- Scrum Guide 2013.
- Go through differences b/w.
  2016 Guide.
- Discuss differences.

# Definition of Scrum.

Scrum (n): A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

Analysis: Scrum is a framework. Not a methodology. Framework implies something you build upon.
A framework implies it is not complete

Analysis, Black + Red + Blue

It addresses complex and adaptive problems because it uses an empirical process to analyze the work done.
By regularly 'checking' to see if the right things are being done.

Productively - By continuously looking for ways to do things better (Retrospective)

Creatively - By using everyones' input to solve problems. Problems are raised through daily scrum & retrospective.

Highest Possible Value through defining Quality through the definition of Done.
Never compromising on quality.

Open to abuse. People who use scrum as a method to get things done quickly leave out quality altogether.

Scrum is:

- Lightweight
- Simple To Understand
- Difficult To Master.

Because it is so simple, it is easily communicated.

Because it is simple, things can be skipped and misinterpreted. Opening up avenues to abuse.
Its simplicity can give a false sense of mastery. Especially when not fully understood.

Scrum is a process framework that has been used to manage complex product development since the early 1990s.
It is not new. Has been around for over 20 years.

Age does not guarantee feasability - But its a good argument for -A-

Scrum is not a process or techniques for building products; rather, it is a framework within which you can employ various processes and techniques.

A framework is something you build upon. A base.

The mere fact it is a framework invits chopping and changing to suit your needs - it implies things can be "taken out"

Scrum makes clear the relative efficacy of your product management and development practices so that you can improve.

Efficacy - The ability to get the job done under ideal conditions.

https://en.m.wikipedia.org/wiki/Efficacy
https://en.m.wiktionary.org/wiki/efficacy

By making things clear, you can change it things are bad.

Under ideal conditions, Doesn't work in the real world! (Bah Humbug)
Conditions are never ideal.
Because you have visibility - nothing is stopping you from changing things to make them ideal. Hence improve!

The scrum framework consists of, Scrum Teams and their associative roles, events, artifacts, and rules. Each component within the scrum framework serves a specific purpose and is essential to Scrum's success and usage.

Everything in Scrum serves a purpose - it is not put there just for the hell of it!
But we don't need all those things. It takes up too much time! We can live without it all.

The rules of Scrum bind together the events, roles, and artifacts, governing the relationships and interactions between them. The rules of Scrum are described throughout the body of this document.

The rules of scrum dictate how the roles, events, artifacts— overall the components interact with one another.

Rules? Who needs rules. Can't you just get the 'gist' of it?

Specific tactics for using the Scrum framework vary and are described elsewhere.

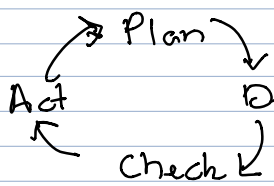
Since scrum is a framework to build upon. How to build upon it is covered outside this document.
It also means scrum can be used outside software development.

Since how to follow scrum is not covered for every situation and you need to do more reading - eg wonder a majority of scrum/agile implementations and transformations fail!

## Scrum Theory

Scrum is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions on what is known. Scrum employs an iterative, incremental approach to optimize predictability and control risk.

- Make an assumption
- Test that assumption
- Evaluate that assumption
- Determine what to do with the assumption
- Repeat.




Use a feedback loop to determine decisions. Base those decisions on evidence gathered. Not speculation.

But I just want to know the one true way that will solve all my problems.
Scrum does not give me that.
It does not explain why it is better (or the best) process!

Three pillars uphold every implementation of empirical process control. Transparency, Inspection, and adaptation.

## Transparency

Significant aspects of the process must be visible to

those responsible for the outcome.
Transparency requires those aspects be defined by a common standard so observers share a common understanding of what is being seen.

Stakeholders/Developers, basically everyone involved must have a common language and point of reference to understand what is being seen - the process itself and all aspects of its implementation.

For example:

- A common language referring to the process must be shared by all participants; and,
- Those performing the work and those accepting the work product must share a common definition of "Done"

You don't need a common language, just 'tell' the developers what to do - They are smart. They will figure it out.

Done is 'done' when I say it is 'done'! So what if it hasn't been tested. Then it will be 'done-done'.

## Inspection

Scrum users must frequently inspect Scrum artifacts and progress toward a sprint goal to detect undesirable variances. Their inspection should not be so frequent that inspection gets in the way of the work. Inspections are the most beneficial when diligently performed by skilled inspectors at the point of work.

In other words, keep looking for problems (undesirable variances). Don't let the looking (and subsequent fixing) of problems get in the way of doing the work. Use the work to find and fix the variances (variances taken from W. Edwards Deming?)
The best inspectors are the ones who both do the work, but also know what variances to look out for. Problems can also be defects!

But I don't have any variances/problems!

Not having variances/problems is a problem in and itself. LOOK HARDER!

## Adaptation

If the inspector determines that one or more aspects of the process deviate outside acceptable limits, and the resulting product will be unacceptable, the process or the material being processed must be adjusted. An adjustment

must be made as soon as possible to minimize further deviation.

Here a feedback loop is established based on quality. If there is a deviation in "quality", be it software quality or process quality then an "adjustment" is made to bring it back in line.
Also, the adjustment must be made as soon as possible, not only in terms of time (which dictates the cycle time of the feedback loop) but also in terms of where the variance is happening.
By this I mean, fix the variance at the source, not after the inspection, i.e. after the fact.

We have tests, we'll raise a ticket so the "variances" are fixed at some point in the near or distant future (if at all).

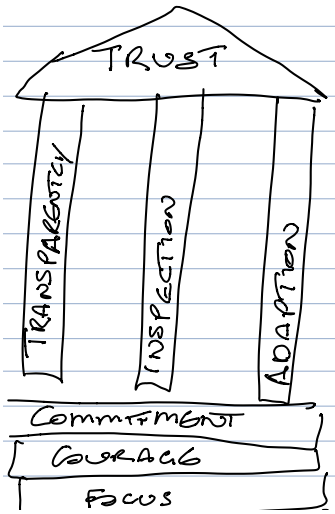
Scrum prescribes four formal events for inspection and adaptation, as described in the Scrum events section of this document:

- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective.

- Meetings, meetings and more meetings.

## Scrum Values

When the values of Commitment, courage, focus, openness and respect are embodied and lived by the Scrum team, the Scrum pillars of Transparency, inspection and adaption come to life and build trust for everyone.

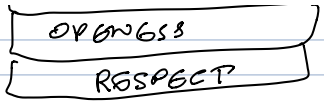



SCRUM HOUSE ?

The Scrum Team Members learn and explore those values as they work with the Scrum events, roles and artifacts.

Successful use of Scrum depends on people becoming more proficient in living these five values.

People personally commit to achieving the goals of the Scrum team. The Scrum team members have courage to do the right thing and work on tough problems.




The right thing is not to use Scrum.

Everyone focuses on the work of the Sprint and the goals of the Scrum Team.
Do not focus too much on just the work, but also improving the work. Focusing on only the work leads you into a rut where you don't get better, just keep doing the same thing - because it gets the work done. In this case, especially when things are most busiest, it takes courage to stop doing the work and focus on improving the work.
The Scrum Team and its stake holders agree to be open about all the work and challenges with performing the work.
Scrum Team members respect each other to be capable, independent people.

Here these values and pillars are used to help build trust in the team.
Trust is when you trust the scrum master to have your best interests. Trust that the product owner will not waste your time on meaningless work.
Trust that your fellow Development team members have your back.

Without trust (5 Dysfunctions of a Team), you know your fellow team members will not have your back. Put you down on your weaknesses. Discourage ideas that are different.
This leads to lack of conflict. The ability to discuss the non-conformative. In other words, absense of courage and openess.
Since people no longer feel happy. They have no say, they switch off. Are no longer committed. They no longer have authority over how to do things. They are just told what to do thus lack the sense of responsibility. Thus have no respect for their fellow team members.
Thus there is no focus on the goals. This is where Scrum fails.

## The Scrum Team

The scrum team consists of a Product owner, the Development Team, and a Scrum Master.
Scrum teams are self organizing and cross-functional.
Self organizing teams choose how best to organize the work, rather than being directed by others outside the team.

By letting teams self organize rather than be told what to do, they are given ownership and authority over their workload.
Studies have shown that when you have a high sense of control (bomat) over your work, you are less likely to suffer from stress, and actually be more happy. You also work harder.
You have more control over your destiny.

Self organizing! Teams would not do anything - People are lazy if they are not told what to do.
The work will never get done.

Cross functional teams have all competencies needed to accomplish the work without depending on others not

part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity.

By having team members that are cross functional - but not nessesarily proficient in all competencies, you can avoid delays when someone is for example sick or on leave. Someone can continue.
The team can also share the workload. No single person is carrying the team because they are the only person who knows that competency.

The term "Jack of all trades - master of none" comes to mind. Good luck finding people who know everything

It is the team as a whole who become the masters. Not individuals.

Scrum Teams deliver products iteratively and incrementally, maximizing opportunity for feedback.
Incremental deliveries of "Done" product ensure a potentially useful version of working product is always available.

You are able to seek feedback sooner. Make corrections earlier and thus reduce the chance of going too far building the wrong thing. Thus reducing wasted effort, saving time and money.

Just get the damn specs right in the first place. Once it locked down and signed in blood. Then it becomes the customers fault if they got it wrong in the first place.

## The Product Owner

The product owner is responsible for maximising the value of the product and the work of the Development Team. How this is done may vary widely across organisations, Scrum Teams, and individuals.

The Product Owner is responsible for making sure that the development team optimises its work to produce value.
This can be done by

- Prioritising work based on value.
- Communicating the backlog to the development team so they understand what needs to be developed.
- I'm getting ahead of myself.

Yes - if you used Scrum to build a house, you would build the roof first as that gives the most value, Then the walls then the floor.
Side Note: Yes - someone has used that argument with me before.

The product owner is the sole person responsible for managing the Product Backlog. Product Backlog management includes:

- Clearly expressing Product Backlog items.
- Ordering the items in the Product Backlog to best achieve goals and missions;
- Optimising the value of the work the Development Team performs;
- Ensuring that the product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and
- Ensuring the Development Team understands items in the Product Backlog to the level needed.

The Product owner needs to make sure the Development team has enough information to hit the ground running when they start an item. This may be before the sprint (preferable) or during - but it is the product owners responsibility.

So what does the Scrum Master Do? Seems like the Product Owner does all the work!.

The Product Owner may do the above work, or have the Development Team do it. However, the Product Owner remains accountable.

The Product owner is one person, not a committee.

This is so that only one persons vision is full filled. Not a bunch of random requirements from different people.

- Robocop II comes to mind where his directives were done by committee. He pretty much could not do anything without violating something.

The Product owner may represent the desires of the committee in the product Backlog, but those wanting to change a Product Backlog item's priority must address the product Owner.

What happens if you get a product owner who ignores the committee or the users?

That is a risk - in which case it is for the committee and product owner to work out.

For the Product Owner To Succeed, the entire organization must respect his or her decisions. The Product Owner's decisions are visible in the content and ordering of the Product Backlog. No one is allowed to tell the Development Team to work from a different set of requirements, and the Development Team isn't allowed to act on what anyone else says.

This is actually to minimize the chopping and changing of tasks and when a developer is asked by 5 different managers to get their little piece of work done

But how does anything get done? Manager A needs his thing. Manager B needs his! I have mine. It all needs to be done and it all needs to be done now!

## The Development Team

The development team consists of professionals who do the work of delivering a potentially releasable increment of "Done" product at the end of each sprint. Only members of the Development team create the Increment.

Basically treat each sprint as a mini project. You do not have projects where you do only documentation or a project that only does testing. You do everything and at the end of the project - you release.
At the end of your sprint - you need to be in a position to release. If needed! This then puts the decision to release on the product owner (the Business) and not hampered by the developers team not being ready.

But we are not in a position to release at the end of the sprint! There is too much to be done!

You have taken on too much work for the Sprint.
- Slow down. Sometimes you need to slow down to speed up.
- You need to break down the work more.

Development teams are structured and empowered by the organization to organize and manage their own work. The resulting synergy optimizes the Development Team's overall efficiency & effectiveness.

The team chooses the work they do and how they do it.
By empowered, they have the authority to accept or reject work as they see fit.
If an item in the backlog does not have enough info for the development team to work on - or for they are uncomfortable doing - they have the authority to say 'No',

The team also works out the best way it can get the work done.

But what if the Development team gets it wrong?
They completely screw up.
They need to be 'told' what to do so this doesn't happen.

Yes a team can screw up. Although the team cannot be 'told' what to do, they can be guided.
If they screw up - then they learn what they did didn't work. Instead of wasting months - only a sprint (up to 1 month) is wasted.
This is a chance for all to learn.

Development teams have the following characteristics:

- They are self organising. No one (not even the Scrum Master) tells the development Team how to turn a product backlog into increments of potentially releasable functionality;
- Development Teams are cross-functional, with all of the skills as a team necessary to create a product increment.
- Scrum recognises no titles for Development Team members other than Developer, regardless of the work being performed by the person; there is no exceptions to this rule;

  - No senior developer.
  - No architect
  - No tester
  - No DBA

  etc,

- So who leads the team if there is no senior?

- Its an Egaletarian structure. Everyone has a say. Everyone is equal.

- There would be anarchy!
Which is why the team is made up of 'Professionals' and not animals!

- Scrum recognises no Sub-teams in the Development team, regardless of particular domains that need to be addressed (like testing or business analysis); there are no exceptions to this rule; and,
- Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the development team as a whole.

All for one and one for all. Everyone needs to look out for one another.

## Development Team Size

Optimal development team size is small enough to remain nimble and large enough to complete significant work within a sprint. Fewer than three Development team members decrease interaction and results in smaller productivity gains.

If you have 1 or 2 developers, the interaction between developers is already there. Scrum could actually decrease that interaction because of the formal times. This causes the productivity gains to not be as much as for larger teams.

So you see - scrum doesn't work - albeit for smaller teams.

Smaller teams may encounter skill constraints during the sprint, causing the development team to be unable to deliver a potentially releasable increment.

- Too few people may not have all the required skills
  - Not always.

Having more than nine members requires too much coordination. Large development Teams generate too much complexity for an empirical process to manage. The Product Owner and Scrum Master are not included in this count unless they are also executing the work of the sprint backlog.

The more people you have, the more communication channels there are. We as humans can only support so much at any one time.
The equation is

$$\text{Communication Channels} = n(n-1)/2$$

So for 5 people ($n=5$)

$$5(5-1)/2 = 10 \text{ channels}$$

for 9 people (n=9)

$9(9-1)/2 = 36$ channels.

For 10 people (n=10)

$10(10-1)/2 = 45$ channels.

At this point (and most likely later) our brains cannot keep up with this number of people at once.

See - Scrum doesn't work with large teams either - Only a specific set. Business doesn't work that way. It might work well within a startup for a web application but not Enterprise -

Enterprise is doing it wrong - There are many studies that show this.

- Mythical Man Month - Brooks.

"Adding more people to a late project only makes it later"

## The Scrum Master

The Scrum Master is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum Theory, practices, and rules.

The Scrum Master is a Servant leader for the Scrum Team. The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.

The Scrum Master:
- Helps the team understand Scrum
- Helps them stay within the Scrum framework.
  Without a Scrum Master, practices may slip back to what is comfortable, which is the process already known
  - Waterfall.
  - Standard Command & Control Management.
  - Or worse - Cowboy development in the name of Agile.

The Scrum Master does this by not being in charge - but by guiding the team.
A Servant leader is one that coaches to get you to do your best. They do not tell you what to do, but help you make the right decision.

Who is in charge then? If the scrum master does not direct, then developers will do nothing.
Its the inmates running the asylum, and the Scrum Master does nothing.

Teams are self directed much like a coach to a sporting team.

## Scrum Master Services To The Product Owner

The Scrum Master Serves the Product Owner in Several Ways, including:

- Finding techniques for effective product Backlog management.
  - Such as user story mapping
  - Impact Mapping
  - etc.
- Helping the Scrum Team understand the need for clear and concise Product Backlog items.
  - The more clear the team is on what needs to be built, the less time is wasted asking silly questions or getting stuck.
- Understanding product planning in the empirical environment

Don't plan everything up front. Do a bit. Evaluate & plan the next move.

- Ensuring the Product Owner knows how to arrange the product backlog to maximize value.

Draw out from the product owner what he could use now, not at the end of the project.

- Understanding and practicing agility; and,
- Facilitating Scrum Events as requested or needed.

Help the product owner become more available to the team.

# Scrum Master Service To The Development Team

The Scrum Master serves the development Team in several ways, including:

- Coaching the Development Team in Self-organized and cross functionality;

Guide the team to make good decisions without making the decision themself.
This is easier said than done as the team will make mistakes. the trick is to let them make the mistake, have them recognize the mistake and let them correct the mistake.

Also have team members work in areas that is not their specialty. This can help the team become more cross functional.

- Help the team create high value products;
  - Don't ignore quality.
  - Reduce waste.
  - Get better at their skills.
- Removing impediments to the Development Team's progress;
  - Anything that stops the team from completing a task.
    - People
    - Process
    - Technology.
    - Early identification.
  - Help the team remove its own impediments.
    - Part of Self organization.
- Facilitating Scrum events as requested or needed; and,
- Coaching the Development Team in Organizational environments in which Scrum is not yet fully adopted and understood.
- Try to prevent the teams from reverting back to their old ways. Especially during the initial transition period.

## The Scrum Master Service To The Organisation

The Scrum Master serves the Organisation in several ways, including:

- Leading and coaching the organisation in the Scrum adoption;
- Planning Scrum Implementations within the organisation;
- Helping employees and Stakeholders understand and enact Scrum and empirical product development;
- Causing change that increases the productivity of the Scrum Team; and,
- Find ways & experiments to help increase productivity.
  - Research different Methods.
- Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organisation.
- Collaborate together
- Share ideas
- Not everyone wants to do Scrum. Don't force it down my throat.
- Hence the Scrum Master needs negotiation skills.
- Not everyone can work under Scrum. This needs to be taken into account.

## Scrum Events

Prescribed events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum.

Note: it says minimize meetings not defined in Scrum. Not exclude them.
All events are time-boxed events, such that every event has a maximum duration.
Once the Sprint begins, its duration is fixed and cannot be shortened or lengthened.
If you decide 2 week sprints. Stick to it. If you decide 1 week. Stick to it.
Some teams start to try different lengths when they first start.

They may try 2 weeks, 1 week or 3 weeks. But once they make a decision - stick to it.
Don't chop and change sprint to sprint when it is convenient. Make it like clockwork.

I don't work in 2 or 3 week blocks. I just do the work.

The sprint gives a regular time everyone knows and can thus work to. It will also force you to create something in the end. The pressure (not too much - not too little) can help foster innovation as you have to produce something.

The remaining events may end up whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process.

By fixing the maximum time, you prevent events from taking too long. It prevents them from going forever.
Also helps focus the event to get a result.

Other than the sprint itself, which is a container for all other events, each event in Scrum is a formal opportunity to inspect and adapt something.

look at each event as an opportunity to look back at what you have done and try to improve. Don't just leave it for the Retrospective.

These events are specifically designed to enable critical transparency and inspection. Failure to include any of these events results in reduced transparency and is a lost opportunity to inspect and adapt.

Also too if you treat the event as a "tick in the box" event, you reduce its effectiveness.

## The Sprint

The heart of Scrum is the Sprint, a time-box of one month or less during which a "Done", usable and potentially releasable product increment is created.

Yes you should create something useful. Not documentation. Not a plan. Not an environment.
It can be small. Only do one specific thing in a narrow field, but its usable.

Sprints have consistent durations throughout the development

effort. A new sprint starts immediately after the conclusion of the previous sprint -

There is no "few days & planning" before the next sprint. Everything is incorporated in the Sprint.

Sprints contain and consist of the Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective -

Wow, with all that stuff, when do you actually work?

A Sprint does nothing outside a normal project. It just does it in smaller blocks. Rather than all planning up front. such as waterfall.

During the Sprint

- No changes are made that would endanger the sprint goal.
- Quality goals do not decrease; and.

This is important. When companies start scrum initially, they push to get things done quicker. When this happens the first thing to go is quality.

But saying that. What is Quality? You need to seriously look at what quality is and strip out the waste.

eg. Documentation. Having a 100 page spec that no-one reads or can understand is not quality.

Having a consise document or diagram that everyone understands is -

Also have coding standards, etc well documented and be able to be verified. Nothing based on individual opinions.

- Scope may be clarified and re-negotiated between the product owner and development team as more is learned -

Scope is not fixed. It varies as you learn more.

Each sprint may be considered a project with no more than a one month horizon. Like projects, Sprints are used to accomplish something - Each Sprint has a definition of what is to be built, a design and a flexible plan that will guide building it, the work, and the resultant product.

How many projects just produce docs or specs?

Each Sprint is a self contained project. Just smaller.

And no - its not quite like a small waterfall. Its so much more.

Sprints are limited to one calendar month. When a Sprint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase. Sprints enable predictability by ensuring inspection and adaption of progress toward a Sprint Goal at least every calendar month. Sprints also limit risk to one calendar month of cost.

## Cancelling A Sprint

A sprint can be cancelled before the Sprint time-box is over. Only the Product Owner has the authority to cancel the Sprint, although he or she may do so under influence from the Stakeholders, the Development Team, or the Scrum Master.

A Sprint would be cancelled if the Sprint Goal becomes obsolete. This might occur if the company changes direction or if the market or technology conditions change. In general, a Sprint should be cancelled if it no longer makes sense given the circumstances. But due to the short duration of Sprints, cancellation rarely makes sense.

Sprints are cancelled if a radical change in Goal is required. Something seriously must be wrong if you cannot wait 2 weeks or a month before you chop and change.

When a sprint is cancelled, any completed and "Done" Product Backlog items are reviewed. If part of the work is potentially releasable, the Product Owner typically accepts it.
All incomplete Product Backlog items are re-estimated and put back on the product backlog. The work done on them depreciates quickly and must be frequently re-estimated.

Knowledge fades fast. If you are working on something, have to put it aside for a period of time, then come back to it, you need to get re-acquainted with the work. That takes time.

Sprint Cancellations consume resources, since everyone has to re-group in another Sprint Planning to start another sprint.
Sprint Cancellations are often traumatic to the Scrum Team, and are very uncommon. They are a waste.

# Sprint Planning

The work to be performed in the Sprint is planned at the Sprint Planning. This plan is created by the collaborative work of the entire Scrum team.

Sprint Planning is time boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches the Scrum Team to keep it within the time-box.

Eight Hours! Seriously. When do you get the work done!

Eight Hours is only 1 day. That is 1/20 th the time. A small price to pay to know what you are doing during the Sprint.

Sprint Planning answers the following:

- What can be delivered in the increment resulting from the upcoming Sprint?
- How will the work needed to deliver the increment be achieved?

## Topic One: What can be done this Sprint?

The development team works to forecast the functionality that will be developed during the Sprint. The Product Owner discusses the objective that the Sprint should achieve and the Product Backlog items that, if completed in the Sprint, would achieve the Sprint Goal. The entire Scrum team collaborates on understanding the work of the Sprint.

The objective of the sprint is some functionality that can be used. Something small, but useful. It may be incomplete, but still useful. The Product Owner needs to get what is in their head into the Developers heard with regards to the objective - not the solution. But - the whole Scrum team needs to work to understand.

This is different to standard practise where a Spec is handed over and everyone works off it. Specs can be interpreted differently. The idea here is to remove the different interpretations, remove ambiguity to the egal.

The input to this meeting is the Product Backlog, the latest product increment, projected capacity of the Development Team during the Sprint, and past performance of the Development Team.

The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team. Only the Development Team can assess what it can accomplish over the upcoming Sprint.

The Development team has the right to refuse work. This is hard because most people want to please. The environment must also be safe enough for the Development team to reject work. Therefore the Scrum Master should touch the team and the Product Owner should respect the decision.

Contrary to current Management where you get what you have been given.

Will never work. Management pushes.

After the Development Team forecasts the Product Backlog items it will deliver in the Sprint, the Scrum team Crafts a Sprint Goal.

The whole Scrum Team Crafts the Goal.
- Dev Team
- Scrum Master
- Product Owner.

The Sprint Goal is an objective that will be met within the Sprint through the implementation of the Product Backlog, and it Provides guidance to the Development Team on Why it is building the increment.

The Sprint Goal is functionality to be delivered.
Eg - For a Blogging product
- Be able to post a blog
- Purchase Order System
  - Be able to raise a purchase order.
- Ambiguous
  - Make CEO happy

## Topic Two: How will the Chosen Work get done?

Having set the Sprint Goal and selected the Product Backlog items for the Sprint, the Development Team decides how it will build this functionality into a "Done" product increment during the Sprint. The Product Backlog items selected for this Sprint plus the plan for delivering them is called the Sprint Backlog.

The second half of Sprint Planning is determining the Plan to produce a done increment.
Remember, the Backlog items should not be a "solution". They are required functionality.

The Development Team usually starts by designing the System and the work needed to convert the Product Backlog into a product Increment. Work may be of varying size, or estimated effort.
Note. Estimation of 'effort' not time.
However, enough work is planned during the Sprint Planning for the Development Team to forecast what it believes it can do in the upcoming Sprint. Work planned for the first days of the Sprint by the Development Team is decomposed by the end of this meeting, often to units of one day or less.
Note. no mention of Stories. Stories are not part of Scrum.
Work is broken down from large to small.
One day or less is not just development, but everything inc. Testing where possible.
The Development Team self organises to undertake the work in the Sprint Backlog, both during Sprint Planning and as needed throughout the Sprint.

Self organises, thus no one including the Scrum Master assigns tasks to people.
The Scrum Master must encourage the team to self Organise.

Without Someone directing, how will people know what to do?
They will end up doing nothing
That is why you need motivated people.

The Product Owner can help to clarify the Selected Product Backlog items and make trade-offs.
This is through negotiation. There is a certain amount of capacity the team can handle. Over exceed and something suffers.
The goal is that the only thing that suffers is scope.

If the development team determines it has too much or too little work, it may re-negotiate the selected product backlog items with the Product Owner.
Too Little - add more
Too much - remove some

The development Team may also invite other people to attend in order to provide technical or domain advice.
But these people do not participate

By the end of the Sprint Planning, the Development Team should be able to explain to The Product Owner and Scrum Master how it intends to work as a self organising team to accomplish the Sprint Goal and Create the anticipated increment.

This is a feedback mechanism to let the P.O. know the Dev team understands the requirements.
It also forces the Dev team to put a strategy together on how to work.

## Sprint Goal

The Sprint Goal gives the Development Team some flexibility regarding the functionality implemented within the Sprint. The selected product backlog items deliver one coherent function, which can be the Sprint Goal.
The Sprint Goal can be any other coherence that causes the Development Team to work together rather than on separate initiatives.
The Goal could be the problem to be solved. It is basically something larger and more meaningful than just the work.
It also needs to be something that guides the developers decisions.

Sprint Goals should be S.M.A.R.T. Goals.

- Specific
- Measurable
- Attainable
- Relevant
- Time Bound.

As the Development Team works, it keeps the Sprint Goal in mind. In order to satisfy the Sprint Goal, it implements the functionality and technology. If the work turns out to be different than the Development Team expected, they collaborate with the Product Owner to negotiate the scope of the Sprint Backlog within the Sprint.

## Daily Scrum

The Daily Scrum is a 15-minute time boxed event for the Development team to synchronize activities and create a plan for the next 24 hours. This is done by inspecting the work since the last Daily Scrum and forecasting the work that could be done before the next one.

15 minutes max - regardless a team size.
It is not a status report.

The Daily Scrum is held at the same time and place every day to reduce complexity. During the meeting, the Development Team members explain:

- What did I do yesterday that helped the Development Team meet the Sprint Goal?
- What will I do today to help the Development Team meet the Sprint Goal?
- Do I see any impediment that prevents me or the Development Team from meeting the sprint Goal?

Its stupid to have everyone standing around answering these questions. Just do the work.

These questions do not specifically have to be answered one after another. The idea is that the team 'huddle' together and work out what needs to be done.

I have heard of teams that do the 'Huddle' to work out what they are going to do. Then the SM or PO or Manager comes in and then they do the 'Stand Up'. The 'Huddle' in this instance would have sufficed.

The Development Team uses the Daily Scrum to inspect progress towards the Sprint Goal and to inspect how the progress is trending toward completing the work in the Sprint Backlog. The Daily Scrum optimizes the probability that the Development Team will meet the Sprint Goal. Every Day, the Development Team should understand how it intends to work together as a self-organizing team to accomplish the Sprint Goal and to create the anticipated Increment at the end of the Sprint. The Development Team or team members often meet immediately after the Daily Scrum for Detailed Discussions, or to adapt, or replan, the rest of the Sprint's work.

This is a formal chance for the Development team to do something if the plan goes haywire. Rather than continue blindly following the plan. Do something about it.

hook - reassess - replan - implement.

The Scrum Master ensures that the Development Team has the meeting, but the Development Team is responsible for conducting the Daily Scrum. The Scrum Master teaches the Development Team to keep the Daily Scrum within the 15 minute time box.

A test to try to see if the Daily Standup is a status meeting rather than a planning meeting is to tell the Scrum Master to Piss Off. But do this after the team is ready.

Another test is if the Daily Standup does not happen if the Scrum Master/Manager is not present.

The Scrum Master enforces the rule that only Development Team members participate in the Daily Scrum.

Daily Scrums improve communication, eliminate other meetings, identify impediments to development for removal, highlight and promote quick decision making, and improve the Development Team's level of knowledge. This is a key inspect

and adapt meetings.

Do you need Daily Scrums if everyone is working on their own thing?

Daily Scrums don't work. Everyone is doing their own things, there is no need for collaboration. - Is this really a team?

## Sprint Review

A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed.
During the Sprint Review, the Scrum Team and Stakeholders collaborate about what was done in the Sprint.
Based on that and any changes to the Product Backlog during the Sprint, attendees collaborate on the next things that could be done to optimize value.
This is an informal meeting, not a status meeting, and the presentation of the increment is intended to elicit feedback and foster collaboration.

This is a formal session to get stakeholder feedback. It does not mean that you need to wait to this point to show stakeholders. You can talk to a few during the Sprint to get feedback soon for relevant items.

This is at most a four hour meeting for one month Sprints. For shorter Sprints the event is usually shorter. The Scrum Master ensures that the event takes place and that attendees understand its purpose. The Scrum Master teaches everyone involved to keep it within the time box.

Again another time boxed meeting. The Scrum master is not the one running the show. They are the event planner. They make sure the event takes place. Helps everyone, team members, product owner and stake holders understand the purpose of the event. They may help direct the event, but the content is all Product Owner and team.

The sprint review includes the following elements:
- Attendees include the Scrum Team and key stakeholders invited by the Product Owner

Key stakeholders should be those who:
- Pay for the work done so they can see where their money is going.
- Those who will use the product because they will have to live with the result and thus should have a say.
- Anyone else directly involved with the product.

You do not want people who just coming along like the whole company, you also don't want walk ins, only those invited by the product owner.
This is also not a town hall. Keep the group small.

- The Product Owner Explains what Product Backlog items have been "Done" and what has not been "Done".

Full transparency here. No hiding ~~behind~~ half done work by not showing that feature. Show everything.

Done is based on the "definition of done".

By showing what isn't done, it stakeholders get angry.

this will lead to the team 'hiding' stuff. Hiding stuff means that the Stakeholders do not see reality and thus cannot make proper decisions.

- Need man.

- The development team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved.

- More transparency. This time from the Development team. Those that do the work.

- The Development team demonstrates the work that has been "done" and answers questions about the increment.

- Show the Stakeholders the progress and also so they can see what they are giving feedback on.

Even better would be to let the Stakeholders do the demo with guidance from the Development team.

Don't try to avoid bugs. Show them so stakeholders are aware of them.

This is where the term "Showcase" comes from. It becomes a 'Show and tell'.

A Showcase alone does not help the product get better.

- The Product Owner discusses the Product Backlog as it stands. He or She projects likely target and delivery dates based on progress to date (if needed).

This is to give transparency to the Stakeholders. So they know what has been worked on.

Based on the work to date, completion of Backlog can be determined based on current progress. Such as Burn down charts.

If for example progress is slow and delivery dates could be missed, the Stakeholders can discuss dropping features that are no longer required.

Ultimately though, the final decision rests with the P.O.

Targets and Delivery dates should not be arbitrary. For example setting a date to get the team to work faster. This adds stress to the overall process. Ready to teams working take into the night to meet a false deadline only to find out that it made no difference. This becomes very demoralizing.

- The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint planning.

The key here is collaboration. Everyone gets a voice

Everyone has input.

- Review of how the Market Place or Potential Use of the product might have changed what is the most valuable thing to do next; and

~~Note~~: Things can change. The Stakeholders, PO may reject what has been done. This is always a possibility. Not only for new work, but for work already done.

- Review of the timeline, budget, potential capabilities, and Market Place for the next anticipated release. (New functionality or capability of the product.

This may include canning the project/work. If enough value has been reached, then work can stop.

This needs to be taken into account.

The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities.

This is a deliverable of the Sprint Review. It is up to the PO. now to determine to include the new items and their subsequent priority. This is just one aspect of the P.B. These items may need further grooming by the P.O. and the Scrum Team.

Everything is up for re-ordering, removal or addition. Nothing is fixed.
Potentially.
The Plan changes yet again

## Sprint Retrospective

The Sprint Retrospective is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.

This is part of the final portion of the Plan → Do → Check/Study → Act cycle.

It is a formal time when the team reviews how they have worked. The output should be a plan (or experiment) on a way to improve the current situation.
It doesn't have to be a big change, but there should be a change. The requirement for change is to prevent stagnation

The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. This is at most a three-hour meeting for one-month Sprints. For shorter sprints, the event is usually shorter.
Again time boxed to a maximum time.

Two week sprints, usually 1 1/2 hours is allocated.

The Scrum Master ensures the event takes place and that attendants understand its purpose.

This is an interesting statement. It means that the Retrospective is for the Team, by the team. It is not the Scrum masters role to "Run" the Retrospective. They merely facilitate the retro.

The Scrum Master ensures that the meeting is positive and productive.

Retro's are just a waste of time.

If the feeling by the Team as a whole is that a Retro is a

waste of time - do something about it. Discuss why it is a waste of time and look at ways to improve them.

One common mistake is to skip the Retro when it is a "waste" of time - This only "hides" the problem, it does not address the problem. Scrums purpose is to expose problems, in this case a meeting for improvements that doesn't do anything, it is up to the team to make the meeting useful. Scrum can't fix passivity.
actively

The Scrum Master teaches call to keep it within the time-box. The Scrum Master participates as a peer team member in the meeting from the accountability over the Scrum Process.

Interesting note. There is no mention of the 3 questions in a retro.

- What went well during the Sprint.
- What didn't go well during the Sprint.
- What needs improving.

While these questions are answered - they are not directly asked. The answers must be encouraged from the team.

What if people have nothing to say? They shouldn't be made to talk.

That is all well and good, but I cannot tell the difference between not having something to say and not saying anything because
- You are afraid to
  - Intimidated not to say something (not a safe environment)
  - Think no-one cares about your views
  - You're pissed off.
  - etc.

The only way to be sure is encouragement of participation to bring issues out in the open and make sure it is safe to do so.

The Purpose of the Sprint Retrospective is to:

- Inspect how the last Sprint went with regards to people, relationships, process and tools.

This isn't a time to go through the work output itself, but how the work was done.

- How the team worked together, if well
- Any experiments on how to work. Ie making work visible, limiting WIP. Working from home. etc.

Hansei - Acknowledge ones mistakes and pledge

- Identify and order the major items that went well and potential improvements.
  - What did we do well we don't want to forget.
  - What did well do well that we can exploit to improve other aspects of the process.

- Create a plan for implementing improvements to the way the Scrum Team does its work.

This I think is the most missed out thing with Retrospectives Planning the improvement. If you do not plan to improve, the Retrospective becomes useless. Its all well and good pointing out areas that need improvement, but not doing the improvements Guarantees failure.

If you do plan, plan it as an experiments. Have a hypothesis, determine the expected result and actually see if the result is reached. Determine a timeline to review. It could be the next day, week sprint, month. But you need

that reviews.

The Scrum Master encourages the Scrum Team to improve, within the Scrum Process framework, its development process and practices to make it more effective and enjoyable for the next Sprint.

This gives you permission to make things better. Challenge the Status Quo and have fun doing it.

This gives permission for those in "death" Scrum to challenge their situation. I know I don't find "death marches" fun. Maybe you do? But I don't.

Change things up to make them enjoyable. If it's not, you are doing Scrum wrong.

During The Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adopting the definition of "Done", if appropriate and not in conflict with Product or organization standards.

This is self inspection of the work process, the Sprint Review covers the improvement of the work output.

By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that they will implement in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself.

Self Improvement.

Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on the inspection and adaptation.

I don't like the statement "Improvements may be implemented at any time". This can insinuate that they be put off due to "other" work and thus never be implemented. I think an additional statement that improvements should be "tested" as soon as possible. I realize that improvements may need to be put off, but additional

"as soon as possible" means they should not sit on the back log forever.
Also, by specifying "tested" acknowledges the fact that an improvement may not actually make an improvement. Thus testing, even a small test will help verify this before implementing a large unwieldly improvements.

## Product Backlog

The Product Backlog is an ordered list of everything that is known to be needed in the Product. It is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.

The Product Backlog is just a list of items. That list doesn't have to be user stories. It also doesn't have to be in Jira.

It can be in excel, cards, a word document, hand written on paper. It isn't specified. The practices of how the Backlog is kept is independant of Scrum.

It is the single source of requirements. That means that if someone wants to do something, it must go on the backlog, and since the Product Owner is in charge of the Backlog, it must go on with their blessing.
That means that even if the CEO wants something done, they don't go to a developer directly, they go to the Product Owner. This means that the PO is a position of respect. Not someone who is a managers lackey.

The list also needs to be ordered so that when the list is looked at during planning, or if the dev team finishes the Sprint Backlog early it is easily known what is the most valuable work first.

What if everything is of equal value. I've had that before.

Then nothing is valuable. This is an antipattern.

the PO wants everything, so everything is of equal value.

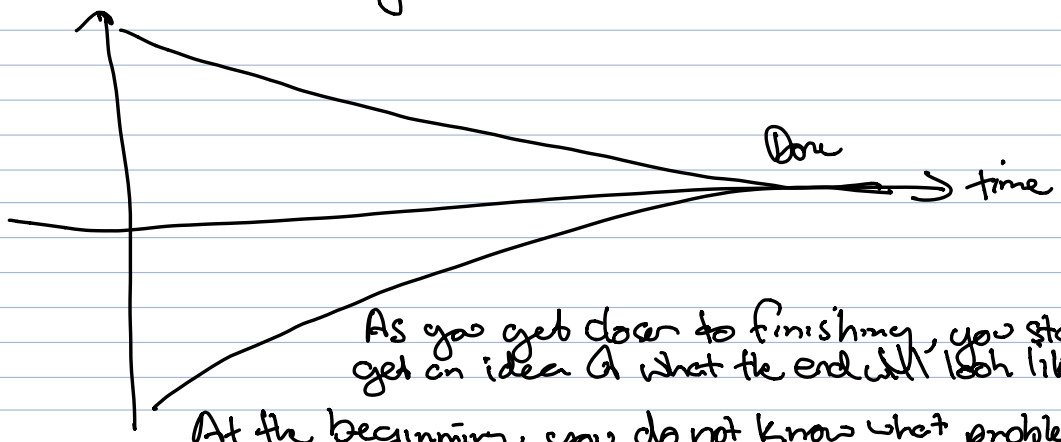
This is then used to rush developers. Which leads to shoddy work as the developers try to complete everything in the Sprint. Which leads to Tech debt and problems in the future.

There is always something of higher importance/value. This needs to go first. Then the next and so forth.

A Product Backlog is never complete. The earliest development of it lays out the initial known and best-understood requirements. The Product Backlog evolves as the Product and the environment in which it will be used evolves. The Product Backlog is dynamic; it constantly changes to identify what the product needs to be appropriate, competetive, and useful. If a Product exists, its Product Backlog also exists.

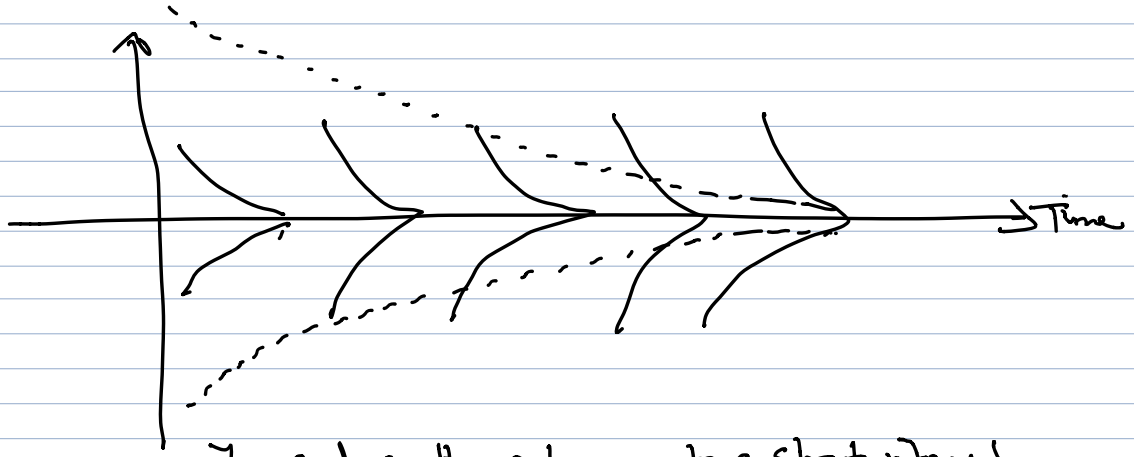
When you first start, you do not need to work out everything initially. It will all change anyway so new things are learned. You cannot know everything up front. What you know initially, start working on. Work out the rest later. The later you can delay decisions, the more time you give it to change. So that when it gets time to do the work, hopefully - not always - the changes will be minimal.

The Cone of uncertainty.




As you get closer to finishing, you start to get an idea of what the end will look like.

At the beginning, you do not know what problems you will face, therefore only work to small goals.




You reduce the unknown to a short interval. Deliver to that interval and at least that small section is known. Yes it can change, but you are delivering value sooner.

This is when the Product Backlog should be dynamic

Even though the 'Project' is complete. Changes, bugs, new features, enhancements will still exists. These are stored in the Product backlog so they can be worked on at a later date, when there is more time or budget.

The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the Product in future releases. Product Backlog items have the attributes of a description, order, estimate, and value. Product Backlog items often include test descriptions that will prove its completeness when "Done".

The Product Backlog contains everything needed for the Product. This includes DB, Servers and other non-functional as well as functional requirements.
Each item should have a:

- Description: These tend to be in the form of stories.

As a ____
I want ____
So That ____!

but they do not have to be. They can be a header, or a paragraph. A video, drawing. The important thing is that they have context to how the item fits in, and convey enough info to know what the work is.

- Order - Generally what is needed and most valuable at the top. Least valuable at the bottom. Make it easy for the team to know what to work on without asking. Also think, if you were to run out of money in the next sprint - what would you want most.
Just don't like mentioned above as it for everything.

- Estimate - So you have a guide as to how long or how much effort the item will take.
Note: Scrum does not specify the method of estimation. You can use story points, t shirt sizes, assigned sizes or done / story / time.
There are reasons why you should not use time. I will go through them here.

- Value - This could be in the form of ROI, Time saved etc. This helps give context as to why the work is done.
Some companies include ROI, even in non-scrum projects.
The problem is that at the end, the ROI is never verified.
You never know if the value is realized or not.
This is not good.

- Test Descriptions - These help the Developer know that they have built the right thing.
Again there are many forms, but the more details included the better an understanding the Developer has.
Tools like Cucumber, Fitnesse, Concordion can help automate the verifications.

As the Product is used and gains value, and the marketplace provides feedback, the Product Backlog becomes a larger and more exhaustive list. Requirements never stop changing, so a Product Backlog is a living artifact. Changes in business requirements, market conditions, or technology may cause changes in the Product Backlog.

The only way to know if the Product will be useful is when it is used. Real hands on. Not labs, not trials, but real people using the product in anger.
You will then know its shortfalls. The fixes, changes, extra functionality then gets added to the backlog.

A 'Product' is a long lived thing unlike a 'Project'. Requirements evolve, there is no end until the product dies for whatever reason.

Multiple Scrum Teams often work together on the same product. One Product Backlog is used to describe the upcoming work on the Product. A Product Backlog attribute that groups items may be employed.

One Product Backlog. One source of value. Everyone works together to get the most value sooner.

Product Backlog refinement is the act of adding detail, estimates and order to items in the Product Backlog. This is an ongoing process in which the Product Owner and the Development Teams collaborate on the details of the Product Backlog items.
During Product Backlog refinement, items are reviewed and revised. The Scrum Team decides how and when refinement is done. Refinement usually consumes no more than 10% of the capacity of the Development Teams.
However, Product Backlog items can be updated at any time by the Product Owner or at the Product Owners discretion.

Product Backlog refinement is where you go through items in more detail. Break down the items to smaller pieces of work. Get Acceptance criteria added. Work out in detail what needs to be done for each item. Not too much, but just enough. How much is enough? You have to work that out

by trial and error. There are no easy rules.

Refinement should not take more than 10% for a two week Sprint, that is 1 day. It could be done all in one day or one hour each day over the Sprint.

Do not take more than 10% of the time, but you can take less.

More meetings, when do you get the work done?

Refinement is done anyway. It's just more hectic and ad hoc when you don't specify a time.

This is a requirements gathering process. Rather than spend a whole heap of time up front, you are doing this during the development.

For Developers, this may seem to be an issue. BA's do the requirements gathering. Put them together and hand them over to the Developers for coding.

In Scrum, There are no BA roles. You are the BA, the Developer, the tester. Look at it as an opportunity to learn. Not a "Not my job" situation.

Higher order Product Backlog items are usually clearer and more detailed than lower ordered ones. More precise estimates are made based on the greater clarity and increased detail; the lower the order, the less detail. Product Backlog items that will occupy the Development Team for the upcoming Sprint are refined so that any one item can be reasonably "Done" within the Sprint time-box. Product Backlog items can be "Done" by the Development Team within one Sprint are deemed "Ready" for selection in a Sprint Planning. Product Backlog items usually acquire this degree of Transparency through the above described refining activities.

The closer you are going to work on an item, it needs to have more detail. This makes sense. Why would you add more detail to an item you won't look at for some time, while neglecting upcoming work?

Get the items to a point you can start and finish the work in a Sprint. Don't do half the work now and the other half next Sprint. Treat the work as if this Sprint is your last. Any subsequent Sprints (even planned) is a bonus. At least during the planning.

There is a definition of "Done" which we will get to later, but

Should there be a Definition of "Ready"?

The Development Team is responsible for all estimates. The Product Owner may influence the Development team by helping understand and select trade-offs, but the people who will perform the work make the final estimate.

Have you ever been in the situation where a manager gives you a piece of work, expects it to be done in a few days, you look at it and it's a few months?

This statement is to give Devs the ability to say it will take a few months. Not a few days.
They should know, they are doing the work.

## Monitoring Progress Towards Goals

At any point in time, the total work remaining to reach a goal can be summed. The Product Owner tracks this total work remaining at least every Sprint Review. The Product Owner compares this amount with the work remaining at previous Sprint Reviews to assess progress toward completing the projected work by the desired time for the goal. This information is made transparent to all stakeholders.

This is giving the description of the Burndown chart, without explicitly saying Burndown chart because it limits thinking of other ways to describe the data.

This is basically the sum of all work remaining (estimates) compared to the previous sum. The difference gives the rate. Based on the rate, the expected completion of the work can be estimated.

Various projective practices upon trending have been used to forecast progress, like burndowns, burn-ups, or cumulative flows. These have proven useful. However, these do not replace the importance of empiricism. In complex environments, what will happen is unknown. Only what has already happened may be used for forward-looking decision-making.

Various examples are given to represent the data. What this data shows is a rate over time.
Think of this rate as sharing your speed on the odometer over time. Your speed on a journey changes.
If you are on a freeway, your speed is 100 kph. On a suburban road it can drop to 50 kph. School zone, 40 kph. This is the known limits and estimates can be derived. but what happens if there is an accident on the

road, road works, traffic, a special event? Your rate changes, your estimated time to commit (delivery) changes.

These need to be taken into account - but often they occur. A good GPS system will do this for a journey. You have to do the same thing for your journey in doing the work.

## Sprint Backlog

The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product increment and realizing the Sprint Goal. The Sprint Backlog is a forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a "Done" Increment.

A Sprint backlog is:

- Selected Items
- A Plan for delivery of the Increment and the Goal

It is a forecast not a contract.

May have been in old Scrum Guides.

I have heard Scrum Masters mention Commitment to the work. This language is dangerous. It means Hell or high water this amount of work will be done. ↖ is taken as

This can lead to late nights, long hours, working weekends to complete an ill conceived Sprint Backlog. This is not what we want. A Sprint is a timebox - not a deadline and we are trying to measure the 'rate' of work at a reasonable pace.

Not "Get all this stuff done before the Sprint Deadline".

We can get booked for speeding on the road. Scrum Masters should be booked for "Speeding" their teams (driving them too hard) otherwise just like an engine, they will burn out.

The Sprint Backlog makes visible all the work that the Development Team identifies as necessary to meet the Sprint Goal. To ensure continuous improvement, it includes one high priority process improvement identified in the previous Retrospective meeting.

This last statement of including at least one improvement is key to getting better. Unfortunately many teams do not do this, or do not see it as a priority. Thus it gets dropped etc.

High Performance teams already do this as mentioned by Ken and Jeff when this version of the Scrum guide was released.

The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum. The Development Team modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint. This emergence occurs as the Development Team works through the plan and learns more about the work needed to achieve the Sprint Goal.

Unlike waterfall, with Scrum you do not work out everything up front. You are not supposed to. How many times have you worked on something only to find a detail that you forgot, didn't know about, had a problem you could not solve and need to figure out a workaround?
Scrum takes these things as fact. Shit happens. So don't plan too far ahead because it is all going to change anyway. Just plan enough to get by for now and a little ahead in the future. We will learn as we go.

As new work is required, the Development Team adds it to the Sprint Backlog. As work is performed or completed, the estimated remaining work is updated. When elements of the plan are deemed unnecessary, they are removed. Only the Development Team can change the Sprint Backlog during the Sprint. The Sprint Backlog is a highly visible, real-time picture of the work that the Developers Team plans to accomplish during the Sprint, and it belongs solely to the Development Team.

As the Developers team works, things missed are added to the Sprint Backlog if it can be absorbed into the Sprint, without overwork. ie working longer hours.

If it can't be absorbed into the Sprint, then it either goes into the Product Backlog, or something drops out in the Sprint Backlog back into the Product Backlog

One Thing about agile is that we try to minimize the work done. If something is no longer needed - drop it. don't do it anymore. Makes sense right? It's amazing how much work is done that is not required.

The Sprint Backlog should be visible to all. This is why you can see Scrum Boards, it's a visible way to see the work done.

By making it visible - everyone can see the progress. Nothing is hidden.

## Monitoring Sprint Progress

At any point in time in a Sprint, the total work remaining in the Sprint Backlog can be summed. The Development Team tracks this total work remaining at least for every Daily Scrum to project the likelyhood of achieving the Sprint Goal. By tracking the remaining work throughout the Sprint, the Development Team can manage its progress.

Much like the Product Backlog, remaining work can be tracked checked and determination of completing done.
This should be done at least daily, during the daily Scrum.

This statement again disproves the "What I did yesterday, what I will do today, any blocks" statements.

These can help determine progress but they are not the only things done during the Daily Scrum.

Again this emphasizes that this is a responsibility of the Development team. The Development Team is masters of their domain.

## Increment

The Increment is the sum of all the Product Backlog items completed during the Sprint and the value of the increments of all previous Sprints. At the end of a Sprint, the new increment must be "Done" which means it must be in usable condition and meet the Scrum Team's definition of "Done". An Increment is a body of inspectable, done work that supports empiricism at the end of the Sprint. The increment is a step toward a vision or goal. The increment must be in usable condition regardless of whether the Product Owner decides to release it.

Basically an increment is all the work done to date that have been fully completed and in a state that the Product Owner will find useful to some degree.

By Definition of 'done' which is discussed later. It must be code reviewed, have error handling, follow standards, be tested and be to a point ready for release'. By doing this up front - even for work that is not in a fully complete state - ie still has many months left to bring it to full spec, you -

- Get feedback sooner. If left till fully ready after months, without checking, you run the risk of not making what the client needs or wants.
- If budget runs out, the client still has something usable. Even after the first 2 weeks.
- It forces you to think in smaller chunks of value rather than just going for the whole thing.

The Product Owner is the best person to know when to release as they know when the product will have the most value to users and Stakeholders.
By having the increments in a "Done" state, there is no reliance on fixing "Bugs" or any other trepidation before release as they should have been looked at beforehand.

## Artifact Transparency

Scrum relies on transparency. Decisions to optimize value and control risk are made based on the perceived state of the artifacts. To the extent that transparency is complete, these decisions have a sound basis. To the extent that the artifacts are incompletely transparent, these decisions are flawed, value may diminish and risks may increase.

Have you ever worked on a project where you know that the whole thing will be a mess, you tell the Project Manager that it will last 6 to 8 weeks before it completely fails, but they release it anyway because the PM told management everything was good?

Ever been on a project where everything was green up till a week before release?

These problems occur because information is hidden. It usually ends badly and it occurs so frequently that it is considered normal.

Scrum relies on everyone knowing everything, and by everything, I mean the correct info. No lying or bending the truth.
That way good decisions can be made.

But for this to happen, there must be an environment where it can happen without repercussions.

i.e. There must be TRUST

The Scrum Master must work with the Product Owner, Development Team, and other involved parties to understand if the artifacts are completely transparent. There are practices for coping with incomplete transparency; the Scrum Master must help everyone apply the most appropriate practices in the absence of complete transparency.
A Scrum master can detect incomplete transparency by inspecting the artifacts, sensing patterns, listening closely to what is being said, and detecting differences between the expected and real results.

It is the Scrum Masters job to try to teach the Product Owner and the Development Team to be transparent.

Some technologies are
- Kanban board
  - To visualize work
- Gather Metrics
  - length of time of a ticket, item not worked on or
- Retrasprtive Techniques

Sometimes by just listening to the team the Scrum Master can get a sense that something is wrong.

For example, mumbles about work, management.
Hiding stuff from the product Owner.

The Scrum Master must help the team expose these so they can be addressed.

The Scrum Masters job is to work with the Scrum Team and the organization to increase the transparency of artifacts.
The work usually involves learning, convincing, and change.
Transparency doesn't occur overnight, but is a path.

The job of a Scrum Master is not an easy one, if done right.

There is so much resistance to true agile. It is much easier to follow the "Process" and be ignorant of the reasons as agile shows problems. It is far easier to hide and ignore those problems.

The road to transparency is long and arduous. It is a journey with out end, which is fine as the journey is what is important - not the destination.

# Definition Of Done

When a Product Backlog item or an increment is described as "Done", everyone must understand what "Done" means. Although this may vary significantly per Scrum Team, members must have a shared understanding of what it means for work to be complete, to ensure transparency. This is the definition of "Done" for the Scrum Team and is used to assess when work is complete on the Product Increment.

Have you ever thought something was done only to find out that there was a requirements, or a portion of the work that needed to be done to complete the piece?
Have you built something only to find out that it wasn't what the customer or end user had in mind and have to start all over again?

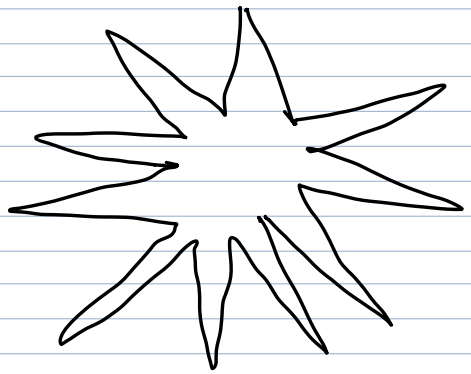
This is what the "definition of done" is for. To make sure everyone is on the same page, has a shared understanding of what needs to be done to complete the work.

Because everyone has their own understanding of a "requirement", the customers understanding needs to be "extracted" from the customers brain and "implanted" into the development team. My favorite method of doing this is through examples.

Stealing from Gojko Adzic's book "Specification By Example"

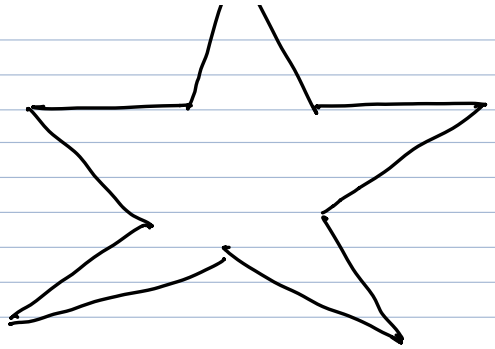
Lets say that the customer has asked for a "10 point star".
A simple requirement.

So... You produce the following...



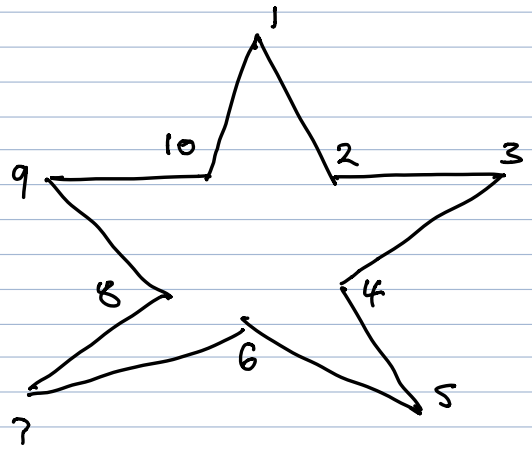
Is this correct? Well, that's up to the customer. In this case the customer says "NO". Go back and try again.

Reaching your brain, you have no idea what the customer is looking for. So you ask them to sketch it out.



Huh! Thats a 5 point star!

No, says the customer. It is 10 points!




Ah! you now say. I know what you are thinking!

A simple example of what is expected can enlighten all on what is required.

This will then reduce re-work and mistakes based on misunderstandings.

Other things can include having a list of what needs to be done. Such as -

- Tested with Automated tests. - follows coding standards
- Documentation complete.
- Meets acceptance Criteria
- Code reviewed
- No known defects
- User Acceptance tested

even migrated to production

The same definition guides the Development Team in knowing how many Product Backlog items it can select during a Sprint Planning. The purpose of each Sprint is to deliver increments of potentially releasable functionality that adhere to the Scrum Team's current definition of "Done"

This means that you select items from the backlog that produce some functionality that is useful.

Not half of a piece of work one sprint and the other half the next sprint.

"There is no ~~Sprint~~ next sprint"

The next sprint should be considered a bonus that is not guaranteed. Therefore select items accordingly. Reduce scope if need be.

Development Teams deliver an Increment of product functionality every Sprint. This Increment is useable, so a Product Owner may choose to immediately release it. If the definition of "Done" for an increment is part of the conventions, standards or guidelines of the development organization, all Scrum Teams must follow it as a minimum.

If the Product Owner chooses to "release" the 'functioning' product, this means that value is returned 'sooner' than if the project was complete.

It also means that any learning from the product in production can be done sooner, thus determining if the work was valuable or not. If it is not, stop working on that feature and try something else. If it does - continue working on the feature.

If the Product is "good enough" then the Product Owner may choose to not continue with the rest of the project.

This is known as "chopping the tail". There is no use continuing on all required functionality is already delivered.

This is another reason - from a customers perspective to get things in use by the users asap.

Also, if the company has existing standards, incorporate them in the Definition of Done, but review them regularly. You may find some no longer relevant or appropriate. A good place to look at this is in a Retrospective, but that should

not be the only place. You can do it during the Daily Scrum, planning sessions etc.

Inspect and adapt.

If "Done" for an increment is not a convention of the development Organization, the Development Team of the Scrum Team must define a definition of "Done" appropriate for the product. If there are multiple Scrum Teams working on the system or product release, the Development Teams on all Scrum Teams must mutually define a definition of "Done"

If there are no company wide or team wide conventions for Done, make your own. Call out what you want to happen to make sure you make a quality product.

Make sure that all Scrum teams that work on the same product share the same quality values and definition of done.
If you do not do this you will find parts of the product in non working order or of dubious quality.

Its just common sense.

Get the Teams to decide their DoD. Don't dictate it to them (unless it is company conventions). If they define their own DoD, they have buy in. They may get more energised.

They may care more about meeting the DoD.

Each increment is additive to all prior increments and thoroughly tested, ensuring that all increments work together.

Each sprints output builds on previous Sprints. This means that you need to make sure that past components developed still work. This means regression testing.

Agile won't work easily with manual procedures. There just isn't enough time. Therefore you need to automate.

- Automated Regression testing.
- Automated Deployment

Reduce the time it takes to do stuff.

But don't do it up front. Do it along the journey. That way you only automate what you need to.

Keep doing the improvement. That is the whole point.

as you go along.

As Scrum Teams mature, it is expected that their definition of "Done" will expand to include more stringent criteria for higher quality. New definitions, as used, may uncover work to be done in previously "Done" increments. Any one product or system should have a definition of "Done" that is standard for any work done on it.

The goal of Scrum is continious improvement - not to keep the Scrum pace, the Scrum steady run. As you get better and stretch your abilities, your definition of Done should also evolve.
This may include:
- Security goals
- Reviewed by Legal
- Penetration testing
- Performance testing.
- Continuous Testing/Integration/Delivery

and any other criteria that you may wish to include.
You can't add this sort of thing until you get better, faster and more 'agile'. Yes, I mean agile in the dictionary meaning - not as in the Process or Methodology.

## End Note

Scrum is free and offered in this Guide. Scrum's roles, events, artifacts, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum. Scrum exists only in its entirety and functions well as a container for other techniques, methodologies, and practices.

Ok, for those that complain that you need certifications to do Scrum. To be a Scrum Master here is the proof that you do not need any of these things.
Scrum is free!
I remember about 20 years ago, you had to pay to use the Rational Process to develop software. This is not the case with Scrum.

Also those places that do "Scrum" but do not do Retrospectives or Daily Scrums/Stand Ups - you are NOT doing Scrum.

Saying that, you can still not be doing Scrum if you follow the "Process" to the letter but ignore the underlying reasons for the process. Which basically is to get better. You get better by finding and fixing problems. Not by blindly following the process. The process is there to guide improvement actively over a Sprint, Not organically as required whenever or every now and then through workshops.

Remember the beginning. Scrum is a framework on which to build improvement. Techniques such as Scrum Boards, User Stories, Scrum Poker, Story points are not part of Scrum but still can be used. This is known as Scrum-ish and as opposed to Scrum-but where parts of Scrum are removed.