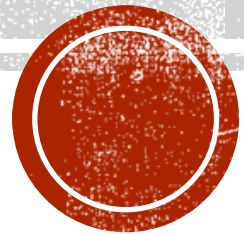
AGILE METHODOLOGY

Yesoda Bhargava



BUT BEFORE THAT…

- Why did we need Agile at the first place?
- What led to it?

HARDLY SURPRISING FACTS

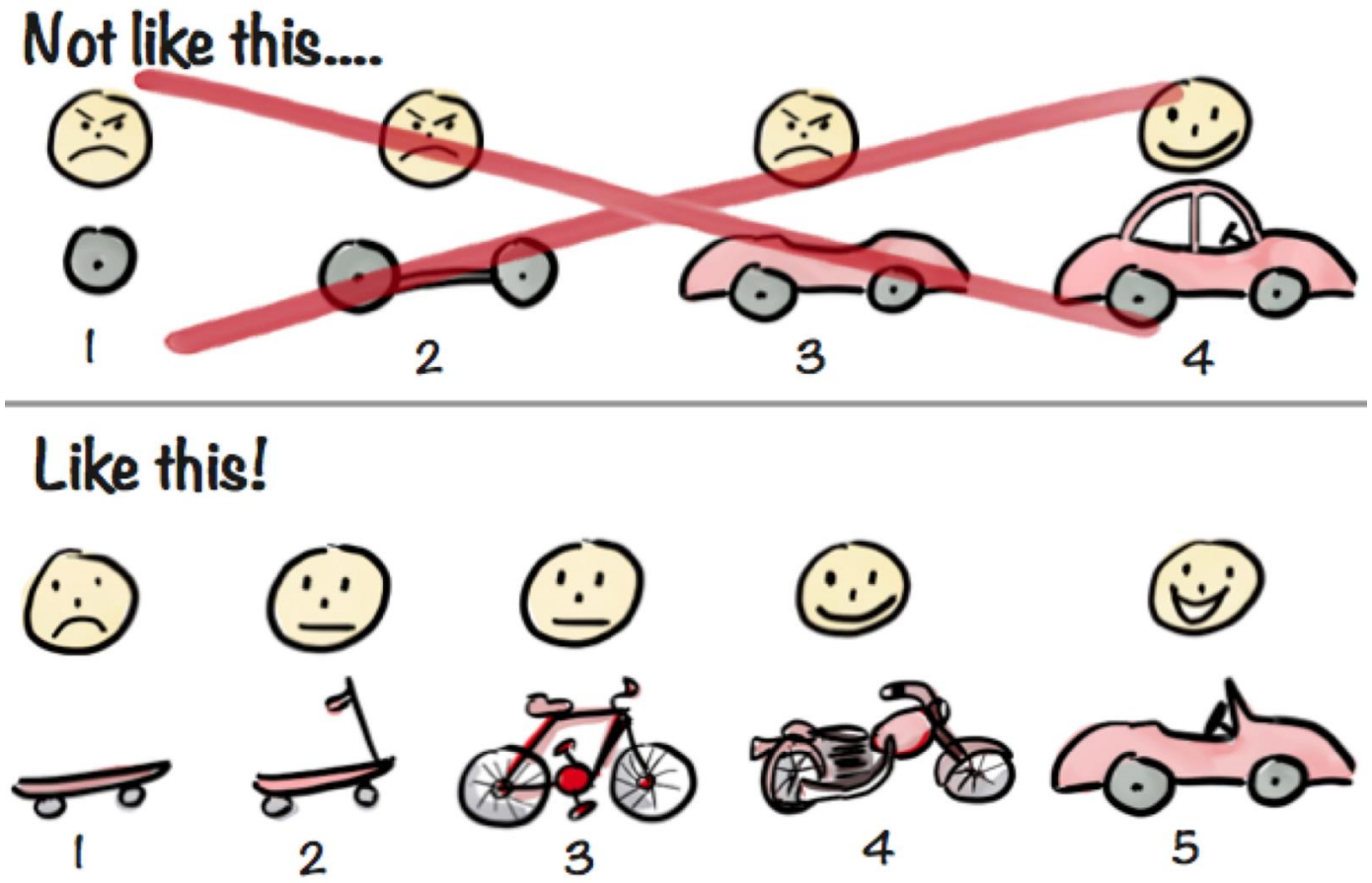
- Most software projects go wrong.
- Most software projects do not follow *any* development process
- Software projects that do follow some process have a much better chance of survival.


17% OF LARGE IT PROJECTS GO SO BADLY, THEY
THREATEN THE VERY EXISTENCE OF THE COMPANY
-MCKINSEY


MOST LARGE IT PROJECTS RUN
45% OVER BUDGET, 7% OVER TIME AND
DELIVER 56% LESS VALUE THAN PREDICTED
-MCKINSEY

WHY TO STUDY THIS?

Software engineers are too busy to learn software processes.



1990s : THE APPLICATION DEVELOPMENT CRISIS

- Also called the Application Delivery Lag.
- Time between a validated business need and an actual application in production was about 3 years.
- In aerospace and defense it could be more than 20 years before a complex system went into actual use.
- Space Shuttle program launched in 1982, used information and processing technologies from the 1960s.
- The crisis? Highly complicated hardware and software systems were often designed, developed, and deployed in a time frame that spanned decades.

FRUSTRATION OF THOUGHT LEADERS

Reason of frustration:

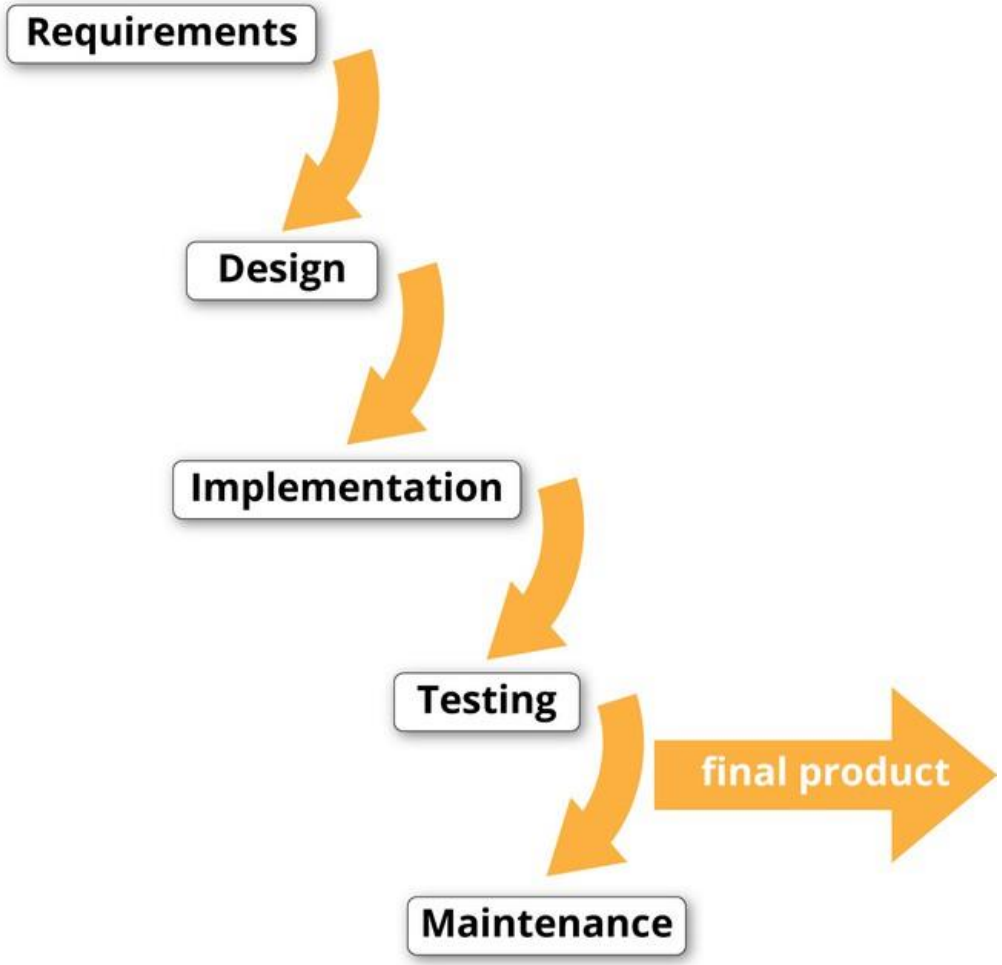
- ✓ Long lead times.
- ✓ Decisions made earlier in the project could not be changed later.

Prevalent software engineering technique at that time?

THE WATERFALL MODEL.

WHAT WAS THE WATERFALL MODEL?

- Its like an industrial manufacturing process.
- Evolved from the traditional engineering project management disciplines where much of the work was predictable and repeatable.
- Waterfall is used to describe a way of building software : think a slow, trickling, stage-by-stage process.
- Like manufacturing a wristwatch.



Why do you think it was called the WATERFALL MODEL?

- Requirements must be complete before moving on to functional design, functional design complete before detailed design, and so on through the sequence.
- And like water not flowing uphill, there are rarely provisions to return to an earlier stage of the process.
- Once you were finished with a stage, that stage was frozen in time.

WATERFALL DEVELOPMENT METHODOLOGY

- Set the project requirements and scope of the work.
- Design a product based on those pre-determined requirements.
- Build the product.
- Test the product.
- Fix any problems discovered during testing.
- Launch a finished product.

Waterfall model prioritized bringing a complete product to the market – it could take years before the team finished the projects at hand.

Development graveyard of unfinished products.

Nature of problem would often change, rendering planned solution out-of-date by the time it reached market.

For customer this meant, critical problems would go unsolved for years. Or when solution became available, the nature of problem would change.

For developer this struggle meant, bringing new products to the market that no longer had a strong market fit.

WHAT WAS THE PROBLEM WITH WATERFALL MODEL?

- A linear approach might work when you know EXACTLY what you want to build.
- But can be too restrictive for some projects – software development.
- “Software Development is an invention process. Software engineers or programmers often go back and forth across the different steps. Such projects are not really sequential.” – Prof. Michael A. Cusumano, MIT.
- With Waterfall model, you have to wait till the end, to be able to test the software.
- If you catch bug at the last stage, it can be messy, even fatal to go back and fix it.
- Some software projects would get stuck and never ship.
- Can you plan out things in advance?
- The model faltered in the 1980s and early 1990s.

WHAT DID COMPANIES REALIZE?

- They can build better products usually if they can change design and specifications, get feedback from the customers, and continually test the components as the products are evolving.
- Waterfall model provides no means for risk assessment and management during the lifecycle of the software.
- In 1986, the Spiral Model was proposed.
- In 2001, Agile Method was born.
- Detailed specifications in Agile Manifesto, a document detailing Agile.

BUT….

- Waterfall model works well when software requirements are well-understood.
- Eg. Softwares such as compilers and operating systems.
- It is also good for contract-based software development since this model is document driven.
- However, Waterfall model does not work well for many classes of software, particularly interactive user applications.
- Specifying the requirements for such applications is difficult since the interface design is highly subjective.

BAGGAGE HANDLING SYSTEM AT DENVER INTERNATIONAL AIRPORT(DIA) : A CASE TO STUDY

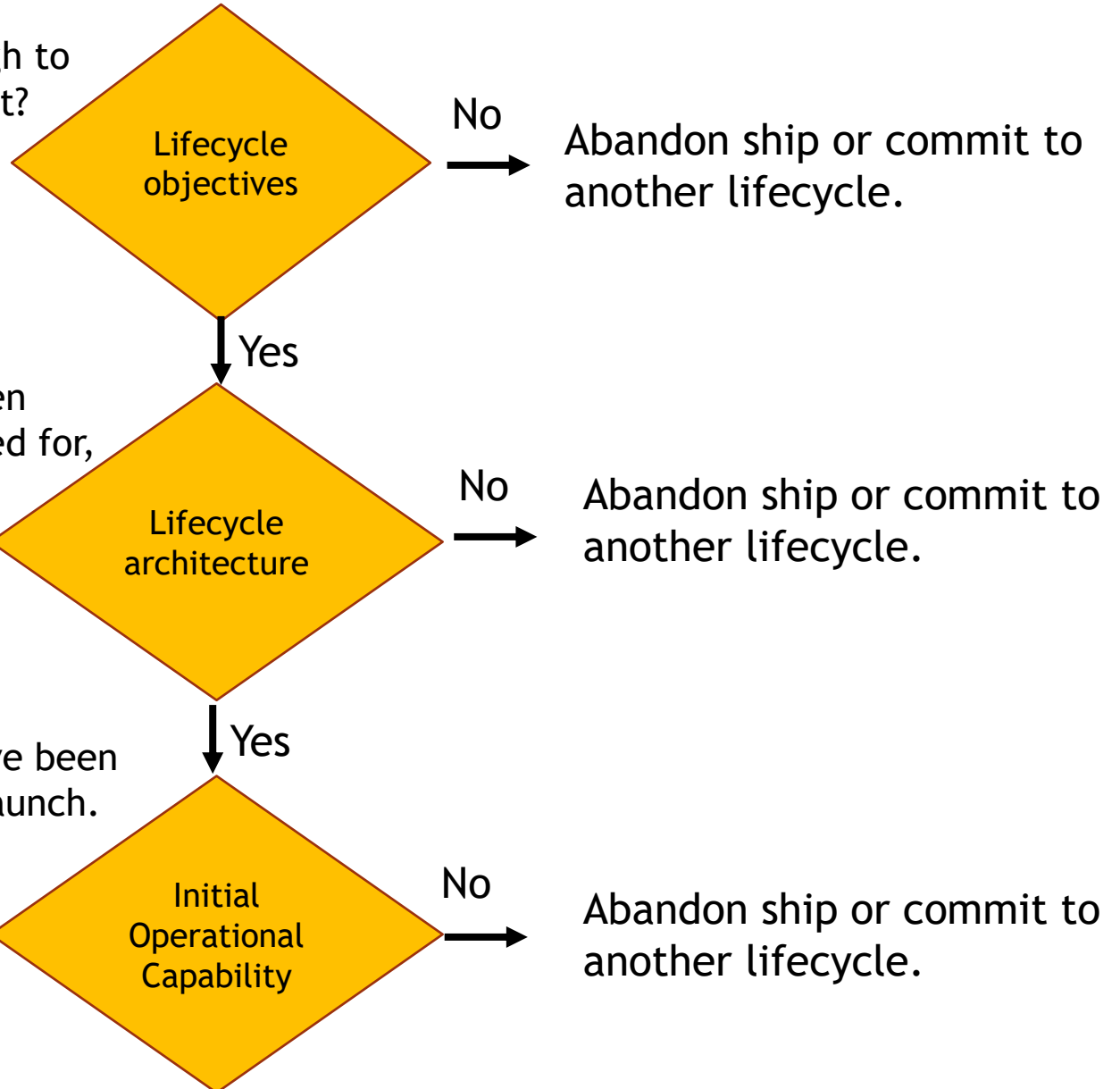
- Initially DIA intended that each individual airline be responsible for its own baggage-handling system.
- But then American Airlines (AA) decided to use DIA as its second-largest airport hub.
- AA commissioned BAE Automatic Systems to develop an efficient automated baggage-handling system.
- As the construction of airport progressed, a larger vision emerged : “creation of airport wide efficient integrated baggage handling system.
- DIA connected with BAE with its request.
- New plan however “underestimated the high complexity of the expanded system, newness of the technology, and the high level of coordination required among entities housed at DIA to be served by the system.”
- This was a huge change in the project : EXPANSION, but no one paid attention to the risk assessment, i.e. risks involved with changing the system.
- In the end, DIA settled with a much less ambitious plan, “six months after de-scaling the system, the airport was able to open and operate successfully”.

SPIRAL MODEL

- To address the inadequacy of the Waterfall model, in 1986, Barry Boehm proposed a more comprehensive lifecycle model was proposed : SPIRAL MODEL.
- Spiral model proposes a risk driven approach to the software process rather than a primarily document-driven or code-driven process.
- Risks in software projects: project costs overruns, changed requirements (DIA example), loss of key project personnel, delay of necessary hardware, competition from other software developers, technological breakthroughs which render the project obsolete.
- Central concept : “minimize risks by repeated use of prototypes”.
- At every step, risk analysis is performed. Decide : “Go/No-Go”. If risks are too great, the project is terminated.
- Works by building progressively more complete versions of the software by starting at the center of the software and working outwards.
- The model addresses the problem of requirement engineering through development of prototypes, and calls for risk assessment at each step of the lifecycle.

THREE ANCHOR POINTS IN SPIRAL MODEL

if the technical approach is well-defined enough to proceed, and the stakeholder condition are met?

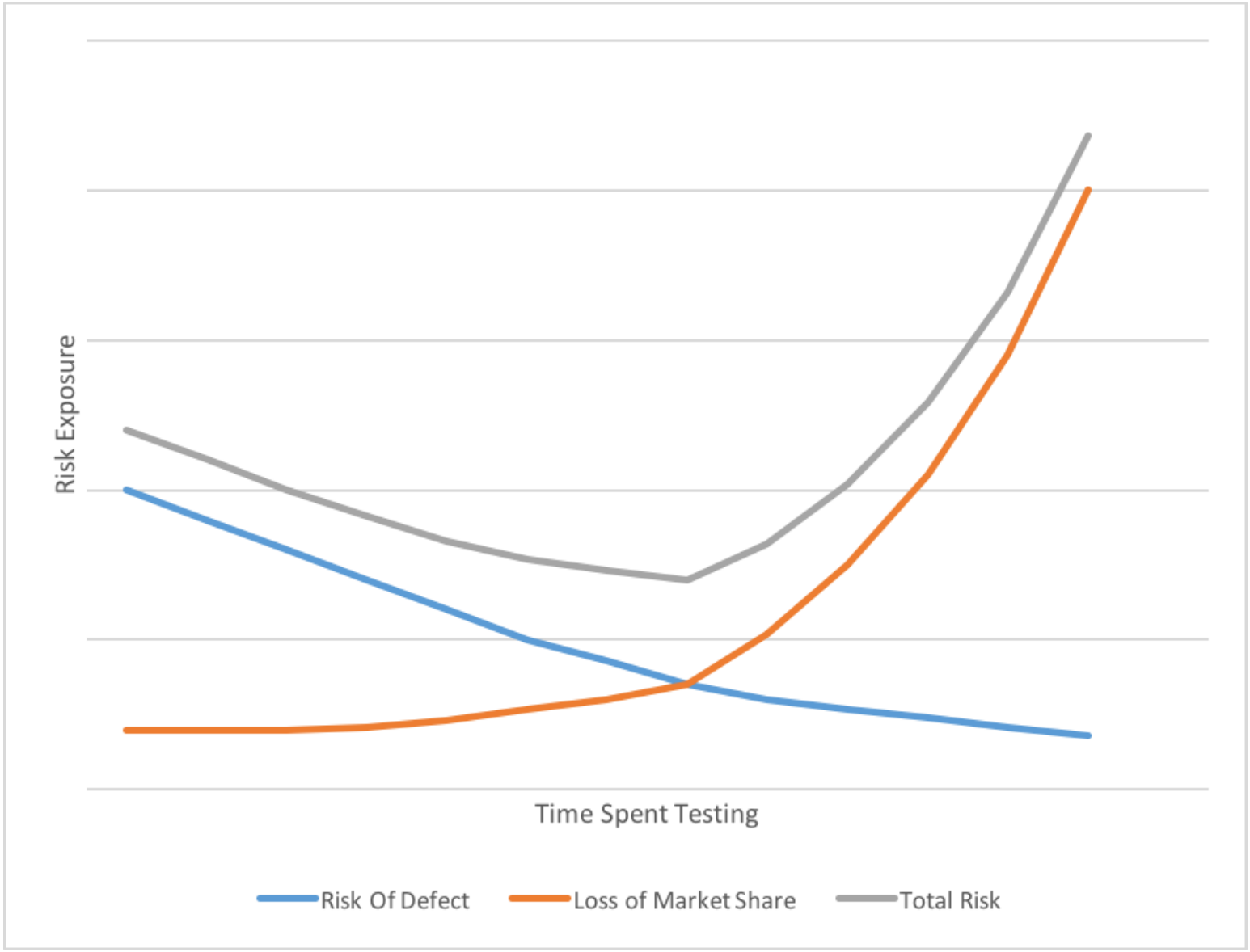


THE IMPORTANCE OF RISK ASSESSMENT

- Ensures that your team is highly flexible in its approach.
- But also aware and prepared for the challenges expected down the road of software development.
- Risk assessment mitigates risks and reduces the potential for setbacks.

EXAMPLE

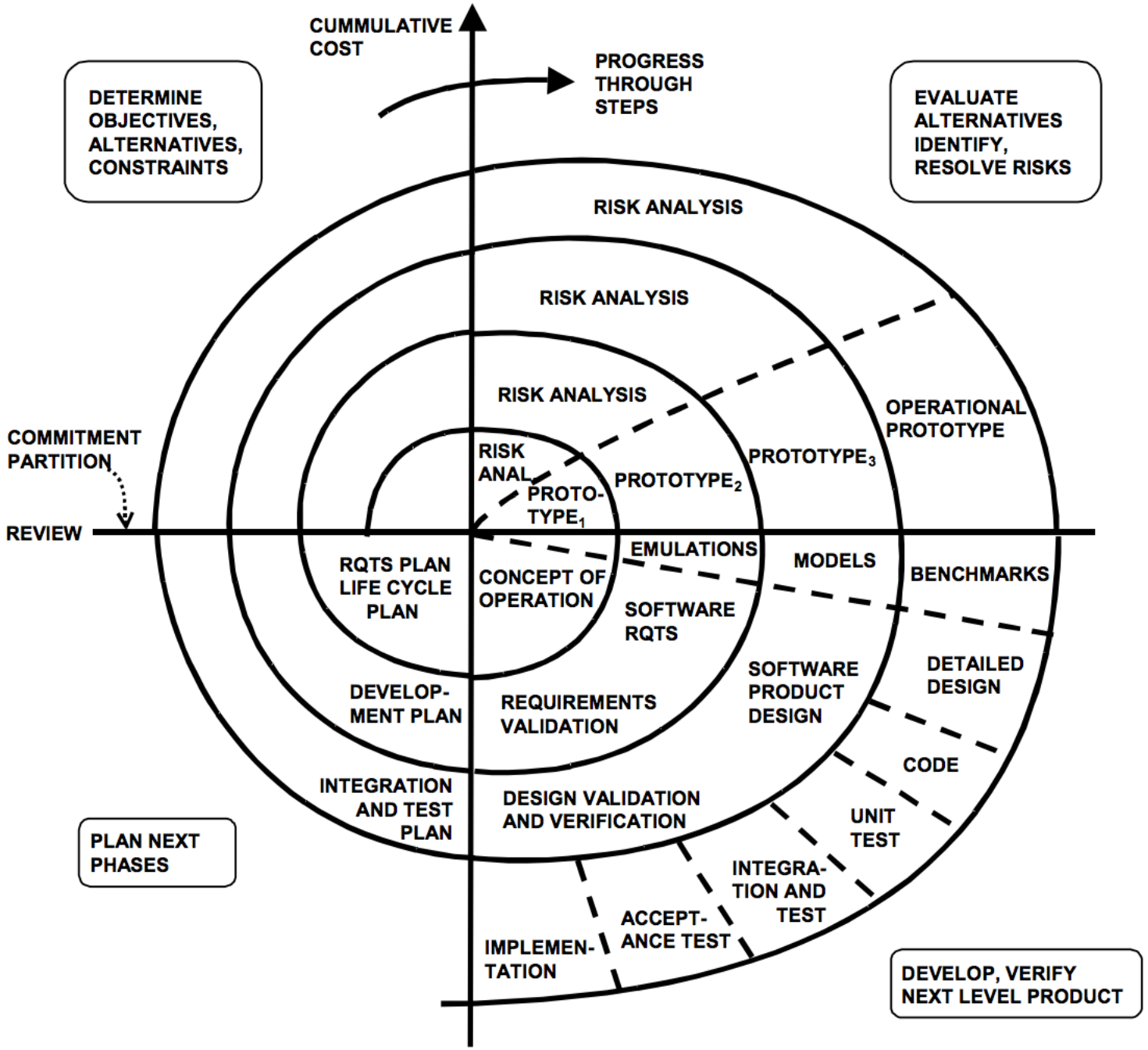
- You are planning to introduce new suite of features to your application.
- More features=happier customers, you think. Is that always the case?
- What if your customers become overwhelmed by the new interface?
- They miss the way the interface behaved previously. Then?
- This situation could be pretty bad – some people might jump out of the ship.
- Always best to gather more data and implement the changes more carefully.



After a certain point, risk of losing market share far outweighs the gain in testing.

BUT

Speed is just as important in software development.

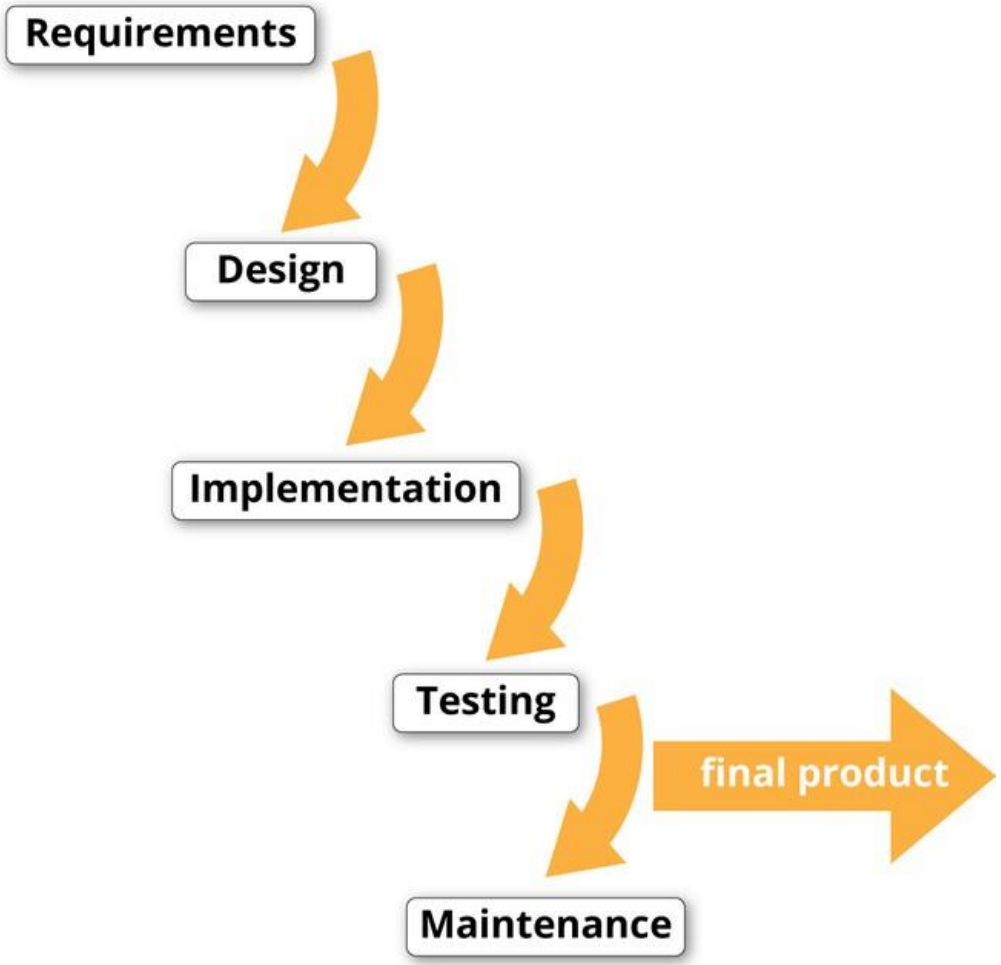


Spiral model

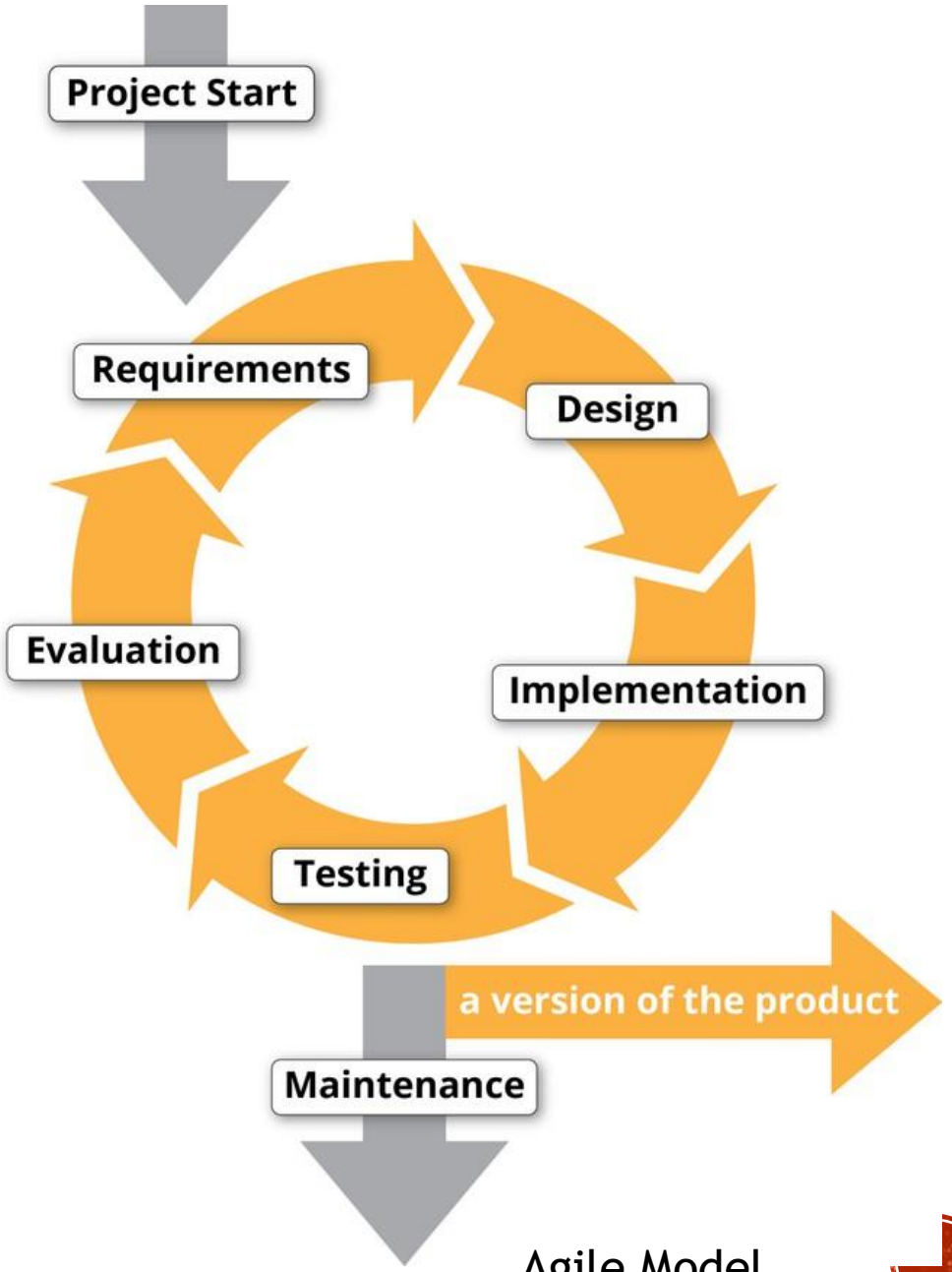
RISK ASSESSMENT TEMPLATE

Template	*Explanation*	*Example Phase*
Objectives	The goals of the software project	Significantly improve software quality
Constraints	Limitations which the project must meet	Within three years Without large-scale capital investment Without radical change to company standards
Alternatives	Possible ways to achieve the objectives	Reuse existing certified software Introduce formal specification and verification Invest in testing and validation tools
Risks	Potential risks for this phase	No cost effective quality improvement possible Quality improvements may increase costs excessively New methods might cause existing staff to leave
Risk Resolution	Strategies for reducing the risks	Literature survey, Pilot project, Survey of potential reusable components, Assessment of available tool support, Staff training and motivation seminars
Results	Results of applying risk resolution strategies	Experience of formal methods is limited - very hard to quantify improvements Limited tool support available for company standard development system Reusable components available but little reuse tool support
Plans	Development plans for the next phase	Explore reuse option in more detail Develop prototype reuse support tools Explore component certification scheme
Commitment	Resources needed to achieve the plans	Fund further 18-month study phase

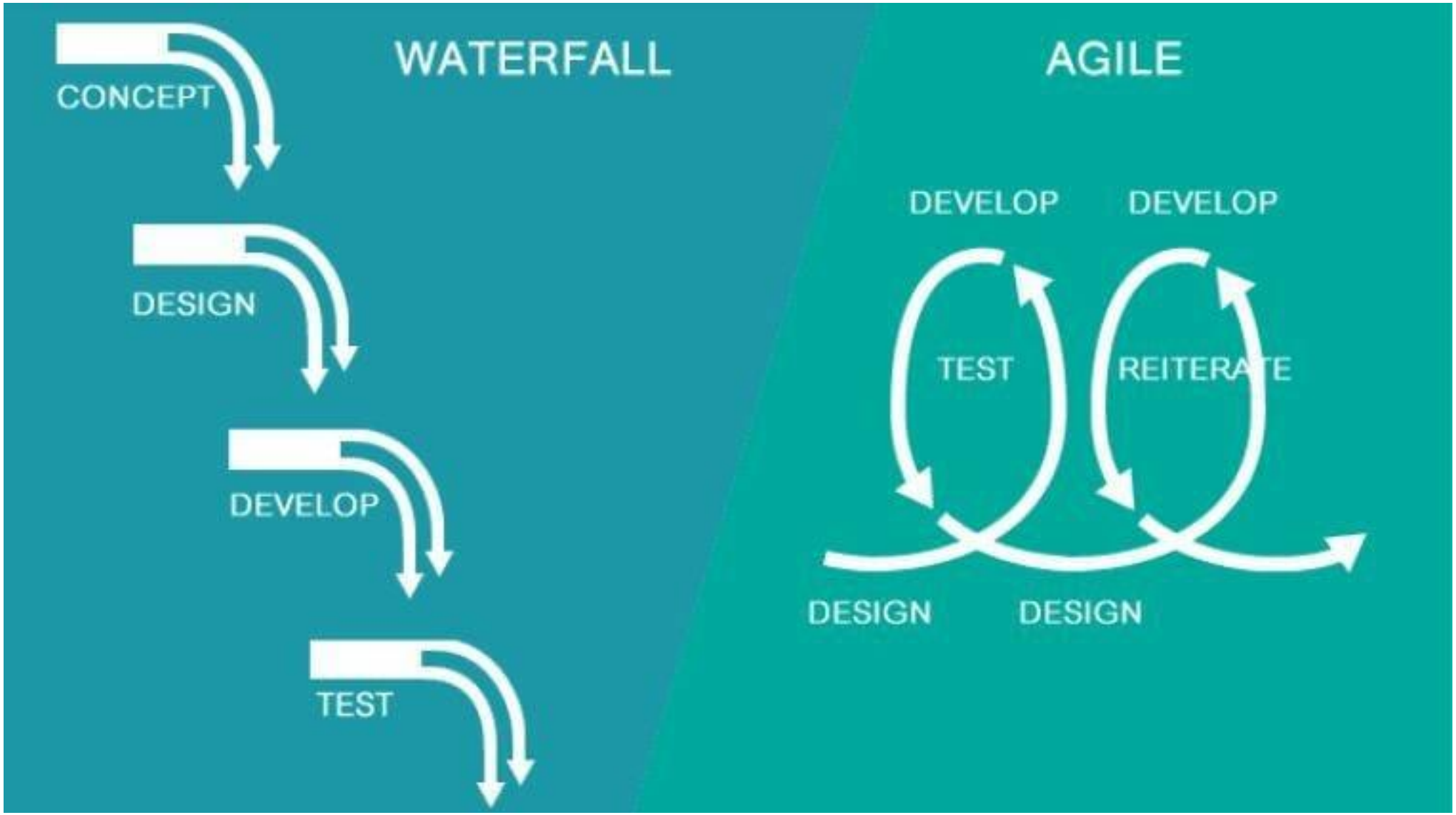
AGILE METHODS



Waterfall Model

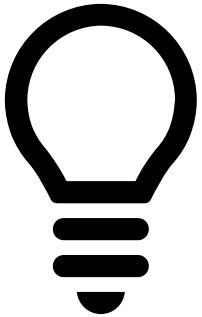


Agile Model


WATERFALL
CONCEPT
DESIGN
DEVELOP
TEST
AGILE
DEVELOP
DEVELOP
TEST
REITERATE
DESIGN
DESIGN

AGILE METHOD OF SOFTWARE DEVELOPMENT

- Agile emphasizes iterative development of software, building software in pieces.
- Agile teams typically work in short cycles – sprints in Scrum, which usually last 2 weeks.
- Both Agile and Waterfall are valid approaches, but different projects call for different methods.
- Agile allowed software projects to respond to changes.
- Agile solutions rather than Agile software.
- Agile is a philosophy, not a set of business practices.



Did you know that Agile was known as “light”, “lightweight” among the initiators of this methodology. They were not able to identify the right name for it that time.

The Agile Manifesto

Individuals and Interactions	over	Processes and Tools
Working Product	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.

www.agilemanifesto.org

12 PRINCIPLES OF AGILE

1. Customer satisfaction through early and continuous software delivery
2. Accommodate changing requirements throughout the development process
3. Frequent delivery of working software
4. Collaboration between the business stakeholders and developers throughout the project
5. Support, trust, and motivate the people involved
6. Enable face-to-face interactions
7. Working software is the primary measure of progress
8. Agile processes to support a consistent development pace
9. Attention to technical detail and design enhances agility
10. Simplicity
11. Self-organizing teams encourage great architectures, requirements, and designs
12. Regular reflections on how to become more effective

FRAMEWORKS IN AGILE

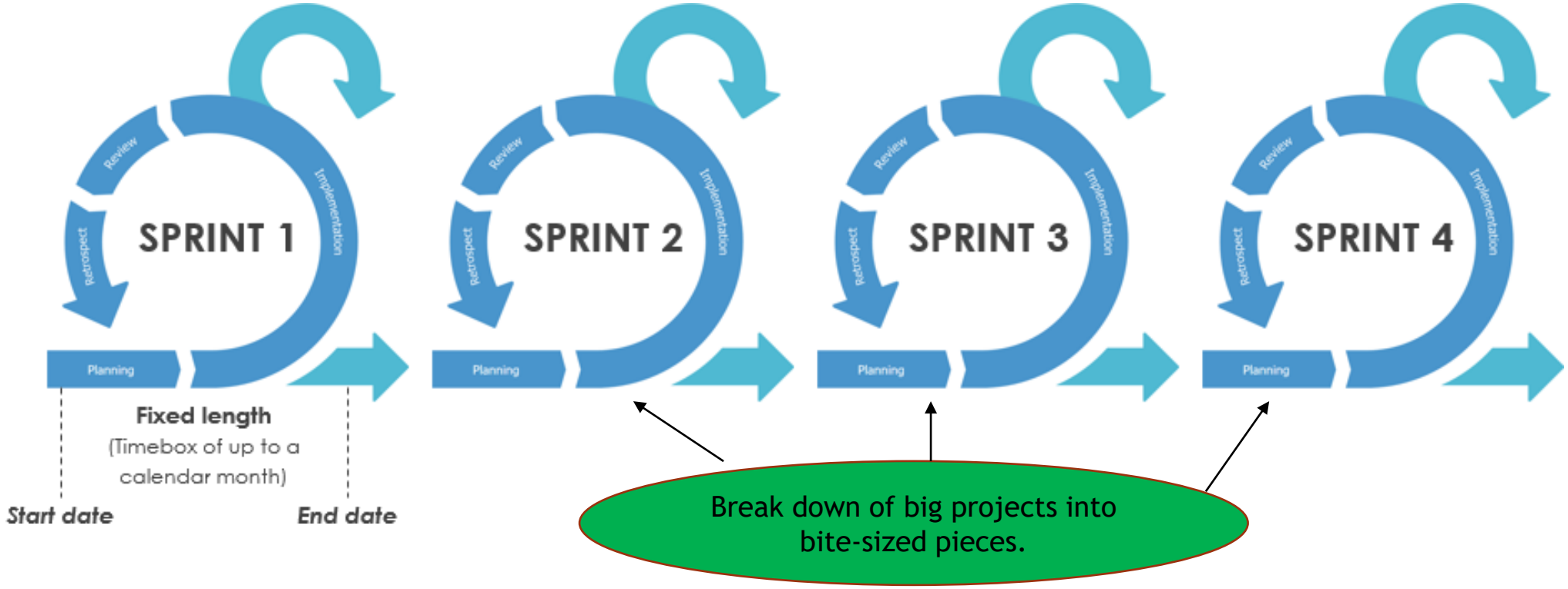
- Agile can be run using different frameworks. The popular ones are:
  - Scrum
  - Kanban
  - Extreme Programming
  - DSDM (Dynamic System Development Method)
- SCRUM is the most commonly used framework in the corporate sector for software development.

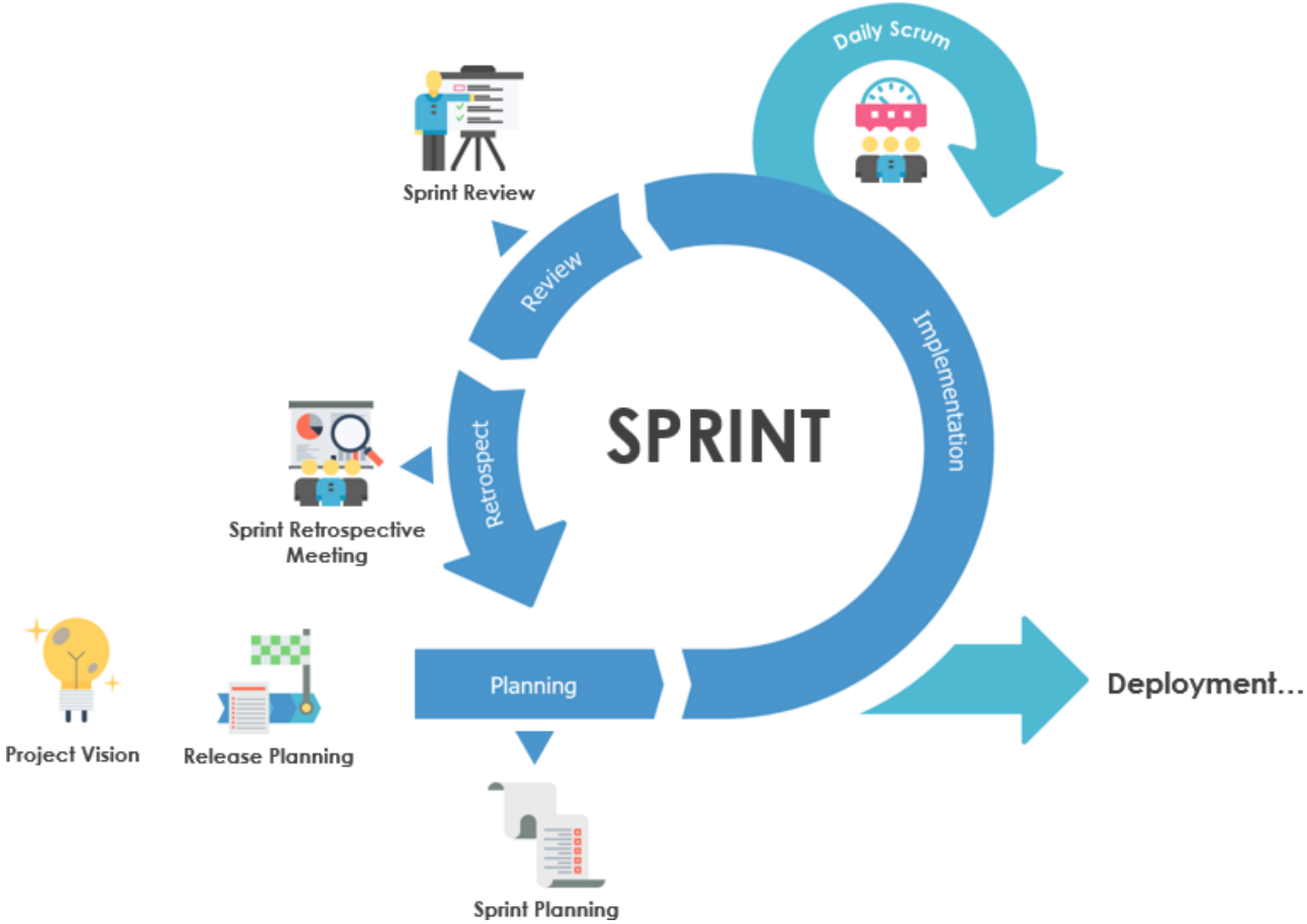
SCRUM BASICS

- Scrum is a framework which helps teams work together.
- Encourages teams to learn through experience, organizing self while working on a problem, reflecting on progress and failure to continuously improve.
- Scrum describes a set of meetings, tool, and roles that work in concert to help teams structure and manage their work.
- Let us watch this video.
- Some vital concepts:
  - Sprints
  - Sprint planning
  - Backlogs
  - Sprint reviews
  - Standups
  - Scrum master
  - Retrospectives

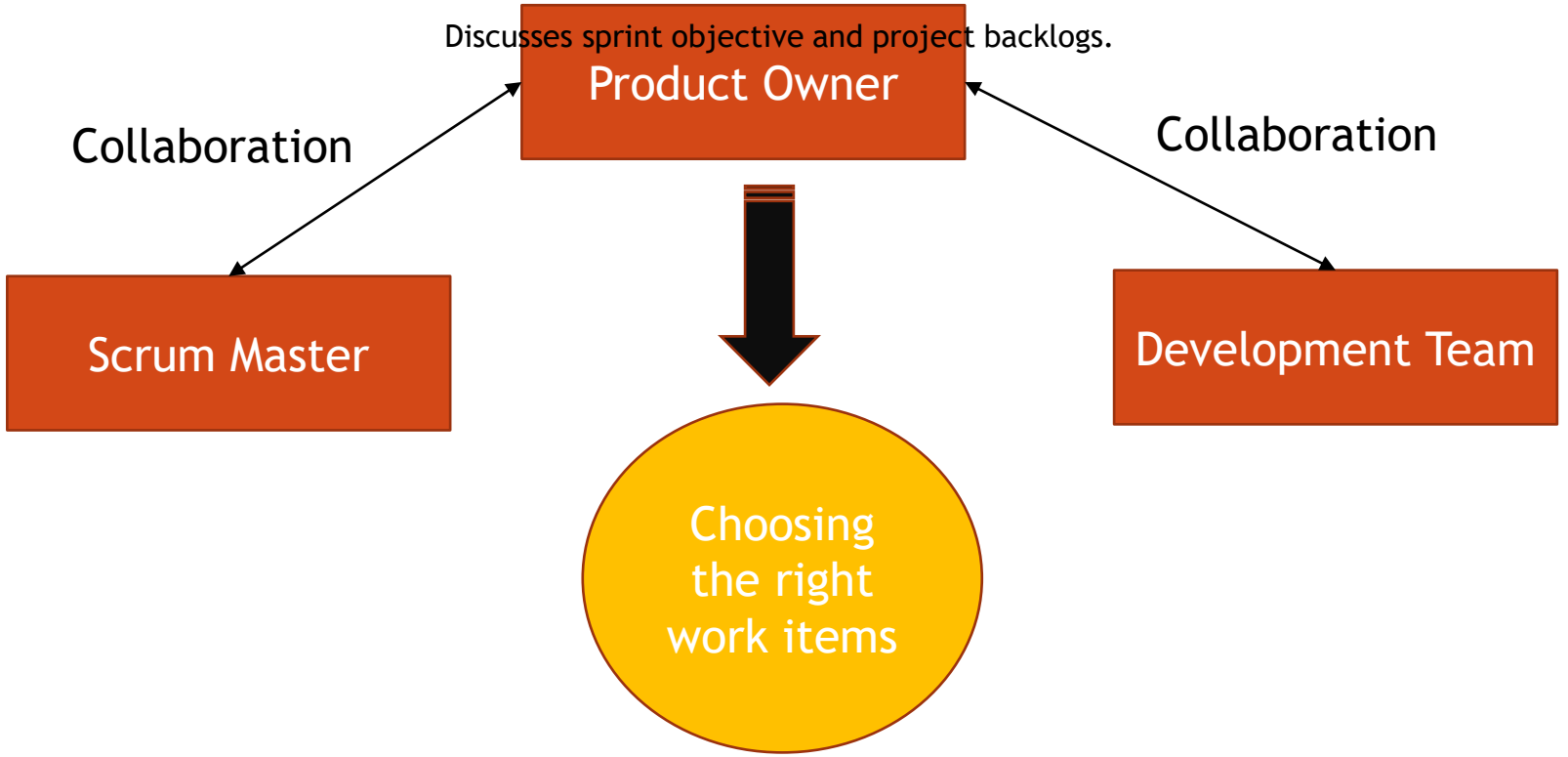
SPRINT

- Short, time-boxed period when a scrum team works to complete a set amount of work.
- Sprints are fundamental to Agile methodologies and scrum.
- Getting your sprints right will help your agile team ship better software, fewer headaches.

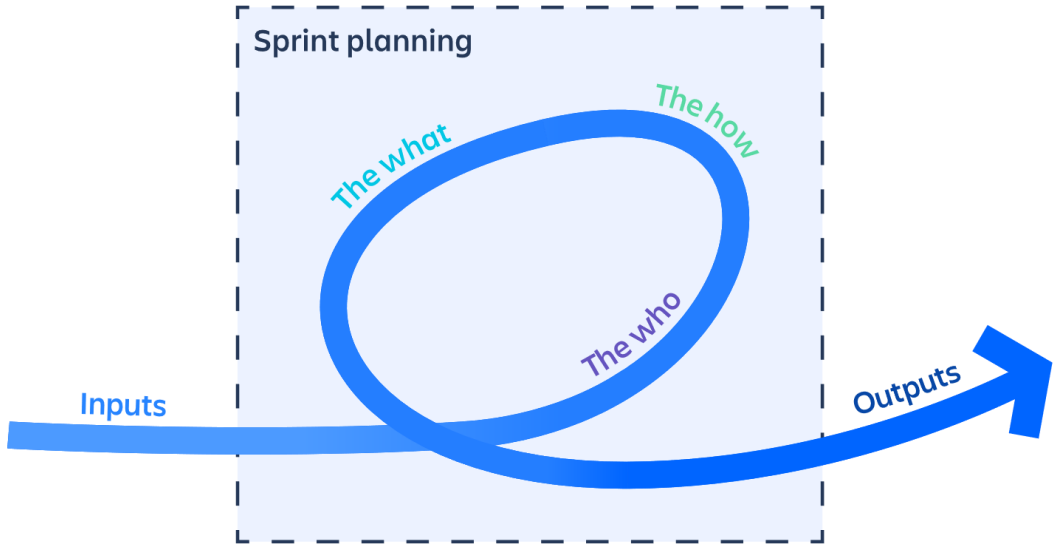



Daily Scrum
Sprint Review
Review
Implementation
SPRINT
Retrospect
Sprint Retrospective Meeting
Project Vision
Release Planning
Planning
Deployment...
Sprint Planning

- Sprints help teams follow the agile principle of "delivering working software frequently," as well as live the agile value of "responding to change over following a plan."
- The scrum values of transparency, inspection, and adaptation are complementary to agile and central to the concept of sprints.
- **SPRINT PLANNING :** planning of upcoming sprints. Two main questions asked:
  - What work can be accomplished in the upcoming sprint?
  - How will the chosen work get done?



- During a sprint, the team has daily stand-up meeting to discuss how the work is progressing.
- The goal of this meeting is to surface any blockers and challenges that would impact the teams ability to deliver the sprint goal.
- **SPRINT REVIEW :** After a sprint, the team demonstrates what they have completed. Showing work to team members.
- **SPRINT RETROSPECTIVE :** Team identifies the areas for improvement for the next sprint.



Do not pretend to know more than you do. Especially when planning sprints.

MUST REMEMBER

- The team must understand the **sprint goals.** Keeps everyone aligned and moving towards a common destination.
- **Do's**
  - Team must understand the sprint goals and how success will be measured.
  - Set your priorities and dependencies in order.
  - Use sprint planning meetings to identify details of the work that needs to be done. Talk about bugs, tasks.
  - Have a clear view of sprint tasks, do not be fuzzy or vague. Nail to the issue.
- **Don't**
  - Don't flood yourself with tasks in sprint planning. That is setting up for failure.
  - Do not take large amount of unknown or high-risk work. Break down large work into small pieces and shift some of them to next sprint.
  - Do not ignore the concerns of your team members. Listen to them.

PRO TIP:

Focus the first part of sprint planning on the objective of the sprint rather than the details of the backlog. By focusing on the goal rather than the work it is possible to find smart alternatives for how that goal is achieved.

FOCUS ON THE OUTCOME

- Easy to get bogged down by:
  - Focusing on which tasks should come first.
  - Who should do it.
  - How long will it take?
- When you start to work on a project, your knowledge is low. It improves as your progress.
- SCRUM is an empirical process, you learn by doing.
- Add clear and measurable goals to the user story.
- Get as much clarity as possible on the work team is focusing on. This leads to transparency in the work.
- Leaving something vague is much worse than describing something as a question.

PRO TIP:

Not knowing something is different from being vague. Don't ignore the unknowns, they are the reality of doing difficult work. But don't hide them by using vague words. Instead, be clear when you don't know something and frame the work in terms of gaining an understanding.

SPRINT PLANNING : BEST PRACTICES

- The goal of sprint planning is not to get every detail right, but “just right enough” for the next sprint cycle.
- A good sprint plan motivates everyone by defining the outcome and a clear plan for success.
- SCRUM framework aims at solving complex problems. Complex problems require empirical approach (learning by doing).
- Empirical processes are hard to plan – know this.

DAILY STAND-UP

- **Attendees: D**evelopment team, scrum master, product owner
- **When:** Once per day, typically in the morning.
- **Duration:** No more than 15 minutes. Don't book a conference room and conduct the stand up sitting down. Standing up helps keep the meeting short!
- **Purpose:** Stand-up is designed to quickly inform everyone of what's going on across the team. It's not a detailed status meeting. The tone should be light and fun, but informative. Have each team member answer the following questions:
  - What did I complete yesterday?
  - What will I work on today?
  - Am I blocked by anything?
- There's an implicit accountability in reporting what work you completed yesterday in front of your peers. No one wants to be the team member who is constantly doing the same thing and not making progress.

PRO TIP:

Some teams use timers to keep everyone on track. Others toss a ball across the team to make sure everyone's paying attention. Many distributed teams use videoconferencing or group chat to close the distance gap. Your team is unique. Your stand up should be, too!

RETROSPECTIVES

- **Attendees:** Development team, scrum master, product owner
- **When:** At the end of an iteration.
- **Duration:** Typically 45 minutes per week of iteration-e.g. a 90-minute retrospective after a two-week sprint.
- **Purpose:** Agile is about getting rapid feedback to make the product and development culture better. Retrospectives help the team understand what worked well–and what didn't.
- Retrospectives aren't just a time for complaints without action. Use retrospectives to find out what's working so the team can continue to focus on those areas. Also, find out what's not working and use the time to find creative solutions and develop an action plan. Continuous improvement is what sustains and drives development within an agile team, and retrospectives are a key part of that.

PRO TIP:

Even if things are going well across the team, don't stop doing retrospectives. Retrospectives provide ongoing guidance for the team to keep things going well.

BACKLOGS

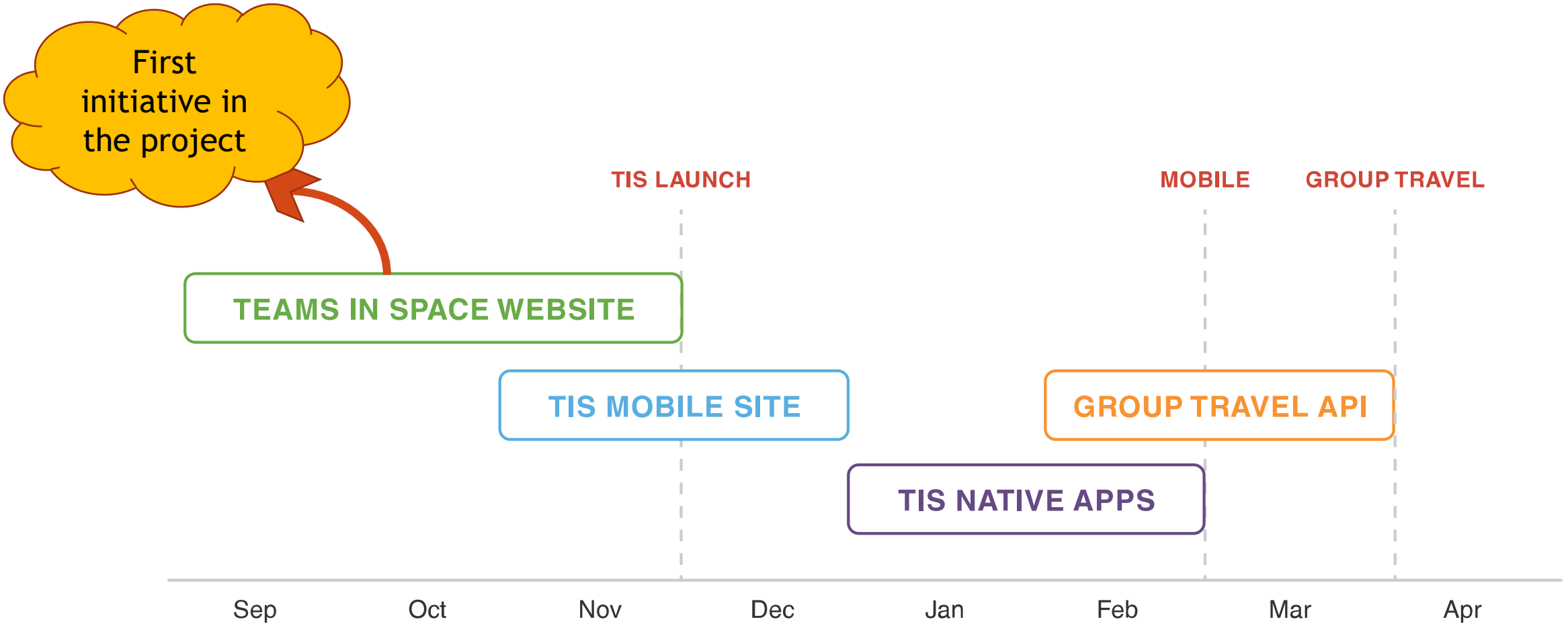
- Product backlog is the prioritized list of work for the development team.
- Most important items shown on the top of the product backlog, so the team knows what to deliver on priority and in sequence.
- Team **roadmap** and **requirements** are essential to provide foundation for product backlogs.
- It is important to keep the backlogs healthy. Regular maintain it to keep pace with the program.
- Product owners need to review backlogs before iteration planning.
- The product owner is free to re-prioritize work in the backlog at any time due to customer feedback, refining estimates, and new requirements.

PRO TIP:

Once the backlog grows beyond the team's long term capacity, it's okay to close issues the team will never get to. Flag those issues with a specific resolution like “out of scope” in the team's issue tracker to use for research later.

Backlogs prompt debates and choices that keep a program healthy–not everything can be top priority.

Imagine a fictitious product : TEAMS IN SPACE



ROADMAP OF TEAMS IN SPACE

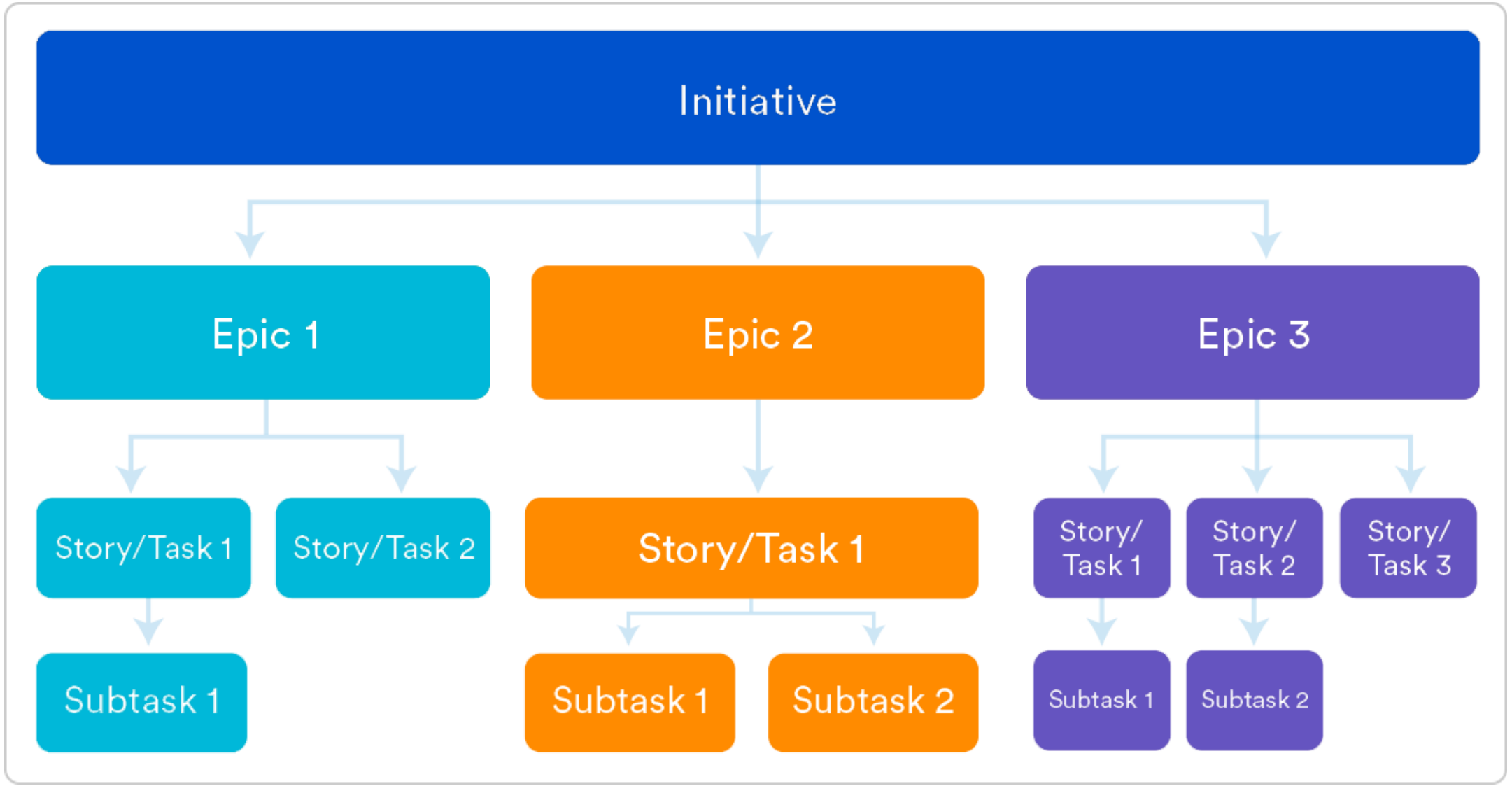
TEAMS IN SPACE WEBSITE

EPICS : Large bodies of work broken down into smaller parts.

User management	Travel reservations	Promos and offers
Create an account	Book space travel	Percentage discounts
Stored payment info	Book a hotel	Companion flies free
Linked family profiles	Book rental space	Customer loyalty
Travel preferences	Book group tickets	Family discounts

USER STORY AND EPIC IN AGILE

- **User story**
  - an informal, general explanation of software feature written from the perspective of the end user.
  - Articulates the value of software feature to the customer.
  - Is user story a software system requirement?
  - Written in non-technical language to provide context for development team.
- Agile User Story
  - The smallest unit of work in an agile framework.
  - An end goal, not a feature.
  - Few sentences in a simple language that outline the desired outcome.

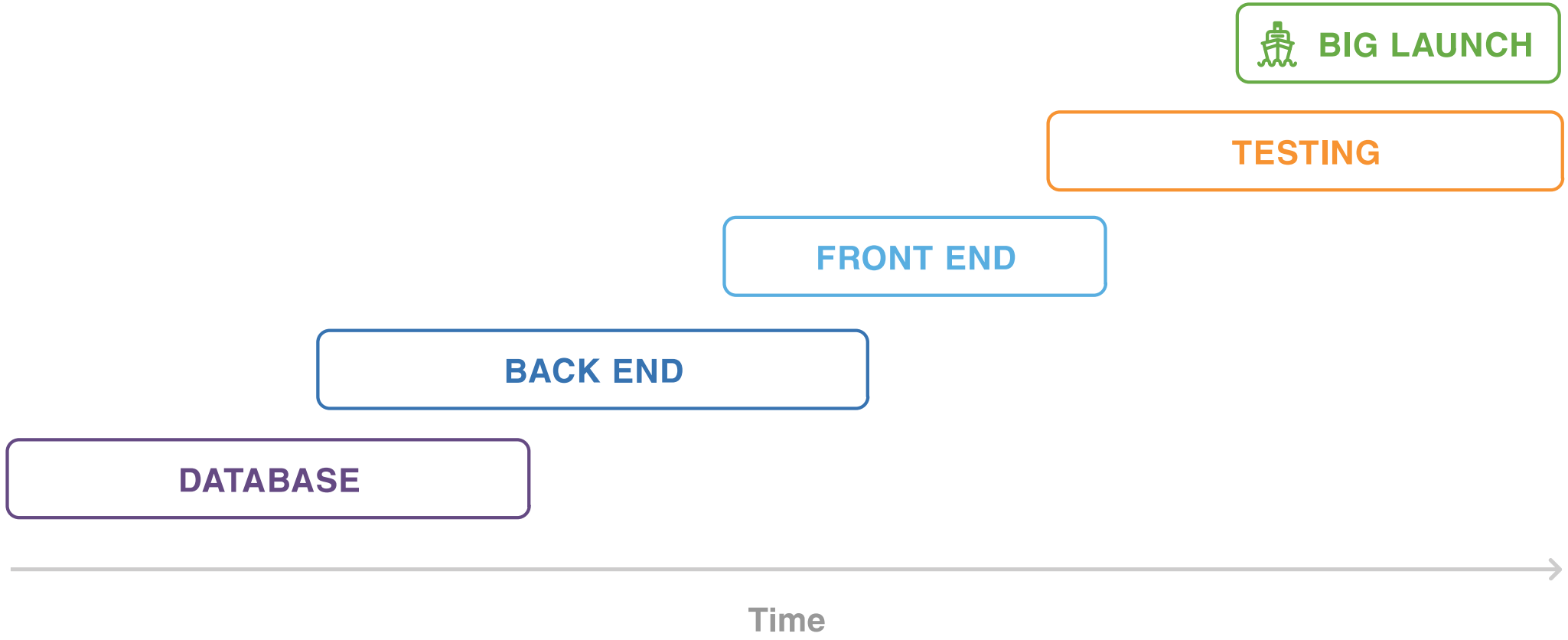

Initiative
Epic 1
Epic 2
Epic 3
Story/Task 1
Story/Task 2
Story/Task 1
Story/ Task 1
Story/ Task 2
Story/ Task 3
Subtask 1
Subtask 1
Subtask 2
Subtask 1
Subtask 2

HOW TO USE AGILE IN SOFTWARE DEVELOPMENT

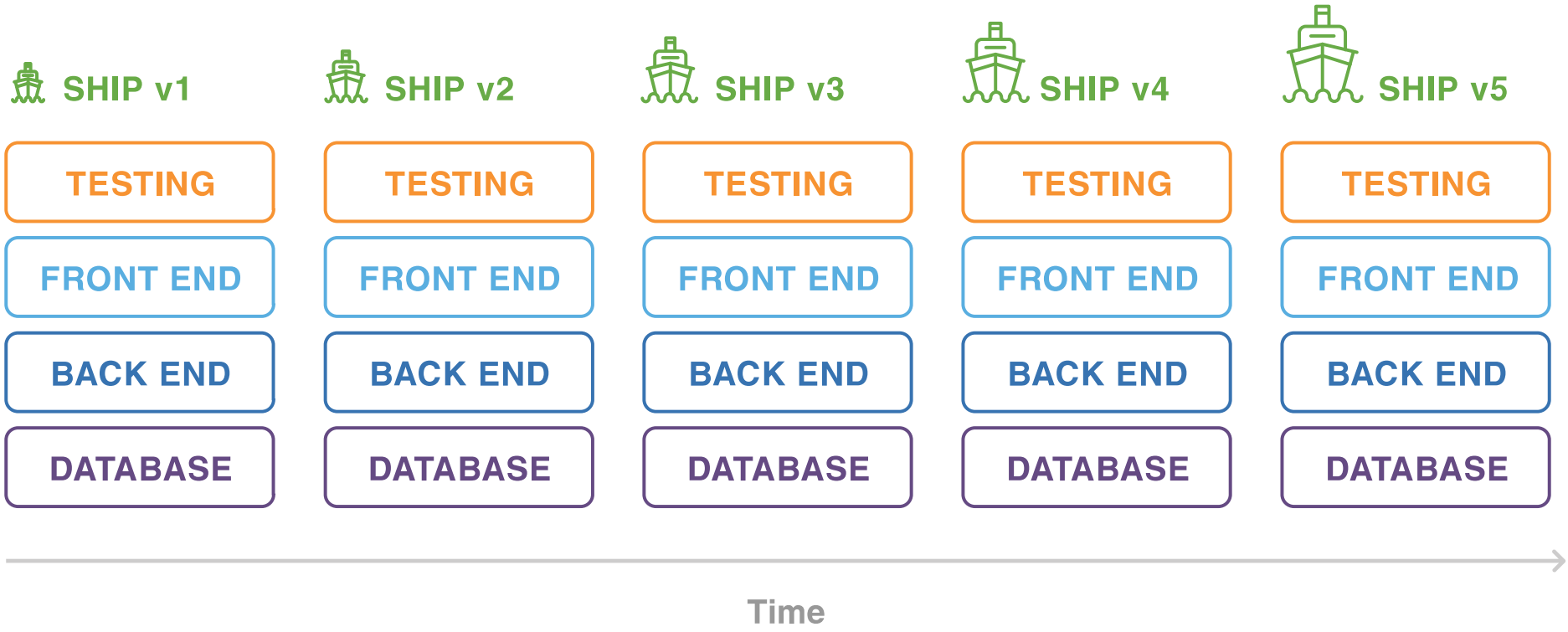
- In general, Agile methodologies call for breaking down software into smaller chunks known as “user stories”.
- This accelerates feedback loop and align product features with market need.

BENEFITS OF AGILE SOFTWARE DEVELOPMENT

- ✓Continuous customer contact.
  - Traditional project management methods connected with customers at the start and end of the project. Unless customer requirements captured correctly in the beginning, rate of failure high. What if customer need change in between? Ongoing contact throughout the project with the client, iterative deliveries ensure team is on track. Thus, end product will be exactly what customer demands.
- ✓The ability to adapt.
  - Mid-change in customer requirement in traditional model would involve high project costs and increase in schedule. Agile incorporates changes with minimal effort.
- ✓Faster delivery.
  - Incorporates a continuous development approach, the client gets working version of the product at shorter intervals of time, typically every two to four weeks.
- ✓Lower project risk.
  - Team developing product versions regularly, customer feedback early on, project failure risk is minimized. Breaking large project into iterations reduces risk of failure. Small problems can be detected early on, rather than later in product development.
- ✓Ongoing innovation.
  - Supports collaboration and continuous improvement. This leads to innovation and development of new products and features.

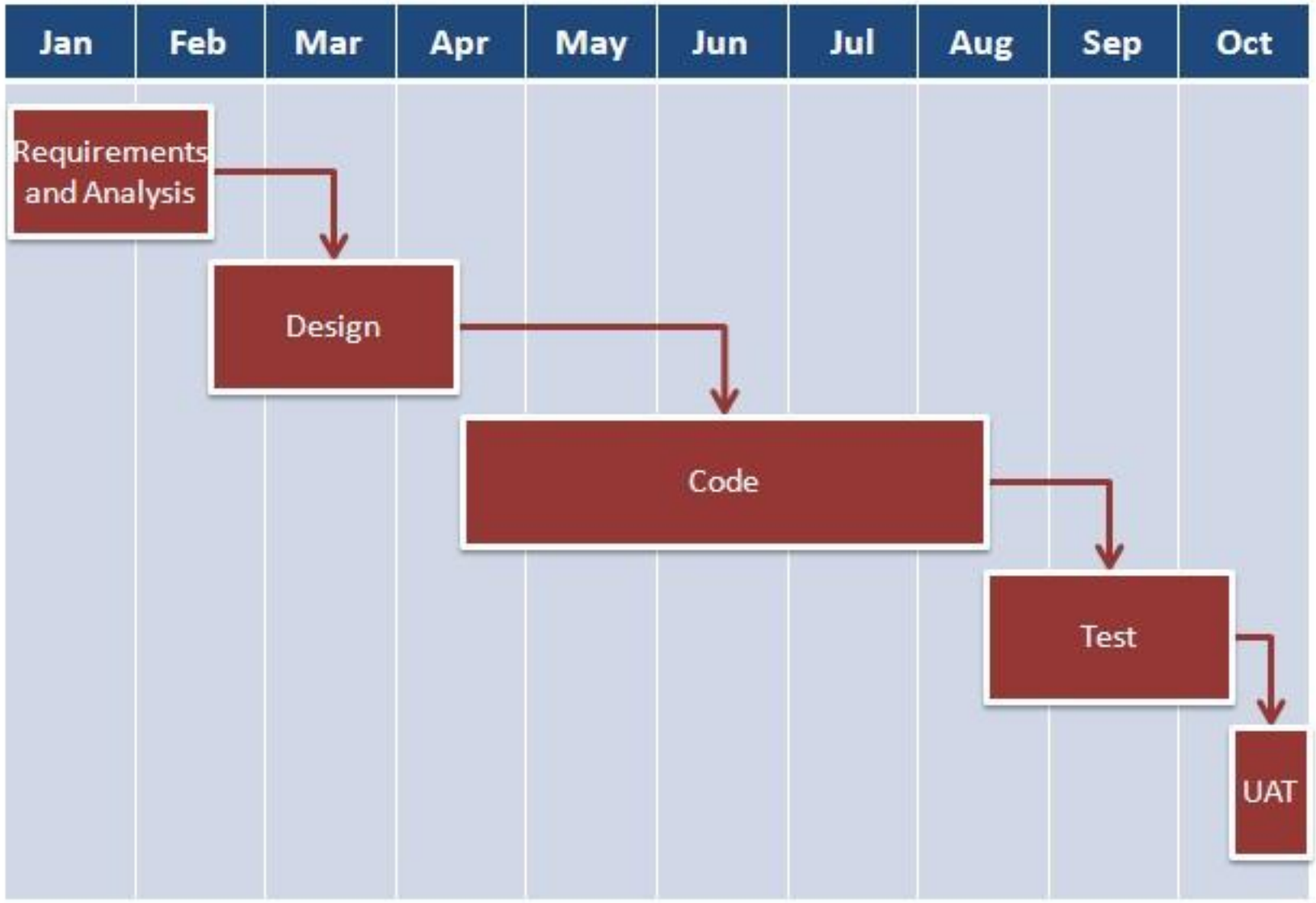


WATERFALL RELEASE PROCESS

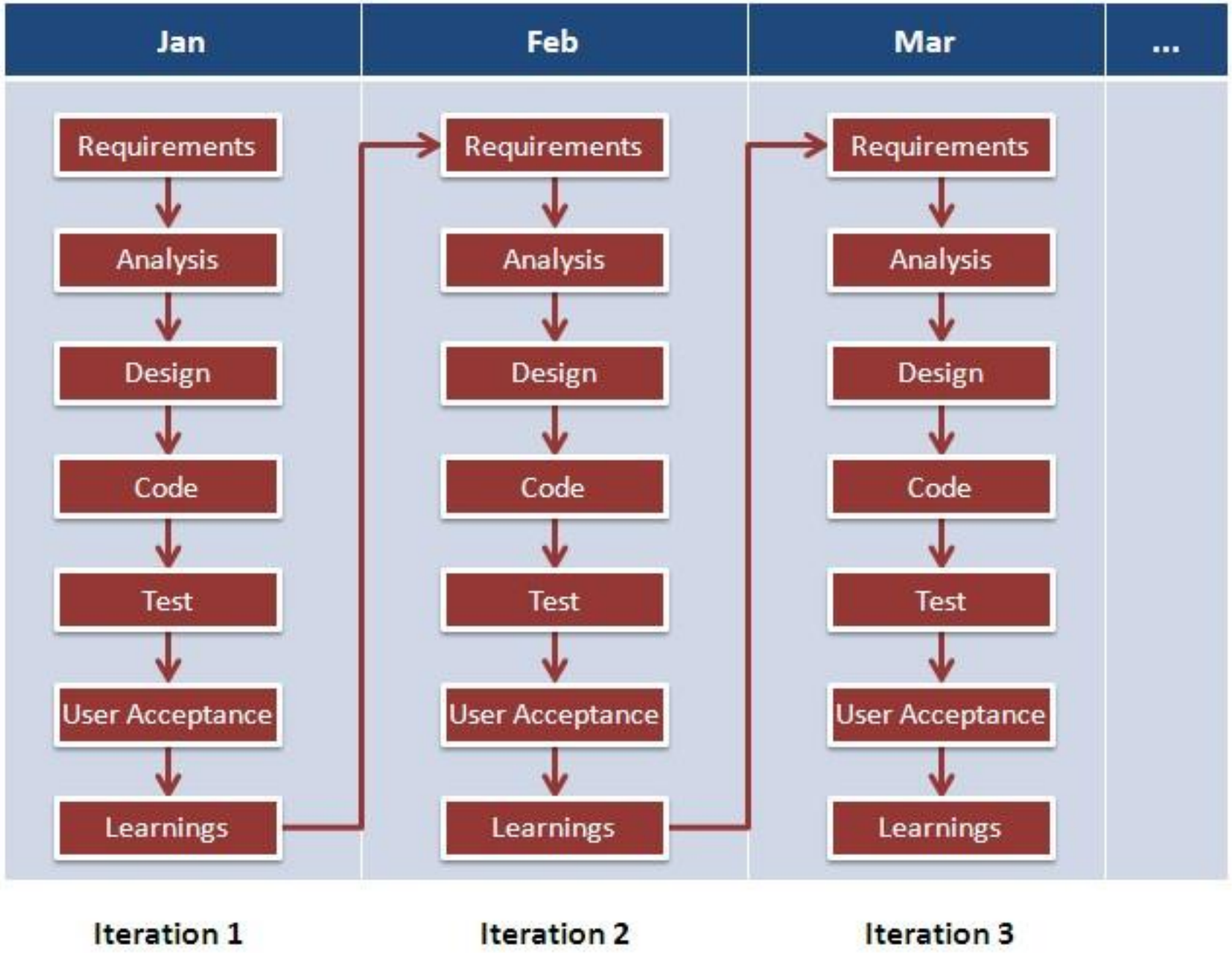


AGILE RELEASE PROCESS

WATERFALL RELEASE PROCESS



AGILE
RELEASE PROCESS



IS AGILE METHODOLOGY ALWAYS THE BEST OPTION?

- Outcome of project is stable and well-understood.
  - Agile mostly used to help reduce the cost of change and uncertainty on a project by breaking it down into iterative project management stages.
  - What if there is little uncertainty and possibility of change?
  - Eg. Working in industries where project requirements are already known.
- Project is a repeatable deliverable
  - Agile is not designed for reproducibility.
  - Eg. Client asks you to deliver identical software for different devices.
- Stakeholders do not want Agile.
  - Agile product required continuous contact with your stakeholders. If project is low value or low risk, stakeholders may prefer traditional approach. They get involved only in key phases.
- Company cannot support Agile.
  - Agile is not well understood, training is not done in the company, then you cannot use Agile. Organization cannot support daily collaboration. Stakeholders are not ready. Organization requires heavy documentation and test reports. Agile may be too costly to adopt.

START THINKING…

- Of the software development philosophy you are going to follow in your project.
- Please arrive at a proper rationale for the same.