Agile estimation techniques

- T-Shirt Sizing
- Sprint Poker
- Three-Point Method
- Affinity Estimation
- Relative Mass Evaluation
- Dot voting
- Maximum allowable size (MAS)
- Big, Uncertain, Small.

T-Shirt Sizing Procedure

1. Product Owner will explain the story to be estimated and the development team will ask questions if they have any issues
2. Each developer gives each story, a t-shirt size.
3. All in members in the development team will raise their cards simultaneously.
4. The development team will discuss the differences.
5. The product owner explains the story further or clarifies misunderstanding if any.
6. The team will Go back to Step 2-Step4 until all are agree with one size.
7. Complete or place the stories in size buckets.
8. Estimate the time to complete all stories in S, M, L, XL buckets.

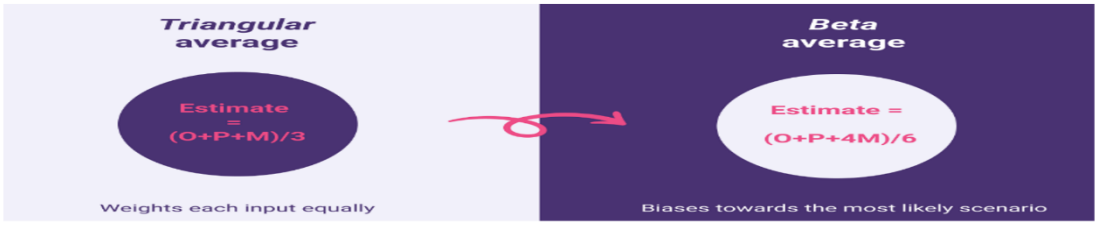
Sprint Poker

1. **The product owner or customer reads an agile user story or describes a feature to the estimators.**
2. **Each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100.**
3. **The estimators discuss the feature, asking questions of the product owner as needed. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time.**
4. **If all estimators selected the same value, that becomes the estimate. If not, the estimators discuss their estimates. The high and low estimators should especially share their reasons. After further discussion, each estimator reselects an estimate card, and all cards are again revealed at the same time.**
5. **The poker planning process is repeated until consensus is achieved or until the estimators decide that agile estimating and planning of a particular item needs to be deferred until additional information can be acquired.**

Three-Point Method

Three-point Estimation looks at three values –

- the most optimistic estimate (O),
- a most likely estimate (M), and
- a pessimistic estimate (least likely estimate (L)).



Affinity Estimation

Three steps of Affinity Estimation are:

1. Silent Relative Sizing
2. Editing the Wall
3. Placing items in correct bucket

Affinity Estimation Procedure

Step 1: Silent Relative Sizing

- First a horizontal scale is chosen. One end of the scale is marked with "Smaller" and the other end is marked with "Larger"
- The product owner provides the user stories to the team.
- Each participant estimates their story solely without any discussion with other participants and after estimation, places it at the correct location on the scale, anywhere between "Smaller" and "Larger".

Step 2: Editing the Wall

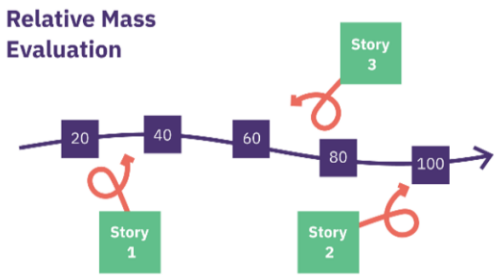
- Once each story is placed on the scale team members edit the relative sizes on the wall. For this team discussed about the story with each other about its design, implementation or any challenges. Team gets all the doubts clarified from Product Owner as well.
- Based on the understanding made during discussion, team re-arranges its story on the scale if required.

Step 3: Placing items in correct bucket

- The scale "Smaller" to "Larger" is divided and marked appropriately with the markers of XS, S, M, L, and XL if we use t-shirt sizing technique or with 0, 1, 2, 3, 5, 8, 13, and so on, if we use Fibonacci series of planning poker estimating technique.
- Team member now places their stories, which were on scale "Smaller" to "Larger", to appropriate bucket of the converted scale.

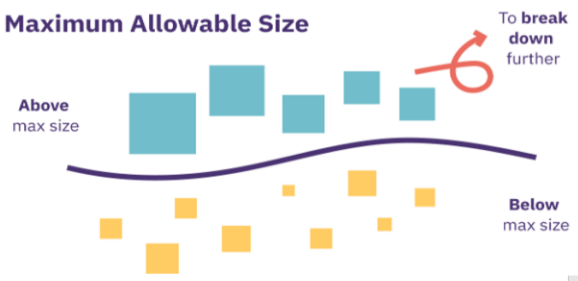
Relative Mass Evaluation

1. Write up a card for each story.
2. Then set up a large table so the stories can be moved around easily relative to each other.
3. Pick any story to start, then get the team to estimate whether they think that it is relatively large, medium, or small.
4. If it's a large story, place it at one end of the table. If it's a small story, it goes at the other end of the table. A medium story goes in the middle. Now select the next story and ask the team to estimate if it's more or less effort than the story that you just put down. Position the story card on the table relative to the previous card, and go to the next card.



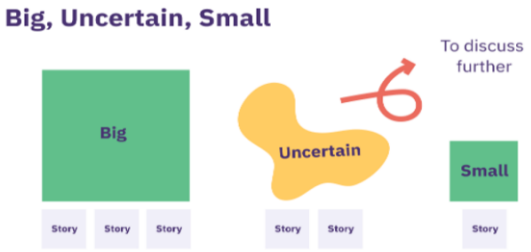
Maximum allowable size (MAS)

1. Set your estimation scale. In this case, it might be worth using a scale according to the number of hours something will take, so you have a concrete idea of the maximum amount of time you want your team to spend on a given piece of work. This will make your filtration more actionable.
2. As a general rule, user stories probably shouldn't take more than 16 hours of work, otherwise they can become unwieldy. So you might want to take 16 hours as your maximum allowable size. If you don't need the same level of precision, use the good old t-shirt sizes to guide you.
3. Now the task is to go through all the stories in your backlog and filter out the ones the team think are higher than 16 hours by voting on them.
4. You can then take those items and discuss how to break them down so they fit below the maximum allowable size.



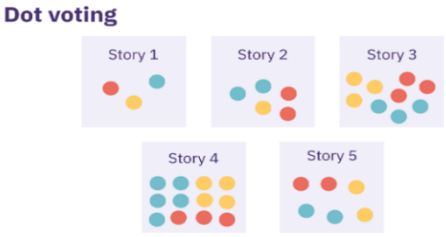
Big, Uncertain, Small

The 'Big/Uncertain/Small' agile estimation method is similar to the bucket system. It involves placing the estimated items in one of the three categories. Teams discuss the items and then a 'divide-and-conquer' technique used to estimate the rest of the items. The items left are distributed among team members and can be quickly estimated in parallel.

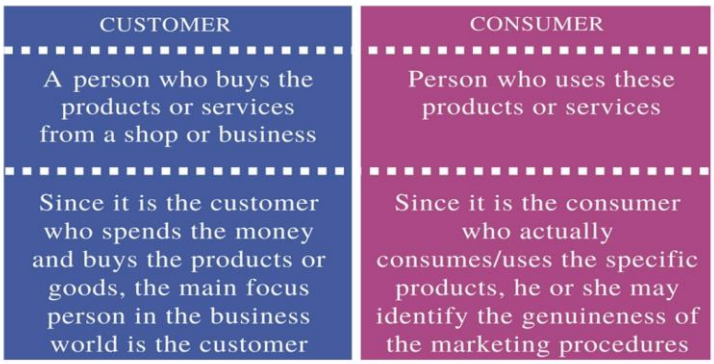


Dot Voting

1. All user stories are put on a wall, virtual or real by the Product Owner.
2. Every team member is given 4-5 votes; these can be small round sticky notes.
3. Every team member is asked to give their votes on the stories they think are bigger.
4. Team member puts/pastes the round red sticky notes on the stories they think are big in size.
5. Every team member performs this process until their all 4-5 votes are exhausted/used. At the end of this process, the story with higher votes is termed as biggest and that with low number of votes are smallest.
6. Product owner then orders the story from higher votes to lower votes.
7. One person can vote more than once for one story.
8. Same technique is used to decide priority as well. More votes means higher priority item.



Customer Vs User



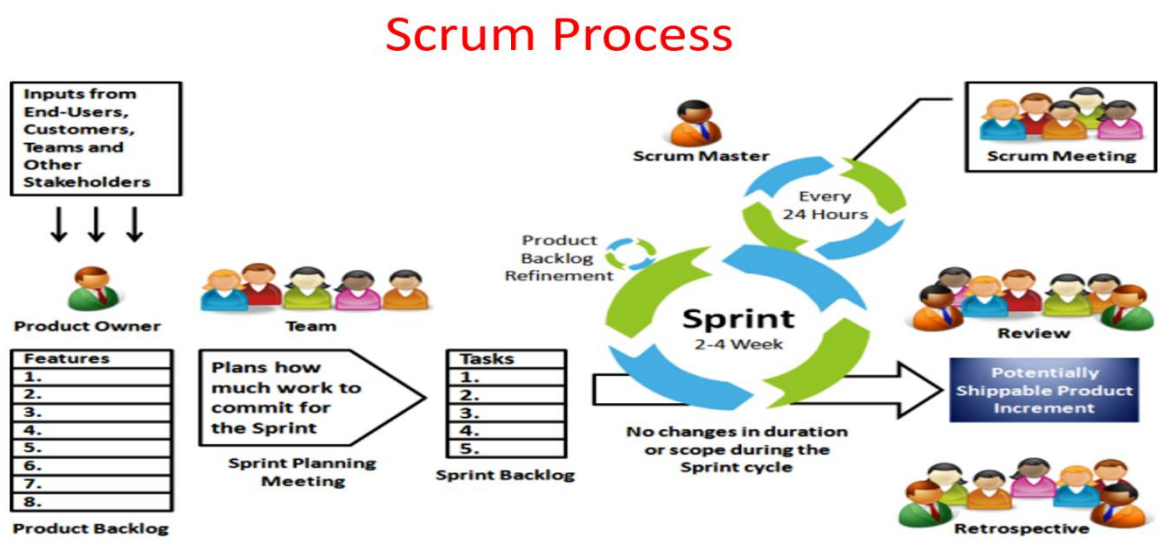
A customer is someone who's paying for your product/service. This person may or may not be an end user.

An end user is someone who will use your product/service. This person may or may not be a customer. It could be someone in your organization, for example, or it could be a person your customer wants to do business with.

DIFFERENCE BETWEEN AGILE AND TRADITIONAL DEVELOPMENT

S. No.	Traditional Software Development	Agile Software Development
1.	It is used to develop simple software.	It is used to develop complicated software.
2.	In this methodology, testing is done once the development phase is completed.	In this methodology, testing and development processes are performed concurrently.
3.	It follows a linear organization structure.	It follows an iterative organizational structure.
4.	It provides less security.	It provides high security.
5.	Client involvement is less as compared to Agile development.	Client involvement is high as compared to traditional software development.
6.	It provides less functionality in the software.	It provides all the functionality needed by the users.
7.	It supports a fixed development model.	It supports a changeable development model.
8.	It is used by freshers.	It is used by professionals.
9.	Development cost is less using this methodology.	Development cost is high using this methodology.
10.	It majorly consists of five phases.	It consists of only three phases.
11.	It is less used by software development firms.	It is normally used by software development firms.
12.	Expectation is favored in the traditional model.	Adaptability is favored in the agile methodology.

SCRUM PROCESS :



Steps :

- Step 1: Product Backlog Creation. ...
- Step 2: Sprint planning and creating backlog. ...
- Step 3: Working on sprint. ...
- Step 4: Testing and Product Demonstration. ...
- Step 5: Retrospective and the next sprint planning

Step 1: Product Backlog Creation

In this step, the large items and functional details are transformed into epics and user stories. The user stories are transformed from large items and are smaller which can be put in the product backlog. The epics can also be included in the product backlog but cannot be included in the sprint backlog without converting it to a user story.

A typical user story example is as an admin I want to add, modify, and delete the tasks for the users in the website.

The following required fields in the user stories are necessary.

- The user stories significance
- The initial estimate made during the meeting
- Demo of how to make the user stories into tasks

Know what is included in the product backlog

Step 2: Sprint planning and creating backlog

The sprint duration is very important so that the user stories are as small as possible. The typical average sprint duration lasts about 2 weeks. If the sprint duration is small than the advantage is that more customer feedback can be received and most of the errors and bugs can be addressed earlier. If the sprint duration is long then it allows the developer to work thoroughly.

The next stage is to do the sprint backlog creation for which the scrum team must select the important user stories and make them into smaller tasks. They need to plan on how to get the task completed. Also, one important thing is to prioritize the necessary tasks. StarAgile conducts CSM certification online training for the professionals aspiring to become a certified scrum master.

Step 3: Working on sprint

The actual user stories are moved as small tasks in the sprint backlog where the actual work starts. This is where the realization of the software application for example the website development begins.

To begin with, a task board also called a Kanban board is made with a lot of cards is used. The cards specify the details about the tasks such as assignee, work details, due date or the time duration, etc. The task board consists of the following columns "Product backlog or the User stories", and the "To Do" lists, "Work In Progress" and then "Testing" and "Work Done" columns.

The cards can be moved from the left to right in the order of preference and based on the completion. The task board is explained thoroughly in the Scrum master certification training online at StarAgile institute.

Step 4: Testing and Product Demonstration

The tasks completed are to be realized as a working product with full life cycle testing. The testing cost can be minimized with the addition of QA or having fewer user stories, however, the first one is the best possible solution. Every sprint that is completed must be demonstrated to the customer for his acceptance and his viewpoint on the complete solution

Step 5: Retrospective and the next sprint planning

The result of this step is to discuss what went well and what can be improved for the next level. Also, you need to discuss the lessons learned and the pitfalls of any particular issues or problems. Then the next sprint planning has to be commenced based on the knowledge that we have for the current processes and past projects. StarAgile conducts Certified scrum master online training for the professionals. Understand the sprint retrospective process.

Scrum has three roles:

product owner

scrum master

development team members.

Group of roles	Role	Description
Leading group	Coach	Coordinates and solves group problems, leads and guides development sessions
	Tracker	Measures the group progress by measures as defined by the team, the customer, and the organization; manages the workspace boards; manages the team diary/collective memory. See also Chapter 5, Measures
	Methodologist	Makes sure that the team works according to the defined development process, answers questions related to the methodology, looks for solutions to problems related to the methodology
Customer group	User evaluator	Performs an ongoing user evaluation of the product (collects and processes feedback received from real end users), holds a user centric approach, serves as the user interface designer. See also Chapter 3, Customers and Users
	Customer	If the project doesn't have a real customer: tells customer stories, makes decisions pertaining to each iteration, provides feedback, defines acceptance tests. See also Chapter 3, Customers and Users
	Acceptance tester	Defines (with the customer) and develops acceptance tests, inspires a test-driven development process. See also Chapter 6, Quality
Code group	Designer	Maintains current design, works to simplify design, searches for refactoring tasks and ensures their proper execution. See also Chapter 8, Abstraction
	Unit tester	Establishes an automated test suite, guides and supports others in the development of unit tests, guides a test-driven development process. See also Chapter 6, Quality
	Continuous integrator	Establishes the integration environment; publishes and encourages rules pertaining to the addition of new code, including testing issues
	Code reviewer	Maintains source control, establishes and refines coding standards, guides and manages the team's pair programming
Maintenance group	Presenter	Plans and organizes iteration/release presentations, demos, and roles; measures presentations
	Documenter	Plans and organizes the project documentation: process documentation, user's guide, and installation instructions
	Installer	Plans and ensures the development of an automated installation kit, maintains the collaborative workspace infrastructure