Wazee Group
Software Construction & Renovation

Evaluating "Team of Teams" Concepts for Large Scale Software Development

Jim Ladd
Wazee Group, Inc.

October 22, 2018

Introduction

This document examines the concepts presented in the "Team of Teams" book by General Stanley McChrystal and evaluates them for large scale software development. In 2003, General McChrystal took command of the Joint Special Operations Task Force. At that time, the Task Force was a hierarchical, highly disciplined organization of thousands of men and women. To defeat Al Qaeda in Iraq, the Task Force leadership had to transform this command and control structure based on decades of refinement and principles to an agile, responsive, and adaptable network, a Team of Teams.

Evolving for the New Paradigm

In the spring of 2003, the American-led coalition invaded Iraq with the objectives of neutralizing the weapons of mass destruction, ending the reign of Saddam Hussein, and to free the Iraqi citizens. The initial invasion phase was swift and successful. However, by the fall of 2003, the campaign had changed into a struggle with the Al Qaeda in Iraq (AQI).

At this time, AQI's acts of terrorism were successful, their organization was organic, adaptable and resilient. By using, at that time, emerging technologies, the AQI remained decentralized and coherent even has they grew in size. While the Task Force was stronger, more efficient, and more robust, AQI was more agile and resilient. The command and control structure that served the military so well in the past was inadequate in the new paradigm.

As the efforts wore on, the Task Force began diagramming on whiteboards what they knew about AQI, what they did not know, and what they could only suspect. Soon, these whiteboards were filled with the relationships between entities but not in the familiar hierarchical structures but rather with crisscross, tangled web-like connections. AQI was a network.

General McChrystal and his staff realized that the command and control structure was too slow and rigid to challenge the AQI network. They would have to pursue adaptability and agility at the cost of efficiency. One of the whiteboards had the insightful statement

of "It Takes a Network to Defeat a Network." To be successful against AQI, the Task Force would have to evolve into their own network.

To state a goal is one thing, to actually achieve it is quite different. In 2004, there was no handbook, no roadmap, no Standard Operating Procedure for morphing from a command and control structure into an agile, adaptive network. For decades, teams have been considered critical to the armed services. According to General McChrystal, "The competitive advantage of teams is their ability to think and act as a seamless unit". This advantage would be leveraged and teams would play a critical role in the new network.

Teams in the military have traditionally been deployed in the command and control structure. Intra-team relationships and behaviors represented a mini-network that relied on the bonds of purpose and trust. Interactions with external entities followed the command and control hierarchy. Although more flexible than the conventional command and control structure, this "command of teams" could not deal with the challenges of the AQI network. A more flexible, agile paradigm was needed.

What the Task Force desired was an organizational architecture with the characteristics of a single cohesive team that could scale to thousands of personnel. What they created was a "team of teams" (ToT). To forge their new network of teams, the leaders of the Task Force would have to bind the traditionally isolated teams together with purpose and trust. Teams would need to share a holistic understanding of their network without losing their specialized expertise and skills. General McChrystal named this state of emergent, adaptive, organization intelligence, a "shared consciousness".

Shared Consciousness

To achieve this shared consciousness on the desired scale, General McChrystal writes that it "demanded the adoption of extreme transparency throughout our force and with our partner forces". This transparency provided every team with an unobstructed, constantly up-to-date view of the rest of the organization and was based on strict, centralized forums of communications.

The leaders of the Task Force had to discard the traditional "need to know" practice of limiting information flow direction and velocity. This process began with the simple act of including more recipients on the cc of emails.

Next in the transformation was the redesign of the physical space and processes of the Task Force's Joint Operations Center (JOC). A bank of screens at the front of the space showed live updates of missions, maps of advances and retracements, log entries on captures, etc. Anyone in the room, regardless of rank or responsibility, could glance at the screens and instantly know the real-time status of the mission. The entire space was designated a top secret security space and almost any document or conversation relevant to the operations could be discussed openly.

The most importance component of the transformation was the daily Operations and Intelligent (O&I) brief. In normal military deployments, the O&I is a regularly scheduled meeting to integrate the operations (i.e. everything the command is doing) with the intelligence (i.e. everything the command knows). General McChrystal greatly expanded the number and range of participants, any member of the Task Force and any of the partners invited, could securely dial in to the meetings and listen. The meetings were scheduled for 6 days a week and were never cancelled.

Trust

To present why rational individuals and organizations might not cooperate even if it appears to be in their best interest, General McChrystal describes the popular example of game theory called Prisoner's Dilemma. In this example, two members of a criminal gang are arrested. The two prisoners are separated and unable to communicate with each other. Each prisoner is given the opportunity to either betray the other by testifying that the other one committed the crime or to cooperate with the other by remaining silent. The scenarios are:

- If A and B each betray the other, each serves two years in prison
- If A betrays B and B remains silent, A will be set free and B will serve three years
- If A and B both remain silent, both will serve one year in prison on a lesser charge

The dominant strategy is that regardless of B's decision, prisoner A should always choose betrayal. For example, if prisoner B remains silent and A betrays B, A will be set free rather than serving one year. If prisoner B betrays A and prisoner A betrays B too, A will serve two years rather than serving three. The dilemma is that mutual cooperation yields a better result (one year in prison) than mutual self-interest (two years in prison) but it is not the rational path from a self-interest perspective.

The leadership of the Task Force had to overcome the challenge of the Prisoner's Dilemma. Each partner feared that sharing intelligence would work against its own interest. Competition rather than cooperation made the agencies reluctant to provide information. The leaders explained how cooperation could be beneficial for everyone but trust needed to be established, relationships in the team network had to be built.

While putting everyone in the Joint Operations Center was a great step forward, the Task Force used embedding and liaison programs to establish the relationships required for true transparency, sharing, and cooperation across the organizational silos. Embedding consisted of taking an individual from one team and assigning him to a different part of the Task Force for a period of six months. For example, an Army Special Forces member may be embedded with a SEAL team. As the new, modified team fused together, both the new member and the host unit would learn from each other and develop a trusting relationship that would extend beyond the reassignment period.

General McChrystal also improved the liaison programs to connect different organizations. In this process, a liaison was sent from the Task Force to an agency and the agency would send a liaison to the Task Force. Traditionally, the liaison was a person on the final tour before retirement or someone who wasn't fitting in. The liaison was often viewed as a spy, were of little value and rarely trusted. The Task Force viewed the liaison program as an important mechanism to build trust and cooperation. They started using high performing commandos to serve as an ambassador in civilian clothes in embassies and agency offices far from the battlefield. The partner organizations responded in kind and began sending high level performers from their ranks and the bonds of trust and cooperation increased.

Empowered Execution

As the shared consciousness of the Task Force grew, General McChrystal states that "we had become vastly more thoughtful, integrated, and insightful, but the Task Force still was not fast enough." A major limiting factor was the requirement to relay information up the chain of command and send decisions down. The risks of not acting in a timely fashion were higher than the risks of letting competent personnel make critical decisions.

General McChrystal began empowering his subordinates to make decisions with the guidance of "If something supports our effort, as long as it is not immoral or illegal". After some growing pains, the decentralization of managerial authority, i.e. empowered execution, not only provided decisions faster but surprisingly, with higher quality. Credit for the increases was given to 1) the individuals who make the decisions are more invested in the outcome and 2) the shared consciousness developed beforehand proved again the relationship between contextual understanding and authority.

The Gardener Role

As the Task Force evolved and gained more success against AQI, General McChrystal realized the leadership paradigm of the team network had to evolve too. Historically, military teams where honed to be chess pieces while the leaders were trained to be chess masters who could view the board and move the different pieces into place quickly and confidently. As the team of teams adapted the empowered execution model, the chess master role gave way to one of gardener, enabling rather than directing. Enabling with the goal that subordinates function with "smart autonomy".

General McChrystal makes the case for an "Eyes-On, Hands Off" approach. Far from passive, ToT leadership is focused on actively maintaining the environment, the teamwork conditions, and the cadence of progress. The chess masters of the old must become the stewards of the new ecosystem.

Challenges of Larger Scale Software Development

Software development is difficult. Anyone who has helped push a significant solution into production can appreciate this simple statement. While nowhere near the levels of importance, risk, or cost as the military engagements described in the book, the software

development process can impact one's professional career, a company's financial health, and, in some industries like medical devices, the safety of the people within the software's realm of influence.

Over the past three decades, our craft of software development has drastically matured. Our understanding of the problem domain has increased, technologies have evolved, information sharing is now common place, and end users have become much more knowledgeable of system capabilities and boundaries. A single, small development team of the right people can build amazing applications in very short periods of time and have a very fun and rewarding journey doing so.

While our industry has entered a local optimization phase for evolving the software development process for individual teams, there are still several areas of improvement for larger scale development processes when two or more teams are required. Mid- to large-software companies and sizeable IT departments are struggling to scale the software development process of single teams to develop major applications and enterprise wide solutions. The efforts often result in project failures or gross inefficiencies.

The obstacles to succeeding in larger scale software development are numerous and complex. I don't pretend to know all of the factors but after 30+ years of experience in fifteen different industries, I do have my top three list.

1. *More is more*

US-based companies value their managers by two major metrics – the number of employees who ultimately report to the manager and the size of manager's budget. Larger numbers for either measurement equates to "more value" to the company and higher rank and compensation. Managers are not recognized for the ROI of a project, the satisfaction of their "internal clients" and stake holders, or the effectiveness and efficiencies of their teams.

I have witnessed managers, directors, and vice-presidents eagerly participate in an managerial "land grab" of a new application or expanded functionality of an existing system. They sought to broaden their headcount base and increase their budget even though the resulting division of responsibility didn't make sense from a system engineering perspective. An enterprise architecture based on organizational structure versus solid design principles can lead to gaps and overlaps in the overall solution. The difficulty and cost of development and maintenance across the organization can drastically increase.

2. *Filling the void*

Contrary to popular belief, software developers are human and act as humans do…at least most of the time. When faced with situations of high anxiety, humans will typically seek a comfort zone. Similarly, when software developers confront a challenge, they will typically seek a familiar approach, tool, or technology. When faced with a void of

information, developers will 1) shut down, 2) work on something else, or 3) fill the void with familiar material. While option 1 appears to be the least desirable action, option 3 has the potential of having a negative impact on the project if the void is filled incorrectly. For example, a developer could decide to implement part of solution in a familiar but inappropriate technology. The follow on issue is that once developers have invested in an approach, most will resist giving it up and switching to a different path.

3. The rogue ones

In almost all sizeable software development organizations, there are the rogue individual contributors, managers, teams, or vendors. These entities self-inflate their expertise, superiority, and importance to the enterprise and act accordingly, typically to the determent of the larger organization. They exceed their natural or stated boundaries and responsibilities by exploiting weak or indifferent upper management. The rogue ones progress in their own direction and at their own pace.

Does Team of Teams Have The Answers?

While this book is the first time I have read the term "shared consciousness", most software development organizations that I know are attempting to become more agile, more adaptable, and more responsive. I believe that most would agree that they are trying to achieve a shared consciousness either through grass-roots bottom up approach or a top down explicit program. The successes of these organizations span the spectrum from not much improvement to self-proclaimed success to actual transformation.

General McChrystal was spot on with the belief that transparency and trust were the driving factors in the quest for shared consciousness. Transparency is difficult to initiate, to be the first to "open the kimono". However, when truly supported by upper management, transparency can be contagious and even refreshing to the participants when they can share the challenges, successes, and even setbacks of their teams.

Regularly scheduled status meetings are becoming more common in multi-team software development projects. I see the trend of these meetings going from once-a-week to daily as presented in the book, a practice that I recommend. While the timeframes for most commercial software development is not at the same level as the Task Force, a daily scheduled meeting will keep the teams focused on the common goals more effectively.

One technique that I use when running a status meeting for either a single team or a team of teams is that everyone (or someone from every team) in attendance must participate. Every person (or team) is a participant, there are no silent spectators.

In addition to a daily status meeting, a "war room", "bullpen" or joint operations center concept is an extremely important practice. Not only can information be shared throughout the day (and night) with members in the single room, personnel beyond the scope of the project or task force will have a single place to go for information and assistance. While some of my clients will create a bullpen a few days before and after

the installation of a major software release, I encourage the use of a central communications center throughout the development of large systems and not only during the deployment phase.

In the book, General McChrystal discusses the challenges of building trust that is critical in the "team of teams" network. The Task Force leadership used embedding and liaison programs to integrate personnel into nodes of this network and create relationships with the target team's members. "Face time" is also important to software development projects (and probably most human oriented activities). While most software projects don't have the budget to send key members to a remote team for a six month period, I do recommend to clients that each person (not just the manager or team lead) on a development team spend a few days with each remote team.

While it is impractical that everyone knows everyone else in a large organization or network, most large scale projects have someone, typically the team's technical leader or manager, know someone else on most of the other teams. One of the goals of General McChrystal was that *each person* knows someone on every team. I have not thought of a project's "network connections" with this type formality before. I believe that it is an important goal and should be a consideration for informal measurement with the possibility of becoming a formal project metric.

As the Task Force gained shared consciousness, they were still not as responsive as needed to defeat the AQI. In an effort to speed the process of decision making, General McChrystal discusses how they migrated toward an empowered execution model. On this point is where I diverge the most from the "team of teams" concepts. The Task Force was comprised of personnel extensively trained in the command and control structure. Roles are understood, responsibilities are accepted, rules are followed, and orders carried out. The Task Force leadership had the challenge of moving from a rigorous hierarchy to a collaboration, network model.

In the US, software development groups are staffed with personnel with very different backgrounds and little or no training in management structures. While the organizational chart will probably be a hierarchical representation and may graphically be similar to a military command structure, software groups lean toward a self-interest and free-wheeling model than the military's role fulfillment and order execution paradigm. I believe it is more difficult to move the self-interest groups toward collaboration than a command and control oriented teams. There should be *more* command and control principles in software development.

Over twelve years ago, a colleague and I examined the different projects we worked on over our careers. The one attribute that was common in the successful projects and missing in the failures (by being abandoned before completion, being completed past the deadline, not meeting the requirements, or exceeding the budget) was a strong decision maker. This person presides over all of the different groups and had the responsibility of

the overall solution and *the authority* to make it happen. Indecisions are short lived. Conflict resolution is swift and final. This trait stood out so much that we termed the concept as "a dictatorship behind the façade of a democracy". Members of the democracy could make the decisions until the point the dictatorship had to intervene with absolute authority. The dictator (typically the chief architect) follows General McChrystal's "Eyes On, Hands Off" policy but can inject her decision at any point and time with complete authority.

At initial glance, this model appears to be a harsh, intimidating, and unpleasant process but, in my experience, it is highly effective and productive with the appropriate people. I think that given the command and control structure of the military and the extensive training of the personnel, the empowered execution model is close in practice to our dictatorship/democracy one. Decisions can be made at one level but with the full knowledge and acceptance that they could be overridden and changed by a superior.

The last topic that I will cover is one that General McChrystal very briefly discussed. Early in the book General McChrystal stated that little of the Task Force's transformation was planned. He wrote "Instead, we evolved in rapid iterations, *changing – assessing – changing* again." The mindset of continuous process improvement was, perhaps, the most important concept of the endeavor. Similarly, my own meta-process is:

1. Measure the process
2. Analyze the data
3. Design the improvements
4. Implement the changes
5. Go to Step 1

Conclusion

First of all, this book is the rare combination of being both informative and entertaining. I highly recommend it to anyone searching for an interesting and engaging read.

The cover on this book has a subtitle of "NEW RULES OF ENGAGEMENT FOR A COMPLEX WORLD". I was expecting new insights and tips for managing large groups with the "team of teams" network for large scale software development. I was actually surprised that many of the rules are practices that I recommend to my clients.

The only major issue I have with these "new rules" for software development is the one involving empowered execution. My argument is that most software development environments need *more* command rather than less. A well respected and highly experienced colleague recently said that the secret to software management at any scale is to "know when to collaborate and when to command". I could not agree more.