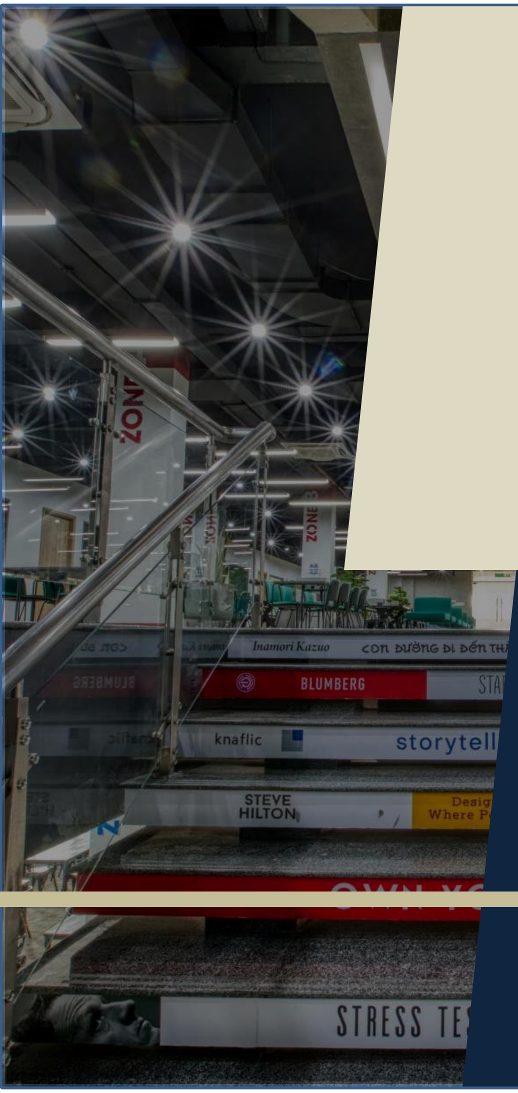
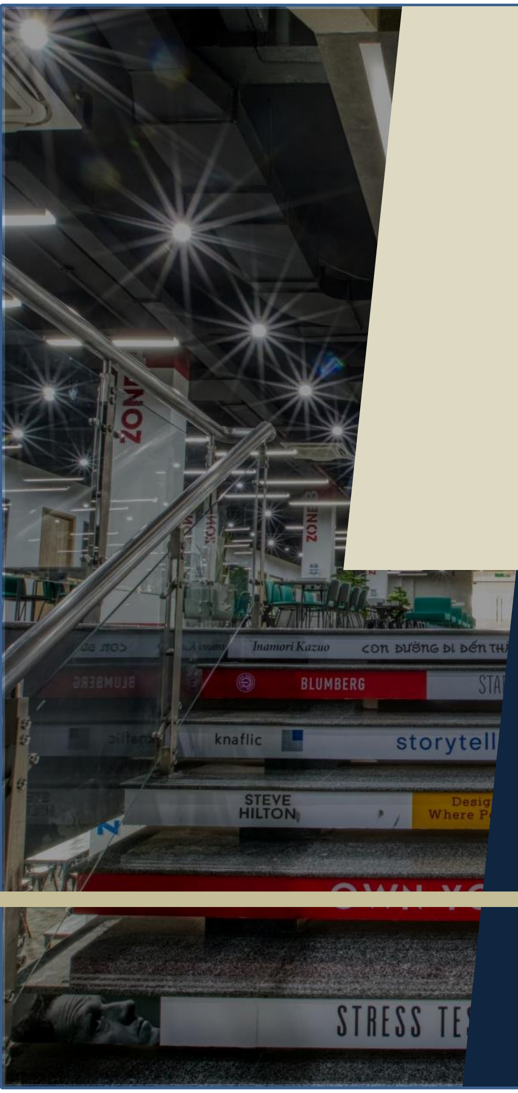


TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

Software Engineering Course's Code: CSE 305



TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

Chapter 3. Agile Software Development

3.1. Agile Process

3.2. Scrum

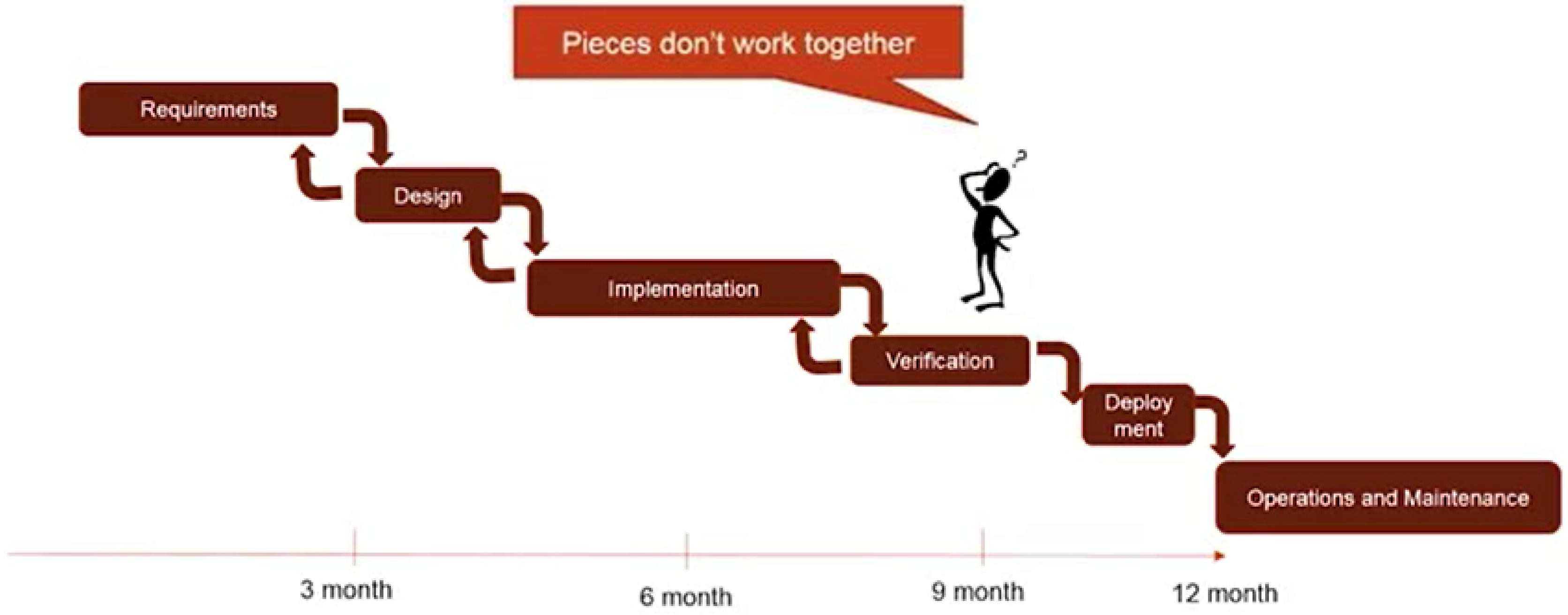
3.3. Kanben

3.4. Extreme Programming

EIU

Why Agile?

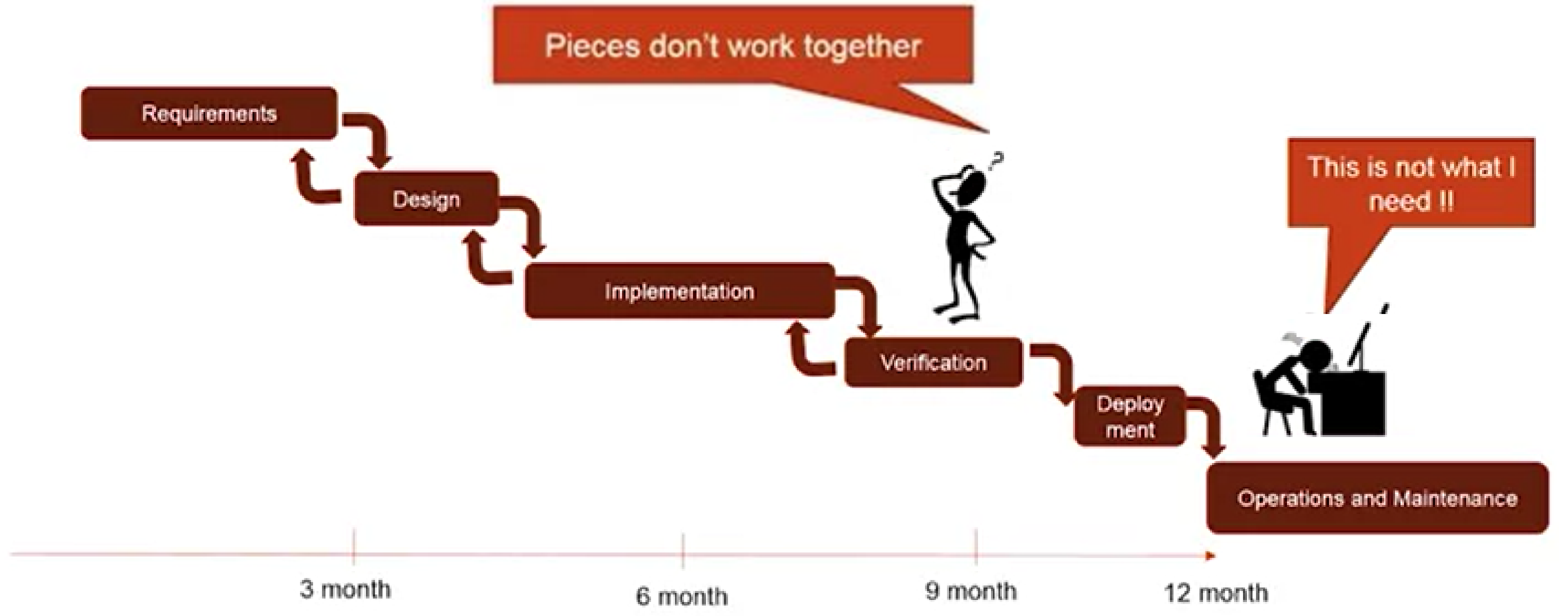
CHALLENGES WITH WATERFALL METHOD



- In the verification phase where you put all the components together and
- you try to find out whether the system is working as expected.
- But, you see pieces don't work together

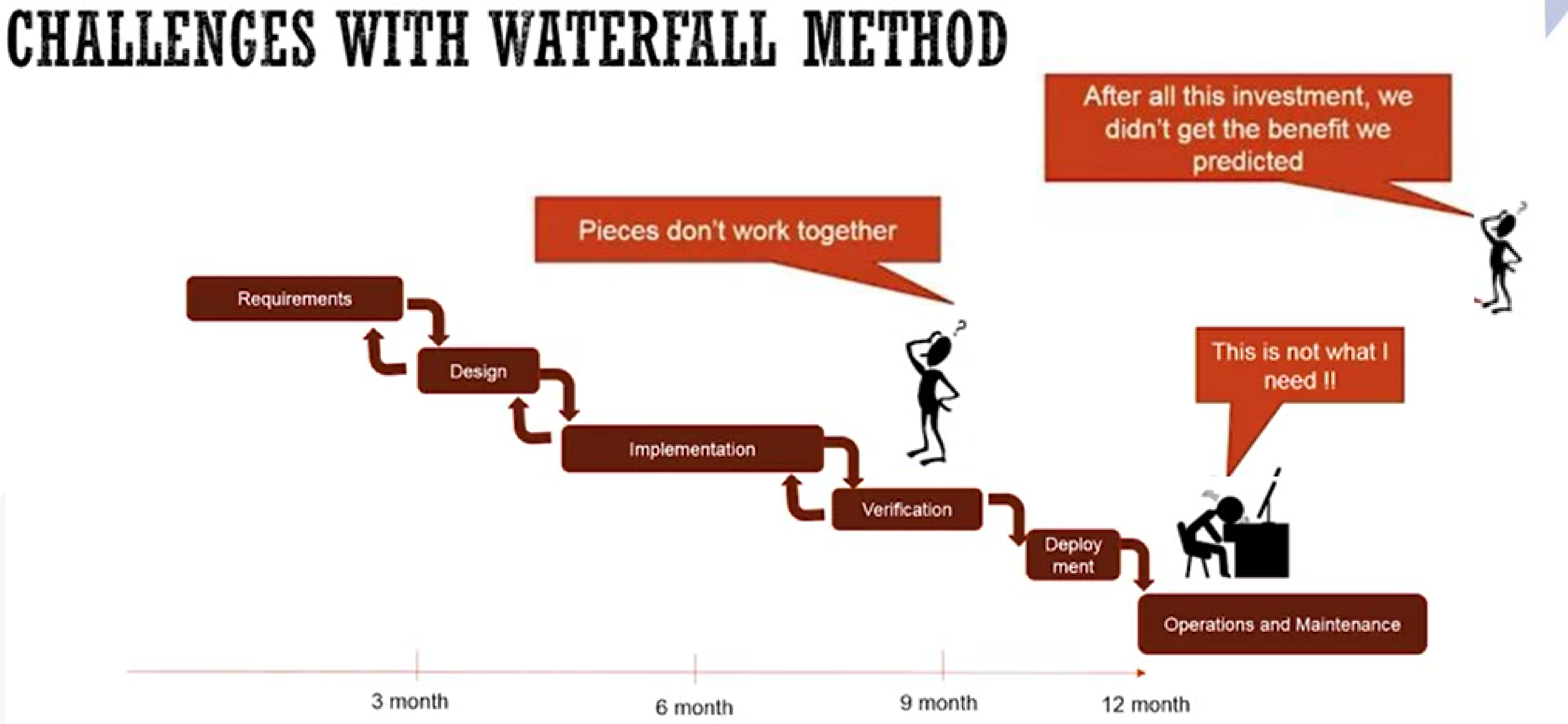
Why Agile?

CHALLENGES WITH WATERFALL METHOD



- Then another problem they were finding was after the software is developed and deployed,
- the users were using the system and they were saying this is not what I was expecting, or this is not what I need, this doesn't work for me

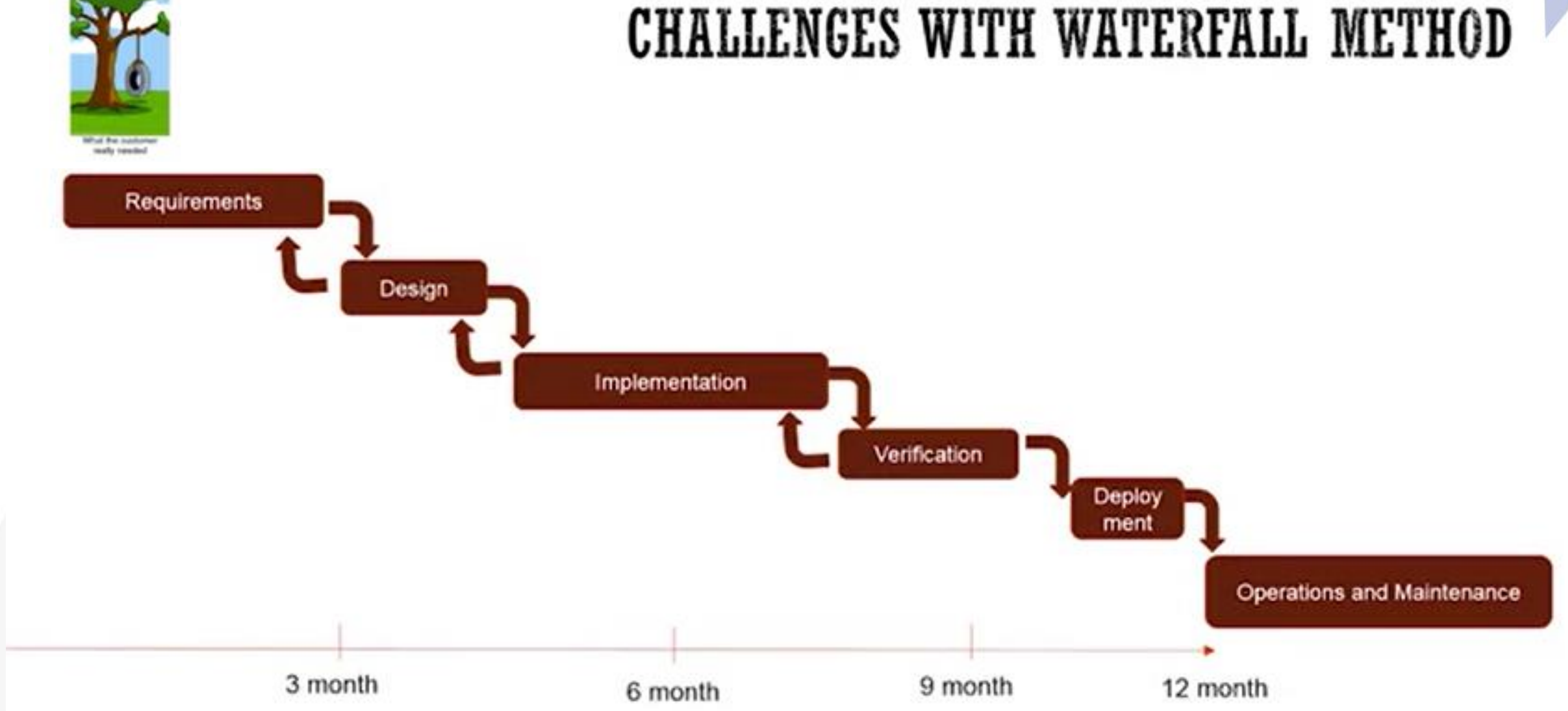
Why Agile?


CHALLENGES WITH WATERFALL METHOD
After all this investment, we didn't get the benefit we predicted
Pieces don't work together
Requirements
Design
Implementation
Verification
Deploy ment
Operations and Maintenance
This is not what I need !!
3 month
6 month
9 month
12 month

➢ The clients were saying well, I invested so much money or so much time in building the software and not getting the benefit that I predicted

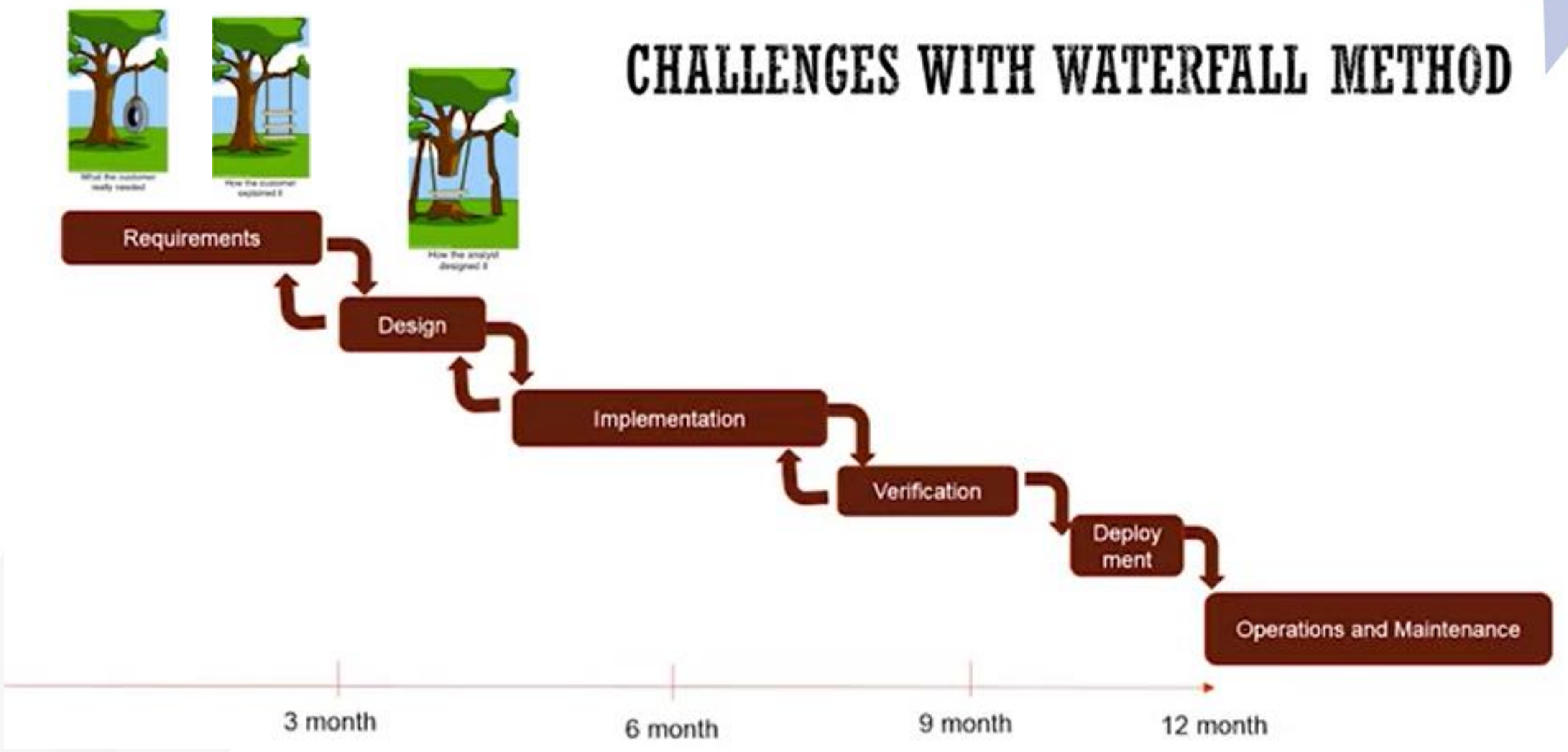
➢ So you may wonder that did we do a good job in requirement? Maybe we didn't spend enough time in the requirements

Why Agile?



- ➢ software industry was finding out that predicting requirements, or predicting user needs, is very difficult
- ➢ For example, suppose user need something like this swing

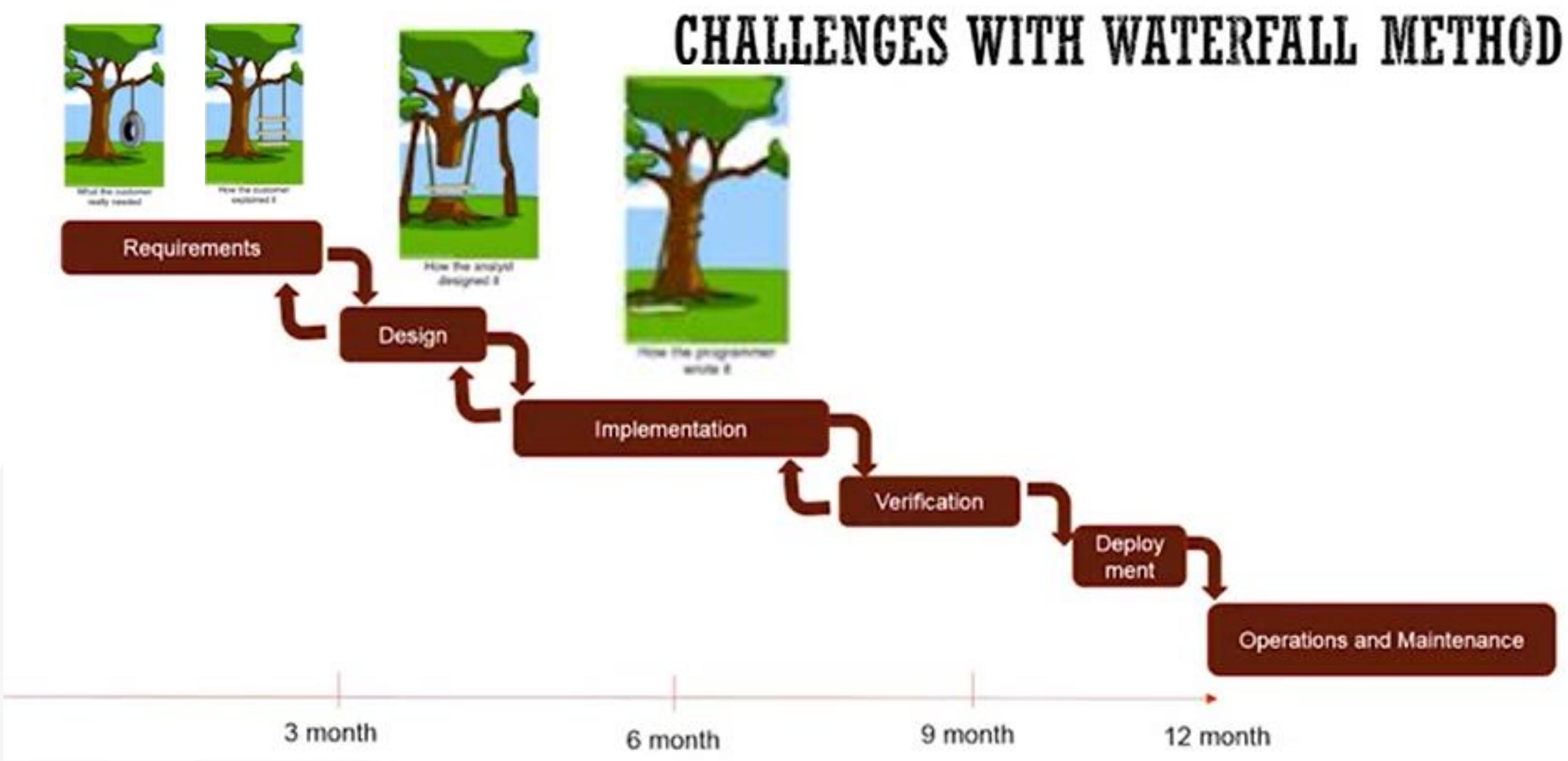
Why Agile?


CHALLENGES WITH WATERFALL METHOD
What the customer really needed
How the customer explained it
How the analyst designed it
Requirements
Design
Implementation
Verification
Deploy ment
Operations and Maintenance
3 month
6 month
9 month
12 month

➢ But we thought they actually needed this

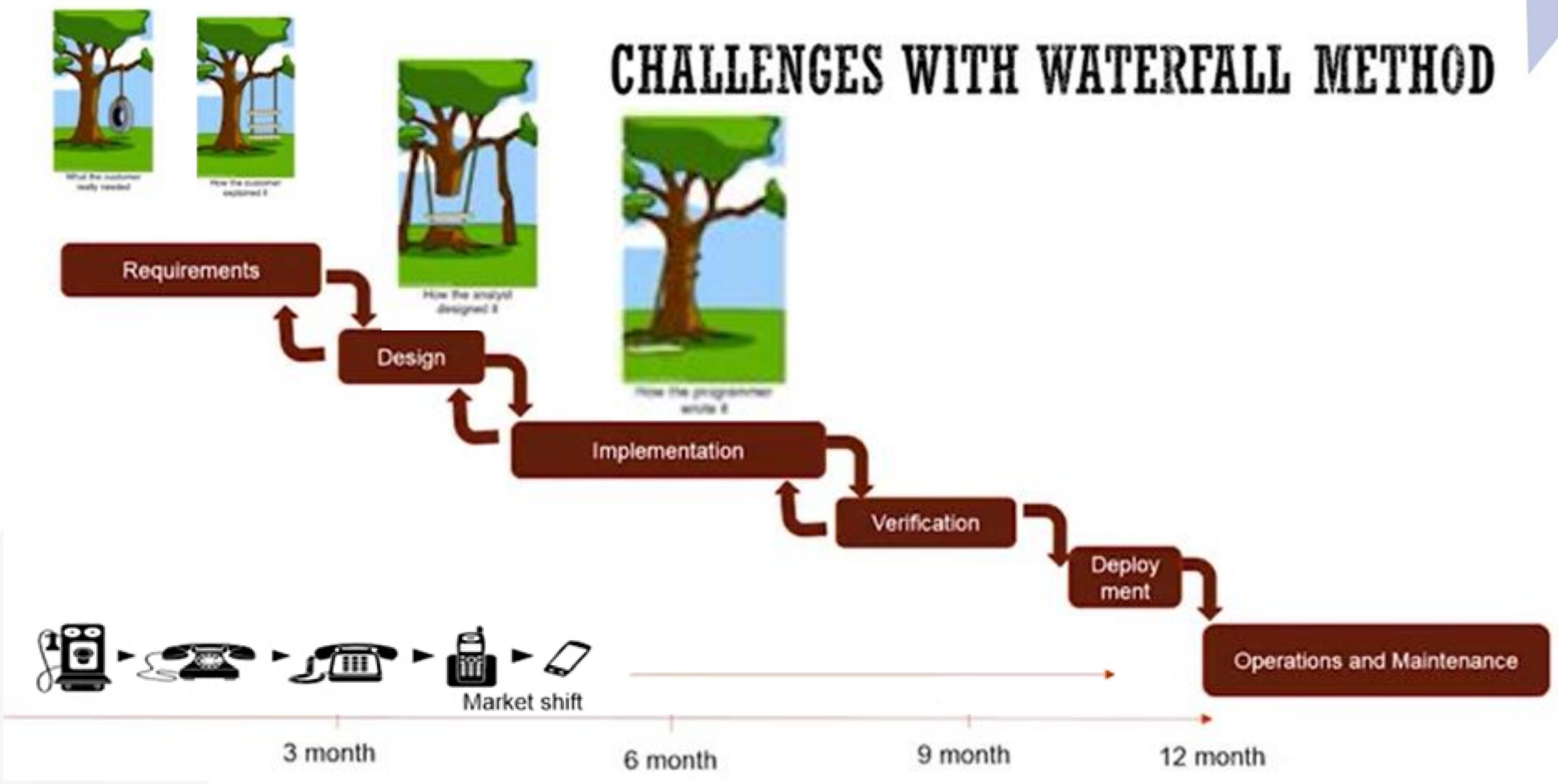
➢ Then the translation problems starts where the architect or the designer thinks actually user needed this,

Why Agile?


CHALLENGES WITH WATERFALL METHOD
What the customer really needed
How the customer explained it
How the analyst designed it
How the programmer wrote it
Requirements
Design
Implementation
Verification
Deploy ment
Operations and Maintenance
3 month
6 month
9 month
12 month

- ➢ They interpreted the requirements incorrectly.
- ➢ And then the developer looks at it and just say this doesn't make sense. They probably need this.

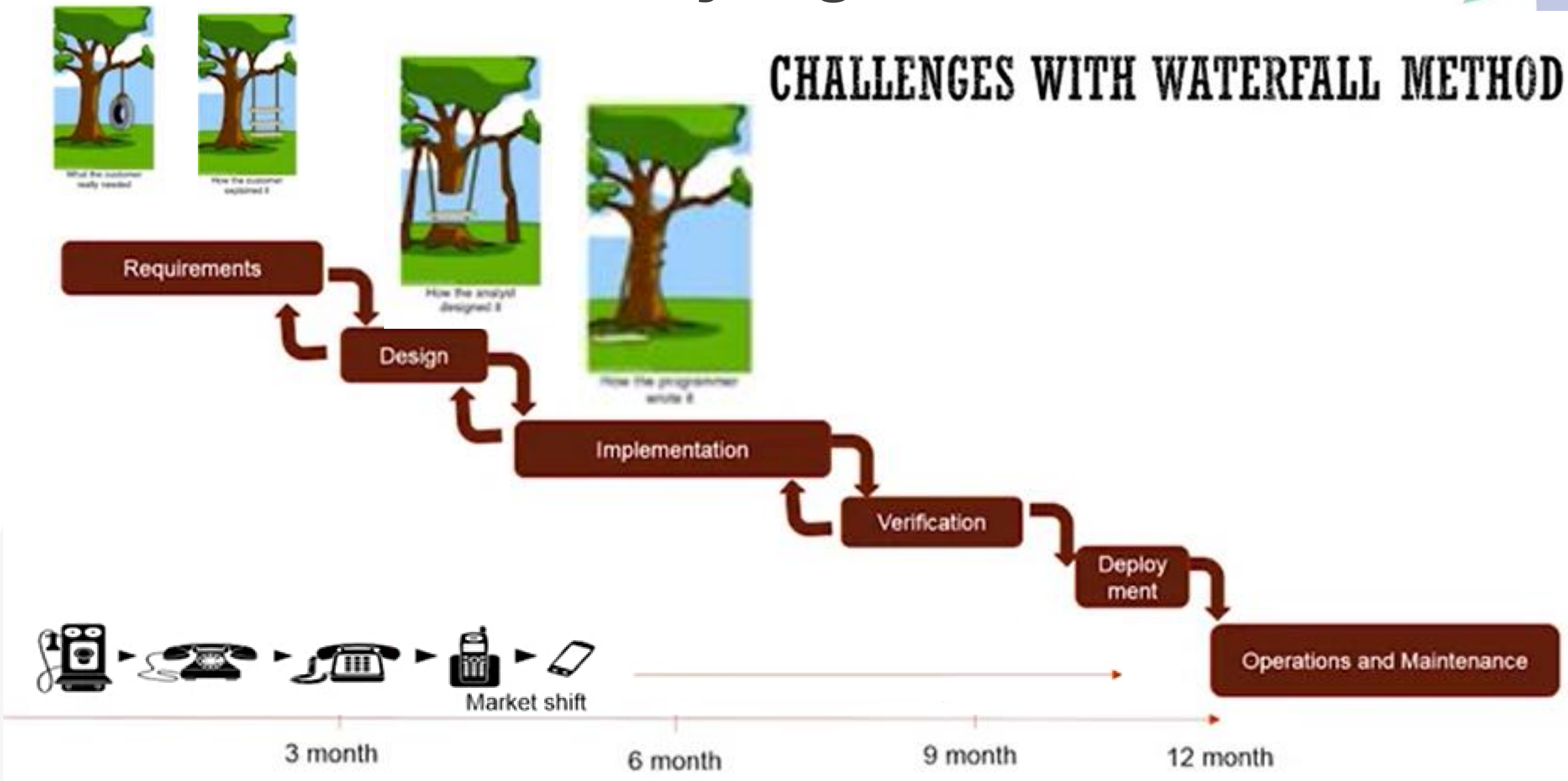
Why Agile?


CHALLENGES WITH WATERFALL METHOD
What the customer really needed
How the customer explained it
How the analyst designed it
How the programmer wrote it
Requirements
Design
Implementation
Verification
Deployment
Operations and Maintenance
Market shift
3 month
6 month
9 month
12 month

➢ So as you can see, the user needed something else, and we built something totally different.

➢ Another challenge is market shift

Why Agile?

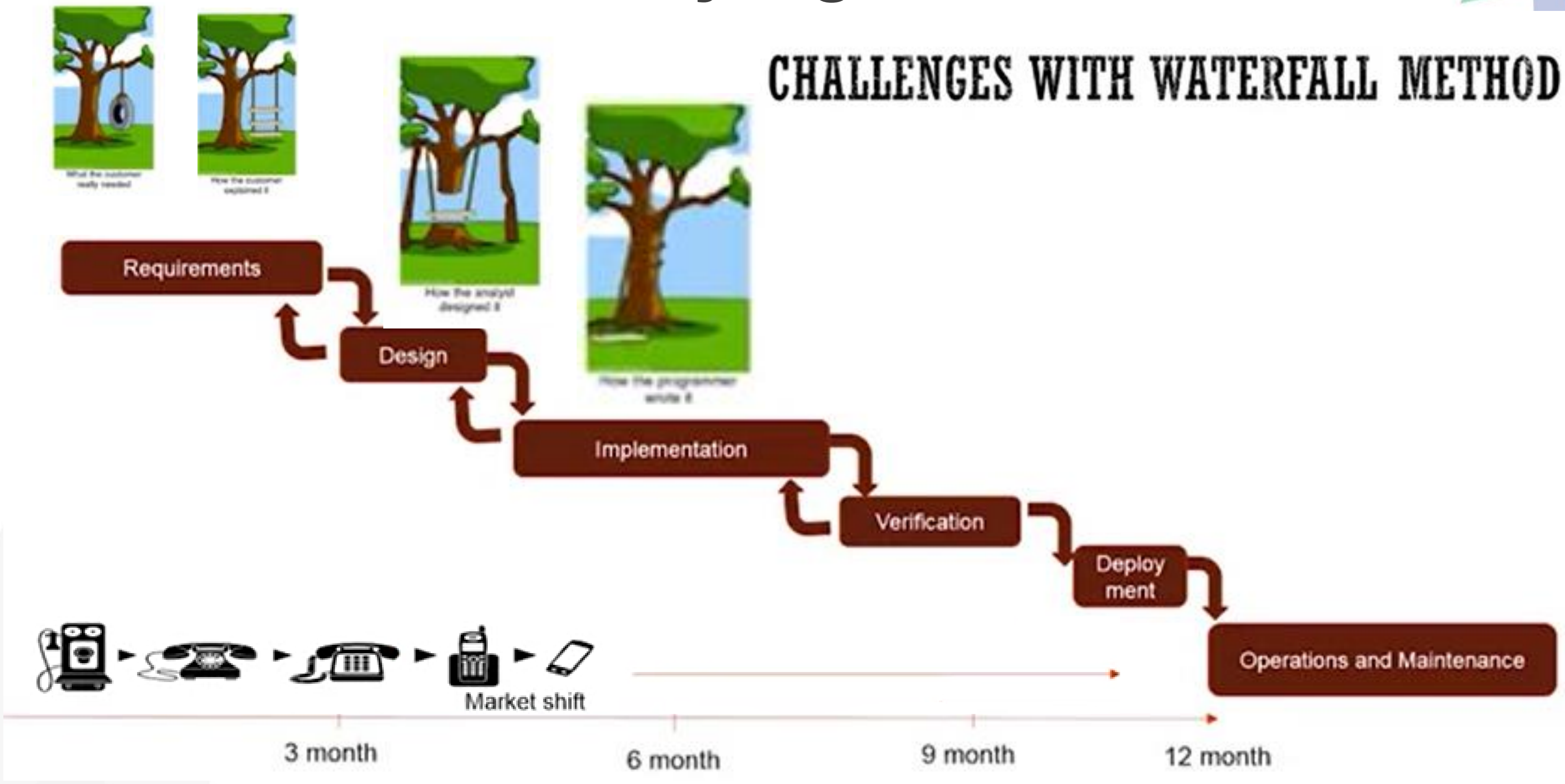


CHALLENGES WITH WATERFALL METHOD
What the customer really needed
How the customer explained it
How the analyst designed it
How the programmer wrote it
Requirements
Design
Implementation
Verification
Deploy ment
Operations and Maintenance
Market shift
3 month
6 month
9 month
12 month

➢ So, when we were writing the requirements, the needs were correctly identified, that this is what the user needs.

➢ But by the long time of software development, after we have done development, the market has shifted and the user doesn't need this anymore

Why Agile?



CHALLENGES WITH WATERFALL METHOD
What the customer really needed
How the customer explained it
How the analyst designed it
How the programmer wrote it
Requirements
Design
Implementation
Verification
Deploy ment
Operations and Maintenance
Market shift
3 month
6 month
9 month
12 month

- ➢ So, the problem is not that we are discovering these problems,
- ➢ but the problem is we are discovering these problems very late, which makes them really difficult to fix, or very expensive to fix.

Why Agile?

- As industry was finding these issues, several teams and several leaders were trying different methods like Crystal, XP, Scrum.
- And they were getting some success.
- The fundamental idea behind most of these models was to reduce the learning cycle
- How can we learn faster? And to increase the collaboration between the team members.

NECESSITY IS THE MOTHER OF INVENTION!

Why Agile?

- So as these teams and as these leaders were finding success in their methods, they thought, "let's get together and find out what is it that we are doing that is making us successful? And what is it that is common?"
- So these 17 individuals, they came together in February, 2001, in a ski resort and talked about each of their methods and said what is working and what is common in all of these methods?

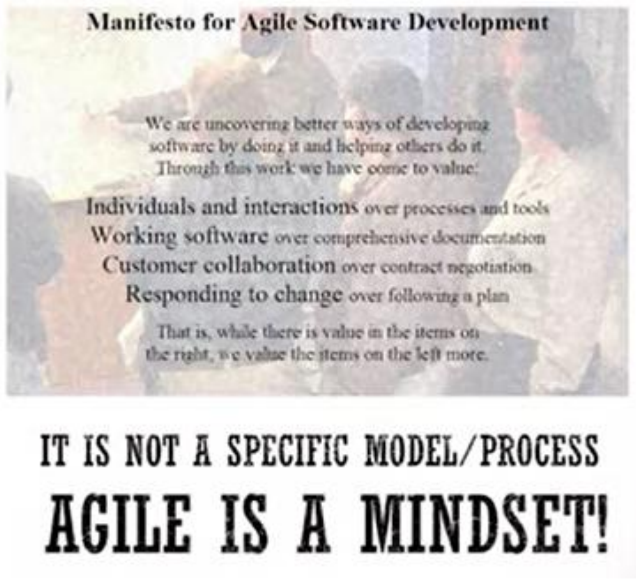
What came out of this meeting was symbolic - The manifesto for Agile soft ware development.

NECESSITY IS THE MOTHER OF INVENTION!

SUCCESS
DSDM
SUCCESS
FDD
SUCCESS
Crystal
SUCCESS
Custom...
SUCCESS
XP
SUCCESS
Scrum

Why Agile?

- The manifesto for Agile software development consists of four values and twelve principles.
- As you can see, they didn't define a new model or a process, they defined a mindset, an agile mindset, that helps teams build better software.


Manifesto for Agile Software Development

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

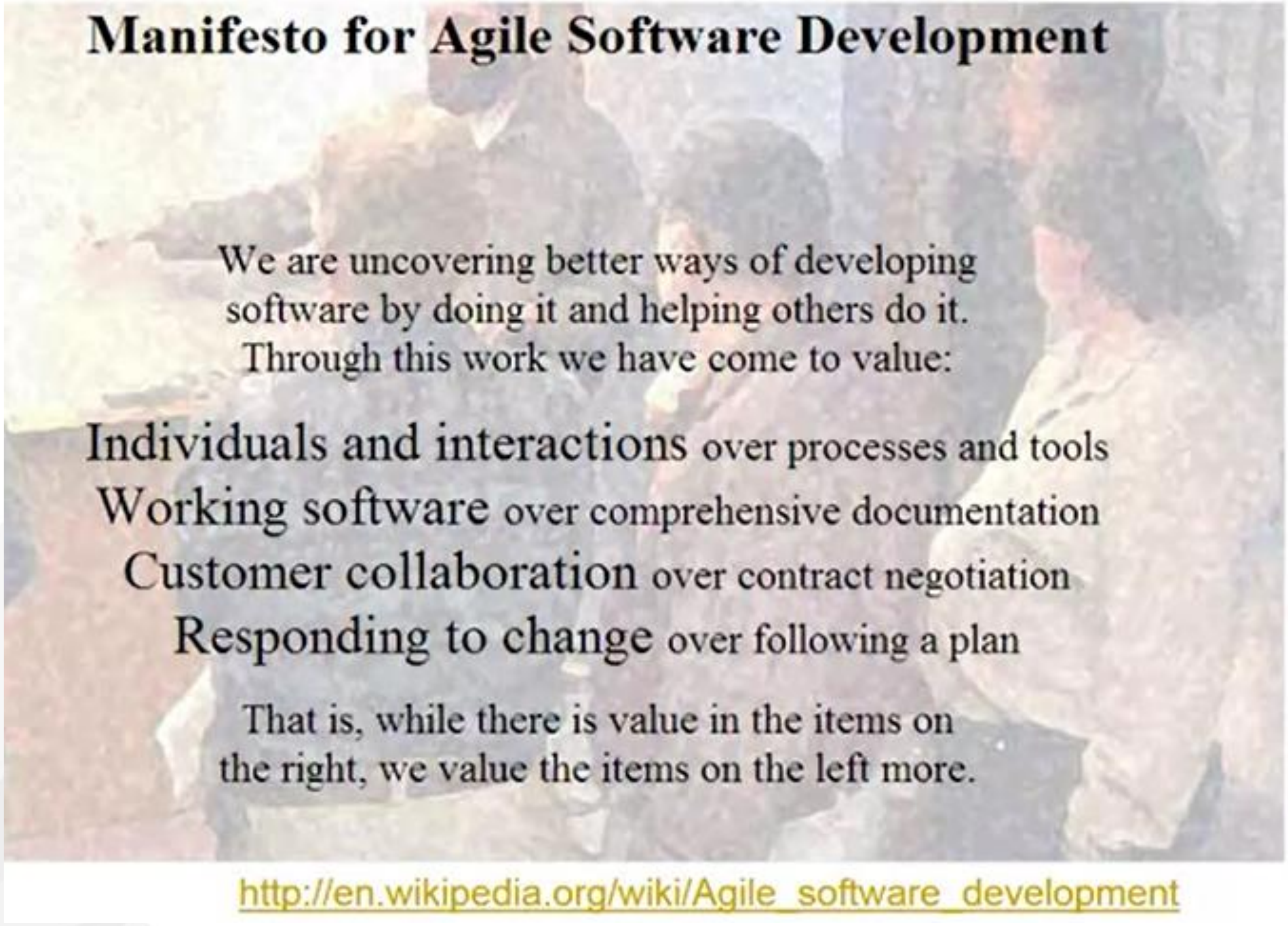
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

IT IS NOT A SPECIFIC MODEL/PROCESS

AGILE IS A MINDSET!

Manifesto of Agile Software Development



Manifesto for Agile Software Development

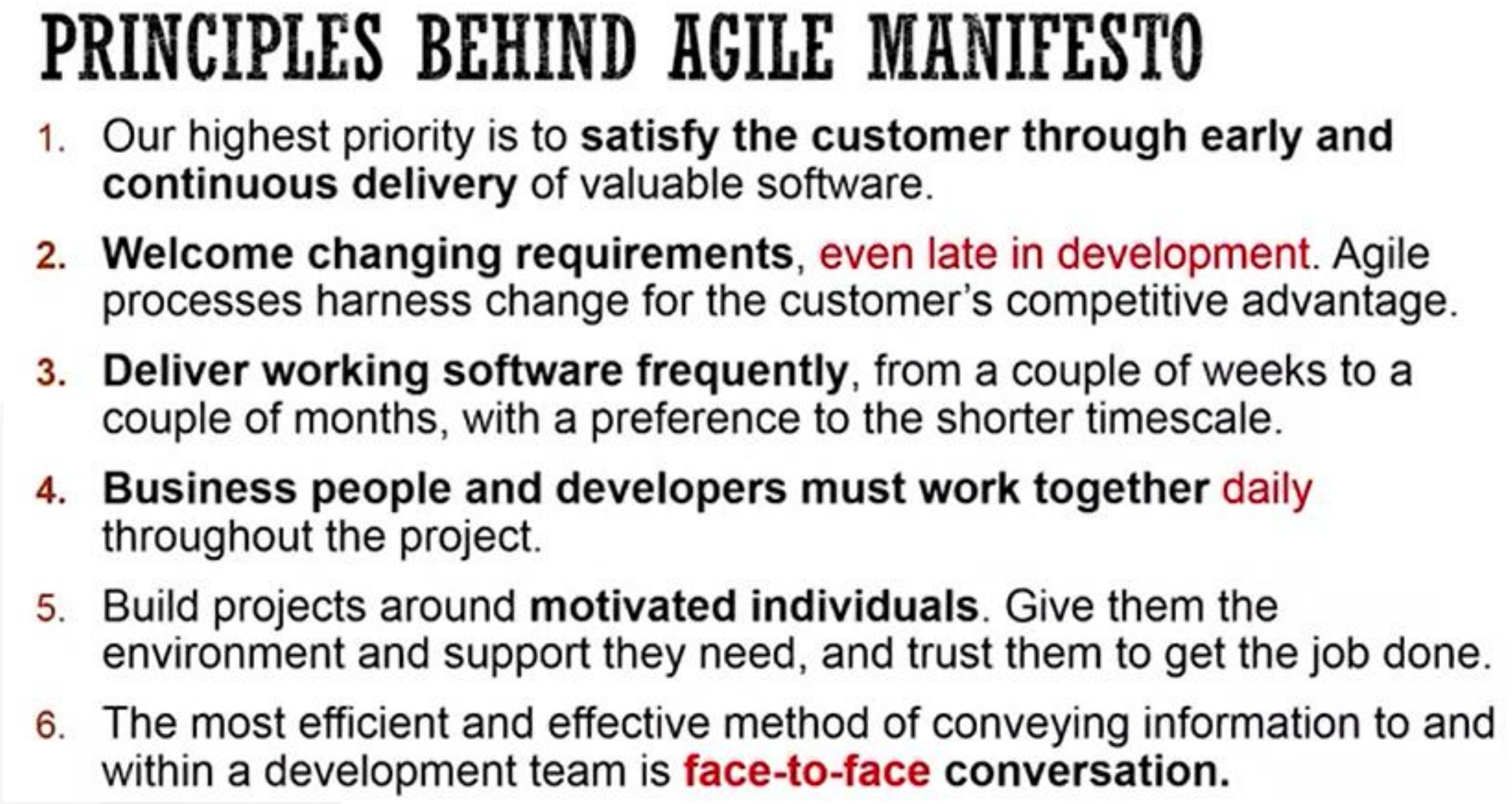
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

http://en.wikipedia.org/wiki/Agile_software_development

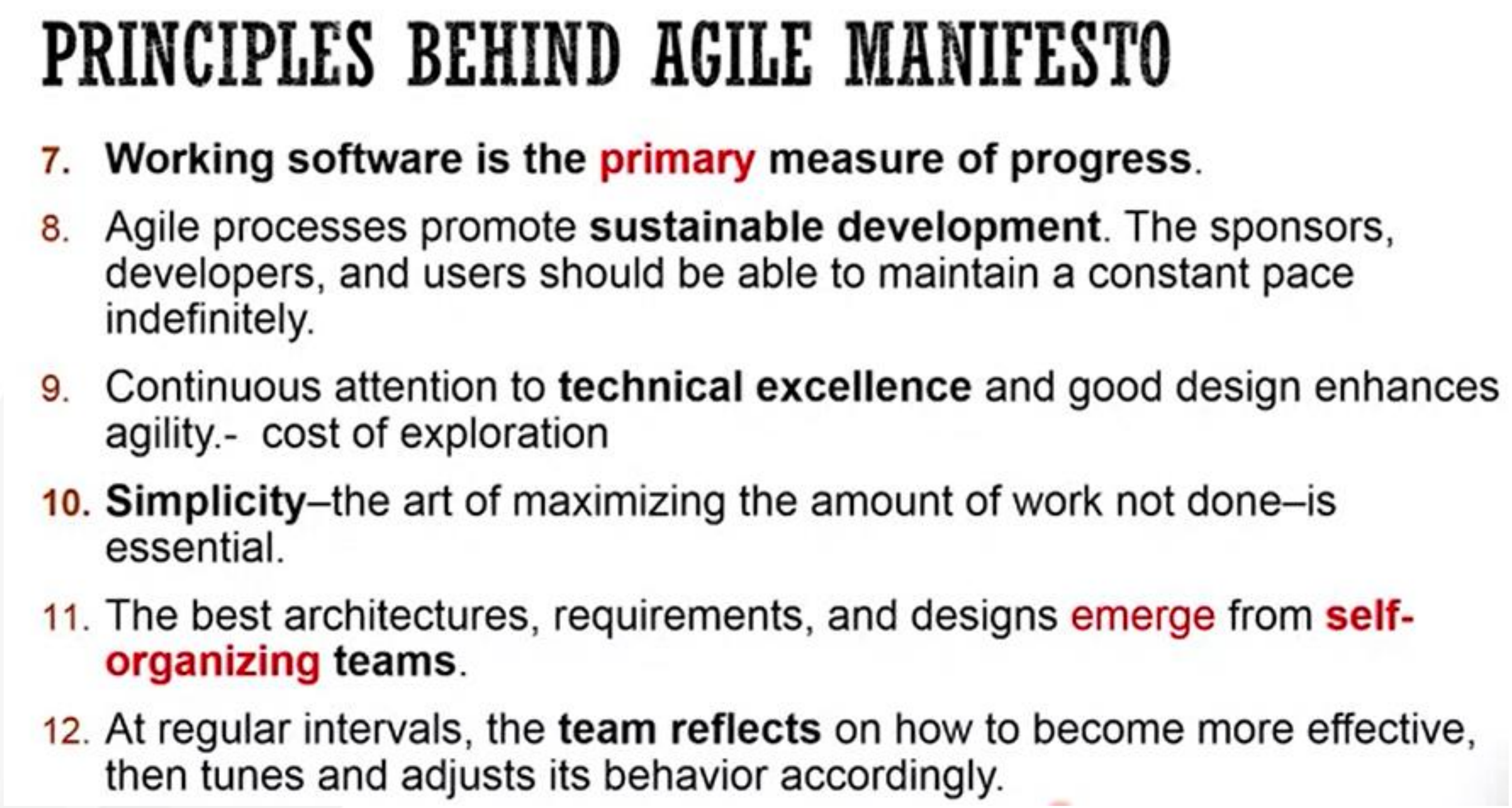
Manifesto of Agile Software Development



PRINCIPLES BEHIND AGILE MANIFESTO

1. Our highest priority is to **satisfy the customer through early and continuous delivery** of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together** daily throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation.**

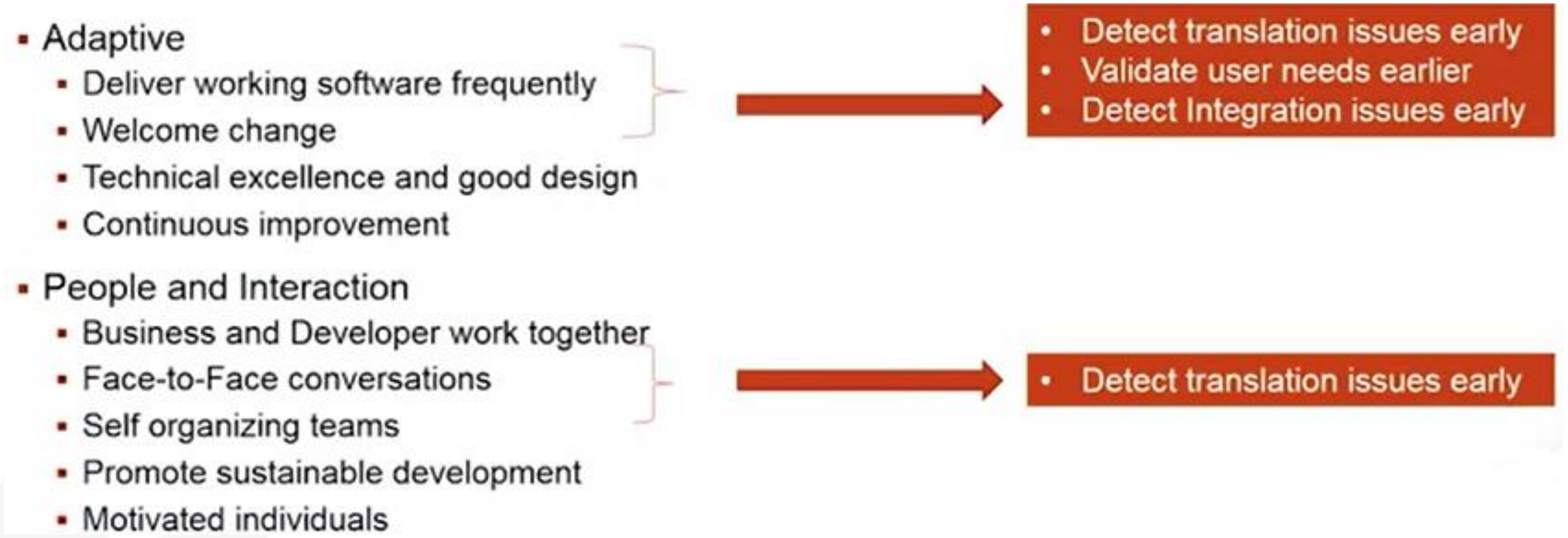
Manifesto of Agile Software Development



PRINCIPLES BEHIND AGILE MANIFESTO

7. **Working software is the primary measure of progress**.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence** and good design enhances agility.- cost of exploration
10. **Simplicity**–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

How does it solves problems of Traditional Software Models



- Adaptive
  - Deliver working software frequently
  - Welcome change
  - Technical excellence and good design
  - Continuous improvement
- People and Interaction
  - Business and Developer work together
  - Face-to-Face conversations
  - Self organizing teams
  - Promote sustainable development
  - Motivated individuals

- Detect translation issues early
- Validate user needs earlier
- Detect Integration issues early

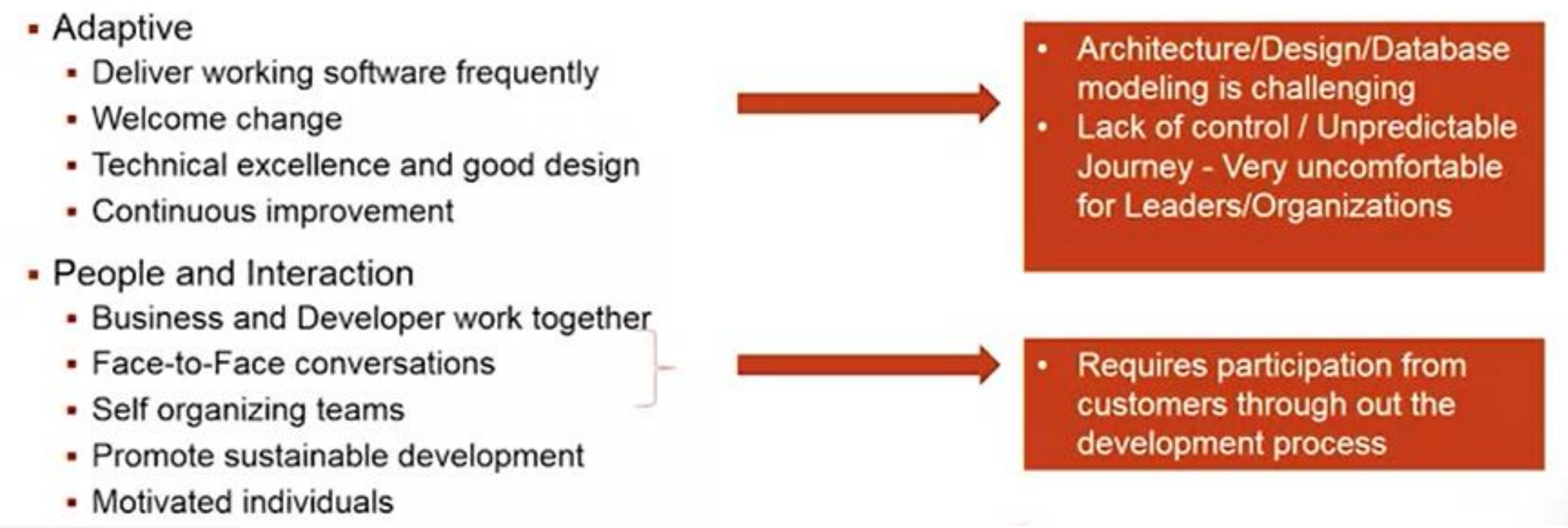
- Detect translation issues early

➢ All the principles can be divided into 2 categories:

➢ Adaptive nature of software development where you build some, you learn some, and so on

➢ Second one is collaboration

What new problems does it bring?



- Adaptive
  - Deliver working software frequently
  - Welcome change
  - Technical excellence and good design
  - Continuous improvement
- People and Interaction
  - Business and Developer work together
  - Face-to-Face conversations
  - Self organizing teams
  - Promote sustainable development
  - Motivated individuals

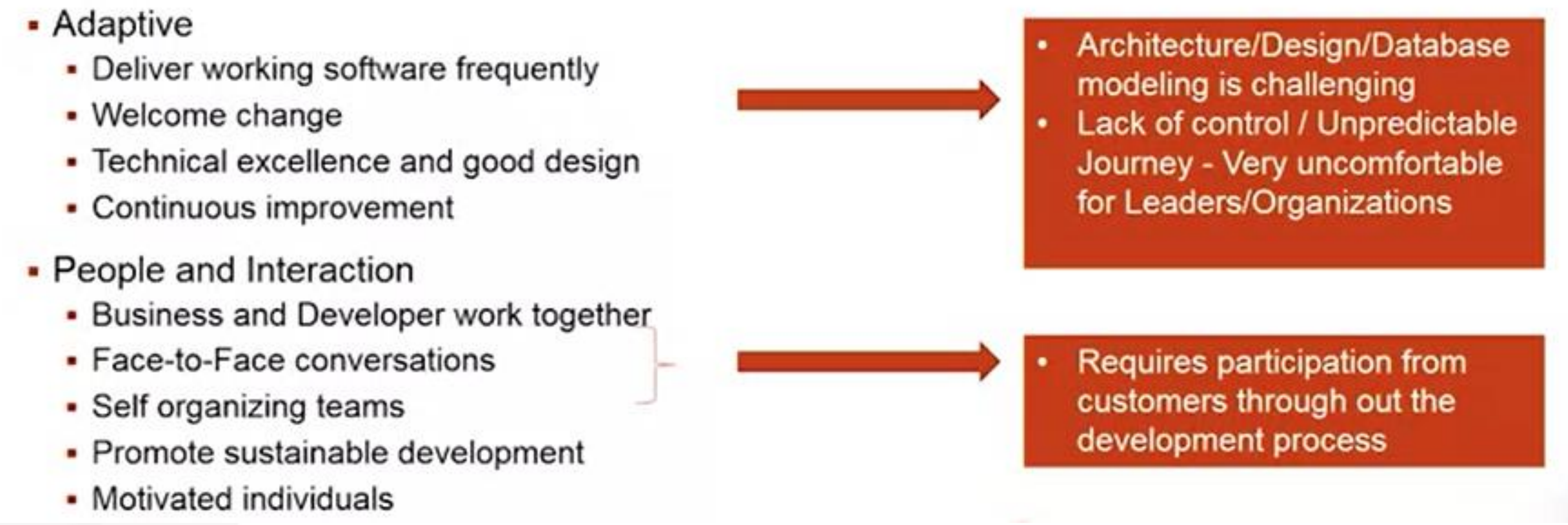
- Architecture/Design/Database modeling is challenging
- Lack of control / Unpredictable Journey - Very uncomfortable for Leaders/Organizations

- Requires participation from customers through out the development process

➢ Since we embrace change and we only look at part of the system to design sometimes the architecture, the design and the database modeling is very challenging.

➢ Since there is always this change, sometimes software engineers feel lack of control.

What new problems does it bring?



- Adaptive
  - Deliver working software frequently
  - Welcome change
  - Technical excellence and good design
  - Continuous improvement
- People and Interaction
  - Business and Developer work together
  - Face-to-Face conversations
  - Self organizing teams
  - Promote sustainable development
  - Motivated individuals

- Architecture/Design/Database modeling is challenging
- Lack of control / Unpredictable Journey - Very uncomfortable for Leaders/Organizations

- Requires participation from customers through out the development process

➢ There is a little bit of unpredictability in what you're building. That makes leaders of the organization a little bit uncomfortable.

➢ Another one is that since the customer has to be involved throughout the process, Software Engineers have to spend a little bit more time on the system.

Questions?

What are some of the challenges with Waterfall methods that prompted the software industry to come up with alternatives like Agile? Select three.

a) Predicting customer needs is difficult.

b) Project teams were geographically distributed.

c) Wrong implementation goes undetected for a long time.

d) Projects cost too much.

e) Integration issues between different components of the software go undetected for a long time. During the testing phase, when all the components are integrated, these issues are discovered but it is very late in the process.

Questions?

What are the four Agile values according to the Agile Manifesto?

a) Planning Properly over Just Executing

b) People and Interaction over Processes and Tools.

c) Responding to Change over Following a Plan

d) Working Software over Comprehensive Documentation

e) Customer Collaboration over Contract Negotiation

How do we apply Agile Mindset? Agile Frameworks

- Lots of frameworks that are available that you can use to apply these Agile mindsets for your team or for your project
- You have to customize this framework to meet the needs of your team, your project, or your organization
- The key to customization is to make sure that you stay true to the principles and the value of Agile

How do we apply Agile Mindset? Agile Frameworks

- Let's familiarize ourselves with some of these frameworks which are out there.
- One of the most common one is called **Scrum**
- Scrum is based **on one to four-week cycle** where **you take part of your project and you define, develop, design, and test your software**, and so your **product is developed incrementally**.
- **Another popular one** is called **Kanban**
- which is **based on a continuous flow model** where you basically try to optimize your existing software development process.

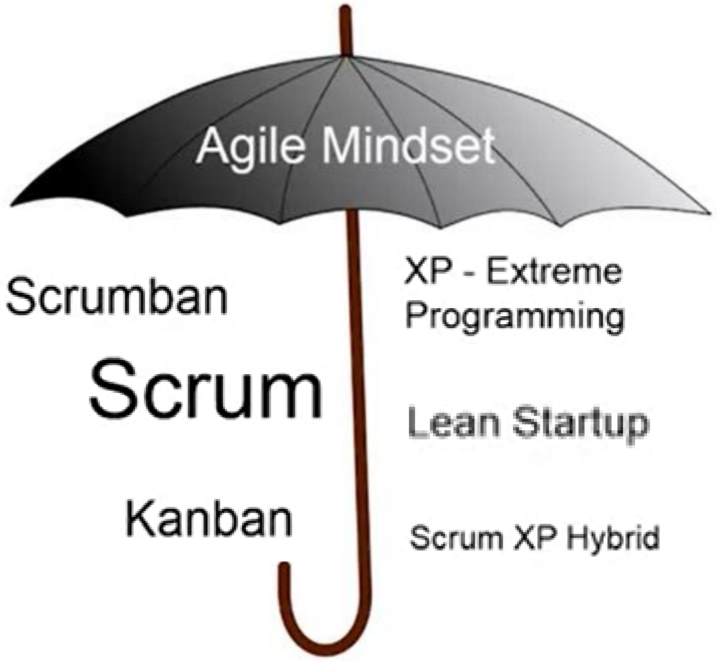
Agile Mindset
Scrum
Kanban

How do we apply Agile Mindset? Agile Frameworks

- There is a combination of these two called

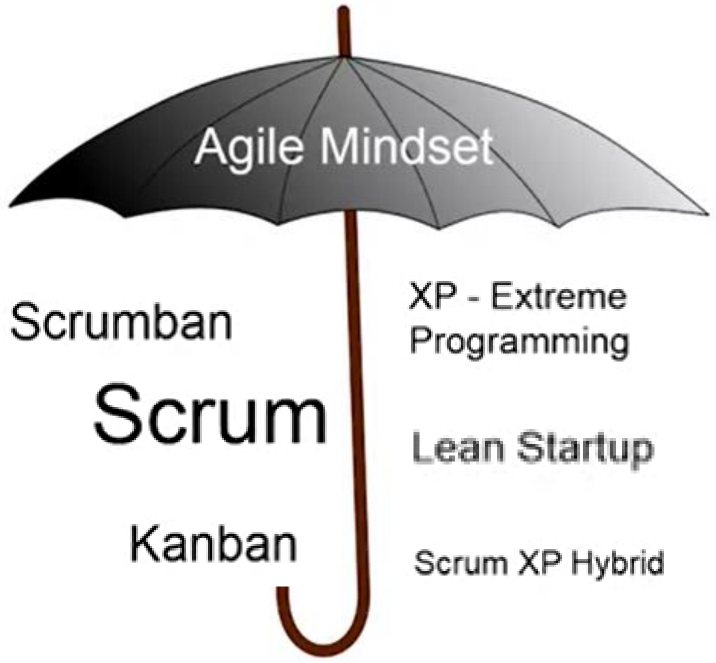
Scrumban

- where you use Scrum as your primary framework and then you use Kanban to optimize your flow within your sprint.
- **Another one** popular which is similar to Scrum is called **XP** and it has most of the practices of Scrum but it also defines some engineering practices which are very crucial for an Agile team.
- Then, of course, **there is the hybrid of a Scrum and XP**.
- **Another** one is called **Lean** Startup which helps you if you have a lot of unpredictable market or industry and you want to really prove your solution before you implement it.

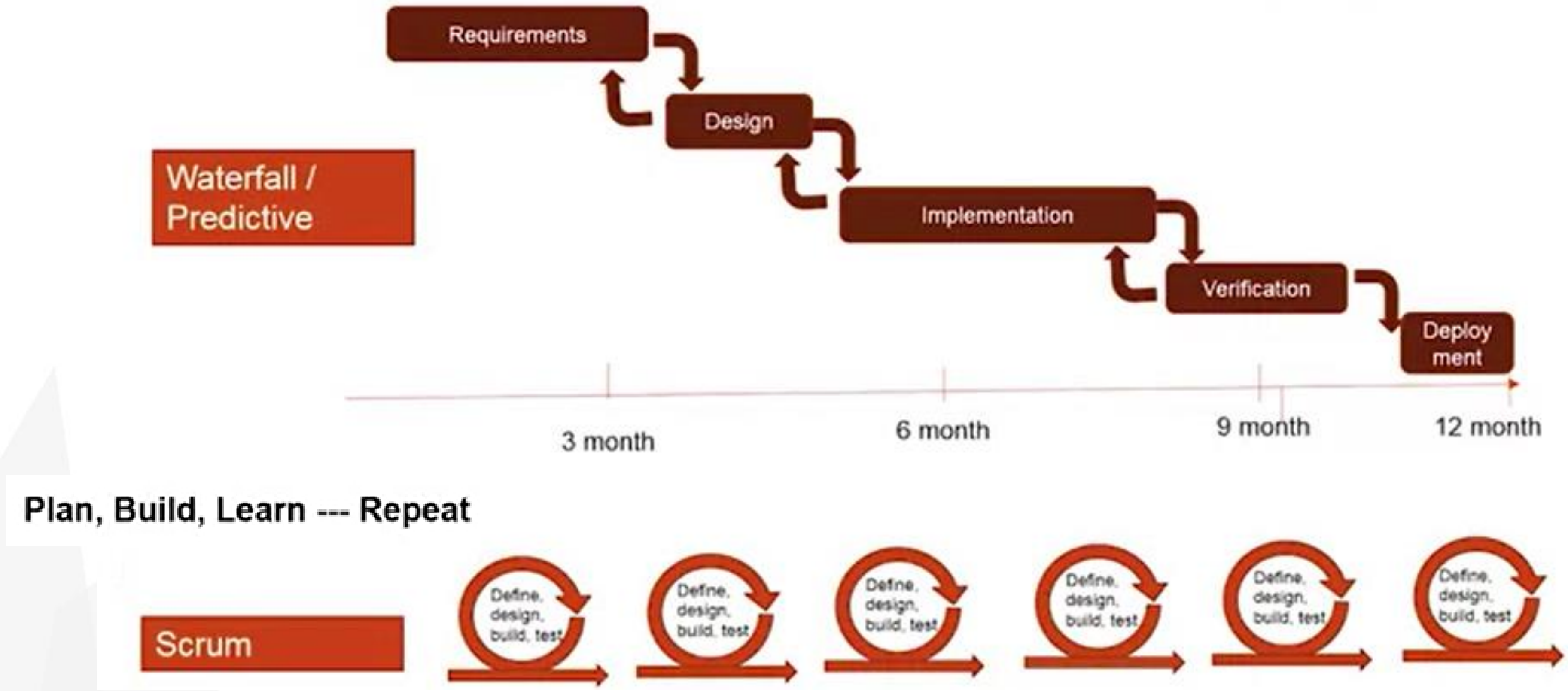


How do we apply Agile Mindset? Agile Frameworks

- ➢ There are many more agile frameworks
- ➢ More often, organizations actually end up customizing some of these frameworks to meet the need of their team or their project or the organization
- ➢ In terms of popularity, **Scrum is by far the most popular of all**.
- ➢ It's about **70% of the Agile teams use either Scrum or one of these variants**.
- ➢ That's how sometimes people equate Agile to Scrum which you, of course, you can see that it's not true



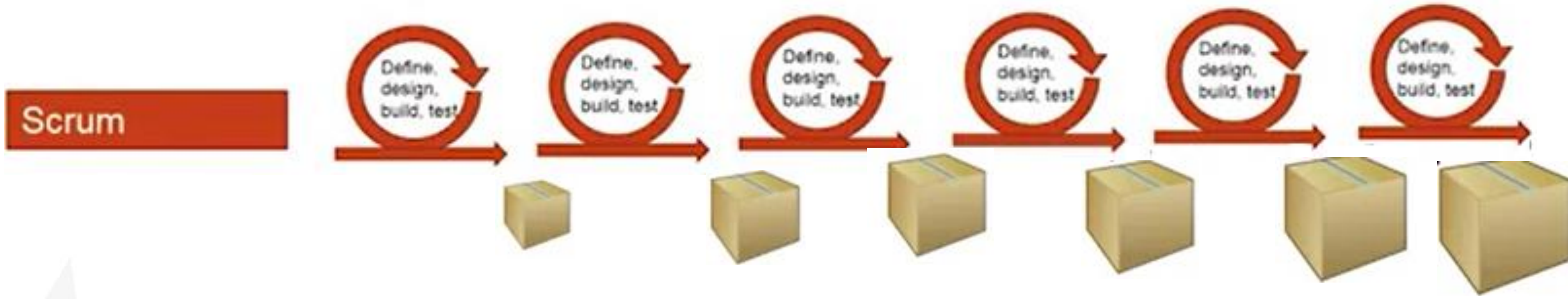
Scrum Framework



➢ To understand the basic idea behind Scrum, let's put Scrum next to a Waterfall method.

➢ So Scrum works on this one to four week sprint where you take part of your product and you define, you design, you build and you test.

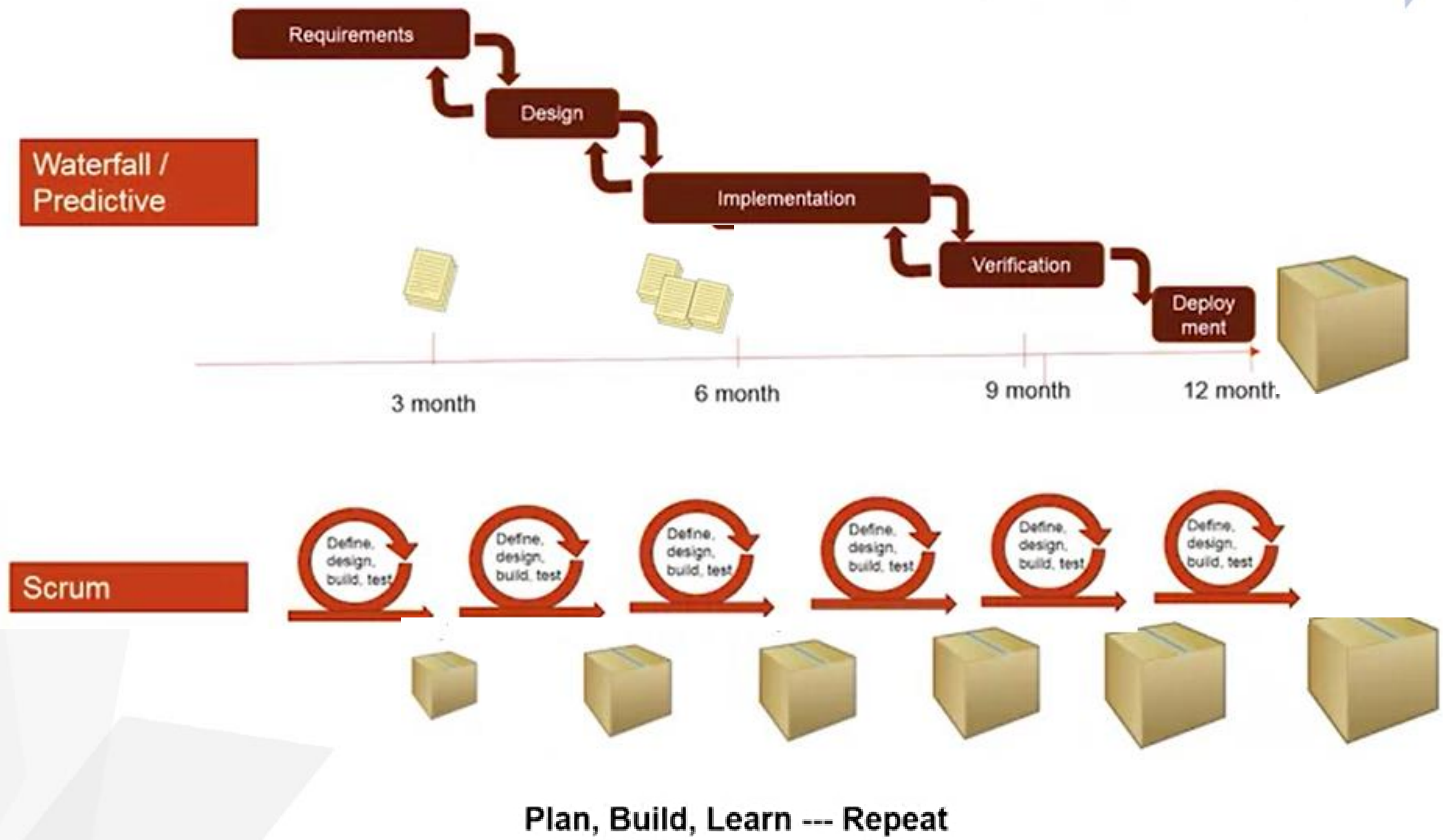
Scrum Framework


Scrum
Define, design, build, test
Define, design, build, test
Define, design, build, test
Define, design, build, test
Define, design, build, test
Define, design, build, test

Plan, Build, Learn --- Repeat

- Then you show your product to your stakeholders and see if you need to adjust your product or if you need to do something different.
- And then again you repeat the cycle again and again. And so this way you are building your product incrementally after every sprint and then at the end, you get all of your product.

Scrum Framework


Waterfall / Predictive
Requirements
Design
Implementation
Verification
Deploy ment
3 month
6 month
9 month
12 month
Scrum
Define, design, build, test
Define, design, build, test
Define, design, build, test
Define, design, build, test
Define, design, build, test
Define, design, build, test
Plan, Build, Learn --- Repeat

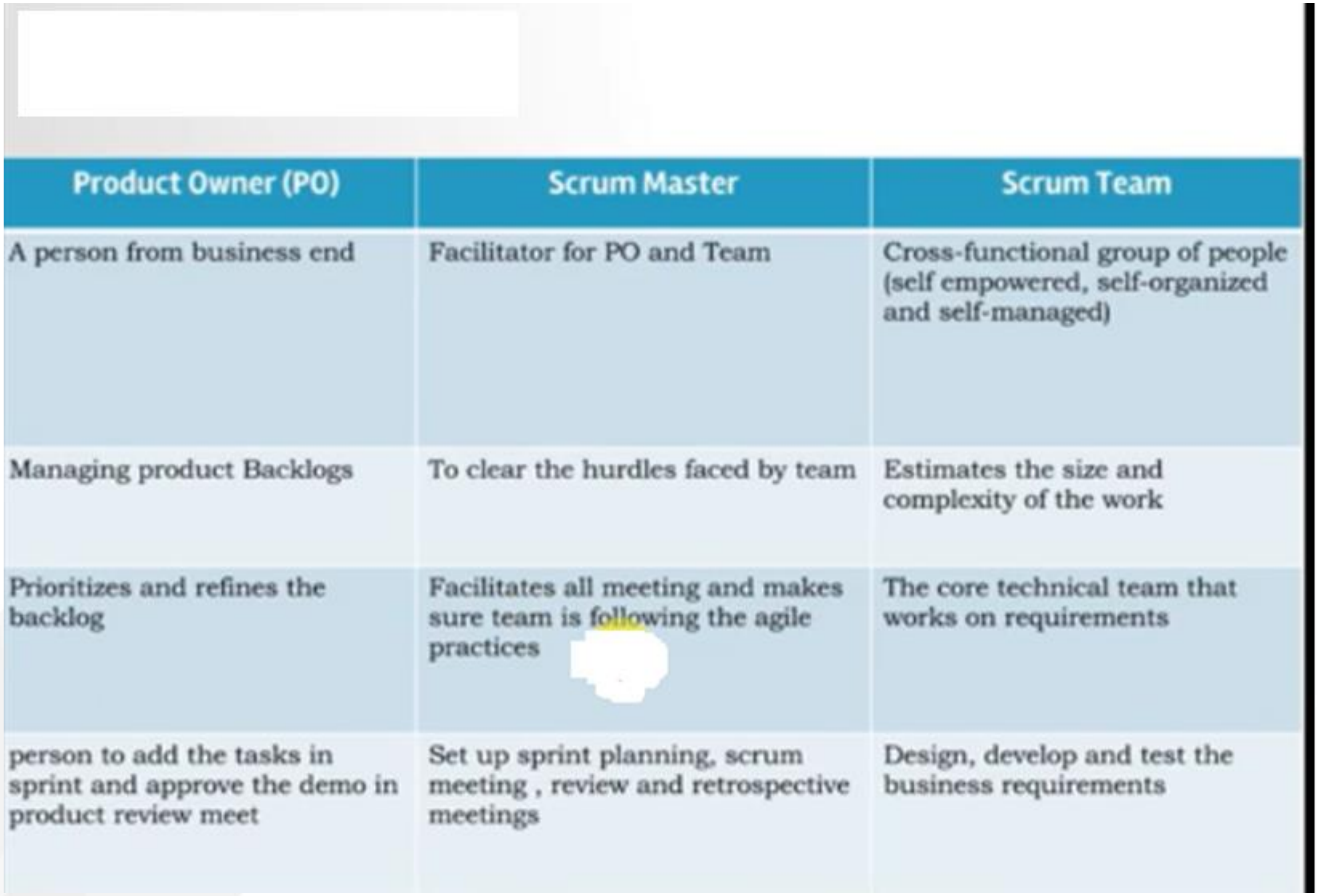
Scrum Framework

- So let's **take an example** of a company that wants **to build a job website**, let's say like Monster.com,
- they want to use a Scrum framework. There are **three roles** that are defined in Scrum.
- **First one is product owner** who defines what needs to be done and in what order. Second is **Scrum master, who** helps the team stay true to the Scrum values and principles.

Scrum Framework

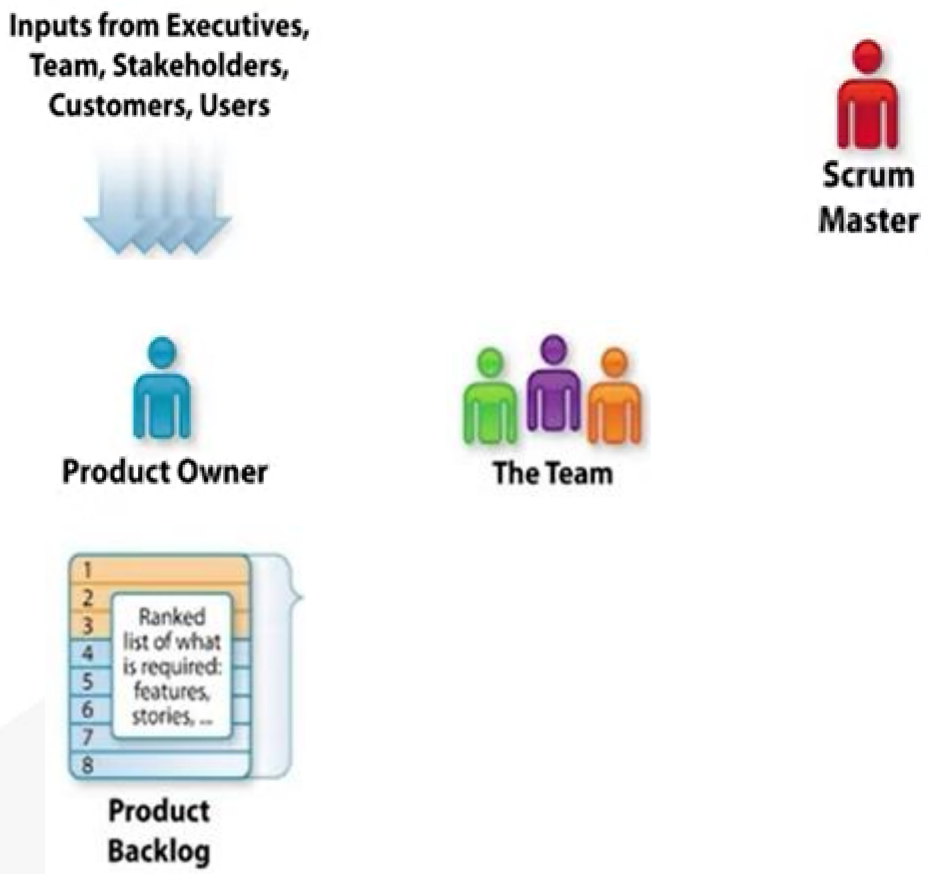
- **Scrum master** helps facilitate most of the meetings in the team. And then if there is any road blocks, then he'll be the one who will drive there solution of some of these road blocks.
- Third is **the team** that is self organizing and they do most of the building of the software.
- **The team includes developers and testers and everyone**.

Scrum



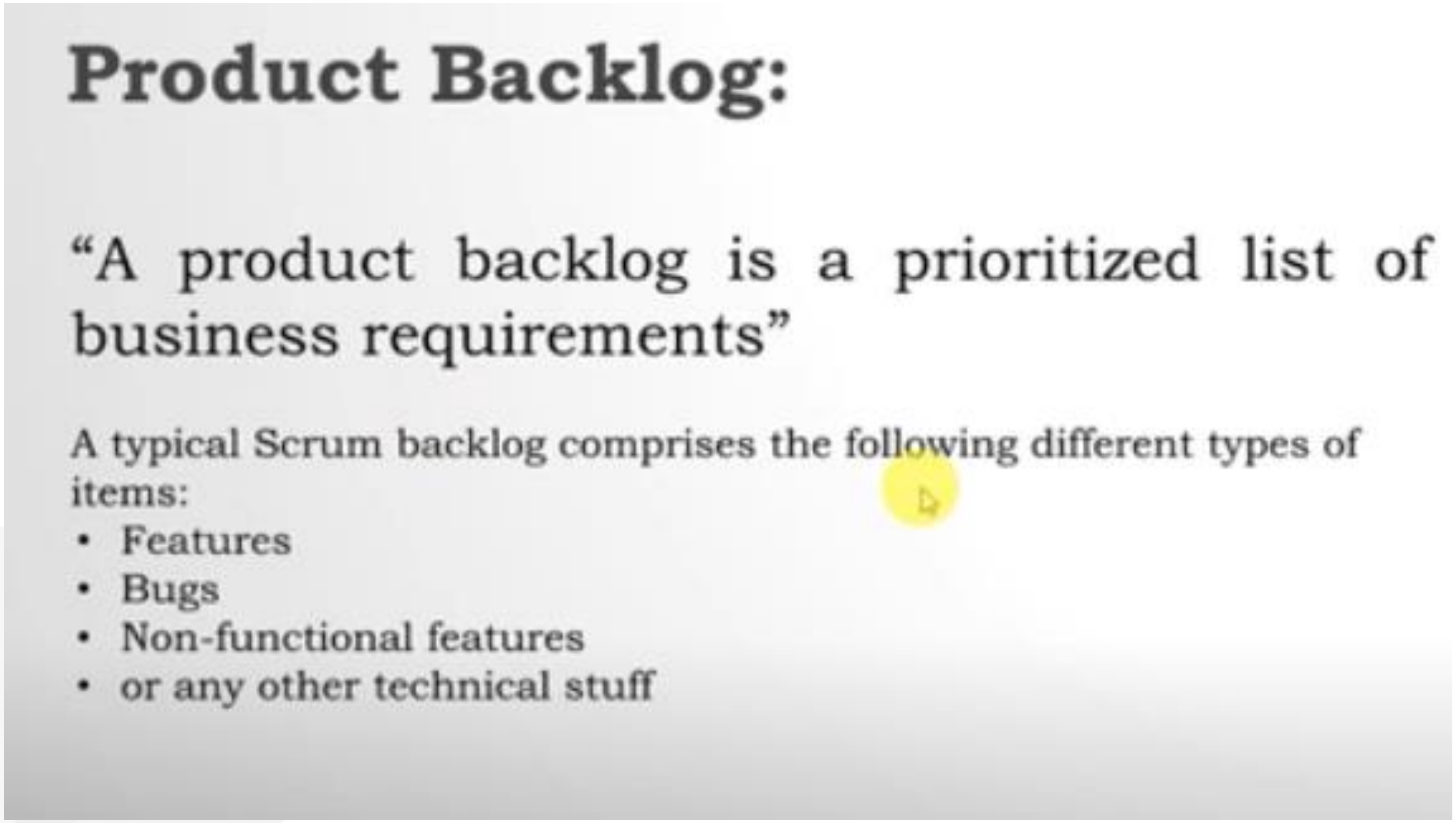
Product Owner (PO)	Scrum Master	Scrum Team
A person from business end	Facilitator for PO and Team	Cross-functional group of people (self empowered, self-organized and self-managed)
Managing product Backlogs	To clear the hurdles faced by team	Estimates the size and complexity of the work
Prioritizes and refines the backlog	Facilitates all meeting and makes sure team is following the agile practices	The core technical team that works on requirements
person to add the tasks in sprint and approve the demo in product review meet	Set up sprint planning, scrum meeting , review and retrospective meetings	Design, develop and test the business requirements

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Scrum Master
Product Owner
The Team
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog

- Let's see how the job website will be created using a Scrum framework.
- So **product owner** is going **to talk** to the executives, the team, the stakeholders, clients, users and will try **to define what exactly needs to be built**.
- That will create something **called a product backlog** which **is basically a list of user stories which are prioritized and defines what needs to be done**.

Scrum Framework



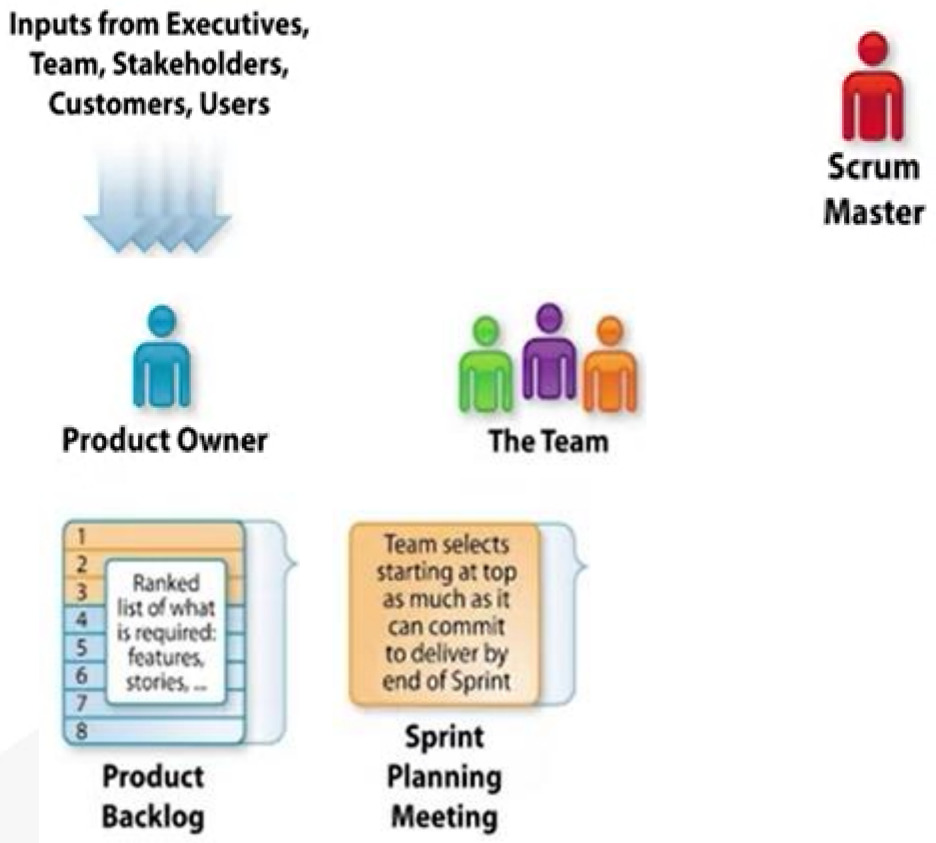
Product Backlog:

"A product backlog is a prioritized list of business requirements"

A typical Scrum backlog comprises the following different types of items:

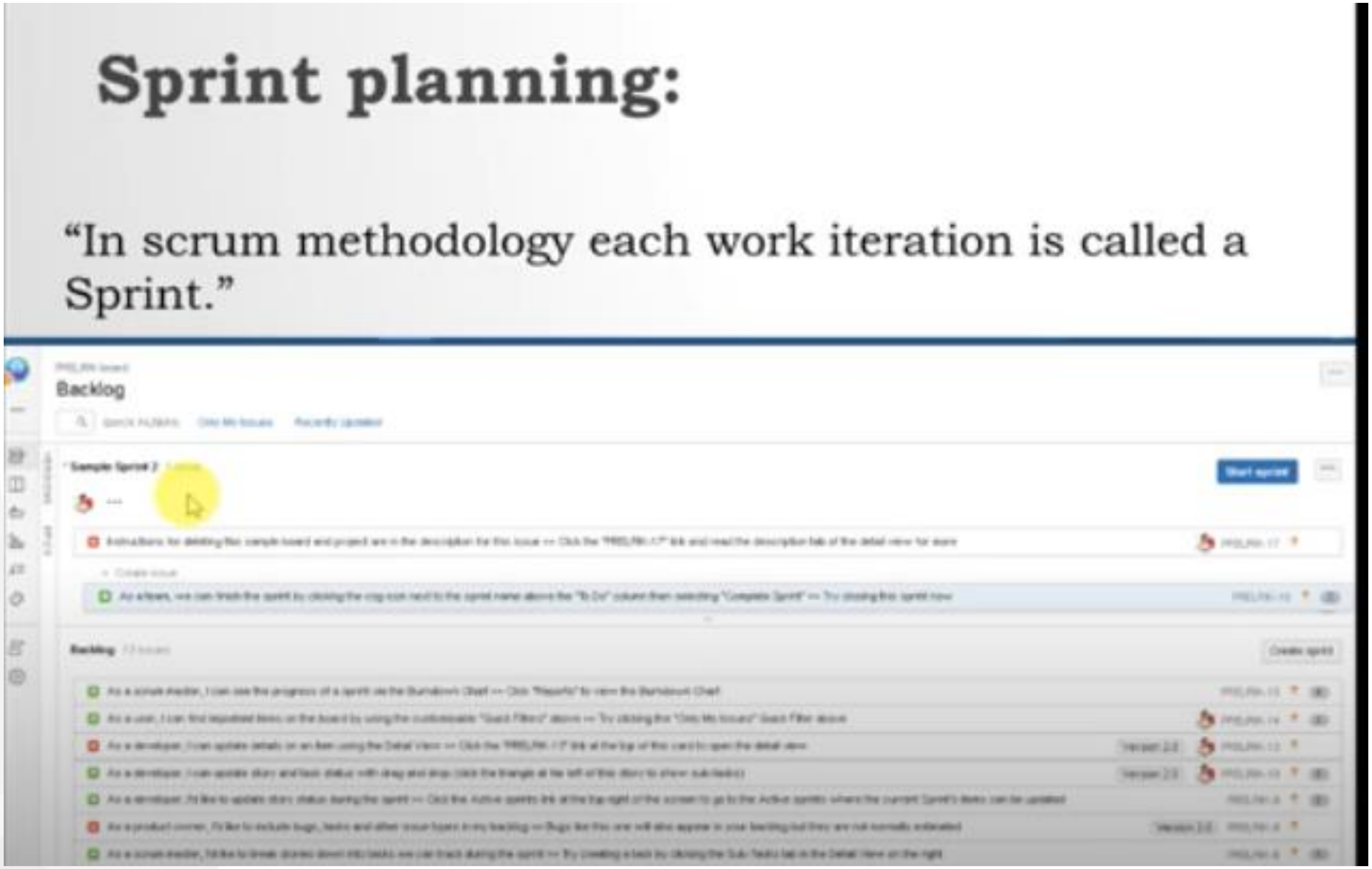
- Features
- Bugs
- Non-functional features
- or any other technical stuff

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Scrum Master
Product Owner
The Team
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting

- ➢ So **once the team is ready to sprint**, they get together for **a meeting called Sprint Planning Meeting**, where the whole team gets together.
- ➢ Then they **pick the top stories that they can work on in the existing sprint**.
- ➢ The **product owner reviews those stories** with the team and helps answer any questions or clarifies anything that is not clear.

Scrum Framework



Sprint planning:

"In scrum methodology each work iteration is called a Sprint."

Backlog

Sample Sprint 2

Start sprint

Instructions for deleting this sample board and project are in the description for this issue >> Click the "PRELIM-17" link and read the description tab of the detail view for more

Create issue

As a team, we can finish the sprint by clicking the cog icon next to the sprint name above the "To Do" column then selecting "Complete Sprint" >> Try closing this sprint now

Backlog

Create sprint

As a scrum master, I can see the progress of a sprint via the Burndown Chart >> Click "Reports" to view the Burndown Chart

As a user, I can find important items on the board by using the customisable "Quick Filters" above >> Try clicking the "Only My Issues" Quick Filter above

As a developer, I can update details on an item using the Detail View >> Click the "PRELIM-17" link at the top of this card to open the detail view

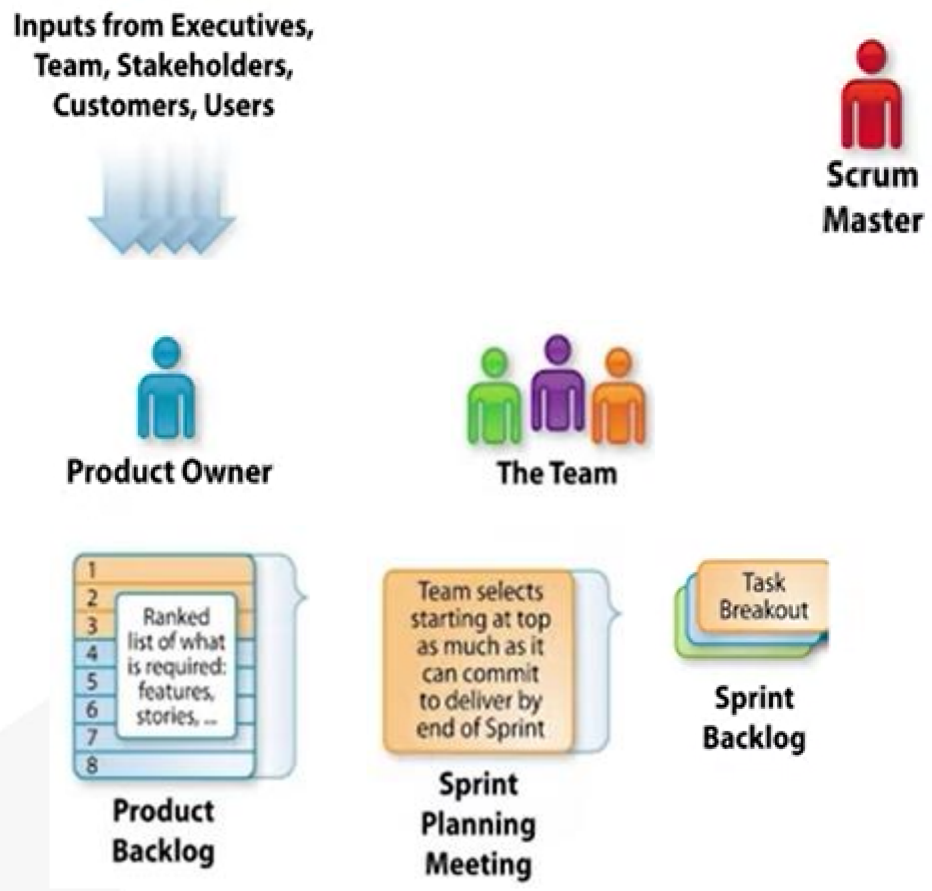
As a developer, I can update story and task status with drag and drop (click the triangle at far left of this story to show sub-tasks)

As a developer, I'd like to update story status during the sprint >> Click the Active sprints link at the top right of the screen to go to the Active sprints where the current Sprint's items can be updated

As a product owner, I'd like to include bugs, tasks and other issue types in my backlog >> Bugs like this one will also appear in your backlog but they are not normally estimated

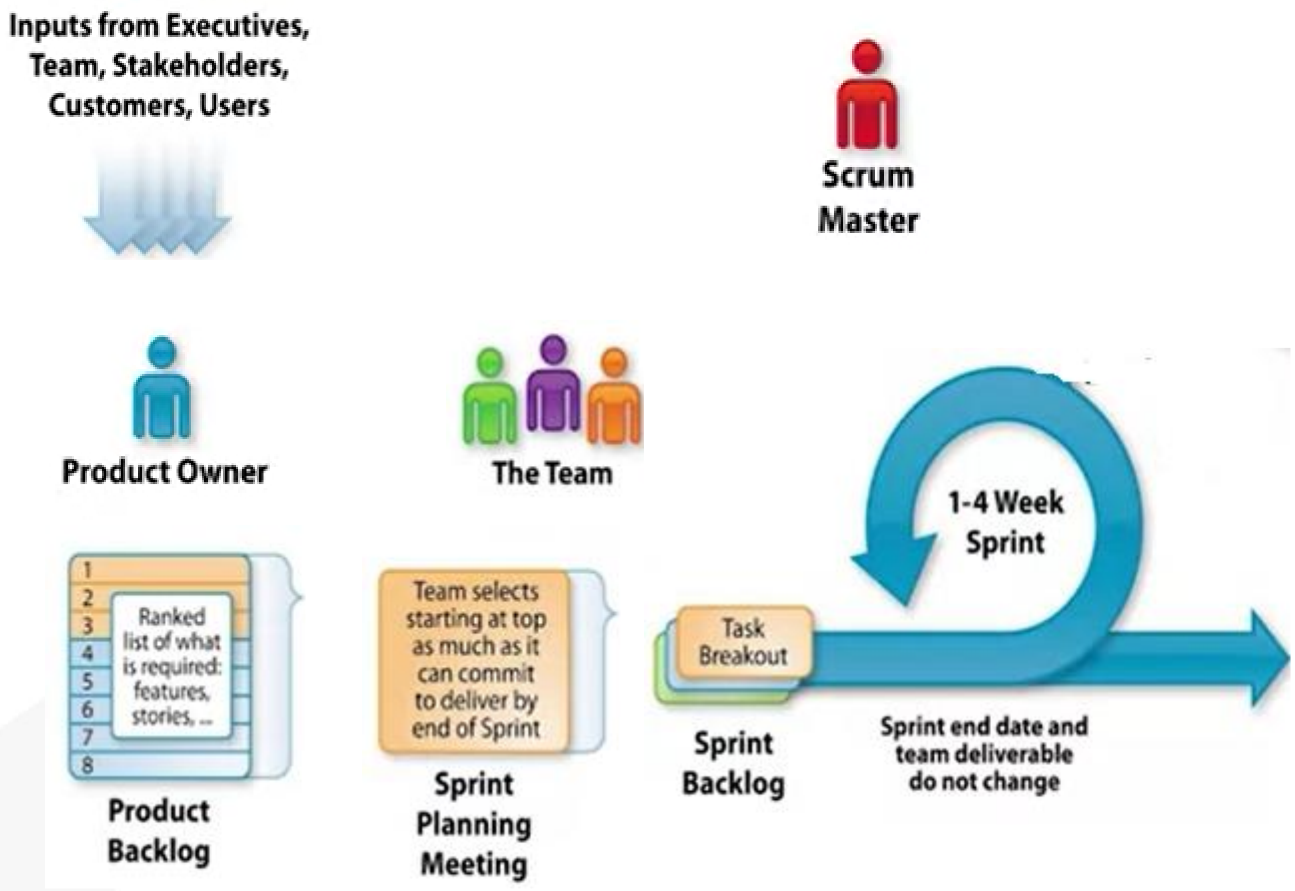
As a scrum master, I'd like to break stories down into tasks we can track during the sprint >> Try creating a task by clicking the Sub-Tasks tab in the Detail View on the right

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Scrum Master
Product Owner
The Team
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog

- Then **the team gets together again and then do a tasking out of the stories**, which means, like what exactly we need to do to build the software
- So in this case, **they may say, well I need to design this database. I need to fill some data and test it on the database. I need to create this UI screen and so on.**

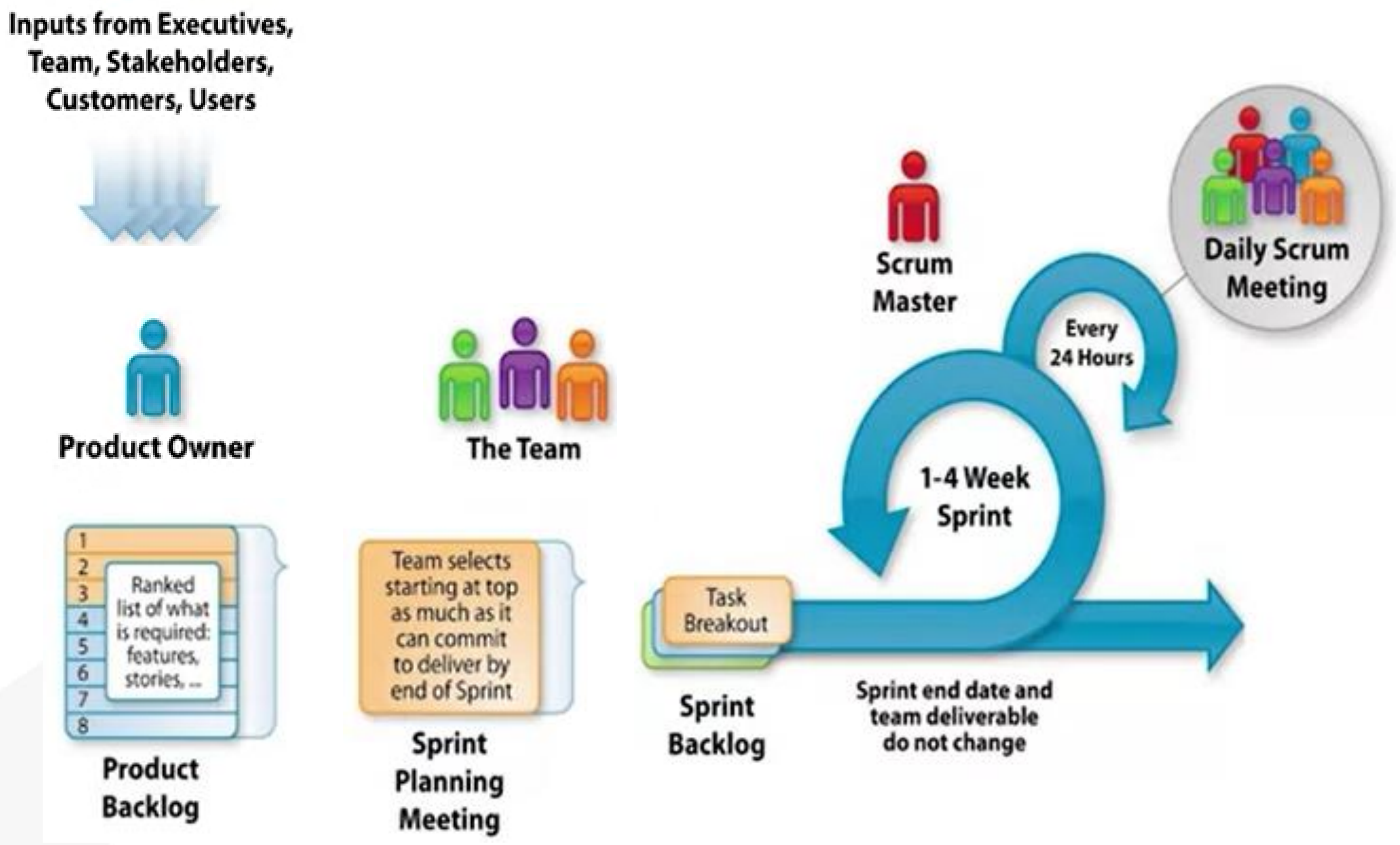
Scrum Framework



Inputs from Executives, Team, Stakeholders, Customers, Users
Scrum Master
Product Owner
The Team
1-4 Week Sprint
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Team selects starting at top as much as it can commit to deliver by end of Sprint
Task Breakout
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Sprint end date and team deliverable do not change

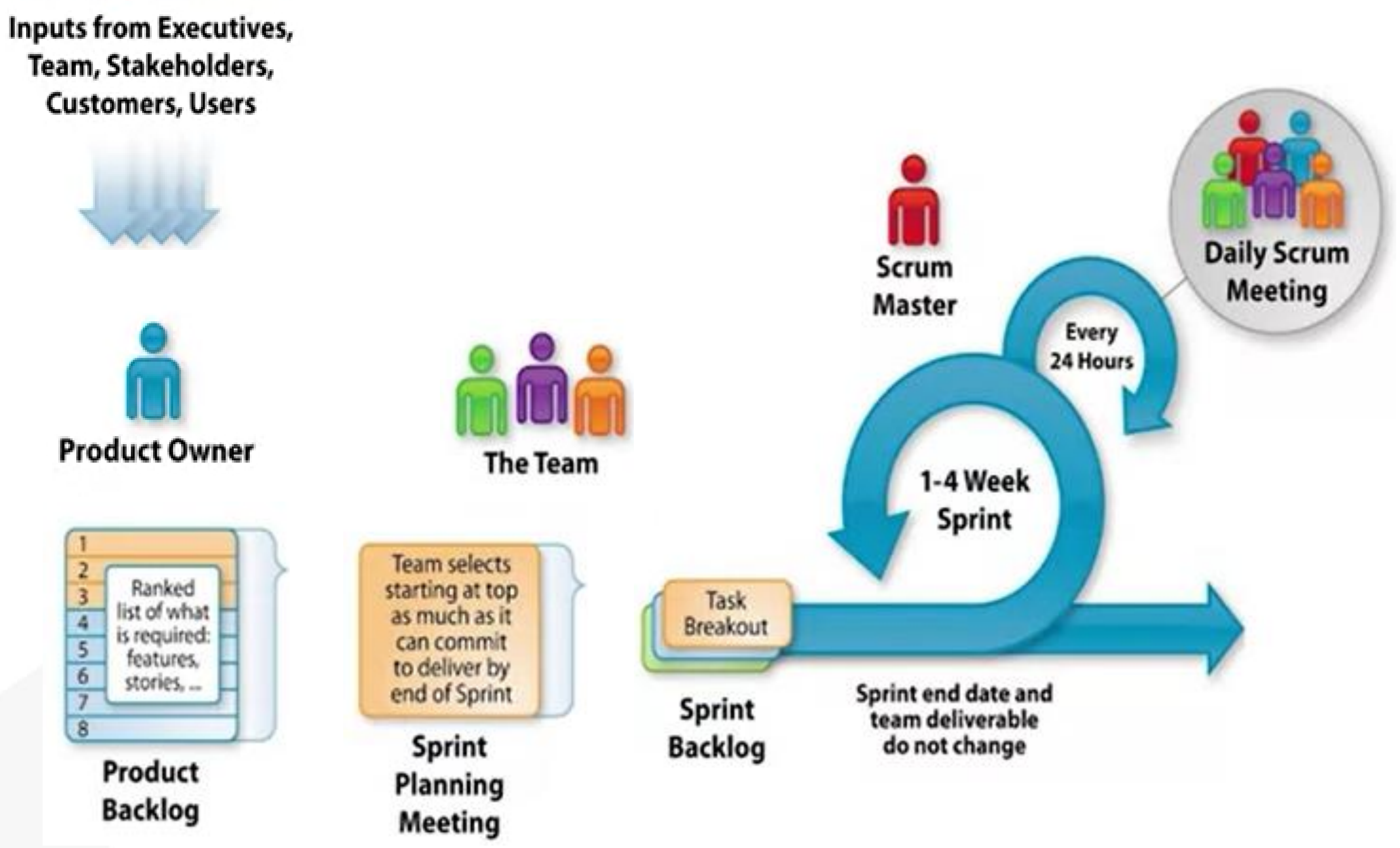
- ➢ So, once the team has looked at the stories and they have tasked it out and they know that they can finish it in the next sprint or the existing sprint,
- ➢ Then they will commit to the stories and say that, okay we'll finish it in this sprint.

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change

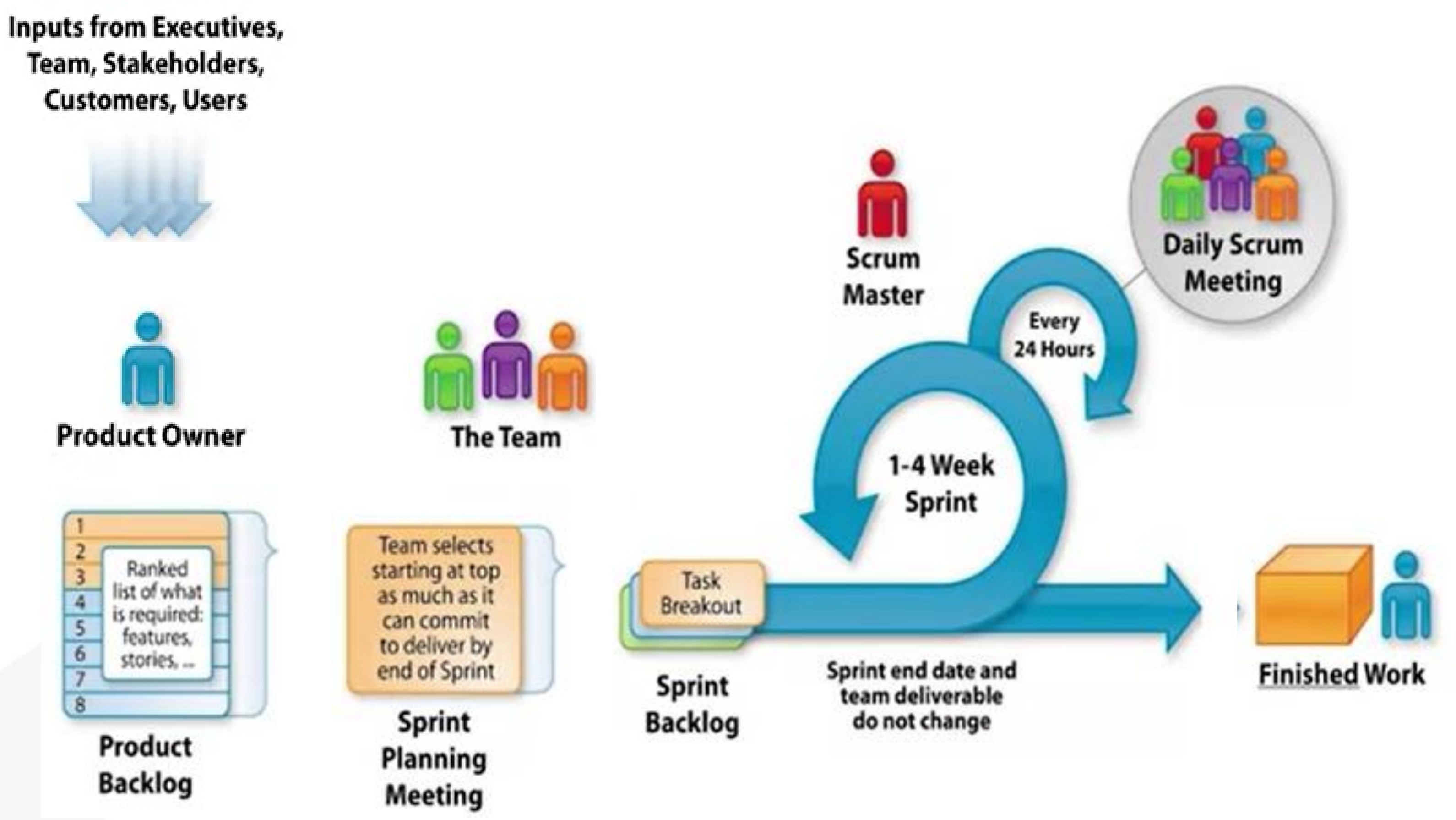
- ➢ Then that starts their execution and that starts the sprint where everybody is working to implement the software.
- ➢ **During that sprint**, the whole team **gets together for a daily stand-up** in which, everybody talks about what they did yesterday, what are they going to do to day and if there are any road blocks.

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change

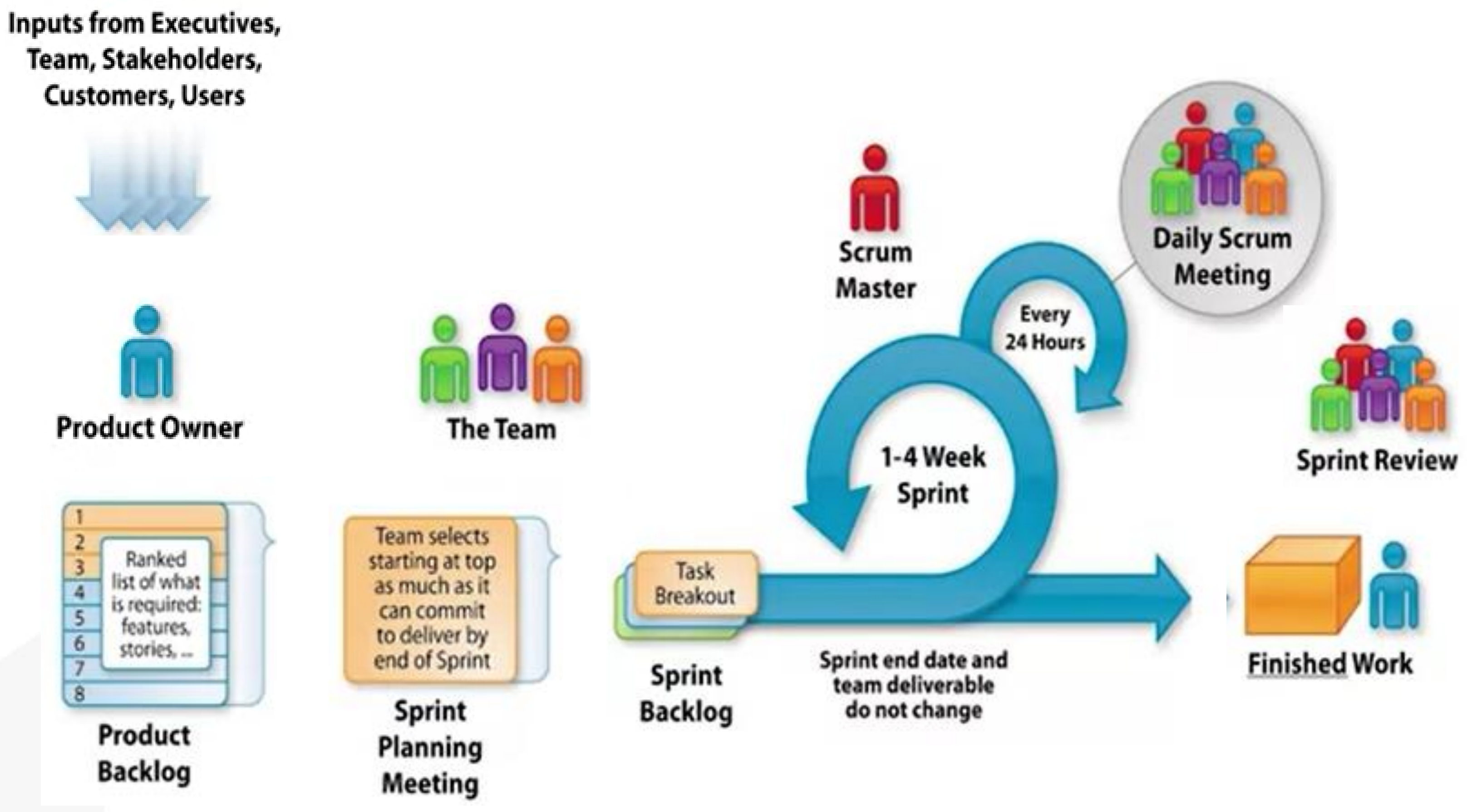
- ➢ So in this case, our database person may say, well, I designed or created the Tables yesterday in the database and then today I'm going to populate it with some test data. And if anybody out there who can help me, that'll be awesome.
- ➢ then he may also say that I don't have the permission of certain things in the database and unless I have that permission, I won't be able to finish my job.

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work

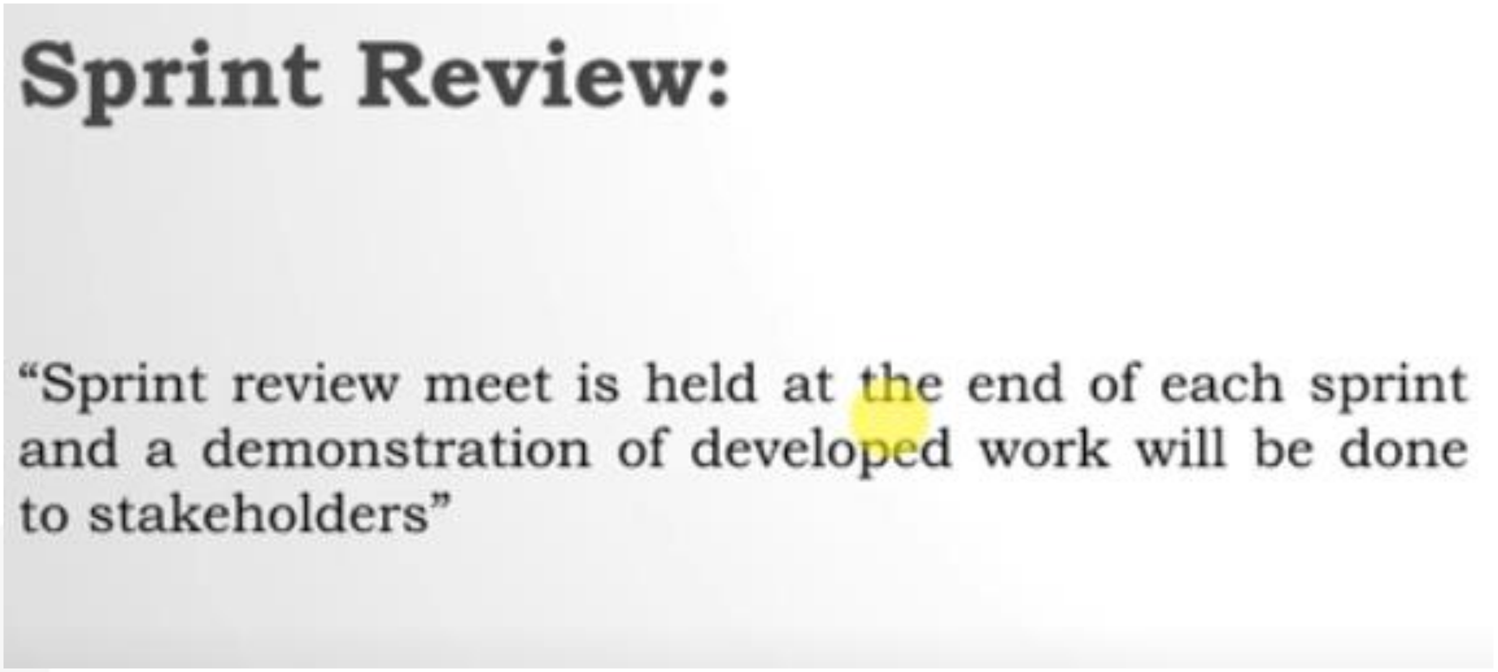
- ➢ So then, somebody might help him like, who do I contact, what do I do to get access to that? And so this thing continues for the rest of the days in the sprint.
- ➢ At the end of the sprint, you end up having your finished product.

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work

- There are **two other meetings that happen at the end of the sprint**.
- The **first one is called a Sprint Review**, where the whole team gets together with the stakeholders, with the client and demonstrate the work that they have done and get the feedback.

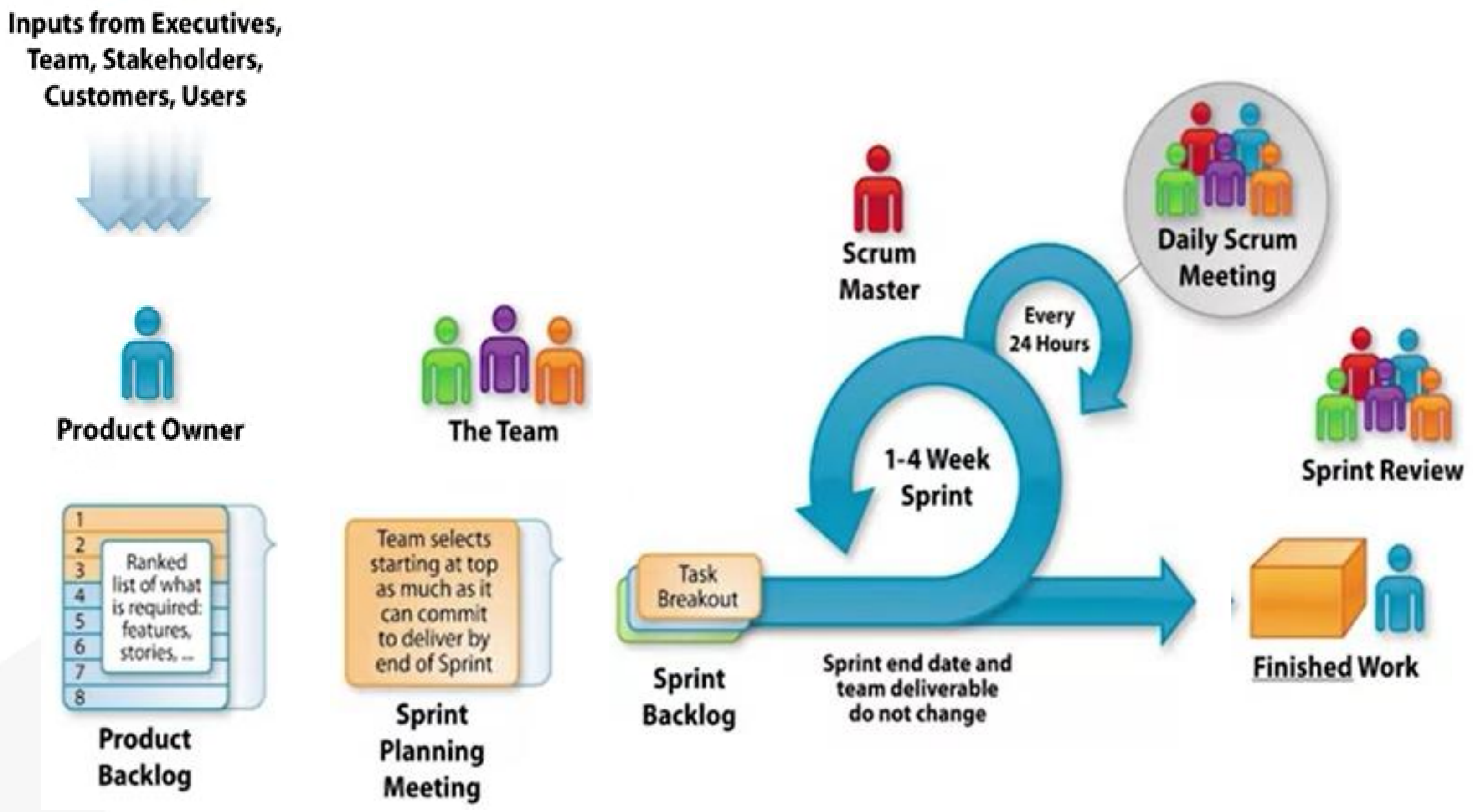
Scrum



Sprint Review:

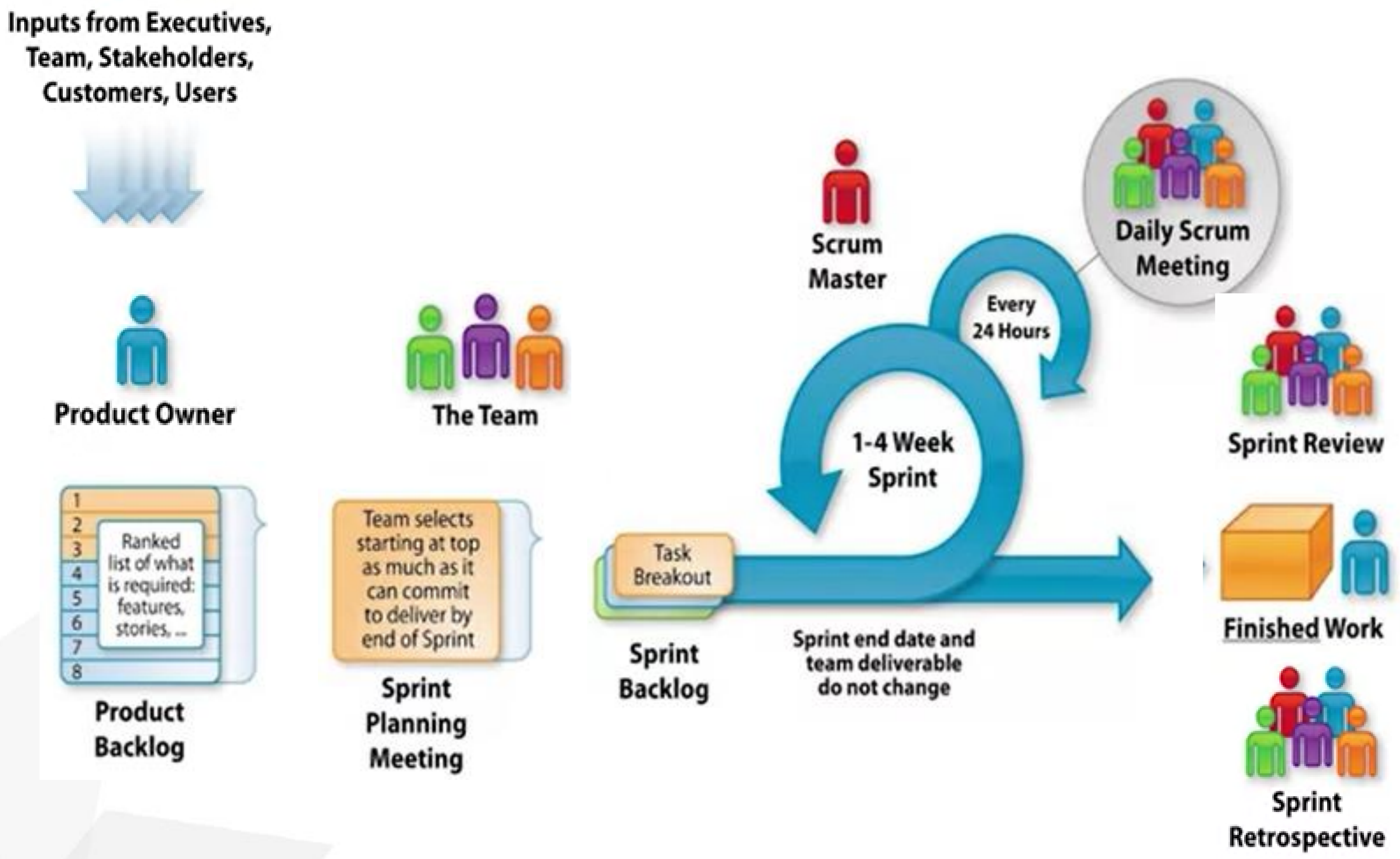
"Sprint review meet is held at the end of each sprint and a demonstration of developed work will be done to stakeholders"

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work

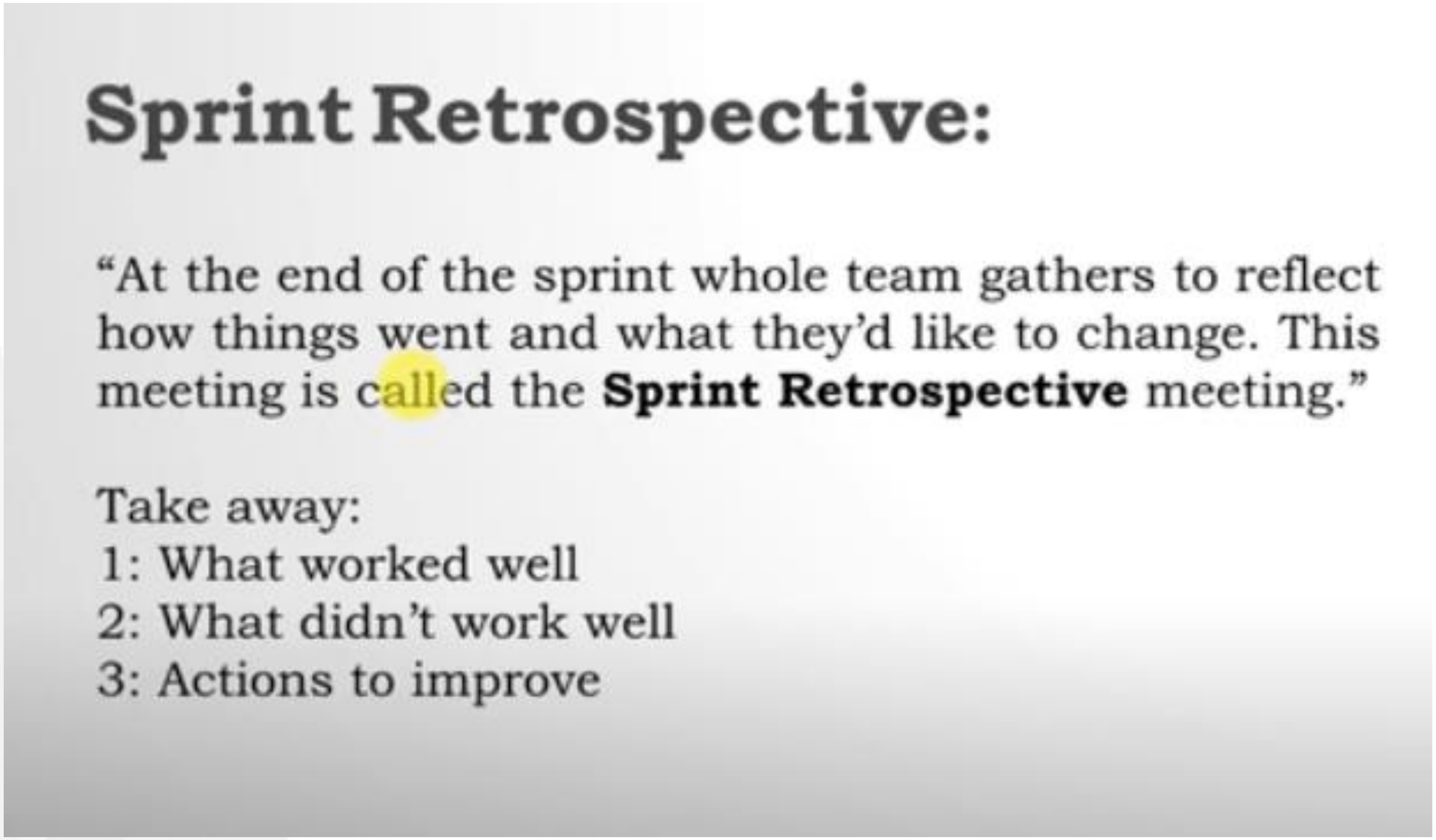
➢ So, in that meeting, the stakeholders may say, well I have additional idea, I think we should do that also. Or, they may say that, let's tweak some of this functionality.

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Sprint Review
Finished Work
Sprint Retrospective

➢ The second meeting that happens at the end of the sprint is called a Sprint Retrospective, where they talk about the process and not about the product.

Scrum

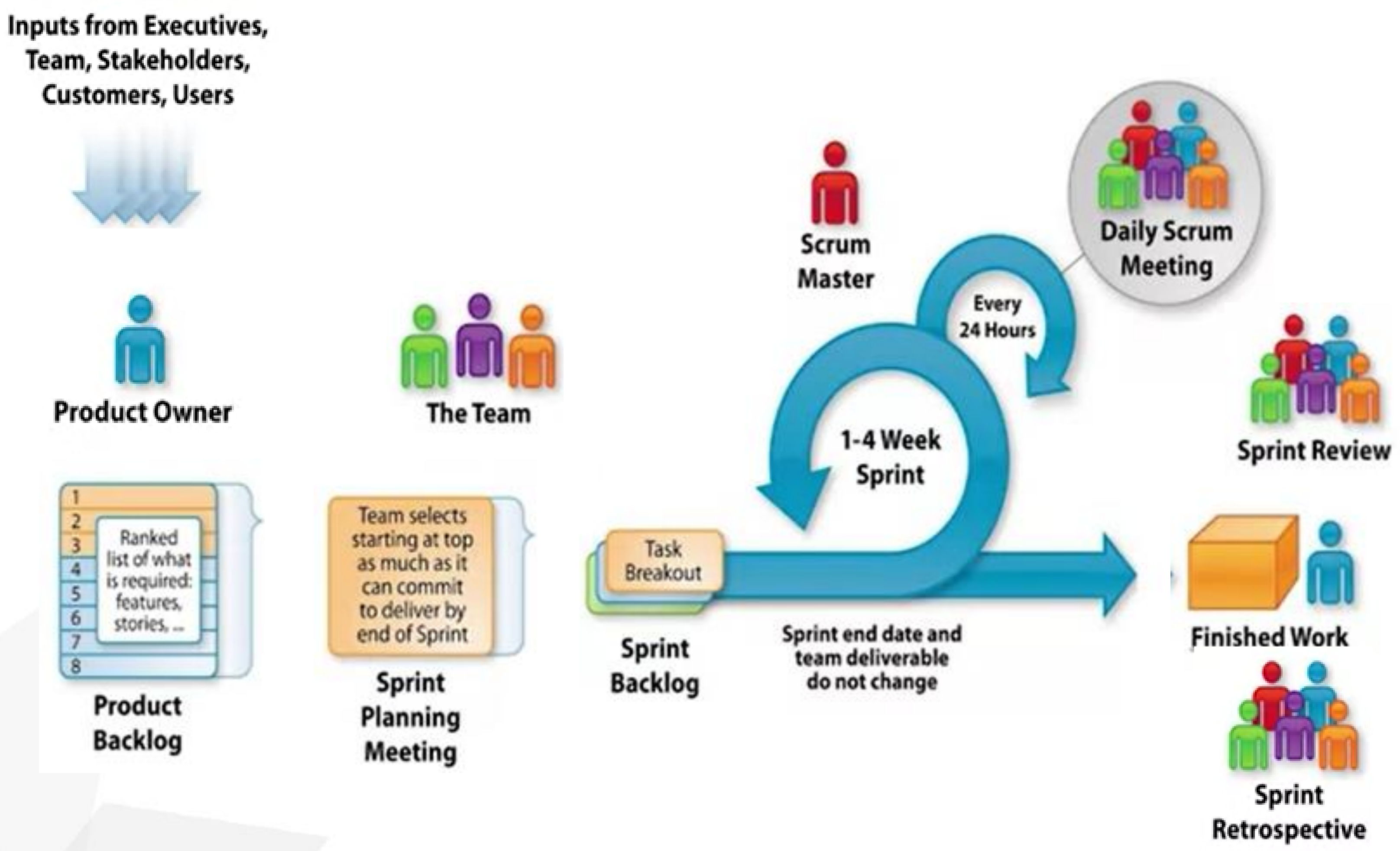


Sprint Retrospective:

"At the end of the sprint whole team gathers to reflect how things went and what they'd like to change. This meeting is called the **Sprint Retrospective** meeting."

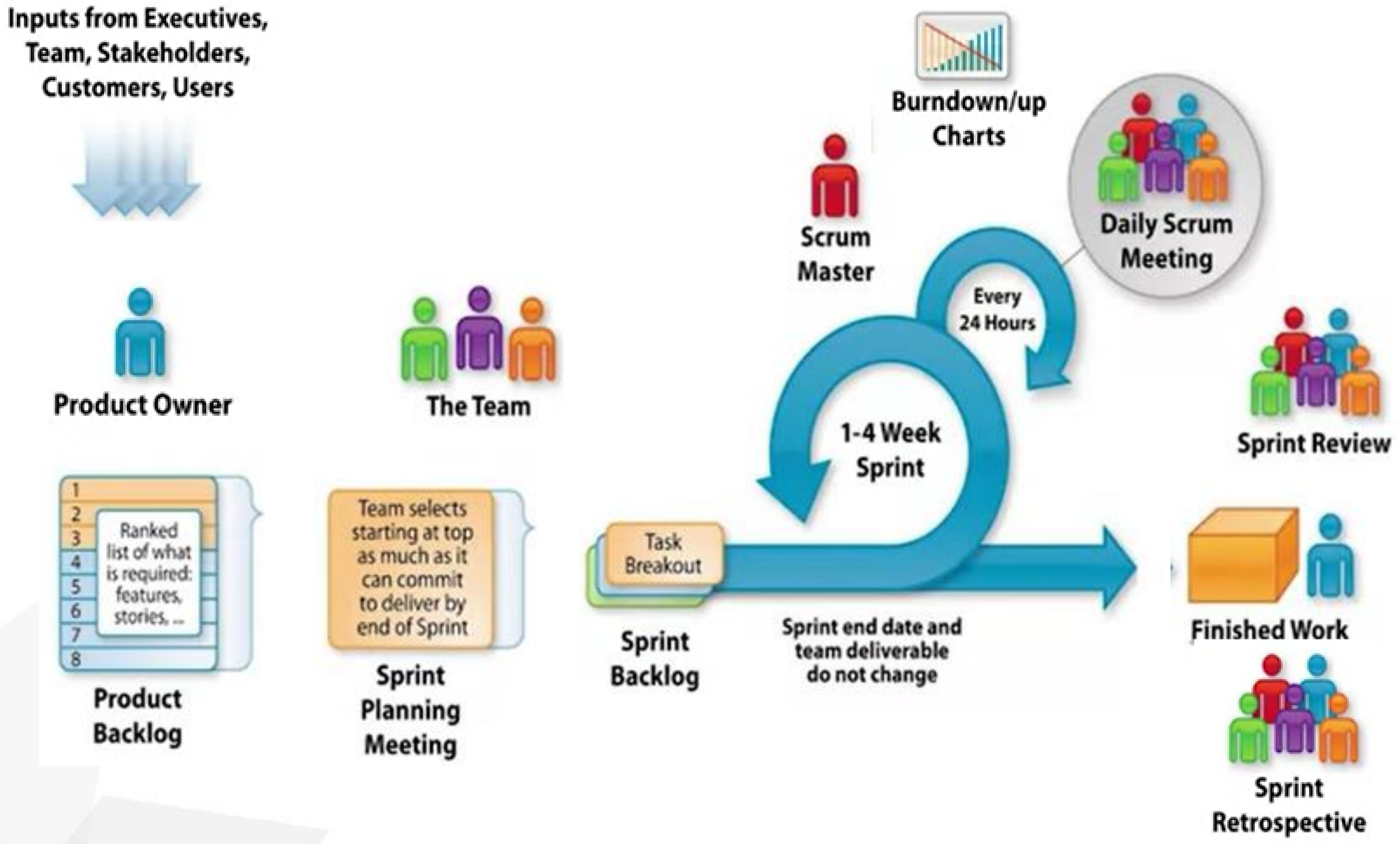
Take away:
1: What worked well
2: What didn't work well
3: Actions to improve

Scrum Framework


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

➢ So they talk about like how can we do better? So, in this case, they talk about what went well in the last sprint, what didn't go well in the last sprint, and how we can do better.

Scrum Framework

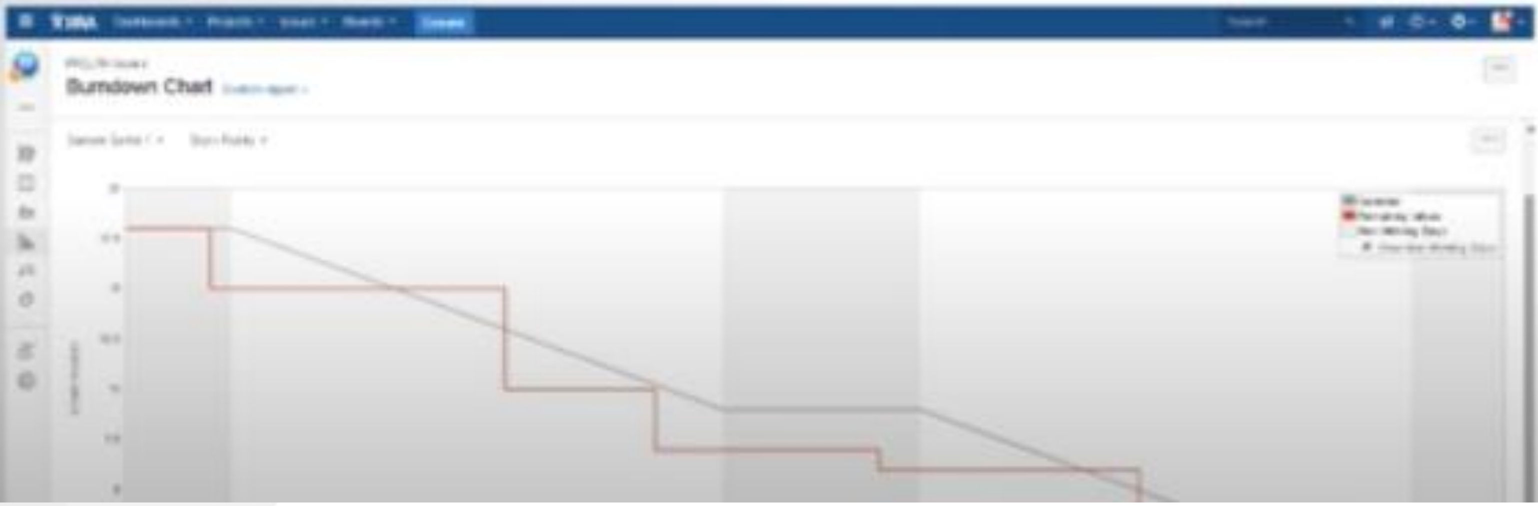

Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

➢ To keep track of the sprint, like are we going to achieve our goal for the sprint, the team uses something called **a Burn down or a Burn up Chart, which shows the amount of work left or the days of the sprint**

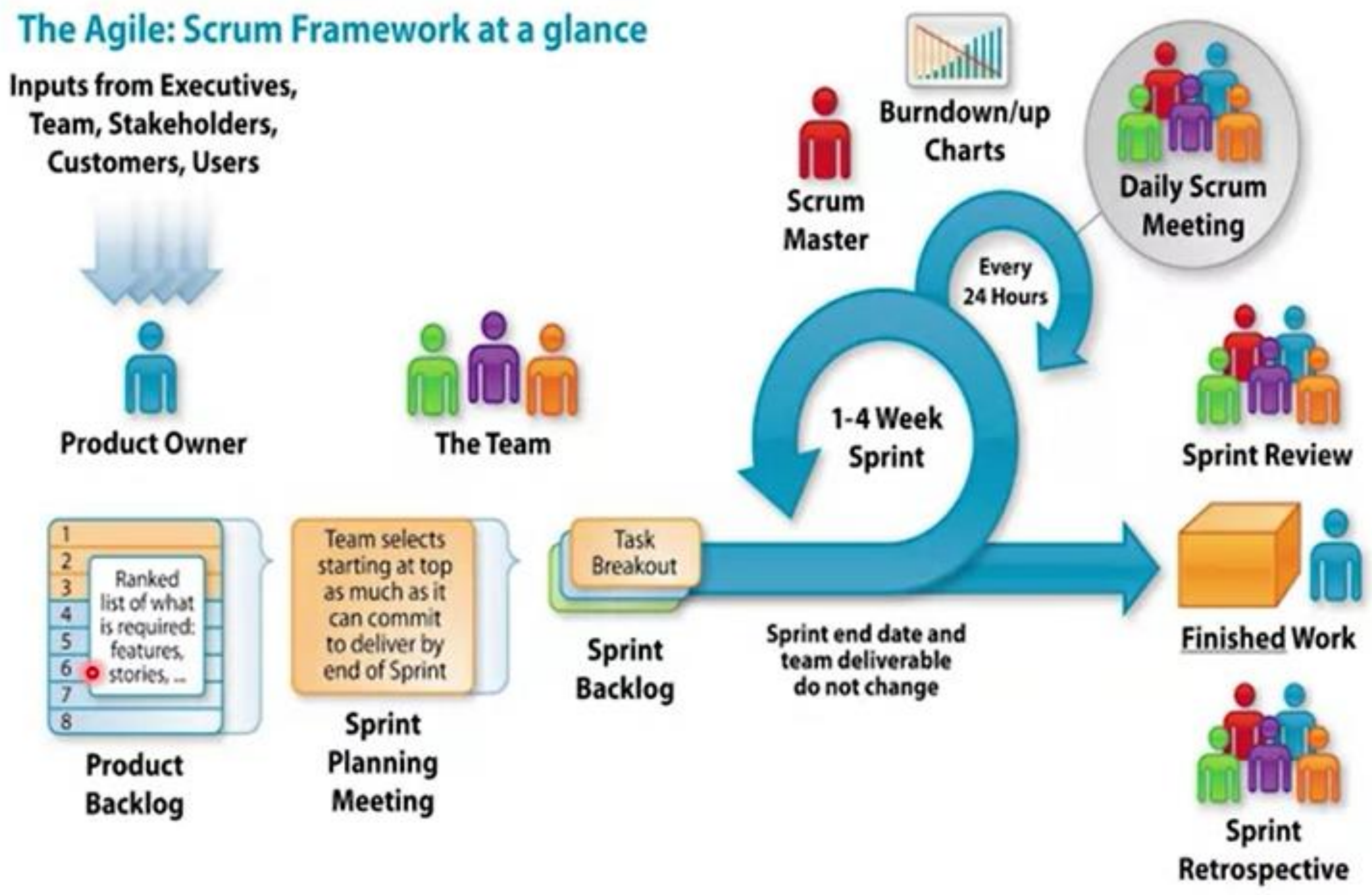
Scrum

REPORT - burndown:

"A **burndown** chart is a graphical representation of work left to do versus time"

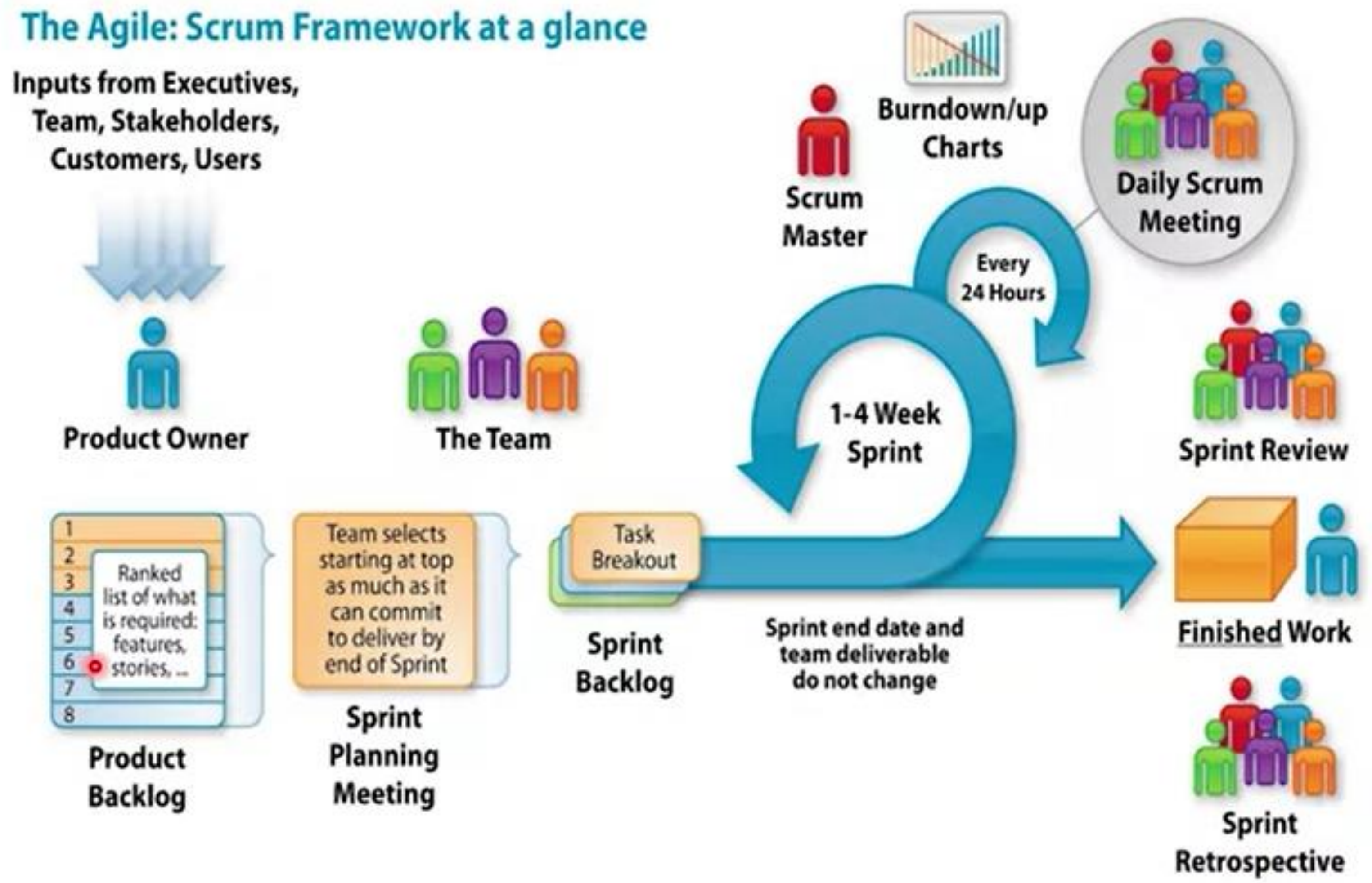


Scrum Framework


The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

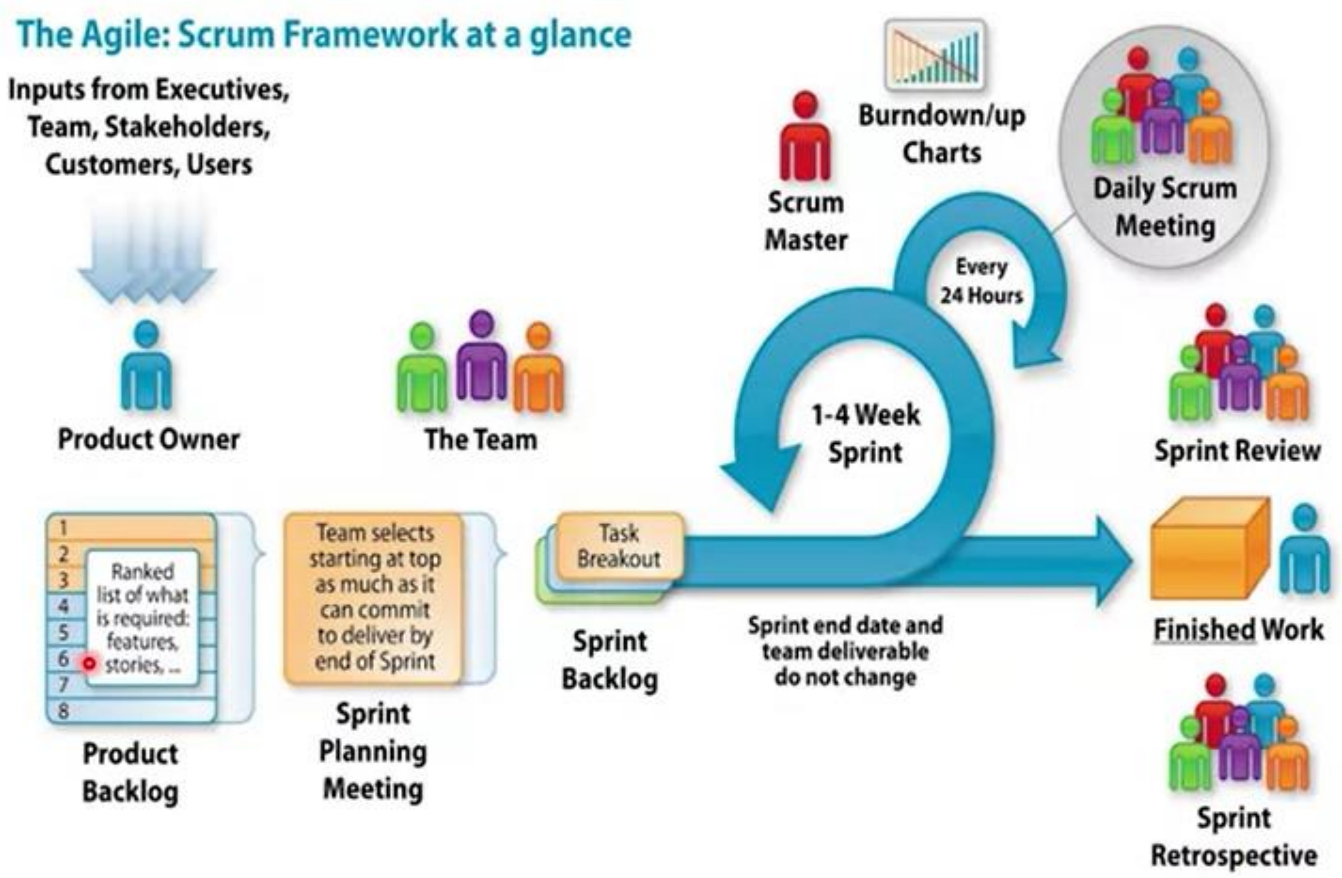
- ➢ **Scrum supports Agile principles**. As you can see, we are **building it iteratively**, so **it addresses change**.
- ➢ After every sprint you can make changes to your product backlog and shift your product to a different direction.

Scrum Framework


The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

➢ It also support **a lot of meetings for collaboration**, a lot of opportunities say for collaborations between the team members with the clients. It also supports **Continuous improvement** with a Sprint Retrospective.

Scrum Framework


The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
Finished Work
Sprint Retrospective
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change

- **Summary**:
- There are three roles in Scrum: Scrum Master, the team, and the product owner.
- There are a couple rituals, like Sprint Planning Meeting, Daily Scrum Meeting, Sprint Review, and a Sprint Retrospective.
- There are some artifacts like Product Backlog, Sprint Backlog and Burndown Chart

Questions?

Which of the following are official rituals/meetings/practices in Scrum?

a) Mid-Sprint Status Review Meeting

b) Sprint Planning Meeting

c) Daily Scrum Meeting

d) Sprint Retrospective Meeting

e) Sprint Review Meeting

Questions?

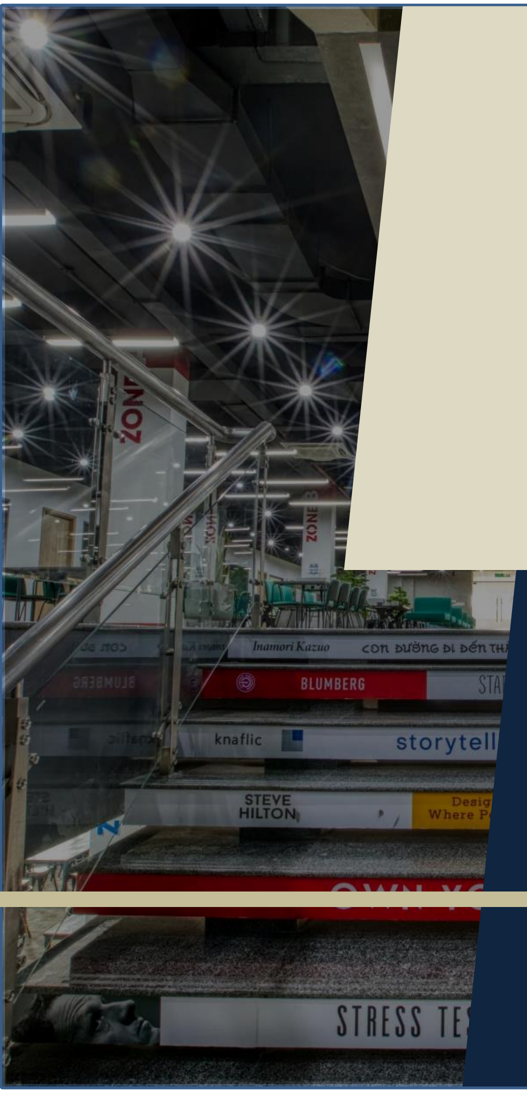
Which of the following is true about product and sprint backlogs? Select two.

a) A sprint backlog is created during the sprint planning meeting.

b) The product backlog is a prioritized backlog with highest priority items on the top.

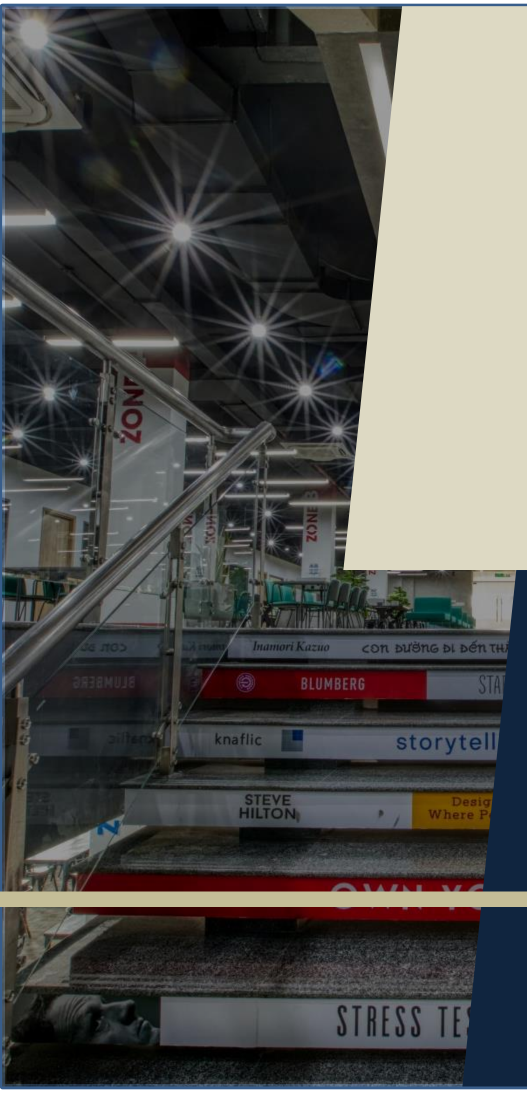
c) Sprint and product backlogs are same thing.

d) A sprint backlog has all of the items contained in the product backlog.



TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Software Engineering
Course’s Code: CSE 305



Chapter 3. Agile Software Development

3.1. Agile Process

3.2. Scrum

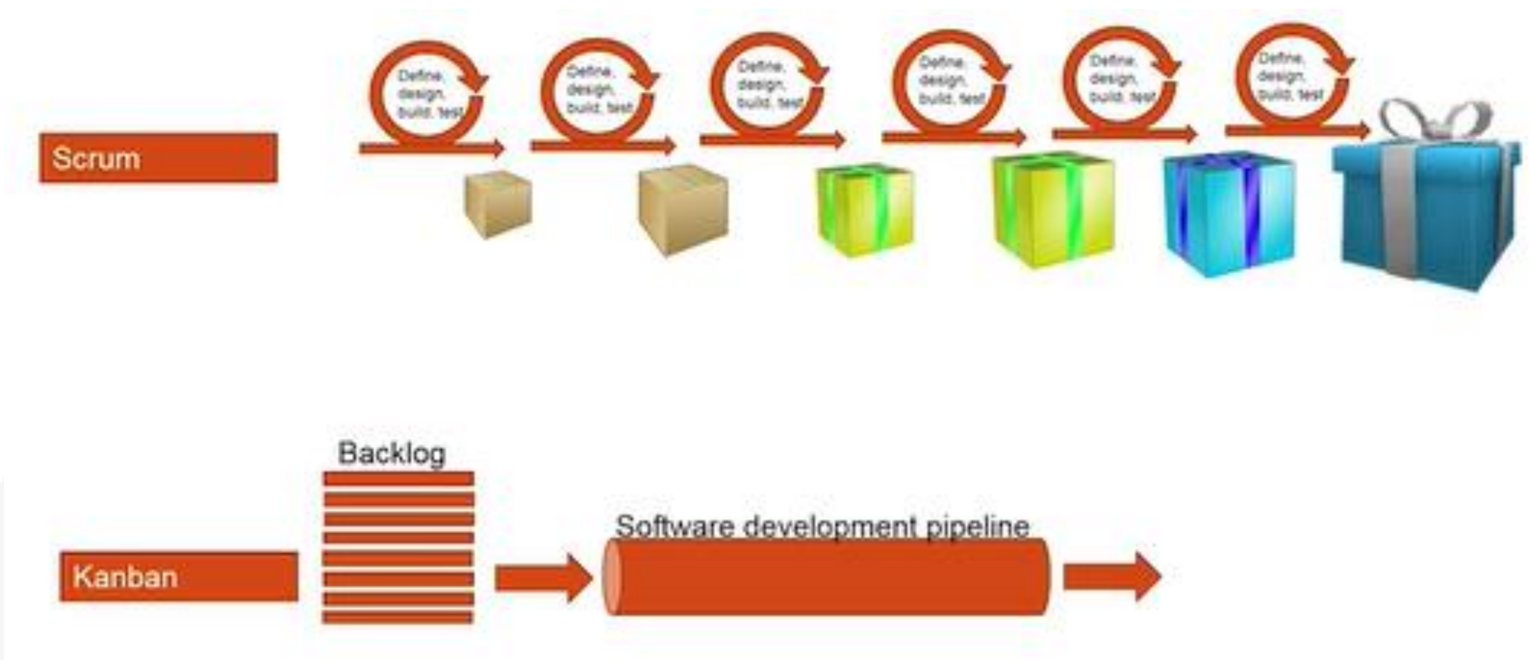
3.3. Kanban

3.4. Extreme Programming

Kanban

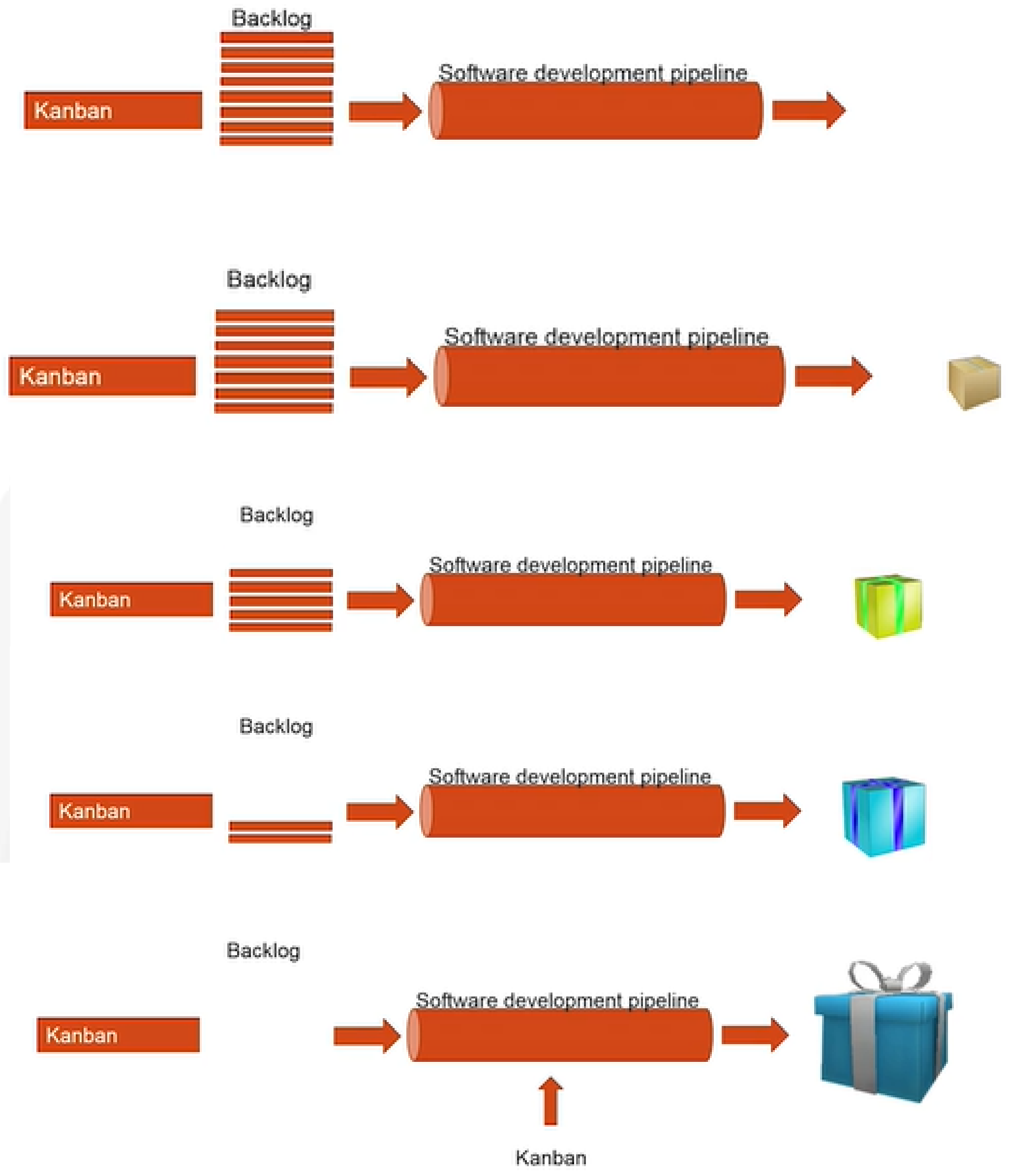
- Kanban is a agile practice that was borrowed from Toyota's production system.
- It basically, helps teams optimize their software development process
- To understand Kanban, let's put it next to a Scrum process so we know how diff erent it is

Kanban



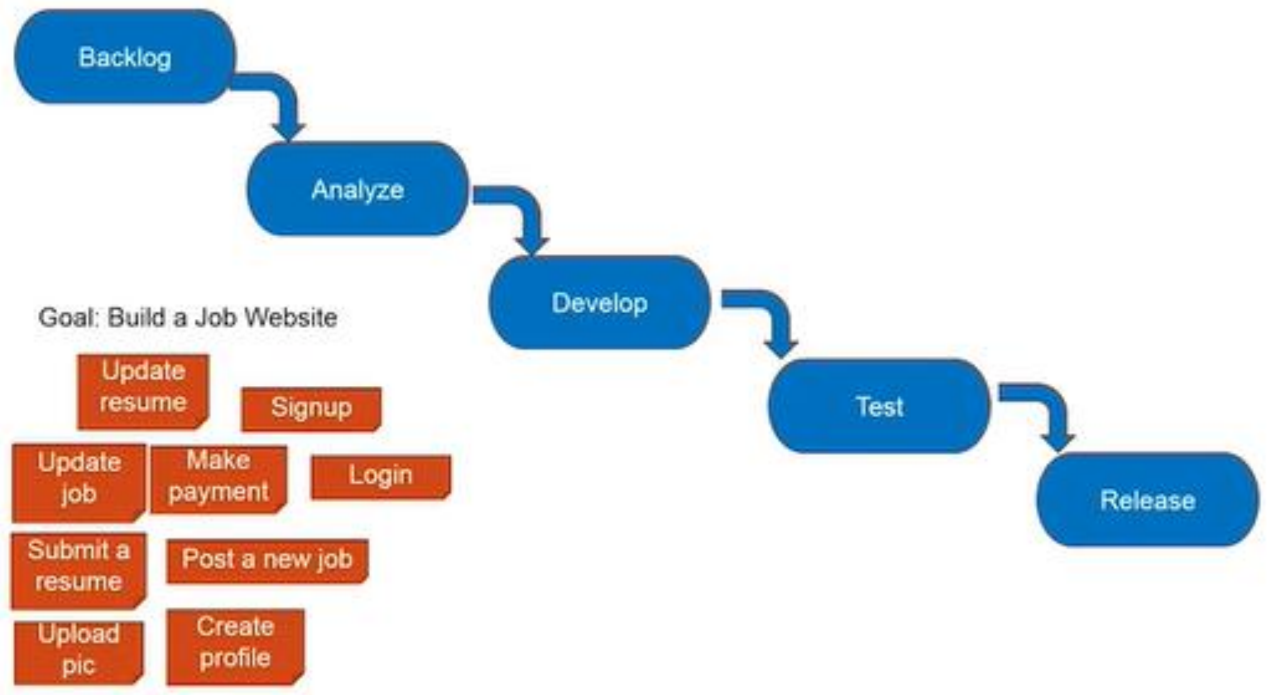
- In **Scrum**, we work in one to four week iterations. As you can notice that the product changes over time. The style and the color is changing.
- That has to **reflect the adaptive nature** of Scrum.
- **Kanban** doesn't prescribe any of these fixed iterations or any new practices. It's just a set of properties and principles that help optimize software development process as long as that process is a **continuous flow**.

Kanban


Backlog
Kanban
Software development pipeline
Backlog
Kanban
Software development pipeline
Backlog
Kanban
Software development pipeline
Backlog
Kanban
Software development pipeline
Backlog
Kanban
Software development pipeline
Kanban

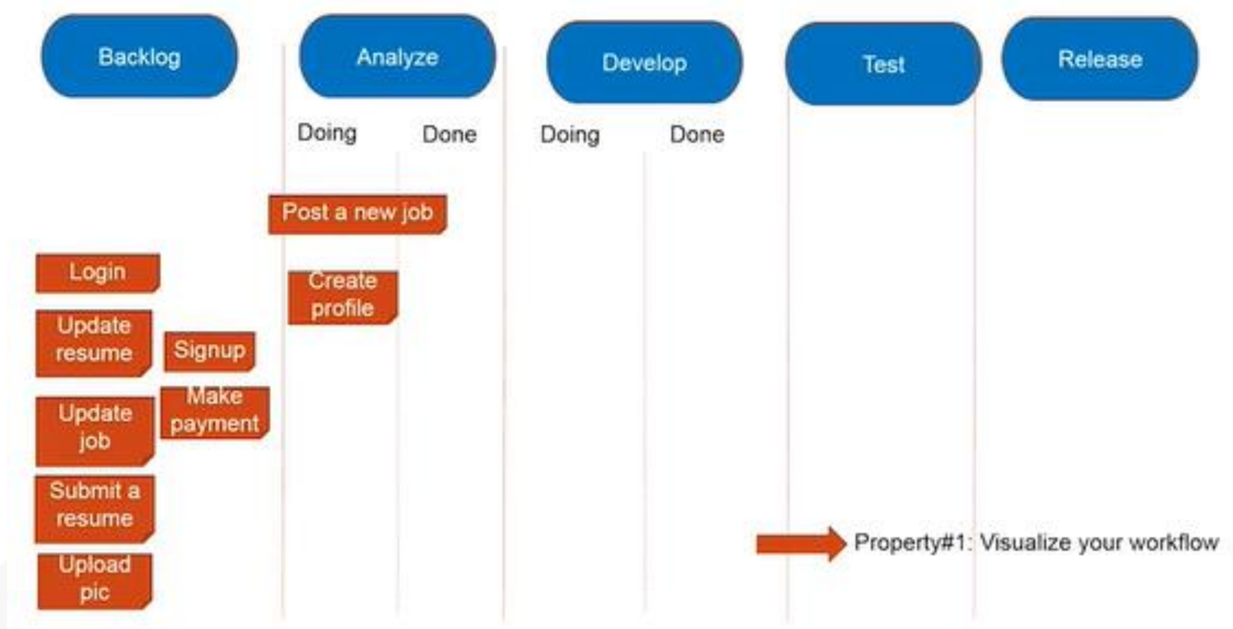
- Continuous flow,
  - Things from the backlogs are moving through the software development pipeline and finished product is coming out at the other end of the pipeline.
- This is also an adaptive process
  - where as things are getting completed, we can get feedback from the customer and feed it into
    the backlog, but at the end, we get the whole product.

Kanban



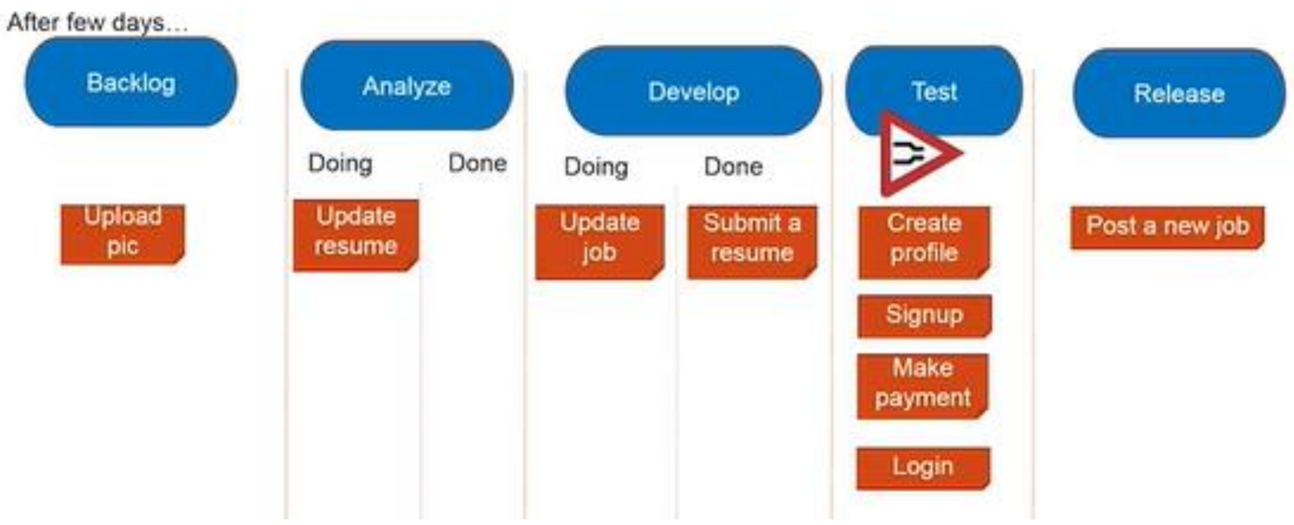
- let's say a team has followed Kanban, where the work items goes from backlog to the analysis or analyze, then develop, then test, and then to release.
- Let’s assume we have asked this team to build a job website which has features. So let's apply Kanban to this situation..

Kanban



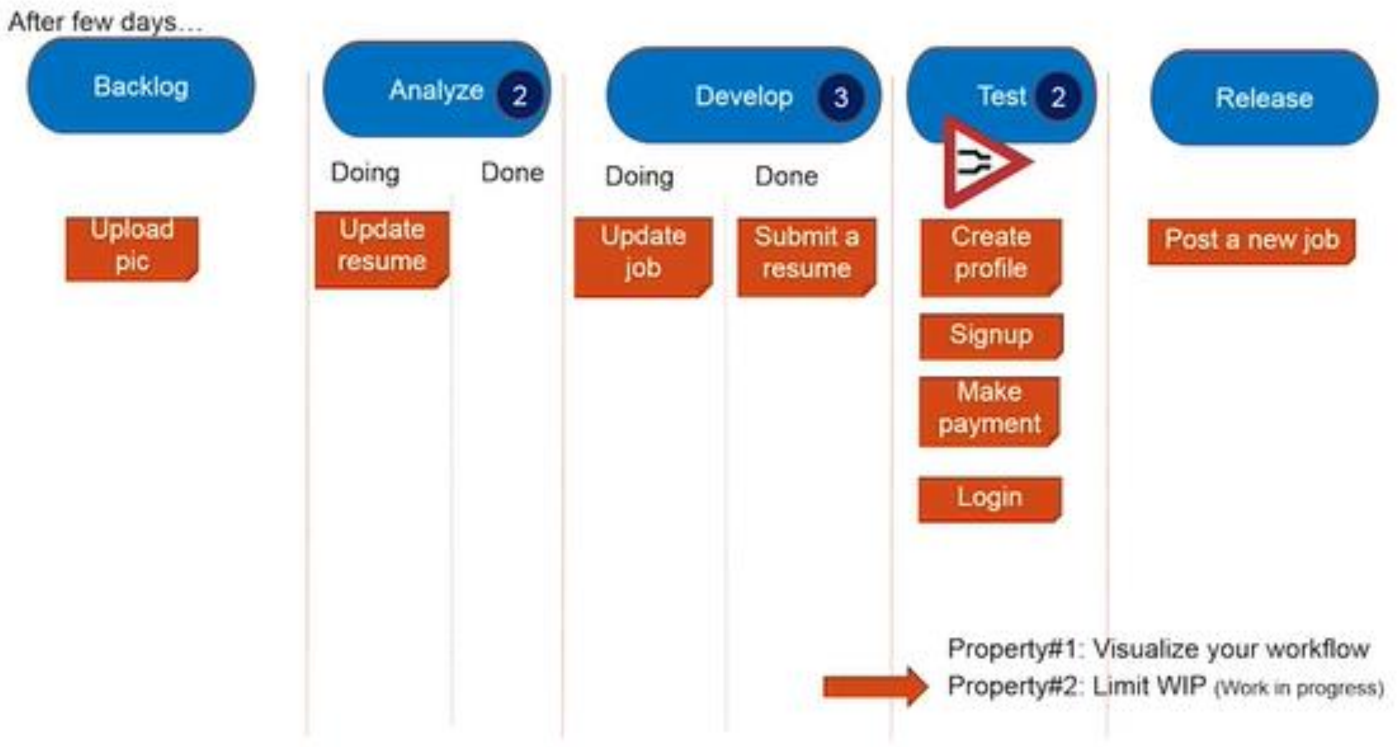
- The first thing that Kanban suggests or the first property of Kanban is to "Visualize your workflow".
- Create a visual board. It can be an electronic or it could be a physical board.
- In some columns, you will see doing and done. That means, if the item is being processed or being analyzed then it will be in the doing, if it is done, then you will move it to the done sub-column.

Kanban



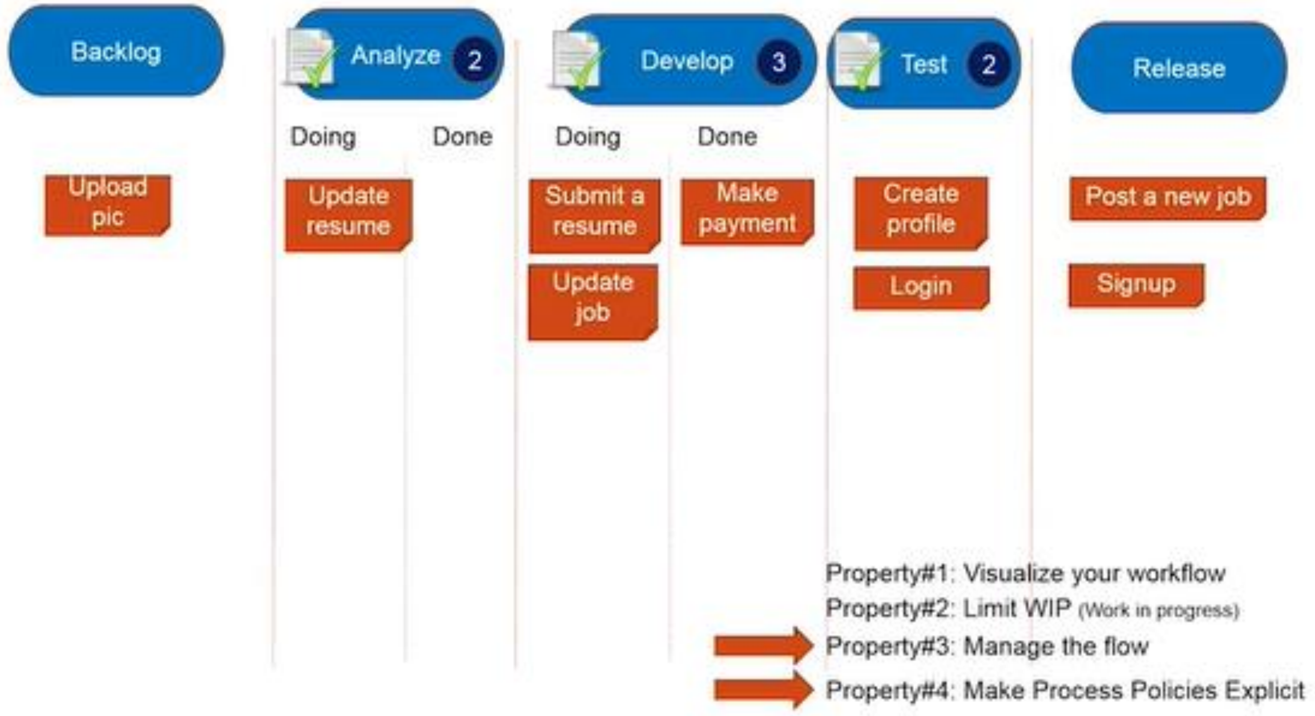
- A Kanban board after a few days may look like this.
- Do you see any problem with this board?
- If you see in the Test column, there is lot of items pending, if we don't do anything right now, the items in the Test column will pile up and will become a bottleneck, and it will slow down the delivery.
- How do you solve this? Kanban supports another property called "Limit Work in Progress" (WIP). What you do is for each of your states, you define what's the maximum number of items that can stay in that state.

Kanban



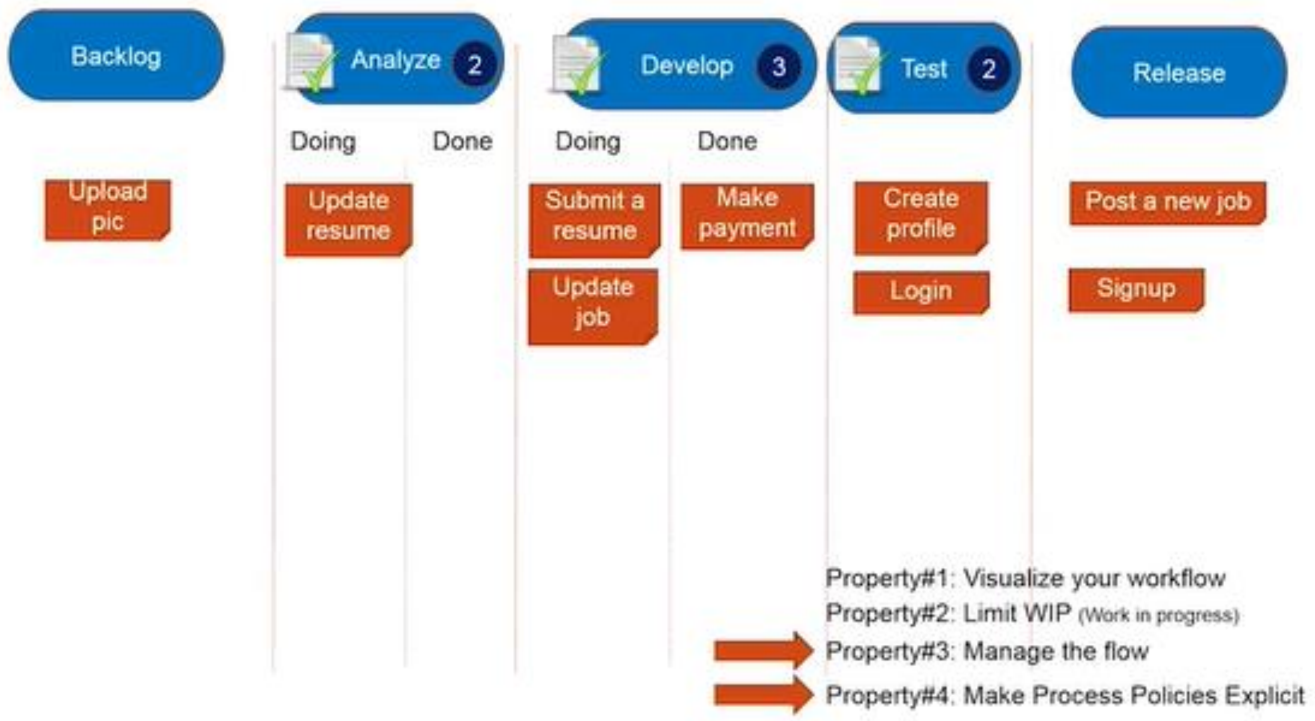
- We need to set maximum WIP for each state
- For “analyze”, maximum WIP is 2; for “develop”, maximum WIP is 3 and for “Test”, maximum WIP is 2

Kanban



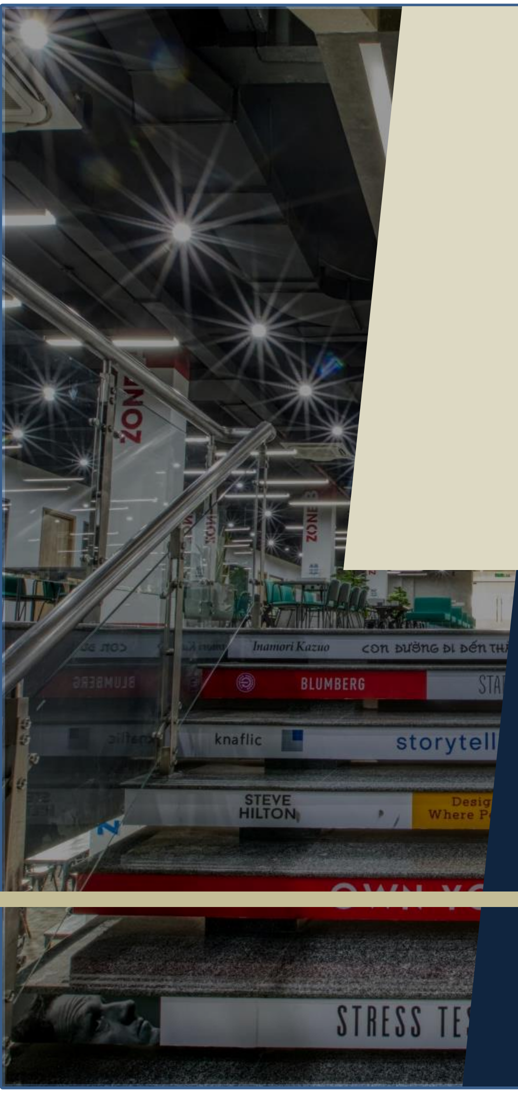
- property #3 is "Manage the flow" :
- For example, we want to find out if it is a permanent resource error, then add more people to the test or if it is buggy code, then what can we do in the development so that we don't have that many defects and then it moves faster. Whatever you do but you try to manage your flow

Kanban



➢ property #4 is “Make Process Policies Explicit" which means, we define exactly, what are the policies that the team will follow.

➢ One example could be, the definition of Done for each of the states. The definition of Done would be, when can I move? When is something done? for the Analyze phase, when the "Update resume" is done?



TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

Chapter 3. Agile Software Development

XP- eXtreme Programming

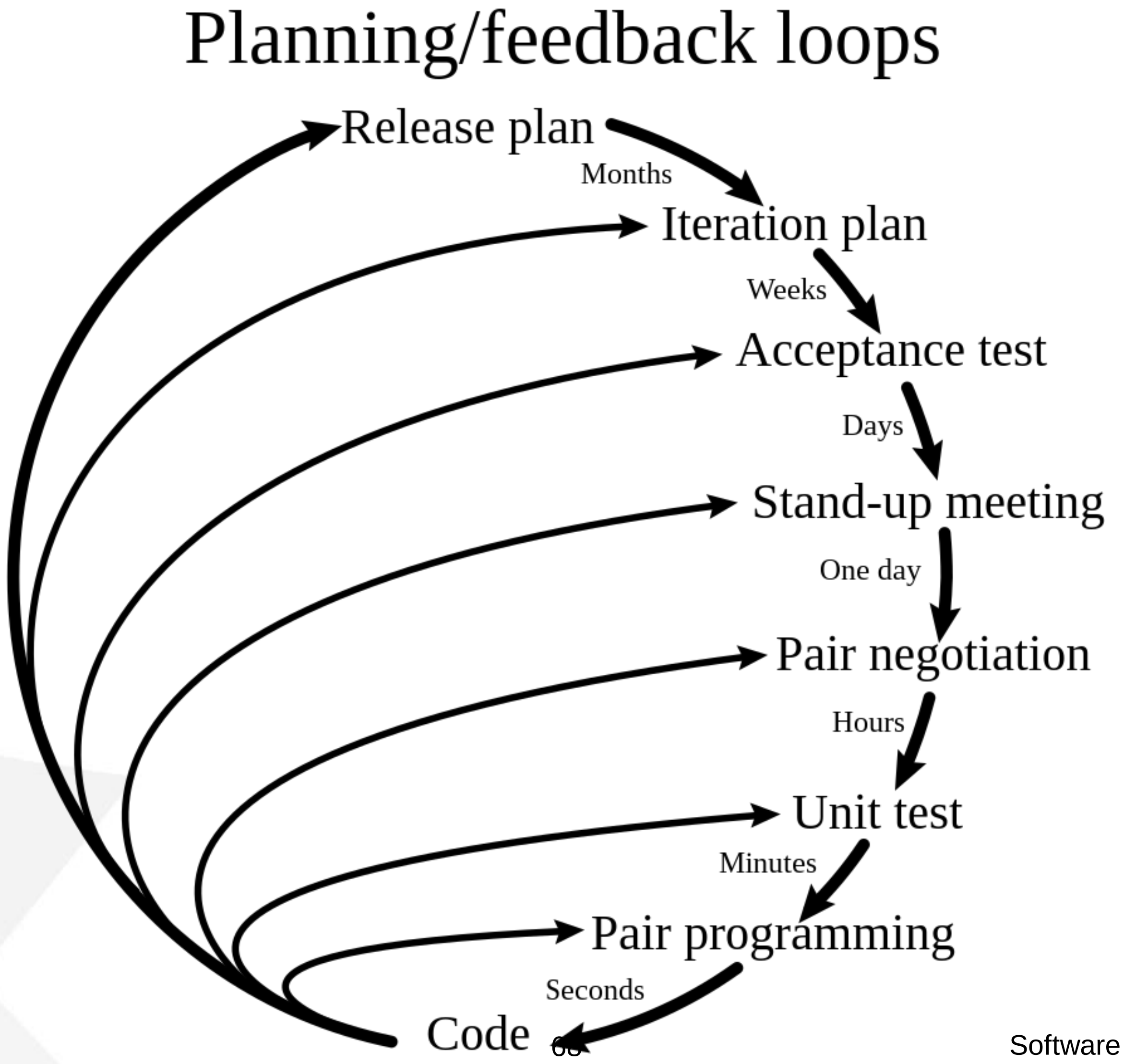
- The word extreme came from, like, if something is good, let's take it to the extreme.

- For example, there is a technique called code review in the software industry where if a developer is done with their code, they get it reviewed by another developer in the team to say whether it is meeting the quality, whether it is doing what it is supposed to do, and is it implementing the functionality correctly?

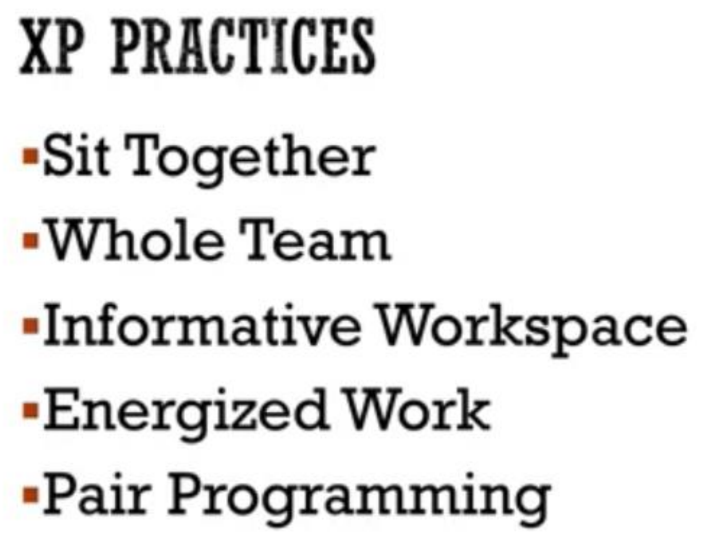
- So, in eXtreme Programming, we take code review to its extreme where, instead of doing it just at the end of development, why don't we get the two developers working together so they're constantly giving feedback to each other, constantly talking to each other

Developing Software using XP

- User stories are a small unit of functionality to be built. Basically, you can say it's a part of requirement.



Extreme Programming (XP)



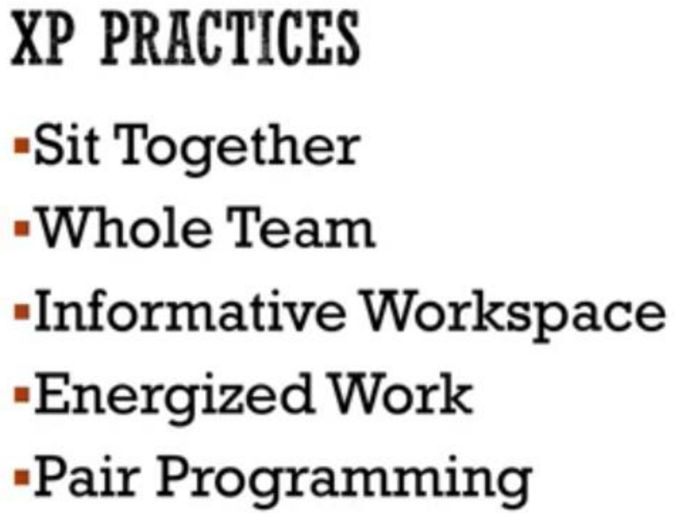
XP PRACTICES

- Sit Together
- Whole Team
- Informative Workspace
- Energized Work
- Pair Programming

Sit Together: which is Sitting Together. Aligning with the value of collaboration, XP recommends the team to sit together in open, collaborative environment.

Whole Team: With the focus on delivering the software in short iterations, it is really important that everyone needed for the project's success is part of the team. if we have the need for database related work and we need a database administrator, we add the person with the required skill to the team.

Extreme Programming (XP)



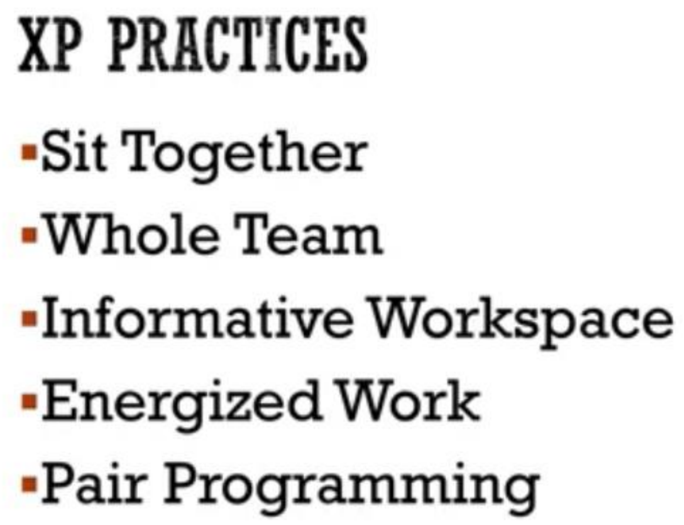
XP PRACTICES

- Sit Together
- Whole Team
- Informative Workspace
- Energized Work
- Pair Programming

Informative Workspace: What it means is that you set up your workspace with information on the walls or appropriate places in the workplace so that if any interested person walks into the space, he or she can get an idea about how the project is doing in less than 15 seconds.

Energized Work: What this means is for team members to work as many hours as they can be productive and only as many hours as they can sustain. Burning themselves out on productivity today and spoiling the next two days do esn't help them or the team.

Extreme Programming (XP)


XP PRACTICES

- Sit Together
- Whole Team
- Informative Workspace
- Energized Work
- Pair Programming

Pair Programming: Pair programming is a practice where two team members are working together on a single computer machine. It is important to allow team members to think about the problem individually.

Questions?

Which XP practice prescribes that "the code [always be] written by two programmers at one machine"?

a) Peer Programming

b) Buddy Programming

c) Pair Programming

d) Twin Programming

Questions?

One of the practices of XP is "Whole Team". Which of the following statements align with its meaning?

a) All the skills necessary to deliver the software product should be present on the team.

b) The whole team should always sit together in a room.

c) The whole team should be working together to meet the team's commitment

d) The whole team should be energized and passionate about the product they are building.