Group Work in Tech:
Step Up, Step Back, Dive In

By Karen A. Razzano razTechPro@gmail.com

Introduction

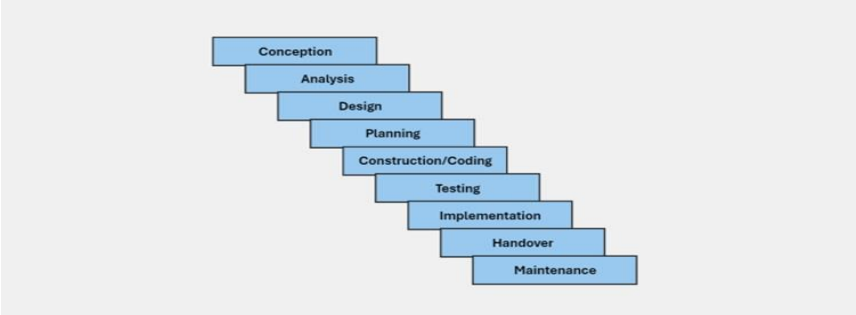
Group work in Technology can be a fun professional experience in software development.

How in the world can work be fun? And doing tech with others?! This paper will convey some of the reasons why, as well as introduce some of the practices that can help software development professionals be successful and open the door to a rich and encouraging professional network.

Approach

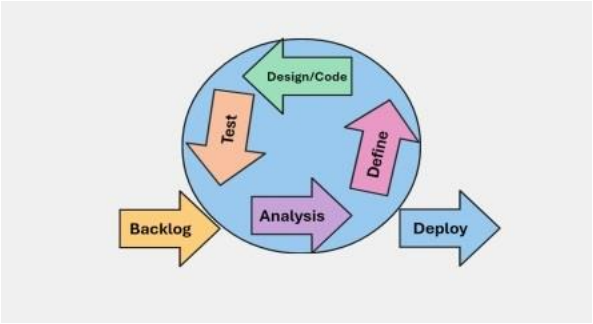
After years of working in Technology as a software developer, I joined a company in the early 2000's that started using Agile. Agile was a new approach to working in teams where the people on the team and their ability to work together are of primary importance.

The new approach was important to all engineers in the software industry. A national group of 17 software industry engineers met to improve software development and authored the Agile Manifesto in 2001. Until that time, most software engineering teams used the Waterfall methodology, a linear, step-by-step process in project management that became more and more cumbersome as software and system needs became more and more nimble. There were rigid phases to Waterfall software delivery, each dependent on the completion of the previous phase. Each phase had its own deliverable timeline. A common visual depiction:



By the time a system was delivered to the user, it was outdated and inflexible. It was common to start a project using today's technologies, complete each phase step by step, and then, a year or more later, produce a system that met last year's needs. See more below, and at the agilemasifesto.org website.

A visual depiction of Agile:



Core deliverables in Agile focus on user needs and are incrementally and iteratively completed by autonomous self-organizing teams, with a mindset to embrace change. Once the team understands the problem they need to solve, it is up to them to figure out how to do it together. People and teamwork mattere more than processes, or as the manifesto reminds us: individuals and interactions over process and tools.

I was asked by my friend and colleague, architect Dave Lyon, if my team of developers would participate in a pilot using the Agile approach to group work. This team, made up mostly of developers new to the industry (under three years of experience), would be given a docket of work that would take about two weeks to complete. They would get a new docket of work when they completed the current docket.

The challenge was they would look at the work and decide what the priorities were, what work items went together, and who would work on what. And then they would need to do it again two weeks later with new tasks, and they needed to measurably improve as a team.

They did not like the approach at all. Why should they have to do the job a manager does? How would they know what was more important or less important? What if they disagreed on who took what? They did not like confrontation!

Manifesto

It helps that Agile has a manifesto with tenets that serve as a foundation for all group work in technology. It is easy to find in the Agile Alliance website at agilealliance.org. The tenets highlight a sustained cadence in delivering excellence in software, *and* the importance of each team member being a good human: communicating face to face, ensuring that every team member's voice matters, rejecting idleness, embracing the idea that each team member owns the work that the whole team is delivering. Team.

Team

I have read many acronyms for TEAM over the years, and the one I like best is Together Excellence, Apart Mediocrity. I have found it to be painfully true; it can be challenging to pull off. Even the best teams with great technical know-how can fail if they do not work well together. All it takes is one person judging themselves as less or judging others as less: less competent, less committed, less capable, less open, less collaborative, less humble, less encouraging, less accountable, less reliable – the adjectives go on and on.

It is constructive and, in fact, common for high performing team members to disagree and engage in professional discourse. Engaging in discourse is a skill that requires professionalism and respectful practice. It requires professional maturity. Trust. Generosity. If done right, what emerges are the best ideas possible.

Teams figuring out system solutions are likely to face a lot of ideas from team members to shake out. In my view, it is imperative for team members to ***step up*** – voice their beliefs, ideas, and concerns – and then ***step back*** – listen whole-heartedly to another team member's beliefs, ideas, and concerns. Then the team decides how to proceed – and ***dive in.*** A team member whose ideas are bypassed has a responsibility to fully support the team's decision, even if they are disappointed, or worse, angry. After all, every team member has talents and skills they brought to the team. Supporting a teammate's idea is important.

So how does a team member do their best and be their best, forge ahead technically and professionally maturely, with both energy *and* humility?

Practice

Agile has several practices that help teams stay disciplined, sustain an impressive pace, deliver quality, ensure effective communication, boost each other up, and provide professional growth for its team members. There are several Agile frameworks that help teams apply the principles, e.g., Scrum, XP, Kanban. There is information regarding these and more online.

Agile is not magic. It requires each team member to be genuinely committed to the team, to the team's promises, to continuous learning, and to each other's success.

The practices that help a team get moving quickly together are:

- Set team policies (how the team will work together)
- Start the day with a stand-up meeting
- Plan, plan, and then plan (this includes brainstorming, designing, and breaking out tasks)
- Use a Kanban board or something like it to visually depict tasks
- Call meetings to discuss issues/share knowledge (any team member can do this)
- Conduct a Retrospective when the work has been completed

But first, let's talk about being good humans.

Group Work in Technology

The discipline, Agile, can be seen as a bit odd by folks who are wired for or comforted by a hierarchy. There is no assigned singular team leader who is in charge. Each team member has an equal voice and responsibility. Each team member ***is*** a leader: one who influences, seizes opportunity, listens, speaks up, brings out the best in others, amplifies the voices of others and champions ideas that help the team become successful.

In successful group work, each team member holds these characteristics:

- *Accountability* ***– no blaming, no complaining, no judging***
- *Commitment* - ***every person is all in and dedicated to the team's goals***
- *Collaboration* – ***meet others where they are, complement each other, listen***
- *Seek out/offer Assistance* - ***raise your hand, practice awareness***

In a technical space, working as a team can feel very different than many people have experienced outside of technology. In many teams outside technology, there is a team leader who makes key decisions and assigns tasks. In tech, the team may receive work from those in leadership but the team decides as a unit what to do, with team members breaking out and voluntarily taking tasks. No team member cedes decision-making to someone else. Being a good Agile team member clearly requires emotional intelligence and a willingness to respect others.

What if team members are brand new? Or the team has an accelerated timeframe for delivering a project? Or team members are working overtime, on weekends? What if there is no time to build trust? What if team members become overwhelmed or lose sleep, or encounter personality clashes? What if a team member leaves without warning? What if in-person team members keep forgetting about the remote team members and have meetings excluding them? What if team members do not like each other or just do not get along? Can't a manager just swoop in and get everyone in line? Can't someone else do the hard people-stuff?

BRAVING Inventory

Brene Brown, the storyteller and sociologist who studies relationships and talks a lot about choosing discomfort over resentment, has an approach that works well in establishing and maintaining professional relationships. This approach aligns with Agile and can be a good guide for teams that are forming together and strive to be high-performing teams that deliver excellence.

It is called the BRAVING inventory:

- **B**oundaries – understand everyone's role/responsibility – make clear what is ok and not ok
- **R**eliability – do what you say you are going to do

- **A**ccountability – no blaming no complaining – own your own mistakes, apologize, make amends
- **V**ault – honor privacy – be a trusted confidante – do not share what isn't yours to share
- **I**ntegrity – professionalism, choose courage over comfort, choose what's right
- **N**onjudgment – do not judge others – you can ask for what you need and so can others, without judgment
- **G**enerosity – extend the most generous interpretation to the intentions, words, and actions of others

Many technologists may not have entered the industry thinking they would have to practice humility and generosity every day, but if they want to be part of a successful team, it is truly what makes the Agile approach work. And if they want to have a rich professional network, it starts here.

Team Policies

Once a team forms, it is imperative for the team members to meet each other and decide on the policies the group will follow to work together effectively. This is one way I would apply the **B** in **B**RAVING. These policies are like community agreements. When will the team meet every day? How? Online? In person? How often? What types of meetings with they have? Will someone keep notes? Who is going to do what? Are there tools they need to use/learn? Will they do design? Who will do that? When will they plan? Are there technologies they need to review? How will they keep track of work? How will they handle disagreements? What are the rules of discourse within the team? Who will speak for the team if leadership requires updates? How will the team handle issues?

A simple beginning list of team policies can look something like this:

1. Daily standups at 9:15am on Zoom. Will run no longer than 5 minutes.
2. Check-ins during the day at 11am, 2pm and 5pm on Zoom.
3. Day 1 of project will begin with a planning discussion after the standup finalizing policies. The team will then identify the work and break out tasks on a Kanban board.
4. Tasks will be updated (added, deleted, edited, completed) as appropriate.
5. Blockers will be solved by the team member reporting the blocker. Team members will assist on request and elevate the issue to leadership if not solved within three hours.
6. Team members will exchange emails/phone numbers to notify each other if necessary.
7. Team members will take tasks as they become available. No team member will be idle.
8. Code repos: Github. Every team member will commit frequently during the day.
9. Only code that compiles cleanly will be committed.
10. [Team member's name] will be the code gatekeeper for this project.
11. Any collaborative needs for meetings will be broadcast to the team in Slack and the team will collaborate in Discord in [channel]. Actions taken will be recorded in Kanban under the appropriate task(s).
12. Team members will communicate professionally and without judgment. Each member will commit to calling a meeting in Discord to discuss any issues within the hour of discovery, if before 7:30pm.

13. When team members disagree, they will each express their views without interruption. The team will commit to coming to consensus and all team members will support the decision of the majority.

Well, sort of simple; these are clearly very detailed statements so that every team member knows what the rules are. Each team member is an author.

Using Team Policies

Once policies are in place, the team needs to review them regularly to ensure each team member is abiding by them. Teams fail when they get lax on their agreements. Or when a team member discovers the policies are not quite right, yet stays silent. Agile provides for flexibility, so if the policies need improvement, the team meets to improve them.

If a team member is not abiding by the policies, any team member can call for a team discussion on the matter. This can be uncomfortable for some. Choose discomfort over resentment and discuss without judgment. Ask for clarity. My former colleague and leader Michelle Gross would start a sentence in this way: "Help us to understand…". It was helpful that she just wanted to understand so she could be part of the solution. And what a relief that she avoided blame.

Stand-ups

Every day, a team will meet to set the stage for the day. They do so by standing up, answering three questions, and launching their day together. The classic stand-up meeting answers these three questions for each team member:

- What did I do since we met last?
- What will I do until we meet again?
- Do I have any blockers?

A blocker is something that keeps the team, or any team member, from making progress. If there are blockers, the team commits to addressing those blockers right away, and often discusses the blocker right after the stand-up.

Note the stand-up is essentially a report-in. It is not a meeting to discuss solutions. It is a short meeting (ten minutes) to inform and communicate achievements up to this moment.

An additional helpful check that can be done at this time is a battery check. Ask your teammates, "how's your battery today?" If I have a teammate with a low battery, it is my job to be sure they know they can count on me to be a good buddy. I am saying "I got you." This can be one of the fastest and most sincere ways to establish trust between team members.

Why do people stand, if they are able, during a stand-up? There are a few reasons: to keep the meeting short, to ensure no one drifts into reading emails or doodling or coding while

others are talking, to focus only on what is important. You start sitting, you start fidgeting. So, if you are able, stand. If you can't, don't fidget!

Forgiveness

One of the hardest things to do as humans is to forgive others when they do not meet the expectations we set for them. Maybe they do not keep their promises. Maybe we expected the wrong thing from them. This is an area where teams easily fail. The importance of being nonjudgmental is the **N** in BRAVI**N**G, and the ability to forgive is the **G** in BRAVIN**G**.

It helps to set expectations together so the team agrees early on that those set are reasonable. You build trust when you allow others and yourself to learn from mistakes made.

Brainstorming and Discourse

The BRAVING approach is very helpful when it comes to team discourse. When a team starts discussing a problem they need to solve, brainstorming ideas together is important. Each team member has a unique perspective when gelling with others on an intellectual challenge, i.e., brainstorming. Each perspective expressed enriches a healthy pool of ideas.

Teams that have no discourse are stagnant (at best) and dysfunctional (at worst). Why? A team with members who are motivated to find the best solution to a problem need to talk through their ideas and not be afraid or impatient with wherever that discussion leads. Often there is more than one way to proceed. Team members who are curious may stimulate interesting solutions and will certainly encourage learning. Katherine Johnson, the great NASA mathematician, scientist, and human computer said it perfectly: without curiosity there is no learning.

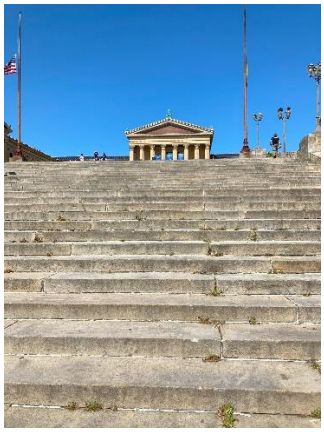
A team that does not learn will never be high-performing.

An open mind and the willingness to brainstorm can work wonders. Try using the Planning Poker technique to nudge good brainstorming, and hence, healthy discourse. There are many websites with a planning poker feature that can be used to help a team estimate effort using the Fibonacci sequence. (A commonly used modified sequence is 0, 1, 2, 3, 5, 8, 13, 20, 40.)

What makes it interesting and useful is this: the team discusses a project, or task, or problem fully and then agrees to a baseline to compare the project/task/problem to. Then they vote on the effort. Usually the first vote results in wildly disparate estimates. And this encourages team members to talk about why they voted as they did. A cool first step into professional discourse. Then they vote again. Often, their estimates as a team will be closer.

It may be helpful to remind ourselves that team members can bring a lot of baggage with them to such a discussion. They bring different experiences, some positive, some negative. Biases. Expectations. Self-esteem. Arrogance. Caring. Maybe a low battery today. A policy to leave baggage at the door can also work wonders for encouraging professional discourse.

Planning Poker Exercise



I asked a group of nine Zipcode Wilmington student engineers to estimate the effort to climb the Philadelphia Museum of Art's 98 steps. I wanted them to use the Planning Poker technique to estimate the effort. They asked me questions, and I purposely did not indicate any timing requirements.

The team decided on a baseline range of 2 (the effort is easy) and 20 (the effort is hard). Then they revealed their estimates: 2, 2, 5, 5, 5, 8, 8, 20, 40.

Well, that's disparate. We started to talk about why they voted as they did. "I figure I have all morning so I'll get to the top eventually," said one team member who voted 2. "I can run far but I can't do steps," said another team member who voted 40. I asked, "Did you know you can drive a car behind the Art Museum and get pretty close to the top steps?"

We talked about the fact that there was no time requirement. Someone asked if there might be help available to get up the stairs. Yes. The team decided to vote again. The votes became all 5's and 8's. Their discourse resulted in a reasonable estimate, as well as some clarity.

In a technical space, the work requiring estimates will relate to applications the team is responsible for. Maybe the team needs to enhance an app that exists. Maybe the team needs to add functionality. Given the problem, the team works to understand what the problem is, discuss each person's understanding and assessment, take in guidance from stakeholders like product owners, architects, designers, consultants, savvy users, and then they can have a round of planning poker. They may break problems up into smaller problems and estimate each of them separately.

These discussed items go into the project backlog for the future. What's a product backlog?

Product Backlog and the Kanban Board

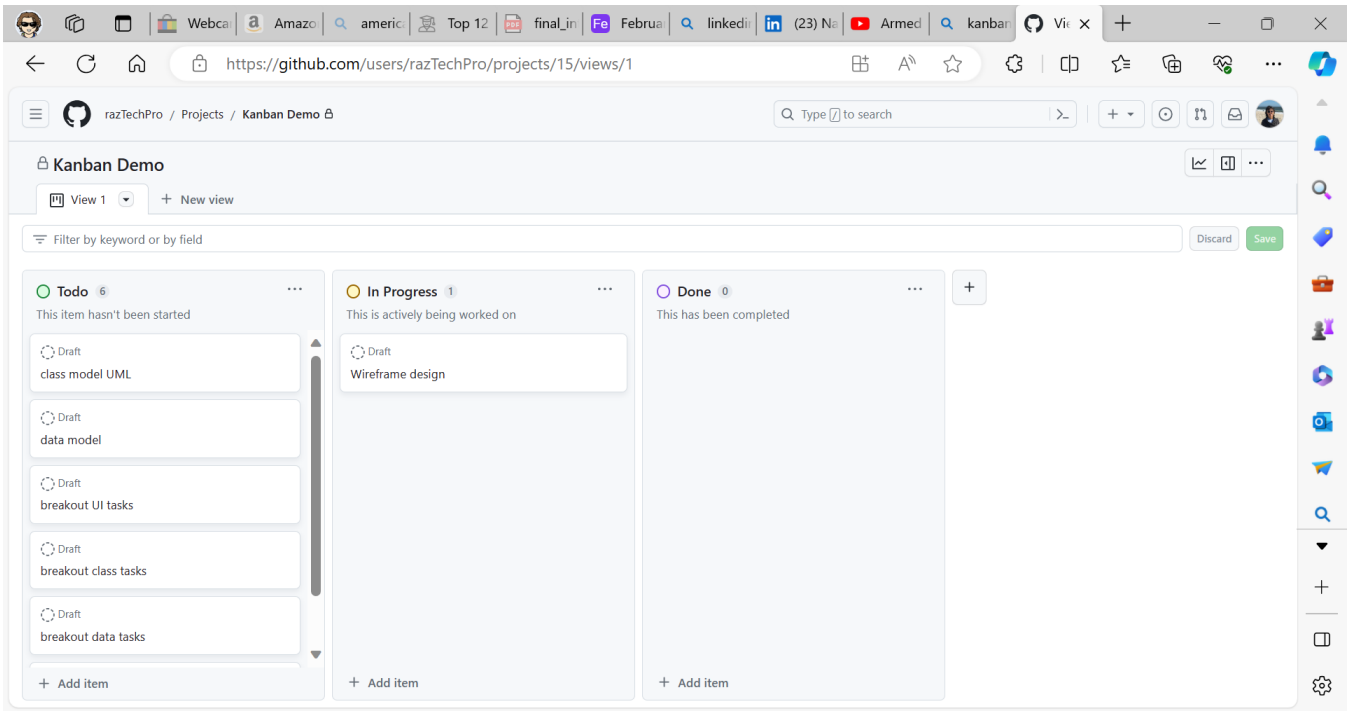
When a team plans their work together, they have an idea of the amount of work they have to do. All that work is reflected in the product backlog. Although it is called a backlog, it is not a list of projects the team is late delivering. It is just the work the team will need to do, maybe for the quarter, maybe for the year, maybe for the semester. The team commits to do all that work within that timeframe. That larger timeframe is defined by a leadership team and stakeholders.

Thus, the product backlog will encompass the work the team knows it needs to do. The team usually knows generally when the work is due, but it is up to them to decide in which manageable timebox they will deliver it. The timebox is called a sprint or an iteration. The

team says we will do a certain amount of work, based on our estimates, and we will do our best to maintain a consistent cadence from sprint to sprint.
Thus, the team delivers the product backlog incrementally, over many sprints within that agreed upon timeframe.

Once the team agrees with stakeholders to move a product backlog item into the sprint backlog, the team digs into each product backlog item (PBI). The Agile framework Kanban has a great tool for visualizing the work *and* the flow of work throughout the sprint: a board.

Kanban's origin is from engineer Taiichi Ohno from Toyota. The Japanese word translates to "visual card", which tracked production in a factory. Applying the concept to an Agile board looks something like this:



Every task the team comes up with are posted on the Kanban board, in this case, the "To Do" column. Then each person takes a task, moving it to the "In Progress" column. If the task is too large, it is broken down into smaller tasks. Only a task being worked on is in the "In Progress" column. Another task is not taken by a particular team member until their in-progress task is completed and moved to the "Done" column.

When the whole team practices this approach, the visual cue is clear regarding the team's progress with the work. The board can be used for paired work, full team work (like designing a UML or roadmap diagram), and individual work. It can be helpful to include not just the project work of a team, but other tasks team members partake in, including classes, professional development, etc. (Note there is *much* more to Kanban, a full Agile framework.)

Blockers or Impediments

When team members start their tasks, it would be wonderful if the unexpected stayed away. But things happen which can block or impede a team from completing its work as planned. A team member calls in sick. The team is missing a skillset that is needed to begin the task. A resource is unavailable. There is a power outage. A senior leader decides something else is more important for the team to work on, so expects them to drop everything. More likely, the team member runs into a problem they need to dig into more than anticipated, and the outcome risks completion of the task. Or the team learns something that was not known at the onset of the work. Maybe a new discussion regarding the effort of the task is prudent, given new information.

There are some blockers or impediments that may cause a team to reconsider a task's inclusion in the sprint backlog. When that happens, the team can swap in another task from the product backlog, if circumstances permit.

There are also times when leadership may want to add items from the product backlog to the sprint backlog after the team has finalized the sprint backlog. After discussion, it may be appropriate, but the team gets to swap out a task of like effort so they can keep their promises regarding deliverables. A team could also define a current task as a stretch goal, something they will try to deliver but may need to push to the next sprint.

Day-To-Day

Although most Agile resources will give you a lot of information about the discipline and practices of Agile, examples of how you function day-to-day are not quite so readily available. How does it work once a team member takes a task? Do others take tasks at the same time? What if one finishes before a teammate? Do they just take items from the board whenever?

It is called Agile for a reason.

Often, during sprint planning, team members will start to express what tasks they want to start with. Assignments happen quickly. Per Kanban practice guidance, a person works on one thing at a time. They start it, work on it, and finish it before they take another task.

Sometimes teams initially create tasks without thinking about breaking them down in to smaller parts, but it is an important practice before the sprint starts. A real benefit is when a task stalls; maybe there is a need for approval or feedback or clarity. If the task is broken out to smaller subtasks, a team member can finish the subtask and take on something else without being idle. The next subtask, in this case the one needing approval or clarity, stays in the "To Do" column until it is ready to be worked on. This way no task stalls in the "In Progress" column.

Successful team members prioritize complementing each other. Complement, meaning make whole. Getting a feel for each other's work ethic, approach, communication styles – these things matter. There may be a team member who is meticulous with details and thus slower to

complete a task, and another who is quick and actionable. There may be one who plans deeply, and another who prefers trial and error. It makes sense for team members to review each other's work especially if they have very different approaches. Team members must genuinely check their egos at the door, whether they are the author or the reviewer: the point isn't to show off knowledge; the point is to deliver quality. To both teach and learn. Humility is required.

Team members can excel together when they talk face-to-face and are honest with each other. There really is no place for judging others. Completing a project together is not a competition between team members; there is no value in keeping count of how many tasks each member does, or how even long a task may take compared to someone else's time in completing a task. I really want my teammates to help me be my best self. I want to return that favor. It is an authentic way to start building your professional network.

It is helpful to remind ourselves that team members don't have to be friends. However, they *must* be professional colleagues who believe in TEAM. Together Excellence, Apart Mediocrity.

A big challenge for most teams is deciding when to take a task that requires a skill that is a team member's strength, or when a less skilled team member can take such a task so that the team member can learn. There are a lot of strategies to do this effectively, for example, pairing up team members of different skill levels to work on a task together (both team members build different skill strengths going forward), giving the task to the skilled team member (the task gets done quickly, though no one else learns that skill), breaking down the task further so multiple team members can work on subtasks while the team executes incremental skill strengthening (while still getting the overall task done fairly quickly).

The team can decide what the best course of action is task by task. In the end, the team wants to not just get the work done, but also gain skills while working toward project completion.

Agile. Agility. Flexibility. Adapting to change.

Retrospective

At the end of every project, every sprint, every set of work that is done, the team reflects on the work: what they did well, what they didn't do well, what they commit to improve going forward. And then they apply those improvements forever.

This is called the retrospective, which is conducted as a final team meeting before moving on to the next sprint or project. It is a quality time event, not a quick stand-up. It is the roadmap to improvement and excellence.

Staying Agile with Agile

Agile is an approach and a practice and there are different frameworks teams can use to deliver software and work together. Team members get better at it by doing it. This means applying lessons learned going forward. And it means working to build and maintain trust between team members. In shorthand, here's Agile:

- Set/review policies and abide by them respectfully (use B.R.A.V.I.N.G.)
- Daily stand-up report in
- Remove blockers quickly/call meetings for clarity and action
- Plan seriously and participate fully
- Break out tasks
- Collaborate humbly and enthusiastically
- Complete each task
- Deliver the sprint backlog
- Do a retrospective without blame or judgment
- Apply what has been learned going forward
- Repeat

There is much more to know about Agile and its frameworks that I invite you to explore and practice. The disciplines are serious and require serious attention. Learn about the disciplines of planning, refining work, collaboration, resolving disputes and conflicts, assessment, issue tracking, continuous integration, scaling Agile, the list goes on and can be applied in different ways in different organizations.

As you work and learn with others in this style of robust working environment, those with whom you have team relationships become a lasting and evolving source of your growing professional network.