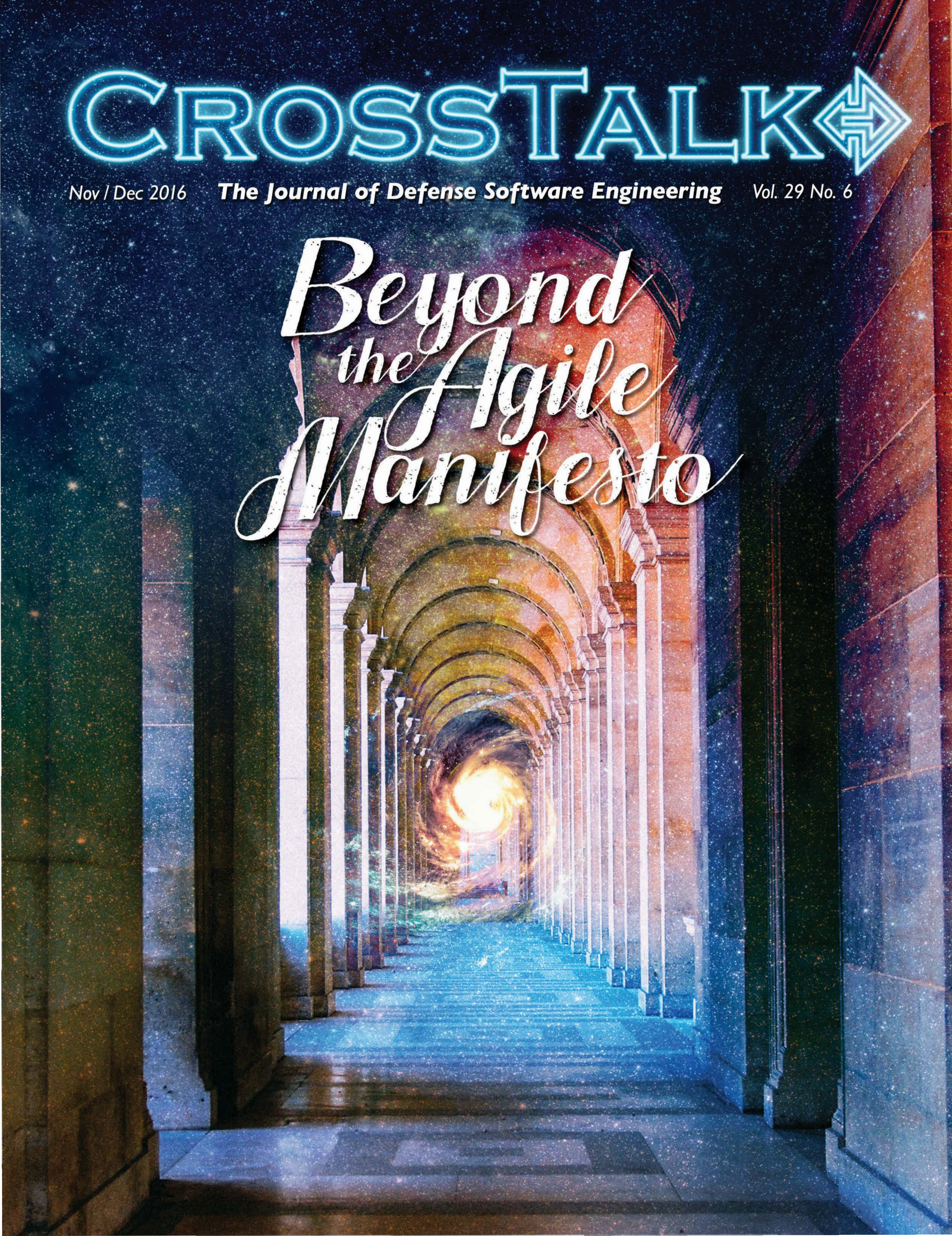
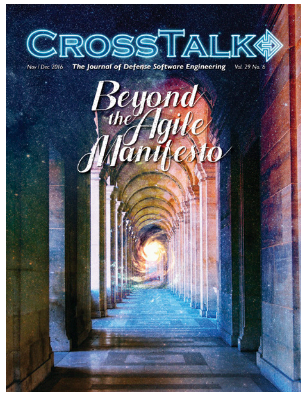
CrossTalk

Nov / Dec 2016 The Journal of Defense Software Engineering Vol. 29 No. 6

Beyond the Agile Manifesto





Cover Design by
Kent Bingham

Departments

Beyond the Agile Manifesto

CrossTalk

NAVAIR Jeff Schwalb
DHS Peter Fonash
309 SMXG Kelly Capener
76 SMXG Mike Jennings

Publisher Justin T. Hill
Article Coordinator Heather Giacalone
Managing Director David Erickson
Technical Program Lead Thayne M. Hill
Managing Editor Mary Harper
Copy Editor Breanna Olaveson
Senior Art Director Kevin Kiernan
Art Director Mary Harper

Phone 801-777-9828
E-mail Crosstalk.Articles@hill.af.mil
CrossTalk Online www.crosstalkonline.org

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). USN co-sponsor: Naval Air Systems Command. USAF co-sponsors: Ogden-ALC 309 SMXG and Tinker-ALC 76 SMXG. DHS co-sponsor: Office of Cybersecurity and Communications in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

Subscriptions: Visit <www.crosstalkonline.org/subscribe> to receive an e-mail notification when each new issue is published online or to subscribe to an RSS notification feed.

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.crosstalkonline.org/submission-guidelines>. CrossTalk does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the authors and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: CrossTalk is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk Online Services:
For questions or concerns about crosstalkonline.org web content or functionality contact the CrossTalk webmaster at 801-417-3000 or webmaster@luminpublishing.com.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

CrossTalk is published six times a year by the U.S. Air Force STSC in concert with Lumin Publishing <luminpublishing.com>.
ISSN 2160-1577 (print); ISSN 2160-1593 (online)

CrossTalk would like to thank NAVAIR for sponsoring this issue.

Beyond the Agile Manifesto

Agile software development describes a set of principles for software development under which products evolve through the collaboration effort of cross-functional teams.[1] It advocates planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.[2] These principles support the definition and continuing evolution of many software development methods. [3]

This culminated in what we know today as the Agile Manifesto that was developed in February 2001. In it four values are presented that state the following tradeoffs 1) individuals and interactions over processes and tools, 2) working software over comprehensive documentation, 3) customer collaboration over contract negotiation, and 4) responding to change over following a plan. This was later followed with 12 principles describing "what" the Agile tradeoffs meant to a working project. The Manifesto contains no direction on how to carry out and achieve these goals. These methods in fact already existed at the time of the Agile Manifesto's inception and already provided a set of choices the answer this "how" question. These approaches include:

- from 1991 rapid application development
- from 1994 unified process and dynamic systems development method (DSDM)
- from 1995 Scrum and Personal Software Process (PSP)
- from 1996 Crystal Clear and extreme programming (XP)
- from 1997 feature-driven development
- from 1998 Team Software Process (TSP)

When a project or an organization simply says they are Agile, that claim is ambiguous until they also state the method being applied. This issue of CrossTalk is aimed at encouraging the community to share stories, data, and experience using these methods regarding the delivery of **high quality products on cost and schedule**. Furthermore, this is also an opportunity for the community to share how various method performs when carried out as prescribed or as tailored. My hope is that it will be an ongoing endeavor in sharing throughout our community of practice.

Jeff Schwalb
NAVAIR Process Resource Team

REFERENCES

1. Collier, Ken W. (2011). Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing. Pearson Education. pp.121 ff. ISBN 9780321669544. "What is a self-organizing team?"
2. "What is Agile Software Development" Agile Alliance. 8 June 2013.
3. Larman, Craig (2004). Agile and Iterative Development: A Managers Guide. Addison-Wesley. p.27 ISBN 978-0-13-111155-4

The Heart of Agile

Alistair Cockburn, Humans and Technology, Inc.
Humans and Technology Technical Report 2016.02

Abstract. Agile has become overly decorated. The remedy is simple: collaborate, deliver, reflect, improve. These four imperatives, already sufficient, expand to cover the complexities of modern development.

Introduction

The Manifesto for Agile Software Development [1] was written in a particularly simple style. It has become apparent to several authors of the manifesto that Agile practice has become decorated to the point of contradicting its roots (see, for example, "Stop Practicing and Start Growing" [2]). This article describes my approach to getting Agile back on track while at the same time moving it forward into the future.

We recover the simplicity and power of Agile by recognizing that it can be expressed in four words:

- Collaborate.
- Deliver.
- Reflect.
- Improve.

These four words are sufficient, simple, and still support the complexities of modern Agile development. For those reasons, I call them the "kokoro," or heart of Agile.

Kokoro Simplifies

In rebuilding Agile from its center, I wanted to honor a minor tradition of looking at Japanese words for skills development. In 1999, my attention was drawn to the concepts of "shu," "ha," and "ri," (守 破 離) which date back to 14th-century Japanese Noh theater. [3, 4]

"Shu" (守) roughly translates to "follow." It captures the stage of learning in which the novice learns by copying a master or a recipe. In general knowledge acquisition terms, "shu" is the starting stage — "learn one technique."

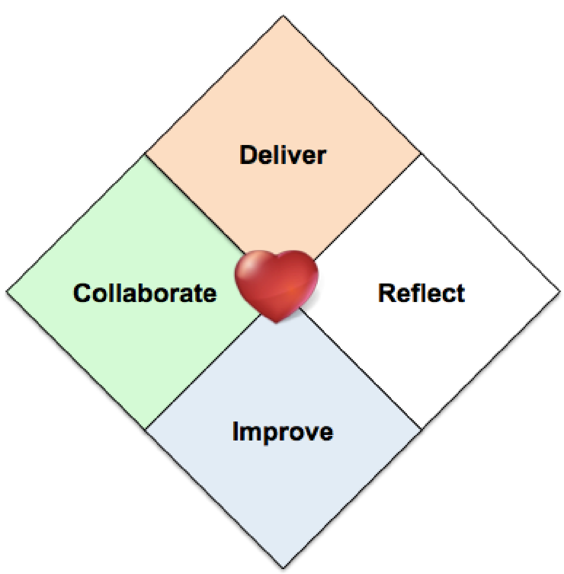


Figure 1. The Heart of Agile

"Ha" (破) roughly translates to "detach." It captures the next stage of learning in which the person learns different tools and techniques, either out of curiosity or by reaching the boundaries of the techniques he or she already knows. "Ha" can be thought of as the learning stage — "collect techniques."

"Ri" (離) roughly translates to "leave." It captures the stage of practice in which the person operates by whole-body response to ever-changing situations, doing something different every time. Ri-level people generally cannot say how they decide on a technique at the moment because it is so ingrained and immediate. In general knowledge acquisition terms, "ri" corresponds to "invent and blend techniques."

In looking for what could come after "ri," I noticed that advanced masters advocate a return to essence and radical simplicity. (Think of Mr. Miyagi saying "Wax on, wax off," in "The Karate Kid." [5]) The Japanese "kokoro," (心) "essence" or "heart," is used in the writings of the 17th-century samurai master Miyamoto Musashi to refer to the essence or heart of the samurai. In other words, "kokoro" (心) is perfect for our needs: the radically simplified essence of a skill area. "Kokoro" represents the teaching stage of the advanced practitioner. It is characterized by the advice, "Just learn the basics."

Figure 2 captures the "shu-ha-ri-kokoro" progression. It shows how practice starts off simple ("shu," learn one technique), becomes more complicated as one learns more techniques ("ha," collect), becomes significantly more complicated at the "ri" level (invent and blend), and finally takes on a simple form again when practiced by the advanced teacher.

You can probably find examples in your own life of a "kokoro"-level teacher telling you, "Just master the basics." That is what we are seeking for Agile development.

The "kokoro," or heart of Agile, is to collaborate, deliver, reflect, and improve — nothing more. I express the heart of Agile with the diamond shown in Figure 1.

The nice thing about these four words is that they don't need much explanation or teaching. With the exception of "reflect," which is seldom done, most people understand these terms well. You know if you're doing them or not.

The Heart Expands

Although the four verbs simply state most of what you need to do, each also suggests a deeper, subtler execution. There is a beginner version of each, and there are competing techniques to improve each. The "shu-ha-ri" concept of skill progression applies to each of the four, and to each of the sub-categories under them.

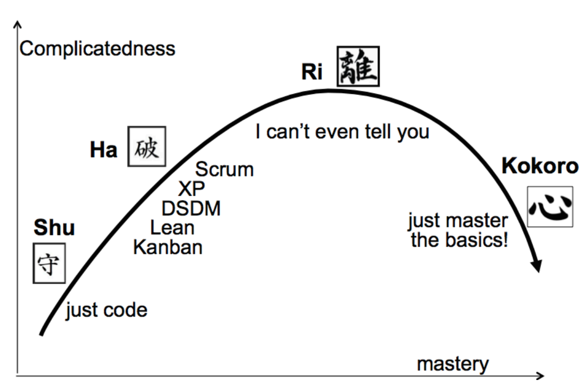


Figure 2. The "Shu-Ha-Ri-Kokoro" progression.

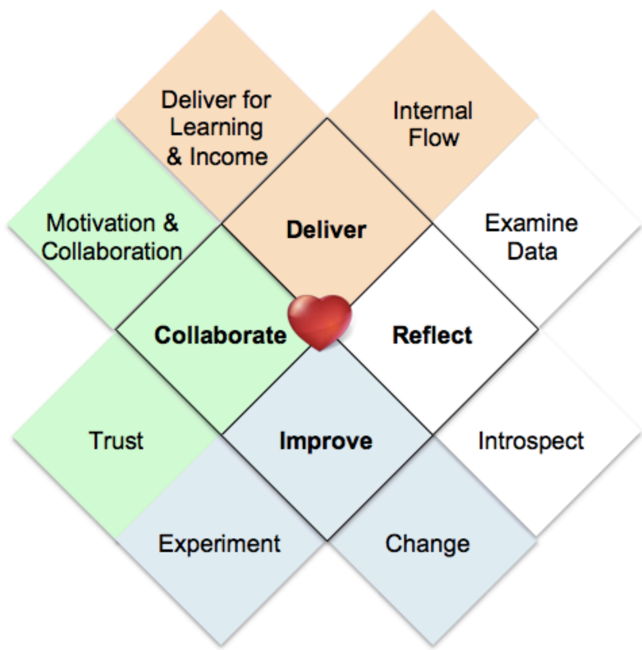


Figure 3. A First-Level Expansion of the Heart of Agile.

Let us look at an expansion to see how we get to modern Agile development. Figures 3 and 4 show two levels of expansion. Other expansions are possible, as we will see.

Collaborate

To collaborate, we want to improve trust, motivation, and the act of collaboration. These are shown in the first-level expansion of Figure 3.

As you might imagine, trust is an enormous topic. A search on Amazon returns over 91,000 book titles on trust. There are survey instruments, improvement programs, institutes and consultancies. Motivation is similarly rich, breaking into intrinsic and external motivation, including power, rewards and politics (see Figure 4). In other words, "collaborate," while easily understood, supports a deep expansion.

To illustrate the "shu"-level expansion of the heart of Agile, I highlight just one way to improve collaboration. Figure 5 shows a fragment of a card set [6] being used in organizations and fields from facilitation training to town management. They are taken from the CrossTalk article "Increasing Collaboration by the Minute." [7] They provide one technique to sensitize people to what helps and hinders collaboration.

Deliver

Delivery has internal and external aspects. In the internal portion we find incremental development, lean manufacturing, queue management, bottlenecks, work-in-progress limits, Kanban, and technology and social processes in the delivery pipeline. In the external portion we find the issues of delivery for learning versus delivery for revenue.

Delivering incrementally, early and often is well understood. [8] Less understood is the idea of delivering just to learn: to learn what market niche a product should address and with what features, [9] and also to learn how to work together, what design assumptions were incorrect, and how long the effort will take. [10]

Reflect and Improve

"Reflect" and "improve" are closely related. They are separated because reflection is so rarely done well. I wish to highlight the need to explicitly stop and examine what is happening before jumping to improvement initiatives.

Reflection breaks into two parts: gathering both subjective, emotional information, usually about the team and the process, and objective information from data analytics about the product and its reception by users and buyers.

Inside the "improve" arena, modern practitioners are studying what is called "solutions focused coaching" [11] to incorporate state-of-the-art techniques in psychotherapy and family coaching, compatible with Agile development.

The Heart at Scale

The current proposed agile scaling methods work from structure: set up scrums of scrums, backlogs of backlogs, multiple levels of product owners, Kanban boards at a high level, and so on. Having the heart of Agile in hand, we see that this changing structure does not yet address attitude or behavior, which are what we want to change. The heart of Agile addresses attitudes and behavior directly.

No matter the size of the organization improving collaboration should advance the situation, and similarly for improving delivery. Improving reflection and improvement accelerates the first two. In other words, rather than relabeling the job titles of workers or introducing new responsibilities, ask everyone the following questions:

- Independent of anything else going on, how will you increase collaboration?
- Accounting for everything else going on, how will you increase trial and actual deliveries to consumers?
- How will you get people to pause and reflect on what's happening to and around them?
- What experiments will your people perform at different levels in the organization to make small improvements?

People can't hide behind vocabulary or job title shuffles to answer these questions. There is nothing but attitude and behavior to improve, which is what we want.

Scaling agile is a difficult topic at the best of times. The most difficult issue might be the conflicting reward schemes across the organization. The heart of agile identifies but does not address this difficult subject.

Getting Started

How would you get started on a program to implement the heart of Agile approach in your company?

1. Ask everyone to list all the people they collaborate with to get their work to a customer or client. For each person they name, ask them to rate the quality of collaboration with that person now and identify what they might do to improve it. This gives each person an action item and produces a social graph, revealing where to start.

2. Examine the size of the increments being developed and the time needed to release each. Train both business and development on how to make those slices finer. Solicit ideas to streamline the delivery pipeline. Learn to deliver for learning, not just for revenue.

3. Stop and reflect. Let people say what social and technology changes might improve their work. Examine product usage

analytics to divine what is really happening on the user side. Run an experiment every month.

4. Publish a newsletter showcasing all the things going on, including what people are doing and what projects are starting. Make progress visible so both workers and executives see that the organization is moving.

Summary

The heart of Agile doesn't remove the complexities of daily life; it only acts a reminder to clear them away for a moment and focus on the basics:

- Collaborate.
- Deliver.
- Reflect.
- Improve.

These four words — the "kokoro," essence, or heart of Agile development — are simple, sufficient, and expandable into useable advice at the forefront of the modern Agile development.

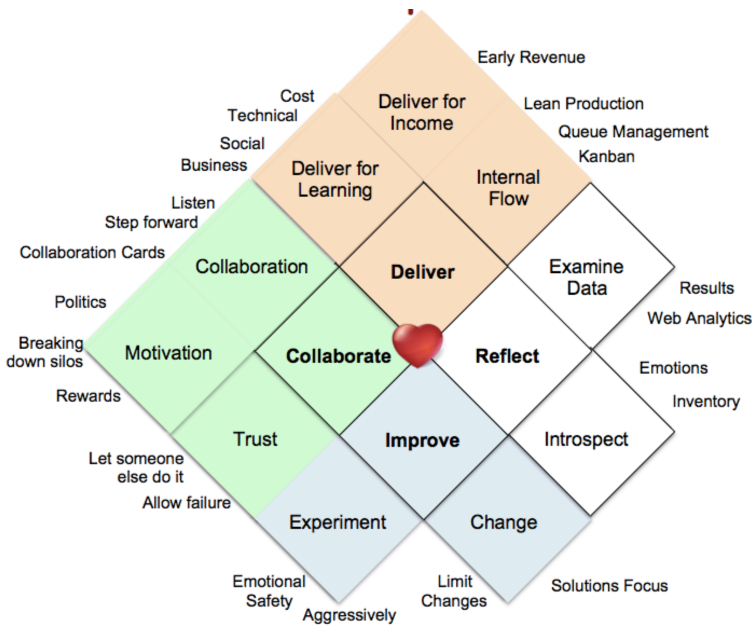


Figure 4. A Second-Level Expansion of the Heart of Agile.

REFERENCES

1. http://agilemanifesto.org.
2. Hunt, A. (2016.) Stop Practicing and Start Growing. http://growsmethod.com/articles/stop_practicing_and_start_growing.html .
3. ShuHaRi, https://en.wikipedia.org/wiki/Shuhari.
4. Shu Ha Ri, http://alistair.cockburn.us/Shu+Ha+Ri.
5. "Karate Kid, 'wax on, wax off' scene." https://www.youtube.com/watch?v=fULNUrOrvEc.
6. http://alistair.cockburn.us/Collaboration+Cards.
7. Cockburn, A. (Jan./Feb. 2016.) Increasing Collaboration by the Minute. CrossTalk, 4-7. Online at http://static1.1.sqspcdn.com/static/f/702523/26767147/1451886700677/201601-Cockburn.pdf?token=oTGZ9syVsnh4d%2BtW8ggVolCglEM%3D
8. Denne, M. & Cleland-Huang, J. (2003.) Software By Numbers. Prentice-Hall.
9. Ries, E. (2011.) The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business.
10. Cockburn, A. (2014, July/Aug.) Disciplined Learning: The Successor to Risk Management. CrossTalk, 15-18. Online at http://static1.1.sqspcdn.com/static/f/702523/25136916/1404242669373/201407-Cockburn.pdf?token=hkVdzBUOIepDbyggvqTqlyOcmHA%3D.
11. Iveson, C., George, E. & Ratner, H. (2012.) Brief Coaching: A Solution Focused Approach. Routledge.

ABOUT THE AUTHOR

Dr. Alistair Cockburn, one of the creators of the Manifesto for Agile Software Development, was voted one of "The All-Time Top 150 i-Technology Heroes" in 2007 for his pioneering work in use cases and Agile software development. A renowned IT strategist and author of the Jolt award-winning books "Agile Software Development" and "Writing Effective Use Cases," he is an expert on Agile development, use cases, process design, project management and object-oriented design. In 2001 he co-authored the Agile Manifesto, in 2003 he created the Agile Development Conference, in 2005 he co-founded the Agile Project Leadership Network, and in 2010 he co-founded the International Consortium for Agile. Many of his articles, talks, poems and blog posts are online at http://alistair.cockburn.us.

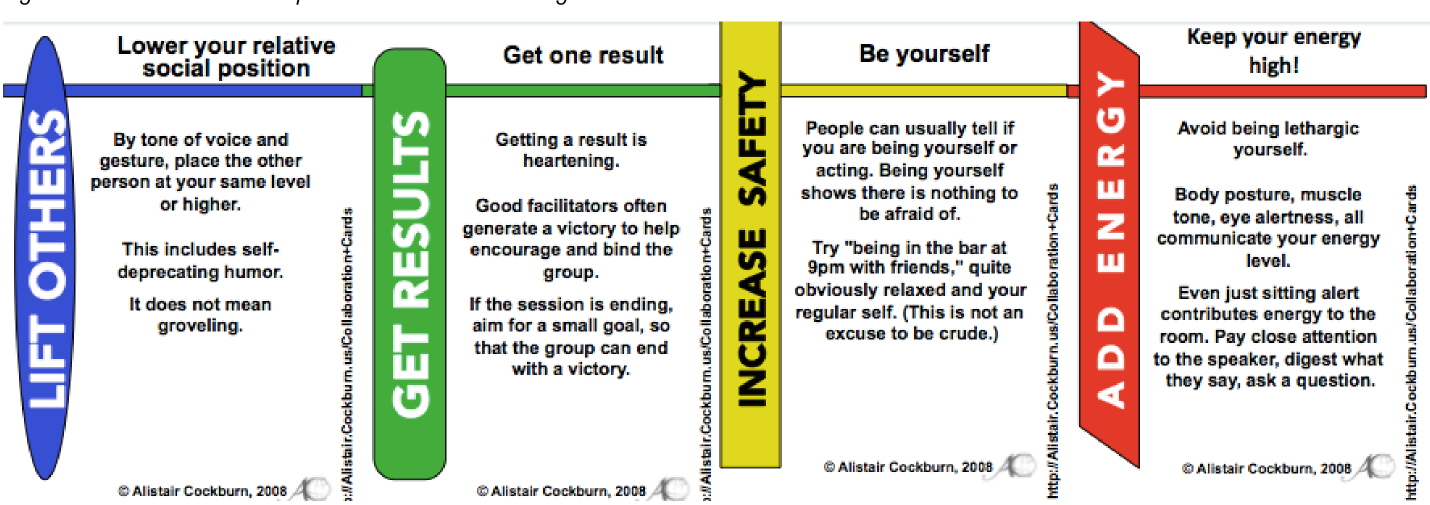


Figure 5. A "Shu"-Level Tool for Improving Collaboration

From the Trenches

Improving the Scrum Daily Standup Meeting

Dick Carlson

Abstract. Among the most important work sessions in "scrum" is the scrum daily standup meeting. This meeting, which must be held every working day in order to be effective, is critical for team members to communicate their work commitments to each other.

The most significant reasons why daily standup meetings are not effective become apparent after years of working with "Agile" teams, but ranking these reasons is difficult. In order to do so, one can rely on personal experiences. During the many years I have spent "in the trenches" facilitating and coaching teams, I have observed that most daily standup problems are caused by problems with either individual team members, project teams, or management.

This article addresses several examples of unfavorable daily standup scenarios that exemplify all three of these causes. It is assumed that readers are familiar with the daily standup process; therefore, steps on how to conduct a standup meeting are not discussed in this article.

Not Holding the Daily Standup Every Workday

Scenario: As an Agile coach and transformational authority, I am often asked to help problematic teams. The first question I ask a problematic team's scrum master is "Does the team hold daily standup meetings?" If the answer is yes, I follow up with a seemingly unnecessary question: "How often does the team run the daily standup?" They usually say that they hold them two or three times a week. This answer helps me to identify the root cause of their problems. I immediately think, "What part of the word 'daily' does the team not understand?"

Solution: The team must know that the daily standup is a short meeting that does not hinder productivity. Rather, the meeting increases project transparency so that everyone knows what the team's sprint goals are and whether the team will be able to meet those goals. In terms of productivity, this means that duplicate work and rework are avoided. It is also a meeting that is owned and run only by the team. Each team member shares his or her work commitments with the entire team openly and honestly. As the team begins to understand the real value of the standup meeting, they eventually adapt and make it an essential part of every workday.

Poor Daily Standup Attendance

Scenario: I have also observed how a lack of team member participation in the daily standup causes a decay of the scrum process. When people do not show up for the daily standup, the team's productivity suffers. This happens frequently, and it is one of the main reasons why teams struggle and often fail. When people do not communicate with each other on a daily basis, things go awry. Daily standup meetings are often poorly attended due to lack of interest.

Solution: When training project teams, I remind team members that showing up late to daily standup meetings or not attending at all is disrespectful to their teammates. In scrum, the daily standup meeting is a work session to which every team member must commit; otherwise, the Agile and scrum approach begins to fail. It is imperative that the entire team attends this important daily meeting so that every team member is well-informed on what other team members have completed and what their plans are going forward. The team also learns about issues and obstacles that may affect their sprint goals and those that have impacted their work in progress.

When a team loses interest in the daily standup, it's usually because team members are ignoring the agreed-to process or the scrum master is failing to detect a uninspired team. Enter the Agile coach! I have made many on-the-spot corrections through counseling and additional training. When standup practices need to be changed, I encourage teams to make those decisions as problems are detected and to resolve problems during sprint retrospectives. This activity supports process improvement of the daily standup.

Not Starting the Daily Standup On Time

Scenario: I have arrived at many daily standup meetings early only to realize that I was the only one there. In many instances, I watched as several members of the team showed up 10 to 15 minutes after the scheduled meeting time. This roadblock is often caused by insufficient scrum training, team members who do not realize the value of the meeting, or inadequate meeting facilitation. In a recent survey by Version One [1], the daily standup was found to be the most widely practiced Agile technique implemented at 83 percent, followed closely by short iterations and prioritized backlogs, at 82 and 79 percent respectively.

Solution: Daily standup meetings often do not start on time because team members believe that the work they are doing is more important than the meeting. This presents an opportunity for the scrum master to remind, through coaching, those who arrive late or not at all of the importance of the daily standup. The scrum master's counseling will eventually change such behavior and help prevent communication and information loss among team members. As a gesture of fun, I invite team members to vote on whether those who arrive late, regardless of the reason, "donate" one dollar to the team's pizza fund. Most people, if not all, agree to the one-dollar fine. It's an inexpensive correction action that adds fun to the activity.

The Daily Standup Lasts Longer Than It Should (Time Boxing Ignored)

Scenario: It is common to see project teams with one or two members who like to indulge in lengthy discussions during daily standup meetings. Examples of unnecessary topics include how a person solved a problem, the wonderful experience another person and his buddies had on their whitewater rafting trip over the weekend, or the nauseating details of how a team member found major design flaws and made corrections. Such lengthy discussions are commonly referred to as "bunny trails."

Solution: As with all scrum work sessions, the daily standup is time boxed. This means that the meeting begins and ends on time, as scheduled. Some people like to talk excessively without regard or respect for other team members who prefer to hear only what other members are doing, who needs help, and more about completed sprint tasks. This is the main reason why the scrum master facilitates these meetings. He or she is not there to run daily standup meetings — that's the team's responsibility. However, especially with new teams, the scrum master attends the meetings to ensure that everyone is following the agreed-to daily standup process and to prevent unnecessary and needless conversation. Discussions that involve solutions should be deferred until after the daily standup meeting has ended. This way, those not affected by or interested in the resolution discussion are not held up discussing a topic that does not affect their work.

Solving Impediments and Other Problems During the Daily Standup

Scenario: Many new teams attempt to solve impediments and other problems during the daily standup. There are times when the problems are so significant that even the scrum master gets pulled into the turmoil of resolving them. These situations can easily distract from the main goal of the meeting, thereby changing the focus of the daily standup and sending the meeting into a chaotic whirlwind.

Solution: Should such problems be resolved on the spot? No. Is it wrong to settle a very bad situation during the daily standup? Yes. Is it wise to ignore the threat when it is identified? No. As obvious as these answers sound, they are often misunderstood. Solving problems during the daily standup is one reason why the daily standup meeting is ignored, branded as "unproductive," or considered a waste of time. New teams are frequently plagued by these situations because they are not yet disciplined or because their scrum master loses focus in the pandemonium. There are ways to solve problems, but they require control and order.

Sure, the problem being discussed may be major and may cause catastrophic results if not corrected in a timely manner, but panic won't help. Should such problems be resolved on the spot? No. When teams become mature and begin working together well, they know what must be done. First, team members should take note of the impediment or problem, then plan a side meeting immediately following the daily standup with individuals who can help with the resolution. Then the team can continue with the daily standup meeting until everyone has communicated his or her situation. The daily standup is a very short and strategic meeting. It would be unwise to stop the team's communication and continuity because of a perceived problem. It is unwise to ignore a threat when it is identified, but unless the problem is an immediate threat to the lives of people, take note of the problem, defer the resolution for a few minutes, and continue with the meeting.

Impediments Are Not Identified or Defined Well During the Daily Standup

Scenario: When I started coaching teams many years ago, it never occurred to me that some people would be afraid to report issues and problems, but they are. The reasons for this vary but are typically either related to fear that the problem is not as severe as the person believes or fear of being labeled a tattletale. Other instances of this occur during a daily standup when a team member fails to mention an impediment. In both situations, failing to report or identify anything that could cause the team discomfort, inconvenience, or a waste of time is far worse than reporting it. The problem could impact the entire team.

Solution: When I teach students the importance of holding the daily standup every day, I emphasize impediment identification and resolution. New team members have to learn how to be open, honest, and forthright. Potential issues, problems, and anything else that might impede team progress must be reported as quickly as possible — either to the scrum master, product owner, project manager, or team members — to ensure prompt resolution. Fear of reprisals, blame, shame, or uncertainty should never become a part of impediment identification and removal.

Too Many Disruptions During the Daily Standup Meeting

Scenario: There are many reasons why daily standup meetings are disrupted. Here are three example situations most can relate to, but there are many more:

1. You arrive at work and walk over to the "scrum room" to attend the daily standup only to see a manager lecturing the team on something that has nothing to do with the sprint's goals.

2. During the daily standup meeting, a senior manager from engineering walks in unannounced and begins a discourse on architectural patterns.

3. While team members are communicating their progress, a few sideliners strike up a conversation that disrupts the flow of the meeting.

Solutions: These examples are common interruptions that occur during daily standup meetings. In all examples, the scrum master is responsible for immediately stopping the disruptions. Although the team owns the daily standup meeting, the scrum master is responsible for facilitating the meeting and protecting the team from unrelated project activities by keeping the team focused on sprint-related work, ensuring the team follows the agreed-to standup process, and ensuring the team is fully functional and productive.

Management Dictates Daily Standup Meetings

Scenario: An organization adopts Agile and right away, management steps in with their "preferred" execution strategy. The manager may have attended a class or a briefing on Agile and believes he or she understands the process well enough to dictate how it should be implemented, beginning with how to run the daily standup meeting. The manager insists on attending the meetings and determines where and when they will be conducted. Management does not show up on time to the meetings, and when they do show up, they interrupt the conduct of the meeting by announcing trivial content unrelated to the project and the team. If that isn't enough, management frequently interrupts the meeting while team members are attempting to successfully conduct the meeting according to the agreed-upon scrum practice.

Solution: The most effective solution to this problem is special management training and coaching. Management must understand the benefits of Agile methods. They must also allow the team to conduct and manage standup meetings for the benefit of the team and to ensure efficient and productive product development. Organizational despots who dictate the nature of daily standup meetings guarantee the failure of the project's Agile execution. Since the team uses the meeting to communicate project commitment progress, the team should be able to manage the conduct of the meeting to ensure effective communication and avoid wasted time.

Kanban (Project Task Boards) Are Not in View During the Daily Standup

Scenario: The Kanban, or project task board, is commonly used by scrum teams to show and track sprint work derived from selected product backlog items (PBIs). It is the team's responsibility to understand the scope of effort for PBIs selected for each sprint. During the daily standup meeting, the task board is a useful tool in determining work in progress.

However, many teams — both new and experienced — conduct their daily standup meetings in locations away from the task board, which reduces project transparency and team effectiveness. This prevents team members, scrum masters, and onlookers from seeing which tasks are problematic and which team members have too much work in progress.

Solution: The task board is a dynamic tool used by the team to track sprint progress and identify issues and problems. Therefore, the most effective way to promote team effectiveness and project transparency is by establishing a designated team or project room at the beginning of the project. Such a room, often regarded as the "Agile" or "scrum" room, defines a specific and centralized location where the team can work in a highly collaborative environment. The room should not be secluded or isolated from the mainstream organization. Rather, it would be prudent if the room were conveniently located within the pulse of the project to ensure that anyone interested in the project could benefit from ongoing project execution activities.

Statistical Evidence

To quantify the reasons for using the daily stand-up and other Agile methods, key metrics have been included that validate the examples in this article. These metrics are based on a recent survey conducted in 2015 by VersionOne the 10th Annual State of Agile Survey.

The survey makes it clear that Agile software development has grown increasingly popular over the last decade. Participation in the survey has grown more than three-fold. In 2006, there were fewer than a thousand respondents to the survey, while the latest survey has 3,880 respondents.

Barriers to Adoption and Success

While adoption of Agile is increasing, there are still obstacles to overcome. The key barriers to further adoption usually hinge around culture, including the ability to change, general resistance to change, and management support. Interestingly, the majority of respondents pointed toward the company's culture as the reason for failed Agile projects as well. Once these barriers are overcome, the limiting factor most often cited has been availability of personnel with the necessary Agile experience.

Top 3 Tips for Success with Scaling Agile

Now that momentum around scaling agile is growing, what are the key factors for success? The respondents said the top three tips for successfully scaling agile are:

1. Consistent process and practices (43%),
2. Implementation of a common tool across teams (40%), and
3. Agile consultants or trainers (40%).

Size of Organization

Percentage of respondents who worked for organizations with:

1. Fewer than 1,000 people: 44%
2. Between 1,001 and 5,000 people: 17%
3. Between 5,001-20,000 people: 15%
4. More than 20,000 people: 24%

Size of Software Organization

1. Fewer than 100 people: 38%
2. Between 101-1,000 people: 31%
3. Between 1,001-5,000 people: 15%
4. More than 5,000 people: 16%

Benefits of Agile

Top 3 Benefits of Agile

1. Ability to manage changing priorities: 87%
2. Increased team productivity: 85%
3. Improved project visibility: 84%

Top 5 Agile Techniques Employed

1. Daily Stand-up: 83%
2. Prioritized Backlogs: 82%
3. Short Iterations: 79%
4. Retrospectives: 74%
5. Iteration Planning: 69%

Success Metrics

Top 3 leading causes of failed Agile projects

1. Company philosophy or culture at odds with core agile values
2. Lack of experience with agile methods
3. Lack of Management support

Top 3 barriers to further Agile adoption

1. Ability to change organizational culture
2. General organizational resistance to change
3. Pre-existing rigid/waterfall framework

Top 3 Project Management tool used and preferred

1. Taskboard: 82%
2. Bug tracker: 80%
3. Spreadsheet: 74%

Conclusion

The various scenarios discussed in this article are real. The solutions applied to the problems are actions and techniques I have used to resolve the problems. If these problems are not resolved, a significant reduction in team productivity will certainly result, which may have an adverse effect on overall project progress. Reclaiming team progress requires your own personal problem resolution action.

There are many other situations that you may have experienced that are not mentioned in this article. I urge readers to share their own experiences by identifying problematic situations and how they were resolved so that others can become better practitioners of Agile.

REFERENCES

1. 9th Annual State of Agile Survey, VersionOne, 2015.

Schwaber, Ken and Beedle, Mike. (2001.) Agile Software Development with Scrum.
Cohn, Mike. (2012.) Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley Signature Series.

ABOUT THE AUTHOR

Dick Carlson has a B.S. degree in business management and is certified as a scrum professional, scrum master, scrum product owner, and in Lean-Agile project management. He has shared successful experiences of agile, lean, and scrum implementations at conferences, workshops and symposia. Dick's engineering career spans 50 years, and he has taught courses in mathematics, electronics, CMMI, configuration and data management, agile, lean, and scrum for more than 30 years.

The ORDERED Process for Improving Agile Engineering Outcomes

Brian P. Gallagher, Senior Vice President, Operational Excellence, CACI International, Inc.

Dr. Kenneth Nidiffer, Director of Strategic Plans for Government Programs, Carnegie Mellon University, Software Engineering Institute

Dr. Ronald M Sega, Director, Systems Engineering Programs, Colorado State University

Abstract. This research explores the use of operational risk identification and mitigation techniques during the engineering process to determine whether this increased focus would have a positive effect on project outcomes. An approach using operational risk considerations to enhance the discovery of end user needs during an Agile engineering process is presented, and the results of a survey are provided.

Introduction

One missing aspect of most engineering processes as implemented on a project is an explicit focus on operational risk — that is, the evolving risk to the needs of the end user. This lack of focus on operational risk allows the creation of a chasm between evolving needs and delivered capabilities. The longer the time between identifying needs and delivering capabilities, the wider that gap becomes. This makes the end user less likely to deem the capabilities operationally effective.

Agile engineering approaches have emerged to help decrease the time between the identified need and the delivered capability by engaging end users actively in various planning and demonstration activities[1]. This active engagement of end-users establishes the groundwork for implicitly mitigating operational risk, however Agile methodologies still fail to explicitly use operational risk as a mechanism to ensure the evolving capability reduces the end user's evolving operational risk. Wrubel and Gross describe this disconnect, stating, "… [R]equirements for any given system are highly likely to evolve between the development of a system concept and the time at which the system is operationally deployed as new threats, vulnerabilities, technologies, and conditions emerge, and users adapt their understanding of their needs as system development progresses." [2]

Operational Risk	*The possibility of suffering mission or business loss.*
Operational Risk Management	*An operational practice with processes, methods, and tools for managing risks to successful mission and business outcomes.* *It provides a disciplined environment for proactive decision making to:* *- continually assess what could go wrong (operational risks)* *- determine which operational risks are most important to deal with, and* *- implement strategies to address operational risk*

Table 1. Operational Risk Definitions

In his 2015 report to Congress on the state of defense acquisition, the Honorable Frank Kendall, under secretary of defense for acquisition, technology and logistics, observed that the Department of Defense was optimizing cost and schedule performance over technical advancement, stating, "… [T]here is evidence that we have been pursuing less complex systems with about the same or less risk since 2009. This aligns with my concern that in some areas we may not be pushing the state-of-the-art enough in terms of technical performance. This endangers our military technical superiority. In my view, our new product pipeline is not as robust as it should be at a time when our technological superiority is being seriously challenged by potential adversaries. Not all cost growth is bad; we need to respond to changing and emerging threats." [3] These emerging threats, vulnerabilities and technology changes increase operational risk.

Operational Risk Management

Operational risk management is widely practiced in the banking industry and in military operations. In the banking industry, operational risk focuses on mitigating catastrophic financial loss and controlling the propagation of that loss to other banks and across international boundaries. In the military, operational risk has an emphasis on safety hazards and their impact on mission outcomes. Both of these applications of operational risk management form a foundation for a more robust treatment of operational risk.

Operational risk within the banking industry is focused on reducing the probability of loss due to events such as fraud, mismanagement, system failures, failed investments or legal considerations. Banks estimate their risk exposure, establish mitigation activities and set aside financial reserves to cover loss. In 1974, the Bank for International Settlements (BIS) established the Basel Committee on Banking Supervision to develop standards for international banking focused on risk reduction. These standards evolved from 1988 through 2010 and were known as the Basel Accord, Basel II and Basel III. The term "operational risk" emerged during this time and became the leading approach for managing banking institution risk in the 1990s. [4]

The U.S. Marine Corps defines operational risk as "the process of identifying and controlling hazards to conserve combat power and resources." [5] The U.S. Navy defines operational risk in OPNAV INSTRUCTION 3500.39B as "The process of dealing with risk associated with military operations, which includes risk assessment, risk decision making and implementation of effective risk controls." [6] The U.S. Air Force defines risk management as "a decision-making process to systematically evaluate possible courses of action, identify risks and benefits, and determine the best course of action (COA) for any given situation." [7] The guidance document emphasizes personnel health, safety and environmental factors. The U.S. Army includes guidance for the management of risk in operational contexts within ATP 5-19 and defines risk management as "The process of identifying, assessing, and controlling risks arising from operational factors and making decisions that balance risk cost with mission benefits." [8] The focus of operational risk in the Marine Corps, the Navy, the Air Force and the Army is on operational hazards rather than on a more general definition of operational risk.

The narrow focus within the banking industry on financial risk and within military operations on safety hazards decreases the potential effectiveness of operational risk activities. To that end, more inclusive definitions of operational risk and operational risk management are provided in Table 1.

With these more general definitions, any risk to the successful accomplishment of mission or business outcomes could be identified and addressed. The operational user community can then participate more fully with the acquisition and engineering communities in helping manage risk during the project life cycle. The more robust risk approach [9] is shown in Figure 1 and would require the acquisition community, the engineering community and the operational community to actively identify risks from their unique perspectives, which can then influence project outcomes.

Operational users identify mission and business needs to the acquisition community based on operational risk and threats. The acquisition community commits to providing enhanced capabilities to the end-user at a certain time, for a certain cost. The acquisition community then translates those needs into requirements which are provided to a set of engineers who develop and delivery enhanced capabilities to the end-user while meeting cost and schedule constraints agreed to with the acquisition community. The end-users participate continuously by providing insight into evolving operational needs and are advocates for the capability under development.

When operational risks are explicitly captured and addressed as part of the project life cycle, the resulting capabilities are more likely to address the evolving operational needs of the end user, increasing the likelihood of operational effectiveness of the delivered capabilities and overall user acceptance.

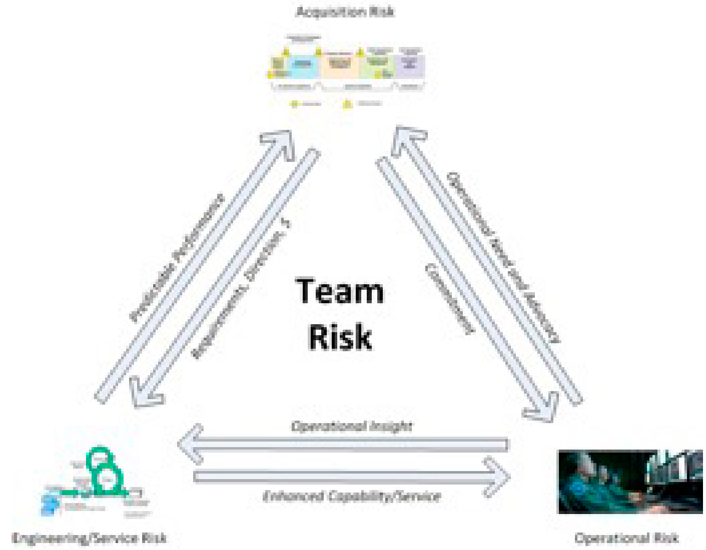


Figure 1. A Robust Risk Approach

Operational Risk-Driven Engineering Requirements/Engineering Development (ORDERED)

ORDERED is a repeatable method designed to influence engineering activities throughout the project life cycle with the purpose of improving project outcomes [10] through the explicit consideration of operational risk. New or enhanced capabilities are driven by the mission and business needs of diverse stakeholders. [11] Mission and business needs increase operational risk when gaps in current capabilities fail to address these needs. As new capabilities are developed, mission and business needs evolve, increasing the operational risk that the new capability will fail to address these changes. The ORDERED method ensures that program requirements and development activities are enacted with a thorough consideration of operational risk concerns. ORDERED does

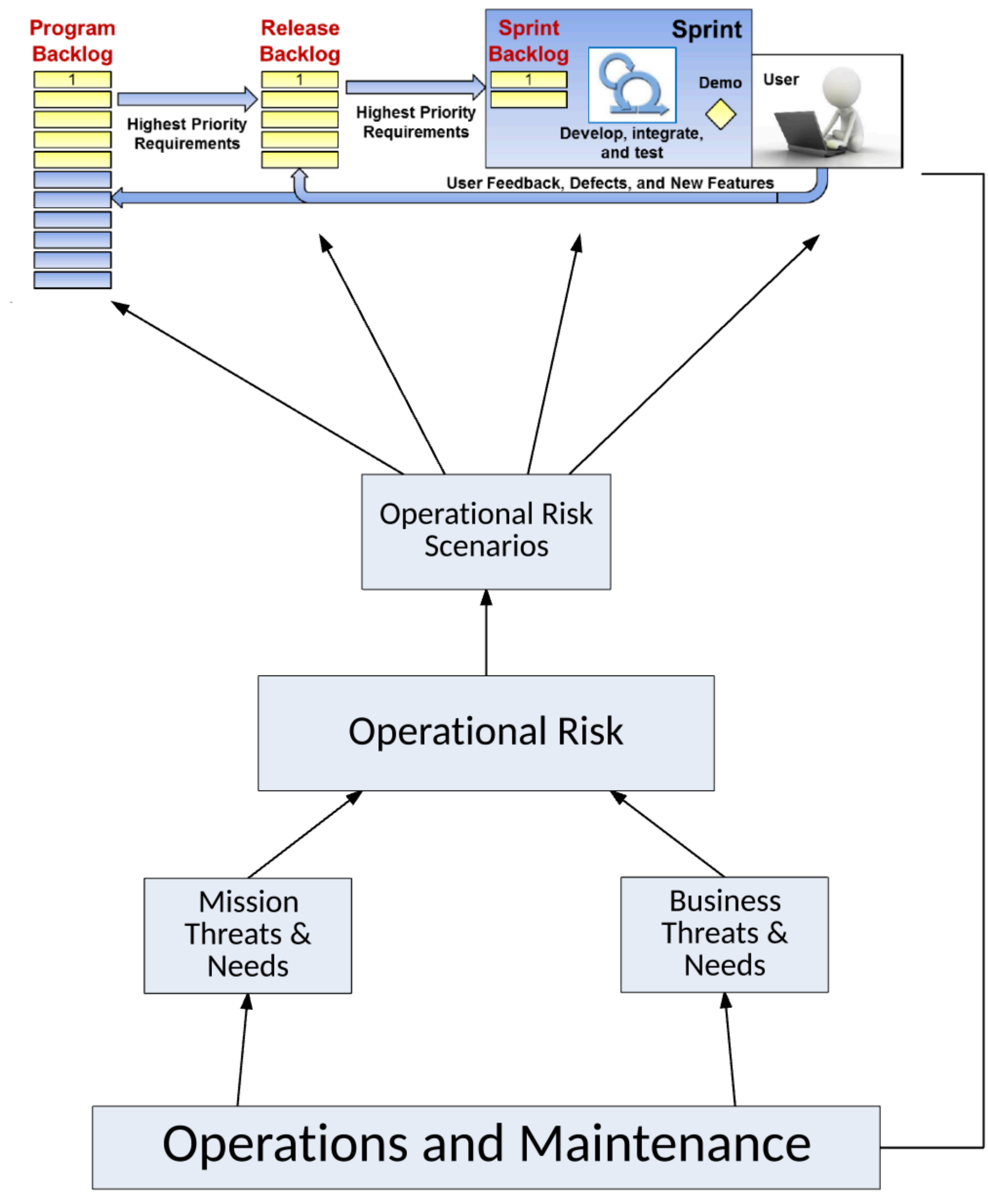


Figure 2. The ORDERED Approach

not replace a program's current engineering methodology, but rather augments current approaches with operational risk considerations. Therefore, ORDERED will work with any life cycle or engineering method. Figure 2 presents a high-level overview of the ORDERED approach as it would apply to a program using an Agile methodology. [1]

Mission and business threats and needs are identified by end-users during operations and maintenance activities. The gap between needs and threats and current systems and operational processes generates operational risk. Operational risk is captured in the form of individual risk statements, which define the potential negative outcomes that could impact mission execution or business operations. Essentially, operational risk is the "loss" that the mission or business may realize.

Operational risk attributes are derived from the risks. These attributes are characteristics of the system or capability. Operational risk scenarios are developed to further describe the risk in terms of the environment or behavior that would negatively impact mission or business outcomes. The scenarios are then used during the Agile engineering process to inform activities such as program and release backlog development and grooming, sprint planning, sprint execution and deployment strategies. As the mission and business needs and threats evolve, operational risks are continually identified, operational risk attributes continue to be identified or refined, and scenarios are developed or updated.

ORDERED uses a taxonomy to help with risk identification. A taxonomy is useful both when exploring sources of risk and when analyzing and classifying identified risks. The ORDERED Taxonomy is shown in Figure 3. The taxonomy was developed and simplified by considering personal experience and several source documents. [12] [13] [14] [15]

Operational Risk and User Stories

Most Agile methodologies capture expected behavior through the use of user stories. [16] User stories are statements of what the end user wants from the system or software. These user stories create the product backlog and are continually updated to ensure the stories represent the end user's prioritized needs. During sprint planning, user stories are moved from the product backlog to the sprint backlog for implementation planning. Active risk management throughout the Agile engineering process is a recognized best practice. The development team shares the responsibility for identifying risks that may impact the sprint, the project, or larger program. The addition of operational risk considerations during risk management activities provides a

ORDERED Taxonomy	
A. MISSION	B. BUSINESS
1. Mission Planning a. Stability b. Completeness c. Clarity d. Feasibility e. Precedents f. Agility	1. Resource Planning a. Workforce b. Budget c. Facilities d. Equipment and Systems
2. Mission Execution a. Efficiency b. Effectiveness c. Repeatability d. Agility e. Affordability f. Security g. Safety	2. Governance a. Policies b. Procedures c. Organizational Structure d. Contracts e. Analytics f. Compliance g. Risk Management
3. Mission Outcomes a. Predictability b. Accuracy c. Usability d. Timely e. Efficient	3. Strategic Planning a. Vision and Mission b. Values c. Goals d. Objectives e. Monitoring
4. Operational Systems a. Throughput b. Usability c. Flexibility d. Reliability e. Evolvability f. Security g. Supportability h. Inventory	4. Stakeholder Management a. Identification b. Stakeholder Mgmt Plan c. Engagement d. Controlling
5. Operational Processes a. Suitability b. Repeatability c. Predictability d. Agility e. Security	5. Continuous Improvement a. Problem Identification b. Opportunity Identification c. Root Cause Analysis d. Improvement Planning e. Implementation
6. Operational Staff a. Skill Level b. Training c. Turnover d. Affordability	

Figure 3. The ORDERED Taxonomy

Mission Objectives	Business Objectives
1. Detect, contain, and remediate cyber security threats.	1. Reduce cybersecurity related incidents.
2. Analyze trends, determine root causes, and improve system resilience.	2. Reduce cost of cybersecurity activities.
3. Educate system operators and maintainers on cybersecurity threats.	3. Position for agency organizational consolidation.

Table 2. CSOC Operational Objectives

Story ID	User Story
S001	As a CSOC analyst, I want to be alerted when an intrusion is detected.
S002	As a CSOC supervisor, I want to be able to evaluate the effectiveness of CSOC analysts.

Table 3. Initial User Stories

valuable mechanism to assist in developing the product backlog and prioritizing user stories for the sprint backlog.

Consider the Cyber Security Operations Center (CSOC) with operational mission and business objectives shown in Table 2.

During planning sessions with end users, the following user stories describing the expected behavior of a new incident detection system could have been captured.

While user stories describe expected or desired behavior, operational risks address unwanted behavior. Using the ORDERED process, the CSOC team conducted a risk identification workshop and identified more than 60 operational risks. The top three are shown below in Table 4, along with their probability of occurrence (P(O)), impact of occurrence (I(O)) and overall risk exposure.

As shown in Table 5, these operational risks were further analyzed by determining their risk attributes using the ORDERED Taxonomy, attribute concerns, and risk scenarios intended to influence Agile life cycle activities.

Scenarios are used routinely in systems and software engineering. The purpose of a scenario is to describe expected results of a system during development in terms of actual behavior. [17] Scenarios describe how the system should behave under certain conditions or when presented with certain stimuli. [18] Operational risk scenarios describe potential future unwanted behavior of the system that would cause mission or business impact to the operational organization. Similar to the concept of anti-patterns, [19] operational risk scenarios describe undesirable outcomes that need to be mitigated because they increase operational risk. The added insight provided by identifying and analyzing operational risks and developing operational risk scenarios can lead to the creation of additional user stories describing capabilities required to mitigate the operational risk. Given the risks in the CSOC example, Table 6 captures additional user stories influenced by a consideration of operational risk.

The addition of explicit identification and analysis of operational risks during Agile planning activities can provide a richer discussion of user stories and improve end user acceptance of the capabilities delivered.

Evaluating the Effectiveness of an Operational Risk Focus

Since few projects explicitly identify and capture their end user's operational risks as

Top "N"	Risk ID	Risk Statement	P(O)	I(O)	Risk Exposure
1	CSOC003	80% of operator time is spent responding to incidents; may not see trends or understand root cause of incidents	4	3	12
2	CSOC001	Incident occurrence is unpredictable; may not have adequate resources to respond during crisis	4	2	8
3	CSOC002	Heavy compliance and oversight make processes rigid; may not be able to adjust quickly to new events	2	3	6

Table 4. CSOC Top 3 Operational Risks

Top "N"	Risk ID	Risk Statement	Risk Attributes	Attribute Concern
1	CSOC003	80% of operator time is spent responding to incidents; may not see trends or understand root cause of incidents	1. Operator: Training, Skill Level 2. Mission Execution: Effectiveness	Inability of inexperienced staff to detect trends and determine causes of incidents
Operational Risk Scenarios 1. Junior staff members become overwhelmed responding to incidents and fail to detect new intrusion within 2 hours 2. During every 8-hour routine shift change, a vulnerability is exploited yet analysts fail to detect the incident or connect the periodic exploitations as related				
2	CSOC001	Incident occurrence is unpredictable; may not have adequate resources to respond during crisis	1. Mission Execution: Repeatability 2. Mission Outcomes: Predictability 3. Resource Planning: Workforce	Ability to predict staffing needs based on expected workload
Operational Risk Scenarios 1. During planning for staffing needs, supervisors fail to account for historical data and seasonal changes 2. During expected upcoming political events, attempts to penetrate the network increase beyond the analyst's ability to respond.				
3	CSOC002	Heavy compliance and oversight make processes rigid; may not be able to adjust quickly to new	1. Operational Systems: Flexibility 2. Operational Processes: Suitability	Inability to adjust the level of controls and reporting in times of crisis
Operational Risk Scenarios 1. During periods of high intrusion activities, required reporting, mandatory system controls, approvals and logging requirements impact the ability of analysis to respond quickly to incidents.				

Table 5. Operational Risk Scenarios

Story ID	User Story
S003	As a CSOC supervisor, I want to be able to predict required analyst staffing based on historical incident activity.
S004	As a CSOC analyst, I want to be able to bypass a set of optional controls during crisis events.
S005	As a CSOC analyst, I want the system to provide trend analysis and alerts.

Table 6. Additional User Stories

part of their risk management process, it is difficult to evaluate the impact of implementing this concept. A survey instrument was developed and administered to explore the relationship between operational risk considerations and project performance. Operational risk considerations were defined as actively eliciting operational risk from end users during the early solution development stages of a project, as well as actively and continually involving end user perspectives during development to identify and mitigate evolving operational risk throughout the project life cycle. Project performance was defined as meeting cost and schedule expectations and delivering capabilities that satisfy the end user's most critical quality attribute requirements and that mitigate operational risk.

The survey was administered to 104 project managers on Oct. 14, 2015. Figure 4 shows how comparing the existence of an operational risk process capability affected project outcomes.

The number of projects exhibiting lower project performance decreased from 50 percent for projects with low risk process capability to 36 percent for projects with medium operational risk process capability and went down to 21 percent for projects with higher operational risk process capability. Projects exhibiting medium project performance increased from 39 percent for projects with low operational process performance to 49 percent for projects with medium operational process performance and increased to 52 percent for projects with higher operational risk process performance. Projects exhibiting high project performance increased from 11 percent for projects with lower operational risk process performance to 15 percent for projects with medium operational risk process capability to 27 percent for projects with higher operational risk process capability. The Gamma score shows a moderately strong to strong positive relationship between the two variables, and the p-value of .006 provides confidence that the relationship is valid.

Given the strength of the relationship and the very low p-value, one can confidently conclude that projects within the sample that focused more on operational risk during the project life cycle also exhibited better project performance than those that focused less on operational risk during the project life cycle.

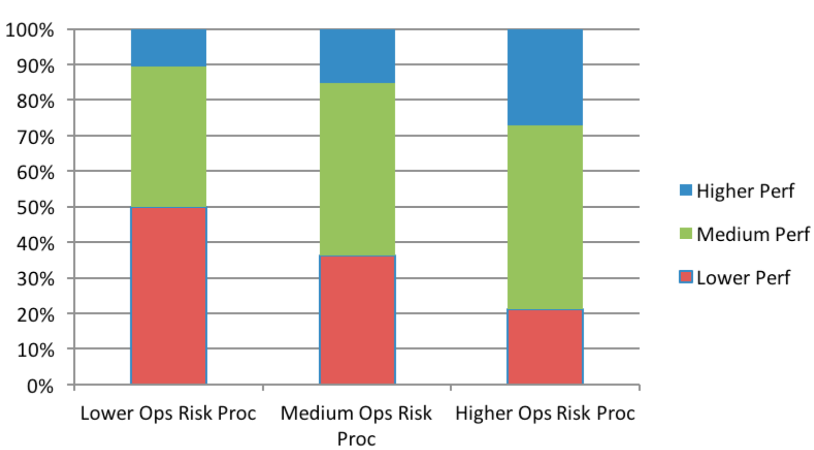


Figure 4. Operational Risk Process Capability and Project Performance

Conclusions

Risk management as an engineering practice is commonplace. However, actively eliciting operational risk considerations during the engineering life cycle is not as common, even in projects using an Agile engineering process. This paper explored the relationship between an operational risk focus during the project life cycle — specifically the use of operational risk to inform the creation of user stories — and improvement in project outcomes. Early results indicate that an explicit focus on the end user's evolving operational risk during the engineering life cycle results in improved project performance. More work is needed to explore techniques to integrate an operational risk mindset into current Agile engineering methods, allowing the explicit mitigation of operational risk and improved project outcomes.

REFERENCES

1. Modigliani, P. & S. Chang. (2014, March.) Defense Agile Acquisition Guide: Tailoring DoD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities. McLean, Va. MITRE Corporation.
2. Wrubel, E. & Gross, Jon. (2015.) Contracting for Agile Software Development in the Department of Defense: An Introduction (CMU/SEI-2015-TN-006). Retrieved August 22, 2015, from the Software Engineering Institute, Carnegie Mellon University website. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=442499.
3. Under Secretary of Defense, Technology, and Logistics (USD[AT&L]), Editor. (2015.) PERFORMANCE OF THE DEFENSE ACQUISITION SYSTEM 2015 ANNUAL REPORT, A.
4. Power, M. (2005.) The invention of operational risk. Review of International Political Economy. 12(4), 577-599.
5. United States Marine Corps. (2002.) MCI, ORM 1-0: Operational Risk Management. Headquarters Marine Corps, Washington D.C.
6. OPNAV, 3500.39 B.(2004). Operation risk management.
7. USAF, Pamphlet 90-803 - RISK MANAGEMENT (RM) GUIDELINES AND TOOLS.
8. Army, ATP 5-19. (2014.) Risk Management.
9. Gallagher, B.P. (2002.) Interpreting Capability Maturity Model Integration (CMMI) for Operational Organizations.
10. Gallagher, B.P. (2015.) Improving Systems Engineering Through Operational Risk Considerations. In the 27th Annual IEEE Software Technology Conference.
11. Susnien, D. & P. Vanagas. (2015.) Means for satisfaction of stakeholders' needs and interests. Engineering economics. 55(5).
12. Gallagher, B.P., et al. (2005.) A Taxonomy of Operational Risks.
13. Gallagher, B., et al. (2011.) CMMI for Acquisition: Guidelines for Improving the Acquisition of Products and Services. Addison-Wesley Professional.
14. ISO, 31000 (2009.) Risk management – Principles and guidelines, in International Organization for Standardization. Geneva, Switzerland.
15. Project Management Institute, Incorporated. (2013.) PMI, A Guide to the Project Management Body of Knowledge (PMBOK® Guide).
16. Cohn, M. (2004.) User stories applied: For agile software development. Addison-Wesley Professional.
17. Mylopoulos, J.; L. Chung & E. Yu. (1999.) From object-oriented to goal-oriented requirements analysis. Communications of the ACM. 42(1), 31-37.
18. Bass, L.; Klein, Mark & Moreno, Gabriel. (2001.) Applicability of General Scenarios to the Architecture Tradeoff Analysis Method (CMU/SEI-2001-TR-014). Retrieved August 22, 2015, from the Software Engineering Institute, Carnegie Mellon University website. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5637.
19. Brown, W.H.; R.C. Malveau & T.J. Mowbray. (1998.) AntiPatterns: refactoring software, architectures, and projects in crisis.

ABOUT THE AUTHORS

Dr. Brian P. Gallagher is the SVP of Operational Excellence for CACI, and is responsible for program management and delivery, process effectiveness, and continuous improvement initiatives. Brian has held numerous positions within Northrop Grumman, Carnegie Mellon's Software Engineering Institute, the Aerospace Corporation, and the United States Air Force. He holds a Ph.D. in Systems Engineering through Colorado State University, an M.S. degree in computer science from the Florida Institute of Technology and a bachelor of technology degree from Peru State College.

brian.gallagher@colostate.edu
bgallagher@caci.com

Dr. Kenneth E. Nidiffer has over 53 years of government, industry and academic experience in the field of software and systems engineering. Ken has successfully executed positions as a senior vice president at Fidelity Investments, vice president of the Software and Systems Consortium, and director of technical operations/engineering at Northrop Grumman Corporation. He is currently the director of strategic plans for government programs at the Carnegie Mellon Software Engineering Institute. Ken received his B.S. degree in chemical engineering from Purdue University, Indiana; his M.S. degree in astronautical engineering from the Air Force Institute of Technology, Ohio; his MBA degree from Auburn University, Alabama; and his D. Sc. in systems engineering from George Washington University, Washington, D.C.

nidiffer@sei.cmu.edu

Dr. Ronald M. Sega serves as director, systems engineering programs and special assistant to the chancellor for strategic initiatives at Colorado State University. He is also the Woodward Professor of Systems Engineering. From 2010 to 2013, he also served as vice president and enterprise executive for energy and the environment at both Colorado State University and The Ohio State University. He holds a B.S. in math and physics from the U.S. Air Force Academy in Colorado Springs, an M.S. in physics from Ohio State University and a Ph.D. in electrical engineering from the University of Colorado. Prior to joining CSU, Dr. Sega was the under secretary of the Air Force from 2005 to 2007, where he served as the DoD executive agent for space and led the Air Force team that won the overall Presidential Award for Leadership in Federal Energy Management for 2006. After 31 years in the Air Force, having served in various assignments at Air Force Space Command and as a pilot, he retired from the Air Force Reserve in 2005 as a major general in the position of reserve assistant to the chairman of the Joint Chiefs of Staff. Dr. Sega was director of defense research and engineering (DDR&E), the chief technology officer for the Department of Defense (DoD), from 2001 to 2005. Dr. Sega was a faculty member in the College of Engineering and Applied Science at the University of Colorado at Colorado Springs from 1982 to 2013, also serving as dean from 1996 to 2001. A former astronaut, he flew aboard Space Shuttles Discovery (1994) and Atlantis (1996).

A Comparison of Commercial and Defense Software

Capers Jones, Vice President and CTO, Namcook Analytics LLC

Abstract. Both commercial software and defense software are major industries with vast economic importance and also major importance to U.S. national security.

Commercial software has created a number of the wealthiest companies on the planet and a significant number of personal millionaires and billionaires.

Defense software has become the key operating component of all modern weapons systems and all intelligence systems and is a major contributing factor in U.S. global military leadership. Both industries are important.

Commercial software is largely created by employees of commercial software companies, whereas defense software is largely created by specialized defense contractors.

This difference has led to very different kinds of development practices and to very different productivity rates. The overhead of contract management has ballooned defense requirements, designs, and status reports so much that producing paper documents is the No. 1 cost driver for defense software.

Introduction

Software is the main operating tool of business, government and military operations in 2016. There are dozens of industries and thousands of companies that produce software. This short study discusses only two types: commercial software and defense software.

Commercial software

Commercial software began in the 1960s and has expanded to become one of the largest and most profitable industries in world history. Some of the major companies in this sector include (in alphabetical order) Apple Inc., CA Technologies, Cisco Systems Inc., Facebook, Google, IBM (where the author worked on commercial applications), Microsoft Corp., Oracle Corp., SAP SE, Symantec and hundreds of others.

Commercial software had a slow start because companies such as IBM "bundled" applications, or gave away software for free when customers purchased computers. As a result of an anti-trust lawsuit, IBM "unbundled" in 1969, giving commercial software a major impetus to grow and expand.

Commercial software started in the mainframe era with applications such as database packages and sorts that were widely used. Today commercial applications run on all platforms and under all operating systems. While there are still some mainframe commercial packages, PC applications, tablet applications, and smartphone applications are dominant. There are not many embedded commercial packages, although there are some.

The largest commercial applications are enterprise resource planning systems, such as SAP and Oracle, at over 250,000 function points. Big operating systems, such as Windows 10 and IBM's operating systems, can top 100,000 function points. There are also hundreds of smaller commercial packages, such as static analysis and test tools, with perhaps 3,000 function points. Some smartphone packages are only a few hundred function points in size.

Commercial software is usually constructed by employees of the software companies. Requirements come from market analysis or from individual inventors. Commercial software needs high quality and reliability to be successful, so the larger commercial software vendors tend to be quite sophisticated in quality control. As of 2016, the major cost drivers for commercial software are the following:

2016 Commercial Software Cost Drivers

1. Finding and fixing bugs.
2. Producing paper documents.
3. Marketing and sales.
4. Code development.
5. Requirements changes.
6. Training and learning.
7. Project management.
8. Meetings and communication.

(The data on commercial software in this report comes from benchmarks carried out by the author and colleagues in over 150 corporations and about 30 civilian government agencies, plus several military clients. We have nondisclosure agreements with clients, so they can't be named specifically.)

Defense software

Defense software began even earlier — in the 1940s — with analog computers. It has become a major factor in all modern weapons systems and in all national defense capabilities, such as radar nets, satellites and intelligence. Some of the major defense contractors include (in alphabetical order) BAE Systems, Boeing Co., Computer Sciences, General Dynamics, General Electric, Honeywell, ITT, Lockheed Martin and dozens of others.

Defense software includes weapons systems, embedded applications for things like aircraft navigation and flight controls, and many applications for logistics, personnel matters and medical records. Defense software probably runs on more embedded computers than any other kind of software, but also runs on mainframes, personal computers, servers, tablets and even some custom smartphones.

The largest defense applications are massive systems, such as the "Star Wars" missile defense system and the worldwide military command and control system (WWMCCS), at over 300,000 function points each. The full suite of applications on an Aegis destroyer can top 200,000 function points. A ship-board gun control system can top 20,000 function points. There are also many smaller defense applications, including those used for aiming torpedoes and for custom smartphones.

Defense software is largely constructed by sophisticated defense contractors rather than by military personnel themselves. Defense applications also need good quality control. As it happens, the higher levels of the capability maturity model integrated (CMMI) do lead to high quality levels.

Since many defense contracts require competitive bidding, defense software contracts frequently lead to litigation from disgruntled vendors who fail to gain the contract or a portion of a contract. Indeed, special courts have been established to handle defense contract litigation. Civilian and commercial software are seldom involved in this kind of litigation. With defense contracts, vendors are often known to each other and the results are also known. For commercial contracts, the potential vendors may not know who else is bidding and probably won't know the terms of the accepted contract. This leads to many more lawsuits regarding defense software than civilian.

Due to the fact that defense software is built under contract, the defense software industry has developed elaborate contract procedures and extensive but rather burdensome contract monitoring and governance methods.

These contracts have the unintended consequence of creating a need for excessive paperwork, starting with requests

CMMI Level	Defect Potential per Function Point	Defect Removal Efficiency	Delivered Defects per Function Point	Delivered Defects
SEI CMMI 1	4.5	87.00%	0.585	1,463
SEI CMMI 2	3.85	90.00%	0.385	963
SEI CMMI 3	3	96.00%	0.12	300
SEI CMMI 4	2.5	97.50%	0.063	156
SEI CMMI 5	2.25	99.00%	0.023	56

Table 1. Software Quality and the SEI Capability Maturity Model Integrated (CMMI) for 2,500 function points

for proposals (RFPs) and including over 20 other kinds of documents. In fact, as of 2016, producing paper documents is the No. 1 cost driver for large defense applications. Overall, the major cost drivers for defense software in 2016 are the following:

2016 Defense Software Cost Drivers

1. Producing paper documents.
2. Finding and fixing bugs.
3. Marketing and contract administration.
4. Code development.
5. Requirements changes.
6. Training and learning.
7. Meetings and communication.
8. Project management.

We have nondisclosure agreements with clients so they can't all be named specifically, but among the author's defense software clients have been ITT (where the author was employed), the Air Force at several locations, naval surface weapons groups, and defense contractors in aerospace and weapons systems.)

Note: The author had a contract from the Air Force to demonstrate the value of the higher CMMI levels. This study showed quality improvements for the higher CMMI levels.

Function point metrics are used to show defect potentials because defects originate in many sources, not just in source code. The approximate 2016 U.S. average for defect potentials is shown in Table 2.

The concept of "defect potentials" originated in IBM circa 1970. It is the probable total number of bugs that will be found in all defect sources. The values for each defect source were based on IBM historical quality data, which was one of the most complete in the world at that time. Of course dozens of companies use defect potentials in 2016, and we study them during all benchmarks and also predict them for new projects.

Defect potentials are paired with another useful metric, defect removal efficiency (DRE). This is the percentage of bugs found and removed before delivery of software to customers. The U.S. average for DRE in 2016 is just over 92 percent, although top projects in both defense and commercial sectors exceed 99 percent.

Function point metrics were developed at about the same time in the 1970s, in part because "lines of code" cannot show all sources of software defects. If they are not removed, the non-code defects in requirements and design eventually find their way into the code. Both the defense and commercial software sectors are good at removing software defects.

Note that although software operational features reside in the source code, code development is only cost driver No. 4 for both commercial software and for defense software. Both industry sectors spend more money on bug repairs and paper documents than they do on the code itself. Both industries also put substantial effort into dealing with requirements changes, which can run from 1 percent per calendar month to over 4 percent per calendar month. Requirements changes are very common in both sectors and are also very expensive.

Because producing paper documents is expensive for both commercial and defense software, it is useful to show the approximate sizes of key documents. Table 3 shows comparative document pages for an application of 2,500 function points (about 150,000 Java code statements):

Note that paperwork volumes in both the commercial and defense sectors increase with application size. Commercial software requires more paperwork than many other industries, such as banking and insurance. However, as far as can be determined, defense software requires a larger volume of paperwork than any other known kind of software.

Since most documents today are produced electronically, actual printing is not the main paperwork cost driver. About 95 percent of document costs go to analysis, writing, editing, reviewing and approving. Paper document printing is not a major cost driver. It is interesting that the document teams in the defense sector are about 15 percent larger than similar teams in the commercial sector. This is due to the elaborate

1. Requirements	0.70 defects per function point
2. Architecture	0.10 defects per function point
3. Design	0.95 defects per function point
4. Code	1.15 defects per function point
5. Security Code Flaws	0.25 defects per function point
6. Documents	0.45 defects per function point
7. Bad fixes	0.65 defects per function point
8. Totals	**4.25 defects per function point**

Table 2. Approximate Average U.S. Software Defect Potentials, circa 2016

	Document Types	Commercial Pages	Defense Pages	Defense Percent
0	RFP	-	268	
1	Requirements	613	2,358	385.00%
2	Architecture	141	387	275.00%
3	Initial design	737	2,875	390.00%
4	Detail design	1,361	5,647	415.00%
5	Test plans	324	575	177.50%
6	Development Plans	138	325	236.36%
7	Cost estimates	141	255	181.38%
8	User manuals	600	2,190	365.00%
9	HELP text	482	875	181.37%
10	Courses	363	555	153.10%
11	Status reports	209	1,015	485.90%
12	Change requests	477	997	209.02%
13	Bug reports	2,628	2,775	105.61%
	TOTAL	**8,212**	**21,097**	**256.92%**

Table 3. Documents for 2,500 Function Points for Commercial and Defense Software Applications

documentation requirements and standards mandated by the Department of Defense.

The amount of defense software paperwork is not due to technical factors but seems to be caused by the elaborate oversight and project and contract governance functions associated with defense software applications. It is governance, for example, that expands the sizes of status reports.

Before proceeding further, we should note that building software is a much more complex task than just coding. The author's benchmark data collection method examines the results of 50 software development activities.

Table 4 shows a side-by-side comparison of 50 typical activity sets for large software systems in the commercial and defense sectors, each in the 10,000 function point size range.

As can be seen, both commercial and defense applications of necessity use similar activities. But the two activity sets are not identical. Defense projects use independent verification and validation (IV&V) and independent testing, which seldom occur in the civilian sector and almost never in commercial software.

Commercial software projects require competitive analysis of similar projects combined with risk analysis, which are not common in the defense sector. Defense projects often use earned-value analysis (EVA), which is seldom used in the civilian sector and has not been used for any commercial software among the author's clients.

Because of the intrinsic need for high quality levels, both defense software and commercial software are among the 10 best U.S. industries in overall software quality control.

Both commercial software and defense software are good in pre-test defect removal, such as static analysis and inspections. However, defense software also uses IV&V, which is seldom used on any civilian software project and almost never on commercial software. Defense software also uses independent testing, which is rare in the civilian sector. These added quality stages in the defense sector raise costs and also have minor quality benefits.

U.S. Software Quality Ranges

Software quality varies widely from industry to industry. In general, the industries that build complex physical products controlled by software have the best software quality. In order to provide context for commercial and defense software, Table 5 shows approximate U.S. quality results for 50 selected industries.

(Note: The author's data includes information from 75 industries, but it was necessary to shorten the tables for publication purposes. The removal of 25 industries changes the averages slightly.)

As can be seen, commercial and defense software have both done well in overall software quality results. No industry releases zero defects in major software applications, but to go beyond the U.S. average and approach 99 percent in DRE is a sign of quality sophistication.

High quality does not come from testing alone. It requires defect prevention such as joint application design (JAD),

		Commercial Activities	Defense Activities
1	Request for proposal (RFP)	N	Y
2	Competitive bidding	N	Y
3	Contract litigation by losing vendors	N	Y
4	Contract administration	N	Y
5	Business and competitive analysis	Y	N
6	Risk analysis for project	Y	Y
7	Risk solution planning	Y	Y
8	Requirements	Y	Y
9	Requirements changes	Y	Y
10	Requirements Inspection	Y	Y
11	Requirements modeling	Y	N
12	Prototyping	Y	Y
13	Architecture	Y	Y
14	Architecture Inspection	Y	Y
15	Project sizing (LOC, function points)	Y	Y
16	Project parametric estimation	Y	Y
17	Project earned-value analysis (EVA)	N	Y
18	Initial Design	Y	Y
19	Detail Design	Y	Y
20	Design inspections	Y	Y
21	Coding	Y	Y
22	Code inspections	Y	Y
23	Reuse acquisition	Y	Y
24	Static analysis	Y	Y
25	COTS package purchase	N	Y
26	Open-source acquisition	N	Y
27	Code security audit	Y	Y
28	Independent Verification & Validatior	N	Y
29	Configuration control.	Y	Y
30	Integration	Y	Y
31	User documentation and tutorials	Y	Y
32	Unit testing	Y	Y
33	Function testing	Y	Y
34	Regression testing	Y	Y
35	Integration testing	Y	Y
36	Performance testing	Y	Y
37	Security testing	Y	Y
38	Usability testing	Y	N
39	System testing	Y	Y
40	Cloud testing	Y	N
41	Cyber-attack avoidance testing	Y	Y
42	Field (Beta) testing	Y	Y
43	Acceptance testing	Y	Y
44	Independent testing	N	Y
45	Ethical hacking	N	Y
46	Quality assurance	Y	Y
47	Installation/training	Y	Y
48	Project measurement - function poin	Y	N
49	Project office status tracking	Y	Y
50	Project management	Y	Y

Table 4: Comparison of 50 Commercial and Defense Software Activity Sets (Assumes 10,000 function points)

	Industry	Defect Potentials Per Function Point 2016	Defect Removal Efficiency (DRE) 2016	Delivered Defects per Function Point 2016
1	Manufacturing - medical devices	4.6	99.50%	0.02
2	**Government - military**	**4.7**	**99.00%**	**0.05**
3	Manufacturing - aircraft	4.7	99.00%	0.05
4	Smartphone/tablet applications	3.3	98.50%	0.05
5	Government - intelligence	4.9	98.50%	0.07
6	**Software (commercial)**	**3.5**	**97.50%**	**0.09**
7	Telecommunications operations	4.35	97.50%	0.11
8	**Manufacturing - defense**	**4.65**	**97.50%**	**0.12**
9	Manufacturing - telecommunications	4.8	97.50%	0.12
10	Process control and embedded	4.9	97.50%	0.12
11	Manufacturing - pharmaceuticals	4.55	97.00%	0.14
12	Professional support - medicine	4.8	97.00%	0.14
13	Transportation - airlines	5.87	97.50%	0.15
14	Manufacturing - electronics	4.9	97.00%	0.15
15	Banks - commercial	4.15	96.25%	0.16
16	Manufacturing - automotive	4.3	96.25%	0.16
17	Manufacturing - chemicals	4.8	96.50%	0.17
18	Manufacturing - appliances	4.3	96.00%	0.17
19	Insurance - Life	4.6	96.00%	0.18
20	Banks - investment	4.3	95.50%	0.19
21	Insurance - property and casualty	4.5	95.50%	0.2
22	Government - police	4.8	95.50%	0.22
23	Insurance - medical	4.8	95.50%	0.22
24	Social networks	4.9	95.50%	0.22
25	Games - computer	3.75	94.00%	0.23
26	Transportation - trains	4.7	95.00%	0.24
27	Public utilities - electricity	4.8	95.00%	0.24
28	Public utilities - water	4.4	94.50%	0.24
29	Accounting/financial consultants	3.9	93.50%	0.25
30	Professional support - law	4.75	94.50%	0.26
31	Manufacturing - nautical	4.6	94.00%	0.28
32	Transportation - bus	4.6	94.00%	0.28
33	Hospitals - administration	4.8	93.00%	0.34
34	Transportation - ship	4.3	92.00%	0.34
35	Oil extraction	4.15	91.00%	0.37
36	Natural gas generation	4.8	91.50%	0.41
37	Games - traditional	4	89.00%	0.44
38	Wholesale	4.4	90.00%	0.44
39	Government - municipal	4.8	90.00%	0.48
40	Government - state	4.95	90.00%	0.5
41	Government - county	4.7	89.00%	0.52
42	Retail	5	89.50%	0.53
43	Stock/commodity brokerage	5.15	89.50%	0.54
44	Education - primary	4.3	87.00%	0.56
45	Mining - metals	4.9	87.50%	0.61
46	ERP vendors	5.7	89.00%	0.63
47	Transportation - truck	4.8	86.50%	0.65
48	Government - federal civilian	5.6	88.00%	0.67
49	Mining-coal	5	86.50%	0.68
50	Food - restaurants	4.8	85.50%	0.7
	National Averages	**4.63**	**93.76%**	**0.29**

Table 5. U.S. Software Quality Circa 2016 for Selected Industries

quality function deployment (QFD), requirements models, and embedded users. It also requires pre-test inspections and static analysis and requires formal test case development combined with certified test personnel.

Examples of organizations noted in 2016 as having excellent software quality include (in alphabetical order): Advanced Bionics, Apple Inc., AT&T Inc., Boeing Co., Ford Motor Co. (for engine controls), General Electric Co. (for jet engines), Hewlett Packard Enterprise (for embedded software), IBM (for systems software), Motorola (for electronics), NASA (for space controls), the Navy (for surface weapons), Raytheon (for defense), and Siemens AG (for both medical devices and telecommunications).

The most important economic fact about high quality is this: projects > 97 percent in DRE have shorter schedules and lower costs than projects < 90 percent in DRE. This is because projects that are low in DRE have test schedules that are at least twice as long as projects with high DRE due to omission of pre-test inspections and static analysis.

U.S. Software Productivity Ranges

As already shown, both the commercial and defense software industries are among the 10 best in overall software quality in the U.S. But when the focus changes from quality to productivity, the results are very different. The commercial software industry has very high productivity, but defense has very low productivity.

This low productivity rate is not because of poor coding — defense projects are quite good — but are due to the huge volumes of defense paperwork and the significant overhead of defense contract administration and status monitoring. Table 6 shows productivity rates for the same 50 industries shown in Table 5.

(Note: ERP packages are commercial software but have low productivity rates. This is because ERP packages are all very large — most exceed 100,000 function points. Other industries have wide ranges of application sizes, with average sizes below 1,000 function points.)

Expressing productivity in terms of function points per month and work hours per function point are the most common methods in 2016. The older “lines of code” (LOC) metric does not work well for projects where requirements, design and other noncoding tasks are the major cost drivers. LOC also penalizes high-level languages and makes older languages, such as assembly, look better than they really are.

Note that this paper uses IFPUG function points version 4.3. It does not use the newer SNAP metric due to the ambiguity and lack of data associated with this metric. Similar results would be shown using other function point metrics, such as COSMIC, FISMA, NESMA, and perhaps automated function points.

Note that the high software productivity rates for commercial software are not all due to technology factors such as methodologies. The high-tech commercial software world works very long hours. Some startup technology companies average almost 200 work hours per month, as opposed to the nominal 160 work hours per month that most companies work. Among the author's clients, commercial software engineering personnel work about 10 hours more per month than the software engineering personnel of the author's defense clients. Startup companies work even more hours per month than established companies. There are many startup commercial software companies as of 2016, but comparatively few startup defense software companies.

Work hours per month is a topic with global impacts. For example, software engineers in India and China work about 194 hours per month while software engineers in the U.S. work about 142. The countries with the lowest number of work hours per month are Germany and the Netherlands, both with less than 120 work hours per month. Most of the overtime hours are unpaid, and these high numbers of unpaid overtime hours raise productivity rates and lower software costs.

The low productivity rankings of defense manufacturing and military applications are not due to poor technology choices by the defense sector. In fact, the team software process (TSP) methodology in the defense software industry is the best available methodology for large software applications in the 10,000-function point range. The CMMI is also beneficial in the defense sector, especially to quality. The poor productivity in the defense industry is due to the high overhead of defense and military software caused by elaborate and cumbersome contract administration practices combined with very large document sets. The plethora of large applications in the defense sector also lowers productivity.

Current Problems for Both Commercial and Defense Software Applications

Although both the commercial and defense software industries have done well in quality control, the entire software industry is troubled by several chronic problems including many canceled projects, major cost overruns and major schedule delays. These problems are proportional to application size and are quite serious in applications above 10,000 function points. In fact, all of the breach-of-contract lawsuits in which the author has been an expert witness have been above 10,000 function points in size.

Consider the normal outcomes of 15 kinds of U.S. software projects. Table 7 shows the percentage of projects that are likely to be on time, late, or canceled due to excessive cost, schedule overruns, or poor quality.

As can be seen, schedule delays and canceled projects are distressingly common among all forms of software in 2016, including both the commercial and defense sectors. This explains why software is viewed by most CEOs as the least competent and least professional form of engineering in the current business world.

Note that the data in Table 7 is from benchmark and assessment studies carried out by the author and colleagues between 1984 and 2016. Unfortunately, data since 2010 is not much better than data before 1990. This is due to several reasons: 1) Very poor measurement practices and distressingly bad metrics, which prevent improvements from

	Industry		**Function Points Per Month 2016**	**Work Hours Per Function Point 2016**
1	Games - computer		15.75	8.38
2	Smartphone/tablet applications		15.25	8.66
3	**Software (commercial)**		**15**	**8.8**
4	Social networks		14.9	8.86
5	Banks - commercial		11.5	11.48
6	Banks - investment		11.5	11.48
7	Insurance - medical		10.5	12.57
8	Insurance - Life		10	13.2
9	Stock/commodity brokerage		10	13.2
10	Insurance - property and casualty		9.8	13.47
11	Manufacturing - telecommunications		9.75	13.54
12	Telecommunications operations		9.75	13.54
13	Process control and embedded		9	14.67
14	Manufacturing - pharmaceuticals		8.9	14.83
15	Oil extraction		8.75	15.09
16	Transportation - airlines		8.75	15.09
17	Professional support - medicine		8.55	15.44
18	Government - police		8.5	15.53
19	Professional support - law		8.5	15.53
20	Accounting/financial consultants		8.5	15.53
21	Manufacturing - electronics		8.25	16
22	Wholesale		8.25	16
23	Hospitals - administration		8	16.5
24	Manufacturing - chemicals		8	16.5
25	Manufacturing - nautical		8	16.5
26	Retail		8	16.5
27	Transportation - bus		8	16.5
28	Transportation - ship		8	16.5
29	Transportation - trains		8	16.5
30	Transportation - truck		8	16.5
31	Manufacturing - automotive		7.75	17.03
32	Manufacturing - medical devices		7.75	17.03
33	Manufacturing - appliances		7.6	17.37
34	Education - primary		7.5	17.6
35	Games - traditional		7.5	17.6
36	Manufacturing - aircraft		7.25	18.21
37	Public utilities - water		7.25	18.21
38	Government - intelligence		7.2	18.33
39	Food - restaurants		7	18.86
40	Government - municipal		7	18.86
41	Mining - metals		7	18.86
42	Mining-coal		7	18.86
43	Public utilities - electricity		7	18.86
44	**Manufacturing - defense**		**6.85**	**19.27**
45	**Government - military**		**6.75**	**19.56**
46	Natural gas generation		6.75	19.56
47	Government - county		6.5	20.31
48	Government - federal civilian		6.5	20.31
49	Government - state		6.5	20.31
50	ERP vendors		6	22
	National Averages		**8.69**	**16**

Table 6. U.S. Software Productivity Circa 2016 for Selected Industries

	Application Types	On-time	Late	Canceled
1	Scientific	68.00%	20.00%	12.00%
2	Smart phones	67.00%	19.00%	14.00%
3	Open source	63.00%	36.00%	7.00%
4	U.S. outsource	60.00%	30.00%	10.00%
5	Cloud	59.00%	29.00%	12.00%
6	Web applications	55.00%	30.00%	15.00%
7	Games and entertainment	54.00%	36.00%	10.00%
8	Offshore outsource	48.00%	37.00%	15.00%
9	Embedded software	47.00%	33.00%	20.00%
10	Systems and middleware	45.00%	45.00%	10.00%
11	Information technology (IT)	45.00%	40.00%	15.00%
12	**Commercial**	**44.00%**	**41.00%**	**15.00%**
13	**Military and defense**	**40.00%**	**45.00%**	**15.00%**
14	Legacy renovation	30.00%	55.00%	15.00%
15	Civilian government	27.00%	63.00%	10.00%
	Total Applications	**50.13%**	**37.27%**	**13.00%**

Table 7. Outcomes of U.S. Software Projects Circa 2016

Function Points	On Time	Cancel	Cost Overrun	Schedule Overrun
1	100.00%	0.00%	0.00%	0.00%
10	94.48%	2.30%	6.03%	5.52%
100	89.26%	6.60%	9.68%	10.74%
1,000	68.30%	10.15%	**15.65%**	**31.70%**
10,000	51.60%	**18.60%**	**29.12%**	**48.40%**
100,000	**27.73%**	**54.21%**	**59.63%**	**72.27%**
1,000,000	**4.00%**	**73.50%**	**97.50%**	**96.00%**

Table 8. Software Schedule Results by Application Size in 2016

Function Points	Defect Potential	Defect Removal	Delivered Defects
1	0.70	99.60%	0.00
10	1.50	99.00%	0.02
100	2.31	97.50%	0.06
1,000	3.85	96.00%	0.15
10,000	**5.75**	**91.50%**	**0.49**
100,000	**6.30**	**89.00%**	**0.69**
1,000,000	**7.30**	**85.50%**	**1.06**

Table 9. Software Quality Results by Application Size in 2016

being widely seen and understood by senior management; 2) Requirements creep averages between 1 percent and 4 percent per calendar month that delay schedules significantly; 3) Testing schedules that are at least twice as long as planned because companies lag in pre-test quality control; 4) Software that continues to require custom designs and manual coding, both of which are intrinsically expensive and error prone. Until the software industry adopts modern manufacturing concepts that utilize standard reusable components instead of custom-built artifacts, software can never be truly cost effective.

Small projects are generally much more successful than large systems, but even so, software delays are endemic and canceled projects are far more common than they should be among both commercial software vendors and defense software contractors.

Table 8 shows the software schedule results for all industries based on application size, using powers of 10 from 1 function point up to 1,000,000 function points:

As can be seen, the risk of schedule delays, canceled projects and cost overruns rises dramatically with application size. Unfortunately, the defense sector has a number of major applications that are between 10,000 and about 300,000 function points in size. The commercial sector has some large packages also, such as the Oracle and SAP ERP packages, in the 250,000-function point range. This explains the significant number of breach of contract lawsuits associated with large applications above 10,000 function points.

Software quality results are also sensitive to application size. Table 9 shows the quality results for the same set, shown by powers of 10.

Table 9 shows average results for all industries. Both the commercial and defense sectors are better than average for quality and typically top 96 percent in DRE, even at 100,000 function points. However, testing alone is not sufficient to achieve high DRE levels. Effective defect prevention, effective pre-test defect removal such as inspections and static analysis, and formal testing by certified test personnel are all needed.

Summary and Conclusions

Because software is the driving force of both industry and government operations, it needs to be improved in terms of both quality and productivity. Today's best combinations of methods, tools and programming languages are certainly superior to older waterfall or cowboy development using unstructured methods, low-level languages and informal defect removal. But even the best current methods still involve error-prone custom designs and labor intensive manual coding.

Both commercial and defense software have done well in quality control. But defense software tends to get bogged down in complex contract procedures that lead to very large volumes of documents and to expensive oversight requirements. These oversight requirements lower the defense software industry's productivity. Professionals in both the commercial and defense software industries need to work hard to reduce cost overruns, schedule delays and canceled projects.

REFERENCES AND READINGS

Abran, A. & Robillard, P.N. (1996, December.) Function Point Analysis, An Empirical Study of its Measurement Processes, IEEE Transactions on Software Engineering, Vol, 22, No. 12. 895-909.

Austin, Robert D. (1996.) Measuring and Managing Performance in Organizations. Dorset House Press, New York, N.Y. ISBN 0-932633-36-6. 216 pages.

Black, Rex. (2009.) Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing. Wiley. ISBN-10 0470404159. 672 pages.

Bogan, Christopher E. & English, Michael J. (1994.) Benchmarking for Best Practices. McGraw-Hill. New York, N.Y. ISBN 0-07-006375-3. 312 pages.

Brown, Norm (Editor). (1995, July.) The Program Manager's Guide to Software Acquisition Best Practices. Version 1.0. U.S. Department of Defense, Washington, D.C. 142 pages.

Cohen, Lou. (1995.) Quality Function Deployment – How to Make QFD Work for You. Prentice Hall, Upper Saddle River, N.J. ISBN 10: 0201633302. 368 pages.

Crosby, Philip B. (1979.) Quality is Free. New American Library, Mentor Books, New York, N.Y. 270 pages.

Curtis, Bill; Hefley, William E. & Miller, Sally. (1995.) People Capability Maturity Model. Software Engineering Institute. Carnegie Mellon University. Pittsburgh, Penn.

Department of the Air Force. (1994.) Guidelines for Successful Acquisition and Management of Software Intensive Systems. Vols. 1 and 2. Software Technology Support Center. Hill Air Force Base, Utah.

Dreger, Brian. (1989.) Function Point Analysis. Prentice Hall, Englewood Cliffs, N.J. ISBN 0-13-332321-8. 185 pages.

Gack, Gary. (2010.) Managing the Black Hole: The Executives Guide to Software Project Risk. Business Expert Publishing. Thomson, Georgia. ISBN10: 1-935602-01-9.

Gack, Gary. Applying Six Sigma to Software Implementation Projects. Accessed online, http://software.isixsigma.com/library/content/c040915b.asp.

Gilb, Tom & Graham, Dorothy. (1993.) Software Inspections. Addison Wesley. Reading, Mass. ISBN 10: 0201631814.

Grady, Robert B. (1992.) Practical Software Metrics for Project Management and Process Improvement. Prentice Hall. Englewood Cliffs, N.J. ISBN 0-13-720384-5. 270 pages.

Grady, Robert B. & Caswell, Deborah L. (1987.) Software Metrics: Establishing a Company-Wide Program. Prentice Hall. Englewood Cliffs, N.J. ISBN 0-13-821844-7. 288 pages.

Grady, Robert B. (1997.) Successful Process Improvement. Prentice Hall PTR. Upper Saddle River, N.J. ISBN 0-13-626623-1. 314 pages.

Humphrey, Watts S. (1989.) Managing the Software Process. Addison Wesley Longman. Reading, Mass.

IFPUG Counting Practices Manual, Release 6. (2015 April.) International Function Point Users Group. Westerville, Ohio. 105 pages.

Jacobsen, Ivar; Griss, Martin, & Jonsson, Patrick. (1997.) Software Reuse - Architecture, Process, and Organization for Business Success. Addison Wesley Longman. Reading, Mass. ISBN 0-201-92476-5. 500 pages.

Jacobsen, Ivar et al. (2013.) The Essence of Software Engineering; Applying the SEMAT Kernel. Addison Wesley Professional.

Jones, Capers. (2014.) The Technical and Social History of Software Engineering. Addison Wesley.

Jones, Capers. (2007.) Estimating Software Costs, 2nd edition. McGraw Hill; New York, N.Y.

Jones, Capers & Bonsignour, Olivier. (2011.) The Economics of Software Quality. Addison Wesley. Boston, Mass. ISBN 978-0-13-258220-9. 587 pages.

Jones, Capers. (1988.) A Ten-Year Retrospective of the ITT Programming Technology Center. Software Productivity Research. Burlington, Mass.

Jones, Capers. (2008.) Applied Software Measurement. McGraw Hill, 3rd edition.

Jones, Capers. (2010.) Software Engineering Best Practices. McGraw Hill, 1st edition.

Jones, Capers. (1994.) Assessment and Control of Software Risks. Prentice Hall. ISBN 0-13-741406-4. 711 pages.

Jones, Capers. (1995, December.) Patterns of Software System Failure and Success. International Thomson Computer Press. Boston, Mass. 250 pages. ISBN 1-850-32804-8. 292 pages.

Jones, Capers. (Due in May of 2000.) Software Assessments, Benchmarks, and Best Practices. Addison Wesley Longman. Boston, Mass. 600 pages.

Jones, Capers. (1997.) Software Quality – Analysis and Guidelines for Success. International Thomson Computer Press. Boston, Mass. ISBN 1-85032-876-6. 492 pages.

Jones, Capers. (1997, April.) The Economics of Object-Oriented Software. Software Productivity Research. Burlington, Mass. 22 pages.

Jones, Capers. (1998, January.) Becoming Best in Class. Software Productivity Research. Burlington, Mass. 40 pages.

Kan, Stephen H. (2003.) Metrics and Models in Software Quality Engineering. 2nd edition. Addison Wesley Longman. Boston, Mass. ISBN 0-201-72915-6. 528 pages.

Keys, Jessica. (1993.) Software Engineering Productivity Handbook. McGraw Hill. New York, N.Y. ISBN 0-07-911366-4. 651 pages.

Love, Tom. (1993.) Object Lessons. SIGS Books. New York, N.Y. ISBN 0-9627477 3-4. 266 pages.

McCabe, Thomas J. (1976, December.) A Complexity Measure; IEEE Transactions on Software Engineering. 308-320.

McMahon, Paul. (2014.) 15 Fundamentals for Higher Performance in Software Development. PEM Systems 2014.

Melton, Austin. (1995.) Software Measurement. International Thomson Press. London, U.K. ISBN 1-85032-7178-7.

Multiple authors. (1996.) Rethinking the Software Process. (CD-ROM). Miller Freeman, Lawrence, Kansas. (This is a CD-ROM book collection jointly produced by the book publisher, Prentice Hall, and the journal publisher, Miller Freeman. This CD-ROM contains the full text and illustrations of five Prentice Hall books: "Assessment and Control of Software Risks" by Capers Jones; "Controlling Software Projects" by Tom DeMarco; "Function Point Analysis" by Brian Dreger; "Measures for Excellence" by Larry Putnam and Ware Myers; and "Object-Oriented Software Metrics" by Mark Lorenz and Jeff Kidd.)

Paulk, Mark et al. (1995.) The Capability Maturity Model; Guidelines for Improving the Software Process. Addison Wesley. Reading, Mass. ISBN 0-201-54664-7. 439 pages.

Perry, William E. (1985.) Data Processing Budgets: How to Develop and Use Budgets Effectively. Prentice Hall. Englewood Cliffs, N.J. ISBN 0-13-196874-2. 224 pages.

Perry, William E. (1989.) Handbook of Diagnosing and Solving Computer Problems. TAB Books, Inc. Blue Ridge Summit, Penn. ISBN 0-8306-9233-9. 255 pages.

Putnam, Lawrence H. (1992.) Measures for Excellence: Reliable Software On Time, Within Budget. Yourdon Press, Prentice Hall. Englewood Cliffs, N.J. ISBN 0-13-567694-0. 336 pages.

Putnam, Lawrence H. & Myers, Ware. (1997.) Industrial Strength Software: Effective Management Using Measurement. IEEE Press. Los Alamitos, Calif. ISBN 0-8186-7532-2. 320 pages.

Radice, Ronald A. (2002.) High Quality Low Cost Software Inspections. Paradoxicon Publishing. Andover, Mass. ISBN 0-9645913-1-6. 479 pages.

Royce, Walker E. (1998.) Software Project Management: A Unified Framework. Addison Wesley Longman. Reading, Mass. ISBN 0-201-30958-0.

Rubin, Howard. (1997.) Software Benchmark Studies For 1997. Howard Rubin Associates. Pound Ridge, N.Y.

Rubin, Howard (Editor). (1998.) The Software Personnel Shortage. Rubin Systems, Inc. Pound Ridge, N.Y.

Shepperd, M. (1988.) A Critique of Cyclomatic Complexity as a Software Metric. Software Engineering Journal, Vol. 3. 30-36.

Strassmann, Paul. (1997.) The Squandered Computer. The Information Economics Press. New Canaan, Conn. ISBN 0-9620413-1-9. 426 pages.

Stukes, Sherry; Deshoretz, Jason; Apgar, Henry & Macias, Ilona. (1996, September 30.) Air Force Cost Analysis Agency Software Estimating Model Analysis. TR-9545/008-2. Contract F04701-95-D-0003, Task 008. Management Consulting & Research, Inc. Thousand Oaks, Calif., 91362.

REFERENCES AND READINGS, CONT.

Symons, Charles R. (1991.) Software Sizing and Estimating – Mk II FPA (Function Point Analysis). John Wiley & Sons, Chichester. ISBN 0 471-92985-9. 200 pages.

Thayer, Richard H. (editor) (1988.) Software Engineering and Project Management. IEEE Press. Los Alamitos, Calif. ISBN 0 8186-075107. 512 pages.

Umbaugh, Robert E. (Editor) (1995.) Handbook of IS Management (Fourth Edition). Auerbach Publications. Boston, Mass. ISBN 0-7913-2159-2. 703 pages.

Weinberg, Dr. Gerald. (1993.) Quality Software Management: Volume 2 First-Order Measurement. Dorset House Press. New York, N.Y. ISBN 0-932633-24-2. 360 pages.

Wiegers, Karl A. (1996.) Creating a Software Engineering Culture. Dorset House Press. New York, N.Y. ISBN 0-932633-33-1. 358 pages.

Yourdon, Ed. (1997.) Death March - The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects. Prentice Hall PTR. Upper Saddle River, N.J. ISBN 0-13-748310-4. 218 pages.

Zells, Lois. (1990.) Managing Software Projects - Selecting and Using PC-Based Project Management Systems. QED Information Sciences. Wellesley, Mass. ISBN 0-89435-275-X. 487 pages.

Zvegintzov, Nicholas. (1994.) Software Management Technology Reference Guide. Dorset House Press. New York, N.Y. ISBN 1-884521-01-0. 240 pages.

ABOUT THE AUTHORS

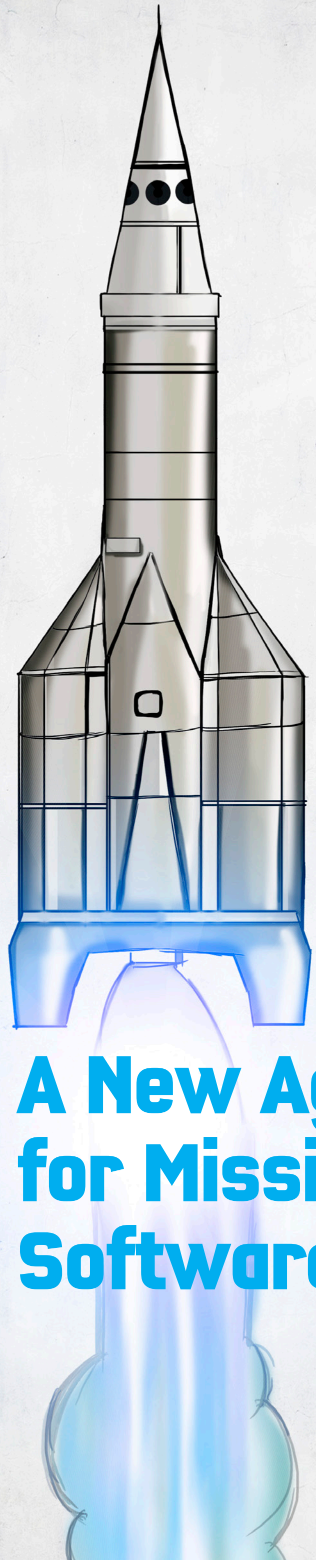
Capers Jones is currently vice president and chief technology officer of Namcook Analytics LLC. Prior to the formation of Namcook Analytics in 2012, he was the president of Capers Jones & Associates LLC. He is the founder and former chairman of Software Productivity Research LLC (SPR). Capers Jones founded SPR in 1984 and sold the company to Artemis Management Systems in 1998. He was the chief scientist at Artemis until retiring from SPR in 2000.

Before founding SPR, Capers was Assistant Director of Programming Technology for the ITT Corporation at the Programming Technology Center. During his tenure, he designed three proprietary software cost and quality estimation tools for ITT between 1979 and 1983. He was also a manager and software researcher at IBM in California where he designed IBM's first two software cost estimating tools in 1973 and 1974 in collaboration with Dr. Charles Turk.

Capers Jones is a well-known author and international public speaker. Some of his books have been translated into five languages. His most recent book is *The Technical and Social History of Software Engineering*, Addison Wesley 2014.

Capers Jones has also worked as an expert witness in 15 lawsuits involving breach of contract and software taxation issues and provided background data to approximately 50 other cases for other testifying experts.

Capers.Jones3@gmail.com
http://Namcookanalytics.com
www.Namcook.com



A New Agile Paradigm for Mission-Critical Software Development

Angelo Messina, Defence & Security Software Engineering Association - DSSEA

Franco Fiore, NATO Communication and Information Agency, Directorate Application Services

Mario Ruggiero, Defence & Security Software Engineering Association – DSSEA

Paolo Ciancarini, Department of Computer Science & Engineering – DISI, University of Bologna & CINI

Daniel Russo, Department of Computer Science & Engineering – DISI, University of Bologna & CINI

Abstract. The Agile paradigm, as intended in the 2001 "Agile Manifesto," brought a disruptive software development methodology. However, with regard to mission- and security-critical organizations, traditional Agile methodologies are quite ineffective because they do not clearly address issues of (1) quality and (2) security. Within the Italian Army General Staff Logistic Department, a new Agile methodology was introduced to tackle quality and security issues in a classified and mission-critical context.

I. Introduction

Continual evolution of the operational environment generates instability of the command and control (C2) systems requirements, obliging developers to work with unstable and unconsolidated mission needs. NATO's new resolute support (RS) mission in Afghanistan is focusing on the training and advising of the Afghan National Defense and Security Forces (ANDSF), introducing a new dimension that transcends the canonical range of military operations. In 2013, the Italian Army General Staff Logistic Department decided to overcome the problem of the "volatile requirement," transitioning to a completely different software development methodology derived from the commercial sector but almost completely new to mission-critical software applications: the so-called "Agile" methodology.

The introduction of Agile in the development of high-reliability software was not easy and required the generation of a brand-new Agile methodology called "Italian Army Agile" or "ITA2." Setting up the LC2Evo (the evolution software of the land C2) required the solution of many problems and the construction of a solid structure based on four principles: user community governance, specific Agile training, new Agile CASE tools and custom Agile development doctrine.

This paper gives two major contributions to the community. First, it is an experience report of the Italian Army C2 system in military scenarios. Second, we outline the new Agile methodology introduced within the Italian Army: iAgile. This paper is structured as follows: Section II gives a major insight about the operational scenario and requirements management. Section III shows briefly the Italian LC2Evo model. In Section IV the new iAgile model is presented with some cost reduction evaluations. Finally, in Section V, we conclude our paper with the presentation of further works.

II. Asymmetric Operations and Volatile Requirements

To run an analysis on the evolution of the C2 requirements, reference is taken to the operation ISAF (Afghanistan 2003 to 2014), in which Italy took part almost since the beginning, and its natural evolution, the operation "resolute support" (RS) (Afghanistan from Jan. 1, 2015 to present). As part of the operation, Italian forces have contributed to the NATO force in Afghanistan and to the Provincial Reconstruction Team.

The initial mission was relatively limited and included providing security for Kabul and its surrounding areas. But as the conflict in Afghanistan continued, more nations, international organizations and non-governmental organizations (NGOs) began various assistance efforts in Afghanistan, and ISAF's mission scope was expanded (UNSCR 1510). In 2003, when Italy joined the operation, ISAF's mandate had been expanded to the entirety of Afghanistan. Many of the assignments of the force on the ground were different from traditional military operations, and so were the connected mission needs.

The first relevant transition in the C2 architecture was caused by NATO's assumption of leadership. Ideally, the coalition network was supposed to transition to a full NATO network, which was almost non-existent at the time. The second relevant change was the geographic dispersion of the forces. At the end of 2006, ISAF expanded its bases of operation throughout Afghanistan, covering the whole territory. Increasing the geographical footprint and adding more members and more non-military organizations made operations radically more complex and unconventional.

The number and type of potential customers of the C2 systems significantly changed as well. Initially, ISAF consisted of roughly 5,000 troops concentrated near Kabul. By 2010, ISAF consisted of well over 100,000 troops from 48 different countries, including NATO, NGOs and Afghan partner institutions. The third relevant change was the "advise and assist" (AA) function, exerted at the regional and central levels, starting at the beginning of 2015. In particular, the AA support at the security institutions level was a new challenge in terms of C2 support. A new specific tool following the Mission Thread approach is in development. Functional military areas to cope with are mainly humanitarian assistance, stability operations, counterinsurgency operations and combat operations.

It is now clear that mission needs are largely unpredictable while the mission support system (hardware and networks) are, for the most part, substantially the same throughout the mission. The volatility of the mission requirements has to be mitigated by the flexibility of an innovative software development process capable of continuity of change. The original concept of network-centric warfare (NCW), ready to deploy in all situations, has been integrated by the "mission-oriented approach" based on mission threads.

III. The Functional Area Service Approach and LC2EVO

The doctrinal reference point to start building an evolutionary C2 system was the mission thread-based approach adopted by NATO with regard to quality and security issues.

A. Scrum is Not Enough

In 2014, the Italian Army General Staff Logistic Department began the development of the Land Command and Control EVOlution system. The product major item is software, while most of the hardware systems are the same supporting standard systems with the addition of some COTS. The main reason for this engagement was the need to support the evolution of the land C2, keeping high customer satisfaction in a volatile requirement situation in a mission-critical context. Another major issue to be solved was the need to substantially reduce the budget necessary for this software development and subsequent maintenance.

The Army software engineers experimented with the principal Agile software development methodology available at the time, "Scrum Agile." This methodology is very successful in commercial environments, where it is the method of choice for a majority of software applications producers, especially Android- and Linux-based products. LC2Evo started by using Scrum Agile with production cycles of three weeks and one experimental "Scrum team" with programmers, subject matter experts, security specialists, a Scrum master and a product owner. The team was composed of both developers from the industry and military people and was based at the Army staff major facility. [1] The initial phase of production was extremely successful, and even the very first sprint (production cycle of three weeks), that was supposed to be only a trial, actually delivered the planned product. While the product became more and more complex and the stakeholders' expectations grew, it became clear that commercial scrum methodology was not capable of handling the peculiarity of a high-reliability software production with an articulated user community as the one in charge of the land C2 operational requirement. [2]

The Army designed a particular Agile software development process called “Italian Army Agile” (ITA)2 and tested it in the LC2Evo production. This methodology is currently shared by people with a broad community of interests, including those in defence industries, people from universities, and software engineers taking part in the DSSEA (Defence & Security Software Engineers Association).

B. LC2Evo’s FAS: A Direct Implementation of iAgile

LC2Evo is based on “core services” and “functional area services” (FAS). Unlike systems in the past, the core services are web-based. FAS are derived by the “mission threads” definition of the ISAF concept of operations and have been adopted by NATO. Both the core services and the individual FAS software components can be separately changed to accommodate particular mission needs defined by the user. At the same time, all the FAS can share the data and the artifacts developed individually, maximizing code reuse.

The LC2Evo has been tested in a NATO exercise for the first time at CWIX 2015 (www.act.nato.int/cwix) with very favorable results. The FAS are under continuous development processes. Some of the most significant are:

Battle Space Management. Originally designed to provide C2 support for homeland security operations such as “Strade Sicure,” battle space management is capable of tracking friendly secure mobile units on various types of cartography. It implements voice and messaging capability and has an integrated NATO JOCWHATCH feature. Most of the code and functions realized for this FAS have been reused for the LC2Evo-Infrastructure FAS, [3] which provides an extensive and detailed set of functionalities needed to manage Army real estate.

Joint ISR FAS. This provides management and analysis functions for the intelligence preparation of the battlefield.

Joint Fires and Targeting FAS. This supports all the coordination and planning activities related to fire power support, including all available effectors.

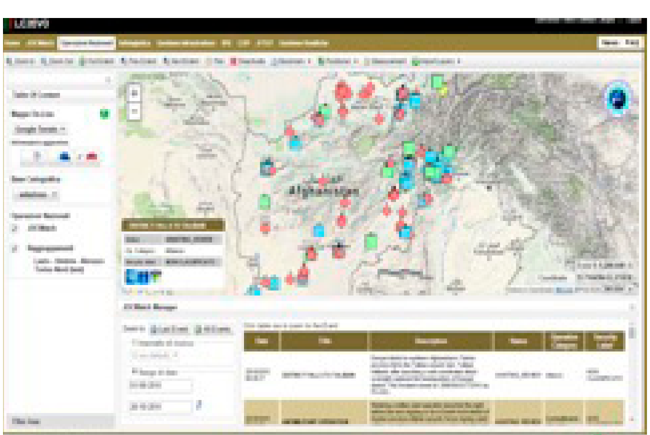


Figure 1. The Battle Space Management: The Events, Simulated Screenshot from Afghanistan

Military Engineer and Counter IED FAS. This was initially designed to support the collection and management of data about unexploded ordinances (UnExO) from World War II found on the national territory, but was soon implemented to provide support to the Counter IED operations (attack the net and force protection).

To comply with SIP’s (Software Intensive Programs - DoD 5000.02) performance principles of NATO, two major Agile principles have to be followed:

- Deliver early and often. This principle is aimed at changing the culture from one that is focused typically on a single delivery at the end of the development phase to a new model with multiple deliveries during development, leading to an ultimate version that supports the full set of requirements supported by the DevOps approach. [4]
- Need for incremental and iterative development and testing. This principle embraces the concept that incremental and iterative development and testing, including the use of prototyping, yields better outcomes than those resulting from trying to deploy large, complex IT network systems in one “big bang.”

These two principles tell us that the old-fashioned waterfall approach in which the customer, after months of software development, is given a release he or she may not be happy with, needs to be replaced by more modern and innovative software engineering techniques and methodologies.

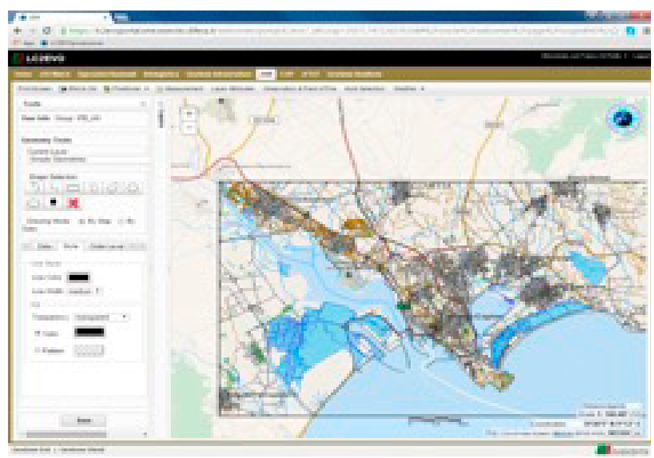


Figure 2. Joint ISR: A Simulated Screenshot of the Intelligence Preparation of the Battlefield

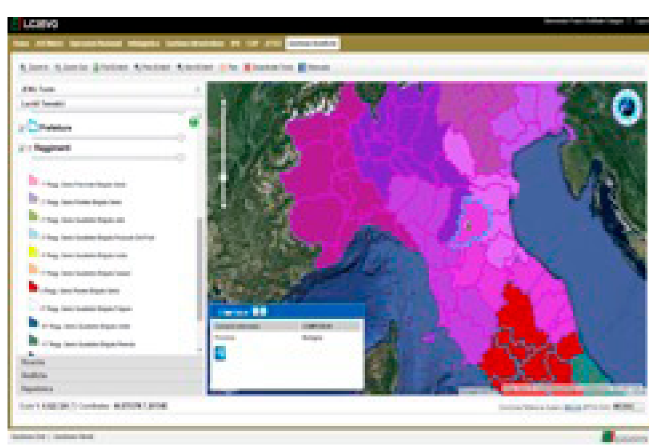


Figure 3. Military Engineer and Counter IED: Simulate Screenshot of the UnExO Areas of Interest.

After two years in the process, the Italian Army General Staff has clearly demonstrated the effectiveness of the new software development methodology, realizing the LC2Evo Command and Control software. The product is a continual development effort that produces new segments every five weeks. The first FAS of the product, the LC2Evo-infrastructure, which was published online in June 2015, serves more than 1,000 users daily and has registered customer satisfaction levels close to 100 percent.

IV. Toward iAgile

The transition to Agile was not only needed to accommodate quicker adaptation to dynamic mission needs changes and quality and security needs but was also mandated by a drastic reduction in the defense budgets experienced in many NATO countries, particularly Italy.

A. Building the Four Pillars

Most of the effort to generate an adequate production structure for the LC2Evo has been devolved to the creation of an innovative cultural and technical environment. Most of the difficulties found during this innovation process were human-based, essentially due to cultural resistance based on consolidated practices. An entire brand-new environment had to be built. The four pillars of this innovative software engineering paradigm are:

- User community governance.
- Innovative Agile training.
- Innovative CASE tools.
- High-reliability Agile doctrine.

User Community Governance Pillar

This is of paramount importance and can be considered a prerequisite for the entire development process. In the area of land command and control, the number and articulation of the reference stakeholders and users is huge. Functions such as the "third dimension control" may have multiple stakeholders and users at the same time — for example, artillery might be using the 3D space to plan its firepower delivery while the same space is used by the Army Light Aviation and the Air Force in joint operations. This situation makes it necessary to rationalize the requirements management. The Army general staff has dedicated a huge effort to creating the coordination lines and the permanent structure to allow an orderly fashion collection of user needs and to provide the availability of subject matter experts to be placed on the development teams. Ad hoc social networks have been designed for this scope.

As stated before, the Agile training easily available from the market was not able not provide the particuliar skills needed to work in the mixed military and industry multidisciplinary teams, and the traditional roles described by the Scrum doctrine, such as the product owner and scrum master, had to be modified to be able to perform in the Italian Army Agile methodology. Within NATO partners, DSSEA is carrying out new training courses to match such specific needs.

Innovative CASE Tools

Replacing the traditional and CASE tools poses a difficult challenge: keeping the momentum of the Agile innovation while implementing new concepts for designing the high-reliability-related software development environments. The core of the Agile methods is the human element, which is positioned at the center of the development process again, using the brain's non-linear capability to overcome the difficulties related to user requirement incompleteness, volatility and redundancy.

Agile methods, properly implemented, can take care of a significant part of this problem by capturing user needs in lists of short user stories and then giving the user the working segments of the product after a few weeks or even days. This way, part of the non-linearity of the requirement conceptual design is overcome by the interaction between humans — the software developer is directly assisted by the user, and they essentially design the application together. In the process, the two different complex 3D representations of the application (run time) imagined by the mind of the user and the one detailed by the mind of the software developer tend to converge. This method also reduces the number of translations needed to convert the requirements into coding tasks, significantly decreasing the loss of relevant information [5].

High-Reliability Agile Doctrine

The last pillar regards the definition of a doctrine as a set of rules and procedures encompassing all the needed practices and artifacts to be used in the implementation of a new Agile methodology. DoD Instruction 5000.02 (Dec. 2013) heavily emphasizes tailoring program structures and acquisition processes to the program characteristics. Agile development can achieve these objectives through:

- Focusing on small, frequent capability releases.
- Valuing working software over comprehensive documentation.
- Responding rapidly to changes in operations, technology and budgets.
- Actively involving users throughout development to ensure high operational value.

These indications are a clear encouragement to use Agile practices to integrate planning, design, development and testing into an iterative life cycle to deliver software at frequent intervals.

Moreover, assuring high code quality for mission-critical applications is a core mission of iAgile. There are two great and opposite tensions for delivering software in such a volatile operational scenario: reliability and velocity. Both are crucial and, apparently, diametrically opposite. The characteristic of iAgile is that the most effort and focus is put on the development. Short and focused sprints are able to provide both velocity, due to time boxing, and reliability, since developers put the most attention into developing high-quality code using reliable and known libraries. Documentation and maintenance efforts are minimal. Development itself is boosted by redundancy by people with different professional expertises. Oriented pair programming in such contexts is useful when one of the components of the pair is a software security expert or a mission-specific application expert. Testing in iAgile has to be performed continually with a Test Driven Development (TDD).

The apparent redundancy of resources does not impact the production effectiveness because of the dramatic reduction of the code rework activity due to errors.

Security is enhanced by both methodology and redundancy. Since the code developed is supervised by a security expert through pair programming, leaks and bugs are fixed before testing. However, penetration testing is used extensively as TDD methodology for security to reveal leaks not fixed in the development phase.

From a quality and security point of view, there is a paradigm shift from "deliver and maintain" to "continuous development."

B. Waterfall vs. Agile: Cost Reduction

The Agile Manifesto in 2001 changed the focus of software engineering, putting the programmers at the center of the production line and creating a straight communication line between the customer (in charge of the requirement) and the developers, minimizing the need for formal documentation. [7] Within the iAgile methodology, every single step of the procedure is monitored, tracked and evaluated by the developers themselves.

An initial internal assessment of LC2Evo product cost per line of code equivalent with respect to other comparable internally produced software showed a cost reduction of 50 percent. To consider those costs, we computed comparable software by dimension (LOC) and functional area (command and control). We considered all relative costs of personnel, documentation and maintenance costs, and fixed costs for office utilities. The assessment after two years showed more significant cost reduction.

Generally speaking, we know from literature that, on average, cost per ELOC in military domains is about $145 and, with regard to ground operation, the cost is about $90. [8] This study in particular was carried out in a "waterfall" or procedural context. Based on Reiter's study, we carried out our evaluation regarding iAgile's cost. It was quite surprising to realize that the measured software LC2Evo had an average cost per ELOC of $10. This was possible because of the decreased maintenance and documentation costs, which represent the most relevant parts of software development cost. [9]

V. Conclusions

The Italian Army experience in developing the command and control software LC2Evo confirms that the major problem in dealing with complex scenarios and rapidly evolving user needs is the management and evolution of the user requirements with both high quality and security standards. The linear development cycles focus on the production process once the requirement is consolidated, but even in the defense and security software applications areas, this is no longer possible. Similar to applications in the commercial sector, military applications experience a quickly changing operational environment. The high reliability, security and quality of the product cannot be pursued with procedures and agents working outside the development cycle, but they have to be embedded in the production cycle.

Standard Agile methodologies seem not to comply with a mission-critical context with high quality and security requirements. Within the Italian Army, a new Agile paradigm that addresses these issues has been developing.

Future work will focus both on the development of a theoretical model and on validation in related domains (e.g., banking sector) with the development of dedicated tools.

Acknowledgements

The authors wish to thank the Italian Army General Staff Logistic Department for the availability of data and the courtesy of some pictures. We also thank the Consorzio Interuniversitario Nazionale per l'informatica (CINI) for the financial support. Special thanks go to Major General Francesco Figliuolo and Col. Franco Cotugno for their support.

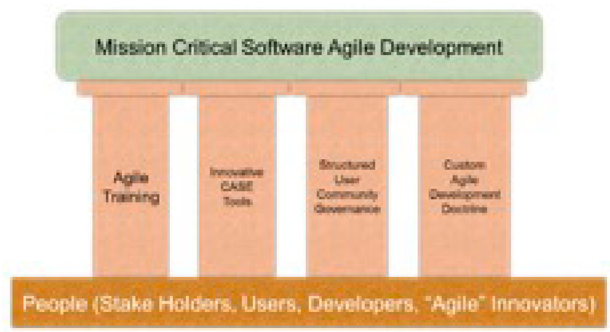


Figure 4. The Four Pillars of Innovation.

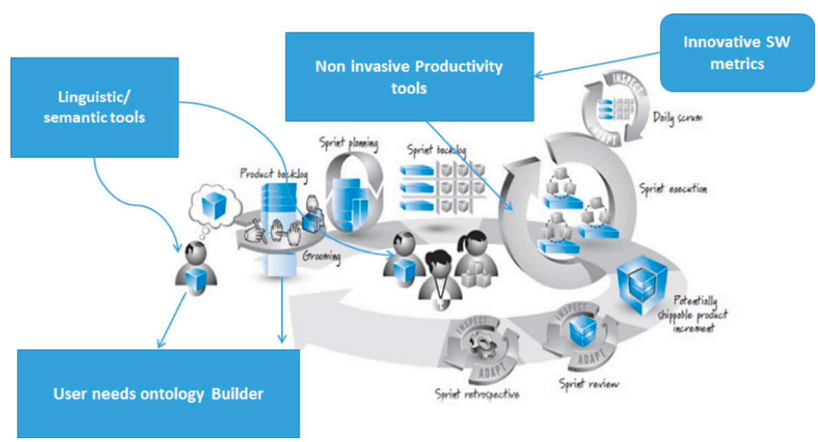
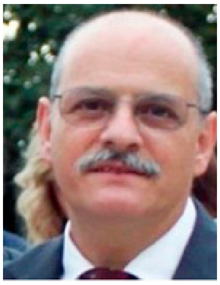


Figure 5. iAgile's Graphic Representation, inspired by [6].

ABOUT THE AUTHORS



Angelo Messina B. General (ret) Ita- Army received a MD Electronic Engineering degree at the Politecnico of Turin and a doctorate degree in non-com, electronics and missile systems. He also attended the Army General Staff Course and Joint General Staff Course. He is secretary of the Defence and Security Software Engineers Association (DSSEA) and an independent consultant in Agile Software Engineering for NATO NCIA. He also serves as senior scientist at CASE Research Ltd. He was previously appointed as Deputy Chief 4th Dep. Army General Staff and head of the 1st Department of the Land Armaments Directorate. In his career, he also served as the European Defence Agency R&T deputy director.

Via A. Bertoloni, 1/E, 00197 – Rome
angelo.messina@esercito.difesa.it

Franco Fiore has been serving as the NATO Communication and Information Systems Agency Service Support and Business Application Service Line since March 2016. As former Italian Army Corps Engineer (EW and Communications), Dr. Fiore spent four years in the USA serving at the NATO Medium Extended Air Defense System Management Agency (NAMEADSMA) as sensor simulation engineer, and joined NC3A (now NCI Agency) in January 2005. He holds master's degrees in computer engineering and complex electronic systems and a Ph.D in telecommunications and electronics.

The Hague, Netherlands
franco.fiore@ncia.nato.int

Mario Ruggiero MG (ret) Ita- Army received a degree in applied military strategic sciences from the University of Turin. After that, he attended the Army General Staff Course, Joint General Staff Course and French Joint Staff Course. He is an active member of the Defence and Security Software Engineers Association (DSSEA). He served as Deputy Chief of Staff Support– Resolute Support Mission (AFGH). He was also chief of the 4th Dep. Army General Staff and the director of the Italian Center for Defense Innovation. In his career, he was also appointed as Chief General Planning Branch at Defense General Staff.

Via A. Bertoloni, 1/E, 00197 – Rome
mario.ruggiero@esercito.difesa.it

Paolo Ciancarini is full professor of computer science at the University of Bologna. He is currently the president of the Italian Association of University Professors in Computer Science.

He is also the vice-director of CINI (National Inter-University Consortium for Informatics). His research focus is on software engineering. He is the author of more than 120 scientific published scientific articles.

Mura Anteo Zamboni, 7, 40126 – Bologna
paolo.ciancarini@unibo.it

Daniel Russo is a research fellow at the department of computer science and engineering at the University of Bologna and a research associate at the Consorzio Interuniversitario per l'Informatica (CINI) in Rome, currently pursuing a Ph.D. in computer science and engineering.

His main research interests include Agile software development methodologies in mission-critical environments and quality and security assurance in mission-critical software.

Mura Anteo Zamboni, 7, 40126 – Bologna
daniel.russo@unibo.it
(Corresponding Author)

The Defence & Security Software Engineers Association is a nonprofit organization aimed at the development of a new software engineering paradigm. The association includes members from the defense and security areas as well as from universities and industry. (www.dssea.eu).

REFERENCES

1. Messina, A. & Cotugno, F. (2014, May.) Adapting SCRUM to the Italian Army: Methods and (Open) Tools. The 10th International Conference on Open Source Systems. San Jose, Costa Rica.
2. Cotugno, F. & Messina, A. (2014, September.) Implementing SCRUM in the Army General Staff Environment. The 3rd International Conference on Software Engineering for Defence Applications – SEDA. Rome, Italy.
3. Ventrelli, C.; Trenta, D.; Dettori, D.; Sanzari, V. & Salomoni, S. (2015, May.) ITA Army Agile Software implementation of the LC2EVO Army Infrastructure strategic management tool. The 4th International Conference in Software Engineering for Defense Applications SEDA. Rome, Italy.
4. Russo, D. (2015, May.) Benefits of Open Source Software in Defense Environments. Advances in Intelligent Systems and Computing, Springer. The 4th International Conference in Software Engineering for Defense Applications SEDA. Rome, Italy.
5. Ciancarini, P., Sillitti, A., Succi, G., & Messina, A. (Eds.). (2016). Proceedings of 4th International Conference in Software Engineering for Defence Applications: SEDA 2015 (Vol. 422). Springer.
6. Rubin, K. S. (2012) . Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison- Wesley.
7. Schwaber, K. (2004.) Agile Project Management with SCRUM. Microsoft Press. ISBN 9780735619937.
8. Reifer, D. J. (2004.) Industry Software Cost, Quality and Productivity Benchmarks. The DoD Software Tech, 7 (2), 3-8.
9. Pressman, R. S. (2009.) Software Engineering: A Practitioner's Approach, 7th Ed. McGraw Hill Int. Press.

Beyond the Agile Manifesto

Epoch of the Team

Chris Alexander

Abstract. The next business epoch will focus on teams and teamwork, specifically the development of high-performing teams, which are characterized by high rates of production, the ability to solve complex problems and create innovative solutions. Agile systems provide processes surrounding how work is done but do not address how team members interact in working together. Beyond process, training and developing teams into high performance will define the best organizations.

The Truth About Teams and Teamwork

Agile frameworks and systems are today becoming a routine part of operational life across an increasing number of industries and business domains. No longer the sole purview of software development or manufacturing, Agile is finding its way into areas as diverse as education, health care, energy, finance and government. Agile is becoming "the standard," and for many organizations, working in an Agile way is "OK."

Yet the advent of such broad adoption brings with it the inherent juxtaposition that as a standard "cost of doing business," Agile no longer holds the allure of being a differentiator or competitive advantage. After all, if every organization can deliver incremental value and react to changing priorities through nimble process workflows, no one holds an advantage.

In the years ahead it will increasingly become the ability of groups within those organizations to perform at the highest levels, in teams, that will enable organizations to compete and succeed in the most complex and competitive environments.

The incontrovertible truth is that teams are far more capable of producing value with quality at a high rate than individuals. Teams are better at solving the most challenging problems, and they are far more capable of conceiving, designing and delivering new innovations.

For many, this contradicts a longstanding mental conception of individuals acting as superheroes responsible for amazing and incredible feats of innovation, problem-solving, design, foresight, vision or discovery. We admire and praise individuals whose "lone wolf" attitude and unbridled individualistic ambition has enabled them to almost single-handedly build an empire from nothing.

Elon Musk. Mark Zuckerberg. Sir Richard Branson. Steve Jobs. Bill Gates. I'm sure you can think of a few names yourselves. Yet we miss the fact that in thinking of a few names off the tops of our heads, we're forgetting the hundreds of millions of people out there working every day who create and contribute amazing work but never reach those levels of acknowledgement or fame — not to mention the fact that most of those individuals were actually part of small but highly effective teams.

Steve Jobs and Steve Wozniak. Bill Gates and Paul Allen. Howard Schultz and Howard Behar. Sergey Brin and Larry Page. Pairs are a powerful and especially captivating type of team, especially given the fact that the most interesting pairs are often comprised from significantly dissimilar individuals. [1]

Nor is it only pairs that have the most significant impacts or deliver the most innovative solutions. At the Kellogg School of Management, Ben Jones and his colleagues conducted an amazing study of 17.9 million research articles spanning five decades and all scientific fields. Their results showed that teams are 37.7 percent more likely than solo authors to introduce innovation into established knowledge domains.

Similarly, the results that Lee Fleming and Jasjit Singh of the University of California, Berkeley found when analyzing more than half a million patented inventions reinforced the same fact. Solo inventors were less likely to discover impactful inventions, were less effective at culling bad ideas and were unable to combine diverse concepts in order to deliver a truly innovative invention when compared to teams. [2]

As Geoff Colvin points out, research from a massive study of over 20 million research papers across 252 fields shows us that in science and engineering alone, the work of a team is 530 percent (yes, 530 percent) *more likely* to be cited one thousand (1,000) times or more than the work of an individual. [3]

As increasingly more teams and organizations deliver amazing results that change our world in profound ways, the ability to work effectively in teams will become a vital skill for individuals in every domain. Moreover, the ability of organizations to enable great teams to form and deliver will become equally crucial. Although Agile can certainly help organizational work structures and business process, as we will see, it is not a surefire way to build effective teams.

Agile Frameworks and Methodologies: 50 Percent of the Solution

I studied Victorian Literature in college. I can clearly recall when I first read the news report that a project was well underway to build a replica of Shakespeare's Globe Theater on the south bank of the Thames in London. I visited the Globe on my second visit to London, and I was absolutely beside myself.

Despite this, upon entering the Globe and seeing the stage for the first time, I recall feeling somewhat deflated (after the initial excitement died down, of course). Today, I understand very plainly why.

The Globe is just a theater. Not just *any* theater, mind you, but a theater nonetheless. It provides a stage and backdrop, lighting, some acoustics, places for the audience — everything necessary for an amazing play or live performance to occur. But it is the actors (and orchestra) who really do all the work and deliver something powerful, memorable and moving.

Agile systems are like The Globe. They provide a significantly better system for overcoming the challenges of complexity than previous systems like Taylorism [4] (also known as "scientific management" and today as "traditional" project management). Agile frameworks enable us to organize in ways that focus our efforts on prioritizing the delivery of the most valuable work first in incremental cycles and incorporating feedback from the customer (internal or external) on the value delivered and the direction to pursue next. Agile systems place the necessary emphasis on the people actually doing the work — a critical aspect to anyone operating in a knowledge-based industry.

However, despite the specific Agile system in application, people working together in teams are required to actually deliver amazing, memorable and powerful results. A framework alone simply sets the stage; teams succeed or fail regardless of whether they are using Waterfall, Scrum, eXtreme Programming, Crystal, DSDM, Lean/Kanban or any other business process.

Agile frameworks and methodologies are, at the end of the day, business processes. They deal with how the work will be organized, prioritized, accomplished and verified, albeit in relatively different and lightweight ways when compared to the heavier processes of traditional project management. However, what Agile frameworks do not address in any direct or meaningful way is how individuals communicate, collaborate and work together. Of course Agile tells us we *need* to do all these things, but in the same way that telling me to paint a painting doesn't help me to be a better painter, simply telling me that I need to collaborate doesn't imbue me with the social skills necessary to do so well.

As a business process, we find ourselves stuck with the same problem which Taylorism also failed to address: while we know what to work on, and we think we know how to work on it, we do not all know how best to work with the human beings right beside us.

In the same way that the ability to reorganize how work should be accomplished was the key advantage of Agile systems over older Taylor-esque project management, we are now faced with a similar challenge in that Agile has reorganized our work process, but not our ability to work optimally together.

Organizations of various types all over the world have transitioned their processes into Agile ways of working, and in doing so have met with varying degrees of success or, at times, outright failure. [5] The simple truth is that not every individual possesses the social, interactive skills necessary for effective teamwork. Perhaps nowhere is this more prevalent than in the world of software development, but it is in no way confined solely to any single domain.

Agile systems universally advocate working in teams, but as anyone who has ever been a part of a successful, high-performing team knows, there is a considerable difference between a group of individuals trying to work as a team and teamwork.

What is missing in Agile systems — indeed, in every process-based system in which teams work — are the key ingredients necessary for individuals working together in teams to perform, problem-solve and innovate at the highest levels. What's missing is a set of human, interactive social skills that enable those working in teams to actually achieve effective teamwork.

The process frameworks and methodologies are there. The stage is set. We just need individuals who possess the skills necessary to be great teammates to take the stage and create something amazing. Most of us know from experience that not everyone possesses the human interactive skills necessary to be a great team member, but years of research and a growing body of empirical evidence show us that while not everyone is born with those critical teaming talents, the overwhelming majority of people can, in fact, learn them.

We Can Learn the Skills that Power Great Teams

"Growing numbers of companies have discovered what the military learned long ago: that the supposedly ineffable, intractable, untrainable skills of deep human interaction are, in fact, trainable." [6]

Decades of research in cognitive and behavioral psychology, social sciences, business and management, and teamwork have taught us that there is one skill which underpins every interaction we have as social, interactive beings: empathy. Empathy, simply put, is the ability to recognize and respond in an appropriate way to the emotional states of others and is something for which our brains have specifically evolved. [7]

When another person walks into the room (think of your spouse, coworker, boss, or best friend), most of us immediately develop a sense of their psychological and emotional state without consciously attempting to do so. That's empathy, or at least part of it. The second part evidences itself in how we respond to the emotional states of others we encounter.

How do you respond to your husband, wife, or child as they emit various emotional states? Comforting a distressed child with a boo-boo, rolling our eyes at their exuberance with a new toy, or knowing that our coworker is always irritable early in the morning and that we shouldn't take it personally because they'll come around once they've had their coffee — these are all examples of basic empathy in action, and it is the skill which underpins all of our interactions.

Meg Bear, a former Oracle executive and senior vice president at Imperva, calls empathy "the critical 21st-century skill." [8]

Although empathy is an individual characteristic, as an innate part of our biological nature, its ability to help us understand and respond to our fellow humans is precisely the reason it is so important to working in teams. It acts as a conduit for the tone and tenor with which we communicate. For example, "It's not so much what you say as it is *how* you say it" is an oft-heard axiom in the world of communications.

Especially in terms of our ability to communicate among team members, empathy is a critical component in enabling effective communication while a lack of empathy challenges our collective ability to communicate well. Those communication networks within teams are one of the foundational components that enable a team to perform.

Alex Pentland, an MIT researcher who has dedicated years to researching teamwork and team performance, informs us that team performance, in terms of productivity, problem-solving and innovation, hinges upon what is known as "collective intelligence." What's more, many of the standard enablers of team performance advocated by business schools and management consultants, such as group cohesion, motivation and satisfaction, were not statistically impactful on the collective intelligence of teams. [9]

What was the basis for teams' collective intelligence? The number and frequency of their conversational turn-taking — in other words, their communication patterns. Moreover, the collective intelligence of the teams studied was independent of the individual intelligence of the team members, and the teams' problem-solving abilities, creativity and innovation were likewise not only independent of the individuals' characteristics but also directly related to the communication patterns and idea exploration among them. [10]

Training teams in the social skills necessary to communicate and collaborate effectively is something that high-reliability organizations like the U.S. military, civil aviation and NASA have been doing for decades. The lessons they have learned, literally at the cost of thousands of lives, can be applied to organizations all over the world.

Arriving on the advent of several easily preventable and wholly unnecessary airline disasters in the 1970s, the commercial aviation community and NASA developed what would become known as Crew Resource Management, or CRM. CRM has also been adopted by the U.S. military and is a routine part of initial and yearly training for virtually any pilot or aircrew involved with commercial or military aviation.

Although in general CRM deals with the ability to rapidly respond to changing situations, a significant portion of its focus is on building effective teams. Former airline captain Chesley "Sully" Sullenberger discusses the significance of these aspects of CRM in his book "Highest Duty." [11] Those team-building lessons from

CRM are today making their way into other domains including oil and gas, nuclear power, and parts of health care organizations, such as surgical teams and emergency rooms.

The vital need for surgical and emergency room teams to be effective is self-evident. Patient safety, recovery and survivability are directly linked to the ability of the teams responsible for those patients' care to work together to solve critical problems under sometimes challenging circumstances.

TeamSTEPPS, as an example, is a team-training framework for health care organizations that has been adapted from CRM. As the creators of TeamSTEPPS acknowledge, "Over twenty years of research and experience in other high-risk industries, such as aviation and nuclear power, showed that teambased collaboration and communication had a positive effect on organizational safety, and that individuals could develop teamwork competencies through carefully designed team training programs." [12]

While not all of us live and work in such mission-critical disciplines as aviation, surgery or space exploration, the important and uplifting lesson we can acknowledge is that the skills that enable the most critical teams to perform effectively are not unique to any specific industry, domain or organization. Rather, they are uniquely human skills that we can all learn and apply in whatever context we're working.

What's more, there are decades of knowledge already in existence that tell us what those skills are. Communication is one, of course. The ability to clearly send and receive information via verbal or nonverbal means, while verifying both understanding and accuracy, is important.

At the coaching and training company I've co-founded, AGLX Consulting LLC, we have adopted additional skills from CRM to build what we refer to as High-Performance Teaming™, or HPT. In addition to communication, HPT includes assertiveness, situational awareness, goal analysis, decision making, agility, leadership, and of course, empathy.

Everyone possesses each of these skills to some degree. However, people also have their own ideas of what's important about each of those skills, and many haven't thought of the challenges and hindrances to the effective implementation of those skills in a team environment.

By addressing these skills in a comprehensive way, we are constructing a type of known, stable interface through which individuals can channel their social interactions within a team. Through such frameworks, teams are able to establish shared mental models for the specific behaviors and characteristics that will form their interactions.

Additionally, as more and more teams are trained in those same social skills, the ability of teams to adjust their membership without impacting performance increases significantly. In essence, through the construction of these shared mental models, entire organizations will become more adaptable and flexible.

Psychological Safety and the Future of Teaming

Several characteristics will dominate the organizations — and teams — of the future. Not only will teams continue to professionalize and optimize their ability to perform and innovate through the adoption of formalized social skills training, but organizations themselves will also continue to shift away from trying to control teams and move toward means of supporting teamwork.

The fundamental reason for this paradigm shift away from the structures of Taylorism is the foundational need for teams to achieve high performance, the conditions of which begin with the business environment in which they operate. In order for teams to solve the most challenging problems and deliver the most creative innovations, they need the organizations in which they work to enable them to learn.

Complex problem solving — and innovation specifically — require organizational cultures that celebrate failure and focus on learning from it. Particularly in older business cultures, failure is typically viewed as something like an unforgivable sin. Yet when facing the most challenging problems, running experiments, learning from failure and incrementally adapting your approach are often the only ways to achieve progress.

To be sure, failure shouldn't be the objective, and we should always aim to succeed wherever we're able. One criticism often repeated today is that the technology startup world seems to be striving to fail. Eric Ries explained the concept of "failing fast" very well in "The Lean Startup," [13] but it was and remains a lesson too often misconstrued.

The lesson of "failing fast" is that we should learn as much as possible from failure *when it occurs*, but we shouldn't actually *plan to fail*. We should always aim for success in our plans through product and business analysis combined with a fair dose of risk management.

One of the great lessons of the "fail fast" mindset is that it affords teams within such an organization something that is critical to enabling them to reach high performance: psychological safety.

In November 2015, Google released a report of one of its internal initiatives known as "Project Aristotle." [14] After a two-year internal study of teams at Google, Project Aristotle discovered much of what we've already covered here. Google's initial hypothesis was in keeping with its big data and engineering culture — that if you analyze the traits and characteristics of the individuals comprising various teams and identify the traits necessary for the highest performing teams, you have yourself a winning recipe for building great teams.

Unfortunately, it didn't work. What they found was what much of the research we've covered already confirms — that the real key ingredients are the interactions of the team members and the structure of their work. The No. 1 characteristic Google identified, and that they asserted made the most difference in whether a team could be high-performing, was the presence of psychological safety.

Amy Edmondson, in her book "Teaming," also found psychological safety to be critical to team success. According to Edmondson, psychological safety is "a climate in which people feel free to express relevant thoughts and feelings." [15] Based on the knowledge we now share about (1) the fact that the highest-performing and most innovative teams possess a collective intelligence that is greater than the sum of their individual members, and (2) that said collective intelligence is constructed through the number and quality of social interactions between the team members, and (3) the skills necessary to power those social interactions can be trained and improved, (4) the final piece is the realization that members of those teams need to feel that they are operating in an environment in which they can not only do their best work but also learn from their failures.

"Expressing relevant thoughts and feelings," as Edmondson aptly puts it, expresses everything we've learned about empathy, communication and the collaborative inputs to teamwork. In knowledge work, where innovative teams solve complex problems, the ability to challenge assumptions, analyze goals, make decisions and learn from failure will be the factors that determine team and organizational success or failure.

Conclusion

To a far greater extent than most teams and organizations realize today, the ability to develop and train high-performing teams is a skill already residing in the world around us. In today's increasingly complex and chaotic world, the current, narrow focus on business processes will continue to result in the occurrence of high-performing teams as a factor of chance or happenstance. Agile systems and frameworks set the stage for great performances, but only a select few teams will ever achieve the fullness of their potential as long as we continue ignoring the vast lessons that researchers and practitioners have to offer.

Great teamwork can be built. The biggest challenges in productivity, problem-solving and innovation can solved by high-performing teams. By leveraging the lessons from researchers, industries and the high-performing teams and organizations around you, your teams can also move beyond basic implementations of Agile systems and the challenges that a 50-percent solution deliver.

REFERENCES

1. Karlgaard, Rich & Michael S. Malone. Team Genius: The New Science of High-Perform-ing Organizations. Harper Business Publishing. iBooks Edition, 130.
2. Ibid., 130-1.
3. Colvin, Geoff. (2015, August 4.) Humans Are Underrated: What High Achievers Know That Brilliant Machines Never Will. Penguin Publishing Group. Kindle Edition, 121-2.
4. Taylor, Frederick Winslow. (2006.) The Principles of Scientific Management. Cosimo Classics. New York.
5. 10th Annual State of Agile Report. (2016.) VersionOne. http:/stateofagile.versionone.com.
6. Colvin, Geoff. (2015, August 4.) Humans Are Underrated: What High Achievers Know That Brilliant Machines Never Will. Penguin Publishing Group. Kindle Edition, 204.
7. Gazzaniga, Michael. (2008.) Human: The Science Behind What Makes Us Unique. HarperCollins.
8. Bear, Meg. Why Empathy is the Critical 21st Century Skill. https://www.linkedin.com/pulse/20140424221331-1407199-why-empathy-is-the-critical-21st-century-skill
9. Pentland, Alex. (2014, January 30.) Social Physics: How Good Ideas Spread – The Lessons from a New Science. Penguin Publishing Group. Kindle Edition, 88.
10. Ibid., 91-3.
11. Sullenberger, Chesley "Sully." (2009.) Highest Duty: My Search for What Really Matters. HarperCollins.
12. Developing and Enhancing Teamwork in Organizations: Evidencebased Best Practices and Guidelines. (2013.) JB SIOP Professional Practice Series. Wiley. Kindle Edition.
13. Ries, Eric. (2011.) The Lean Startup. Crown Business.
14. Google, Inc. The five keys to a successful Google team. https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/
15. Edmondson, Amy C. (2012.) Teaming: How Organizations Learn, Innovate, and Compete in the Knowledge Economy. Wiley. Kindle Edition, locations 21412144.

ABOUT THE AUTHOR

Chris Alexander is a former naval officer who flew and instructed in the F-14 Tomcat and served abroad as a foreign area officer. He is a full-stack web developer, Agile coach, scrum master, and is the co-founder of AGLX Consulting, LLC. He specializes in high-performance teaming, the training and coaching of individuals, teams and organizations in the social, interactive skills necessary to achieve high performance. He currently resides in Seattle, Washington.

1058 S Director St.
Seattle, WA 98108
chris@aglx.consulting
http://www.aglx.consulting

Upcoming Events

Visit ‹http://www.crosstalkonline.org/events› for an up-to-date list of events.

IoCon 2017
January 9-12, 2017
San Diego, CA
http://www.cert.org/flocon/index.cfm

RAMS 2017
Jan 23-26, 2017
Orlando, FL
http://www.rams.org/

2017 IEEE 11th International Conference on Semantic Computing (ICSC)
Jan 30-Feb 1, 2017
San Diego, CA
ieee-icsc.org/icsc2017

2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)
Feb 20-24, 2017
Austin, TX
http://cgo.org/cgo2017/

Developer Week Conference + Festival 2017
Feb 11-18, 2017
San Francisco, CA
http://www.developerweek.com/

MobiSecServ 2017: Third Conference on Mobile and Secure Services
Feb 17-18, 2017
Gainesville, FL
http://perso.telecom-paristech.fr/~urien/mobisecserv/mobisecserv2017/

ICASSP 2017
March 5-9, 2017
New Orleans, LA
http://www.ieee-icassp2017.org/

Software Solutions Symposium 2017
March 20-23, 2017
Arlington, VA
http://www.sei.cmu.edu/sss/2017/index.cfm

Software Engineering Institute (SEI) Architecture Technology User Network Conference (SATURN) 2017
Denver, Colorado
1-4 May 2017
http://www.sei.cmu.edu/saturn/2017/

Too Agile For My Own Good

Short quiz – What did this column used to be called in the early days of CrossTalk?

a. Trick question – it was always called Backtalk
b. Software Engineering Chatter
c. Curmudgeon's Corner
d. The Lighter Side of Software

-Yep – c. As I have mentioned before – a curmudgeon is "a bad-tempered or surly person." This is a much better definition that "a crusty, ill-tempered, and usually old man" – because I AM NOT old yet. Lately, however, the "ill-tempered or surly" seems to occur with increasing frequency.

The university I teach at is in a relatively small Texas town (about 33,000) – and when you say you're going shopping; you don't have to say where. Groceries, clothing or whatever – you're going to WCENSOREDt. Where else WOULD you go? Meet friends, shop for anything you can think of, go for milk, bring back two full carts of stuff.

Several months ago, for some reason they rearranged several aisles as part of their "ongoing efforts to improve service". I fail to see how switching the bread from the right side of the "Bread, Jelly and Peanut Butter" aisle to the left side (and moving the jelly and peanut butter from the left to the right) improves service a bit. But I adapted. The first straw (that broke the curmudgeon's back) was rearranging the "Coffee and Tea" isle. First of all, they quit carrying my favorite blend of coffee. **Strike 1**. They carried so many new blends of coffee that they split the tea into two DIFFERENT aisles. **Strike 2**. And my favorite iced tea blend (Peach-Mango Green Iced Tea) which I have been drinking for years? Somehow it's in aisle 11 ("Sports Drinks") instead of aisle 6 – "Coffee and Tea". **Strike 3**. I have asked the "aisle manager" why – and he agrees with me that all of the iced tea on aisle 11 should be on aisle 6 – but there's no room there! "Why then" I ask, "did you move everything"? "Corporate policy" he said. Seems that moving everything around occasionally forces shoppers to spend a few minutes looking, where they inevitably find something new. It improves sales.

A few weeks later, I was shopping for bananas. I like my bananas slightly green and very firm. I have a banana most mornings for breakfast, and usually one for a mid-day snack. They are PLU (Price Look-Up) code 4011 if you self-checkout. Except that the produce rack where I have located bananas for 7 straight years now was filled with sweet potatoes and yams (PLU 4546 for the potatoes, 3275 and 3276 for yellow/white yams). Where, pray tell, did them move the bananas? To a new aisle, at the bar end of produce. Seems so many people shop for bananas, making folks walk through produce for them is a great marketing strategy.

I buy bananas every time I go grocery shopping. Instinctively enter the store, immediately turn right, buy bananas. It upset my routine! As a good curmudgeon should, I complained every time we went shopping (oddly enough my wife starting shopping without me – a coincidence, I suppose).

Except that rearranging the produce aisles brought in a better variety of items – pre-sliced melon. A wider selection of fruit. Better selection of potatoes (those little microwavable bags – perfect for supper). I don't complain about the bananas anymore.

Remember the old days of 2167 and 2167a? Agile methods sure look good now, don't they. Usable products as opposed to

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

delivery of six cycles of requirements analysis? Stand-up morning meetings lasting 10 minutes, as opposed to 3-hour long weekly status meetings? Product over process, happy customers over endless (and unread) documentation? Have you actually read the Agile Manifesto?

Now I'm as process-oriented as the next software engineer, but you have to admit that those left-hand items make for satisfied customers. Agile is as much an idea and an ideal. Remember that Agile is **not** a set of methodologies. The Agile movement seeks alternatives to traditional project management. Agile approaches help teams respond to unpredictability through incremental, iterative work cadences and empirical feedback. Agilists propose alternatives to waterfall, or traditional sequential development. It's meant to be flexible.

What comes after Agile? Agile doesn't really have an "after" – it just keeps going, It seeks alternatives that work. But if you really need an answer - **Whatever works for you and your organization!** Software development is a dynamic process. Languages come and go – why shouldn't your practices? Applications and systems change frequently – perhaps a static process doesn't give you the happy, satisfied customers you want. Don't embrace random or meaningless change – but don't be scared to give potentially profitable ideas a try. I remember hearing a developer agrue with me back in the 70s that terminals were a fad – all you'll ever need is a card punch, a card reader, and a line printer. Keep what works, discard what doesn't.

Who knows – on the way to the bananas, you might find out that Honeycrisp apples (PLU 3283) are much better than Galas (PLU irrelevant).

David A. Cook, Ph.D.
Professor, Stephen F. Austin State University
cookda@sfasu.edu

P.S. While relocating the bananas ultimately turned out to be a good change (for me), the separation of the iced teas still bothers me. So much so, that I view it as my duty to complain to somebody every time I shop. As a good curmudgeon should!

CMMI Drives Agile Performance

Agility Depends Upon Capability

Companies are increasingly turning to CMMI to improve performance of agile initiatives

CMMI is being used increasingly around the world to build scalable, resilient, high performance organizations and empower those organizations to deliver on the promises of agile methods. Organizations leverage the CMMI to scale and strengthen agile methodologies and address business problems outside the scope of agile methods. Globally, organizations are discovering that CMMI is a necessary companion for successful agile implementations. CMMI is lifecycle agnostic, and is useful with any software development methodology. With widespread, growing adoption in government and commercial sectors, CMMI has become a *defacto* model for performance improvement for software and systems engineering organizations throughout the world.

Successful organizations need both agility and stability

In "Agility: It Rhymes with Stability" (cite: http://www.mckinsey.com/business-functions/organization/our-insights/agility-it-rhymes-with-stability), Wouter Aghina, Aaron De Smet, and Kirsten Weerda argue that truly agile organizations must be both stable and dynamic. To achieve this necessary combination of stability and speed, organizations must "design structures, governance arrangements, and processes with a relatively unchanging set of core elements—a fixed backbone. At the same time, they must also create looser, more dynamic elements that can be adapted quickly to new challenges and opportunities."

Organizations are embracing the combination of CMMI and agile methodologies to achieve this seemingly paradoxical combination that creates true organizational agility. The discipline, organizational learning, and consistency provided by the adoption of CMMI supports organizations in making their agile methods even stronger and more effective.

The CMMI provides a framework or map of "what" a high performance organization must do. Agile methods provide particular approaches that prescribe "how" to do it. As methods and techniques are adapted and evolve, the CMMI provides the foundation upon which organizations can iterate or tailor their techniques in a way that is appropriate to the dynamics of their business environment. For software engineers, a simple analogy would be to think of the CMMI as the "requirements" or "story points" for their organization and various agile ceremonies or techniques as a particular instantiation of those requirements.

Organizations are using CMMI to address common problems with agile projects by mapping ceremonies to the CMMI framework as shown in this illustration:

Problems	Agile Ceremonies /Techniques	CMMI Process Areas
REQUIREMENTS ARE ALWAYS CHANGING.	• Backlog Grooming • Sprint Planning • Product Backlog • User Stories/Epics • Definition of Done (for User Stories)	• Requirements Management • Project Monitoring and Control • Requirements Development

TIP: A project should have a Product Backlog to track all user stories (requirements). The Product Owner (customer) is responsible for keeping this backlog up to date, so if requirements change the backlog should be updated. During each Sprint Planning session, user stories are selected to be included. When Sprint Planning is completed, the Sprint Backlog should be frozen. Any changes that come in during the Sprint/Iteration should go into the Product Backlog. During a project, there should be regular backlog grooming to continue breaking down user stories for upcoming sprints.

Organizations use CMMI to improve agile performance

Agile organizations struggling with issues of performance are increasingly turning to the CMMI for proven results. The CMMI provides a framework to look beyond mere team performance to apply lean principles at the system level. For example, Minacs IT Services experienced a 30 to 40 percent increase of attaining sprint commitments, a 30% increase in the number of user stories delivered in each sprint, and 40% increase in on-time delivery after applying CMMI to existing agile processes. Minacs also transformed its internal work culture from silo-heavy to unified and aligned to a single common vision.

Organizations use CMMI to identify performance gaps in their processes and operations, and to provide a baseline for continuous improvement based on industry best practices. By addressing those gaps, organizations build the stability with CMMI to be more agile in their projects and programs and cut costs, improve quality, and improve on-time delivery.

CMMI helps to scale and sustain agile across the organization

Organizations leverage CMMI as a platform to scale, align and unify operations across geographically distributed operations of large multinationals. Cognizant has sustained a CMMI maturity level 5 rating, and uses the CMMI framework with agile methods to encourage process improvement across its globally distributed organization to meet customer-centric business objectives.

The discipline, organizational learning, and consistency provided by the adoption of CMMI practices allows organizations to use CMMI to make their agile methods even stronger and more effective. In fact, Honeywell India used CMMI and agile across their enterprise with 7,000 engineers to improve problem-solving skills and resolve issues earlier in the development process. Results included a 12-15% decrease in functional defects; 15% improvement in implementation of Kaizen strategy and a shortened learning curve for employees.

CMMI is rapidly growing in global adoption with firms using agile methods

In 2015 alone, CMMI adoption grew 17% globally with 28% growth in the US. In 2015, more than 1,900 high-performing organizations earned a CMMI maturity level rating. With adoption in over 100 countries and a world-class Net Promoter Score of 41, organizations deploying CMMI are very pleased with the results they are achieving.

Adoption of CMMI in organizations implementing agile methodologies is steadily increasing. In 2015, over 70% of CMMI appraised organizations reported using one or more agile methodology. [Sourced from CMMI Institute appraisal records]. Multinational companies with technology centers in China, India and Latin America are using CMMI to scale agile practices and export their capability into geographically distributed operations. For example, CMMI and agile methods are used harmoniously at Perficient Chennai, where the organization was able to reduce defects on projects by 70%. Nearly 85% of the organization's project teams have adopted CMMI maturity level 4 and 5 practices along with agile methods for predicting a project's performance and velocity.

A platform for government and private-sector firms both large and small

While CMMI continues to have a strong footprint in the Aerospace and Defense industries with users ranging from GE Aviation, Boeing, Lockheed, Northrup Grumman, BAE Systems, and Raytheon, its most rapid growth is in commercial sectors where 90% of CMMI adoption is found in commercial sectors such as mobile, finance, telecom, and IT services, at firms such as Honeywell, Samsung, Ericsson, and Fujitsu.

While CMMI is relied upon heavily by large-scale multinational operations, the highest adoption is among small, high performance business units. In fact, 68% of organizations that implement CMMI have fewer than 100 employees and 22% of CMMI-appraised organizations have fewer than 25 employees.

CMMI Institute advances research to improve organizational performance around the world

In 2012, after decades of increasing commercial and government adoption, the CMMI Institute spun out of the Software Engineering Institute. This change in structure leaves the CMMI Institute better able to execute its larger mission: advancing research in operational best practices and elevating organizational performance for the global community.

Since the transition, the CMMI Institute has greatly expanded the industries and global perspectives that contribute to its research, model development, and strategic direction. The Institute is actively collaborating with leading organizations around the world to advance the state-of-the-practice and help deliver on the promise of the agile manifesto to cultivate genuinely dynamic and adaptive high-performance organizations.

Learn more at http://cmmiinstitute.com/cmmi-and-agile

CrossTalk thanks the above organizations for providing their support.