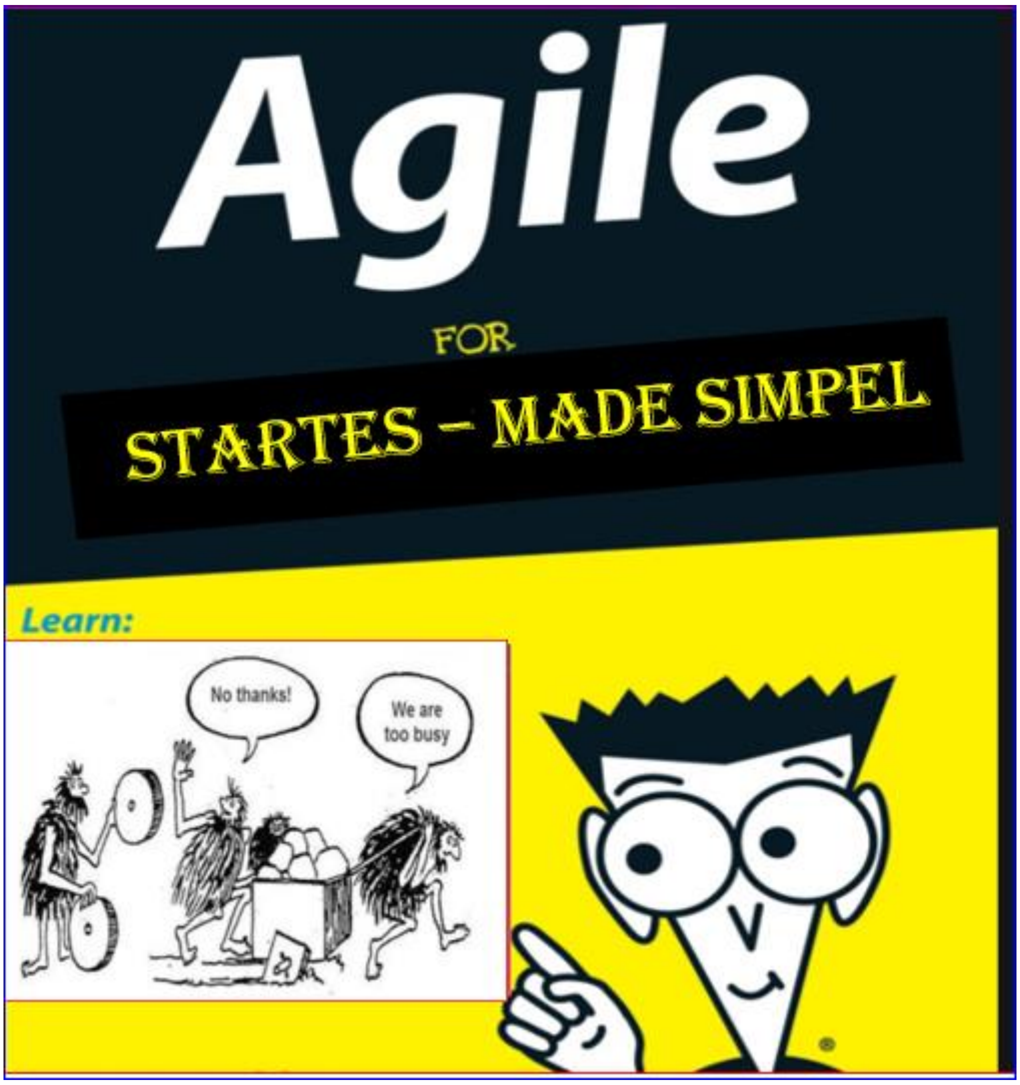
Agile Journey discussion series:
("Sharing is Learning"):

Agile introduction made simple

By Yesper Olsen, Update March 2021
(Request/Feedback: yesper@yescoaching.dk)



Introduction:

In the series: "*Agile Journey discussion series*" this small article is simply setting some very basic stepstone that you should know in your agile journey.
Simple in many senses. First of all because this is simply inspired by what Microsoft/Azure DevOps consider as the basic of understanding the context of this

tool, but also because that Agile basically is easy to understand but often difficult to do and being.

So what you get here is simply my extract for agile basic you have to know to pass the basic Azure DevOps education.

So if you will catch the souvenirs yourself you simply should follow this very simple education track of Azure DevOps:

Link: https://docs.microsoft.com/en-us/learn/browse/?products=azure-devops

Or simply be lazy and continue with my extract in this article ☺

If you continue, you will find inspiration around this:

- Recommendations for adopting Agile
- What is Agile Development?
- Recommendations for managing multiple Agile teams

And much more – have a nice reading journey….

Agile inspiration:

Recommendations for adopting Agile

The team is getting ready to take their first steps toward adopting Agile. Here are some general recommendations that any team can use to incorporate Agile into their organization.

Create an organizational structure that supports Agile practices

For most organizations, adopting Agile can be difficult. It requires a mind-shift and a culture-shift that challenges many existing policies and processes within the organization. Traditionally, most companies use a horizontal team structure. In practice, this means teams correspond to the software architecture. For example, there might be a team responsible for an application's user interface, another team responsible for data, and another team responsible for the service-oriented architecture.

However, vertical teams provide better results for Agile projects. Vertical teams span the architecture and are aligned with product outcomes. For example, there might be a team responsible for the email portion of the app and team members come from all three of the abovementioned disciplines. Another benefit of the vertical team structure is that scaling occurs by adding teams.

Mentor team members on Agile techniques and practices

When they first start to adopt Agile techniques and practices, some teams decide to hire external coaches. Coaches may even work with multiple teams to help remove organizational roadblocks and silos, so they often have both teaching and managerial skills. They can also train team members in Agile techniques such as how to run stand-up and review meetings. Over time though, it's important for team members to develop an ability to mentor each other. This means that most work should be done collaboratively and not by individuals who spend most of their time working alone.

Enable in-team and cross-team collaboration

If collaboration is the key to becoming successful at Agile, what are some of the ways you can encourage it? Here are some ideas.

Cultural change

When changing a culture, keep a few things in mind. It's important that team members have a quiet, comfortable place to work. They need spaces where they can focus, without a lot of distractions and noise.

Meetings are a fact of life and they can feel like they take over a person's working life. To give team members more control, meetings need an agenda and strict time frames.

Asynchronous communications, such as email and messages, can feel overwhelming and people often feel they have to be answered right away. Make it clear that not all of these communications need an immediate response.

Remote team members are now the norm in many companies. Everyone needs to feel comfortable with all their team members and to treat them equally, whether they're in the office or working offsite. Collaboration via communication should become part of the organization's DNA.

The importance of good communication can't be overemphasized, even when there are disagreements. Conflict resolution is a good skill for any Agile team to have.

Cross-functional teams

Just as it's important for team members to work collaboratively, it's also important for teams to collaborate with each other. Cross-functional teams add new skills and perspectives that can broaden everyone's ability to solve challenges creatively. Cross-functional teams also make the entire organization more cohesive. They reduce turf wars and increase the sense that everyone is working toward a common goal.

Tools for collaboration

Good tools can help your Agile team members collaborate more effectively, both within the team and with other teams. Here are a few suggestions to help you get started.

- Microsoft Teams . This is an application that provides a workplace for chat, meetings, notes, and file storage.
- Skype . Skype is easy to use and a good general-purpose tool. Many people have it already installed.
- Slack . Slack provides many separate communication channels, all from a single interface. These can be organized in many ways, such as by project, team, or topic. Conversations are retained and are searchable. It is very easy to add both internal and external team members. Slack directly integrates with many third-party tools, like GitHub for source code.

Other common tools include Google Hangouts, Asana, Trello, GoToMeeting and monday.com. Try to familiarize yourself with the options to see which of them suit the needs of your team and your company.

What is Agile Development?

By: Dan Hellem

Agile development is a term used to describe iterative software development. Iterative software development shortens the software development lifecycle. Agile development teams execute the entire software development lifecycle in smaller increments, usually called sprints. Sprints are typically 1-4 weeks long. Agile development is often contrasted with traditional or waterfall development, where larger projects are planned up front and executed against that plan.

Delivering production quality code every sprint requires the agile development team to account for the accelerated pace. All coding, testing, and quality verification must be done each and every sprint. Unless a team is properly set up, it can fail. And sometimes fail miserably.

This article lays out a few key success factors for agile development teams:

- Diligent backlog refinement
- Integrate early and often
- Minimize technical debt

Diligent backlog refinement

An agile development team works off of a backlog of requirements, often called "user stories". The backlog is prioritized so the most important user stories are at the top. The product owner owns the backlog and adds, changes, and reprioritizes user stories based on the customer's needs.

One of the biggest drags on an agile team's productivity is a poorly defined backlog. A team cannot be expected to consistently deliver high quality software each sprint unless they have clearly defined requirements.

The product owner’s job is to ensure that every sprint, the engineers have clearly defined user stories to work with. The user stories at the top of the backlog should always be ready for the team to execute on. This is called backlog refinement. Keeping a backlog ready for an agile development team takes an incredible amount of effort and discipline.

When refining the backlog, remember the following:

Refining user stories is often a long-lead activity

Elegant user interfaces, beautiful screen designs, and customer delighting solutions all take time and energy to create. Diligent product owners refine user stories 2-3 sprints in advance. They account for design iterations and customer reviews. They work to ensure every user story is something the agile team is proud to deliver to the customer.

A user story is not refined unless the team says it is

The team needs to review the user story and agree it’s ready to work on. If a team has not seen the user story until day 1 of a sprint, that’s a big red flag.

User stories further down the backlog can remain ambiguous

Don’t waste time refining lower priority items. Stay intently focused on the top of the backlog.

Integrate early and often

Continuous integration and continuous delivery (CICD) sets your team up for the fast pace of Agile Development. As soon as possible, automate your build, test and deployment pipeline. It should be one of the first things you complete when starting a new project.

With automation, your team will avoid slow, error prone, and time-intensive manual deployment processes. And let’s face it, if you are releasing every sprint, your team doesn’t have time to do this manually.

CICD also influences your software architecture. They ensure you deliver buildable and deployable software. When you implement a difficult-to-deploy feature, you become aware immediately as the build and deployments will fail. CICD forces you to fix deployment issues as they occur, ensuring your product is ever ready to ship.

CICD key activities are:

Unit testing

Unit tests are your first defence against human error. Unit tests should be considered part of coding and checked in with the code. Executing unit tests should be considered part of your build. Failed unit tests mean a failed build.

Build automation

Have your build system automatically pull code and tests directly from source control when builds execute.

Branch and build policies

Configure your branch and build policies to build automatically as your team checks code into a specific branch.

Deploy to an environment

Setup a release pipeline that automatically deploys your built project(s) to an environment that mimics production.

Minimize technical debt

With personal finances, it’s easier to stay out of debt than to dig out from under it. The same rule applies for technical debt. Technical debt includes anything the team must do to deploy production quality code and keep it running in production. Examples are bugs, performance issues, operational issues, accessibility, and others.

Keeping on top of technical debt requires courage. There are many pressures to delay fixing bugs. It feels good to work on features and ignore debt. Don’t be fooled. Sooner or later, somebody must pay the technical debt. Just like financial debt, technical debt becomes harder to pay off the longer it exists. A smart product owner works with their team to ensure there is time to pay off technical debt in every sprint. Balancing technical debt reduction with feature development is a difficult task. Read Creating productive, customer focused teams for ideas on how to manage this.

Always Be Agile

Being agile means learning from what you do and continually improving. Agile development provides many more learning cycles than traditional project planning. Each sprint provides something new for the team to learn.

For example:

- A team delivers value to the customer, gets feedback, and then modifies their backlog based on that feedback.
- They learn that their automated builds are missing key tests and include work in their next sprint to address it.
- They find that certain features perform poorly in production and make plans to improve performance.
- Someone on the team hears of a new practice and the team decides to try it out for a few sprints.

If you are starting with agile development, expect more learning opportunities. Way more. An Agile team doesn’t waste those opportunities.

Recommendations for managing multiple Agile teams

An Agile approach, along with Azure DevOps, can substantially improve project transparency and predictability. However, projects may still run into traditional challenges, often related to personnel or miscommunication. Here are a few things to consider as you scale your Agile efforts.

Build trust in your people and processes

Early detractors from Agile implementations are often skeptical about their ability to improve team performance. It's important for thought leaders within the organization to build trust by illustrating how the tools and processes produce results. Sometimes these results are improvements in productivity, and those are easy to quantify. However, don't forget to highlight the team wins that occur by circumventing potential problems, such as avoidable schedule slips or quality issues. As people begin to associate the benefits with the process that achieved them, you will get more enthusiasm.

Elevate the organization above the team (and individual)

Some teams and individuals get territorial when new processes or policies are proposed. Rather than framing new policies as negatively exposing the performance of specific teams or individuals, highlight how the new transparency across the organization informs everyone of expectations. Having a single place where anyone can trace how their work relates to the organization meeting its goals will drive home the importance of their commitment to the process.

Foster a culture of transparency

Unfortunately, the term "transparency" gets a bad wrap. Nobody asks for more transparency when everything is going great. Instead, transparency (or lack thereof) is often blamed when teams are struggling. Even with all of the opportunities for transparency afforded for Agile teams, it's still subject to the honesty of individuals and teams. Emphasize that one of the reasons for transparency is to be able to identify and address potential issues before it's too late. Everyone understands that people sometimes run into circumstances that prevent them from meeting schedule deadlines. But if they don't feel safe in reporting disappointing news until the last possible moment, it can have a much more destructive impact. Building a comfort level with transparency can start with thanking people for reporting expected delays as early as possible.