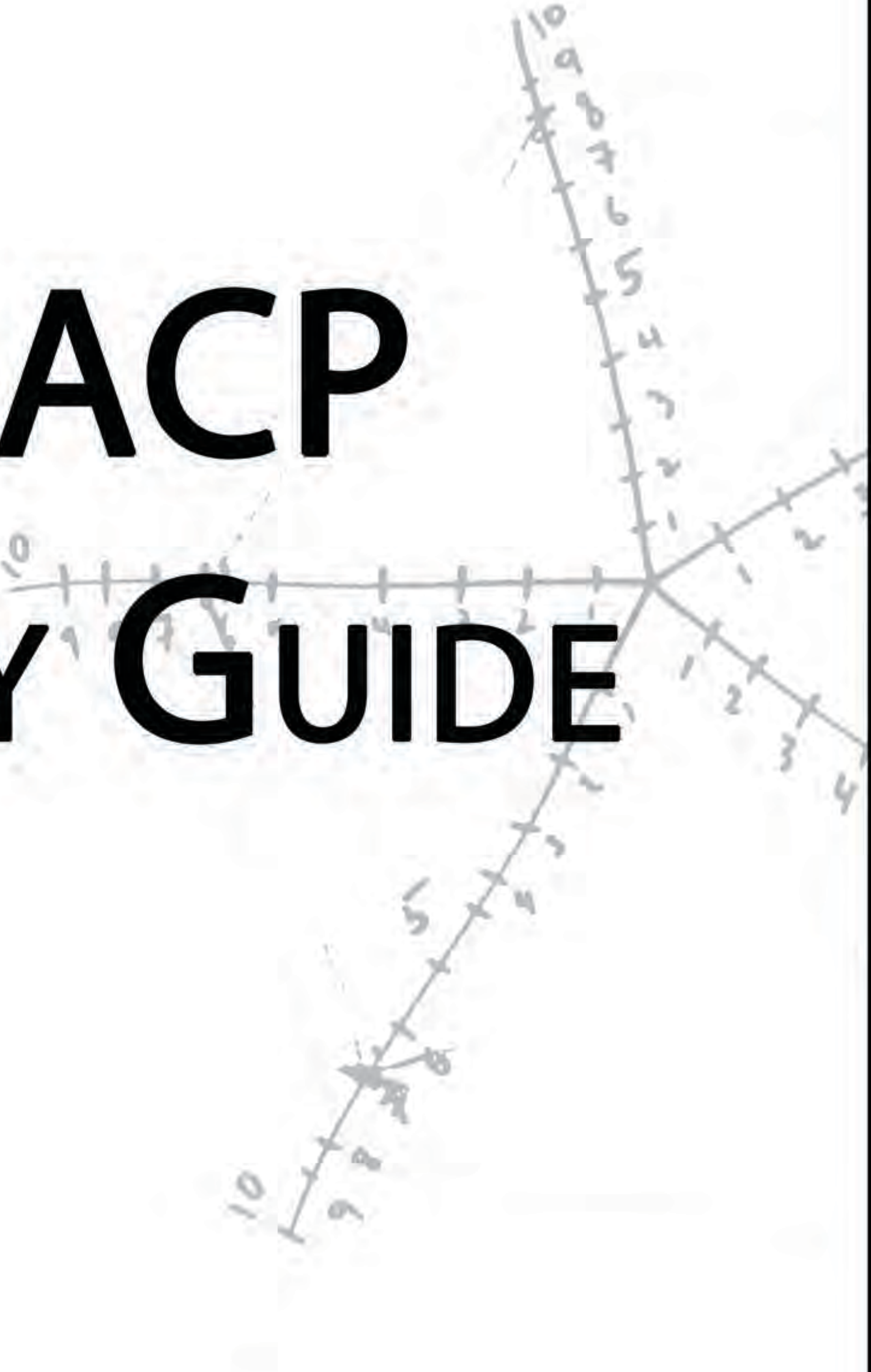
santeon group
innovation forward

PMI-ACP Study Guide



APPENDIX A: AGILE TOOLS AND TECHNIQUES (PMI-ACP EXAM TOPICS PART I)

COMMUNICATIONS TOPICS

INFORMATION RADIATOR

WHAT IS IT?

Alistair Cockburn first coined the term and helped define what an information radiator is,

> *"…a display posted in a place where people can see it as they work or walk by. It shows readers information they care about without having to ask anyone a question. This means more communication with fewer interruptions.*
>
> *"…a large display of critical information that is continuously updated and located where the team can see it constantly"*

WHAT SHOULD A TEAM RADIATE?

The team should radiate anything it thinks is beneficial and reduces information access time for them and management. This could include the following:

- Working agreements: vision, goals, chartering output, people on the team, schedules
- State of the work
- Schedules
- Commitments: e.g. turnovers, dates, impending deadlines,
- Impediments/roadblocks/risks
- Action items from retrospective (things to improve)

WHAT VALUE DO THEY HAVE?

There are numerous benefits to the use of Information Radiators. Some of the significant benefits for the team and overall organization include:

- Increase *transparency*: anybody at any level can see status and have access to information.
- Increase *trust*: the team will correct what needs to be corrected.
- *Empower* teams: teams will take ownership of work and find solutions.
- Shows project information without having to ask anyone a question. This means *more communication* with *fewer interruptions.*

WHO ARE THEY FOR?

Information Radiators are first and foremost for the project team so that they have a continuous set of information that is useful to them. A secondary audience is external stakeholders so that they can get information immediately at a glance about the project without needing status reports or status meetings; this reduces the burden on the team. Teams may also find it beneficial to have individual radiators that express such things as individual progress, moods, or simply whether a person can be disturbed or is currently in deep thought.

WHEN DO YOU CREATE THEM?

Information Radiators are created, updated, and sometimes “deleted” throughout the life of the project depending on their usefulness to the needs of the project team.

IMPORTANT THINGS TO KNOW ABOUT INFORMATION RADIATORS

Information Radiators gently “push” information so that members of the intended audience get what they need from simply looking at the radiator. This could be a bullet list, a priority list, or a chart. When teams are distributed, thought needs to be taken as to how this “push” will take place. It could be an email notification, a web page that automatically refreshes, or news feed. ***One of the most impressive information radiators teams have adopted is to use the second monitor most people have nowadays as an information radiator instead of more desktop space.***

The Information Radiators, like the work itself, will likely change or evolve over the course of the project as more is learned about the needs of management and the team. This brings us to the difference between Information Radiators and project documentation. Information Radiators are designed to convey information for a (relatively short) period of time in terms of relevance to the project context where as documentation is intended to have some permanence about it even if it evolves. There is a temporal element to the Radiator that is not necessarily true of the documentation.

TIPS AND TRICKS

Information Radiators have a set of “good” characteristics that make them effective:

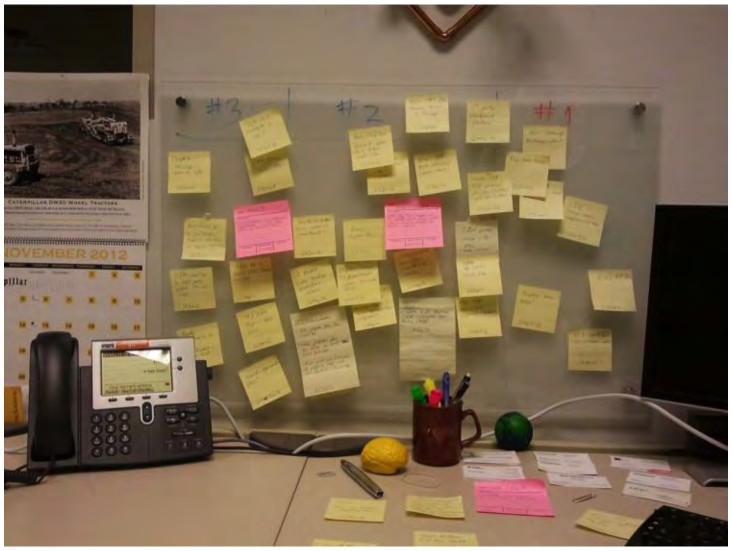
- Must radiate information to the right people
- Large display
- Understood at a glance
- Always relevant
- Easy to update
- Aesthetically pleasing

As the space for an Agile team has Information Radiators posted, it begins to take on the characteristic of becoming an Informative Workspace.

As it does so, conversations will either be designed around some of the Radiators or will simply occur.

Other Radiators will simply replace status reports or status meetings. Some radiators will supply a running health indicator of the project or product. Other radiators reflect the overall mood of the team.

Here's an example of a personal task board as an information radiator:

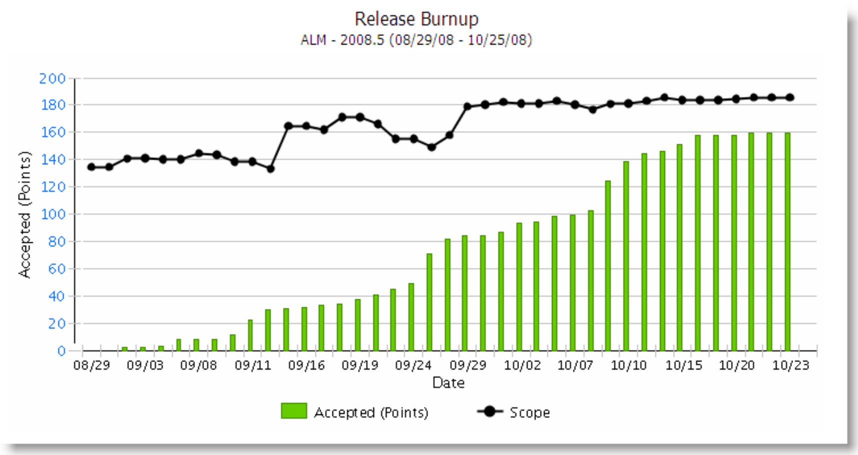


Not only is it (most) beneficial to the person using it, but visitors to this person's office space can get a sense of what work is being done and its importance, particularly with the use of different colors.

Information radiators can be electronic in nature as well.

When using electronic versions of information radiators, it is important to ensure they are a part of the team's regular routine so that the team doesn't overlook valuable information (consider making a check of the information radiators part of the team's working agreements in the Team Charter).

Charts, such as burn-up charts for measuring team velocity on its work, make excellent radiators for project health. They are easily understood and relay a lot of information very quickly.



DESIGNING INFORMATION RADIATORS

When discussing information radiators with the team it may be simple to using the following discussion template

- What needs to be radiated?
- To Whom?
- How?

Below are some examples to get the discussion started:

1. All issues and roadblocks are radiated on a single sheet in a room and all project team members and stakeholders meet in that room to review the information radiators at least once a week.

2. Project Management plan radiating the critical path is printed and handed to each person to put in their work spaces.

3. Project Milestones for each project printed and handed to each person to put in their work spaces.

4. Defect rate is radiated as a line graph on a monitor and updated hourly automatically from data in the backlog. The monitor is located in a central location close to the team members' work spaces.

5. On time/ on schedule status is radiated using a visual display in a central location.

6. At-Risk problems are radiated on each individual's desktop background real-time.

7. Status of each team member and what they are working on is radiated on a wheeled white board at the entrance to the team work area.

8. Work and status information is radiated via a team wiki.

9. Version numbers of shared software components are radiated on a white board near the team area.

10. A team's vacation schedule is radiated on a white board.

11. A web camera is used to radiate the radiator of one team to another remote team.

12. A project backlog is radiated to multiple remote teams using a smart board so each team can see what the other is changing.

13. Updates on tasks completed are radiated using a SharePoint dashboard.

DAILY STAND-UP

WHAT IS IT?

The 15 minute daily stand-up is a practice that high performing teams use to coordinate the most pressing things for the day to move the project forward. It's a simple ceremony whose purpose is to keep the team in high communication; it is not about status.

WHAT VALUE DOES IT HAVE?

The daily stand-up helps team members remain focused on the tasks at hand and jointly review where others may need help in making the team's commitments. It is all about communicating in a simple manner so that everyone clearly understands what is currently needed to be completed.

Some of the expected values are as follows:

- Convey progress and create awareness to the whole team
- Identify obstacles and collaborate on removing them
- Understand interactions between team members, which helps build the team
- Track progress, the flow on velocity, and understand where bottlenecks are occurring
- Offer and receive help on issues or learn other sources where help can be sought
- Communicate work dependences
- Align the team on short-term goals and timeline
- Prioritization of tasks (resolve conflicting priorities) and determining what tasks aren't truly needed
- Learn the status of dependencies and how they impact the work

WHO IS IT FOR?

Daily stand-ups are for the project team performing the work. Often when subject matter experts or business experts are a part of the team in a temporary capacity, they are also included.

WHEN IS IT CONDUCTED?

Every work-day at the same time; everyone on the team should block this time off from other activities so that they can attend. If there are remote members in a different time zone, consideration needs to be made so that they can attend via video or tele-conference.

IMPORTANT THINGS TO KNOW ABOUT THE DAILY STAND-UP

Some of the important characteristics of a good a daily stand-up are:

- Remaining within the 15 minute time box and discussing only those things that are most valuable to the members within the team. If items need more detailed discussion, the team is committed to placing them into a parking lot for discussion after the stand-up.

- Establishing this touchpoint so that it occurs each day at the same time and in the same location (or on the same conference call line).
- Identifying any obstacles that stand in a team members way and discuss high priority open items/roadblocks.

A beneficial side-effect to the daily stand-up is you can eliminate weekly status meetings and/or reports that take longer and usually have less important detail (particularly about obstacles). Additionally, the team can hold each other accountable to ensure everyone is remaining on track for the needed deliverables. This is usually facilitated by a team member.

It is important that the team performing the stand-up does not turn this into a status meeting. It needs to be focused on ensuring team member alignment to the project, holding one another accountable, and identifying obstacles to getting work done so that action can be taken to remove them.

TIPS AND TRICKS

A typical stand-up may take the form of answering the following three questions:

- What did I just complete?
- What do I plan to work on for the next 24 hours until the next stand-up?
- What things are in my way?

Another form is for the facilitator to ensure the following gets answered –

- Are there any changes to the priorities or tasks that are needed based on customer feedback or other discoveries?
- What tasks are being worked by whom (usually done round-robin style)?
- What successes can the team claim?
- Does anyone have a problem they need help with..?

Some teams that use a Kanban board or task board that shows the checklist of criteria for items to be completed walk backwards from the items just completed to items to new items that have just been pulled for work. At each backwards step, they have people pull their stories or items (thus sharing what is moving to the team) and ask what will be worked on next in this column and by whom, and lastly are there any impediments.

It is important that the team hold each other accountable for progress and ensure people are remaining focused on the objectives and tasks at hand.

If a subject comes up, usually around a decision that needs to be made or someone needing help or an obstacle, the person facilitating should immediately ask who needs to be involved and establish a follow-up meeting (preferably immediately after the stand-up) to discuss the topic. This places the item in the parking lot and keeps the team on track to keep the meeting short, while still ensuring the item gets addressed.

Ensure actions to be taken as a result of an obstacle or dependency identified in the daily stand-up are followed up, preferably with at least some progress made within 24 to 48 hours. This keeps the team moving forward. These actions should have one person accountable even if the entire team is responsible for its implementation.

It is beneficial to have this meeting stand-up during this meeting; it keeps everyone cognizant that the meeting is to be short.

Also, for more significant task completion, it serves as a morale boost for the team to encourage one another as a small amount of celebration.

PLANNING, MONITORING, AND ADAPTING TOPICS

RETROSPECTIVE

WHAT IS IT?

A fundamental team practice is to retrospect or reflect on how the team is performing with an eye towards making improvements, while still keeping what is working well. Unlike lessons learned or post mortems, which occur at the end of a project, retrospectives are held in cadence with the development cadence. The retrospective usually produces a set of *actions or experiments* for improving a set of items the team feels need to be addressed. It is this bias towards action to make improvements during the project that makes retrospectives different from lessons learned or post-mortems which usually are applied to the next project.

These actions could address any of number of items:

- The development process being used
- The pace of development
- Coding standards
- The development environment or equipment
- Office arrangement
- Communication within the team or to external stakeholders
- Quality improvements
- The way in which the team interacts
- Interpersonal challenges

WHAT VALUE DOES IT HAVE?

Retrospectives are conducted to help teams continuously improve their performance by reflecting on areas to improve and establishing actions for making the improvements. They are used to not only find areas to improve the team, but also obstacles that may need to be elevated to management to be resolved.

WHO IS IT FOR?

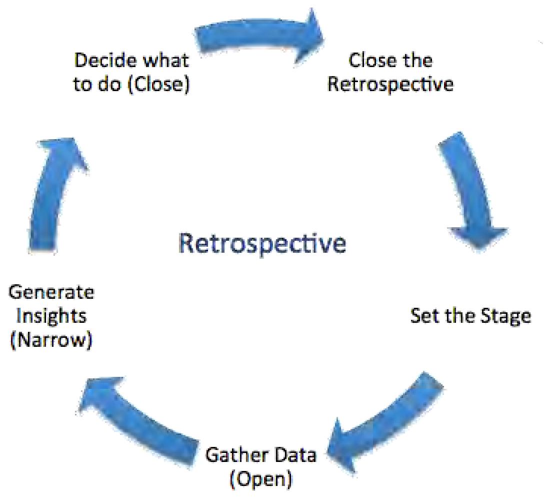
Retrospectives are conducted so that the team can incorporate continual learning and improvement into the work they are doing. The entire team, Developers, Leads, Architects, Testers, tech Writers, etc., should attend as well as any subject matter experts or business representatives the team feels could provide valuable contributions. In particular, individuals that contributed to the work since the last retrospective should be involved so they can provide feedback for improvements. Facilitators for the session are typically Scrum Masters, Project Managers, or team members.

WHEN IS IT CONDUCTED?

Retrospectives should be conducted at a regular and on-going basis, preferably no more than a month apart with a bias towards a shorter interval. This keeps information fresh in people's minds as they look for ways to improve.

IMPORTANT THINGS TO KNOW ABOUT RETROSPECTIVES

The retrospective is conducted as a facilitated meeting usually immediately following the demonstration of the working software to date and before the planning of the next set of work to be pulled from the Backlog. There is a suggested process for conducting a retrospective to discover the most valuable set of actions or experiments. The following diagram shows this process approach.

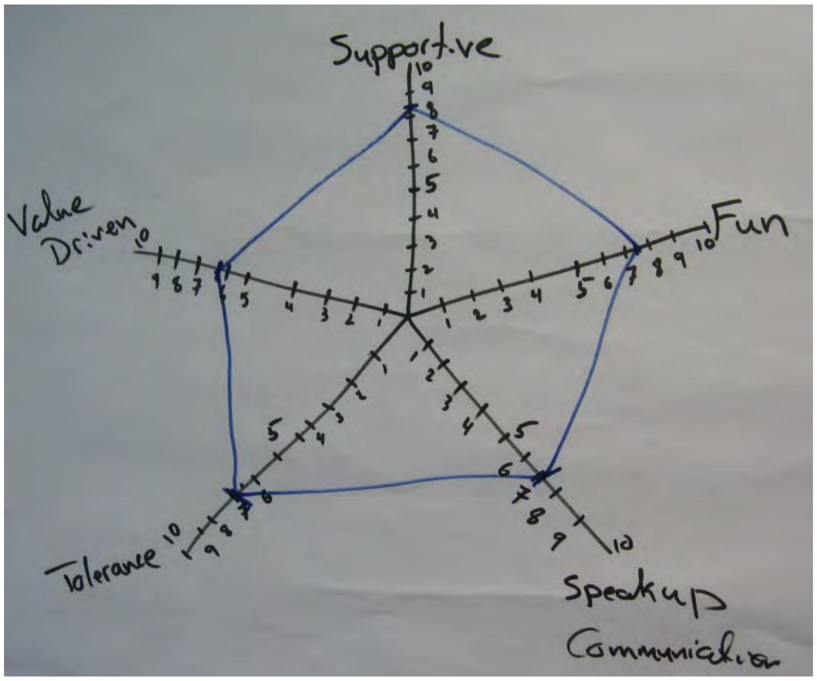


The process starts with Setting the Stage, which places the team in reflective mode. From there it moves to gathering data (perspectives, opinions, or facts), creating understanding of these to narrow on the core addressable problems, and then finally making decisions on one to a few actions to take. Once these actions are decided, the retrospective is closed with a mini-reflection on how well the retrospective went as well.

There is a Prime Directive[1] about conducting a retrospective:

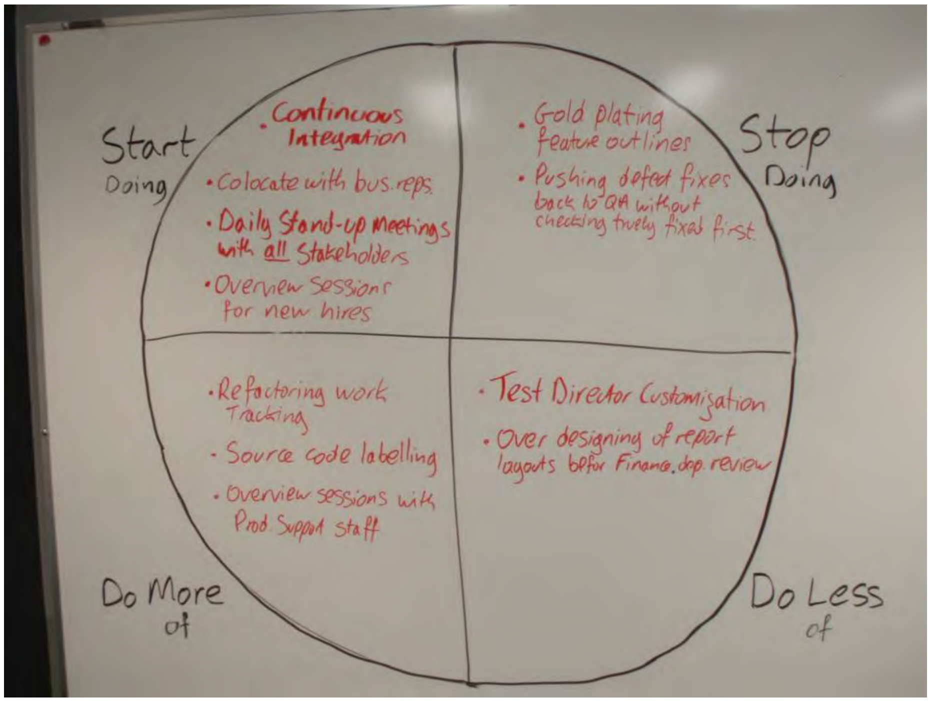
> *Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand.*

Retrospectives are about the way employees work together and not about the product or schedule. They should be conducted at regular intervals so that an atmosphere of continuous improvement can be nurtured. A number of structured exercises can be used; the following photos show the results of some of the facilitated exercises that have been used in Retrospectives.

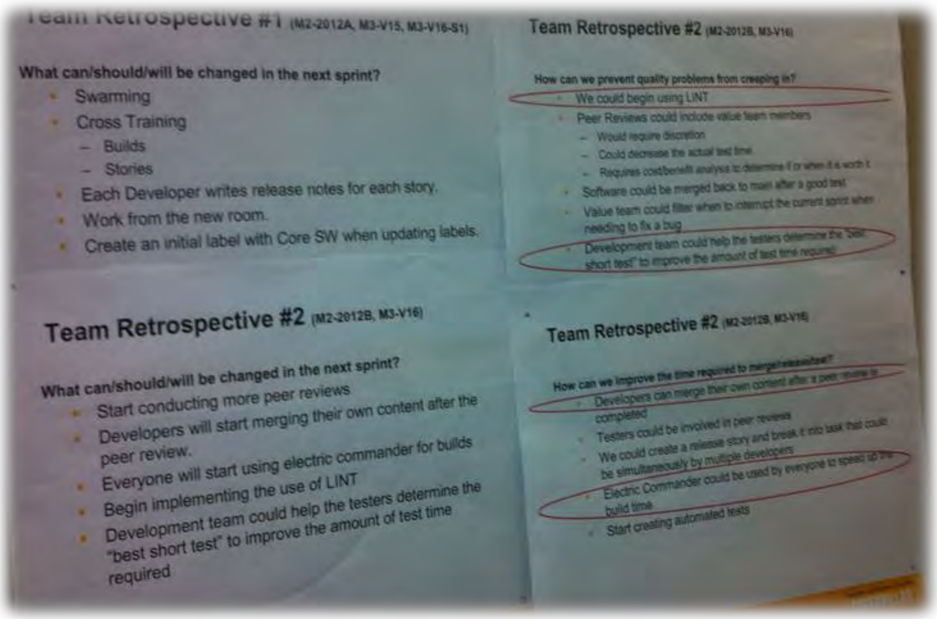


In the example above, the team developed a RADAR chart showing how well they were doing in each of five categories they had identified (most likely as part of developing the Team Charter). It's important to run these as facilitated exercises and discuss outlier opinions and perspectives. If for example the last two weeks was fun for all but 1-2 people, understanding the 'why' behind this is important to ensure the entire team is as effective as possible.

[1] The Prime Directive was created by Norman Kerth, see http://www.retrospectives.com/pages/retroPrimeDirective.html


Start Doing
• Continuous Integration
• Colocate with bus. reps
• Daily Stand-up Meetings with all Stakeholders
• Overview Sessions for new hires
Stop Doing
• Gold plating feature outlines
• Pushing defect fixes back to QA without checking truly fixed first
• Refactoring work Tracking
• Source code labelling
• Overview sessions with Prod. Support staff
Do More of
• Test Director Customisation
• Over designing of report layouts befor Finance dep. review
Do Less of

The above diagram is typical of many Retrospective exercises where actions are being evaluated (which could also be what to stop doing/bad habits to break). Retrospectives can also be posted as Information Radiators, as the following diagram shows.



Team Retrospective #1 (M2-2012A, M3-V15, M3-V16-S1)

What can/should/will be changed in the next sprint?

- Swarming
- Cross Training
  - Builds
  - Stories
- Each Developer writes release notes for each story.
- Work from the new room.
- Create an initial label with Core SW when updating labels.

Team Retrospective #2 (M2-2012B, M3-V16)

What can/should/will be changed in the next sprint?

- Start conducting more peer reviews
- Developers will start merging their own content after the peer review.
- Everyone will start using electric commander for builds
- Begin implementing the use of LINT
- Development team could help the testers determine the "best short test" to improve the amount of test time required

Team Retrospective #2 (M2-2012B, M3-V16)

How can we prevent quality problems from creeping in?

- We could begin using LINT
- Peer Reviews could include value team members
  - Would require discretion
  - Could decrease the actual test time
  - Requires cost/benefit analysis to determine if or when it is worth it
- Software could be merged back to main after a good test
- Value team could filter when to interrupt the current sprint when needing to fix a bug
- Development team could help the testers determine the "best short test" to improve the amount of test time required

Team Retrospective #2 (M2-2012B, M3-V16)

How can we improve the time required to merge/release/test?

- Developers can merge their own content after a peer review is completed
- Testers could be involved in peer reviews
- We could create a release story and break it into task that could be simultaneously by multiple developers
- Electric Commander could be used by everyone to speed up the build time
- Start creating automated tests

.

TIPS AND TRICKS

Always conduct the retrospective as a facilitated meeting with a facilitator that can remain neutral during the meeting. This prevents bias towards a path that may not have team buy-in. The facilitator should plan adequate time to determine and develop the exercises to be used. There should be at least one exercise per "phase" of the retrospective. It is also a good idea to have an alternative exercise for each phase should one exercise appear not to be producing desirable outcomes.

It should be remembered that in a retrospective there is no place for analyzing individual behaviors or blaming anyone; additionally, many of the possible insights will highlight organizational issues outside the control of the team. Blaming creates an environment where some team members will stop contributing and buy-in will be lost. Since these behaviors are outside of scope of authority of the team, the team needs to determine whether a work-around can be put in place and/or whether to raise these issues with management.

A retrospective should also not focus on product improvements, design, or other project specific items; the focus needs to be on the team's working relationships. Retrospective facilitators should design retrospective exercises such that it draws from all team members as much as possible.

Here are some simple suggestions with respect to Retrospectives:

- Discuss any personal, team or process issues openly.
- Discuss what worked and what needs to change.
- Agree on top items to be addressed and fixed.
- Review these at the beginning of the next retrospective.
- Add the 'Appreciation' game every now and then. (A simple exercise that helps improve morale by having team members express appreciation for each other.)

When teams find Retrospectives getting boring often they stop holding them[2]. This is extremely detrimental as this self-reflection is focused on helping the team improve during project execution. The easiest way to avoid this is by using different exercises to keep the retrospectives interesting while still providing the needed value. A good source of exercise material and guidance is the book *Agile Retrospectives by Esther Derby and Diana Larsen* (easily found on Amazon.com) and the website Tasty Cupcakes (http://tastycupcakes.org). Several of the "games" at Innovation Games (http://innovationgames.com/) can also be adapted for retrospectives and the site provides a platform for performing their exercises with teams working in remote locations.

It is important to follow-up on actions that the team decides with which they want to go forward. This ensures accountability in getting the action taken and allows feedback to see if the action is working for the team. One method for making the decision on the action visible is to have it as an information radiator; this is particularly important if it is a habit the team has decided to adopt (such as check in code at least daily!). Another is to start the next retrospective with a review of the action from the last retrospective.

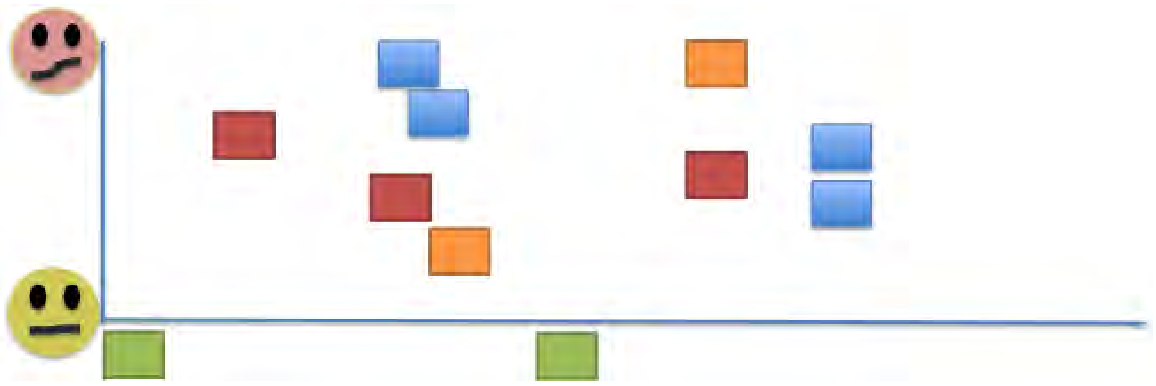
[2] Nick Osstvogels has an excellent post on this at http://skycoach.be/2009/07/27/boring-retrospectives/

EXAMPLE EXERCISES

TIMELINE EXERCISE

The Timeline Exercise is very useful for gathering data.

Create a vertical axis to show how work was impacted, with the upper bound being very disruptive and the lower point being a mild annoyance. The horizontal axis represents the timeline being examined (usually since the last retrospective).



Team members place different colored stickies for different categories of issues that may have occurred. Perhaps it was a breakdown in **communications**, a **technology** element that wasn't working, or a **process** step that seems to add no value. The categories should be tailored to some understanding of concerns the team may have. The members place these on the timeline according to their memory of when they occurred and the impact on them. If they had a specific bright moment that made them happy, have them use a color sticky (such as green) and place it below the timeline when it occurred.

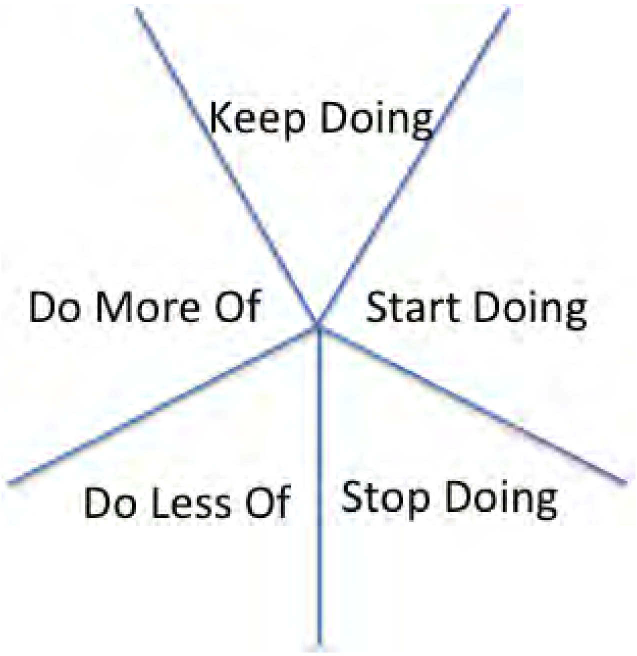
Once all the stickies have been posted, look for patterns. Were there similar posts? Was there a specific point in time where many occurred? Did one event cause several others? Are there more of a particular category?

The Timeline exercise generates a lot of data quickly about events and impact. Look for high impact events that occurred to dig into root cause analysis (with a different exercise).

STARFISH EXERCISE

The Starfish Exercise is very useful for deciding on an action plan.

There are five areas drawn out: Keep Doing, Do More Of…, Start Doing, Do Less Of…, and Stop Doing.

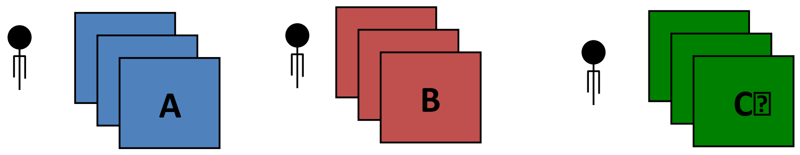


Have team members write things they experienced/observed for items in each category. Perhaps it is bad habits they want to discard (Stop Doing) or they want to set-up and use Continuous Integration (Start Doing). Perhaps the release notes that a member is writing are very useful and the team would like her to train the rest on how she does them (Do More Of…).

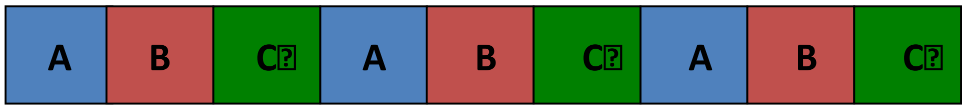
The Starfish Exercise generates and organizes ideas for potential improvements very quickly. If there is an overabundance of ideas, try grouping around a specific concept or multi-vote (dot vote) to show the ones that people are most interested in. Remember to keep these documented so they can be revisited at a later point to see if they are still relevant.

MULTI-TASKING AND PRIORITIZATION

Imagine that you have 3 managers, and 3 equally important initiatives.

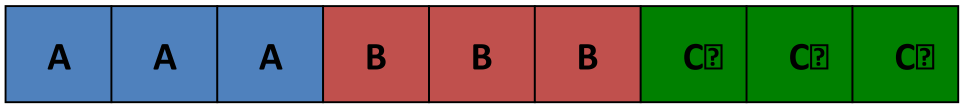


You can try to show progress every month, something like this …

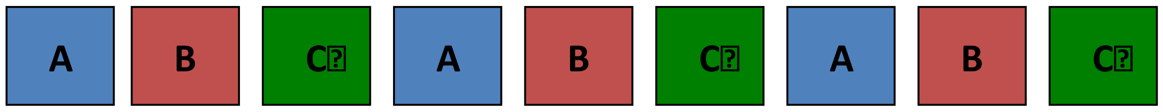


This way each manager sees progress every period …but the business gets no benefit before timeslot #7, when the last task for Initiative A is finished.

Or you can establish an organizational priority, possibly something like this …



With organizational priority, the business can potentially see completion of the higher-priority projects sooner, but when you factor in task switching and interruptions, in reality, for the people doing the work, it will likely still look more like this …



AGILE ESTIMATION

PURPOSE / VALUE OF ESTIMATING:

The primary reason to perform estimation is less about the estimate and more about the conversation within the team about the work and its complexity. By working together to understand how an estimate can be made, team members gain a shared vision into each other's thoughts on the work at hand.

ESTIMATING TECHNIQUE: RELATIVE SIZING

The first step towards Agile estimation is usually relative sizing. The most common of these is a set of T-shirt sizes: S, M, L, XL, XXL. This allows team members to gain an understanding from low-effort/complexity to high. By doing this, it helps focus the team into what areas present the most risk in terms of jeopardizing delivery (particularly if the project is date constrained).

Another set that Matt Barcomb came up with is to use Woodland Creatures[3]:

- Shrew
- Squirrel
- Badger
- Boar
- Deer
- Bear
- Elk
- Caribou
- Dragon*

The larger the creature, the more challenging it will be. His reasons for using Woodland Creatures as opposed to t-shirts or other more "stepped" sizings are as follows:

1. One can't do math with them (though they do multiply!)
2. They imply a size or category, but not an accuracy; and
3. They keep the team from taking the activity too seriously, but differences in opinion still get discussed.

Avoiding math is directly aligned with not implying an accuracy or precision that is beyond an understanding the team has.

*The Dragon has a special use; it is where the team has very little understanding and where additional work may be needed to slay it and gain that knowledge. This could be done with a development spike, additional research, or some other set of special tasks the team may need in order to gain that understanding. It is basically indicating a high level of uncertainty that no one on the team can provide the insight into how to solve.

[3] See Matt's blog post http://blog.risingtideharbor.com/2011/05/woodland-creature-story-sizing.html

ESTIMATING TECHNIQUE: STORY POINTS

Many Agile teams do estimation in story points; this is normally a Fibonacci sequence (1,2,3,5,8,13,21…) or a modified Fibonacci sequence (such as 0.5,1,2,3,5,8...) Use of such a sequence is helpful because the comparisons aren't simply twice as big or a third of the size, which implies a precision that is unlikely to be true. Because the numbers don't follow this convention, it provokes conversation into the work to be done, rather than focusing on the numbers.

Many teams use Planning Poker to generate a team-based sizing in story points. The idea behind Planning Poker[4] is simple. Individual stories are presented for estimation. After a period of discussion, each participant chooses from his own deck the numbered card that represents his estimate of how much work is involved in the story under discussion. All estimates are kept private until each participant has chosen a card. At that time, all estimates are revealed and discussion can begin again. The benefit to using Planning Poker is some basic understanding of the story is generated, followed by a check occurs to see what the team collectively thinks of the sizing. If there is a consensus, a short simple discussion can occur on the tasks that need to occur. If there is a large disparity with some of the team members, then there needs to be further discussion as to why this disparity exists.

ESTIMATING TECHNIQUE: INCH-PEBBLES[5]

Many teams really feel a need to have their estimates in hours or days. When moving from high level release plan level estimates to iteration level estimates, stories and tasks can be split down further with the rule of thumb becoming that the tasks need to be no more than two days of work. This is what is referred to as "inch pebbles". This ensures the team fully understand the work well enough that the estimates are likely to be accurate, even though they may not be precise (i.e. a two day estimate may really be 3 days, but is unlikely to be a full week).

In order to get the stories or tasks this small for the iteration, there needs to be considerable discussion to break them down. This is perfect for the rolling wave/adaptive planning and requirements elaboration that is expected on Agile software projects.

ESTIMATION IS WASTEFUL?[6]

Lean Thinking suggests that estimation is to be considered wasteful. Instead, the throughput time (lead time) for planned items is projected by using previous measurements. This is very much how Velocity is used. After a few (2-4) iterations, there is a sense to how much work can be performed.

---

[4] http://www.planningpoker.com/ has a tool useful for distributed teams that want to use Planning Poker. This short description comes from that site.

[5] Inch-Pebbles were developed by Johanna Rothman, see http://www.jrothman.com/1999/01/how-to-use-inch-pebbles-when-you-think-you-cant/ for more detail.

[6] For further information on this concept in much greater detail read Kanban by David Anderson and see http://qualityswdev.com/2010/01/07/is-estimation-waste/ .

Lean approaches such as Kanban measure the cycle-time of how long each item takes to be performed. Once a statistically significant sample is collected, the average of these along with the standard deviation (or two standard deviations for high level of confidence) provides a prediction of how long any particular item is likely to take. As more items (features/stories/tasks) get completed, the accuracy and precision of how long any item will take can be better predicted. Thus estimation is never done, only the calculated predicted cycle-time range based on the average with 1-2 standard deviations. This particular technique is *very useful for maintenance type activity.*

AGILE ANALYSIS AND DESIGN TOPICS

BACKLOG

WHAT IS IT?

A Backlog is a value-based compilation of work items (stories) that is ordered by importance. The Backlog is a tool for the team to use to prioritize the stories. The Backlog allows for longer range planning of work. The Backlog is a planning tool that allows the business or subject matter experts to communicate their needs to the development team. It allows for the prioritization, estimation and scheduling of development activities within the releases.

WHAT VALUE DOES IT HAVE?

The Backlog is essential for allowing teams to collaborate in one space and gain a prioritized list of activities and clear vision of the work that needs to get done. This Backlog allows customers to see easily when their deliverables can be available. The Backlog enables setting priorities within the team through the visibility it provides.

WHO IS IT FOR?

The Backlog is intended for the entire team and ensures they know what work is next in line to be done and the priorities. It allows the business to adjust priorities of items that before they get worked. In Agile the **Team** defines and divides the desired work into **vertical slices** of **valuable** business functionality. There usually is one individual for (in Scrum, the Product Owner) responsible for ensuring this reprioritization occurs within the team. The team uses this list to ensure it is working on what is most important to the business.

WHEN IS IT CONDUCTED?

Within the context of the overall flow of planning and work, the Backlog is created after the team is created and aligned. The Backlog is one of the artifacts/outputs of the initial alignment, as part of the Sprint Zero or Discovery work. The Backlog gets constantly updated throughout the project, usually through grooming at the start of each development with the business or subject matter experts regarding the prioritization at regular intervals.

IMPORTANT THINGS TO KNOW ABOUT BACKLOGS

The Backlog is not a static list. Work items are continuously prioritized to ensure that the delivery team is aware of highest priority items, and that the team is providing the most value out of near term deliverables. Continued prioritization of this Backlog ensures continued discussion of the work ahead as well as reflection of the priorities of the current items. Wide-band Delphi size estimation of the Backlog items allows us to more accurately predict when the work will be completed based on current priorities.

In short, maintaining a prioritized Backlog will ensure that we are always working on the highest priority items and allow us to estimate when our work will be complete.

Simply stated, the Backlog serves as a tool for communication, prioritization, and collaboration. It is important that this be a single, prioritized list is about prioritization and value, such that on-going discussions of work to be done. This allows adjustment of priorities up until work begins.

The Backlog is created and managed by the team in conjunction with the business. Each work item represents a vertical slice of functionality that delivers value to the user, customer, or organization. Each work item is a placeholder for a later conversation to clarify the intent, the details, the acceptance criteria, and gather any other information that is necessary to deliver the functionality.

Serving as a to-do list, the Backlog represents *all known work* that the team has yet to do. While it is relatively common to have more work delivered to the team during the lifecycle of a project, that work should also be added to the Backlog. All work on the Backlog should have defined *value* that it will deliver, and the Backlog should be prioritized on an ongoing basis. That is, at any given point in time, the Backlog reflects the Team's and Customer's[7] current assessment of priorities.

The amount of work selected from the list should be pulled based on the team's capacity. Over time, this will be discovered. All efforts should avoid pulling in more work items off the Backlog than can be handled as it leads to multi-tasking and slows the work done on each work item down. The intentional limiting of the amount of work being pulled is known as limiting work-in-progress (WIP).

When determining what to pull, the level of granularity of the work should be divided such that it is expected to, at a maximum, only take few days of effort. This keeps work items from taking two long and has the following benefits:

- The team members gain a sense of accomplishment of getting work completed.
- It lessens the likelihood of pressures to pull in additional work, which would increase multi-tasking.
- It ensures the work is well understood.

TIPS AND TRICKS

There are several attributes of a Backlog to consider:

- Good visual for the work that needs to get done (*Information Radiator, Visualization*)
- Prioritize work (*Prioritization*)
- Tracking of work (*Visualization, Information Radiator, Project Management, Product Management*)
- List of to-dos (*Planning*)
- Planning tool (*Planning*)
- Builds collaboration (*Collaboration*)
- What's being done (*Visualization*)
- Something to talk about related to work and priority (*Collaboration*)

[7] Customer being Process Partner or otherwise.

- Ensures we are working on highest priority items (*Product and Release Management*)
- Helps gain consensus (*Collaboration*)
- Tool to know what is on our plates (*Planning*)
- Long range planning and resource requirements (*Planning*)
- Get a better idea on when things might be delivered (*Planning*)

In Agile the **Team** defines and divides the desired work into **vertical slices** of **valuable** business functionality.

These vertical slices are not defined in detail, to allow room for discovery. They are insufficient to implement without a **conversation** between the customer and delivery team[8].

To **predict and protect** the project's time commitments these vertical slices must have just enough detail to estimate[9] them.

These vertical slices should focus on features that describe components of end-to-end working piece of software. These will be progressively elaborated to flesh out the tasks needed; the level of granularity of these features – or work items – gets finer and finer until the team begins the development. At this point of the planning, the stories should be at an *optimal level of granularity*. There is no fixed definition for this. Each team defines it ("optimal level of granularity") for itself learned through experience. A working guideline for this is that no work item should be measured in minutes, and should probably not be smaller than a half-day. However, as noted earlier, this is not hard and fast.

Through breaking a story down into tasks, when appropriate, the team will discover whether a story is too large or not. Some teams use a particular size that the team discovers is what is good for committing to being done. Others use the concept of inch-pebbles; these are stories or tasks that members feel can be completed within just 1-2 days. This short duration keeps tasks from taking so long that their completion can't be understood easily.

CHARTERING

Chartering is the process of coming to agreement on what is to be done, by whom, and how the members of the project team are to behave with and toward each other. In the following sections on Team and Project Chartering, these details are spelled out.

It is important to note that these are two separate events with two separate sets of outputs produced. The order of the two events is not predetermined.

[8] The Three C's: Card, Conversation, Confirmation.

[9] Estimating is covered briefly in the course. More detail can be found in James Grenning's original paper on Planning Poker.

TEAM CHARTERING

When forming a team to execute a project, it works under a charter. Both traditional and Agile approaches have the concept of a Project Charter, but high performing Agile teams also have a Team Charter. Additionally, Project Charters have some differences when created by Agile teams from traditional charter approaches.

WHAT IS IT?

A Team Charter is designed to communicate the established team norms with the purpose of helping the team lay down the ground rules they need to self-organize and self-govern and to move towards high-performance. The Team Charter helps answer the following questions:

- How will we make decisions?
- How will we handle conflict?
- How will we run our meetings?
- How do we communicate?
- What are our values?
- How will we manage our work?
- How do we improve our processes?
- How do we have fun?

WHAT VALUE DOES IT HAVE?

Many teams adopt working agreements based on ensuring consideration is made for all team member's perspectives, the methods and conditions for team decision-making, and other key team dynamics. When teams don't have a Team Charter, there is a risk that the team members will not have a shared understanding of how they operate and communicate amongst themselves and how the inevitable conflict is handled so that it results in a constructive approach. This leads to unneeded frustration and stress that can degrade team performance. The Team Charter eliminates the confusion and establishes the expectations for each individual as he or she contributes. By using a facilitated session in its creation, everyone has a say in how it is formed. The Team Charter ultimately empowers the team to align themselves to develop and adapt a work environment that results in greater effectiveness and collaboration.

The Team Charter is used for the following:

- Establishing how the team wants to work together
- Defining specific roles and responsibilities

- Documenting expectations about execution
    - Align the team and get the most effective team
    - Establish roles and responsibilities
- Gaining an understanding of what all of our team members value
- Creating high-performing teams
- Building relationships between team members
- Establishing how the team wants to resolve conflict
- Agreeing on what tools team members are going to use
- Creating a common, shared understanding
    - Work as a TEAM to work towards a goal
    - Create a common focus for the team – marching in the same direction
- Determining what teamwork means for the team to work towards a goal
- Creating an atmosphere where individuals put the team above themselves
- Making work a fun environment and increasing employee satisfaction
- Helping people work together and make effective use of time
- Creating better quality products, and lastly;
- Reducing wasted time that can result from simply guessing what to do

To develop and communicate these items, a Team Charter may contain sections that answer the following questions:

- Who are we?
- What are our team values?
- How will we communicate?
- How will we make decisions?
- How will we handle conflict?
- How will we run our meetings?
- How to give each other feedback?
- How to hold each other accountable?
- How will we manage our work?
- How do we have fun?
- How do we improve our processes?
- What is our working agreement?
- What is important information to us?

- What information do we need to radiate and how?
- What are the roles and responsibilities of everyone on the team?
- What is our definition of done? (Before anyone on the team can call something done, what is the team's expectation of activities to be completed – there is a whole section on this practice)

The facilitator of the Team Charter should put time and effort in the planning of the charter to make sure that they address the most important issues the team needs to align on and discuss. It is rare that a team charter would include all of the above topics.

WHO IS IT FOR?

The Team Charter is intended for the entire team regardless of whether the team member represents the business or the development activities.

A team, by definition, consists of the group of people that work or communicate together frequently towards a common deliverable or outcome.

The Facilitator (the leader of the team who is responsible for the delivery the team's output or content), is responsible for facilitating the Team Charter session(s) and making sure that the team has an updated team charter.

WHEN IS IT CONDUCTED?

A Team Charter is initially created using a collaborative/facilitated session at the beginning of the project team's formation, with every member of the team involved in its creation. It is a living document and may be updated by the team as it learns during the work it performs. These updates usually occur when the team membership significantly changes, whenever the team feels there is a lack of alignment, or as a result of decisions made during a Retrospective (covered in Section 6).

IMPORTANT THINGS TO KNOW ABOUT TEAM CHARTERING

The Team Charter should be focused on the Team and how it will work together. It should be created with participation of all team members performing the work since ultimately this helps the team 'gel' into a cohesive unit. This is why the use of a facilitated session is so important as it gathers every member's perspective.

As the project is executed, the working agreements may need updating when team membership significantly changes or from learning that occurs on the project. The latter usually happens as a result of a decision from a retrospective.

The Team Charter is not intended to have any specifics about the Project itself; that is for the Project Charter.

TIPS AND TRICKS

Having a common understanding on working agreements is essential to high performance. Developing these using facilitation and allowing each member to voice what concerns them in how they work will produce not only this common understanding, but an understanding of each member of the team and their preferences.

Many teams are beginning to adopt the Core Commitments and Core Protocols[10] as a starting point for improving their team effectiveness. These provide a means for establishing communications and working relationships that allow people to manage their emotions (as a team) during work while not ignoring that they exist. To learn more on these, check the References section.

It is common that the Working Agreements that are created become an Information Radiator posted within the team's space serving as a reminder of the agreed upon norms for the team.

Even if a team has been working together for years, it is good to go through a Team Chartering session at the beginning of a new project to revitalize the working agreements and assumptions everyone has been using.

[10] See http://www.mccarthyshow.com/wp-content/uploads/2011/02/The+Core+Protocols+3.03.pdf for a copy of the Core Protocols and Core Commitments

PROJECT CHARTERING

WHAT IS IT?

A Project Charter is focused on aligning the team on different aspects of the project work that they will be doing including but not limited to: vision, goals, values, and measures of success. It is created at the start of any project or program and renewed semi-annually or whenever the program sponsor changes. A typical project charter is created using a series of Chartering sessions that consists of a series of facilitated exercises to –

- Orient the team on the product idea
- Bring the team up to speed on the feasibility work that has already been performed
- Synchronize the customer and the project team on the project's value and goals

It also produces a living document that creates a shared understanding with the value team and delivery team as well as external stakeholders on the reason for the team's existence. A project charters may contain sections such as:

- Business Case/Opportunity Statement/Vision
- Goals
- In-Scope/Out-of-Scope
- Measures of Success
- Trade-off Matrix
- Strengths/Challenges/Roadblocks
- Definition of Done (for the Project)
- Team Members/Stakeholders; Names and Availability (if not dedicated to the Team)
- Milestones

Since the purpose of the chartering exercise is to gain alignment between the various team members on the project we recommend considering the following sections to your project charter:

- Definition of Done (Release level)
- Who will work on the backlog
- Communication Plan (what will be communicated, and how, throughout the project)
- What information needs to be radiated and how?

WHAT VALUE DOES IT HAVE?

There are two specific goals of the Agile Project Charter: 1) align all the stakeholders as to what the customer value is and 2) empower the team (both value and delivery) to make decisions on features that align with this agreed upon customer value.

WHO IS IT FOR?

The Project Charter is for not only the team performing the work; it is also for all the external stakeholders. This provides a clear shared understanding of why the team is doing the work and why these stakeholders should support them. These external stakeholders may participate in the Project Chartering session(s). Even if sub-system teams have conducted their own team or project charter, they should participate to the “entire team’s” project charter. The purpose is to align everyone involved which means that we may need to conduct another chartering session with everyone involved. Moreover, the chartering session may include elements from the Team Chartering session if the facilitator feels that alignment is needed not just about the project content but also about how the greater team will work together.

WHEN IS IT CONDUCTED?

Project Charter sessions should be conducted at the beginning of the project once the Team Chartering session has been completed. It may take more than one session to cover all the needed areas to be investigated. Projects should re-charter if there is major changes in the team composition or in the direction of the project.

IMPORTANT THINGS TO KNOW ABOUT PROJECT CHARTERING

The Project Charter provides a means for orienting people as to the purpose, expected benefit, and other significant attributes of the project. It serves as a common reference/reminder to the project team throughout project execution and allows external stakeholders to quickly understand the purpose of the project.

TIPS AND TRICKS

Project Charters should always emphasize the value of the project. A great way of expressing the customer value of the product to be developed is the Elevator Statement, which could be included in the Business Case/Opportunity Statement/Vision section. The Elevator statement takes the following form:

FOR <*target customers*>
WHO <*statement of need*>
THE <*product name*> **IS A** <*product category*>
THAT <*key benefit, reason to buy*>
UNLIKE <*competition, alternative*>
OUR PRODUCT <*differentiating statement*>

This allows expression of the product value in a concise format upon which people can easily align.

The Trade-off Matrix helps align decision-making and risk mitigation strategies so that team members can understand some of the scope of authority to which they have. There are three categories of allowable change:

- Fixed: Will not change
- Firm: Will only change under extreme circumstances
- Flexible: Flexible

A sample format of a Trade-off Matrix is shown in the following diagram. While this chart shows the generic Scope, Schedule, and Cost (the Iron Triangle), these could be any relevant project categories. Other possible categories could be Target Price, Market Share, or Allowable Defects.

	Fixed (1)	Firm (1)	Flexible (n)
Scope			X
Schedule		X	
Cost	X		
…			

If all of the category items are fixed, then the team effectively has no trade space; this should be discouraged or further investigation should be done to find factors that can provide some flexibility.

Another form of the Trade-off Matrix is the Project Success Sliders, which are available as a simple web-based application at Mountain Goat Software. These provide a means of adjusting one particular topic or category and when all are in balance is there a green check mark indicating balance. Thus if one wanted to say staying within budget has the greatest importance, something else would have to decrease in importance in order to compensate. You can find the Success Sliders at http://www.mountaingoatsoftware.com/tools/project-success.

Ideally Project Charters should be created after Team Charters so that the working agreements that have been put in place are in effect during its creation. This provides for more effective collaboration during its development and allows the team to exercise the working agreements that were created before the project begins execution.

PRODUCT QUALITY TOPICS

DEFINITION OF DONE

WHAT IS IT?

Per the Agile Alliance, *the team agrees on*, and normally displays prominently usually as an Information Radiator, a list of criteria, which must be met before a product increment (often described as a specific feature requirement, use case, or a user story) is considered "done". This is the Definition of Done. Failure to meet these criteria at the end of the work at hand normally implies that it should not be counted toward that project's progress or velocity (i.e. the team may not take credit for this work as working software in the iteration). It is important to obtain this agreement as a part of the Team Charter, since it defines and communicates this as a common understanding to the team and its stakeholders.

A simple example for a team's Definition of Done could be:

- Finished writing code
- User has approved feature
- Unit tests all run successfully
- No issues remain open in tracking system

This list would then be the minimum requirements for a work item to be considered complete.

WHAT VALUE DOES IT HAVE?

Having a Definition of Done has the following benefits to the Team and the organization:

- Everyone is on the same page for what the completion of work means.
- Since it was developed by the team, there is buy-in.
- It helps in estimating; the 'done' criteria helps the people doing the work think through what is involved as far as tasks.
- It helps with elaborating requirements from the high level to the detailed level.

WHO IS IT FOR?

This is for the project team that is delivering the work. If external considerations should be adopted into the definition, a stakeholder that has that consideration should be a part of the chartering to help define this piece.

WHEN IS IT CONDUCTED?

Creating the Definition of Done for the team is done during the Chartering and updated if necessary when Retrospectives discover the need to do so.

IMPORTANT THINGS TO KNOW ABOUT THE DEFINITION OF DONE

It must be stressed this is a team definition and needs to be communicated clearly to every member once the team makes the decision on this norm. Here are some examples:

Team 1:

- Write code according to coding standards
- Create Automated Tests for code
- Pass All Automated Tests
- Ensure all code is checked in
- Conduct Peer Review
- Update design documentation
- Update Release Notes

Teams will adopt a Definition of Done that best fits the context of work that they are providing during their Chartering sessions and can update it as they learn more performing the work; the updates usually occur as an action from a retrospective.

TIPS AND TRICKS

The Definition of Done could take the form of Must Do, Should Do, Could Do. That is to say, before you can claim anything as done, there are items that must be completed (Must Do), ones that should be completed to have others feel that it was done correctly, though it may meet the team's requirements of being satisfactory (Should Do), and those optional things that would improve the experience for everyone involved (Could Do).

It is often useful to have the Definition of Done for the team posted as an Information Radiator so that everyone clearly understands what is it is; this creates a shared understanding and orients new team members or external stakeholders what 'done' means to the team.

The Definition of Done is more than acceptance criteria; that is only a portion of 'done' from a user perspective. There are often additional criteria such as documentation, testing, or reviews that have to be performed to meet the team's definition.

APPENDIX B: AGILE KNOWLEDGE AND SKILLS (PMI-ACP EXAM TOPICS PART II)

LEVEL ONE TOPICS

THE 12 AGILE PRINCIPLES

1. EARLY AND CONTINUOUS DELIVERY

> *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*

This begins by expressing the commitment to the customer (Process Partner, user). Additionally, it addresses the idea that by delivering[11] valuable software early, the customer has a chance to provide valuable feedback, help make decisions about redirection, and constantly assess the value of the work that has been done.

This is reflected in practices such as continuous integration, iterative development, continuous prioritization, and the structure and content of user stories. This principle, and the practices that manifest it, contribute to higher quality, more feedback, better collaboration, and higher satisfaction.

2. WELCOME CHANGING REQUIREMENTS…

> *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*

The world changes. Technology changes. The business and customer needs change as their understanding and experience change. This principle reflects the need for a process – iterative with constant reprioritization – that fully recognizes that changes happen and contribute to the success of the project. Projects – and Agile processes – specifically address this with the goal of turning change into value, rather than resisting change and delivering what the customer thought they wanted when the project was being defined.

While the idea of "customer's competitive advantage" may seem less relevant at Caterpillar Electronics, this can readily be translated to "successful completion of sales efforts." Having a process that permits teams to accept and embrace changes during the process of development means that end customers are likely to be far happier with the results.

[11] "Deliver" does not necessarily mean pushed to production. Rather it means making the software available to users/partners/testers in a manner that allows them to work/play with it.

3. DELIVER WORKING SOFTWARE FREQUENTLY…

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

As noted in the previous principle (changing requirements), delivering software frequently provides an opportunity for the customer/user/partner/tester to provide valuable feedback. Unlike the traditional waterfall approach in which the customer doesn't see anything until it's all done, this principle leads us to practices that provide the customer the ability to change direction and directly influence what is being developed, low level to high level.

In addition to the benefit to the customer, working in shorter timeframes drives the project team to reflect, inspect, and adapt more frequently. It allows them to improve their design and architecture through the process of testing, integrating, and deploying.

4. BUSINESS PEOPLE AND DEVELOPERS…

Business people and developers must work together daily throughout the project.

Traditionally, business people are "them" – the enemy, the other, the demander, the unsatisfied – rather than being considered partners. Conversely, business people frequently perceive the members of the development team as uncooperative, inflexible, and resistant to change. Survey after survey has revealed this dichotomy and the antagonism that goes with it.

This Agile principle leads to practices that contribute to collaboration rather than conflict. When the developers (testers, architects, designers, UX analysts and designers, and so on) collaborate on a daily basis with the business people, the commitment and sense of ownership *on both sides* increases significantly. In addition, accountability and responsibility sharing leads to a higher level of integrity.

5. MOTIVATED INDIVIDUALS…

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

There are several big words in this principle: motivated, support, and trust. Many people mistake this to refer only to the developers, or the delivery team, or the core project team. It applies to anyone and everyone who is a direct (and sometimes indirect) contributor to the success of the project. This refers not only to developers and testers, but to the business people, Process Partners, and others who are a part of the project. If an organization is to embrace principle #4, it means that the organization – the leadership – must support the commitment of time and effort on the part of the members of the organization who usually drop in and out of the life of the project team.

Experience with Agile project teams shows that teams of individuals, especially when motivated by a shared commitment to succeed and a shared sense of ownership, will find ways to achieve that success.

Sometimes this requires leadership/management to trust the team to get their work done. It requires trust to allow a team to experiment, to push boundaries, and to do things in a different way.

6. FACE-TO-FACE CONVERSATION.

> *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*

Research shows that the quality and effectiveness of communication goes down dramatically as the distance between the people engaging in that communication goes up. The greater the distance, the greater the likelihood of lag in communications. Email is a form of inventory/work in progress. Telephone calls are better, but lack the richness of nuance and communication that is found in face-to-face communications. Being present also allows for spontaneous and immediate communication that is relevant, contextual, and reduces the context switching that is needed otherwise.

7. WORKING SOFTWARE…

> *Working software is the primary measure of progress.*

Traditionally, a team spends a significant amount of time writing specifications, architecture documents, test plans, and so on. While all of these items might be considered to be valuable work product, until there is software there is no value being delivered. Working software – that is, software that actually does something useful, as opposed to a piece of something – represents real value being delivered.

When assessing whether a project is approaching completion or not, ultimately the real measure is whether software is being developed, assessed for quality, integrated, and able to be used.

8. SUSTAINABLE DEVELOPMENT.

> *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*

Generally speaking, software development should be considered to be more like a marathon and less like a sprint. Sprinters cannot run races indefinitely, as the extreme effort in each race is exhausting and it takes time to recover. Software projects frequently conduct sprints – not the Scrum sprint, but a race-like sprint – with the idea that everyone on the team can produce that same level of effort indefinitely. While it is done, it generally leads to fatigue, dissatisfaction, and frustration. Marathoners, on the other hand, work to achieve a pace that they can sustain for hours, rather than seconds. Software projects should be considered as marathons, with the goal of making steady, continuous progress that leads to the goal.

Sustainable pace is the idea that we should be able to specify and manage our work in such a way as to work a reasonable day and week, while making steady progress.

9. TECHNICAL EXCELLENCE AND GOOD DESIGN…

Continuous attention to technical excellence and good design enhances agility.

In this principle, the word "agility" is being used in the generic sense: quick and nimble. Good design implies the ability to rapidly and effectively change both the details of the code and the design itself. Technical excellence – referring to code, design, test, and so on – is a reference to the commitment by the team to maintain certain standards. If the team sticks to its commitment, and works to produce good design, it reduces the effort needed to make changes when they are needed.

And change happens.

10. SIMPLICITY…

Simplicity – the art of maximizing the amount of work not done – is essential.

This is not about doing as little work as possible. Rather, it is about doing only as much work as is needed to achieve the goal. It is about committing to the principle of Last Responsible Moment, which suggests that we don't expend effort until it will contribute directly – or closely – to a specific goal.

This is frequently reflected in practices such as not designing a whole database up front, but rather designing and implementing only as much as is needed to support the current work. It is reflected in the practice of building software in smaller pieces in order to learn and discover, rather than assuming that we know everything up front.

11. SELF-ORGANIZING TEAMS

The best architectures, requirements, and designs emerge from self-organizing teams.

Self-organization is the idea that the team, rather than a manager or project manager, makes decisions about the work, how it will be implemented, how it will be architected and designed, and how it will be communicated. Of course, for this to be effective, the definition of "team" must include all parties who have a significant influence on or contribution to the project's completion and success. While traditionally architects may be considered separate from and a contributor to the team, an Agile team considers the architect to be a working member of the team, alongside the "business" (product owner, customer, business analyst, and so on).

In the principle about building teams around motivated individuals, we discussed trust. With the right team members in place, trusted to make decisions and product valuable work, we achieve more effective architectures, requirements, and designs.

12. REFLECTS…

> *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior effectively.*

Agile teams live in a world of continuous improvement, also known as kaizen in the Lean world. Continuous improvement includes process, practices, tools, techniques, communication, collaboration, and more. Agile teams commit specific time and effort on a regular basis to collectively reflecting on how things are going and what they'd like to change. The practice that most directly manifests this principle is the retrospective.

Note that this principle doesn't talk about motivations or feelings, but rather talks about behavior. Behavior includes everything from how we work together to how we write code to the standards we embrace.

LEVEL TWO TOPICS

WHAT IS SCRUM & EXTREME PROGRAMMING?

Scrum and Extreme Programming (XP) are not the same thing as Agile, but rather Scrum and XP are processes and frameworks and Agile is the Mindset upon which these process frameworks were built.

SCRUM

Scrum is a process framework that has been used to manage complex product development since the early 1990s. Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of your product management and development practices so that you can improve. The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.

- Roles (**Scrum Master**, **Product Owner**, and **Development Team**),
- Events (**Sprint** [a time-box of a month or less which is a container for the following additional events]):
    - (**Daily Scrum** [aka Daily Stand-Up], **Sprint Planning Meeting**, **Sprint Review**, **Sprint Retrospective,** and **development work**)
- Artifacts (**Product Backlog**, **Sprint Backlog**, **Increment** [the Increment is the sum of all Product Backlog items completed during the Sprint and the previous Sprints])
- Rules (things that bind together the Roles, Events, and Artifacts, for example, the concept of **self-organizing** and **cross-functional teams,** and **iterative and incremental delivery of "done" product**)

Reference: The Scrum Guide
http://www.scrum.org/Scrum-Guides

EXTREME PROGRAMMING (XP)

Although XP is similar to Scrum in the sense that it too is a process framework, it differs in that it specifies a specific set of practices that should be used during a software project.

XP PRACTICES

- Practices to provide fine scale feedback
    1. **Test Driven Development**
    2. **Planning Game**
    3. **Whole Team**
    4. **Pair Programming**

- Practices to provide continuous process rather than batch
    5. **Continuous Integration**
    6. **Refactoring**
    7. **Small Releases**
- Practices to promote shared understanding
    8. **Simple Design**
    9. **System Metaphor**
    10. **Collective Code Ownership**
    11. **Coding Standards**
- Practices to promote programmer welfare
    12. **Sustainable Pace**

XP PRINCIPLES

The XP principles upheld by teams performing the practices are:

1. **Rapid feedback**
2. **Assume simplicity**
3. **Incremental change**
4. **Embrace change**
5. **Quality work**

XP VALUES

The XP values shared by teams performing the practices are:

1. **Communication**
2. **Simplicity**
3. **Feedback**
4. **Courage**
5. **Respect**

References:

Extreme Programming Core Practices
http://c2.com/cgi/wiki?ExtremeProgrammingCorePractices

Extreme Programming Values
http://c2.com/cgi/wiki?ExtremeValues

PrinciplesOfXP
http://martinfowler.com/bliki/PrinciplesOfXP.html

Cohn, *User Stories Applied for Agile Software Development*

Shore & Walden, *The Art of Agile Development*

CHAPTER 1. VISION AND APPLICABILITY

1.1 VISION AND PURPOSE

As practitioners of project management, we are committed to doing what is right and honorable. We set high standards for ourselves and we aspire to meet these standards in all aspects of our lives—at work, at home, and in service to our profession.

This Code of Ethics and Professional Conduct describes the expectations that we have of ourselves and our fellow practitioners in the global project management community. It articulates the ideals to which we aspire as well as the behaviors that are mandatory in our professional and volunteer roles.

The purpose of this Code is to instill confidence in the project management profession and to help an individual become a better practitioner. We do this by establishing a profession-wide understanding of appropriate behavior.

We believe that the credibility and reputation of the project management profession is shaped by the collective conduct of individual practitioners.

We believe that we can advance our profession, both individually and collectively, by embracing this Code of Ethics and Professional Conduct. We also believe that this Code will assist us in making wise decisions, particularly when faced with difficult situations where we may be asked to compromise our integrity or our values.

Our hope that this Code of Ethics and Professional Conduct will serve as a catalyst for others to study, deliberate, and write about ethics and values. Further, we hope that this Code will ultimately be used to build upon and evolve our profession.

1.2 PERSONS TO WHOM THE CODE APPLIES

The Code of Ethics and Professional Conduct applies to:

1.2.1 All PMI members

1.2.2 Individuals who are not members of PMI but meet one or more of the following criteria:

.1 Non-members who hold a PMI certification

.2 Non-members who apply to commence a PMI certification process

.3 Non-members who serve PMI in a volunteer capacity.

1.3 STRUCTURE OF THE CODE

The Code of Ethics and Professional Conduct is divided into sections that contain standards of conduct which are aligned with the four values that were identified as most important to the project management community. Some sections of this Code include comments. Comments are not mandatory parts of the Code, but provide examples and other clarification. Finally, a glossary can be found at the end of the standard. The glossary defines words and phrases used in the Code. For convenience, those terms defined in the glossary are underlined in the text of the Code.

Santeon Note: Comments, Appendices, and Glossary omitted.

1.4 VALUES THAT SUPPORT THIS CODE

Practitioners from the global project management community were asked to identify the values that formed the basis of their decision making and guided their actions. The values that the global project management community defined as most important were: responsibility, respect, fairness, and honesty. This Code affirms these four values as its foundation.

1.5 ASPIRATIONAL AND MANDATORY CONDUCT

Each section of the Code of Ethics and Professional Conduct includes both aspirational standards and mandatory standards. The aspirational standards describe the conduct that we strive to uphold as practitioners. Although adherence to the aspirational standards is not easily measured, conducting ourselves in accordance with these is an expectation that we have of ourselves as professionals—it is not optional.

The mandatory standards establish firm requirements, and in some cases, limit or prohibit practitioner behavior. Practitioners who do not conduct themselves in accordance with these standards will be subject to disciplinary procedures before PMI's Ethics Review Committee.

CHAPTER 2. RESPONSIBILITY

2.1 DESCRIPTION OF RESPONSIBILITY

Responsibility is our duty to take ownership for the decisions we make or fail to make, the actions we take or fail to take, and the consequences that result.

2.2 RESPONSIBILITY: ASPIRATIONAL STANDARDS

As practitioners in the global project management community:

2.2.1 We make decisions and take actions based on the best interests of society, public safety, and the environment.

2.2.2 We accept only those assignments that are consistent with our background, experience, skills, and qualifications.

2.2.3 We fulfill the commitments that we undertake – we do what we say we will do.

2.2.4 When we make errors or omissions, we take ownership and make corrections promptly. When we discover errors or omissions caused by others, we communicate them to the appropriate body as soon they are discovered. We accept accountability for any issues resulting from our errors or omissions and any resulting consequences.

2.2.5 We protect proprietary or confidential information that has been entrusted to us.

2.2.6 We uphold this Code and hold each other accountable to it.

2.3 RESPONSIBILITY: MANDATORY STANDARDS

As practitioners in the global project management community, we require the following of ourselves and our fellow practitioners:

Regulations and Legal Requirements

2.3.1 We inform ourselves and uphold the policies, rules, regulations and laws that govern our work, professional, and volunteer activities.

2.3.2 We report unethical or illegal conduct to appropriate management and, if necessary, to those affected by the conduct.

Ethics Complaints

2.3.3 We bring violations of this Code to the attention of the appropriate body for resolution.

2.3.4 We only file ethics complaints when they are substantiated by facts.

2.3.5 We pursue disciplinary action against an individual who retaliates against a person raising ethics concerns.

CHAPTER 3. RESPECT

3.1 DESCRIPTION OF RESPECT

Respect is our duty to show a high regard for ourselves, others, and the resources entrusted to us. Resources entrusted to us may include people, money, reputation, the safety of others, and natural or environmental resources.

An environment of respect engenders trust, confidence, and performance excellence by fostering mutual cooperation — an environment where diverse perspectives and views are encouraged and valued.

3.2 RESPECT: ASPIRATIONAL STANDARDS

As practitioners in the global project management community:

3.2.1 We inform ourselves about the norms and customs of others and avoid engaging in behaviors they might consider disrespectful.

3.2.2 We listen to others' points of view, seeking to understand them.

3.2.3 We approach directly those persons with whom we have a conflict or disagreement.

3.2.4 We conduct ourselves in a professional manner, even when it is not reciprocated.

3.3 RESPECT: MANDATORY STANDARDS

As practitioners in the global project management community, we require the following of ourselves and our fellow practitioners:

3.3.1 We negotiate in good faith.

3.3.2 We do not exercise the power of our expertise or position to influence the decisions or actions of others in order to benefit personally at their expense.

3.3.3 We do not act in an abusive manner toward others.

3.3.4 We respect the property rights of others.

CHAPTER 4. FAIRNESS

4.1 DESCRIPTION OF FAIRNESS

Fairness is our duty to make decisions and act impartially and objectively. Our conduct must be free from competing self-interest, prejudice, and favoritism.

4.2 FAIRNESS: ASPIRATIONAL STANDARDS

As practitioners in the global project management community:

4.2.1 We demonstrate transparency in our decision-making process.

4.2.2 We constantly reexamine our impartiality and objectivity, taking corrective action as appropriate.

4.2.3 We provide equal access to information to those who are authorized to have that information.

4.2.4 We make opportunities equally available to qualified candidates.

4.3 FAIRNESS: MANDATORY STANDARDS

As practitioners in the global project management community, we require the following of ourselves and our fellow practitioners:

Conflict of Interest Situations

4.3.1 We proactively and fully disclose any real or potential conflicts of interest to the appropriate stakeholders.

4.3.2 When we realize that we have a real or potential conflict of interest, we refrain from engaging in the decision-making process or otherwise attempting to influence outcomes, unless or until: we have made full disclosure to the affected stakeholders; we have an approved mitigation plan; and we have obtained the consent of the stakeholders to proceed.

Favoritism and Discrimination

4.3.3 We do not hire or fire, reward or punish, or award or deny contracts based on personal considerations, including but not limited to, favoritism, nepotism, or bribery.

4.3.4 We do not discriminate against others based on, but not limited to, gender, race, age, religion, disability, nationality, or sexual orientation.

4.3.5 We apply the rules of the organization (employer, Project Management Institute, or other group) without favoritism or prejudice.

CHAPTER 5. HONESTY

5.1 DESCRIPTION OF HONESTY

Honesty is our duty to understand the truth and act in a truthful manner both in our communications and in our conduct.

5.2 HONESTY: ASPIRATIONAL STANDARDS

As practitioners in the global project management community:

5.2.1 We earnestly seek to understand the truth.

5.2.2 We are truthful in our communications and in our conduct.

5.2.3 We provide accurate information in a timely manner.

5.2.4 We make commitments and promises, implied or explicit, in good faith.

5.2.5 We strive to create an environment in which others feel safe to tell the truth.

5.3 HONESTY: MANDATORY STANDARDS

As practitioners in the global project management community, we require the following of ourselves and our fellow practitioners:

5.3.1 We do not engage in or condone behavior that is designed to deceive others, including but not limited to, making misleading or false statements, stating half-truths, providing information out of context or withholding information that, if known, would render our statements as misleading or incomplete.

5.3.2 We do not engage in dishonest behavior with the intention of personal gain or at the expense of another.

Project Management Institute Code of Ethics and Professional Conduct:
http://www.pmi.org/About-Us/Ethics/~/media/PDF/Ethics/ap_pmicodeofethics.ashx

APPENDIX C: ADDITIONAL TOPICS

SHU – HA – RI

Shu-Ha-Ri is a learning model that comes from the martial art of *Aikido, and Alistair Cockburn (co-author of Agile manifesto) introduced it as a way of thinking about learning techniques and methodologies for software development.*

Shu: In this beginning stage the student follows the teachings precisely. S/he concentrates on how to do the task, without worrying too much about the underlying theory. If there are multiple variations on how to do the task, s/he concentrates on just the one way being taught.

Ha: At this point the student begins to branch out. With the basic practices working s/he now starts to learn the underlying principles and theory behind the technique. S/he also starts learning from others and integrates that learning into his practice.

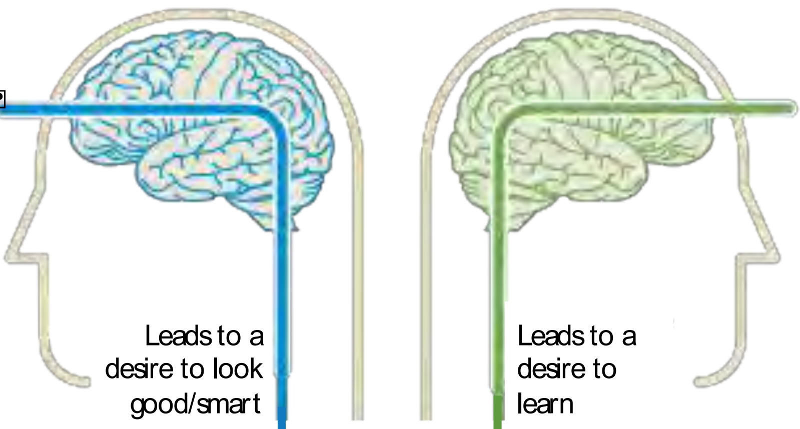
Ri: Now the student isn't learning from other people, but from his/her own practice. S/he creates her/his own approaches and adapts what has been learned to the particular circumstances.

So progression moves from obeying the rules (Shu – to Obey), consciously moving away from the rules (Ha – to Break), and finally unconsciously finding an individual path (Ri – to Separate).

A good practical illustration to use is the concept of driving. Think about when you are learning to drive a car. You follow the rules by the letter to pass the driving test. You have to think through each operation (moving from drive to reverse, turning on the lights, signaling, etc.) That was Shu. Contrast this with a Ri level, where driving has become second nature, and you are making changes to routes unconsciously based on the conditions you encounter on the road.

Any introductory class has many elements that are taught at a Shu level. HOWEVER, the reason it is necessary to spend time talking about the mindset and underlying principles behind Agile is to avoid getting stuck in the "Shu box." That is, understanding the theory and what an organization is trying to achieve makes it possible to get to Ri more quickly. That is, the organization gets to a point where it can invent new practices "on the fly."

More Information can be found at:
http://alistair.cockburn.us/Shu+Ha+Ri

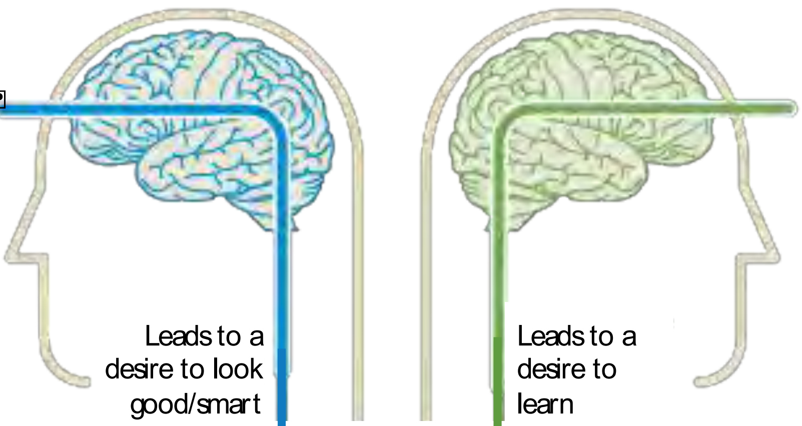

Fixed Mindset
Ability is inherent and static
Demonstrate Ability
Leads to a desire to look good/smart
Growth Mindset
Ability can grow
Develop Ability
Leads to a desire to learn

Fixed Mindset

- Looking good and avoiding challenges and obstacles because it is a risk for failure and will make them look bad.
- Stick to what they know and can do and as a result achieve less than their full potential.
- Feedback and criticism is personal as it impacts self-image.
- They don't change or improve much with time, if at all, and so to them this confirms that "they are as they are".

Growth Mindset

- Embracing challenges because they will learn something new or they will fail and that is an opportunity to learn also.
- Not afraid to put lots of effort to learn and master something new
- Feedback and criticism is not about them but about current capabilities
- View feedback as a source of new information that encourages them to keep learning and improving.



Fixed Mindset
Ability is inherent and static
Demonstrate Ability

Growth Mindset
Ability can grow
Develop Ability

Leads to a desire to look good/smart

Leads to a desire to learn

Aim to look good by delivering planned results

Assumes that customer knows exactly what they want therefore tries to fix the requirements (change is not desired)

Assumes that developers know exactly how to build it (and learning is punished)

Assumes and hopes that nothing will change along the way

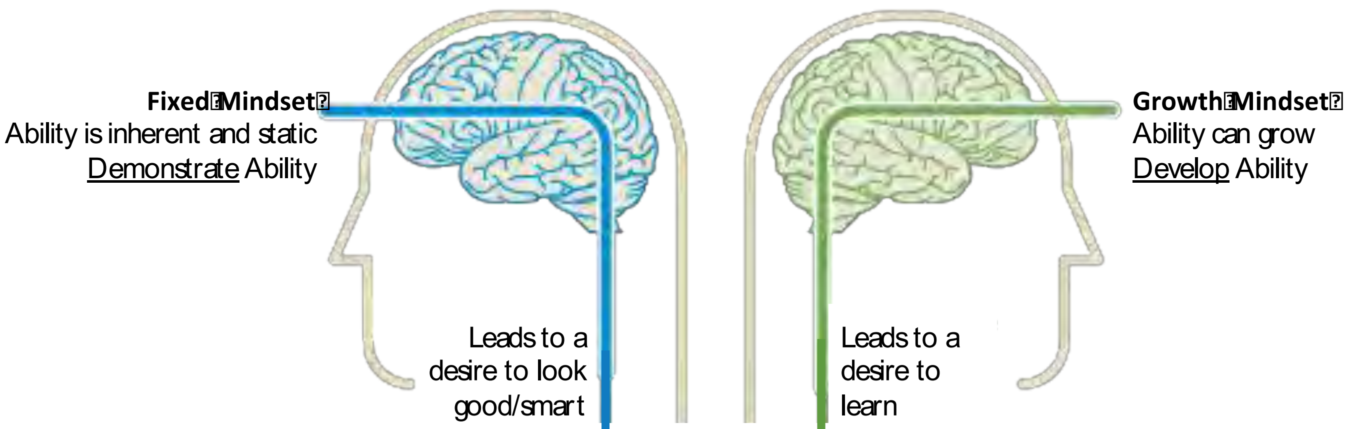
Aim to learn and deliver desired results

Allows the customer to learn and discover what they want when they see it and experience it. (Change is welcome)

The developers discovers how to build it when they build it (learning is rewarded)

Many things change along the way

FIXED VERSES GROWTH (AGILE) MINDSET MANAGING UNCERTAINTY


Fixed Mindset
Ability is inherent and static
Demonstrate Ability
Leads to a desire to look good/smart
Growth Mindset
Ability can grow
Develop Ability
Leads to a desire to learn

Aims to be successful by ***looking good and delivering <u>planned</u> results***

Wants to reduce uncertainty by asking the "customer" to **confirm** exactly what they want (change is not desired)

Wants to reduce uncertainty by **confirming** that "developers" know exactly how to build it (and learning is not encouraged)

Assumes and hopes that nothing will change along the way

Aims to be successful by ***learning and delivering <u>desired</u> results***

Wants to reduce uncertainty by allowing the "customer" to learn and discover what they want when they see it and experience it. (Change is welcome)

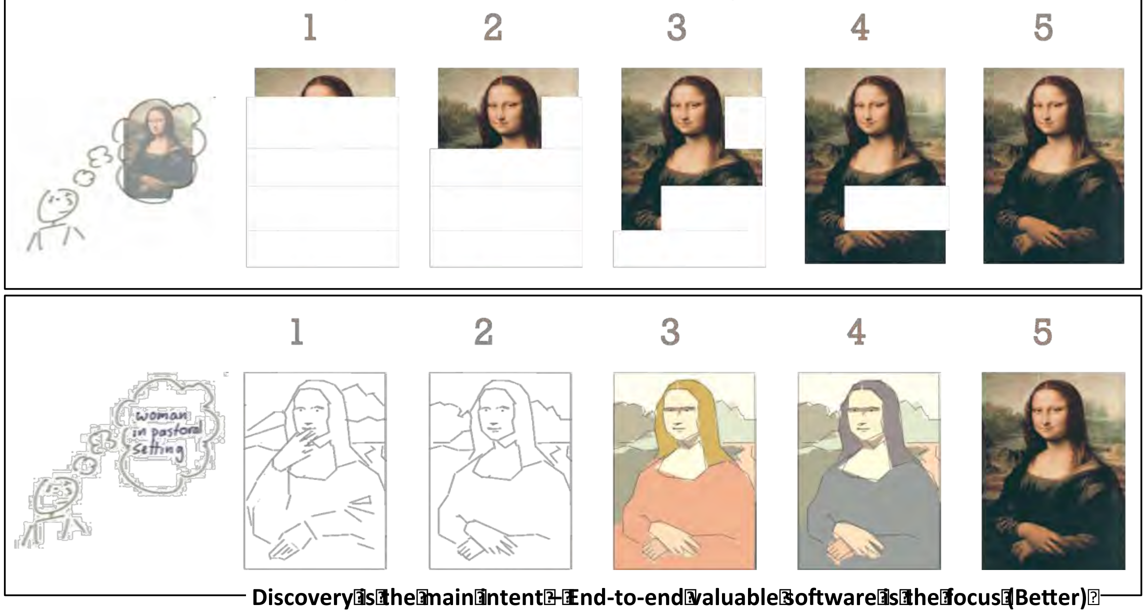
Wants to reduce uncertainty by allowing the "developers" to discover how to build it when they build it (learning is encouraged)

Assumes that many things change along the way

VALUE-BASED DISCOVERY: FOOD FOR THOUGHT

This picture shows how a practice like iterations can be used with a fixed mindset or an agile mindset and yields a different approach to development. With the fixed mindset (top row), the assumption is that the customer knows what they want exactly and the purpose of iterations is more to phase the delivery and get “some feedback”. Essentially the process of discovery is not the most efficient because they are not getting the maximum amount of feedback they can get from the process, they are getting modular or siloed feedback (piece by piece). The other picture (bottom row) shows how iterations can be used using an Agile mindset to truly discover what the customer wants. The aims of the 2 approaches are different even though both are using iterations. The bottom row provides higher efficiency during discovery since the customer is seeing an end-to-end slice of the system which would in-turn allow the customer to give more feedback.

Discovery is not the main intent – but finishing the work in increments is (Good)



Discovery is the main intent – End-to-end valuable software is the focus (Better)

The other value of the bottom row is that each delivery could be a valuable delivery worthy of going to market.

For each of the 2 rows, ask people around you the following question: In what column could you decide that you have a complete product? The answer for row 1 will be Column 5. The answers for row 2 will vary from 1-5. This shows that the value is in the eyes of the customer. Some just need an end-to-end slice, some need color, some need a very rich composition like column 5. The point is, this method of delivery gives the customer choice. If higher priorities crop up and #3 is good enough, you can still have a viable product and move on to another project. That is how you achieve flexibility and agility.

EMPIRICAL VERSUS DEFINED PROCESS CONTROL

There are two major approaches to controlling any process:

- The defined process control model.
- The empirical process control model.

The defined process control model requires that every piece of work be completely understood. Given a well-defined set of inputs, the same outputs are generated every time. A defined process can be started and allowed to run until completion, with the same results every time. **The defined process control model provides and exercises control through *planning, coordination and control*.**

The empirical process control model is suitable for processes that are *imperfectly defined* and generate *unpredictable and unrepeatable outputs.* An empirical process can not be rehearsed but will provide a great deal of learning, experience and discovery which may or may not be relevant to the next time the process is executed. **The empirical process control model provides and exercises control through *frequent inspection and adaptation*.**

For many years the majority of people involved in software development have based the control model on the defined process. But software development isn't a process that generates the same output every time given a certain input. It is a creative process. Agile software development is based on the empirical process control model of **inspect and adapt**.

The more defined, less creative and less dynamic processes are conducive to the "Coordination and control" style. Think of construction or manufacturing. This style fits and works well in those environments.

On the other hand, knowledge work, or inherently creative empirical processes work much better with an "Inspect and Adapt" management approach. Encourage learning and shorter cycle time to incorporate feedback quickly for continuous improvement and delivery of value.

KNOWLEDGE WORK

Knowledge workers are employees who have a deep background in education and experience and are considered people who "think for a living." They include doctors, lawyers, teachers, nurses, financial analysts and architects. As businesses increase their dependence on information technology, the number of fields in which knowledge workers must operate has expanded dramatically.

What differentiates knowledge work from other forms of work is its primary task of "non-routine" problem solving that requires a combination of convergent, divergent, and creative thinking. On average, knowledge workers spend 38% of their time searching for information.

Even though they sometimes are called "gold collars," because of their high salaries, as well as because of their relative independence in controlling the process of their own work, current research shows that they are also more prone to burnout.

Note: Content based on the Wikipedia entry most recently titled “Knowledge worker.”
http://en.wikipedia.org/wiki/Knowledge_work

THE SEVEN FORMS OF WASTE

Manufacturing	**Software**	**Sources and Comments**
Inventory	Partially Done Work	✦ Un-coded Requirements ✦ Unsynchronized Code ✦ Untested Code ✦ Undocumented Code ✦ Un-deployed Code
Overproduction	Extra Features	✦ More features lead to code bloat; the software becomes larger and more complicated. ✦ The extra features must be debugged, documented, and supported while potentially not adding value and not worth the Return on investment
Extra Processing	Relearning	✦ Poor knowledge capture ✦ Poor Discovery techniques ✦ No collaborating with the right people ✦ Poorly structured or written code
Transportation	Handoffs	✦ Each handoff leaves behind approximately 50% of the knowledge we intend to transfer
Waiting	Delays	✦ Delays prevent your customer from realizing value as soon as possible. Possible causes: ✦ Starting development work on a project long after the initial customer contact or requirements gathering activities ✦ Review or approval processes ✦ Gaps between the completion of development work and the beginning of QA/verification work;
Motion	Task Switching (Multitasking)	✦ Minimize the amount of context switches that you must do in a given time period to maximize productive flow.
Defects	Defects	✦ The amount of waste created by that defect increases with the amount of time that it lies undetected.

Source: Poppendieck, Mary and Tom. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley, 2006.

PRACTICE VERSES BEHAVIOR

Practice: The actual application or use of an idea, belief, or method as opposed to theories about such application or use.

Behavior: Behavior is the range of actions and mannerisms made by organisms, systems, or artificial entities in conjunction with their environment, which includes the other systems or organisms around as well as the physical environment. It is the response of the system or organism to various stimuli or inputs, whether internal or external, conscious or subconscious, overt or covert, and voluntary or involuntary.

5 DYSFUNCTIONS OF A TEAM

According to the book, The Five Dysfunctions of a Team by Patrick Lencioni, the five dysfunctions are[12]:

Absence of trust—unwilling to be vulnerable within the group

Fear of conflict—seeking artificial harmony over constructive passionate debate

Lack of commitment—feigning buy-in for group decisions creates ambiguity throughout the organization

Avoidance of accountability—ducking the responsibility to call peers on counterproductive behavior, which sets low standards

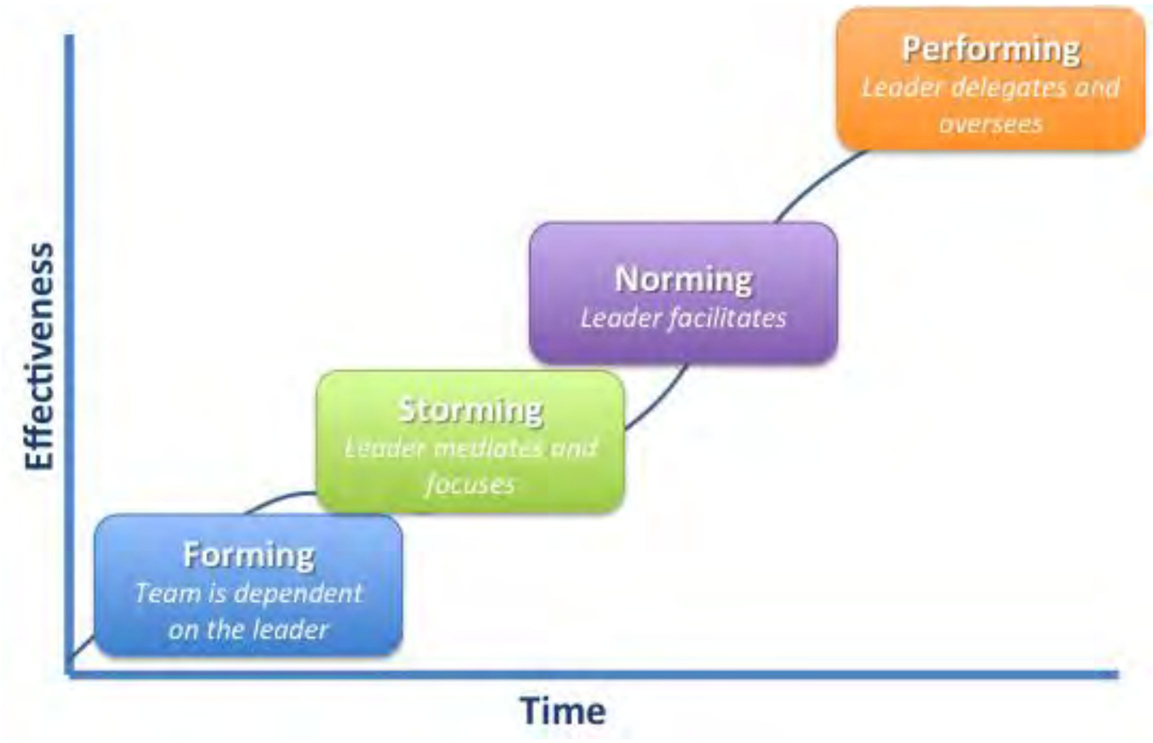
Inattention to results—focusing on personal success, status and ego before team success

Each is inter-related such that only a single dysfunction is present. It's important that to break through these dysfunctions, the role of the leader is to focus on having the team take ownership of these collectively. Moving from a team leader accountability model to a peer-to-peer accountability model within the team helps remove these.

[12] Source: http://en.wikipedia.org/wiki/The_Five_Dysfunctions_of_a_Team

TUCKMAN MODEL OF TEAM FORMATION[13]

The **Forming** – **Storming** – **Norming** – **Performing** model of group development was first proposed by Bruce Tuckman in 1965, who maintained that these phases are all necessary and inevitable in order for the team to grow, to face up to challenges, to tackle problems, to find solutions, to plan work, and to deliver results. This model has become the basis for subsequent models.


Performing
Leader delegates and oversees
Norming
Leader facilitates
Storming
Leader mediates and focuses
Forming
Team is dependent on the leader
Effectiveness
Time

FORMING

In the first stages of team building, the forming of the team takes place. The individual's behavior is driven by a desire to be accepted by the others, and avoid controversy or conflict. Serious issues and feelings are avoided, and people focus on being busy with routines, such as team organization, who does what, when to meet, etc. individuals are also gathering information and impressions - about each other, and about the scope of the task and how to approach it. This is a comfortable stage to be in, but the avoidance of conflict and threat means that not much actually gets done.

[13] Source: http://en.wikipedia.org/wiki/Tuckman's_stages_of_group_development

The forming stage of any team is important because, in this stage, the members of the team get to know one another, exchange some personal information, and make new friends. This is also a good opportunity to see how each member of the team works as an individual and how they respond to pressure.

STORMING

Every group will next enter the storming stage in which different ideas compete for consideration. The team addresses issues such as what problems they are really supposed to solve, how they will function independently and together and what leadership model they will accept. Team members open up to each other and confront each other's ideas and perspectives. In some cases storming can be resolved quickly. In others, the team never leaves this stage. The maturity of some team members usually determines whether the team will ever move out of this stage. Some team members will focus on minutiae to evade real issues.

The storming stage is necessary to the growth of the team. It can be contentious, unpleasant and even painful to members of the team who are averse to conflict. Tolerance of each team member and their differences should be emphasized. Without tolerance and patience the team will fail. This phase can become destructive to the team and will lower motivation if allowed to get out of control. Some teams will never develop past this stage.

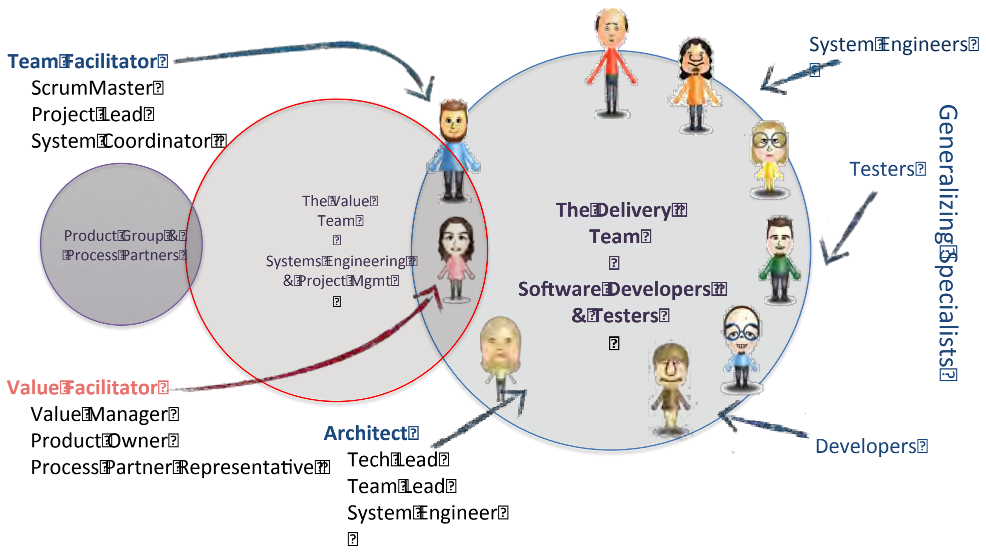
NORMING

The team manages to have one goal and come to a mutual plan for the team at this stage. Some may have to give up their own ideas and agree with others in order to make the team function. In this stage, all team members take the responsibility and have the ambition to work for the success of the team's goals.

PERFORMING

It is possible for some teams to reach the performing stage. These high-performing teams are able to function as a unit as they find ways to get the job done smoothly and effectively without inappropriate conflict or the need for external supervision. By this time, they are motivated and knowledgeable. The team members are now competent, autonomous and able to handle the decision-making process without supervision. Dissent is expected and allowed as long as it is channeled through means acceptable to the team.

Potential Designs for Agile Teams: Delivery Team



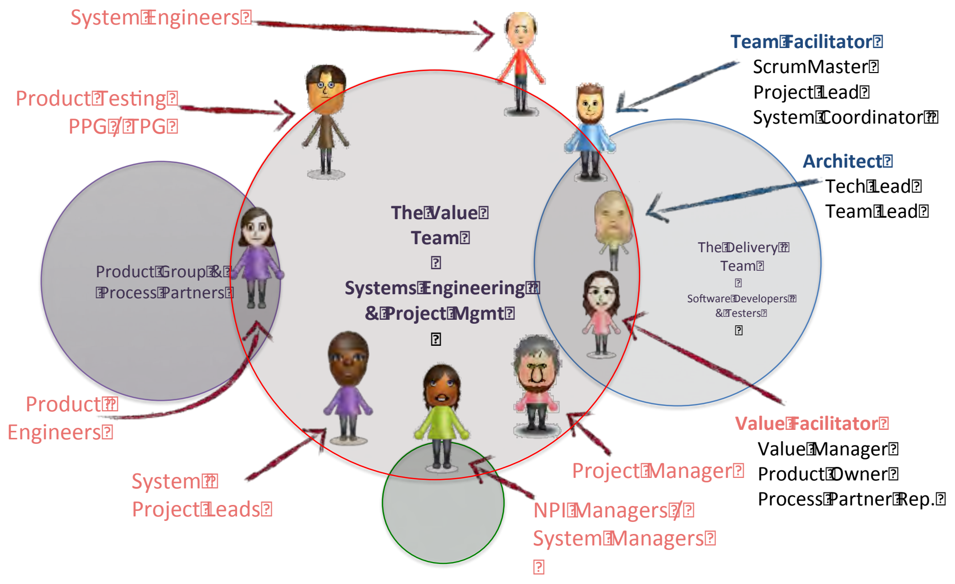
Team Facilitator (Project Leads):

- Holds Delivery Team to Process
- Supports the Delivery Team
- Clears Impediments
- Promotes communication
- Facilitates collaboration
- Builds trust
- Helps people improve – servant leader
- Ensures progress is radiating & plan is alive
- Ensures the Delivery Team is functional
- Facilitates Daily Stand-Ups
- Facilitates Iteration Planning
- Facilitates Iteration Review
- Facilitates Retrospectives
- Participates in Release Planning Meeting
- Facilitates & follows up on Improvements

Architect (Technical Leads):

- Establishes technical excellence
- Provides technical guidance
- Monitors technical debt
- Answers technical questions
- Ensures technical collaboration
- Facilitates Pairing
- Coordinates code reviews

Potential Designs for Agile Teams: Value Management Team



Value Team Facilitator:

- Primary responsibility for refining the Product Backlog
- Prepares for and facilitates Release Planning Meeting
- Prioritizes backlogs (e.g. content of next iteration)
- Can change features and priority every iteration
- Facilitates multiple opinions and relays one voice to the Delivery Team
- Responsible for the business value of the project
- Ensures the Value Management Team is functional and productive
- Facilitates the gathering of feedback during the Iteration Review

Project Manager:

- Provides visibility to Execs
- Establishes policies and strategies
- Limits Work in Progress (WIP)
- Establishes flow
- Manages expectations
- Updates the plan
- Forms the team
- Manages risk
- Manages vendors and contracts
- Manages project accounting (EVM, etc.)