ADAPTING THE AGILE FRAMEWORK TO THE MANAGEMENT OF NON-IT KNOWLEDGE WORKERS

Robert Orwig, University of North Georgia
bob.orwig@ung.edu

Bryson Payne, University of North Georgia
bryson.payne@ung.edu

Nick Kastner, University of North Georgia
nick.kastner@ung.edu

ABSTRACT

Agile software development describes a set of software engineering management methodologies in which solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement and encourages rapid and flexible response to change (Beck et al., 2001a). The Agile Manifesto specifies four key value propositions for the agile development framework. Since development of the Agile Manifesto, the use of agile methods has moved into other areas such as project management and marketing.

The authors compare extant management, quality, and knowledge worker literature with the Agile Manifesto to build the case for using components of the agile framework in managing non-IT knowledge workers, from finance to health care professionals and beyond. This paper examines each concept and describe how each relates to traditional management thought, with a special emphasis on leadership in the agile framework. Using traditional management language, the authors create a new Knowledge Worker Manifesto, utilizing agile but applying it more broadly to all knowledge workers. The paper proposes leadership as a significant moderating variable to the earlier manifesto concepts.

INTRODUCTION

Agile development refers to the concepts of collaborative, iterative software engineering recorded in the Agile Manifesto (Beck et al., 2001a). The philosophy evolved from a group of software development methods that used self-organizing, cross-functional teams to promote adaptive planning, evolutionary development, early delivery, and continuous improvement while encouraging rapid and flexible response to change.

The roots of agile software development go back to the mid-1980s at Dupont and the works of James Martin and James Kern, proponents of Rapid Application Development and original signatories of the manifesto. Software development at the end of the 20th century had a collection of methods including competitive engineering, evolutionary project management and other incremental development methods (Larman & Basili, 2003). These methods grew into a collection of so called *lightweight* software development methods including unified process and dynamic systems development method, Scrum, crystal clear and extreme programming (Newkirk & Martin, 2001), as well as adaptive software development and feature-driven development. These methods predate the Agile Manifesto in 2001 and are collectively viewed as agile methods. The Agile Manifesto itself is a document that specifies how to complete software development projects productively.

Specifically, agile software development proposes to value:

- Individuals and interactions (over processes and tools),
- Working software (over comprehensive documentation),
- Customer collaboration (over contract negotiation), and
- Responding to change (over following a plan).

Beck et al. (2001b) further advocated twelve principles for agile software development:

1. Satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (in weeks rather than months).
4. Close, daily cooperation between business people and developers.
5. Build projects around motivated individuals, who should be trusted.
6. Face-to-face conversation is the best form of communication (co-location).
7. Working software is the primary measure of progress.
8. Sustainable development, able to be maintained a constant pace.
9. Continuous attention to technical excellence and good design.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly.

In the 16 years since the original publication of the Agile Manifesto, the use of agile has moved from software development into other management fields. Project Management has been the area of management literature that has most aggressively pursued agile since 2001, but by connecting the concepts of agile to the foundational management literature and broadening the concepts to include knowledge workers across multiple industries, leaders in other knowledge-intensive organizations can benefit from the thoughtful application of components of the agile framework to teams in their own fields.

BACKGROUND

Software developers and engineers commonly prefer agile methodologies to more traditional SDLC (software development life cycle) methodologies. The most traditional and basic alternative

to agile in the coding world is the waterfall model in which heavily documented user specifications are slowly converted into finished applications over periods ranging from months to years. The primary argument against the waterfall model was that the gap between the customer's desires and programmers' product diverged, predictably, over time, resulting in costly rework and cumbersome change requests, to which the Agile Manifesto offered a more customer-focused alternative. There is broad understanding within the software development industry that while agile is routinely preferred, the full implementation of agile development methods can be much more difficult than one might expect.

In the following sections, we will dissect each of the four guiding concepts in the Agile Manifesto, and provide an analysis of how each concept applies more broadly to knowledge workers beyond software development using the traditional language of quality. We will suggest that the Agile Manifesto's difficulties with implementation are linked to two problems. The first is that the Manifesto does not recognize its ties to extant management literature. The quality literature provides insight into potential implementation problems at least as early as 1970. Edward Deming (1986) famously provided his fourteen points in the book titled *Out of the Crisis*. In that seminal text, Deming explains the quality program that revolutionized Japan's economic approach. The values of employee empowerment, continuous improvement, customer-focus, and change management are all identified in Deming's work as crucial to the implementation of a quality program.

The Agile Manifesto highlights these four concepts in a different and interesting way. The Manifesto itself would benefit by recognizing this link. Second, the Manifesto does not incorporate leadership into its proposal. Deming and others writing about quality spend a great amount of time recognizing the importance of leadership. We suggest that the leadership gap that exists in the original Agile Manifesto should be acknowledged. When the Manifesto is adjusted with quality language, and the moderating variable of leadership is added, the result is a new manifesto more appropriate for all knowledge workers. This broadening of the Manifesto has implications for both practitioners and academics.

CONCEPT 1: INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS

The first concept in the manifesto focuses on the individual worker and their interactions with other team members, software and customers. According to Layton (2012), by focusing on the value of an individual and their interaction with a given project versus the processes and tools available, he states that productivity increases compared to traditional project management methodologies as there is less focus on conforming to existing processes and tools and more effort placed on new ideas and innovation. To support this concept and juxtapose agile project management with traditional project management methods, in interviews with 31 project managers from 10 different industries, Collyer, Warren, Hemsley and Stevens (2010) found that managers who used traditional project management methodology had difficulty in dynamic environments with three types of change including changes in materials, resources, and tools. Focusing more on individuals and interactions in the case of agile project management would help alleviate difficulty in these types of dynamic environments.

In addition to the benefits mentioned above in productivity and dealing with dynamic environments, there is a third improvement noted by researchers. Programming aptitude tends to define the success of a given project more so than the processes and tools used. According to Gnambs (2015), individuals that are more conscientious, open and introverted tend to have greater programming aptitude. Agile's focus on the individual allows for a closer, more intimate work experience and many supporters of agile focus here. Ironically, the coding world where agile originates was renowned for loners and introverts. The coder receives a job to do and months later would emerge with a completed piece of software.

The authors of the Agile Manifesto provide limited definition to how this value, as well as the remaining three values, are applied (Conforto, Salum, Amaral, da Silva, & Magnanini de Almeida, 2014), however they do provide some guidance utilizing the additional *12 Principles behind the Agile Manifesto* (Beck et al., 2001b). Principle 5 most clearly demonstrates the concept of valuing individuals and interactions over processes and tools: "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done" (Beck et al., 2001b). Though this principle provides some understanding of the value of individual workers, it does not define how this principle relates to the uses of processes and tools.

The focus of the first concept is simply a version of employee empowerment. Deming and others have championed it. Focused teams require it. Managers repeat, in mantra-like fashion, that "our people are our most important asset." With knowledge workers, that fact appears multiplied. Japanese executives in the 1980s were known for how they empowered their people to solve problems. Perhaps the best example of employee empowerment is found in the Toyota production system. This was identified at the time as a crucial difference in Eastern and Western management thought. Historically, employee empowerment is a traditional element of most quality programs over the last 30 years.

CONCEPT 2: WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION

It seems obvious, in software engineering, that a working piece of software would be better than documentation. The struggle here is with bureaucracy. It is difficult to support the idea of comprehensive documentation. The nature of documentation requires less than complete and comprehensive contracts. All sides agree that there is a point beyond which more documentation is useless and in fact is utterly wasteful. The essence of the manifesto is to remove waste.

The second value statement, "working software over comprehensive documentation" (Beck et al., 2001a), alludes to the iterative, incremental nature of agile software development. The *Principles* document states, "Working software is the primary measure of progress" (Beck et al., 2001b). In fact, these small, frequent deliverables in the form of working code help to enable the other three agile values. Producing working prototypes every few weeks enables developers to better collaborate with the customer; demonstrating proposed functionality in tangible form empowers both parties to provide more effective feedback more often. It is this rapid prototyping cycle that promotes the value of individuals and interactions, while providing a built-in mechanism for

responding to change. Working software is more useful than presenting documents to clients in meetings.

If the essence of the manifesto is to reduce waste, and therefore add value, it reminds one of the academic manufacturing concept of JIT (Just in Time)/lean. JIT/lean, broadly defined, is the elimination of waste. A key aspect to eliminating waste is to add value in all that you do. It is clear that working software has value, even if it does not fully solve the customers' problems. The idea of iteratively refining working products, adding new features through multiple beta-testing cycles, has been used successfully in software engineering for decades. Reducing paper requirements is recognized as a superb method for adding value while staying on track, and has been in the management literature for decades, as well (Deming, 1986).

The iterative nature of the agile software development process can be compared to the concept of the minimum viable product (MVP), a term coined by Frank Robinson in 2001, concurrent with the development of the Agile Manifesto, and made popular in books like *The Lean Startup* (Ries, 2011). A minimum viable product is one that "maximizes return on risk for both the vendor and the customer" (Robinson, 2001), by iteratively delivering the product that meets the customer's most immediate needs at each iteration. An MVP has just enough features to satisfy early adopters and provide a working beta product capable of eliciting feedback for future product development. The MVP form of iterative development mirror's traditional JIT/lean reduction of waste, as each iteration includes the minimum acceptable deliverable from the customer's perspective and maximizes productivity on the part of the knowledge worker.

For a knowledge worker, waste could be defined as time and effort devoted to products or features that do not meet the customer's most pressing needs. Distinct from Deming's quality paradigm, knowledge workers typically do not produce physical products, but information goods and services. Any time, effort, or focus by a knowledge worker that does not contribute to a next or near-term iteration of a minimum viable product could be considered waste just like overproduction or material waste in the manufacturing analogue. By focusing on the next iterative cycle's MVP, agile knowledge worker management embraces both the concept of a "working," minimum viable product in the new information age, and the waste reduction of traditional JIT/lean approaches.

A further key element to producing working, minimum viable products at each iteration cycle is the inclusion of quality assurance (QA) testers, preferably embedded in the agile team (Payne & Stretch, 2016). This aspect of modern agile teams borrows both from Deming and from Smith and Harry's Six Sigma, developed at Motorola in the early 1990's (Tennant, 2001). Embedded QA professionals in a knowledge worker team both test functionality and design test simulations for use by knowledge workers themselves in automating QA processes to ensure security, functionality, and user experience in information products and deliverables. While traditional QA teams can provide much of the same support, the adoption of embedded QA provides the more immediate, even continuous, feedback required in agile and Scrum environments that may be ideal for knowledge worker teams.

CONCEPT 3: CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION

Contract negotiation is another bureaucratic annoyance for the coder that is reduced by the implementation of this agile concept. However, it also seems similar to the customer focus required for high quality outputs. Requirements for the software development cycle require repetitive customer or stakeholder involvement. It is impossible to have a full understanding of the customer's needs without such involvement and yet prior to the mid-to-late 20th century, stakeholder involvement was not common practice (Lindborg, 2013).

The knowledge worker, in this case the computer programmer, is in the perfect position within the company to meet customer needs. Customer focus has a long management history, cited by many academicians over the years, notably by Deming (1986). Furthermore, it remains a major element in most quality programs (Heizer & Render, 2016). Programming prior to agile did not concern itself with customers until the time arrived for the contract to be fulfilled. At that point, they often found that they had created something that would not meet the customers' needs. Frequently, significant gaps resulted between customers' desires, needs and the programmers' interpretations of these needs. The primary reason for this was either there was a misunderstanding at the beginning of a given project as to what the customer expected, or the customer's needs had changed during the project. If a customer was not in consistent collaboration with the programmer or project manager, the end deliverable would fall short of his or her needs.

While software developers have always included customer input, it typically occurred, pre-agile, at the beginning of the project during the initial contract. This presented many problems and reduced productivity. For instance, the international web development firm Macronimous utilizes a more traditional development approach. In describing their eight-step "systematic development process," the majority of customer input takes place in only the first step. Additional feedback opportunities for the customer is only available at the end of each of the remaining steps (Benny, 2007).

This process cycle (receiving customer feedback, completing a section of the given project, sharing the completed section of the project to the client for additional feedback, address concerns, repeat) is a functional strategy, however there are also consequences. According to Ambler (2014), "traditional software developers will often adopt change management processes which are designed to prevent/reduce scope creep, but when you think about it these are really change prevention processes, not change management processes." Many software developers view scope creep as a tactic to maintain customer satisfaction without significantly altering the budget or contract with the client. However, additional variables including time, cost, personnel and their experience level, use case and defect counts can influence customer satisfaction (Madhuri, Rao, & V, 2013) dependent on industry and customer needs. Scope creep alone can cause significant and unforeseen budget changes. Lack of direct contact between the customer and team participants causes scope creep (Larson & Larson, 2009).

Agile software development requires up front planning (Coram & Bohner, 2005). Similar to other traditional methodologies, however, communication with the customer helps ensure that the first software release meets the needs of the customer (Serrador & Pinto, 2015). According to Hoda,

Noble, and Marshall (2011), lack of customer involvement in the agile process was one of the largest challenges agile teams face. Agile project management demands customer involvement to ensure success. A lack of customer collaboration results in adverse consequences both on the confidence of self-organizing teams, as well as the customer's satisfaction with the final product.

Whether it is utilizing predictive analytics to increase customer satisfaction (Hair, 2007) or developing a strategic approach to customer satisfaction through corporate culture development, distinctive customer value propositions and community development (Power, 2011), all of these efforts are equal to or more important than simple customer collaboration. Knowledge workers should have a more strategic approach to customers than just the tactic of collaboration. Customer collaboration requires a strategic approach, which, in turn, requires leadership.

CONCEPT 4: RESPONDING TO CHANGE OVER FOLLOWING A PLAN

For all business, one issue revolves around change and how the firm should respond to it. Following a plan has value, but agile argues that there is more value in responding to change, especially in the case of changing customer requirements. The final value statement, "responding to change over following a plan" (Beck et al., 2001a) is perhaps the foundational element of agile development, and it is likely the quality that differentiates the agile methodology most distinctly from the frameworks that preceded it.

Beck suggests that welcoming change is critical "for the customer's competitive advantage" (Beck et al., 2001b). Older software development methodologies, like the much-maligned waterfall method (Bell & Thayer, 1976), had rigid, sequential steps like *system requirements*, *detailed design*, *implementation*, *testing*, and *maintenance*, with each step proceeding only when the previous step was completed.

Proponents of such methodologies cite the need for heavy documentation up front because of the disproportionately higher cost of making significant changes late in the development process, as noted by Bell and Thayer (1976). However, agile enthusiasts entertain the notion that, perhaps, this strict adherence to early documentation and linear separation of procedures contributes significantly to the high cost of change later in the project. It may be that following a flawed plan (or one that misunderstood the customer's need, or that documented it so long ago that the customer's needs have since changed) is what makes the rather common, predictable need for change so costly. While older methodologies penalized "scope creep," as customer change requests were known, the agile framework reduces the cost of change through the rapid, iterative, incremental development and frequent feedback, enabling the customer to respond to changes in understanding, in technology, or in the business environment. As Fowler and Highsmith note, "facilitating change is more effective than attempting to prevent it" (2001).

Planning is a critical management function. Good planning is essential to successful programming outputs. It is also essential to respond to change. It is an essential value whenever one has two elements contending for supremacy. To value one over the other is difficult and sometimes counterproductive. Proper response to change enhances your project's success immeasurably.

Respond too much or too little and you quickly lose efficiency. Therefore, the implementation phase requires the next element that was completely lacking in the original Agile Manifesto.

LEADERSHIP, THE MISSING ELEMENT

> "That is, while there is value in the items on the right, we value the items on the left more."
> (Agile Manifesto, 2001)

The problem with the assertion by proponents of agile that the "items on the left" of each of the four agile principle statements (individuals and interactions, working software, customer collaboration, and responding to change) are more valuable than the "items on the right" is that it might be true most of the time, but it simply cannot be true all the time. If there is value in anything on the right (processes and tools, comprehensive documentation, contract negotiation, and following a plan) then there must be times when those values take priority for the team over the canonical agile elements on the left. The authors propose that leadership, an essential function of good management (Deming, 1986), is the variable missing in the original Agile Manifesto as documented.

Leadership is the element that balances these concepts, and has the opportunity to reduce frustration and remove friction between elements. While much of the agile and Scrum literature discusses the impact of the "Highest Paid Person's Opinion", otherwise known as (HiPPO), and the negative impact of group thinking and the creation of nonexistent boundary conditions and expectations (Seelochan, 2015), self-organizing teams within the agile Scrum framework contain leadership in different roles such as Scrum Master and Product Owner.

In Schwaber and Sutherland's *Scrum Guide* (2016), the role of each team member in a Scrum team is clearly outlined. It is understood that the Product Owner serves as the team member assigned to managing the Product Backlog and ordering and creating the lists of tasks to be completed in order to achieve the goals and missions put before the team. This role can be considered a leadership role as they are responsible for presenting the tasks for completion. This, in and of itself, is a function of leadership, however it is not acknowledged as necessary within the original Agile Manifesto. Scrum, while considered one of the most successful implementations of agile, adds the crucial element of leadership back into the self-organizing team dynamic.

Another role that acknowledges the importance of leadership on a Scrum team is that of the Scrum Master. The Scrum Master's role on the team is focused on adherence to the Scrum process, responsibility that all tasks at hand are understood and that no impediments are placed in the way of those on the development team. This role can be viewed as that of a servant leader. In Greenleaf's famous work *The Servant as Leader* (1991), he stated that servant leaders "make sure that other people's highest-priority needs are being served." Scrum Masters serve in this role as many times impediments may include meeting fundamental hierarchical needs in order to ensure tasks may be completed. It is not unusual for a Scrum Master to deliver medication to an ill team member, address conflict within the team, or address resource issues (Overeem, 2016) so that the other team members may focus on the tasks at hand rather than issues that may prevent progress.

Of particular interest in our leadership research for managers in knowledge worker industries is the concept of the Daily Scrum meeting. The Daily Scrum is an intentionally brief (usually 15 minutes), stand-up meeting in which all team members answer three questions:

1. What did you do yesterday?
2. What will you do today?
3. Are there any impediments in your way?

All team members are required to attend the Daily Scrum, all team members must participate, and the Scrum Master is charged with ensuring that the meeting adheres to the Scrum methodology. No cell phone conversations (voice or text) are allowed during the meetings, everyone must arrive on time (with only 15 minutes, punctuality is especially important), stand for the full meeting, and participate meaningfully by answering the three questions. In most cases, the Daily Scrum is held in the same room and at the same time each day, ideally in the morning, as it helps set the context for that day's work. It also serves as a point of accountability, as each team member commits to the work they plan to do each day, and the following day they report on whether they accomplished that work.

One item worth noting is that only team members may speak in the Daily Scrum, but anyone else who wants to hear updates is allowed to attend. This may include department managers, VPs, salespeople, or developers from other projects, but they are only allowed to listen, not to speak or interfere in the meeting, and the Scrum Master is authorized and expected to maintain this rule, even if that means letting a VP know that they may voice concerns after the meeting one-on-one, but that the VP is not allowed to derail the meeting or interfere with the team's progress during the Daily Scrum. This level of employee empowerment, and commensurate employee accountability, is a hallmark of the Scrum implementation of agile principles, and the Daily Scrum meeting facilitates relevant, focused communication and trust among team members and the Scrum Master, especially as the Scrum Master resolves impediments holding team members back from delivering their work products.

The Scrum Master and Product Owner roles are the key leadership components missing from the original Agile Manifesto, and including this type of leadership in an agile framework for the management of modern knowledge workers is a crucial factor for success. Both the Scrum Master and Product Owner are committed, participating members of a team, and the Scrum Master further embodies the characteristic traits of a servant leader, as evidenced above – the Scrum Master keeps everything moving forward both by ensuring that the team and outsiders (including senior management not directly committed to the project) adhere to the Scrum framework, and by removing impediments to maximize their team members' productivity and time-on-task. Further, the Scrum Master must demonstrate agile, high-level adaptive and situational leadership, managing self-organizing teams of independent workers, each of whom has a significant impact on the overall team performance when developing daily work products that build upon one another's teammates' products. This is especially the case in highly knowledge-intensive industries, including the financial and healthcare sectors, an area of particular interest to the authors.

When incorporated in a balanced way, using all experience gained, the leadership roles on an agile team result in a higher success rate when judged against the quality variables in the original manifesto. As an example, let us inspect the concept of the value of "responding to change versus the value of following a plan", the fourth element of the manifesto. It seems a reasonable assertion that both of these elements are important. Most would agree that if you had to choose between the two, responding to change would be more important. However, we also could agree that there are times when following the plan takes precedent over responding to change.

It is precisely at this point that leadership, a concept absent from the original manifesto, becomes the most important element of all. The purpose of a leader on any team would be to identify which of these concepts would dominate at a particular time and space. Therefore, the leader directs and monitors the team as it makes choices between "following the plan" and "responding to change." Thus, the four concepts on the left are more valued, and leadership should recognize that. The four concepts on the right are also valued at times, depending on the situation, and the high-level, situational leader should recognize and respond to that as well. This contingency approach to leadership has a long history in the management literature. The observations above suggest a new manifesto for knowledge workers proposed in the next section.

PROPOSAL: A KNOWLEDGE WORKER MANIFESTO

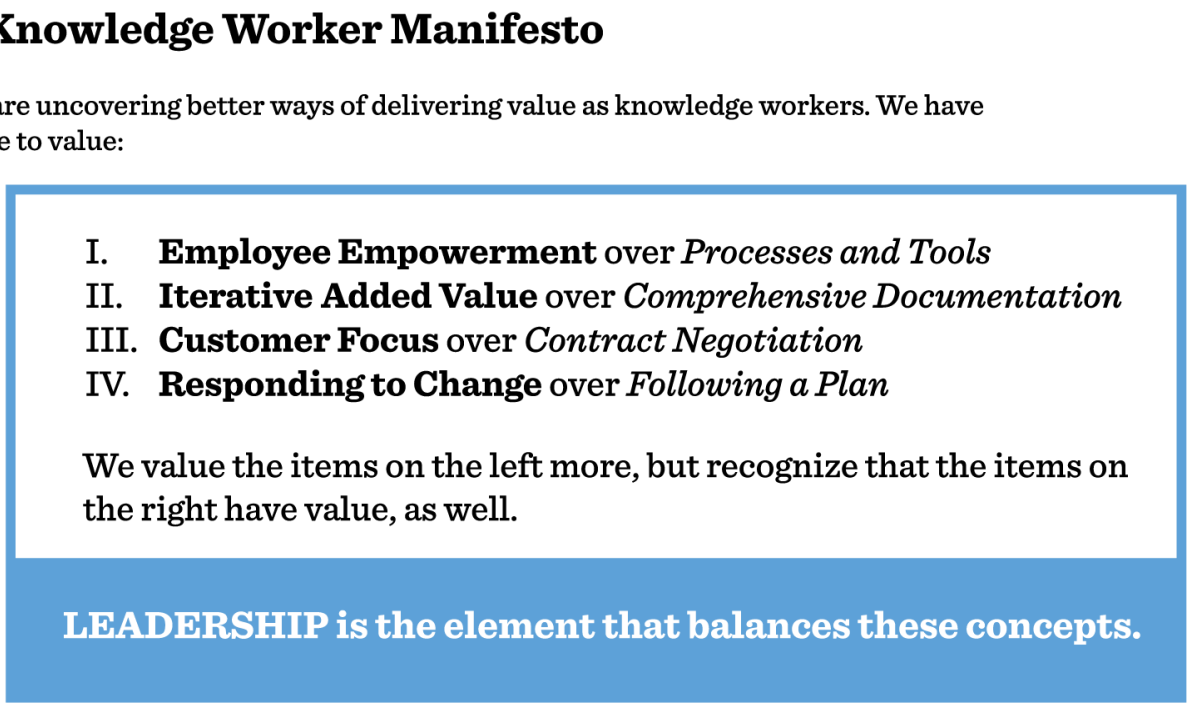

FIGURE 1: A KNOWLEDGE WORKER MANIFESTO

Employee Empowerment over Processes and Tools

Giving knowledge workers more autonomy, specifying what is to be done, not how it is to be done, and emphasizing collaboration are hallmarks of employee empowerment.

Iterative Added Value over Comprehensive Documentation

Reducing waste and focusing on iteratively developing the minimum viable product for each stage of the larger product development cycle leads to better deliverables.

Customer Focus over Contract Negotiation

Customer engagement and interaction are crucial in developing and testing each deliverable, providing valuable feedback and input at each stage in the development of a knowledge product.

Responding to Change over Following a Plan

The customer's needs, the team's capabilities, and the business environment can change rapidly in a knowledge worker environment, and teams and their leaders need the flexibility to respond to those changes.

Leadership: The Crucial Element

Product Owners and Scrum Masters in an agile/Scrum knowledge worker environment provide the critical leadership necessary to focus the team on the deliverables for each day and each sprint cycle, to remove obstacles and impediments that might hinder teammates from accomplishing daily goals, and to interact with both customers and team members to ensure that features are being developed that support the customer's needs, even as those needs evolve.

CONCLUSION

The essence of agile relates to a focus on managerial concepts that have existed for a long time. Specifically, employee empowerment, added value or waste reduction, customer focus, and change management. What makes agile unique is twofold. First is the focus on elements that are clearly important while attempting to identify competing elements that are not as important. Second is the

collection of these concepts together as a manifesto. This collection's value for programmers lies in the synergy among these concepts for those whose task is to use knowledge to create a product or service that serves someone else. The creative, highly educated individual in such organizations more often than not prefers management that minimizes bureaucracy, while giving him or her access to the customer in a time sensitive manner, all the while recognizing the contribution he or she makes.

One aspect of this approach that seems missing is motivational, effective leadership. When two concepts are set against each other as in the manifesto the pressure on leadership is obvious. Much of knowledge work in today's world revolves around multifunctional teams. A parallel exists with software developers where the Agile Manifesto was first developed. The team is responsible to create a product that will meet and exceed the customers' needs. The accomplishing of this task requires the balancing of the agile concepts against one another. For example, in the first concept we find, "individuals and interactions over processes and tools." The assertion is made that both sides of this equation are valued, but the one on the left, "individuals and actions", is valued more. If both sides have value and one side is more valuable than the other it requires leadership to choose. In agile teams, this conundrum must come up again and again as a project moves forward. Clearly, this requires a hands-on style of leadership that inspires and directs, yet without micromanaging, while minimizing the impact of paperwork and bureaucracy.

The Agile Manifesto does that better for teams of software developers and their management than any collection of management concepts to date. However, this group of concepts sometimes brings difficulty in implementation. Our suggestion is that a stronger connection with the literature preceding it, coupled with the addition of leadership as a moderating variable would allow for improved implementation and an extension to all knowledge workers. We find that over the last sixteen years since the publishing of the Agile Manifesto a gradual move into the management literature has occurred, first into project management and then progressing into other knowledge management fields including marketing, real estate, and others. The present work could pave the way for more applications among different knowledge worker groups. Also, we believe that further research may also be needed into the varying roles on a Scrum team and the leadership roles found within all three primary roles: Product Owner, Scrum Master and the team itself.

Programmers see this group of concepts and are immediately enamored with them. However, their experience finds quickly that while almost all programming work is under the auspices of agile, programmers still frequently find themselves bogged down in meetings that do not seem agile at all. Frequently they find themselves hampered with poor implementation despite agile talk. Implementation is the job of management leadership and is obvious when not viewed as part of the solution. One programmer said during an interview about agile, "Every meeting and project that I have is agile, but none are!" Leadership is the missing element allowing for successful implementation of agile techniques and methods.

Academics see this group of concepts and are immediately enamored with them. However, they are frustrated by the use of management concepts without recognizing the roots and the extensive research about how these concepts function. They are also frustrated by the ignoring of leadership that seems to exist with the first Agile Manifesto. The frustrations of both practitioners and academics would be lessened by a new manifesto – A Knowledge Worker Manifesto.

FUTURE WORK

We see particular promise in the practical application of the principles of the proposed Knowledge Worker Manifesto to knowledge workers in financial professions and in the health care industry, as well as other knowledge-intensive fields. Our plan is first to survey professionals in both the financial and health care sector who consider themselves to be knowledge workers to determine the extent to which agile knowledge worker concepts are already being implemented in workplaces in both industries.

The next stage of research would be, based on expected gaps found between the survey results and the agile knowledge worker concepts presented here, to provide training and leadership development workshops for mid-level managers in knowledge-intensive industries, beginning with financial sector and healthcare leaders. Our focus is on mid-level managers due to their direct ability to impact teams of knowledge workers, as well as their immediate ability to change the perceived work environment in these industries. Low-level and mid-level managers that directly interact with teams could benefit from training in agile/Scrum methodologies, and from evaluating which components of agile/Scrum development could be incorporated in their own functional working teams. Of chief interest to our research is the adoption of the Daily Scrum stand-up meeting, with each team participant providing the answers to the three questions: what they accomplished yesterday, what they'll accomplish today, and what impediments they have encountered or anticipate that impact their ability to deliver the work they've committed to do. Above and beyond the Daily Scrum meeting itself, the adaptation by agile leaders to assume the role of a true servant leader in the style of the Scrum Master, removing obstacles and resolving impediments to their team's progress – "clearing a path" for their team to maximize each team member's productivity and time-on-task – will be a key research focus. Particularly, two proposed research questions include: "Does the Daily Scrum meeting activity and the assumed role of a Scrum Master by a knowledge worker team leader or manager positively impact the perceived and actual productivity of the team members?", and, "Do these components of agile knowledge worker management positively impact employee engagement and employee satisfaction in knowledge-intensive organizations?"

In addition to further work in management, quality and operations research, the implications of employee empowerment, iterative added value, customer focus, responding to change, and agile leadership reach could reach far beyond these boundaries – for example, recent cybersecurity research suggests that low employee engagement is a significant factor in predicting insider computer crime (Willison & Warkentin, 2013). So-called lean startups based on iterative added value have spawned an entire subculture in entrepreneurship circles. Customer focus and responding to change are well-developed concepts in business disciplines from management to marketing and beyond, and the term "agile" has begun to be applied more frequently to leadership research in the past eight to ten years. The authors propose that the particular matter of preparing leaders in knowledge-intensive organizations to adopt the agile components of employee empowerment (to include self-organizing teams, and particularly the agile/Scrum practices of the short Daily Scrum stand-up meeting) and iterative added value could be the most impactful, and significant additional work is needed in the application of this approach in knowledge worker management.

REFERENCES

Ambler, S. W. (2014). *2014 Agile Adoption Mini-Survey.* Ambysoft.

Bandow, D., Gerweck, J., & Self, T. B. (2015). Supporting & empowering knowledge workers & communities of practice. *Quarterly Review of Business Disciplines*, *2*(1), 1-17.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001a). *Manifesto for agile software development*. Retrieved from http://www.agilemanifesto.org.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001b). *Principles behind the Agile Manifesto*. Retrieved from http://www.agilemanifesto.org/principles.html

Bell, T. E., & Thayer, T. A. (1976). Software requirements: Are they really a problem? *Proceedings of the 2nd international conference on software engineering*. Los Alamitos, CA: IEEE Computer Society Press (pp. 61-68).

Benny, A. (2007) Web site development process – The life cycle steps. Macronimous Web Solutions. Retrieved from https://www.macronimous.com/resources/web-development-life-cycle.asp

Collyer, S., Warren, C., Hemsley, B. & Stevens, C. (2010). Aim fire, aim – project planning styles in dynamic environments. *Project Management Journal, 41*(4), 108-121.

Conforto, E., Salum, F., Amaral, D., da Silva, S., & Magnanini de Almeida, L. (2014). Can agile project management be adopted by industries other than software development? *Project Management Journal, 45*(3), 21-34.

Coram, M., & Bohner, S. (2005). The impact of agile methods on software project management. *12th IEEE International Conference on Engineering of Computer-Based Systems (ECBS'05).*

Coyle, J. J., Langley, C. J., Novack, R. A., & Gibson, B. J. (2017). *Supply chain management: A logistics perspective* (10th ed.). Boston, MA: Cengage Learning.

Deming, W. E. (1986). *Out of the Crisis*. Cambridge, MA: Massachusetts Institute of Technology.

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development Magazine*. Retrieved from http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf

Gnambs, T. (2015). What makes a computer wiz? Linking personality traits and programming aptitude. *Journal of Research in Personality, 58,* 31-34.

Greenleaf, R. K. (1991). *The Servant as Leader* (Rev. ed.). Indianapolis, IN: Robert K. Greenleaf Center.

Heizer, J., & Render, B. (2016). *Operations management: Sustainability and supply chain management* (12th ed.). Boston, MA: Pearson Publishing.

Hair, J. F. (2007). Knowledge creation in marketing: The role of predictive analytics. *European Business Review*, *19*(4), 303-315.

Hoda, R., Noble, J., & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology 54,* 521-534.

Larman, C., & Basili, V. (2003). Iterative and incremental development: A brief history. *Computer, 36*(6), 47-56.

Larson, R., & Larson, E. (2009). Top five causes of scope creep … and what to do about them. Paper presented at the meeting of PMI® Global Congress 2009 – North America, Orlando, FL.

Layton, M. (2012). *Agile project management for dummies* (1st ed.). Hoboken, NJ: John Wiley & Sons.

Lindborg, H. (2013, June). Stake your ground: Unearthing the origins of stakeholder management. *Quality Progress*. Retrieved from http://asq.org/quality-progress/2013/06/career-corner/stake-your-ground.html

Madhuri, L. K., Rao, J. J., & V, S. (2014, November). Effect of scope creep in software projects – its bearing on critical success factors. *International Journal of Computer Applications, 106*(2), 9-13.

Newkirk, J. W., & Martin, R. C. (2001). *Extreme programming in practice*. Reading, MA: Addison-Wesley.

Overeem, B. (2016, March 26). The Scrum Master as an impediment remover. *Barry Overeem – The Learning Facilitator*. Retrieved from http://www.barryovereem.com/the-scrum-master-as-an-impediment-remover/

Payne, B. R., & Stretch, K. (2016). Software testing within the framework of the agile software development method (abstract). *Proceedings of the 2016 International Academy of Business Disciplines Annual Conference*.

Power, B. (2011, April 21). Customer-centric continuous improvement. *Harvard Business Review*. Retrieved from https://hbr.org/2011/04/continuously-improving-custome

Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. New York: Crown Business.

Robinson, F. (2001). Minimum viable product (SyncDev white paper). Retrieved from http://www.syncdev.com/minimum-viable-product/

Schwaber, K., & Sutherland, J. (2016). The Scrum guide: The definitive guide to Scrum. Retrieved from https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

Seelochan, S. (2015, June 7). Agile: Beware of the hippo in the room. Retrieved from http://www.microlise.com/blog/agile-beware-the-hippo-in-the-room/

Serrador, P., & Pinto, J. K. (2015). Does Agile work? A quantitative analysis of agile project success. *International Journal of Project Management, 33*(5), 1040-1051.

Tennant, G. (2001). *Six Sigma: SPC and TQM in manufacturing and services*. Farnham, UK: Gower Publishing, Ltd.

Willison, R., & Warkentin, M. (2013). Beyond deterrence: An expanded view of employee computer abuse. *MIS Quarterly 37*(1), p 1-20.

APPENDIX A: MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

Agilemanifesto.org: Reprinted with Permission (Creative Commons Attribution)

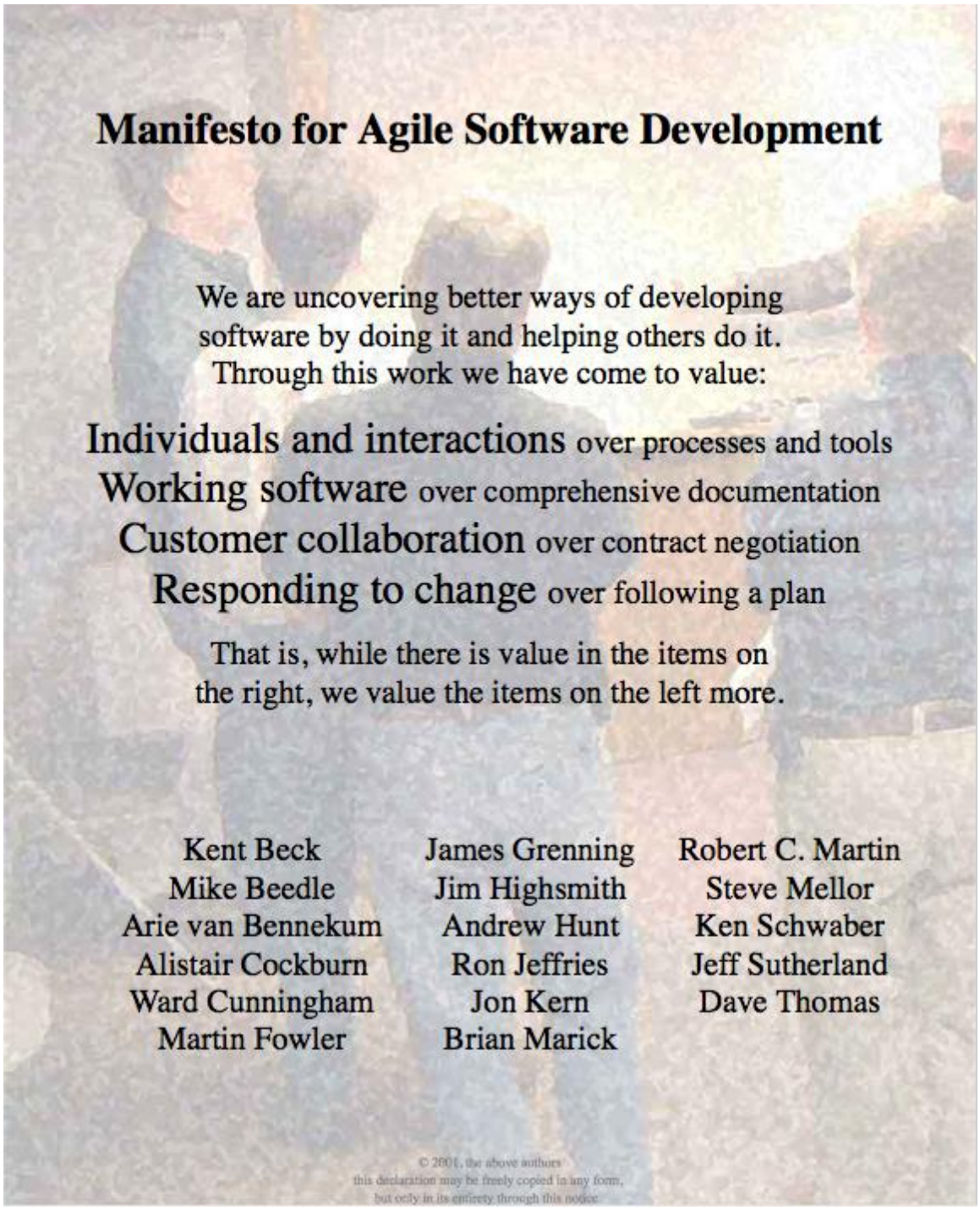

FIGURE 2: MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT (2001)

INTERNATIONAL JOURNAL OF INTERDISCIPLINARY RESEARCH

VOLUME 6, NUMBER 2, December 2017 ISSN 2165-3240

A PUBLICATION OF EASTERN WASHINGTON UNIVERSITY AND THE INTERNATIONAL ACADEMY OF BUSINESS DISCIPLINES

WWW.IJIR.NET