2009

Model of Critical Factors for Outsourcing Agile Development

Thomas Gradel
Temple University, tom@gradel.org

John T. Nosek
Temple University, nosek@temple.edu

Model of Critical Factors for Outsourcing Agile Development

Thomas Gradel
Temple University
tom@gradel.org

John T. Nosek
Temple University
nosek@temple.edu

ABSTRACT

Companies are beginning to combine outsourcing with Agile software engineering techniques with the goal of receiving the benefits of both – faster time to market, greater quality, and smaller costs. Since Agile was originally developed to work principally with small collocated teams, scalability of Agile to the enterprise, and simultaneous use of Agile and outsourcing are questions concerning applicability of Agile techniques to global business environments. This paper first summarizes current experience studies and research in Agile, enterprise Agile and Agile outsourcing, to identify factors likely to affect success on Agile projects. It then extends a model originally developed by Chow and Cao (2007) to account for these factors. Finally it outlines an experiment whose goal is to determine which of these factors drives successful projects that use both Agile and outsourcing.

Keywords

Agile, Model, Extreme Programming, XP, Scrum, Outsourcing, Global System Development

INTRODUCTION

Enterprise-level global applications of Agile are here, and have been for the past five years at Microsoft according to a recent report by Ade Miller (2008). Industry reports are gradually adding evidence that outsourced, distributed, and massive software applications have been, and are currently being developed using Agile software engineering techniques, proving that Agile can be successfully applied to large, geographically disperse, and internationally diverse environments. The original definitions of XP (Beck, 2000), Scrum (Schwaber and Beedle, 2001) and other techniques that adhere to the Agile Manifesto (Agile Alliance, 2001) focused principally on small collocated teams. Subsequent refinement of XP (Beck and Andres, 2004) and Scrum (Schwaber, 2007) provided some guidelines for enterprise-level Agile and some success stories and support for claims that Agile could scale and provide significant benefits in quality, cost and time-to-market. Case studies and industry reports such as those in the Agile'08 conference are adding experiences and lessons learned when applying these techniques.

Agile is a philosophy that encompasses twelve principles of the Agile Manifesto (Agile Alliance, 2001), twelve practices of XP (Beck, 2001) and a myriad of other techniques from Scrum (Schwaber and Beedle, 2001) when XP and Scrum are used together. These techniques form a coherent and mutually interdependent methodology for software development that should be studied together, but rarely are. Instead, much of past research focused on specialized Agile subsets such as pair programming (PP), test driven development (TDD) and applications of Agile to teaching software engineering (Dingsøyr, Dybå and Abrahamsson, 2008). There is little theoretical support to identify which Agile techniques drive project success, even though there is growing evidence that correct application of these techniques can bring about the desired effect, namely timely project completion with adequate scope, cost and quality.

The need for outsourced software development is growing (Carmel and Tjia, 2005), and 20% of those who responded to a recent survey on Agile plan on applying outsourcing and Agile together (VersionOne, 2008). The research community should provide theoretically sound recommendations for these new projects. This paper first presents relevant background in Agile and outsourcing, then presents a model for measuring critical factors that affect Agile, and finally extends this model to account for outsourcing. Subsequent research will validate the model and model extensions to identify critical factors that affect Agile outsourcing.

BACKGROUND

Agile is a family of software engineering (SE) techniques that adhere to the philosophy and principles of the Agile Manifesto (Agile Alliance, 2001), which defines a people-centric approach to developing software focusing on adaptability (agility), by working closely with customers and producing frequent releases of working software. Agile began to get wide recognition with Kent Beck's seminal work defining Extreme Programming (XP)(2000), which was later followed by Schwaber and Beedle's definition of Scrum (2002). XP defines a set of twelve developer-centric practices that identify how code is to be

defined, built, tested and integrated into periodic software releases named iterations. Scrum adds management practices to iterations, defines the requirement for daily stand-up meetings for intra-team coordination and formalizes what XP would name the "planning game", a set of practices that defines, prioritizes and estimates the effort necessary for incrementally integrating customer-specified features named "backlog items" (Scrum), or "user stories" (XP).

Outsourcing is a form of global software development (GSD) that involves a contract to an organization that is supplying a service to a consumer organization. Normally the outsourced supplier has an offshore component, so at least a portion of the work is accomplished in a country different from that of the consumer organization. The outsourcing company may also have a onshore component, in the same country as the consumer organization, but this is still considered outsourcing. Outsourcing and offshoring are often used interchangeably even though they have different definitions. Offshoring can, for example, refer to projects shared by various teams within a multi-national organization.

This section provides a brief summary of critical issues related to outsourcing projects using Agile.

Rationale for Outsourcing Agile-based Development

Agile adoption is growing and 20% of companies use, or plan to use Agile in outsourced development projects (VersionOne, 2008). Agile outsourcing attempts to blend the benefits of Agile (Table 1) with the benefits of outsourcing (Table 2) to realize the rewards of both (Moore and Barnett, 2004). Early adopters of Agile using outsourced teams are beginning to document experience reports that provide insights into interactions between these factors (Paasivaara, Durasiewicz and Lassenius, 2008; Summers, 2008; Uy and Ioannou, 2008; Armour, 2007; Miller and Carter, 2007; Danait, 2005; Hazzan and Dubinsky, 2005; Yap, 2005; Martin, Biddle and Noble, 2004; Kussmaul, Jack and Sponsler, 2004;Simons, 2002)

Agile Benefit	Description
Time-to-benefits	Frequent releases, continuous integration (CI) and burn-down charts show progress against milestones and provide timely feedback to customers and management.
Overall quality and efficiency	Test driven design (TDD) and pair programming (PP) are commonly used techniques that improve quality. Iterative development adds efficiency because customers can correct product deficiencies early, eliminating costly rewrites.
Team morale	Teams are self-directed and aided by management facilitators, leading to higher levels of developer buy-in, and corresponding levels of job satisfaction.
Improved relationship between IT and business staff	Business analysts are part of the team working along side development staff, and therefore have a greater opportunity for mutual trust.

Table 1. Benefits of Agile Processes

Outsourcing Benefit	Description
Cost savings	Cost savings is the primary reason why companies are outsourcing software development. Savings vary, but most report a minimum of 15% .
Time-to-market	Follow-the-sun strategies can provide round-the-clock development, provided daily handoff can be managed.
Overall quality and efficiency	CMM Level 5 provides mature development processes for achieving specified quality and efficiency goals, and many outsourcing firms have been assessed at this level.

Table 2. Benefits of Outsourcing

Agile Manifesto versus Outsourcing Agile

Adhering to Agile involves accepting a set of practices that involve substantial changes to the methods traditionally used for outsourcing. Table 3 summarizes why the philosophy suggested by the Agile Manifesto (Agile Alliance, 2001) could be antithetical to techniques commonly used when outsourcing. However, evidence is provided that outsourcing with Agile is still possible despite these inherent problems.

Even though CMM/CMMI are technically complementary, firms that have achieved CMM Level 5 certification have invested significant resources in creating elaborate plan-based process management systems that are not easily adapted to

Agile Manifesto Philosophy	Effect on Outsourcing
Individuals and actions over processes and tools	India has many companies certified in plan-based approaches such as Capability Maturity Model Integration (CMMI). Beliefs and training are against ad hoc SE methods such as Agile.
Working software over comprehensive documentation	Agile relies heavily on face-to-face communications to reduce the need for comprehensive documentation. Pair programming uses code as documentation supplemented by verbal communications to develop working software.
Customer collaboration over contract negotiation	Outsourcing involves a contractual relationship between a supplier and consumer, and success factors are strictly stipulated in the contract. Contracting consumers often prefer fixed price contacts which allow little room for negotiation on deliverables and timetables, and little room for agility.
Responding to change over following a plan	Changes are often controlled by a formal Engineering Change Notification (ECN) process, so that the size and impact of the change can be gauged before a change is permitted.

Table 3. Agile Manifesto and Outsourcing Compared

using Agile. Agile and Capability Maturity Model (CMM) are both ways to reduce risk, but their approaches are different. Agile reduces risk by releasing software in short iterations that allow the customer to verify whether the product answers the customer's needs. CMM reduces risk by careful control of software processes. Paulk (2001) argues that Agile and CMM need not be antithetical and this is now supported by Software Engineering Institute (SEI), the founders of CMM, in a recent technical note (Glazer, Dalton, Anderson, Konrad and Shrum, 2008). However, common practices for reducing risk involve extensive plan-based controls (Hussey and Hall, 2008), processes that are not normally part of Agile project management.

Working software requires extensive amounts of communication to define and refine requirements and to assure these meet the customer's needs. Agile's dependence on face-to-face and informal communication reduces the need for comprehensive documentation, but these communication modes become difficult with physical, language and cultural distance, forcing the need for additional communications channels. For example, XP's pair programming is still possible, but developers sit together via virtual, rather than physical terminals. Tabaka (2006) describes a number of techniques to enhance communications that could be implemented for distributed and outsourced Agile projects, and many of these techniques are reported in case studies involving outsourced Agile (Miller and Carter, 2007; Ramesh, Cao, Mohan and Xu, 2006).

Companies must move to more flexible outsourcing relationships to allow the contractor to respond to Agile dynamism. Contracting relationships between companies define and constrain service suppliers to certain duties and obligations, and these contracts can have profound effects on whether or not a service supplier can move to Agile software development. For example, a fixed price contract provides little freedom for a service organization to supply Agile teams subject to partially defined requirement that can evolve as software products are delivered. However, organizations prefer fixed price contracts and fixed deliverables (Leffingwell, 2007).

The SE community has mixed opinions with regard to how much planning is desirable in various organizational environments. Agility versus planning is an ongoing debate in the Agile community (Beck and Boehm, 2003) with Beck favoring the need for XP principles and light-weight planning, and Boehm favoring more extensive planning, particularly in larger organizations. Both agree that Agile is a disciplined approach, but Boehm makes the point that while 60% of software projects have ten or fewer developers and are thus more likely candidates for XP, this accounts for only 17% of the world's software development. Larger projects, which develop the bulk of the software, are better suited to plan-based approaches. Boehm and Turner (2004) compare plan-based and Agile-based mechanisms for evolution of the processes inherent in their respective approaches, and conclude that each approach can gravitate toward the other when there are environmental factors that favor the other approach. An Agile-based approach can evolve more plans and controls where necessary communication and feedback do not exist, and a plan-based approach can create more light-weight processes to respond more quickly to change.

Agile Outsourcing Models

Traditional outsourcing models need to be adapted to account for outsourcing Agile-based development. The traditional assumptions for labor costs and outsourcing constraints gives rise to the Pyramid of Activities shown in Figure 1, adapted

from a similar figure by Roussev and Akella (2006). The arrow direction indicates increasing amounts of the specified factors, for example the labor cost arrow indicates increasing labor costs at the bottom of the pyramid, and the arrow on the right indicates that risk increases moving toward the apex. This chart shows that applications programming is a more likely candidate for outsourcing than architecture definition because it has high labor cost and lower risk (Carmel and Tjia, 2005).

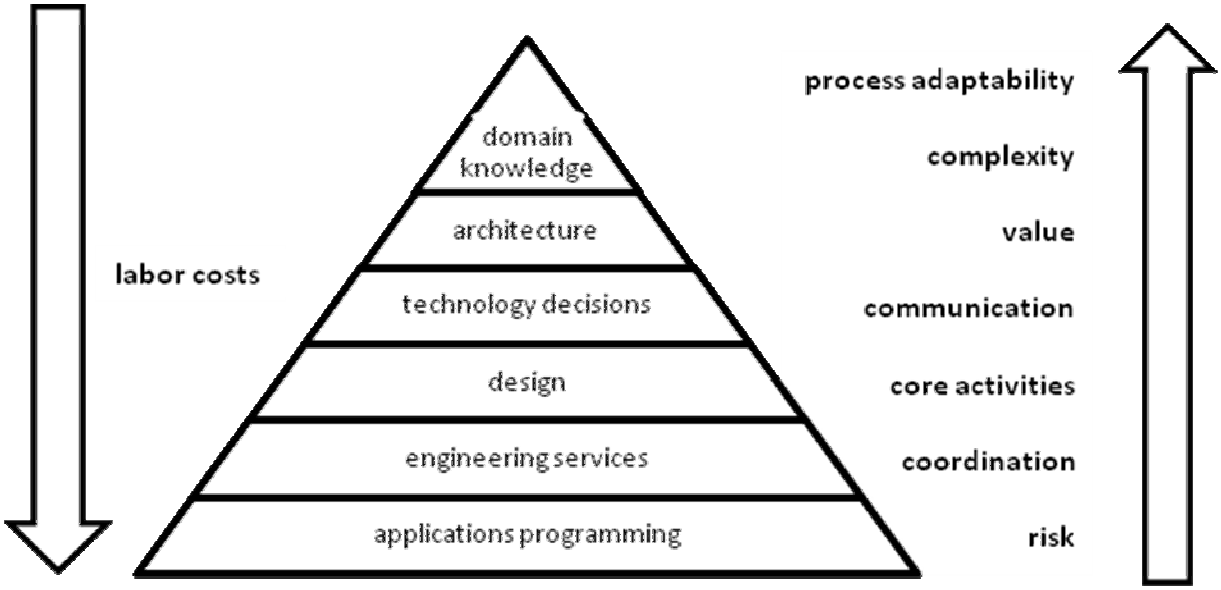

Figure 1. Pyramid of Activities

Traditional outsourcing models rely on cost curves for software development where costs dramatically increase when changes occur later in the development process. XP and other Agile methods challenge the traditional cost curve by using an incremental development process. Figure 2 illustrates the cost-of-change curve that occurs when using XP versus the traditional SE processes (Beck and Andres, 2004).

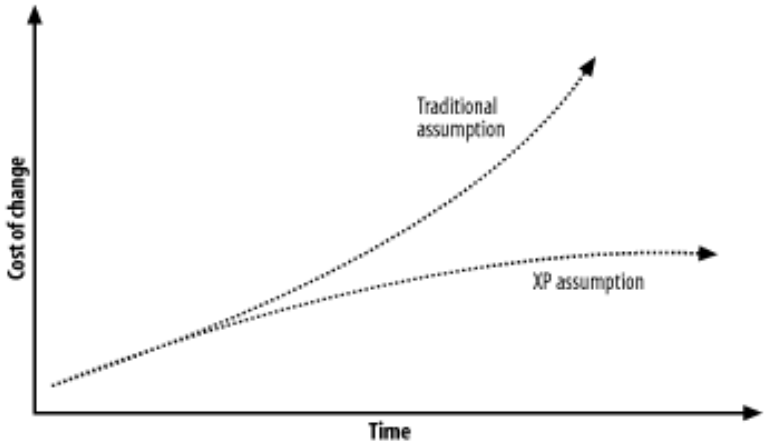

Figure 2. Cost-of-Change Curve Using XP and Traditional Development Processes

There is only limited empirical evidence to support the XP cost curve (Boehm and Turner, 2004), although there is rationale that iterative development could flatten the cost-of-change curve. XP does not simply divide software development into pieces and apply a mini-waterfall to each piece. If so, one would expect that the costs of each part would sum to at least the

total cost of the entire product, and probably more due to inefficiencies in separating and combining the pieces. Instead, the relatively flat cost-of-change curve for XP is due to refactoring (Fowler, Beck, Brant, Opdyke and Roberts, 2000) followed by adding new functionality, a two-step process where code is first adapted to accept new functionality, then new features are added to the code.

The team structure for Agile does not lend itself well to the traditional Pyramid of Activities model. Traditional business processes are shown adjacent to the right arrow in Figure 1, which shows for example, that complexity and communication needs increase moving toward the tip of the pyramid. A change to the domain knowledge has complex impacts that need to be communicated through the various layers down to the applications programming team. However, corresponding impacts would not be expected for Agile teams since these teams contain business analysts, architects and developers that jointly coordinate, so changes in domain knowledge can be immediately shared among all team members. However, communication and complexity increase because coordination and communication among outsourced team members depends on a set of factors not covered by the traditional outsourcing model.

Scaling Agility to Outsourcing

Leffingwell (2007) systematically analyzes how Agile can be scaled to large, enterprise-wide organizations, including those that use outsourcing. He identifies two sets of practices: those that natively scale to the enterprise level, and those that need to be adapted for enterprise organizations. Although Leffingwell includes two case studies that include outsourcing, he does not specifically identify practices that are sensitive to outsourcing. Nevertheless, sensitivities exist because in one of the case studies, he establishes parallel roles for the outsourcing team, with overlapping responsibilities between onshore and offshore, thus apparently duplicating effort to offset communications issues between onshore and offshore teams. In the other case study, the outsourced team is responsible for integration and testing, rather than being an equal partner in both development and testing.

Leffingwell's (2007) scaleable and non-scaleable practices are described in Table 4 and Table 5, annotated with research questions related to outsourcing.

Practices that scale to the enterprise	Outstanding questions related to outsourcing
Define/build/test component team	Teams should each define/build/test their own components. Are parallel teams (case study 1), or non-developer teams (case study 2) due to the effects of outsourcing?
Two-level planning and tracking	Does outsourcing require changes to the course-grain (release) or fine-grain (iteration) planning? Are there any special integration requirements due to integration with outsourced teams?
Mastering the iteration	If offshore teams are integrated with onshore teams, does increased communications delay make iterations longer? If teams are separate, does integration of offshore and onshore features add additional overhead?
Smaller, more frequent releases	Two-week iterations and quarterly releases are typical for collocated teams. Should the iteration size or release schedule be adjusted for outsourced teams?
Concurrent testing	Will outsourced teams be equal, or will they receive special testing due to potential miscommunication? Will the reverse occur, where testing is relegated to outsourced teams who are not as deeply involved with development?
Continuous integration	Will the offshore team be equal and responsible for maintaining a common code base, or will there be special rules that apply?
Regular reflection and adaptation	Will the outsourced team or team members be given equal weight when considering process modifications?

Table 4. Practices that Scale to the Enterprise Annotated with Outsourcing Questions

Practices that do not scale to the enterprise	Outstanding questions related to outsourcing
Intentional architecture	Agile normally has an evolutionary architecture that changes as new features are added, but this becomes problematic when scaling to the enterprise. Does outsourcing add additional needs for more precise architecture or design?
Lean requirements at scale: vision, roadmap and just-in-time elaboration	Agile develops minimally sufficient requirements (lean) which need additional definition for larger project (scaling). The vision and roadmap for doing this need to be understood by all teams so that elaboration is just-in-time, or wasted effort will occur. Specification of non-functional requirements such as performance, reliability and scalability of the system must be done first, since these are needed to develop individual components. Are there any special constraints for sharing the vision and roadmap to outsourced teams?
System of systems and the Agile release train	Individual Agile teams are optimized to write software quickly, so integration and management of releases becomes a limiting factor. Are there any special considerations necessary for integrating both offshore and onshore products from feature teams?
Managing highly distributed development	There are unique aspects, such as contract management that apply to outsourced teams. Are there other unique characteristics and how should these be managed?
Impact on customers and operations	This refers to impacts on sales and marketing due to more frequent releases of software. There appear to be no implications to sales and marketing specifically due to outsourcing.
Changing the organization	Agile requires organizational changes to remove impediments for developer teams. Are there any special impediments that apply to outsourcing?
Measuring business performance	Optimizing teams to produce more code is not the only measure of performance. Measuring performance and rewards for outsourced teams will impact future performance. What specific needs exist for outsourced teams and their companies?

Table 5. Practices that Do Not Scale to the Enterprise Annotated with Outsourcing Questions

Conchir, Holmstrom and Agerfalk (2006) provide an evaluation of the assumed benefits of Global Software Development. Table 6 provides these benefits together with the potential benefits of applying Agile.

Benefits of global software development	Potential benefit with Agile
Reduced development costs, for example an eight-fold salary differential between the US and India (Carmel and Tjia, 2005)	Salary costs are only one factor, and determining precise cost benefits is a complex metric. However, cost benefits that exists for existing outsourced development activities is likely to be transferable to an Agile environment.
Leveraging time-zone effectiveness by using follow-the-sun for development activities, or subsets of development such as testing, or production	Teams that include a mix of onshore and offshore resources will need daily handoffs to work as an integrated Agile team, but this could provide follow-the-sun benefits. Special offshore teams can exist for integration and production. Offshore resources can also provide additional feature teams increasing overall project throughput. Regardless of the structure, each team must feel responsible for project goals; there should be no "second class" teams.
Cross-site modularization of development work can be used to reduce the amount of cross-site communication.	There are no specific requirements in Agile for how teams should be established, or what work they should do. Additional architecture and design decomposition is normally a part of distributed software development projects even though most Agile methods minimize the effort spent in up front design.

Table 6. Potential Benefits of Global Software Development With Agile

Access to large skilled-labor pool.	High attrition levels in outsourced rapid growth areas can lead to reduced team effectiveness, since team dynamics change whenever a team member is added or removed. This might have a larger affect in Agile than traditional methods which depend on documentation as a means of recording project history. This could be countered if pair programming is used to provide total redundancy for each developer
Innovation and shared best practice	Agile encourages innovation at the team level and encourages trust to share best practices across teams
Closer proximity to market and customer	Agile encourages increased travel to disseminate customer and team goals, so the resulting product should be more aligned to markets.

Table 6. Potential Benefits of Global Software Development With Agile (continued)

MODEL FOR OUTSOURCING AGILE SOFTWARE DEVELOPMENT

The following sections propose a way to measure the effects of outsourced development. First a model is developed that will allow a factor analysis of those items that affect success. Then a set of propositions are developed based on the model.

Model Development

Agile involves a complex set of interactions among people, processes and environments. Studies of individual practices such as test-driven design (TDD) or pair programming (PP) provide insights into the merit of particular facets of Agile, but do not account for the effects of synergies between these techniques. Instead, a more cohesive model is necessary to determine which factors are critical to development success. Chow and Cao (C&C) (2007) performed such a study, and a summary of their results are shown in Figure 3. They classified various Agile principles and techniques into organizational, people, process, technical and project factors, and prepared a survey-based study to determine which of these caused significant impact to project success. Factors that impact the success measures of quality, scope, time or cost are identified in the figure with an asterisk (*).

A proposed model for determining the effects of outsourcing Agile development will expand on the C&C model by augmenting each of the categories to account for additional factors due to outsourcing. Table 7 summarizes the items that will be added to the model.

Hypotheses

The following are a set of hypotheses (expressed in the negative) that will be tested in the augmented C&C model. A survey-based approach will be used to test these hypotheses.

Hypothesis 1: There is no effect on project success factors when the teams are collocated versus when they are outsourced.

Survey questions will include information on the team locations, total team size, and size at each offshore location. A qualitative success factor for the most recent project (Agile or non-Agile) will also be recorded.

Hypothesis 2: There is no effect on project success factors when teams have same- versus cross-country members.

Survey results will indicate whether most teams have members all from the same country, or whether most teams have members from each of the distributed locations.

Hypothesis 3: There is no effect on project success when the reward factors are tied to team's success versus measures of effectiveness tied to the contracted company.

A question will exist as to whether rewards were by company, by team, both, or neither.

Hypothesis 4: There is no effect on project success if customer representatives exist for each country versus a single customer representative for several countries.

One or more questions will exist to determine the periodicity of travel for customer representatives and technical team members. Disbursement of product/customer knowledge will likely be examined by determining whether customer liaisons exist from all countries involved, or only from one.

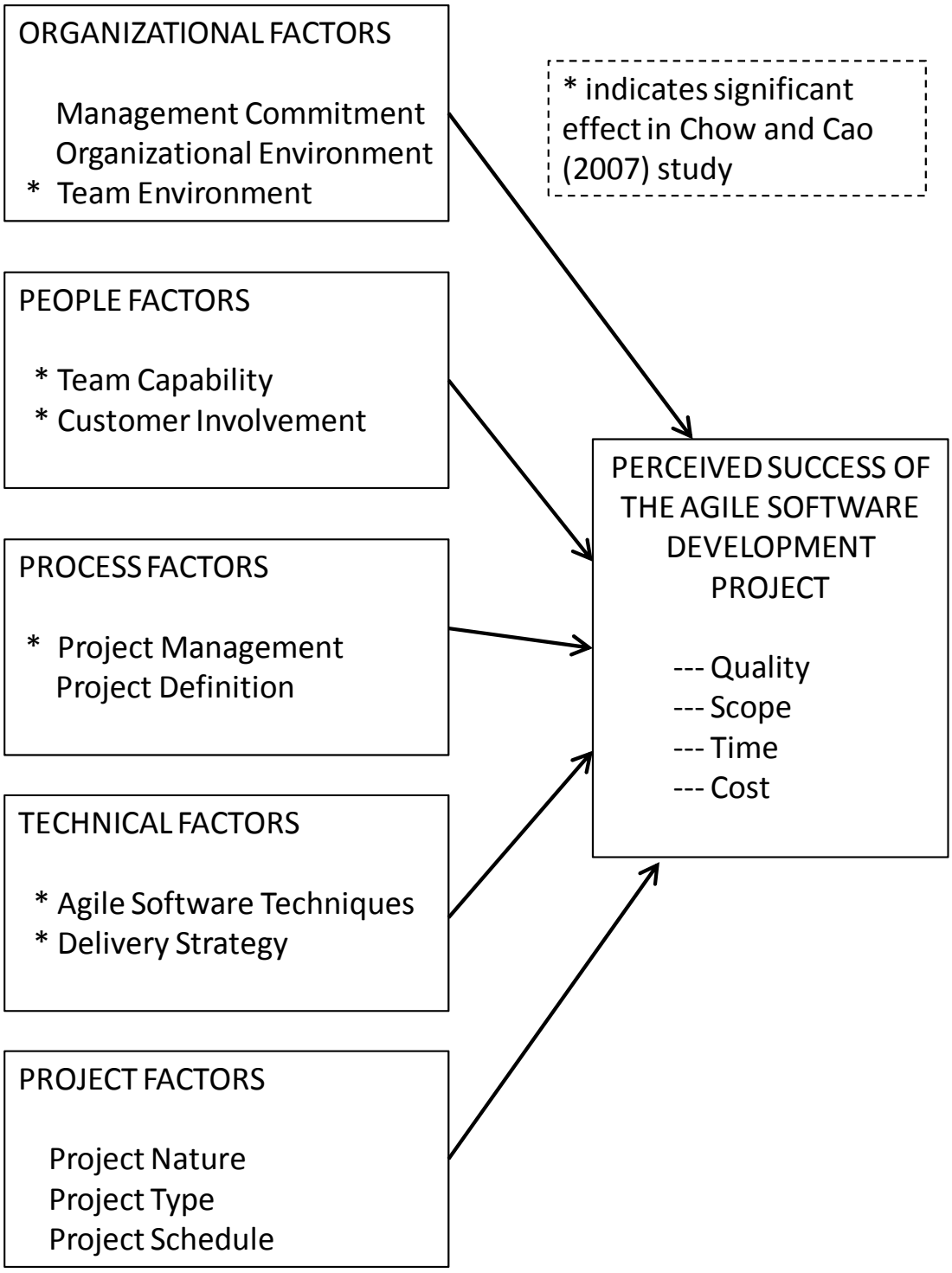

Figure 3. Model of Critical Agile Success Factors (Chow and Cao, 2007)

Additional Factors due to Outsourcing	Description
Organizational Factors	Outsourcing creates additional organizational requirements necessary to deal with offshore and contracted relationships. The team environment can consist of collocated feature teams, increasing integration difficulties, or follow-the-sun teams that required daily handoffs. The reward system must be appropriate to Agile, so team success rather than contractual success will affect long term team health. Tools play an increasingly important role, since these must help to minimize communications gaps.
People Factors	Different countries have varying work habits and cultures. Different countries offer labor pools with different skill levels. Managers and staff are more or less rigid and more likely to work well within an Agile environment. Customers, or customer spokespersons are most likely to come from one country, so interfaces to the customer may be better for same-country rather than cross-country teams. The ability to "buy-in" may be cultural and may need to be learned.
Process Factors	The degree which the design needs to be architected, designed and documented could change. Honoring regular working schedules may be difficult if there is need for face-to-face communication.
Technical Factors	Pair programming may be less feasible with cross-country teams than with same-country teams. Technical training could vary due to language barriers, because material not translated from English may not be read by non-native English speakers who prefer training material in their native language.
Project Factors	Highly innovative projects may result in the need for more communication, and thus may work better with some of the factors above, and worse with others.

Table 7. Augmenting C&C Model for Outsourced Global Software Development

Hypothesis 5: There is no effect on project success by increasing the amount of required specifications and documentation for outsourced teams versus maintaining the same level as for collocated teams.

Questions will be asked about whether specific types of documentation were prepared such as a high level architecture document, high level design documents and low level design documents.

Hypothesis 6: There is no effect on project success when adjustments in time and schedule are made to maximize overlap in face-to-face communication versus maintaining a standard work week for teams.

Questions will be asked about adjusting working hours to accommodate face-to-face communication, and whether this was all teams, team leads, or management.

Hypothesis 7: There is no effect on project success if specialized training is made to equalize team capabilities versus allowing teams that use their existing skill sets.

Questions will be asked about training in Agile and technical training before and during the project.

Hypothesis 8: There is no effect on project success for highly innovative software versus software with known previous versions.

A question will be asked about whether the project was an extension to an existing piece of software or a major change and/or new software release.

SUMMARY AND FUTURE RESEARCH

A model for measuring the factors that effect Agile outsourcing was presented, together with a set of hypotheses to validate the model and provide answers to sensitivities surrounding combining Agile with outsourcing. Future research will validate

this model and use it to provide insights into factors that drive Agile project success. The proposed experiment is expected to have the following outcomes:

- Repeats and independently validates the C&C model (2007)
- Determines whether there are significance differences between outsourced and non-outsourced Agile environments
- Provides insight into exactly which factors are important to driving success when combining Agile and outsourcing

We hope that readers of this paper would like to contribute, and encourage them to contact the author for additional information.

REFERENCES

1. Agile Alliance (2001) Manifesto for Agile software development, http://www.agilemanifesto.org, retrieved February 15, 2008.
2. Armour, P. (2007) Agile and offshore, *Communications of the ACM*, 50, 1, 13-16.
3. Beck, K. and Boehm, B. (2003) Agility through discipline: a debate, *Computer*, 36, 6, 44-46.
4. Beck, K. (2000*)* Extreme programming explained: embrace change, Addison-Wesley, Boston.
5. Beck, K. and Andres, C. (2004) Extreme programming explained: embrace change, Addison-Wesley, Boston.
6. Boehm, B. and Turner, R. (2004) Balancing agility and discipline: a guide for the perplexed. Addison-Wesley Professional, Boston.
7. Carmel, E. and Tjia, P (2005) Offshoring information technology, Cambridge University Press, Cambridge.
8. Chow, T. and Cao, D. (2008) A survey of critical success factors in Agile software project, *Journal of Systems and Software*, 81, 6, 961-971
9. Conchúir, E. O., Holmström, H., Ågerfalk, P. J., Fitzgerald, B. (2006) Exploring the assumed benefits of global software development." *IEEE International Conference on Global Software Engineering (ICGSE'06)*.
10. Danait, A. (2005) Agile offshore techniques: a case study, *Proceedings of Agile Conference 2005*, 24-29.
11. Dingsøyr, T., Dybå, T. and Abrahamsson, P (2008) A preliminary roadmap for empirical research on Agile software development, *Agile 2008*, 83-94
12. Fowler, M., Beck, K., Brant, J., Opdyke, W. and Roberts, D. (2000) Refactoring: improving the design of existing code. Addison Wesley Professional, Boston.
13. Glazer, H., Dalton, J., Anderson, D., Konrad, M. and Shrum, S. (2008) CMMI or Agile: why not embrace both! http://www.sei.cmu.edu/pub/documents/08.reports/08tn003.pdf, accessed February 13, 2009.
14. Hazzan, O. and Dubinsky, Y. (2005) Clashes between culture and software development methods: the case of the Israeli hi-tech industry and extreme programming, *Proceedings of Agile Conference, 2005*, 59-69.
15. Hussey, J. M., and Hall, S.E. (2008) Managing Global Development Risk, Auerbach Publications, Boca Raton.
16. Kussmaul, C., Jack, R. and Sponsler, B. (2004) Outsourcing and offshoring with agility: a case study, *4th Conference on Extreme Programming and Agile Methods,* Calgary, Canada, Springer Berlin / Heidelberg
17. Leffingwell, D. (2007) Scaling software agility: best practices for large enterprises, Addison Wesley Professional, Boston
18. Martin, A., Biddle, R. and Noble, J. (2004) When XP met outsourcing, *5th International Conference, XP 2004*, Garmisch-Partenkirchen, Germany, Springer Berlin / Heidelberg
19. Miller, A. (2008) Distributed Agile development at Microsoft patterns & practices, http://download.microsoft.com/-download/4/4/a/44a2cebd-63fb-4379-898d-9cf24822c6cc/distributed_agile_development_at_microsoft_patterns_and_practices.pdf, retrieved Februay 15, 2009.
20. Miller, A. and Carter, E. (2007) Agility and the inconceivably large, *Agile 2007,* Washington, D.C., IEEE Computer Society, 304-308
21. Paasivaara, M., Durasiewicz, S and Lassenius, C. (2008) Using Scrum in a globally distributed project: a case study, *Software Process Improvement and Practice*, 13, 527-544.
22. Paulk, M. C. (2001) Extreme programming from a CMM perspective, *IEEE Software*, 18, 6.

23. Ramesh, B., Cao, L., Mohan, K. and Xu, P. (2006) Can distributed software development be agile?, *Communications of the ACM*, 49, 10, 41-46.
24. Roussev, B. and Akella, R. (2006) Agile outsourcing projects: structure and management, *International Journal of e-Collaboration*, Oct/Dec, 37-42.
25. Schwaber, K. (2007) The enterprise and scrum, Microsoft Press, Redmond.
26. Schwaber, K. and Beedle, M. (2001) Agile software development with scrum, Prentice Hall, Upper Saddle River.
27. Summers, M. (2008) Insights into an Agile adventure with offshore partners, *Agile 2008*, 333-338.
28. Tabaka, J. (2006) Collaboration explained: facilitation skills for software project leaders, Addison Wesley Professional, Boston
29. Uy, E. and Ioannou, N. (2008) Growing and sustaining an offshore scrum engagement, *Agile 2008*, 345-350.
30. VersionOne (2008), State of agile development survey 2008, http://www.versionone.com/pdf/-3rdAnnualStateOfAgile_FullDataReport.pdf, retrieved February 15, 2009.
31. Yap, M. (2005) Follow the sun: distributed extreme programming development, *Proceedings of Agile Conference, 2005*, 24-29.