning (has) the need to think insightfully*

about complex issues ... to tap the potential of many minds'. Other statements about team learning include *'... team learning involves mastering the practices of dialogue and discussion ... there is a free and creative exploration of complex and subtle issues ... '*.

This implies, it is suggested, the chaos that is present in the participant behaviour modelled by the Model of Concurrent Perception, and then the learning team converges on the order that is the hoped for outcome of *'divergent perceptions to convergent perceptions'*.

Similarly, when Maani & Cavana ([13], op.cit.) refer to *'Generative discussion: A deep and meaningful dialogue to create unity of thought and action'*, we can reasonably interpret the *'deep and meaningful dialogue'* to be the chaos and the *'create unity of thought and action'* to be the emergence of order, all of which seems readily defined by the Model of Concurrent Perception.

Another viewpoint here is what has been called 'the wisdom of the crowd', or 'the wisdom of crowds'. In *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, James Surowiecki [14] proposes the idea that the aggregation of information in groups results in decisions that are often better than could have been made by any single member of the group. Its central thesis, that a diverse collection of independently-deciding individuals is likely to make certain types of decisions and predictions better than individuals or even experts. This concept of group decision making has also been termed, somewhat inelegantly, as 'the wisdom of the herd' [15].

8. Leadership and Teams

Katzenbach et al ([5], op.cit) describe the work of a team involved in planning and implementing a major new business strategy for a large railroad company in the US, in the early 1980's. The efforts of the team were ultimately highly successful, in the face of entrenched opposition and even sabotage within the higher management of the company. The role of the leader of the team was a major factor in this success. The success of the team's project was attributed to (1) dedication to a common purpose (2) acceptance of a performance challenge (3) a sense of mutual accountability (4) candor and mutual respect between team members (5) a shared affection for each other, arising from the shared experience, and the shared success.

It is doubtful that a traditional project manager dictating the activities of the 'subordinates' in a 'command and control' style of management, could have achieved the same highly successful result. This has been the experience in many development projects, with the project manager ultimately disgraced for his 'failure' to estimate 'correctly', control rigorously, decide correctly, and designate appropriately.

Another telling case study on the idea of leadership as a success factor, published in Maani & Cavana ([13], op.cit.) is where, inn 1995, a team from New Zealand won the famous and prestigious yacht trophy, called the Americas Cup. This was only the 2nd time in 146 years that a non-US syndicate had won the trophy ... Australia had won it once before.

The amazing thing was that the NZ team's performance surpassed any previous campaign.

How did they do it? The success has been attributed to:

- The inspirational leadership of the syndicate Leader
- The strong sense of community within the team
- The openness of communication between team members
- 'Customer'- led development – the sailors!!!
- The sustained rate of continual improvement (of the boat speed)
- The level of commitment and purpose by all participants

This syndicate exhibited many of the valuable traits of a 'learning organization'. The contribution of 'leadership' to this outstanding success also cannot be underestimated.

So what is a team? Katzenbach et al ([5], op.cit) again provides a suggestion. *'A team is a small number of people with complementary skills, who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually responsible.'* It is suggested further that the 'small number' be optimally 10 or less.

A recent article [16] about Pixar (www.pixar.com), the organisation that develops full-length animated movies, is revealing as to Pixar's successful approach to 'product development', and its project management in a creative environment, which, as has been implied often in this article, are characteristics of software development projects. In summary, the product development approach has the following characteristics:

- a cohesive team moving from project to project. *'A team of moviemakers who know and trust one another in ways unimaginable on most sets.'*
- daily meetings where *'they ruthlessly "shred" each frame'*.
- an environment where mistakes are seen as opportunities for learning and improvement. *'We know screwups are an essential part of making something good.'*
- early identification of mistakes. *'our goal is to screw up as fast as possible'*.
- Upper management support and involvement. *'The upper echelons also subject themselves to megadoses of healthy criticism'*. This is leadership, not 'management'.

Clearly, this approach, which has demonstrably been shown to produce highly successful products, is everything that 'agile thinking' tries to be; highly iterative, fast feedback cycles, total transparency of

progress and outcomes, self-managed, validation and verification frequently and systemically.

9. Applicability & Relevance to Agile Methodologies

Included under the heading of Agile Methodologies for the purpose of this paper are development approaches that have been called Software Prototyping, Rapid Application Development, Iterative Development, and specifically the 'agile' approaches:

- People Focused: Collaborative, Self-Organizing and Self-Managing Teams.
- Empirical and Adaptive: 'empirical', 'adaptive', 'evolutionary', 'experiential' development, planning and estimating.
- Iterative: a series of short iterations each of which produces a useable enhancement to the system.
- Incremental: delivering increments to the system.
- Evolutionary: requirements *in detail* are continuously discovered, and are continually evolving.
- Emergent: The characteristics of the system emerge as parts are added.
- Adaptive: adaptive planning and estimating.
- Just-in-Time Requirements Elicitation: Requirements are stated in detail 'just in time' to develop them.
- Knowledge-Based: knowledgeable, self-managing team, continual knowledge sharing and learning.
- Client Driven, 'Pull-Based' development: Only develop what is asked for by the Client, and when the Client asks for it.

Agile methods emphasize project transparency, continual communication and collaboration between project partners.

10. Conclusion

It is suggested that computer software is now so ubiquitous in everyday private and commercial life that no discussion on business, business strategies etc. can ignore matters pertaining to software. The recent 'fall from grace' of the Toyota Motor Company, once an icon of quality and effective management, demonstrates in the negative the impact of computer software in business and daily life of virtually everyone. So the manner in which computer software development is managed must be of interest in any business forum.

Software development is seen as a chaordic activity, defying orderliness and certainty, and therefore demanding an appropriate approach to the management of that activity. A paradigm of project management that includes elements of what has been termed The Model of Concurrent Perception, the Learning Organisation, Leadership and Team Dynamics, has been discussed. The software project management approach traditionally used, based on, and inherited from civil engineering and construction project management practices is seen to have failed, as has the management model described as 'command and control'. The 'agile' approaches proffered by a growing number of experts in software development and software project management is the anti-thesis of this command and control style, and is well supported from the literature on management styles and practices, and case studies, that never seem to make it into software project management and software engineering curriculum.

11. References & Bibliography

[1] Royce, Winston, *Managing the Development of Large Software Systems*, Proceedings IEEE WESCON, August 1970, IEEE.

[2] McConnell, Steve, *Rapid Development: Taming Wild Software Schedules,* Microsoft Press, Published June 1996

[3] Senge, Peter, *The Fifth Discipline – The Art & Practice of the Learning Organization*, Currency Doubleday, 1990

[4] Champy, James, *Reengineering management: The Mandate for New Leadership,* Harper-Collins Publishers, 1995

[5] Jon R. Katzenbach, Douglas K. Smith, *The wisdom of teams: creating the high-performance organization*, Harvard Business School, 2008

[6] Hock, Dee, *Birth of the Chaordic Age,* Visa International, 1999

[7] Standish Group (1994). *The Chaos Report (1994)* [online]. Available WWW: http://www.standishgroup.com/sample_research/chaos_1994_1.php Accessed December 14th, 2003

[8] Garmus, David and David Herron (2001*), Estimating Software Earlier and More Accurately*, excerpted from *Function Point Analysis Measurement Practices for Successful Software Projects*, Addison-Wesley Information Technology Series, 2001.

[9] Arthur, L.J., *Rapid Evolutionary Development: Requirements, Prototyping & Software Creation*, Wiley, 1992

[10] Rubinstein, Moshe F. and Iris R Firstenberg, *The Minding Organisation*, John Wiley & Sons, 1999.

[11] Liker, Jeffrey K., *The Toyota Way*, McGraw-Hill, 2004

[12] Wide-Band Delphi, http://en.wikipedia.org/wiki/Wideband_delphi, accessed May 31st, 2010.

[13] Maani, Kambiz E. & Robert Y. Cavana, Systems Thinking, Systems Dynamics – Managing Change and Complexity, Pearson Education NZ, 2007

[14] James Surowiecki, James, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, Random House, 2005

[15] *'the wisdom of the herd'*, http://en.wikipedia.org/wiki/Collective_wisdom, accessed May 31st, 2010

[16] Wired Magazine, *Animating a Blockbuster: How Pixar Built* Toy Story, http://www.wired.com/magazine/2010/05/process_pixar, May 24th, 2010, accessed August 30th, 2010.