DATACOM

Agile & DevOps Handbook

Samantha Laing + Corneile Britz

The Datacom Agile & DevOps Handbook

Samantha Laing and Corneile Britz

This book is for sale at
http://leanpub.com/theagiledevopshandbook

This version was published on 2018-12-04

Tweet This Book!

Please help Samantha Laing and Corneile Britz by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I am busy reading The Agile & DevOps Handbook and it is great. (and free - so go grab yours!) #AgileDevopsHandbook

The suggested hashtag for this book is #AgileDevopsHandbook.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#AgileDevopsHandbook

Contents

Foreword

by Vernon Kay - Director, Datacom

Datacom has been involved in countless customer transformation projects, over more than 50 years in the technology industry. While we are known as a tech company, our customers are increasingly asking us to take an active role in transforming their people and their processes too. We believe that this change represents a much larger shift in the role technology is playing within organisations. Digital technology is transforming entire industries and reshaping operations across every industry vertical.

In order to adapt to these exciting new opportunities, organisations are having to change more than just their hardware and software systems. They are developing entirely new business models, new digital strategies and aligning their people, capabilities and processes to make it all happen. It is clear with the increasing pace of this change that organisational agility is the only sustainable competitive advantage moving forwards. Agile and DevOps are therefore fundamental enablers for organisations seeking to avoid disruption and gain sustainable competitive advantage.

Datacom and the tech industry are not immune to disruption either. We're constantly transforming our organisation, pivoting and adjusting our priorities to meet the ever-changing needs of our market. Part of this transformation journey has been one of moving from

an organisation that does Agile to one that is Agile. It's tempting for technology people to want to focus on the process and technology aspects of Agile, but our experience has shown us that in fact the greatest benefits are found in the people and culture areas. Despite its origins in the software development arena, Agile is also most successful when we apply it across the entire organisation, not just the technology teams.

DevOps is similar, in that it originated in software development but in fact seeks to extend agility across an organisations' entire value chain. Through DevOps, we empower teams to move from idea to final version faster, while improving sales, stability, performance and trust through numerous incremental improvements.

This book is a compilation of our own learnings, as well as our observations supporting others. It is intended to be a simple guide that each of us can refer to on our Agile and DevOps journeys, in addition to training services and hands-on work with coaches. We hope you enjoy it and take an active part in sharing your own learnings with to us and contributing to the wider Agile and DevOps community.

Kind regards,

Vernon Kay Director, Datacom

Introduction

Why this book

There is an increasing need for teams, projects and organisations to work in an agile way. This need arises in various ways, from clients dictating that work will be done in an agile way, all the way to teams requesting to work in an agile way. There has also been a large move for companies to understand and use DevOps principles and practices.

Unfortunately, this huge push for change has also led to many misconceptions, misinterpretations and generally bad practices done with the best of intentions.

We want to help you.

This book will highlight what you should pay attention to, when training is a good idea, and when getting a coach involved will help you get to that next level.

This book encompasses over 30 years of experiences from all over the world. It is by no means the only way of doing Agile or DevOps. It is also not the best way. It is simply our combined experiences of what seems to work, when we focus on certain elements.

NOTE If you are reading a printed out version of this book you will not be able to see the hyperlinks. We have listed them, per chapter at the end of the book in a chapter called *Links*.

As this is the internet, links do tend to die. If one of these links no longer works - please try googling the terms, as the author may have just moved the post.

Who is this book for?

This book is intended for those who are fairly new to the world of agile and its way of work. The methods explained will borrow ideas from various frameworks that you might be familiar with like: Scrum, Kanban, Lean and a few others.

Over many years of working with teams from large corporates all the way through to small start-ups we have noticed some patterns in how agile creeps in. Mostly someone is told that Project X must be run in an agile way, or that there is a requirement to deliver more, faster and cheaper. Very seldom is there a slow transition with training and change management.

Just to be clear: getting in an Agile Coach to help you with your agile project and transition is the best idea. You should do that.

This book will help you understand some concepts and give you an indication of things you can do to work in an agile way. Agile and DevOps is more of a mindset than a set of practices. I'll repeat that incase you weren't paying attention.

Agile and DevOps is more of a mindset than a set of practices.

What do we mean by that? Well, most people want to implement the practices, check the boxes and see the change. Sure. Any practice will give this to you, whether its agile or not. The mindset part means completely

relooking at how you understand work should be done. It's a mindshift. And that takes time, patience and commitment to get right. A good Agile Coach will help you adopt practices and over time guide you and your teams along the journey of this mindshift.

It is not a magically fairy land fueled by hugs and high fives. It has been proven to work in many companies all over the world in vastly different industries. The mindshift leads to higher productivity, motivation and general happiness. The practices only lead to small productivity increases. Which is better than nothing!

The easiest way to adopt the mindset is to be guided by someone who gets it - usually a great agile coach. The road where you do it your self is not impossible but will take much much longer and have some difficult lessons along the way.

What is agile?

In 2001 the Agile Manifesto[1] was created. It states:

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

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

[1] http://agilemanifesto.org/

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

Seems simple right? Well turns out following these 4 statements is rather tricky, and so frameworks to guide you have popped up, namely: Scrum, Kanban, XP etc. Aside from the numerous frameworks, many practices have been created to help you, like: User Stories, Behavior Driven Development, Test Driven Development etc. So where should you start? This book will help, keep reading.

What is DevOps?

From wikipedia[2]:

DevOps (a clipped compound of "development" and "operations") is a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops). The main characteristic of the DevOps movement is to strongly advocate automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management. DevOps aims at shorter development cycles, increased deployment frequency, and more dependable releases, in close alignment with business objectives.

Most people think that DevOps is: *Automating builds Automating tests Environment provisioning*

Thinking of DevOps in this manner is like owning a telescope and thinking you understand astronomy.

[2] https://en.m.wikipedia.org/wiki/DevOps

DevOps done well will also cause you to rethink how you work and what you are working on.

How to use this book

Start by reading everything - it should only take an hour or so. Don’t take notes or get distracted. Just read the book, cover to cover.

The book is set up as follows:

Summary

There are four main sections: Start, Manage, SetUp and Work. Then there are two chapters on practices and more tricky topics. Finally, there are the ways that the agile coaches and devops coaches at Datacom can help you. Where appropriate the chapter might contain links to articles on the internet. These are there for you to dive deeper into certain topics.

We are hoping to instill in you some basics concepts of agile and devops and the reasoning behind a different

way of doing things. This will be challenging. You will want to argue or think it’s the same as <insert previous thing here>. It is not.

Once you have the whole picture in mind, now you can reread the book. This time take notes. Make decisions. Setup sessions.

And, if possible get an agile coach and devops coach to help set you up for success.

Why are you reading this book?

Is it because you want to be agile or use devops? And why is that?

For some it's because they've been told they must. Try and find out why. What is the underlying reason? This is the very start of your agile and devops journey. Almost every change in agile and devops is done as a measured experiment. Understanding what you are hoping will be different, improved etc with this change will give you a starting point for a metric.

Perhaps you want better quality? Perhaps you want to deliver faster? Perhaps you want to increase morale on your team?

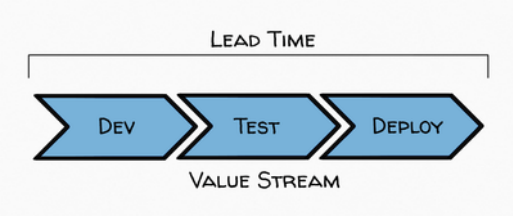
How will you know if this is happening? Try to come up with at least three different ways of noticing if this is (or isn't) happening. This can be tricky if you are not used to metrics. Pose your questions to others and see if their answer would work for you. E.g.: If you wanted better quality, how would you measure if it was happening?

For now write these down in your notebook - we will use this information later.

Process Planning

As mentioned, most people think of agile as being a set of practices or a methodology. Usually they decide up front if they should use Scrum or Kanban (these are the two most popular frameworks to help you become agile).

Scrum and Kanban are great. We have used these often over the last 10 or so years. Scrum has more rules than Kanban and this can provide nice structure if you are new to agile and not sure what to do. Kanban has it's roots in the manufacturing business. It is easier to implement, as there are few rules and no roles. Both of these can also be used in totally non-agile ways.



DevOps in Reality

In DevOps, we define a technology value stream as the process required to convert an idea into solutions that deliver real and measurable value to customers.

We need to be very aware that we need to improve the entire value stream, and not just fall into the trap of just looking at our own small, local patch whether it be local

process optimisation or focusing on a key technology. This directly results in reduced lead time and costs, with improved satisfaction and morale.

If you had the option of making it better for everyone, why would you only make it better for yourself?

If you have decided (or someone else has) that you will go the Scrum or Kanban route then at the very least get training for everyone involved on your project. This will give everyone a common language and base point to start from. You can do certified training - this usually proves to be expensive. The alternative is to get hold of an agile coach and ask them to do a custom 1 day course for you.

If you are not sure, but would like to know more about Scrum and Kanban we recommend these:

- Free 1 hour Scrum video[3] - just watch the videos to understand how it works in under an hour
- Kanban book[4] - very simple quick read.

In the following pages we will borrow ideas from Scrum, Kanban, Lean and others, that we have found useful no matter what framework you choose.

We will start with the management around this.

How do you manage differently? How will you know what's happening? When will work be done by? Is the project on track? What is holding you back? How are you improving? And what are your metrics telling you?

Then we will move onto setup.

[3] http://growingagile.thinkific.com/courses/scrumbasics
[4] https://leanpub.com/kanbanworkbook

What should the team look like? What are the skills and roles needed? What are the team agreements? How will this look time wise?

And finally we look at how the work gets done.

We focus on planning, getting feedback, and improving how we work. What techniques can be helpful here, and how do you do this?

Ready? Lets go!

What do you need to do?

You need to be able to clearly articulate what you want your team(s) to do. Not the technical detail of how, but rather the business explanation of what and why. This is at a fairly high level with a bit of detail in the beginning.

The best way I have found to do this is a technique called User Story Mapping. If you are familiar with the technique, great. If not, get help. An agile coach can run a workshop for you, to teach you and the team how to do this, and to help you create your initial Story Map.

A Story Map allows you to better visualise in two dimensions your list of to-do items or ideas or requirements. It gives you the ability to see how the system will flow when complete and what pieces each part of the system is made up of at a high level. It is done from a business point of view rather than a technical point of view, and as such is easily understood by all. If you would like an example of a Story Map, please see the chaper near the end of the book called Story Map.

It also allows you to prioritise these pieces and decide what is the absolute minimum you need for your first release. And, no, it is not everything. If you think it is - call that agile coach, they will help you explore the bare minimum in an easy way.

The idea is to create this with your teams to spread and share all the knowledge you have about what needs to

happen and by when. If you have any dates or deadlines - put them up and make them visible. You might not yet know what is possible by a certain date but everyone should be aware that a date exists.

What do you now have?

- A visual board of all work at a high level
- An indication of what needs to be there for the first release
- A priority decision for each part of the system
- Every member of the team is on the same page, understands the requirements and any due dates

Advanced Learning

- TED talk on The Power of Why[5] by Simon Sinek
- Book on User Story Mapping[6]

[5] https://www.ted.com/talks/simon_sinek_how_great_leaders_inspire_action

[6] https://www.amazon.com/User-Story-Mapping-Discover-Product/dp/1491904909

Visualise

We want everyone on the project to be on the same page and to know what is happening at any point of time. A key part of this is making everything visible.

Part one was creating the Story Map. This needs to be placed on a wall where everyone on the project can see it and interact with it. We call this an information or visual radiator. Anyone who walks past will be able to see what is happening, the status and a bunch of other information. Visual radiators are very popular in agile teams and this book will share quite a few with you.

As you complete something on the Story Map you place a large tick through it. If something gets removed then it gets a big cross through it and moves to the bottom of the column as an indicator of priority. Don't take it off the board. Leaving it on the board conveys that a decision has been made to not do it. As things change, update this board. These boards tend to grow quite large - so I would allow a large wall space for this - think 6 metres wide by 2 metres high.

The Story Map helps with everyone seeing the work the team needs to do. All updates should be done as a team, so that everyones knows what has happened. More about this later.

Often there is a bunch of other stuff that needs to happen for the project to be done, that involves other people or companies. Things like, getting a contract with the client signed, passing testing levels, invoicing

etc. The technique we recommend for visualising this is Portfolio Kanban.

If you have never created a portfolio Kanban board before, call up an agile coach and have them run a workshop with you. The magic of these boards is that they are unique to you and your project and that they change over time. That also makes it tricky to learn from a book only.

Portfolio Kanban gives you a very high level view of what is happening and when it needs to happen. The focus should be on work flowing through your system. Here you will notice any dependancies, minimise bottlenecks, and improve feedback loops.

- A look at different board designs[7].
- The story of one projects porfolio kanban board[8].

What do you now have?

- Another visual board indicating progress of all parts of the project
- An indication of any dependencies the project might have on external providers
- A quick way to notice bottlenecks
- A way to visualise feedback loops
- Everyone on the team being able to read this information radar and provide suggestions to improve

[7] http://brodzinski.com/2015/04/portfolio-kanban-board.html
[8] http://brodzinski.com/2011/11/project-portfolio-kanban.html

Advanced Learning

The Bottleneck Rules[9] book by Clarke Ching

[9] https://www.amazon.com/Bottleneck-Rules-More-Working-Harder-ebook/dp/B07DCFR7B4

Improvements

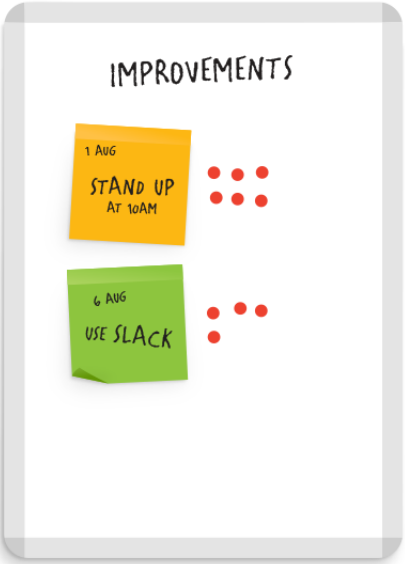
Another visual radiator will be the Improvement board. You might be wondering what this has to do with delivering a project. Not much, and yet everything!

One of the most important parts of agile is Inspecting and Adapting (more on this later) by making small improvements every couple of weeks.

This board is where we track these improvements and see if they are still helping us and if we should continue doing them. It's also a great way to see how far a team has come in growing and learning together.

I like making this a living document. This means it's not static, it is ever changing. The improvements get written on a card with the date they are created. On the back of the card you place a list of things you expect to change if the improvement works. eg: Our first improvement is to change our Daily Standup to 10am. We expect this will mean: everyone is at the office and can attend, less people will be late due to commuting issues, and it will happen on time and thus everyone will be in sync.

Everytime the improvement has a positive impact on someone or something, have them place a dot next to it. This allows everyone to see if the improvement is working and the impact it is having on the team.



Improvement Board

What do you now have?

- A way to track improvements made by your team
- Visual indicator of which improvements are working
- A way to check that the change you thought the improvement will bring, is actually happening.

Advanced Learning

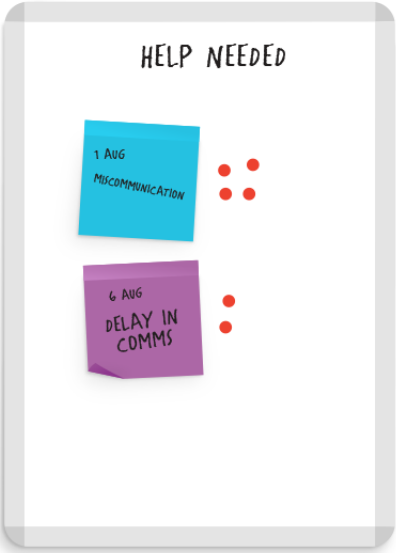
Kaizen[10] and how it is used.

[10] https://www.slideshare.net/mobile/mgraban/mark-graban-intro-to-lean-frisco/42-Kaizen_Card_Idea_Card_Standardized

Help Needed

This is another visual information radiator and very similar to the Improvements Board. This board is for anything that the team thinks will help them work faster. Each card has a problem and the date it was created. Each time the team feels the pain of that problem they put a dot next to it.

These should only be things the team feels they can’t resolve. Perhaps as in the example below, the team feels that sitting together would solve these issues but they are not allowed to move. Everytime these problems occur - they mark it with a dot.



Help Needed

Once a month as a large group you would look at the problem causing the most pain (the one with the most dots) and use a technique to explore various solutions.

Some recommended techniques to explore solutions are Impact Mapping[11] or using a Problem Solving Tree. These techniques will help you come up with many different ideas to try, and some of these might be easy. Those that are not within the teams control, might need a manager to decide if the cost is worth the benefit. Talk about this openly. If sitting together is not an option and will never happen - let the team know so that they can think of other potential solutions.

[11] https://www.impactmapping.org/

I was once coaching a team who mid-workshop mentioned that it would be nice if they sat together. I asked what that meant as I thought they did sit together. They explained that whilst they were in the same area, people used the space between them as a shortcut walkway to the kitchen and so they didn't often turn around to talk to each other. I asked them what they would prefer and they said ideally they would like a table between them so that they could turn around and talk around the table, and that no-one would walk through. They then said that space was a problem in the office so it would never happen. I mentioned this to their boss, who surprised me by saying he was planning a reshuffle of office seats in 5 weeks time and so would do what he could to give the team their ideal solution. He was totally unaware that the team had this problem. Often just making people aware of things, somehow gets them resolved without too much effort.

What do you now have?

- A way to see anything that is slowing down your team
- Visual indicator of how often these things are hurting your team
- A way to identify which problem to deal with first
- Techniques to use to identify many different ways of solving the problem

Management

In agile teams how management is done changes slightly. Instead of managing the work and people you instead focus on managing the environment.

We rely on the visual information radiators (like the Story Map, Portfolio Kanban, Improvement Board and Help Needed Board) to manage the work, and we rely on the people to manage themselves. Regular check-ins with the whole team help the teams to understand their new responsibilities and also help managers see things are under control and they don't need to worry about the work.

Managing the environment means doing everything you can around the teams and the work to ensure things go as fast as possible. Here is a list of things you might focus on, but there are many others.

- Help Needed Board: Check this regularly and see if there is anything you can just solve for the team to make life easier.
- One-on-one meetings: Schedule one of these for each day, and rotate through everyone on your team. This is your time to get to know each and every individual on your team a bit better. It doesn't need to be a formal meeting in a meeting room, you could go out for coffee or go for a walk around the office - be creative. The focus here should be on that person. Think to yourself "What can I do to help make this person even more awesome?".

Listen to them, understand their needs, their problems, their lives. If you have never done this before it will feel awkward at first, keep trying. Building a relationship and trust with each individual helps in so many ways! You will also be able to connect people across your team, for example Anne might be into mountain biking and you know that Bob enjoys this too, so you introduce them to each other. A technique to help with this is Personal Maps.

- Training: Notice what is happening on your team and ask if there are needs for training. Often people feel guilty about the time and cost associated with training. Find out what training or conferences are happening and suggest to team members that they should attend. Create a culture of learning, and you will have a team that just grows stronger and stronger and thus works better and faster.
- Manage the boards: This doesn't mean owning the board, rather, ensure everyone understands the boards and you should talk to the boards as much as possible. Encourage changes to the board to help the team own it. Ensure there is no confusion and if you notice any clear it up immediately. Have as many conversations at the boards as possible - pointing to them and talking to them. This will help the team and individuals mimic your behaviour and start using the information radiators and keep them updated.
- Think about a manager in your past that was amazing, that understood you and helped you become great. What did they do to earn your trust? How can you do that for each person on your team?

It is very difficult to let go of the control to allow a team

to become self-organizing. If you find yourself wanting status reports and chasing people, then get some help from an agile coach.

What do you now have?

- A way to build trust and relationships with your team
- A way to show your team how you are helping them
- A training plan for your team
- Feedback loops between you and each person on your team

Advanced Learning

This is an interesting article on peoples behaviours and how environment affects this[12]. When you're seeking to predict or explain a person's actions, looking at the social norms, and the person's context, is usually a pretty safe bet. Situational constraints typically predict behavior[13] far better than personality,[14] intelligence, or other individual-level traits.

[12] https://medium.com/@dr_eprice/laziness-does-not-exist-3af27e312d01

[13] http://psycnet.apa.org/record/1979-28632-001

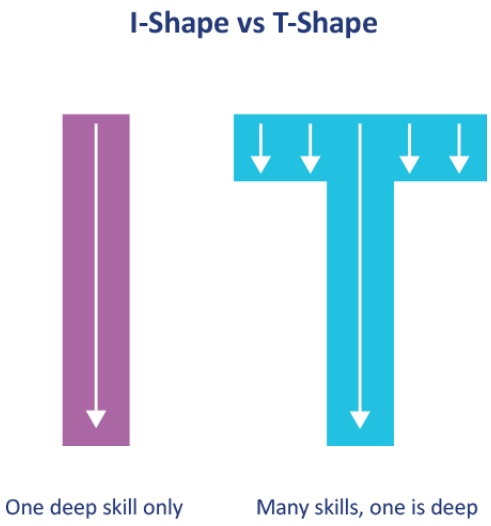
[14] https://pdfs.semanticscholar.org/a85d/432f44e43d61753bb8a121c246127b562a39.pdf

Team Setup

The ideal team will vary for each situation. We can’t predict how many developers or business analysts you need. If you already have a team - then they will be perfect! It’s easier to teach existing team members new technical skills rather than teach strangers to work together as a team.

What we can say, is that you need all the skills to deliver your project. Try not to think of roles or people, but rather skillsets.

In agile we think of people as being I-shaped or T-shaped. This means they have more than just one skill.



I-shape vs T-shape

Here is some reading on this topic: - Are you an I or a T?[15] - On I shaped and T shaped skills[16]

Why do we create teams? Well, people working together with a common purpose will share skills and collaborate to achieve that purpose. Usually each team will have all the skills to deliver your project. This is different to traditional thinking of teams being groups of people with the same skillset, eg: a team of testers or team of analysts.

With agile and devops we want multi-disciplinary teams, able to deliver your project. So a team might include analysts, developers, testers etc. Any skills that are missing or scarce will be the first to go onto your Help

[15] https://www.forbes.com/sites/andyboynton/2011/10/18/are-you-an-i-or-a-t/2/#4db1cb343cf7

[16] http://blog.ipspace.net/2015/05/on-i-shaped-and-t-shaped-skills.html

Needed Board.

Some suggestions are: pairing[17], mobbing[18] and of course training, though pairing and mobbing spread knowledge and skill transfer much quicker. An agile coach or technical coach can help you experiment with various learning styles.

It helps to educate the people on your project on what teams are and encourage them to pick their own teams to work within. Every 3 months or so you can redo this exercise so people can try working with others or on different areas of the project. Another alternative is Teaming[19], though we would say this is more for teams who have a dedicated agile coach to assist them.

If you are following a particular framework like Scrum, then the team size might be recommended. In Scrum you will have a Scrum Master and a Development Team of 5 or 6 people and a Product Owner. In Kanban there isn't a team size as people usually work by themselves or form small teams for particular items (like with teaming).

One thing to note, is that the more time a group of people spend working together the better they get at working together. Most of their communication becomes non-verbal simply because they have learned to work together well. Think of your partner or spouse, you can tell alot just by one look, no talking needed! For this reason, you should try and keep teams together as much as possible.

[17] https://medium.freecodecamp.org/the-benefits-and-pitfalls-of-pair-programming-in-the-workplace-e68c3ed3c81f

[18] https://underthehood.meltwater.com/blog/2016/06/01/mob-programming/

[19] https://www.growingagile.co.za/2018/05/teaming-could-this-be-the-future-of-agile/

This goes very much against the traditional thinking of resources and utilization of people. Instead create awesome teams of people that work well together and they will move mountains. This is referred to as flowing work to people as opposed to people to work.

- Small team, multiple projects[20]
- Multiple projects with a small team[21]

[20] https://davidmarquis.wordpress.com/2011/12/03/small-team-multiple-projects-agile-planning/

[21] https://medium.com/@anttihavanko/how-to-handle-multiple-projects-with-a-small-team-ebdfde5f6714

Team Agreements

Team Agreements are just that. Agreements between team members that they hold each other accountable to. These are decided up front and then need to be tweaked and changed as the team learns and grows. They are NOT created by other teams or other managers or other people. They are created by the team members and owned by the team members.

Team Agreements help people feel safe. They create a boundary that gives them freedom to move within, with very clear parameters of when they should ask for permission. It also prevents underlying assumptions and grievances from boiling over as time goes on.

I once worked with a team, where everyone was upset with one guy, lets call him Bob. Bob often worked from home. The team felt Bob was lazy and didn't actually do much at home. Digging deeper, we realized this was a perception because the team couldn't see when Bob was at his desk so to speak. We sat together as a team and came up with some agreements for anyone who wanted to work from home. For them this included being online and available on slack, and if you were away from your desk, to put into slack when you would be back. eg: Going for lunch, back at 1pm. They also decided that on a Wednesday everyone had to be in the office, and that was when they would have planning meetings as they preferred that face-to-face. After these agreements, the team was much happier, and a few more people decided to try out working from home as the understood the

rules around it now.

Here are a set of cards to help you[22] get started with Team Agreements. Talk through each point, ensure everyone is on the same page with everything. Don't assume that people know what the company policy is, instead make it explicit. Have the team write up the agreements and make them big and visible. Regularly use this to help become more accountable and not as a stick to punish. For example if the agreements is to have a stand-up at 9am everyday and Jenny misses a bunch of these, the first thing to do is to find out why Jenny is missing them and if moving to to a later time-slot will help. If Jenny is missing them because she hates them, this is an opportunity to improve. Why is Jenny getting no value from these? Are other people getting value? What is the purpose of a standup? And does the team understand the purpose?

[22] https://www.growingagile.co.za/?s=team+agreements

Meetings

So what will a typical week or month look like for your team?

If you are using a framework like Scrum, you will have set periods for your sprints as well as defined meetings and times for those meetings. If you are using Kanban there are no meetings prescribed.

Regardless of the framework you have chosen - this is what we would recommend.

An assumption I'm going to make is to define "success" as: you want to complete a project, and want it to be useful to your customer or you want to complete a product that is loved by end users.

Review

Key to "success" (as defined above) is getting regular feedback, and as early as possible so that you know if you are on the right track or not and have time to change tracks.

Scrum does this by using Sprints. The most popular sprint length is two weeks. So every two weeks you are delivering something on the project that is ready to get feedback on. At the end of the sprint you have a review to show what you've done, and get feedback on it.

If you're not doing Scrum, it is useful to have a regular cadence to check what has been done and update all

your visual boards appropriately. We would suggest weekly at first.

In a review, you talk about what has been completed, anything you learned that might be of use down the line to others, any feedback on what was completed and the project status. Some people invite others to this meeting to get feedback, we prefer having a regular Feedback slot for that purpose, more on that later. The project status will be to see how the team is progressing with the project demands and timelines and if everything is still on track. If not, everyone should talk about how they can get the project back on track and action that immediately.

This review should take no longer than an hour, and must be valuable to the whole team. It is a way for them to see their contribution towards the overall goal and also to see how the project is going and if they can help out in other areas.

If you find yourself thinking nothing can be de-scoped, and the only solution is throwing more bodies at the problem or copious amounts of overtime, please stop. Call your agile coach and explain what is happening. We can promise you that overtime will only make your problem worse down the line (20 years of experience with many teams tells me this). We can also promise that your agile coach will be able to help in a more sustainable way that won't affect your quality. Win-win.

Planning

This is much more difficult than it sounds. Unlike traditional development this requires looking at your story map, picking the highest priority item and then breaking

it down into small pieces that are still testable and useful to the customer so that you can get feedback. Think of items that take 2 to 3 days to complete. Most teams struggle to break items down in a way that still makes sense to the end user. Instead their instinct is to break things into technical pieces the end user has no idea even exist. The good news is that this is like learning to ride a bike. Once you've done it a few times, you get used to it, and find it easy. If your team is struggling, call in that agile coach and have them run a few planning workshops with your team until they feel comfortable doing this by themselves.

In addition to the value that will be delivered, consider and plan the tasks needed to automate quality as part of the pipeline, including security and compliance. If this sounds strange or difficult, call in the devops coach and let them help you.

This meeting is crucial in conveying information, concerns, dependancies and choices. Analysts should be explaining concepts and taking notes. Only document what is necessary and document it in the easiest way possible for the team to consume. In 99% of cases that means photos of the conversation with notes as a reminder.

Mindmaps are my favourite technique for doing this.

If you are planning something that is large and won't necessarily be worked on for a few weeks then only talk about the customer requirement and need, and how you will test that you have met that need.

If you are planning something that you are going to be working on in the next week, then you need to go into alot of detail. You should have spoken about this item before, and thus understand the customer

requirement, need and testing assumptions. If not, have that conversation first. Then get into the nitty gritty detail of what all the work is. Think tasks that take no longer than an hour. What are the test cases? What are the database tables? What are the User Guide Documentation requirements? Where will we automate this: in the code with unit tests or at the api level? Every bit of work that needs to be done must be thought of and discussed. Where will this be deployed, who is the best person outside of the team to do User Acceptance Testing?

In Scrum, planning is done at the start of the Sprint, so every 2 weeks, for around 4 hours. If you are not doing scrum, you could schedule this for a particular day eg: every second Tuesday from 9am to 12am. Or you could have this meeting on a pull basis. eg: Once a team has finished what they were working on they have a planning session for the next piece of work.

Daily Stand-Up

Most people are familiar with this meeting, though mostly we see a status report meeting - which is not the purpose of this meeting.

A daily standup happens everyday for no more than 15 minutes, and to help you stick to this, it is a stand up meeting (as opposed to a sit down). It is not for the manager or the scrum master. It is for the team.

The team should ideally have their own visual board that goes down to a deeper level of detail - those one hour tasks created during planning. The daily stand-up is where they will gather to see what the board is telling them about their progress. Are they going to meet their

commitment? If not, what can they do as a team to get back on track? Are there any bottlenecks on their team? If so, who has the skills to help alleviate that bottleneck, or are there other ideas? If someone has nothing to do, what is next on the priority list, or what needs attention to help the team go faster?

You will notice we have left off the usual 3 questions: What did you do yesterday, what are you doing today and any impediments? Those questions tend to encourage individual work, and status reporting. They imply it's better to be busy. Instead we want the questions to drive attention to finishing work. What can we do to help get this piece of work to done. Regardless of who is suppose to do what.

Here are a list of impediments words. Everytime you hear one, there is something slowing down your team. Note them down, add them to your Help Needed board. This is something a Scrum Master usually does. we will talk more about the value and purpose of a Scrum Master in the Trick part of this handbook.

Impediment Words

Retrospective

The last meeting we would recommend is also the most important. The Retrospective. I would go so far as to say if you only do this meeting you are being agile. In Scrum this is held at the end of the Sprint, after the review, so every 2 weeks. We would recommend a regular cadence for this, two weeks is good, one month is too long.

This meeting is for the people on your project to improve how they work together. It is about their process. It should result in one improvement action for your im-

provement board. Don't fall into the trap of having many improvements each time. Rather pick one that affects most of the team and have them decide how they can all contribute to this. Talk about it in great detail and add it to the team board as the most important piece of work for the team to do.

Agile is all about Inspect and Adapt, and this meeting is designed for that. You should vary the facilitation techniques used to run this meeting to stimulate new ideas or thoughts. This is a site[23] that helps with these facilitation techniques.

Final Thoughts

If your team works really well together and is constantly collaborating, they might not need any meetings. This is very unusual on larger teams, but quite normal for smaller teams of 2-3 people. Similarly, your team might come up with improvements that involve creating additional meetings - that is great. Let them own their decisions. Just make sure to regularly check that the meetings are still valuable, otherwise question their value.

We have teams decide on what meetings they need. And then list what they want to get out of each meeting, who should be there and what they will talk about. This forces them to own the meetings and keep them relevant. Teams should also schedule their own meetings. This is difficult in some offices where meeting rooms are like hens teeth. Your team shouldn't have to beg and plead to get together to talk about work. Set aside a permanent table and chairs for adhoc meetings in your

[23] https://retromat.org/en/

team area. No booking needed. Collaboration when ever it is required.

Invest in everyone learning facilitation techniques. We spend many valuable hours in meetings and so you should aim for these to go as well as possible. Your agile coach should be able to run a facilitation workshop to help all meetings run well.

Work

How does the work get done in agile?

Well, it gets done all the time. In Scrum, teams do all their work in sprints (usually a 2 week time-box) and there is no break between sprints. In Kanban teams work off a list of work and never stop, instead they just pull in the next piece of work in this list.

The big difference between traditional work and agile work is **how** work gets done. Traditionally, everything gets analysed and planned and documented. Then it gets passed over to a team of developers that do all the development work. Then it gets passed over to the testers to do the testing work. and finally it gets passed over to an operations team that will do deployments.

Even if you're not a development team, traditionally you will have a step by step process that involves passing work from one group of people to the next. This comes all the way from the 1940's and how people worked in factories.

In agile we talk about the work but we only talk about the next 2 weeks work in detail. This is referred to as JIT (Just in Time). This allows us to focus on the most important stuff, and it allows the less important stuff to change over time without impacting us too much as we haven't spent much time on it.

As mentioned before, the bulk of this is done in planning and is very difficult to do. So if your team is struggling, get help from your local agile coach.

I have heard some horror stories like the customer won't pay for testers so we don't test, and we are developers so we are not capable of testing our own work. Quality belongs to the entire team and is part of the delivery, not a option that can be cut. They should own this and define this. Everyone has a different definition of quality so it is important that this is discussed and agreed as a team. Write it up and make it visible. This is the beginning of your Definition of Done.

For some this means doing unit testing for all work. For other this means using a source control repository. For some it includes deploying work to a certain environment. If you're not in IT, it might mean sending a document to a particular person.

For a team focusing on DevOps, the Definition of Done is typically defined as the following:

- Automated quality implemented
- Telemetry visible
- Checks passed
- Pipeline is green
- Feature is available and working

If your quality drops, then look at your Definition of Done and improve it. For the sake of consistency, having a organisational Definition of Done for the organisation adds a great deal of value to guide teams.

I once worked with a team, lets call them Team Bingo. Team Bingo was working well together and everything was going smoothly. Suddenly every time they had a review they were noticing tiny bugs creeping in. Silly things, like mis-spelling of words and wrong colours on buttons. These were little, but also annoying to fix and

horrible for the customer to point out. The team decided to change their board and put in an additional column called “Show Me”, before the done column. This meant every small task has to go to “Show Me” and be shown to someone else on the team, before it could move to done. This also meant, most small issues (and some big ones!) were found early, and easy to fix in the moment.

The Show-Me Column[24]

Team Bingo looked at their process and ran an experiment with the show-me column. Your teams should be running experiments to improve how they work continuously.

[24] https://www.growingagile.co.za/2018/05/showme-column-agileweeklytip/

Planning

Planning is all about understanding the requirement.

Here are some of the questions you might ask during a planning session:

- What needs to be done and why?
- Is there a better way?
- How will we know it is done?
- How can we test this?
- Are we making any assumptions?
- Can we test those?
- Are there any risks?
- How can we check those risks earlier and get feedback sooner?
- Do we have dependencies?
- Is there a way to do this without dependencies?
- Who are the ideal people to give us feedback?
- How am I planning to deploy this feature?
- What do I need to do to implement tests?
- How will we measure that this is working?
- What telemetry is helpful?

The aim is to figure out the answers before you start any work. So try and have a planning session a few weeks before you are going to work on something. This gives everyone enough time to investigate the answers to

questions. For example, you might not understand part of the requirement, an analyst can visit your customer and get clarity around this. Or perhaps you think there is something already in the code that might help, you now have enough time to quickly check that and feed back to everyone what you find. It also gives you time to think about the problem and other potential solutions, before you start working on it.

In the planning session just before you work on something, make sure all questions have been answered or that assumptions have been noted.

My favorite technique for planning is mindmaps.

I recommend doing this with hand-drawn maps as people have visual and spacial memory so using colors and where things are placed on a page matters. Most (not all) online tools re-order mindmaps, and you don’t want that.

Feedback

Quick feedback loops can save your project. It can also save you a lot of time and stress. Agile and Devops like to bake this into the process.

In agile we have feedback loops with people fairly often. There is the Review meeting and we would highly suggest a User Experience meeting. Schedule users to come in on a regular basis. Maybe 3 users every Wednesday from 9-10am. These do not have to be real users of your system. In fact almost anyone in the hallway will provide useful feedback. The day before, decide what is the best use of this hour. Perhaps it's going through a paper design of something with them, perhaps it's user interviews, or maybe you put your latest done work in front of them and see how they use it. Is it intuitive, what are their comments and frustrations?

In my experience, if you don't set this up from the start, you just leave it to the end - when it is too late to really act effectively on any feedback.

In DevOps we like to think of how the system can give us feedback as early as possible. Do we need alarms or do we have a dashboard to see what is happening in our system? If we are creating a website and the number of users falls to 0, should we be notified? How can we tell if the system is healthy? What other monitoring is needed?

When these things are planned for and built into a

system from the start then our projects start to work for us.

Think about the environments you need and set them up immediately - with the first piece of work you do. Yes, it will take a bit longer, and it should make all work going forward that much easier. Automated testing is another type of feedback and there are many ways of doing automated testing. Is your team trained in this? Most teams aren’t and so when they start they plan really poorly, duplicate test effort and create a maintenance nightmare. Again, if your team is new to this, get your friendly agile and devops coach to either run a workshop on agile testing and automation or ask them to recommend someone to help.

Deploying and integration should happen easily and regularly as these are also forms of feedback on your work. If your team is not doing this from the start, get help from your agile coach.

As you work more with feedback loops you will notice how they can be helpful and also when the feedback loop is missing, or when it is happening too late. That is great! Now you can put one in place and test if it gives you the feedback you need in the time that makes sense.

Inspect and Adapt

The Retrospective is the main place for “Inspect and Adapt” to happen for your teams process, and should be happening every 2 weeks. If you have many teams you might want to run an Across Team Retrospective for all of them to focus on how they can learn from each other and work together better. We usually recommend an Across Team Retrospective every 2-3 months.

Improvements are fascinating. We tend to think just because we’ve called it an improvement it will be one. Sometimes it isn’t though. So when you are thinking of things you want to try, think of it as an experiment with a hypothesis. What are you changing? What are you hoping will happen as a result of this? What can you notice along the way if it is working? What will you notice along the way if it is not working? (These are your built in feedback loops). On what date will you check in and reflect if you should keep this improvement, bin it or perhaps tweak it?

Also bear in mind that if it is a fairly large change - you might not be very good at it at first. We try and keep a change on for a few weeks, to help people get over the learning curve of trying something new. If you had only given bicycle riding or swimming one try, chances are you are not ever going to do them again. Some things need a bit longer to get right. Your agile coach has years of experience with many teams. They are finely tuned into the signs of things that don’t work, and when your team should persevere. They also know when you’re

headed down a dangerous path and can help course correct you to a safer learning path.

Often teams new to scrum don’t finish anything in their first sprint. This is because it’s a huge change to how they work, and for the most part they don’t change **how** they work initially. They start and then just work the same way. Only after failing to deliver do they realize they need to change how they work. Then the second and third sprints are great. Conversely teams that succeed in their first sprint sometimes only learn this lesson (of work being done differently) a few sprints in, and then it hurts so much more.

Don’t be afraid to fail. It’s a sign that you can learn and improve. The trick is to fail safely and easily without harming anyone. Again, short feedback loops help make this possible.

Practices

These are practices found in the agile community. They are not required for you to be agile. They are simply helpful to some, and thus shared.

Rubber Duck

Yes, this is the yellow rubber duck found in bath tubs. In this case you can find them on peoples computer screens. The premise is that if you explain the problem to your rubber duck out loud, you usually frame and think of the problem differently and then can solve it yourself. Wikipedia link[25]

No Bug Rule

Agree that no more bugs are allowed in. This means if any bugs are found they are either immediately fixed or they become features. This policy allows you to start being honest about all those tiny bugs that you add to the backlog but have no plan of ever doing. Or you may still be under the illusion that you'll get to it one day? Well, if its been there for more than 3 months I can almost guarantee that it just wont get done. So delete it. Or fix it now. If you delete it and its really important it will make its way back to your team, and then you can

[25] https://en.m.wikipedia.org/wiki/Rubber_duck_debugging

fix it! No more huge bug lists. And no more releasing anything with a bug.

Boy Scout Principle

This is well known in software development circles[26], but can be applied to anything. Basically leave the campground cleaner than you found it. When you make a change or a fix to something, clean up around it too.

TDD

This stands for Test Driven Development. Many teams are familiar with unit testing but tend to add them after the fact. TDD is actually a way to code using the test first to guide your design and help you write clean code. It takes a mindshift. If your team doesn't do unit testing or maybe adds them sometimes, afterwards, then get a technical coach to help them see the value of this. TDD is also a excellent way perform pair-programming, having one person write tests while the other implements. "That does not mean the senior person code and the junior does the tests"

CI/CD pipeline

This is your Continuous Integration and Continuous Deployment pipeline. Some teams never think of this and just do it manually. If your deployments are slow and painful then this possibly doesn't exist for your

[26] https://medium.com/@biratkirat/step-8-the-boy-scout-rule-robert-c-martin-uncle-bob-9ac839778385

team. We like to encourage teams to think of this from the start. And to set it up immediately and continuously tweak it to work for them. Deployments should be a simple automated process with no stress - YES it is possible. Integration should be happening ALL the time, not in the last week of a project! Again, a technical coach will be able to work with a team to set this up and help them level up their skills to maintain this. The pipeline is the most important aspect of delivery, and needs to be fixed as a priority above all other tasks.

Telemetry

Designing systems in a manner so that they continuously deliver measurements and information, for the purposes of monitoring. This is not using a tool to instrument applications or monitor infrastructure, this is developers creating metrics for operations and application management.

Agile Testing

This has to do with thinking about testing in a completely different way. Usually we see testing as a phase after development to ensure quality by trying to break the system and report bugs. Instead we want testing to be a continuous thought in the entire teams heads. If you can prevent bugs up front and talk through test cases before any development gets done, the chances are much higher that there will be no misunderstandings. Think of this as the team (developers and testers) working together to build the best possible system. For more hands on practical examples of how to do this, ask

your agile coach to run an Agile Testing workshop with your team.

Hackathons

These rituals are often focused on product innovation, rather than improvements and paying down technical debt. Use this time to resolve issues that causes pain, and please involve your entire team, not just developers.

Tooling

Hopefully through this book, you have realised that DevOps and Agile is about getting a group of people to work effectively from requirement capture to development through testing and into deployment. Appropriate tooling for our team can allow them to both work and get more efficient communication. Think back to our ‘Improvements’ and ‘Help Needed’ boards. A good tool should help us to address bottlenecks and pain points. If it’s not, then you have to question whether the tooling is really needed.

Most people have their favourite tools and then try to use those tools for everything. Instead we suggest you select your tools as a team, don’t force tools on a team. Just because you have a license for a tool does not mean the team should use it at all costs. Encourage your teams to test various tools and discuss what works and doesn’t work. NEVER build your tool - this just leads to complications and distractions!

Most importantly, if it takes more than 2 hours for for

anyone on the team to figure out how to use the tool, ditch it. The added complexity is just not worth it.

Here are some of the tools we come across:

Communication Tools

Slack is our preferred communication tool. It keeps conversation together and allows me to dip in when I want to. The channels also allow me to opt in or out of certain topics of conversation. As with any communication tool - you should set up some general team guidelines on the use of this and when it is appropriate or not.

Decide as a team what your primary tool will be and stick to it. It's good to have a secondary tool, just in case someone can't use the primary one.

With smaller teams I prefer WhatsApp or even just skype left open all the time.

Source Control

Again most teams have this setup but don't talk about who must use it and what should be in there. The answer is everything. If your team decides to use Github then use that for documents as well. Teach everyone how to use it effectively. Think about scripts and databases and text files - all of these things should be kept together, safely, in a logical manner. Again, discuss this as a team. All too often someone tells me the file is in X, and then when I search I find 8 different copies of X in various folders being updated by different people.

From a DevOps perspective, every piece of code, script, library, configuration and even your delivery pipeline should be in source control. Yes operations, even your scripts.

Jira or similar

We would definitely recommend a team start with a physical board. When a team starts with agile, they iterate a few times until they find the ideal process for them. On a physical board this is easily done, by adding a column or changing a name. When tools are involved this become a little more difficult.

Do you have permissions to make changes? Do you need to create your own workflow (if other teams use the same workflow)? Can your tool even do what you would like it to do?

If you are using a tool, we recommend creating a sandpit(copy of your setup) to play with settings and not affect others in your organization. Our other rule is to simplify as much as possible. Get rid of compulsory things, have as few fields as possible. Simple is always your friend.

IDE

This is the Integrated Development Environment. Some teams use the same one, others have their favorites. Again, this is up to your team. Think about a new team member joining, what will be best? Ask your team what they would prefer. Some IDE’s cost money, but the benefits it brings to your team is worth a lot more than the cost. If your team says they need it - just get it. You wouldn’t want to chop a tree down with a blunt axe, so don’t expect your teams to.

Environment maintenance and deployment

It’s important that that our environments from test environments through to production can easily be updated

with new software your teams create, as well as be kept up to date with patches.

Not only can this allow single, repeatuable push-button deployment which reduces the requirement for support during production deployment. It also helps to deliver newly build code to a test environment for testing, meaning code is getting tested sooner and reducing our feedback loops.

Test automation

Although there is still a need for manual testing within DevOps and Automation, typically it's use is a lot more strategic. Within many teams we deal with there is only one or two testers. They cannot be expected to achieve similar testing in a single sprint to that which was achieved with a team of 8 over three months under waterfall.

To prevent testing becoming a bottle neck, test automation is often used to check a new build as soon as it's compiled. There are multiple different types of automation available – at the unit, API or user interface level. We recommend looking into the principle of the Testing Pyramid to see how multiple tools can help you build efficient coverage.

Ideally with your automation you need a suit which will run quickly (under an hour is ideal), and covers key pain points of the system which if failed highlight a painful failure of your system. You also need your test automation to be reliable, so that when you get a red flag, you take it seriously and don't behave with "it always does that".

Web Analytics

Web analytics allow you to understand more about how a system is being used. It collects together statistics about how people use your system.

Examples can include:

- What device do they use the site from?
- What features are most popular with users?
- Are there any point where users drop off?

An example could be an application form where a user is asked for their IRD number before they can continue, and causes 60% of users to drop off their application at this point. A future story might involve how it would be possible to make this process friendly in order to increase the user retention.

Tricky

These are topics that we consider to be tricky over time. This means that right now none of these might be a problem, but the more time you spent in an agile mindset, the chances are these topics will become burning issues for you and your team.

There is no one perfect answer. There are many opinions, research and experiences. We have decided to give you reading on various viewpoints on the topics. This way you can learn about the different viewpoints and find your own answers. Keep in mind that what works for you today might be different to what works tomorrow. And what works for one team, might not for another.

Value of a Scrum Master

The Value of a Scrum Master[27]

Does a ScrumMaster bring value?[28]

Estimates

The no-estimates Movement[29]

[27] https://www.growingagile.co.za/2018/07/the-value-of-a-scrum-master/

[28] https://www.agileleantransformation.com/blog/does-a-scrum-master-bring-value

[29] https://ronjeffries.com/xprog/articles/the-noestimates-movement/

What are Story Points[30]

Project Work

Flow Work to People[31]

Small Team Multiple Projects[32]

How to Handle Multiple Projects with a Small Team[33]

Deadlines

Fixed Date Fixed Scope Scrum[34]

Holidays

3 Ways to handle the Year End Holidays[35]

Overtime

Agile and Working Overtime[36]

Breaking the Rules of Agile - Working Overtime[37]

[30] https://www.mountaingoatsoftware.com/blog/what-are-story-points

[31] https://www.dropbox.com/home/Growing%20Agile/Resources/Articles?preview=WorkFlowToPeople.pdf

[32] https://davidmarquis.wordpress.com/2011/12/03/small-team-multiple-projects-agile-planning/

[33] https://medium.com/@anttihavanko/how-to-handle-multiple-projects-with-a-small-team-ebdfde5f6714

[34] https://www.growingagile.co.za/2012/10/fixed-date-fixed-scope-scrum/

[35] https://agileforall.com/3-ways-handle-end-year-holidays-agile-team/

[36] https://agileforall.com/agile-antipattern-working-overtime/

[37] https://www.mitchlacey.com/blog/breaking-the-rules-of-agile-working-overtime

Timesheets

Invoicing Without Timesheets[38]

5 Reasons to Kill Timesheets[39]

Growth Mindset

Why Growth Mindset Matters For Organisational Agility[40]

Growth Mindset and Agile Mindset[41]

Release Planning

Release Planning With Scrum[42]

Distributed Teams

Remote Agile Teams[43]

The Remote Backlog Grooming Session[44]

---

[38] https://www.growingagile.co.za/2015/10/invoicing-without-timesheets/

[39] https://www.agile42.com/en/blog/2014/08/21/5-reasons-kill-time-sheets/

[40] https://www.agilebusiness.org/blog/why-growth-mindset-matters-for-organisational-agility

[41] https://www.scrum.org/resources/blog/growth-mindset-agile-mindset

[42] https://www.growingagile.co.za/2012/10/release-planning-with-scrum/

[43] https://www.growingagile.co.za/2017/07/remote-agile-teams/

[44] https://www.growingagile.co.za/2017/04/the-remote-backlog-grooming-refinement-session/

User Story

10 Tips writing good User Stories[45]

Progress Tracking

The Only Software Metric you Need[46]

Metrics[47]

Team Self Selection

Self Selection Teams[48]

Team Self Selection Kit[49]

[45] https://www.romanpichler.com/blog/10-tips-writing-good-user-stories/

[46] https://www.growingagile.co.za/2014/03/the-only-software-metric-you-need/

[47] https://www.atlassian.com/agile/project-management/metrics

[48] https://www.infoq.com/news/2017/11/self-selection-teams

[49] http://nomad8.com/team-self-selection-kit/

Appendix

Technique Mind Maps

This technique was obtained by permission from GrowingAgile.co.za

What you can learn

Using mind maps in backlog grooming can help ensure everyone has a clear understanding of what is involved for a particular requirement. The can help identify outstanding questions, clarify what is in scope and out of scope, identify test cases, and identify how stories can be split.

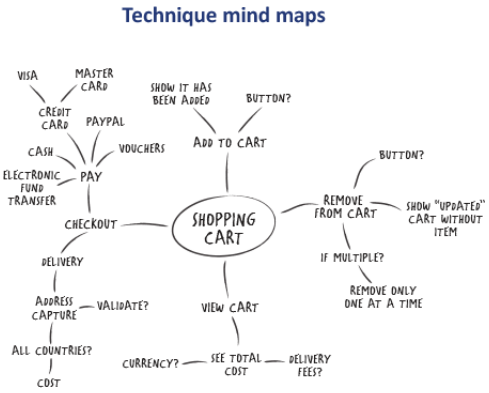
What you need

- Paper
- pens in a few colours

How to do this

Pick a feature that you need to groom and write it in the middle of a piece of paper. Now ask people what they need to know about this feature or what it entails. Draw each thing that is mentioned in a branch off the centre item. Group things as makes sense. Note this can explode very quickly. At this stage write everything down that people can think of, whether they should be included or not.

Here is an example for a shopping cart feature.

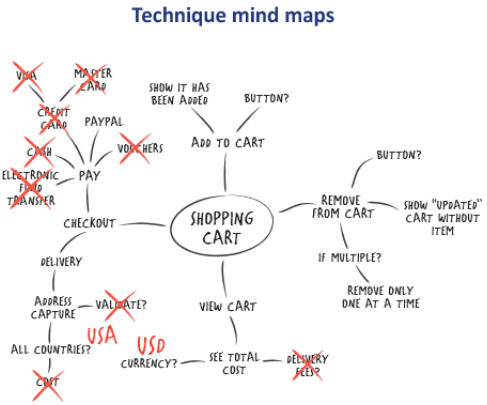


Questions

As the discussion progresses, if there are any questions that need to be answered, write those in a different colour, that will make it easy to see what the outstanding questions are.

Scope

Once the discussion has gone broad and covered everything people can think of, you can now make scope decisions. For example, for now we are only accepting Paypal payments. When you do this, put a cross through anything that is out of scope. This ensures everyone is clear on what is included and what is excluded. It is better to talk about something and then exclude it, than to leave it as an unspoken assumption.



Splitting stories

Once you've created the mind map it is very easy to split stories. Almost every branch off the mind map can be a separate story.

How we've used this

Although there are plenty of tools for mind maps, we prefer doing them on paper. They are a great visual reminder of the grooming conversation.

Who shared this with us

Janet Gregory

Technique Personal Maps

What you can learn

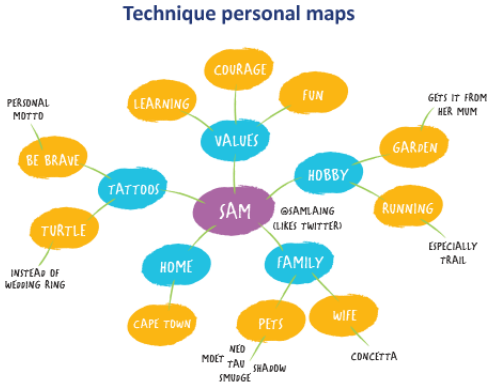
This technique can help you to better understand a colleague. It gives you a visual map of how well you know someone.

What you need

- Blank paper and pens

How to do this

On your blank paper write the name of someone in the middle. Then similarly to a mindmap, write some categories around this: home, education, work, hobbies, family, goals, values. Expand each of these categories with what you know about this person. Do they love cats? Write that down. Did they educate themselves online? Make a note. Now analyze your map - which areas are a bit blank? What are some ways you can fill in this blank area?



How we've used this

We have used this technique with teams to bring them closer together. They all create maps together for each other, asking questions and expanding as they go along. Very nice and informal setting works best, with food and drinks.

Who shared this with us

This is taken from the book Management 3.0 Workout[50] by Jurgen Appelo.

[50] http://www.management30.com/workouts/

Technique Problem Solving Tree

This technique was obtained with permission from GrowingAgile.co.za

What you can learn

Learn how to break a big complex problem down into small actionable steps that people can implement themselves.

What you need

- Post it notes.
- Markers.
- Flipchart paper or large wall to build a tree.

How to do this

Start with a problem you need to solve. Write this on a sticky note, and stick it at the top of the tree.

Now ask what you can do to solve the problem. Write each idea down on a sticky note and add them to the tree, just below the problem statement. For each idea, break the idea down another level into the tasks that the idea is made up of.

Use a different sticky note for each task, and add them to the tree under each idea.

Now ask if everyone understands the tasks and can do them immediately. If the answer is no, break it down into

smaller tasks and make another level in the problem solving tree.

Once a task is clear to everyone and can be done immediately, it is a leaf node. There is no need to break it down future. When you have lots of leaf nodes, dot vote to see which ideas to tackle first. Pick only a few to get started.

How we've used this

We have used this to brainstorm many ideas to solve difficult impediments. We have also used it to come up with retrospective actions once a team has identified a problem they want to solve.

Who shared this with us

Bob Sarni shared this technique in a workshop as a Scrum Gathering.

Technique Story Maps

This technique was obtained by permission from GrowingAgile.co.za

What you can learn

This is a technique to visualise a product and how it gets used from a users point of view. It is particularly helpful in identifying what the minimum requirements for a release are.

What you need

- A really long wall or table or empty floor space
- Sticky notes
- Markers

How to do this

Firstly identify a few users (or personas) for your product. Start with at least one, and no more than three. Start with the first persona, and discuss at a high level how that person interacts with your product.

For example: imagine we are looking at my morning routine from waking up to getting to work. I would list things like: Wake up, switch off alarm, go to bathroom, read twitter, coffee, dog food, shower, get dressed, medicine, breakfast, dog walk, pack bag, go to work.

We now look for items to group.

In my example, we might group switch off alarm under the high level activity ‘wake up’. Go to bathroom,

shower and get dressed might be grouped under ‘get ready’. Dog food and dog walk might be grouped under ‘dogs’. Coffee might be grouped under ‘breakfast’.

This top level is called the activities of the story map. We usually order it by time. i.e. the items that happen first on the left. This creates a high level timeline of a person interacting with your product across the top of your story map.

In my example the activities now look like this: Wake up, get ready, read twitter, dog, medicine, breakfast, pack bag, go to work.

Activities

Next we take each activity and identify what is called the backbone. The next level items in order of how your persona would do them.

In our example: if we break down get ready, we would get: go to bathroom, shower, get dressed.

Next we take each of these backbone items and break them down into small user tasks.

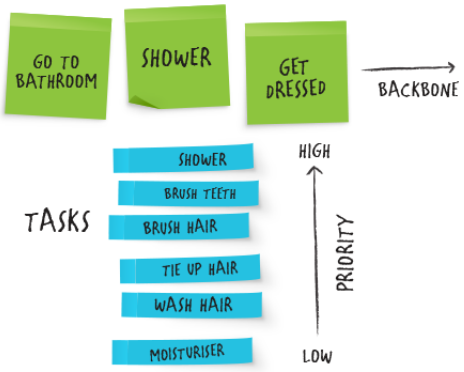
In my example if we take shower, I could break down the

tasks as follows: shower, wash hair, brush teeth, brush hair, tie up hair, put on moisturiser.

Once you have done the above steps for all the activities, we look at the tasks (level 3) and prioritise them.

In my example for shower I would prioritise as follows: shower, brush teeth, brush hair, tie up hair, wash hair, put on moisturiser. Meaning brushing my teeth is more important than washing my hair.

For the vertical direction it is not about what happens first, but rather what is more important.



Once all your tasks are prioritised, you can take your next persona and fill in any extra activities, backbone or tasks they might have.

Prioritising

You now want to go through each of the activities, look at their tasks and discuss what HAS TO BE in the next release. This is a great technique to see what really is high priority

In our example, let's think about what would happen if I woke up late and realised I only had 10 minutes to do everything.

My top level activities would become: Wake up, get ready, dog, medicine, go to work. (so read twitter, breakfast, pack bag are NOT going to happen!)

Within those activities there is much more that I could drop. For example, get ready would still have the same backbone: go to bathroom, shower, get dressed but the tasks under shower would become just brush teeth and tie up hair.

When we do this exercise we realise that although the shower and brush hair tasks were important, we now realised if really pushed for time we wouldn't do them. As a result we can update our story map to move these items lower.

Note : everyone has their own version of non-negotiables for personal hygiene! The details aren't important here, rather the principle of discussing each item to decide what MUST be included.

On a story map we show the minimum scope by drawing a line under the last item that HAS TO BE in the release. It is a lovely visual reminder of what to focus on.

How we've used this

We have used this in numerous companies to assist with visualising the work required for a release and to help

with scoping decisions of what should be in the release. We have also used this technique to release our website (multiple releases over a 4 week period) as well as our book series.

Who shared this with us

Jeff Patton created Story Maps.

Jeff Patton[51]

User Story Maps Book[52]

---

[51] https://jpattonassociates.com/user-story-mapping/

[52] https://www.amazon.com/User-Story-Mapping-Discover-Product-ebook/dp/B00NF07FHS

Datacom

Here at Datacom we want to help you do the best you can. Some ways we can help is to work with your teams on:

Assessments and workshops, such as team setup, impact mapping, agile testing, team building or even a codingdojo. We also offer industry accredited training.

Ultimately our goal is to help you deliver quality results that your customer truly values.

We have highly experienced coaches, two of which wrote this guide: Samantha Laing and Corneile Britz. On the following pages you can learn more about our coaches and their backgrounds

For more information please contact us at agile@datacom.co.nz[53].

[53] agile@datacom.co.nz

Samantha Laing

“Be brave”

Samantha Laing

Samantha lives by this personal motto which encourages her to embrace uncertainty and new challenges. This gives her a positive can-do attitude that teams find infectious.

Sam has worked in software development for nearly 20 years and has spent the last 8 years working as an Agile Coach and Scrum Master. She lives the Agile principles and values in everything she does, from planning her wedding with a Kanban board, to using an Agile approach to creating online training videos. If you visit her house, there will be sticky notes everywhere!

As a recent immigrant to New Zealand, you’ll still notice her South African accent. Before joining Datacom, Sam was the co-founder of Growing Agile, a highly successful Agile coaching company in South Africa, with a global client base. Besides being a creative sketch-noter, she’s

an accomplished author of 14 books on various Agile topics[54], and a keynote speaker that has spoken at numerous Agile conferences around the world.

Sam's key strength as an Agile coach is the ability to ask probing powerful questions that help people to view their situations with new lenses. She is a lifelong learner always looking to learn new ideas and techniques from courses and conferences.

Qualifications

- Bachelor of Science in Mathematics and Computer Science
- ICAgile Certified Instructor – Certified Agile Practitioner
- Certified Professional - Certified Agile Practitioner (ICAgile)
- Certified Professional - Foundations of DevOps (ICAgile)
- Certified Scrum Product Owner (Scrum Alliance)
- Certified Scrum Master (Scrum Alliance)
- Certified Corporate Coach (University of Cape Town)

Agile Knowledge

With over 10 years of experience using Agile, 8 of them in a coaching role, Sam has a huge depth of knowledge on many Agile frameworks and practices. She can explain the difference and benefits of Scrum vs Kanban, as well as teach a range of techniques to help

[54] https://leanpub.com/p/growingagile

with Agile testing, breaking down user stories, building user story maps to identify minimum scope, managing priorities and backlogs. As a coach she has worked with hundreds of teams in a range of different domains and technology stacks. She’s also helped a number of non-software teams apply Agile with great success.

LinkedIn Profile- https://www.linkedin.com/in/Sam Laing/[55]

[55] https://www.linkedin.com/in/SamLaing/

Corneile Britz

“Just start”

Corneile Britz

Corneile has over 20 years’ experience in solutions architecture, software development, business analysis and ultimately delivering business value. He has significant industry experience in financial services, and consultancy at the Big Four banks in South Africa. He has transformed teams and organisations during this time, developing a passion and deep skillset for DevOps, where optimising the whole value chain is vastly more important than optimising a part.

Driven by quality, innovation, simplicity and scalability, Corneile uses and promotes DevOps on a personal and professional level. He leverages his extensive technical background to transform teams beyond the theory, by providing practical and pragmatic coaching to improve even the most experienced teams.

Qualifications

- SAFe Program Consultant (SPC4)
- SAFe for Teams
- ICAgile Certified Instructor – Foundations of DevOps
- Microsoft Certified Solutions Developer (MCSD)
- Microsoft Certified Application Developer (MCAD)
- Web Security Specialist Essentials
- Business Analysis

Consulting Experience

Corneile is driven to improve the entire value chain of any organisation. As a DevOps Consultant, Corneile has effectively transformed and coached teams in numerous departments and sectors to adopt new culture, processes and toolchains. Having often faced the obstacle of adoption resistance, he has successfully mentored and trained individuals and teams to deliver beyond their own expectations.

Corneile believes that a team should be trusted and autonomous, free to deliver maximum value. Once completed, the teams are capable of improving on their own, understanding the greater purpose of their day-to-day work.

Bringing a deep technical development background to the consulting space, he is capable of giving the direction as well as mentoring as a peer.

LinkedIn Profile- https://www.linkedin.com/in/corneilebritz/[56]

[56] https://www.linkedin.com/in/corneilebritz/

Links

All the links for this book can be found here, per chapter. This is the internet - so links can dissapear. If the link is broken below, first try googling the terms - the owner may just have moved the post.

Datacom Agile and Devops

- Please email agile@datacom.co.nz[57] for more information about how we can help you.

Who is this book for?

- Agile Manifesto[58]
- Wikipedia Devops[59]

Process Planning

- Free 1 hour Scrum video[60]
- Kanban book[61]

What do you need to do?

- The Power of Why[62]

[57] agile@datacom.co.nz
[58] http://agilemanifesto.org/
[59] https://en.m.wikipedia.org/wiki/DevOps
[60] http://growingagile.thinkific.com/courses/scrumbasics
[61] https://leanpub.com/kanbanworkbook
[62] https://www.ted.com/talks/simon_sinek_how_great_leaders_inspire_action

- User Story Mapping[63]

Visualise

- different board designs[64]
- one projects porfolio kanban board[65]
- The Bottleneck Rules[66]

Improvements

- Kaizen[67]

Help Needed

- Impact Mapping[68]

Management

- peoples behaviours and how environment affects this[69]
- Situational constraints typically predict behavior[70]
- better than personality[71]

[63] https://www.amazon.com/User-Story-Mapping-Discover-Product/dp/1491904909

[64] http://brodzinski.com/2015/04/portfolio-kanban-board.html

[65] http://brodzinski.com/2011/11/project-portfolio-kanban.html

[66] https://www.amazon.com/Bottleneck-Rules-More-Working-Harder-ebook/dp/B07DCFR7B4

[67] https://www.slideshare.net/mobile/mgraban/mark-graban-intro-to-lean-frisco/42-Kaizen_Card_Idea_Card_Standardized

[68] https://www.impactmapping.org/

[69] https://medium.com/@dr_eprice/laziness-does-not-exist-3af27e312d01

[70] http://psycnet.apa.org/record/1979-28632-001

[71] https://pdfs.semanticscholar.org/a85d/432f44e43d61753bb8a121c246127b562a39.pdf

Team Setup

- Are you an I or a T?[72]
- On I shaped and T shaped skills[73]
- pairing[74]
- mobbing[75]
- Teaming[76]
- Small team, multiple projects[77]
- Multiple projects with a small team[78]

Team Agreements

- set of cards to help you[79]

Meetings

- This is a site[80]

[72] https://www.forbes.com/sites/andyboynton/2011/10/18/are-you-an-i-or-a-t/2/#4db1cb343cf7

[73] http://blog.ipspace.net/2015/05/on-i-shaped-and-t-shaped-skills.html

[74] https://medium.freecodecamp.org/the-benefits-and-pitfalls-of-pair-programming-in-the-workplace-e68c3ed3c81f

[75] https://underthehood.meltwater.com/blog/2016/06/01/mob-programming/

[76] https://www.growingagile.co.za/2018/05/teaming-could-this-be-the-future-of-agile/

[77] https://davidmarquis.wordpress.com/2011/12/03/small-team-multiple-projects-agile-planning/

[78] https://medium.com/@anttihavanko/how-to-handle-multiple-projects-with-a-small-team-ebdfde5f6714

[79] https://www.growingagile.co.za/?s=team+agreements

[80] https://retromat.org/en/

Work

- The Show-Me Column[81]

Practices

- Wikipedia link[82]
- software development circles[83]

Tricky

- The value of a Scrum Master[84]
- Does a ScrumMaster bring value?[85]
- The no-estimates Movement[86]
- What are Story Points[87]
- Flow Work to People[88]
- Small Team Multiple Projects[89]
- How to Handle Multiple Projects with a Small Team[90]

---

[81] https://www.growingagile.co.za/2018/05/showme-column-agileweeklytip/

[82] https://en.m.wikipedia.org/wiki/Rubber_duck_debugging

[83] https://medium.com/@biratkirat/step-8-the-boy-scout-rule-robert-c-martin-uncle-bob-9ac839778385

[84] https://www.growingagile.co.za/2018/07/the-value-of-a-scrum-master/

[85] https://www.agileleantransformation.com/blog/does-a-scrum-master-bring-value

[86] https://ronjeffries.com/xprog/articles/the-noestimates-movement/

[87] https://www.mountaingoatsoftware.com/blog/what-are-story-points

[88] https://www.dropbox.com/home/Growing%20Agile/Resources/Articles?preview=WorkFlowToPeople.pdf

[89] https://davidmarquis.wordpress.com/2011/12/03/small-team-multiple-projects-agile-planning/

[90] https://medium.com/@anttihavanko/how-to-handle-multiple-projects-with-a-small-team-ebdfde5f6714

- Fixed Date Fixed Scope Scrum[91]
- 3 Ways to handle the Year End Holidays[92]
- Agile and Working Overtime[93]
- Breaking the Rules of Agile - Working Overtime[94]
- Invoicing Without Timesheets[95]
- 5 Reasons to Kill Timesheets[96]
- Why Growth Mindset Matters For Organisational Agility[97]
- Growth Mindset and Agile Mindset[98]
- Release Planning With Scrum[99]
- Remote Agile Teams[100]
- The Remote Backlog Grooming Session[101]
- 10 Tips writing good USer Stories[102]
- The Only Software Metric you Need[103]
- Metrics[104]

[91] https://www.growingagile.co.za/2012/10/fixed-date-fixed-scope-scrum/
[92] https://agileforall.com/3-ways-handle-end-year-holidays-agile-team/
[93] https://agileforall.com/agile-antipattern-working-overtime/
[94] https://www.mitchlacey.com/blog/breaking-the-rules-of-agile-working-overtime
[95] https://www.growingagile.co.za/2015/10/invoicing-without-timesheets/
[96] https://www.agile42.com/en/blog/2014/08/21/5-reasons-kill-time-sheets/
[97] https://www.agilebusiness.org/blog/why-growth-mindset-matters-for-organisational-agility
[98] https://www.scrum.org/resources/blog/growth-mindset-agile-mindset
[99] https://www.growingagile.co.za/2012/10/release-planning-with-scrum/
[100] https://www.growingagile.co.za/2017/07/remote-agile-teams/
[101] https://www.growingagile.co.za/2017/04/the-remote-backlog-grooming-refinement-session/
[102] https://www.romanpichler.com/blog/10-tips-writing-good-user-stories/
[103] https://www.growingagile.co.za/2014/03/the-only-software-metric-you-need/
[104] https://www.atlassian.com/agile/project-management/metrics

- Self Selection Teams[105]
- Team Self Selection Kit[106]

Technique Personal Maps

- Management 3.0 Workout[107]

Technique Story Maps

- Jeff Patton[108]
- User Story Maps Book[109]

Samantha Laing

- LinkedIn Profile[110]

Corneile Britz

- LinkedIn Profile[111]

[105] https://www.infoq.com/news/2017/11/self-selection-teams
[106] http://nomad8.com/team-self-selection-kit/
[107] http://www.management30.com/workouts/
[108] https://jpattonassociates.com/user-story-mapping/
[109] https://www.amazon.com/User-Story-Mapping-Discover-Product-ebook/dp/B00NF07FHS
[110] https://www.linkedin.com/in/SamLaing/
[111] https://www.linkedin.com/in/corneilebritz/