VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SWE 2029 – Agile Development Process

Module – 6

AGILE PLANNING and ESTIMATION

Dr. Rajesh M
School of Computer Science and Engineering
Vellore Institute of Technology [VIT]
Chennai, India.

Module	Topics	L Hrs
VI	AGILE PLANNING and ESTIMATION : Principles of Agile Metrics – Release, Planning and Estimation in Scrum.	5

AGILE PLANNING and ESTIMATION :

Introduction

- Planning and estimation are two of the main factors that can influence the **outcome of any project.**
- So with the wide adaption of agile methodologies in the IT world, questioning the effectiveness of the ordinary style of planning and estimation is starting to rise.
- Together we will try to explore the world of agile planning and estimation, to offer a better understanding of this issue and hopefully to be **able to decide if it is the right thing to do for your organization or not.**

AGILE PLANNING and ESTIMATION :

Introduction

- The agile project management by default **requires a specific kind of planning.**
- Planning that is **flexible enough to handle the nature of the agile environment.**
- Due to its **fluid nature**, it might look like it's non-existent to the untrained eye.
- This is probably what led to the old myth of **"There's no planning in agile".**

AGILE PLANNING and ESTIMATION :

Introduction

- Agile planning like other types of planning needs estimating and measuring tools and methods to stay alive.

- However in spite of the fact that a lot of the general estimating and measuring rules and tools which are used in ordinary (non-agile) projects can be applied to agile projects (with some modifications to its style), its implications can vary and its implementations are different.

AGILE PLANNING and ESTIMATION :

Introduction

- So how is the agile planning done?
- And how does it estimate assigned projects?
- There is no one answer for this, because although there are a lot of the shared ideas and concepts between the different agile methodologies, **each one has its own distinct style of applying them.**
- For this, **concentrate on the general ideas shared by most of the agile methodologies**, like Scrum and Lean methodologies.

AGILE PLANNING and ESTIMATION :

Introduction

Planning

- Why Agile Planning?

- As Mike Cohn defines it, "**Agile planning balances the effort and investment in planning with the knowledge that we will revise the plan through the course of the project. An agile plan is one that we are not only willing, but also eager to change** "

- This concept exists mainly to avoid the weakness of the ordinary (non-agile) planning.

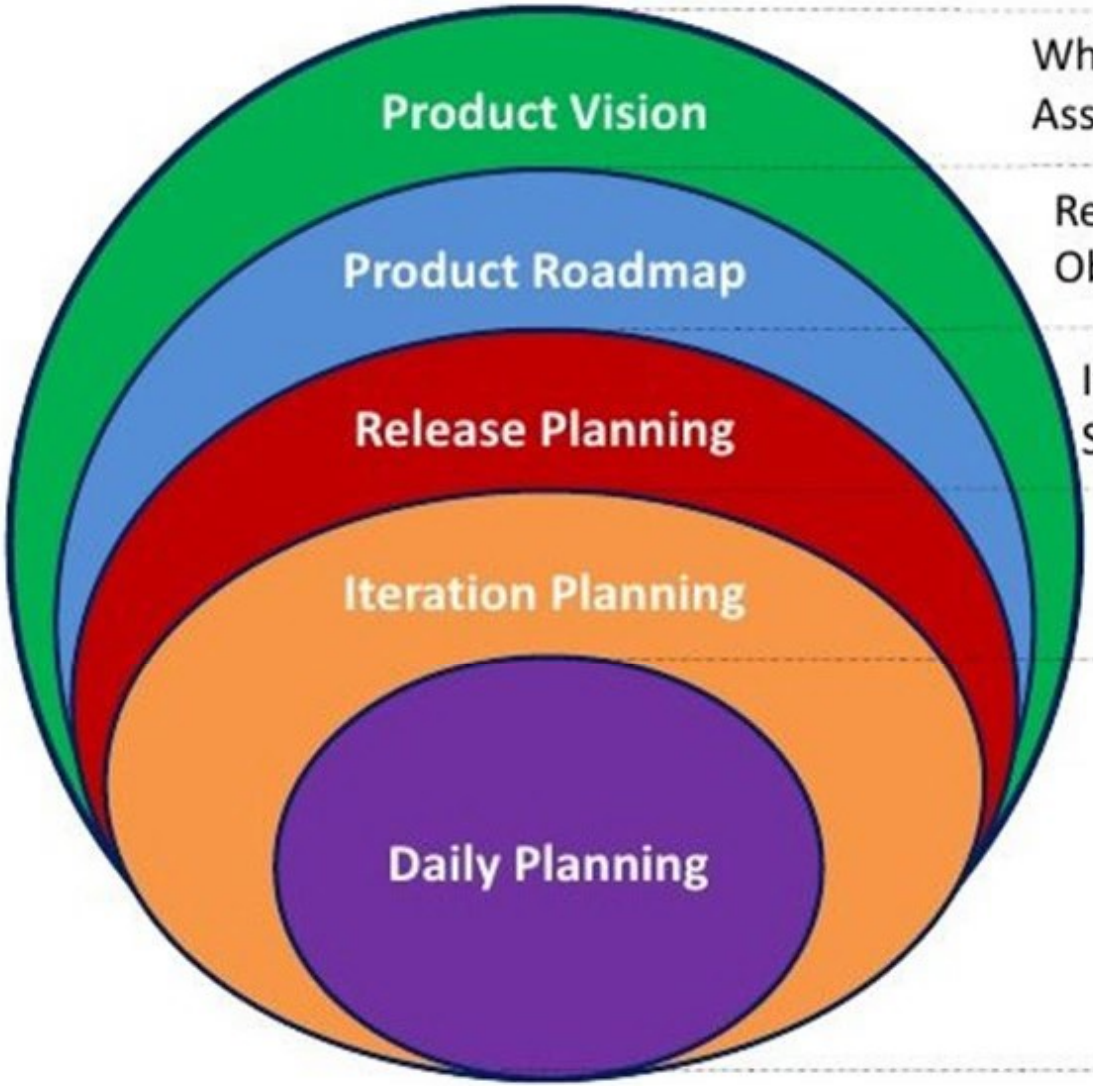
Explore Agile Planning & Estimation

These **weaknesses** include:

- **Concentrating on activities** rather than delivered features
- **Ignoring the prioritization**
- **Ignoring the existence of uncertainty**
- Using estimations as commitments

- Such weaknesses are what made the ordinary (non-agile) planning not able to keep up with the agile projects and their dynamic environments. For that the planning has to evolve to become agile as well.

5 levels of Agile Planning



What, Who, Why, When, Constraints, Assumptions

Releases – Date, Theme/Feature Set, Objective, Development Approach

Iteration, Team Capacity, Stories, Priority, Size, Estimates, Definition of Done

Stories – Tasks, Definition of Done Level-of Effort, Commitment

1. What did I do yesterday?
2. What will I do today?
3. What is blocking me?

Explore Agile Planning & Estimation

Agile Planning Levels:

- Agile planning needs to be done on **5 different and integrated levels** of focus.

They are :

- **Product Vision Planning**,
- **Product Roadmap Planning**,
- **Release Planning**,
- **Iteration Planning**, and
- **Daily Commitment Planning**.

Explore Agile Planning & Estimation

Product Vision Planning :

- The highest level of planning that **concentrates on the future of the product and its ultimate goal** with the possible changes to them and their likelihood.
- This creates the macro image of the product's world and its location in the universe of the stakeholders, allowing us to establish the general line of priorities and their estimations.

Plan shaping done by:

- **99% Product Owner** & his team
- **1% Project Manager** & his team

Explore Agile Planning & Estimation

Product Roadmap Planning :

- After knowing what the product world shall look like, the next step would be planning on partitioning this world into separate buildings and **lining them on the road which we will follow to build the product's world**.
- This allows us to estimate the releases with respect to the general functionalities and time needed.

Plan shaping done by:

- **70% Product Owner** & his team
- **30% Project Manager** & his team

Explore Agile Planning & Estimation

Release Planning :

- After knowing all the buildings of the product's world, we start planning how each of these buildings will look like.
- Each of these buildings is a deliverable part of a value that is usable by the stakeholders. This allows us to **estimate the iterations of each release and how they are related**.

Plan shaping done by:

- **50% Product Owner** & his team
- **50% Project Manager** & his team

Explore Agile Planning & Estimation

Iteration Planning :

- After deciding on how the assigned building in the product's world will look like, we start planning the required floors of each building and how each floor will look like from inside (for example, how many rooms and bathrooms).
- This allows us to be **more specific about the assignment of the iterations with respect to the individual teams' abilities and experience.**

Plan shaping done by:

- **30% Product Owner** & his team
- **70% Project Manager** & his team

Explore Agile Planning & Estimation

Daily Commitment Planning :

- So after knowing which part of the floor your team is building, we **start planning on the steps that we will follow in finishing the assigned rooms** (for example constructing the walls, windows, doors, etc).
- This will help us get more exact in our expectations and projections.

Plan shaping done by:

- **1% Product Owner** & his team
- **99% Project Manager** & his team

Agile Planning & Estimation

Agile Planning Life Cycle :

Agile planning: 2 months iteration

#1 Iteration - Week 1-2	Owner	Status	Priority	Deadline
Review Dev Environment		Done	Medium	Jan 6
Review Production Environment		Done	Must Have	Jan 9
Plan Release		For review	High	Jan 11
Analyse Deployment's Data		Ongoing	Low	Jan 14

#2 Iteration - Week 3-4	Assignee	Status	Priority	Estimation
Release Plan Review		For review	Medium	Jan 19
Build Release		Done	Low	Jan 23
Test Plan Approval		For review	High	Jan 16
Test User Acceptance		Ongoing	Low	Jan 26

Explore Agile Planning & Estimation
Agile Planning Life Cycle :

Transparency :

- Agile methodologies by their nature are about **minimizing the gap between the business world and the technical world**.
- Agile planning is no different. **Transparency is essential** to make all stakeholders **understand the result of their decisions** and involvement.

Understand Current Situation :

- **Your project is the world you are creating**. However as any other world, it has to exist in a universe; a universe which you don't have much control over. **So do not assume what your universe will look like, go and see it yourself to understand it.**

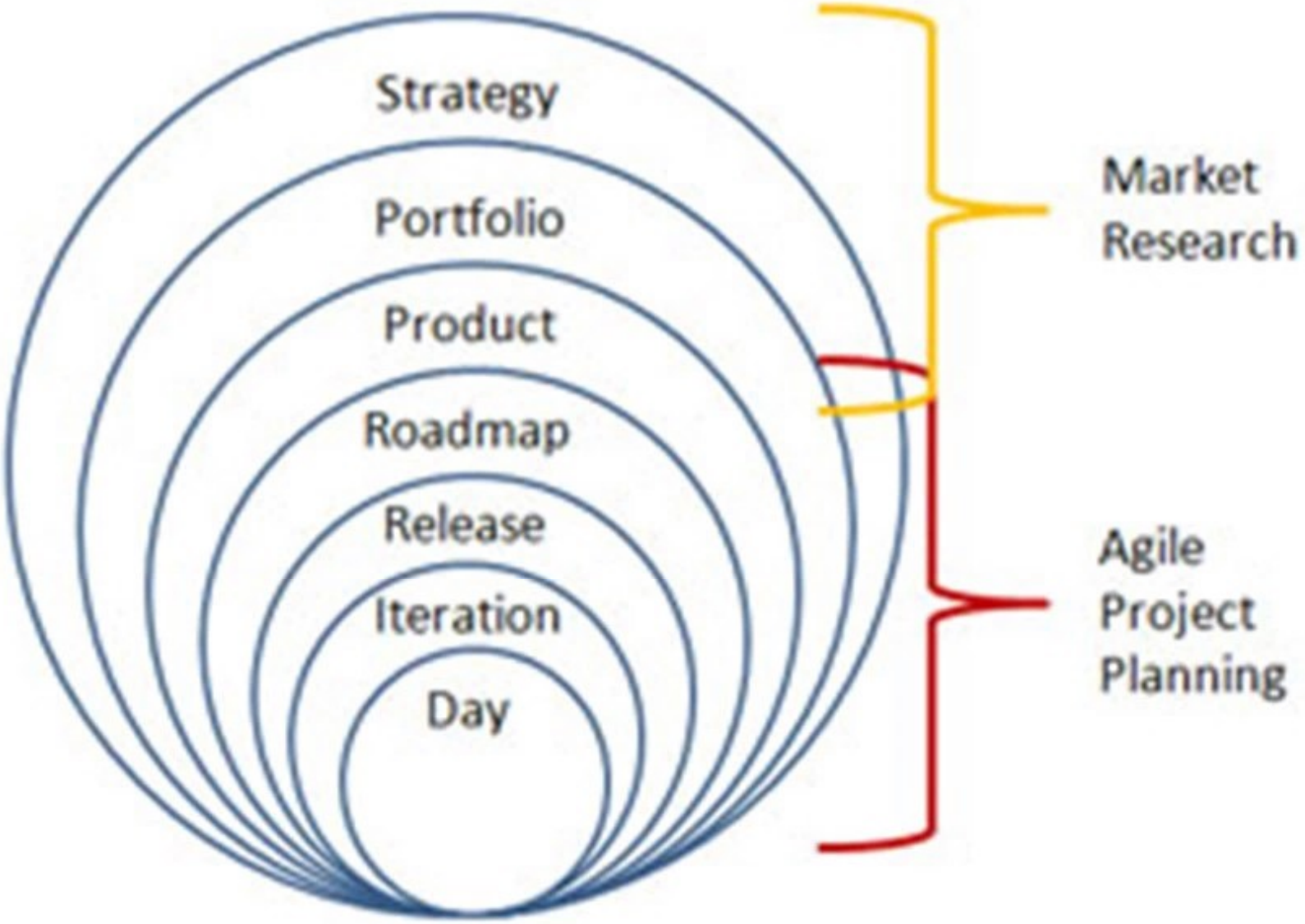
Explore Agile Planning & Estimation
Agile Planning Life Cycle :

There is always a place for improvement :

- **Your project plan is a living creature**. As any other living creature it will not stop growing, changing, and improving till it ceases to exist.

- So your planning is not done when the project's execution starts, you **have to continually improve it.**

Agile Planning Onion



Agile Planning & Estimation

Estimation :

Explore Agile Planning & Estimation
Estimation :

Estimation Projection :

Estimations in general are mostly driven from one or more of the following **techniques**:

- **Expert Opinions:** Through ***asking people with previous experience*** on the assigned tasks, estimation can be reached.
- **Analogies:** Through ***comparing known and unknown tasks*** to each other, we can find a **relative scale for estimation**.
- **Disaggregation:** Through ***breaking complicated tasks into more familiar and smaller ones,*** we can have more accurate estimation.

Explore Agile Planning & Estimation
Estimation :

Estimation Validation :

- Estimations are meant to be changed, so do get attached to them. Lean Startup encourages this through its "**Innovation Accounting**" which is a method for validating your estimations. This method is a loop of **3 steps:**
  1. **Establish real data** through creating the minimum viable product. This will allow you to see how realistic your projections are.
  2. **Enhance your viable product** to become closer to your projections.
  3. **"Pivot or preserve" your projections**. Depending on the results, you have to decide on keeping the projections as valid or changing them accordingly.

Explore Agile Planning & Estimation

Estimation :

Using Agile Planning & Estimation :

- So going back to our question, is agile planning and estimation suitable for you? It **depends on the project's agility level.**
- **Most projects usually are neither 100% agile nor 100% waterfall** (non-agile), but they **sit somewhere in the middle.**
- This goes for the agility level of planning and estimation too.

Agile estimation techniques: Planning Poker

Explore Agile Planning & Estimation
Estimation :

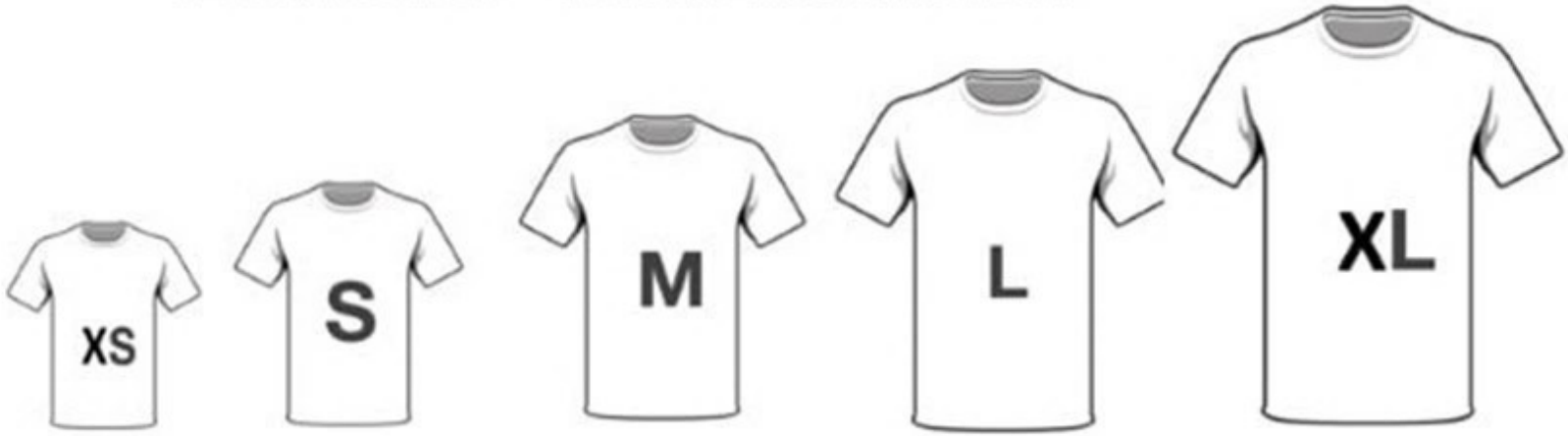
Agile estimation techniques:

Planning Poker

- All participants use **numbered playing cards** and estimate the items. **Voting** is done anonymous and discussion is raised when there are large differences. **Voting is repeated till the whole team reached** consensus about the **accurate estimation**. Planning poker works well when you have to estimate a relative **small number of items (max 10) in a small team (5-8 people).**
- **Tip:** try to keep the voting between affordable numbers. Maximize the **highest card to 13 points.**

Agile estimation techniques: T-Shirt Sizes

T-Shirt Size – Effort Estimation



T-Shirt Size	XS	S	M	L	XL
Estimate	1	2	3	5	8

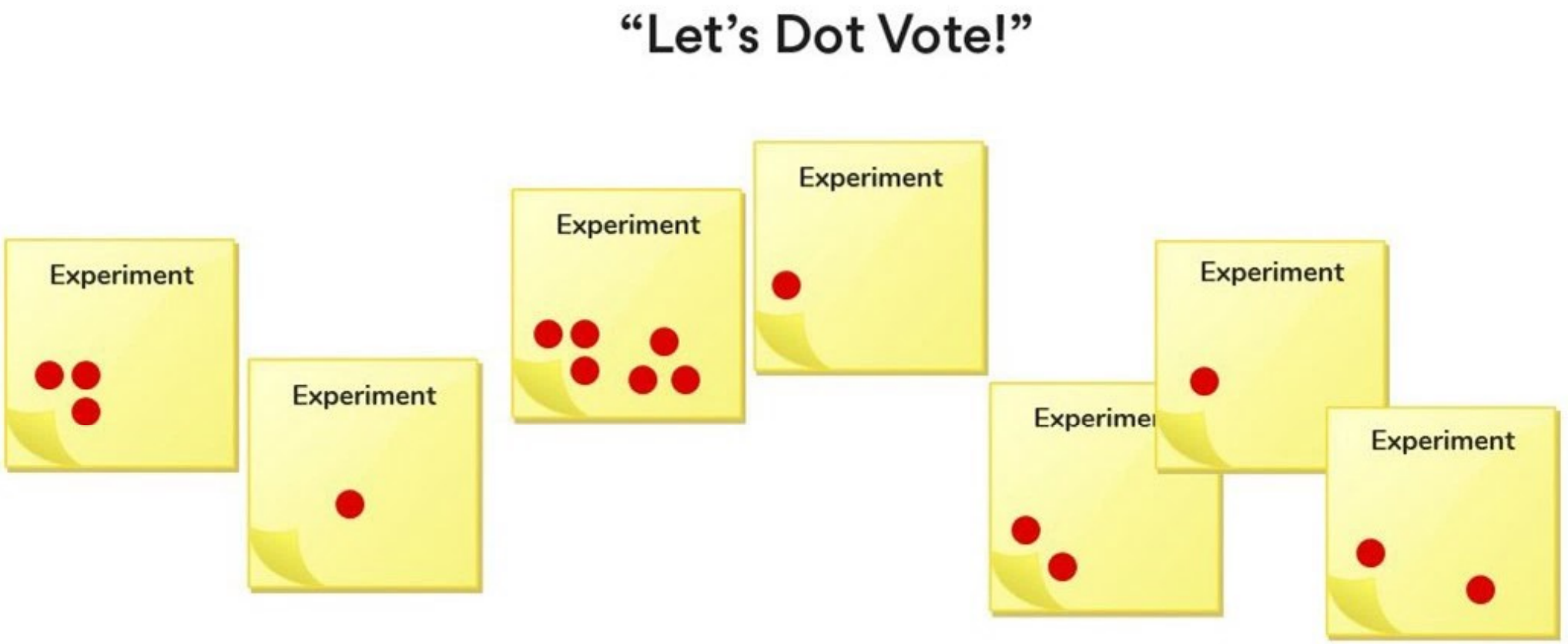
Source: PMTips.xyz

Agile estimation techniques: T-Shirt Sizes

- This is a **perfect technique for estimating a large backlog** of relative large items. Especially when you have several concurrent scrum teams working on the same product. Items are estimated into t-shirt sizes: **XS, S, M, L, XL.** The decision about the size is based on an open and mutual collaborative discussion.

- This method is an **informal and quick way to get an rough feeling** about the total size of your backlog.

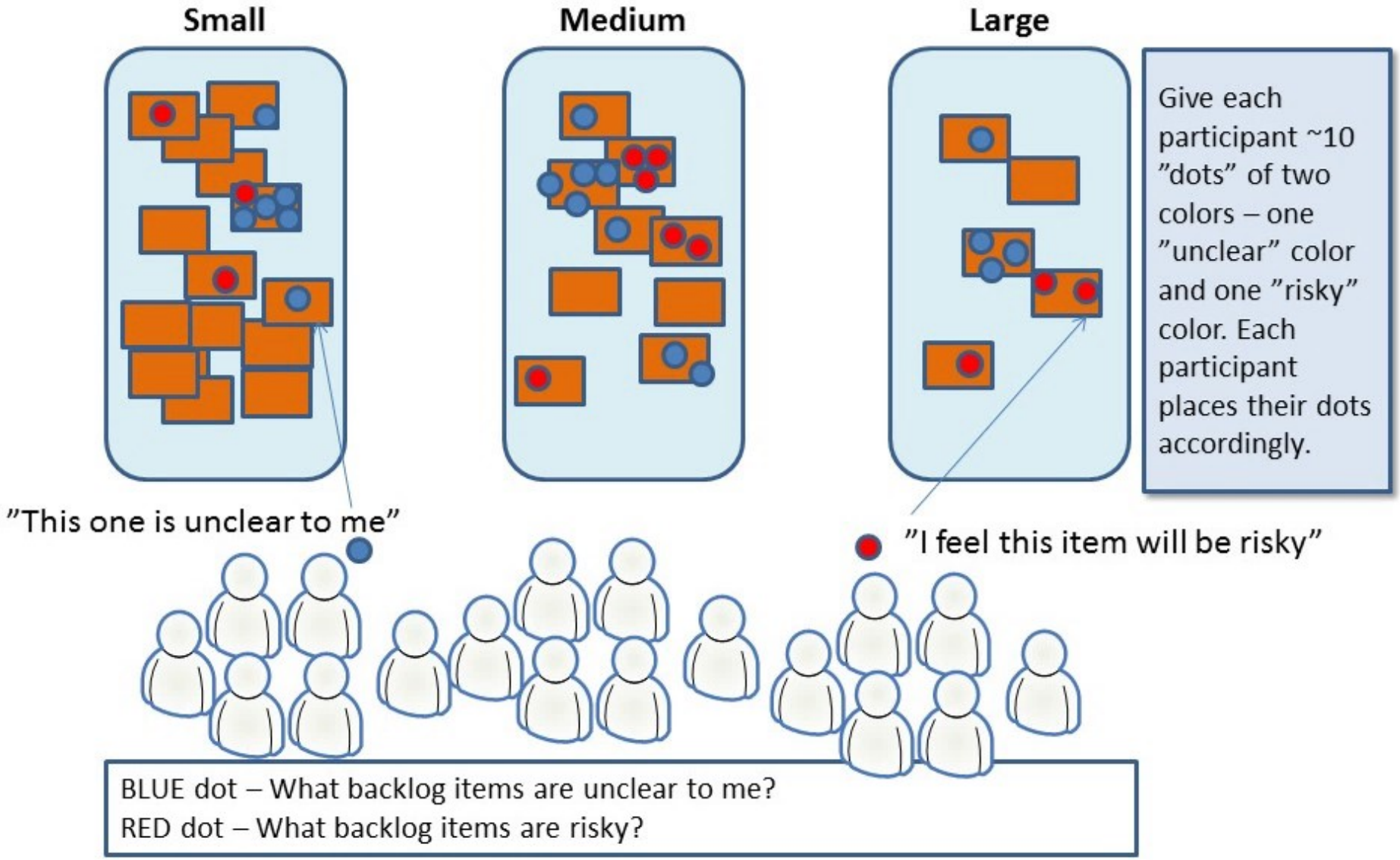
Agile estimation techniques: Dot Voting



Agile estimation techniques: Dot Voting

- When you are faced with a relative small set of items and in need of a super simple and **effective technique to estimate** you can use Dot Voting.
- This method has originated form **decision making** and you can use it for estimating.
- Each person gets a **small number of small stickers** and can choose to vote for the individual items. The **more dots is an indicator of a bigger size.**
- **Works well in both small and large group**. You have to limit the number of estimated items.

Agile estimation techniques: The Bucket System



Agile estimation techniques: The Bucket System

- Much **faster than planning poker is the Bucket System.** This system is a good alternative when estimating a **large number of items** with a **large group of participants.**

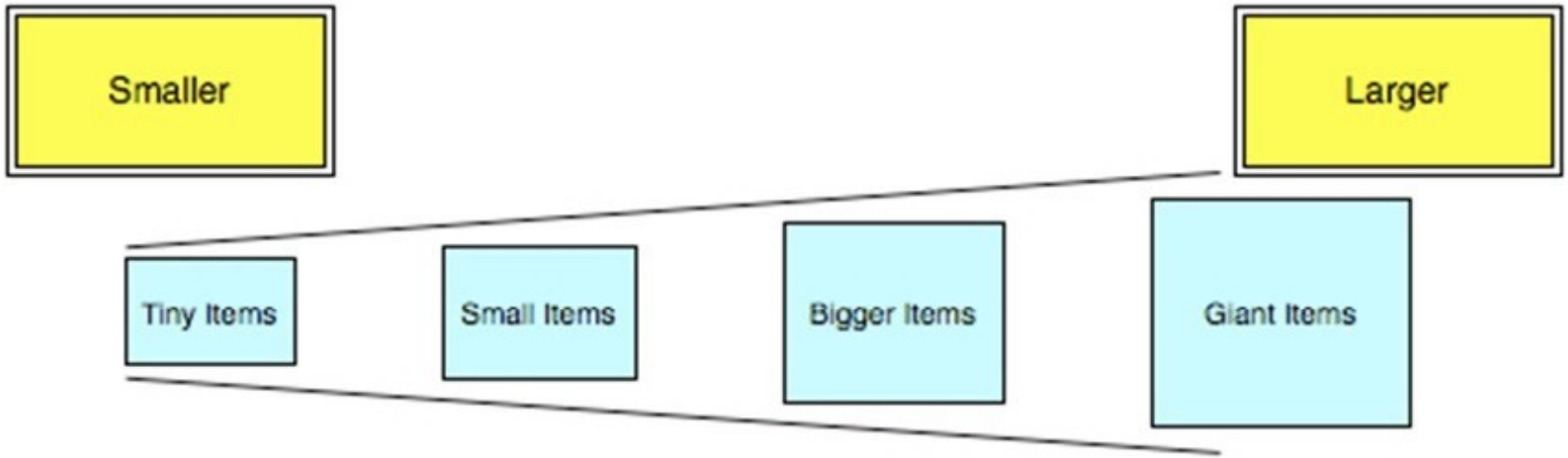
- **Create several buckets i**n the sequence of planning poker. The group estimates the items by placing them in these "buckets".

- **Buckets are usually different sheets of brown paper** where **you can place the sticky note** with the item. But you can also use actual baskets to limit discussion about already processed items.

Agile estimation techniques: Large/Uncertain/Small

- A very fast method of **rough estimating** is the Large/Uncertain/Small method. The team is being asked to place the items in one of these categories.
- The **first step is to categorize** the obvious items in the **two extreme categories.**
- Next the group can discuss the **more complex items**. This is actually a simplification of the bucket system.
- The system is especially good to use in **smaller groups** with comparable items. Next you can assign sizes to these 3 categories.

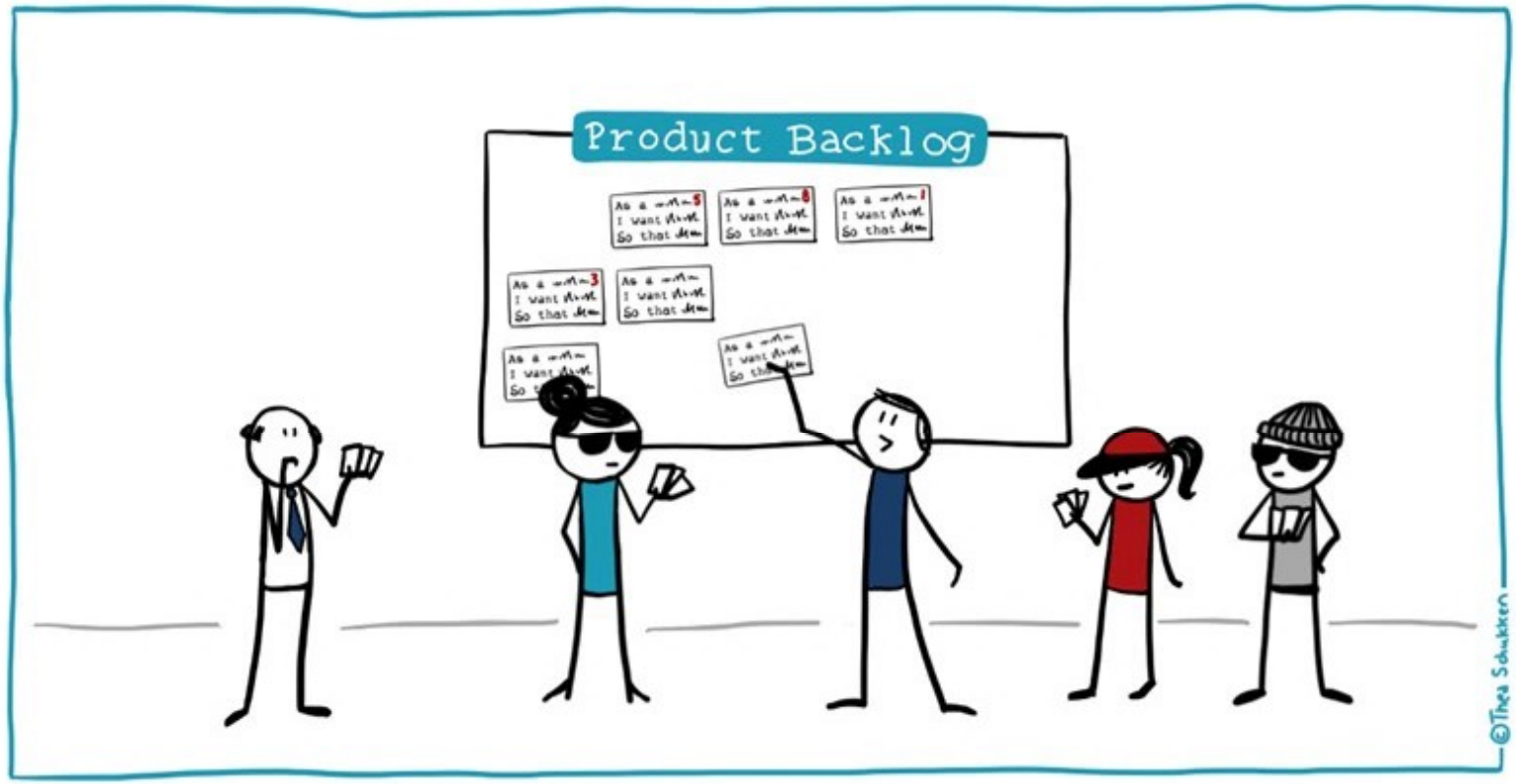
Agile estimation techniques: Affinity Mapping



Agile estimation techniques: Affinity Mapping

- This method is based on **finding similarities** in the estimated items. The team is asked to **group them together.**
- Best way is to execute this is a **visual way and order** them form **small groups to large.**
- It works best with a **small group of people** and a **relative small number of items.**
- You can **assign estimation numbers** to the **different groups.**

Agile estimation techniques: Ordering method



Agile estimation techniques: Ordering method

- This is an exercise where you get **an accurate image on the relative size of items.**
- This works best in a small group of expert. **All items are placed in random order** on a scale label ranging from **low to high**.
- **Every participant** is being **asked to move one item on the scale**.
- **Each move** is just **one spot lower or one spot higher** or pass the turn.
- **This continues till no team member want to move items** and passes their turn. The ordering protocol is a method of getting fine grained size estimates.
- **Works best with a relative small group of people** and a large number of items.

Agile Metrics

Agile Metrics

What are Agile Metrics?

- Metrics are nothing but **standards of measurement**. Agile metrics are standards that help a software team in **monitoring how productive a team is across the different phases of the SDLC**.

- Agile metrics help agile development teams and their management **measure the development process, gauging productivity, work quality, predictability**, **health of the team** and products being developed.

- A key focus of agile metrics is on **value delivered to customers** – instead of measuring "what" or "how much" we are doing, we measure **how it impacted a customer.**

Agile Metrics

Types of Agile Metrics:

There are **three important families** of agile metrics:

- **Lean metrics** – Focus on ensuring a **flow of value** from the organization to its customers and eliminating wasteful activities. Common metrics include lead time and cycle time.
- **Kanban metrics** – Focus on **workflow**, organizing and prioritizing work and getting it done. A common metric is a cumulative flow.
- **Scrum metrics** – Focus on the **predictable delivery** of working software to customers. Common metrics include the burndown chart and team velocity.

Agile Metrics

The Importance of Agile Testing Metrics :

- Agile methodologies place a special emphasis on **quality** because the **end goal is delivering working software to users** – buggy or unusable software is not working software.

- **Quality** is also manifested in internal aspects that are not directly visible to customers, such as **code quality, maintainability and technical debt.**

Agile Metrics

The Importance of Agile Testing Metrics:

- Agile testing metrics can help teams **measure and visualize the effort spent in software quality**, and to a certain extent, the results of this effort.

- For example, the **escaped defects** metric measures, across versions, sprints or product lines, **how many bugs were discovered in production** – whereas ideally bugs should be discovered and fixed during the development stage.

Agile Metrics

What makes for a powerful metric in an agile environment?

- Agile environments require metrics that are well understood by teams and can help learn and improve processes.

Here are a **few qualities that make a metric powerful**, in the sense that it can help drive positive improvement in an agile team:

- **The metric is used by the team** – Agile metrics should not be imposed or measured by management, they **should be used voluntarily by agile teams to learn and improve.**

- **The metric is surrounded by conversation** – Metrics should not just be numbers, they **should be the starting point of a conversation about process** and roadblocks affecting the team.

Agile Metrics

What makes for a powerful metric in an agile environment?

- **The metric is part of a specific experiment** – Metrics should be used to answer a specific question about agile processes, not just measured for the sake of measurement.

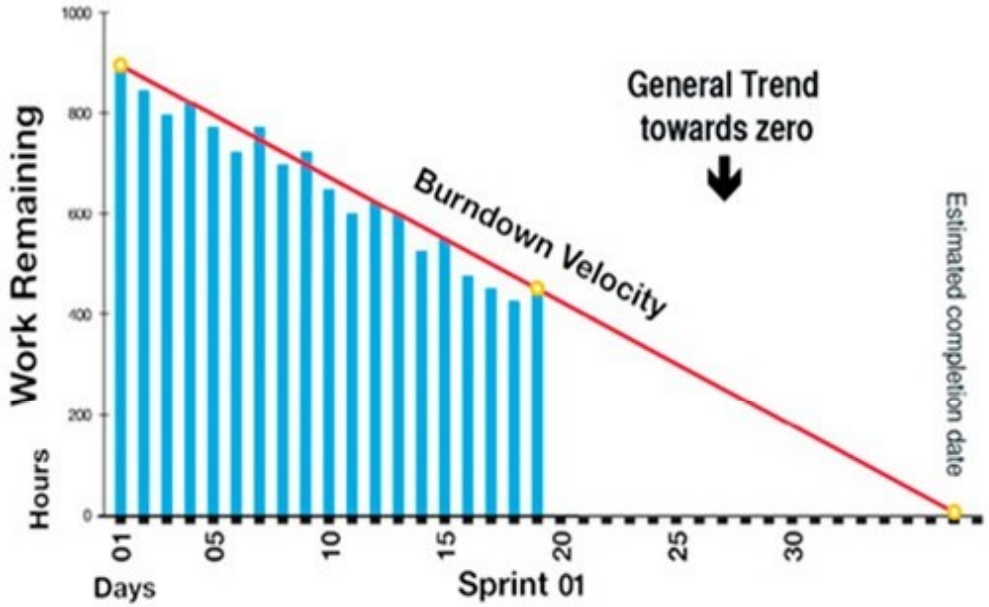
- **The metric is used in tandem with other metrics** – Even a great metric, if used alone, might lead to tunnel vision, and incentivize teams to maximize that metric at the expense of all else. **Using several metrics together provides a balanced picture of agile activity.**

- **The metric is easy to calculate and understand** – Metrics that are overly complex or not fully understood, even if they provide good insights about a team's work, are not useful in guiding day-to-day activities.

10 Powerful Agile Metrics

1. Sprint Burndown :

- The sprint burndown chart **visualizes how many story points have been completed during the sprint and how many remain,** and helps forecast if the sprint scope will be completed on time.

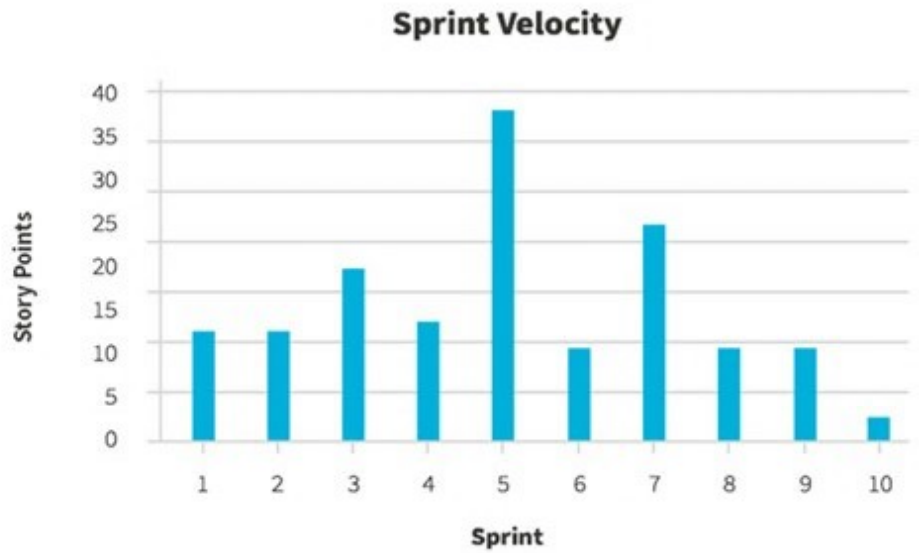


- **Why it is powerful:** Makes **it instantly clear how much value a sprint has already delivered and how close we are to completing our commitment** to customers.

10 Powerful Agile Metrics

2. Agile Velocity:

- Velocity measures **how many story points were completed by a team, on average, over the past few sprints.** It can be used to predict the team's output in the upcoming sprints.

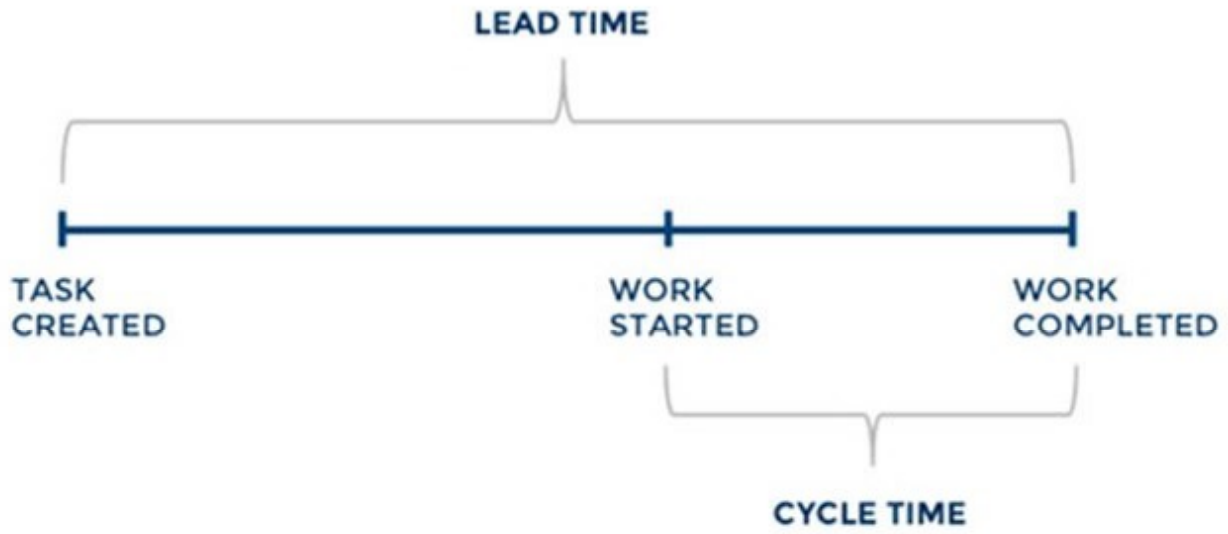


- **Why it is powerful:** Velocity is powerful because it's **a result metric – how much value was actually delivered to customers** in a series of sprints. Be careful not to compare velocity across teams because story points and definition of done can vary between teams.

10 Powerful Agile Metrics

3. Lead Time :

- Lead time measures the **total time from the moment a story enters the system (in the backlog), until it is completed** as part of a sprint, or released to customers. It measures the total time for a requirement to be realized and start earning value – the speed of your value chain.

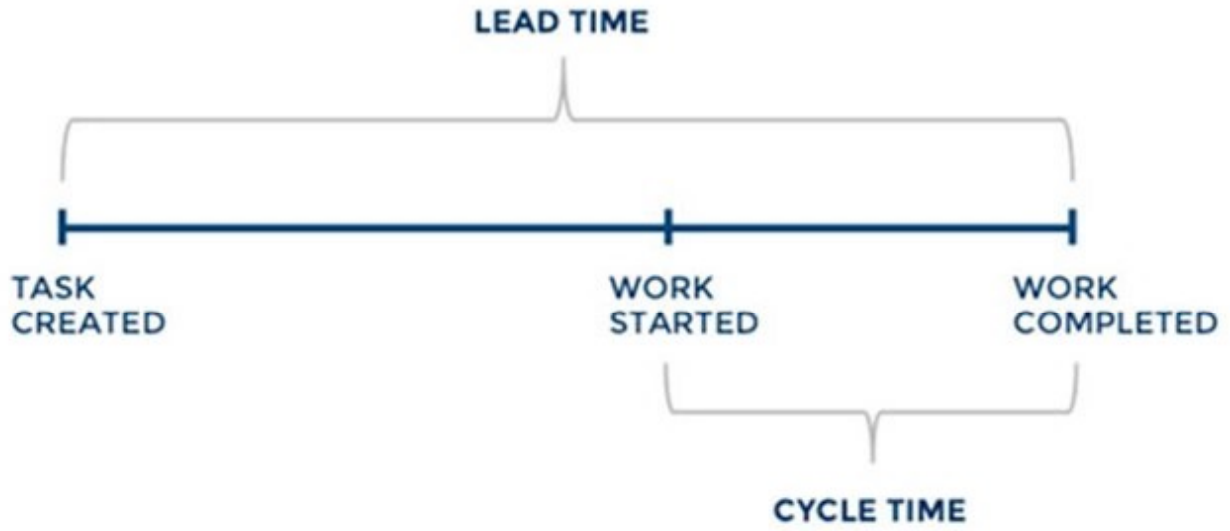


- **Why it is powerful:** In a sense, **lead time is more important than velocity because it measures the entire agile system f**rom end to end. Reducing lead time means the entire development pipeline is becoming more efficient.

10 Powerful Agile Metrics

4. Cycle Time :

- As illustrated above, the cycle time is a **subset of lead time** – it measures the **time for a task to go from "started" or "in progress" to "done".** Normally, cycle times should be around **half the sprint length**. If cycle times are longer than a sprint, teams are not completing work they committed to.



- **Why it is powerful:** A very simple metric that can raise a red flag when items within sprints across your entire system are not moving forward.

10 Powerful Agile Metrics

5. Code Coverage :

- Code coverage measures the **percentage of your code which is covered by unit tests.** It can be **measured by the number of methods, statements, branches or conditions** which are executed as part of a unit test suite.

- **Why it is powerful:** Code coverage can be **run automatically as part of every build** and gives a crude picture showing how much of the codebase has been tested. **A low code coverage almost always indicates low code quality**. However, a high coverage may not equal high quality, because there are other types of tests – such as UI or integration tests – which are not counted.

10 Powerful Agile Metrics

6. Static Code Analysis :

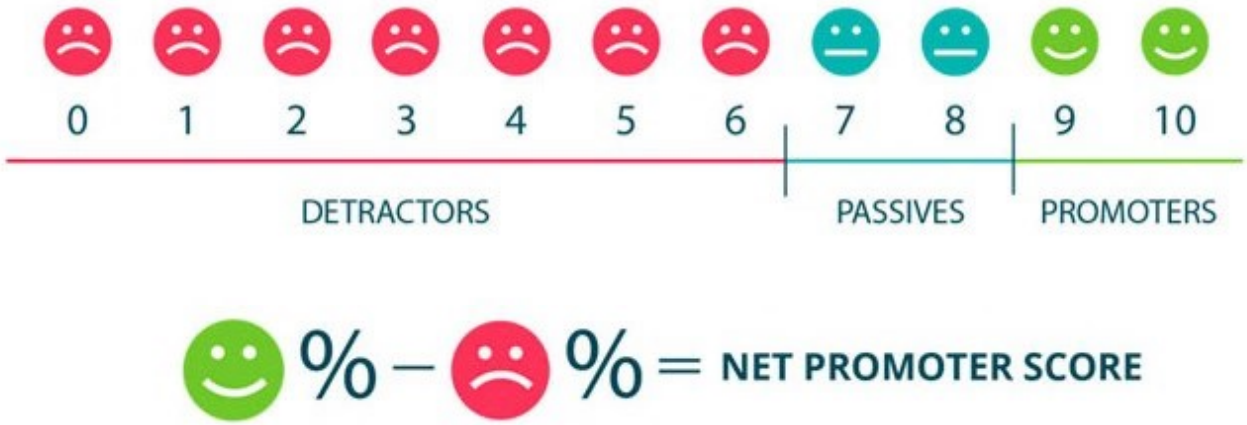
- While not exactly a metric, this is an automated process that can provide insights into code quality and **clean code from simple errors redundancies**. Code quality, while difficult to define and measure, is known to be a key contributor to software quality in general, and in particular, software maintainability.

- **Why is it powerful:** Static code analysis provides a safe baseline for code quality. However, it is no substitute for human input into code quality, via manual code reviews, pair programming or other methods.

10 Powerful Agile Metrics

7. Release Net Promoter Score :

- **Net Promoter Score (NPS), calculated for a software release, measures whether users would recommend the software to others,** do nothing, or recommend against using it. It is an important gauge of customer satisfaction.

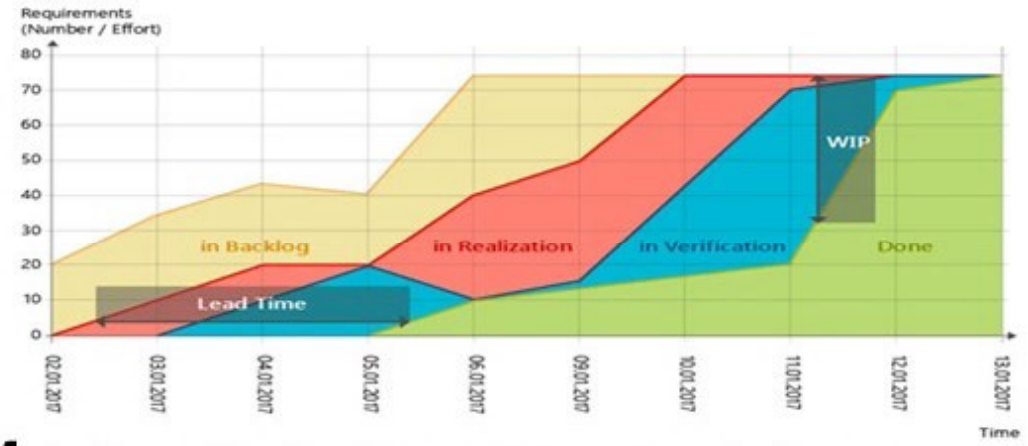


- **Why is it powerful:** The ultimate test of agile development is providing value to a customer. If customers are recommending this new release to others, that is a clear indication of success. If not, you can use this as a warning metric and use other data to understand what's wrong.

10 Powerful Agile Metrics

8. Cumulative Flow :

- This is a **kanban metric which shows the status of tasks** – in a sprint, a release or across software teams. It can visualize bottlenecks in the process – a disproportionately large number of tasks in any of the workflow stages indicates a problem. For example, a big "bubble" in the chart in a verification or testing stage indicates this stage has insufficient resources.



- **Why is it powerful:** As with the burndown chart, the power of this metric is in its visual simplicity – you can grasp a process in one glance and immediately identify issues. Cumulative flow lets you catch problems in mid-process before they result in delayed delivery.

10 Powerful Agile Metrics

9. Failed Deployments :

- Measures the number of deployments (either to test, production environments, or both). Can help understand how solid environments are and whether teams are really building potentially shippable software.

- **Why it is powerful:** Especially when applied to production environments, this metric can provide a clear indication that sprints or releases are production ready, or not.

10 Powerful Agile Metrics

10. Escaped Defects :

- The number of bugs discovered only after a build or release enters production. **Escaped defects should ideally be zero**. Measuring them across releases or teams provides a crude, but still highly relevant, a measure of deployed software quality.

- **Why it is powerful:** Production bugs, especially if frequent, are a problem in the agile process. Just like in lean manufacturing, we should "stop the production line" and discover what's wrong.

Principles of Agile Metrics

Principles of Agile Metrics

- Outcome Over Activity
- Value Over Volume
- Trends Over Absolute numbers
- Assessment Over Measurement
- Improvement Over Fault finding
- Transparent and visible at all times
- Team Over Individual
- Provide meaningful insights
- Have a balanced set-compensating metrics
- Don't compare teams on absolute numbers
- Incentivize "Right" behaviors
- Not be prone to gaming

Principles of Agile Metrics

Metrics should be used by the team :

- This sounds like an obvious one, but it isn't. In fact, **99% of the time people are using agile metrics,** they're not following this rule.
- Most of the time, **someone outside the team, usually a manager**, wants to "spy on" or "measure" or "assess" the team, usually for "checking their productivity".

This is a **poor pattern for a number of reasons:**

- These **metrics don't measure productivity.**
- There isn't an easy way to measure productivity.
- **Productivity isn't the primary measure of success – delivering valuable software is instead** (an unproductive team who delivers valuable software over a productive one that isn't delivering software or is delivering useless software any day of the week)

Principles of Agile Metrics

Metrics should be used by the team :

- Most of these metrics are **supposed to be starting points for conversations**, and if "managers" are reading these metrics in some kind of report, then they're probably not being involved in those conversations.

- **Most of these metrics have meaning** because of the context around them, and the only people who have experienced and will understand that context are the people in the team.

- So the team should be the ones collecting the metrics, and **the team should be the ones sharing and using the metrics.** Try to avoid these metrics leaving the team and going to outsiders as much as possible. The last thing you want is some random middle manager wanting to know why team X has a velocity of 40 and team Y has a velocity of 50. That is not a productive conversation.

Principles of Agile Metrics

Metrics need to be surrounded by conversations :

- **Numbers are important** and can help tell a story, but they need to be part of an actual conversation (not an email trail or a spreadsheet, an actual conversation between actual people).

- **Just showing numbers into a presentation will not tell a story. If you want to tell a story, weave the numbers into a conversation** about how you collected them, when, where, and why. And what you plan on doing with those numbers when you get them (or what you have done with them now you have them).

Principles of Agile Metrics

Metrics should be used as part of a specific investigation or experiment :

- As you will see by reading all metrics, there are a lot of metrics that you can capture when doing software development.
- **If you just collect them all, hoping to understand everything at once**, you will get overwhelmed and not get anywhere.
- Each metric tells a certain story and can be used as part of a specific investigation ("what are we having problems with?"), or experiment ("how can we improve that?").

Principles of Agile Metrics

Metrics should be used as part of a specific investigation or experiment :

- Ideally, this investigation or experiment will come out of a retrospective or similar activity.

- **For example**, a **team notices** that they are **having trouble meeting sprint goals.** They decide to **investigate their cycle time and find out that it has gone up.**

- They then **decide to an experiment to reduce the number of code** and design reviews and measure their cycle time again to see if it has improved.

- That is a perfect example of using metrics **around a conversation, an investigation and an experiment.**

Principles of Agile Metrics

- **Don't produce metrics that no one wants.** Too many metrics are overwhelming, and the sheer volume can bury important problems. Minimize the overall number of metrics, and emphasize the ones that tell a story. Try eliminating a questionable metric and see if anyone complains.
- **Be honest about how management uses metrics.** As soon as developers find out that metrics are used as the basis for punitive measures, even the most honest developer will find a way to game them.
- **Don't use metrics to compare teams.** Metrics are best viewed as probes and indicators of problem areas that may warrant further investigation.
- **Don't introduce metrics that require significant work to produce.** If an automated build or tool can't produce a metric, it's probably not worth the effort.

Principles of Agile Metrics

- **Take team maturity into account when selecting metrics.** Metrics should change over time. For a team just learning TDD, a simple count of unit tests should emphasize the simple behavior of getting the team to write more tests. The team should quickly mature beyond this phase, and in two iterations should be looking at a more meaningful metric (such as code coverage). And code coverage, too, should be de-emphasized as its numbers become less interesting. Be willing to change metrics.
- **Ensure that metrics don't demoralize the team.** A metric that shows little hope of improvement can crush the spirits of a team. For example, in the early phases of trying to cover a large, legacy codebase with tests, a coverage metric is going to show very, very low percentages.
- **A single metric on its own has minimal use.** As mentioned earlier, a management team shouldn't emphasize fixing a single metric, such as code coverage, as the goal. Instead, management should emphasize instead fixing the real problem (perhaps the answer is, "get them to really do TDD").
- **Use metrics as a basis for discussion, not as a final decision point.**

Principles of Agile Metrics

- Measure the project, the teams, and the adoption separately
- Start collecting metrics early and often
- Stay focused
- Be consistent
- And, most importantly, measure responsibly.

Scrum Release Planning

Scrum Release Planning

- A very **high-level plan for multiple Sprints** (e.g. three to twelve iteration) is created during the Release planning.
- It is a **guideline that reflects expectations about which features will be implemented and when they are completed**.
- It also **serves as a base to monitor progress** within the project.
- Releases can be **intermediate deliveries** done during the project or the final delivery at the end.

Scrum Release Planning

To create a **Release Plan** the following things have to be available:

- A **prioritized and estimated Scrum Product Backlog**
- The (estimated) **velocity of the Scrum Team**
- Conditions of satisfaction (goals for the schedule, scope, resources)

- **Depending on the type of project** (feature- or date-driven) **the release plan can be created in different ways:**

- **If the project is feature-driven, the sum of all features within in a release can be divided by the expected velocity.** This will then result in the number of sprints needed to complete the requested functionality.

Scrum Release Planning - Sample

Release Goals:

(1) First Release after user able to login & logout.

(2) Second Release after user can manage items

(3) Final Release when all stories are done

First Release

Stories: 7, 1 10 → Estimation = 5 Points

Estimation / Velocity = 5/3 → 2 Sprints

Second Release

Stories: 9,2,4,3,5 → Estimation = 15 Points

Estimation / Velocity = 15/3 → 5 Sprints

Final Release

All Stories → Estimation = 30 Points

Estimation / Velocity =30/3 → 10 Sprints

ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

Velocity: 3 Points / Sprint

Releases after Sprint 2, 7 and 10

Scrum Release Planning

- If the project is **date-driven** we can simply **multiply the velocity by the number of Sprints** and we'll get the **total work** that can be completed within the given timeline.

Release Goals:
Release every 6 weeks

ToDo List

ID	Story	Estimation	Priority	
7	As an unauthorized User I want to create a new account	3	1	
1	As an unauthorized User I want to login	1	2	
10	As an authorized User I want to logout	1	3	
9	Create script to purge database	1	4	
2	As an authorized User I want to see the list of items so that I can select one	2	5	*Release*
4	As an authorized User I want to add a new item so that it appears in the list	5	6	
3	As an authorized User I want to delete the selected item	2	7	*Release*
5	As an authorized User I want to edit the selected item	5	8	*Release*
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9	*Release*
8	As an administrator I want to see the list of accounts on login	2	10	
Total		**30**		

Velocity: 3 Points / Sprint **Sprint length 2 weeks**

Scrum Release Planning

- Like the Scrum Product Backlog the Release plan is **not a static plan**. It will change during the whole project when new knowledge is available and

  e.g. **entries in the Scrum Product Backlog are changed and re-estimated.**

- Therefore the **Release Plan should be revisited and updated in regular intervals**, e.g. after each Sprint.

Estimation in Scrum

Estimation in Scrum

- In Scrum Projects, **Estimation is done by the entire team** during Sprint Planning Meeting.

- The **objective of the Estimation** would be to **consider the User Stories for the Sprint by Priority** and by the **Ability of the team** to deliver during the Time Box of the Sprint.

- **Product Owner ensures that the prioritized User Stories are clear**, can be subjected to estimation, and they are brought to the beginning of the Product Backlog.

Estimation in Scrum

- As the **Scrum Team in total is responsible for the delivery of the product increment,** care would be taken to select the User Stories for the Sprint based on the size of the Product Increment and the effort required for the same.

- The **size of the Product Increment is estimated in terms of User Story Points**.

- Once the size is determined, the effort is estimated by means of the past data, i.e., **effort per User Story Point called Productivity**.

Estimation in Scrum

Scrum Estimation Techniques :

- The Scrum Estimation of User Stories is in terms of the **degree of difficulty** for each of the User Stories. To assess the degree of difficulty, a particular scale is used.
- There are several **types of scales** that are used in **Scrum Estimation**. Following are some examples -
  - **Numeric Sizing (1 through 10)**
  - **T-shirt Sizes (XS, S, M, L, XL XXL, XXXL)**
  - **Fibonacci Sequence (1, 2, 3, 5, 8, 13, 21, 34, etc.)**
  - **Dog Breeds (Chihuahua,.....,Great Dane)**
- The estimation technique is normally chosen in such a way that the entire scrum team is acquainted and comfortable with scale's values. The **most commonly used and most popular technique is Planning Poker which is based on Fibonacci sequence.**

References

- https://dzone.com/articles/agile-planning-estimation
- Cohn , Mike . Agile Estimating and Planning . Prentice Hall, 2005. Print.
- Fowler, Martin. "C3." Martin Fowler. N.p., N.d., Web. 30 Dec. 2012.
- Kishawy, Mohamed. "Agile Methodologies - The Missing Path." DZone. MN, USA, 14 May 2012. Web. 30 Dec. 2012.
- Ries, Eric. The Lean Startup. New York: Crown Business, 2011. Print.
- Smits, Hubert. "Scaling Agile Processes: Five Levels of Planning." Pragmatic Marketing. N.p., 7 May 2007. Web. 30 Dec. 2012.