كلية الهندسة
FACULTY OF ENGINEERING

Agile Software Development

Instructor: Dr. Mouhib Alnoukari

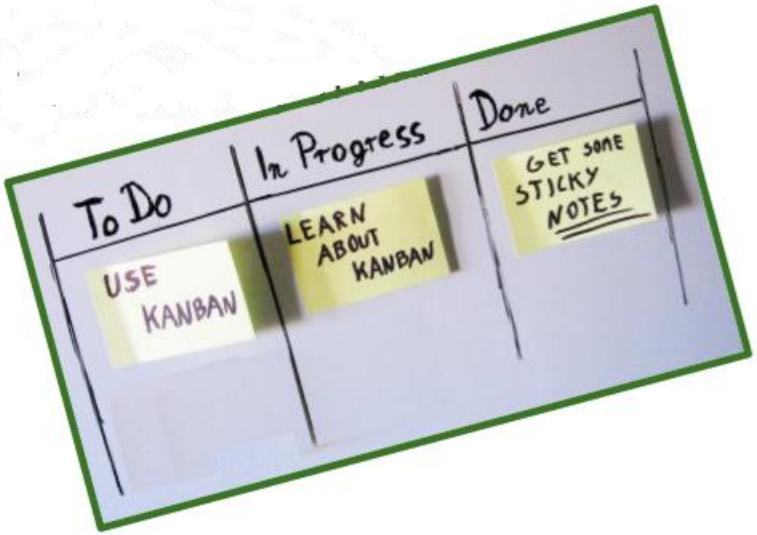
Agile Overview

Rapid software development

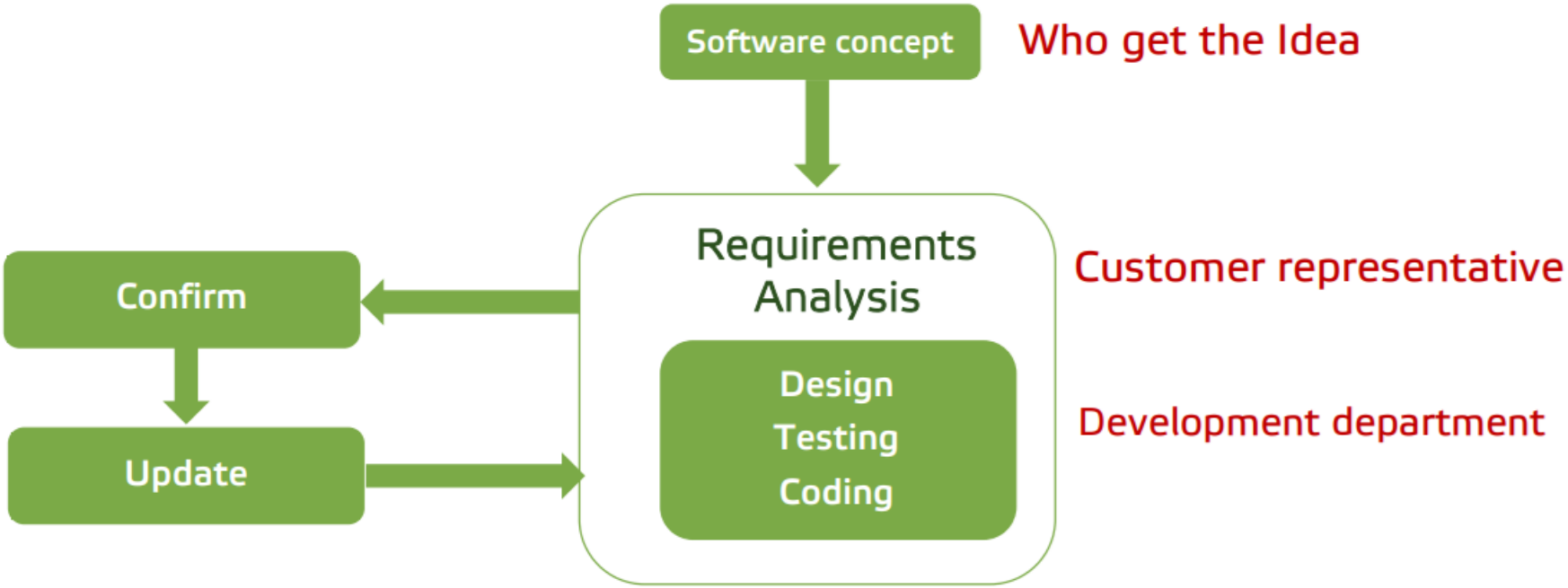
- Why?
  - Need to react to changes more quickly than 2 year long waterfall projects
  - 2 years and then you got the design wrong anyway! Small deliveries aren't abstract
- How?
  - Goal - Deliver working software quickly
    - Compromise - less functionality in a delivery, not lower quality
    - Less documentation
  - Focus on the code rather than the design
  - Interleave
    - Specification, design and implementation are inter-leaved
  - Deliver small versions and get user (stakeholder) input

Lean

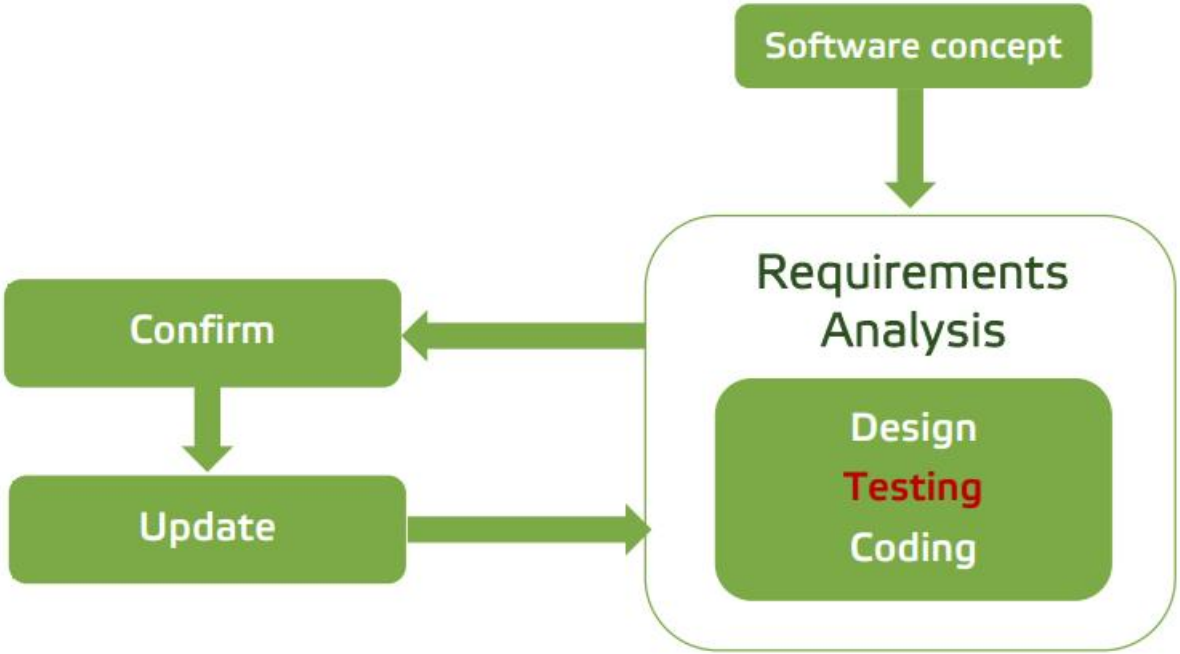
Taiichi Ono is founder of TPS (Toyota Production System) 1988. Approach, where lean manufacturing was used as part of the system.



Agile

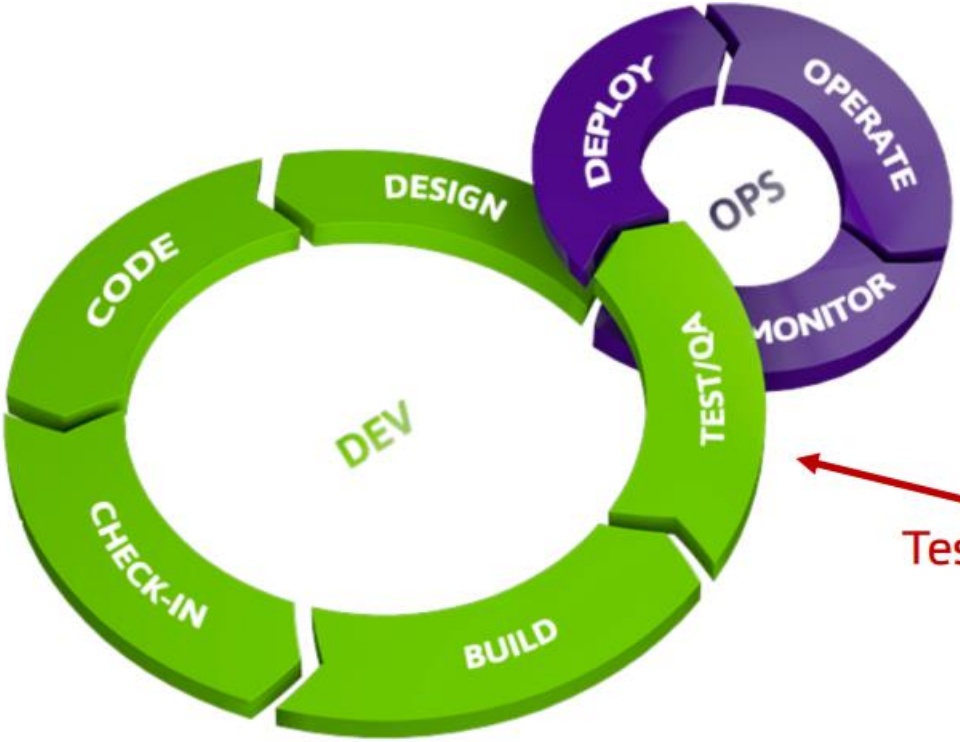

Software concept
Who get the Idea
Requirements Analysis
Customer representative
Confirm
Update
Design
Testing
Coding
Development department

Agile



Don't we need testing department?

DevOps



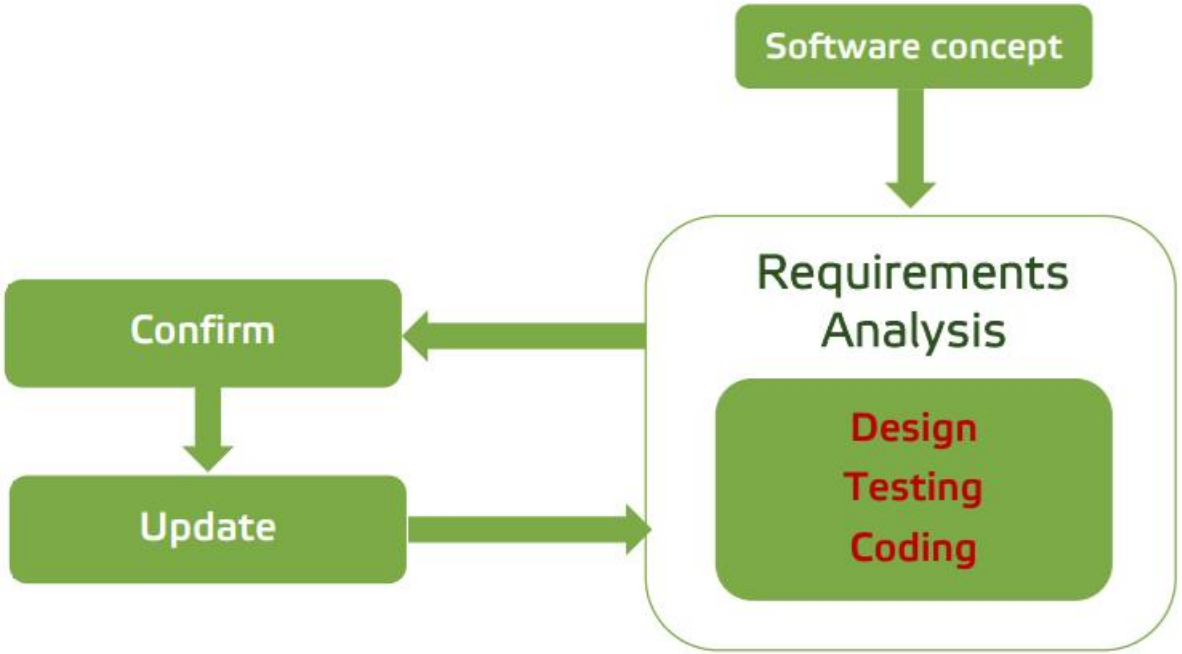
Testing departament

Types of Tests

Unit testing — Development team

Integration testing
Regression testing
Load testing
Penetration testing
Mutation testing
Soak testing
— Testing department

Agile


Software concept
Requirements Analysis
Design
Testing
Coding
Confirm
Update


Same person?!

Agile – Team Structure

Functional

Common functional expertise

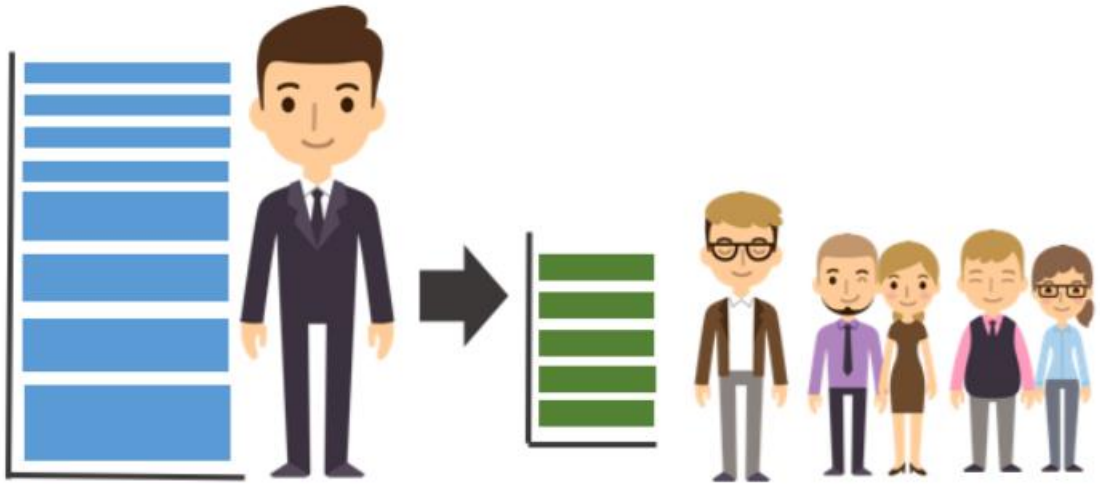
Cross-functional

Representatives from the various functions

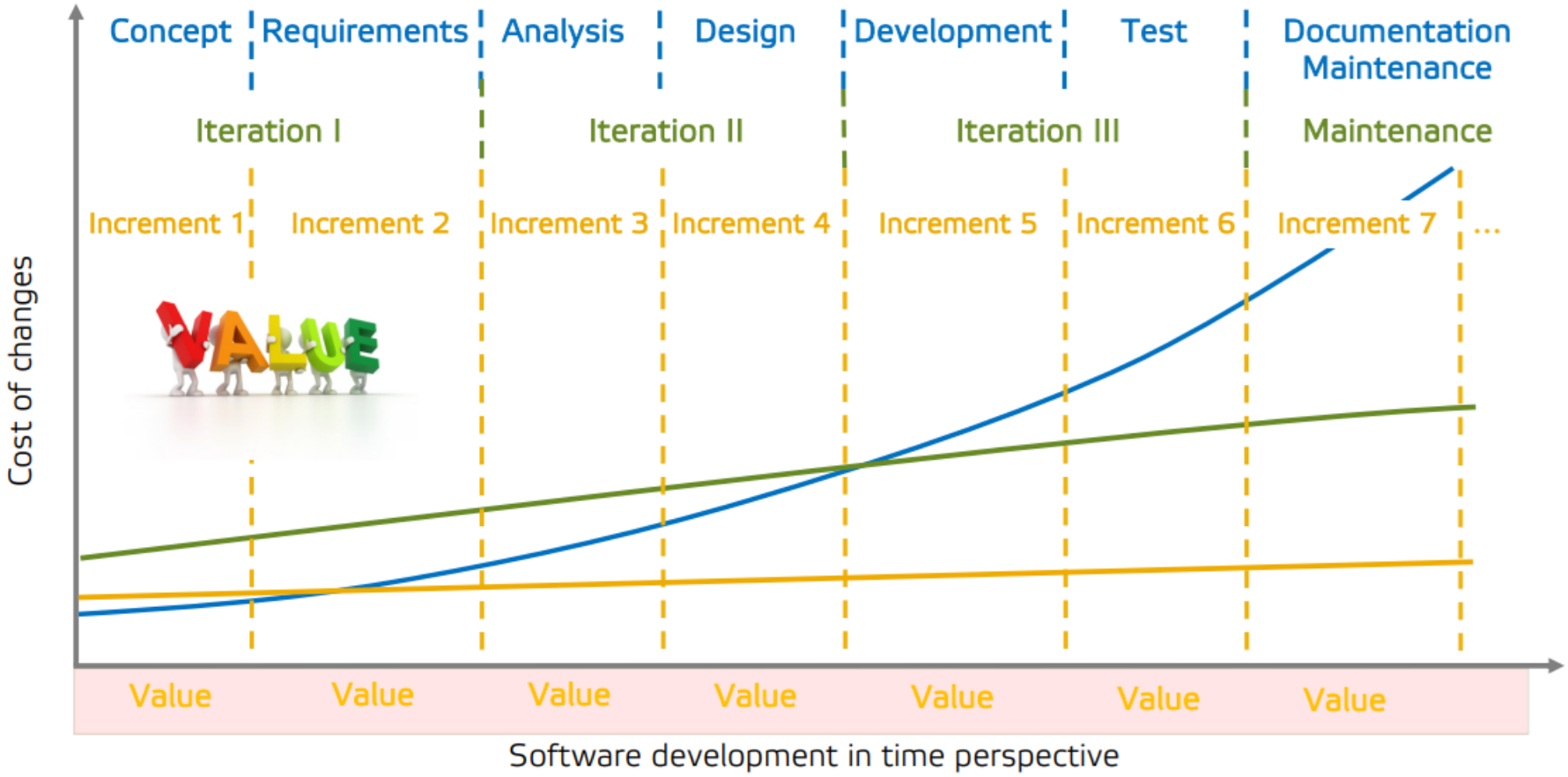
Team overlap

Agile – Product Owner

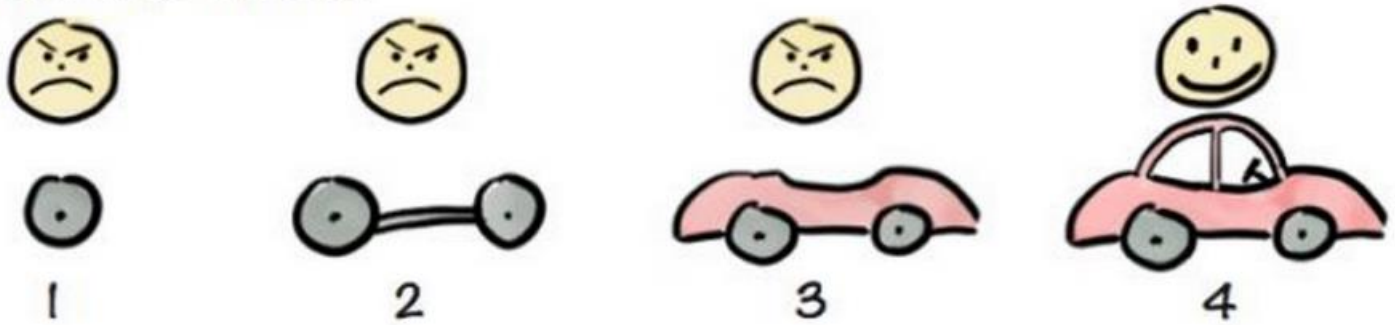
- Customer representative (part of the development team)



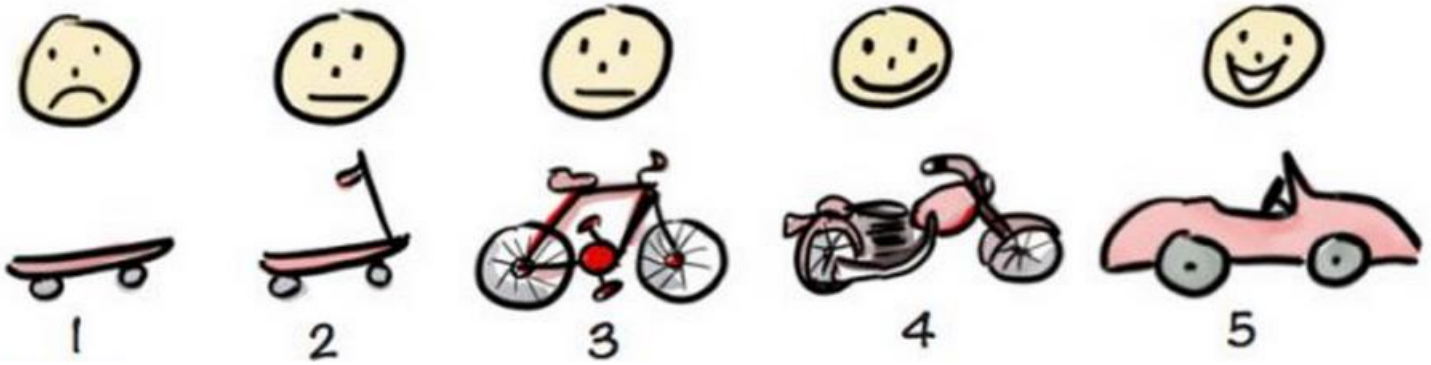
Agile – Value


Concept
Requirements
Analysis
Design
Development
Test
Documentation
Maintenance
Iteration I
Iteration II
Iteration III
Maintenance
Increment 1
Increment 2
Increment 3
Increment 4
Increment 5
Increment 6
Increment 7
...
VALUE
Cost of changes
Value
Value
Value
Value
Value
Value
Value
Software development in time perspective

Not like this....



Like this!



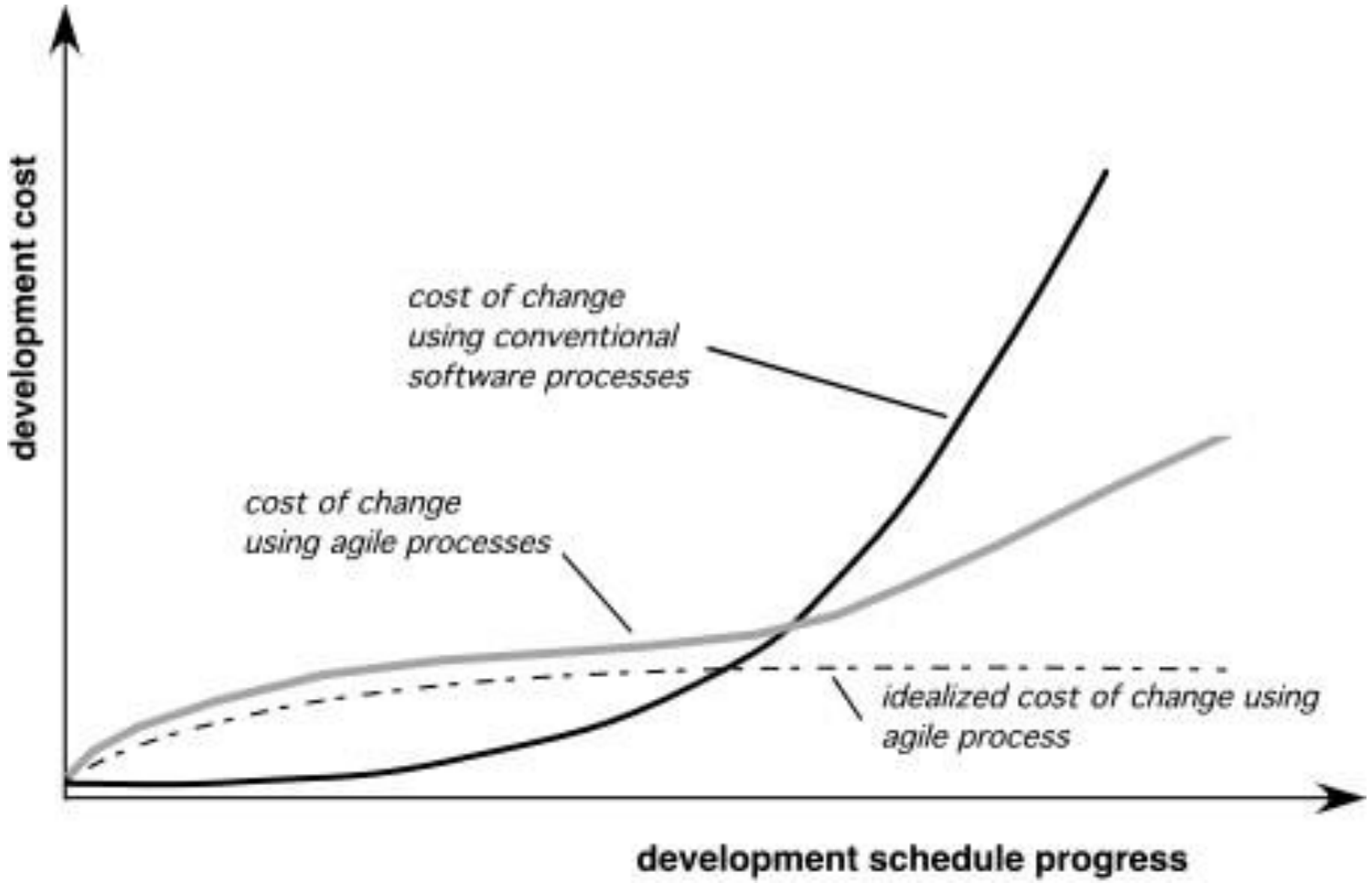
What is “Agility”?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

Agility and the Cost of Change



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.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

The Agile Manifesto

- On February 11-13, 2001, The Lodge at Snowbird ski resort in Utah witnessed a meeting between seventeen advocates of lightweight methodologies, seeking to discuss and identify any common ground for software development:
  - The word “Agile” was chosen to lay emphasis on ways that software development is expected to react to changing business circumstances.
  - The seventeen attendees formed a group and christened themselves as “The Agile Alliance.”
  - Each of the experts spent time listening to others and presenting their nuggets of wisdom based on their individual experiences that started several years ago before this meeting was decided.
  - The Agile Software Development Manifesto emerged. The manifesto had four core values and is discussed in the next section.
  - Along with the Agile Manifesto, the experts also agreed on twelve detailed statements that further explains agility.

Four Core Values of the Agile Manifesto

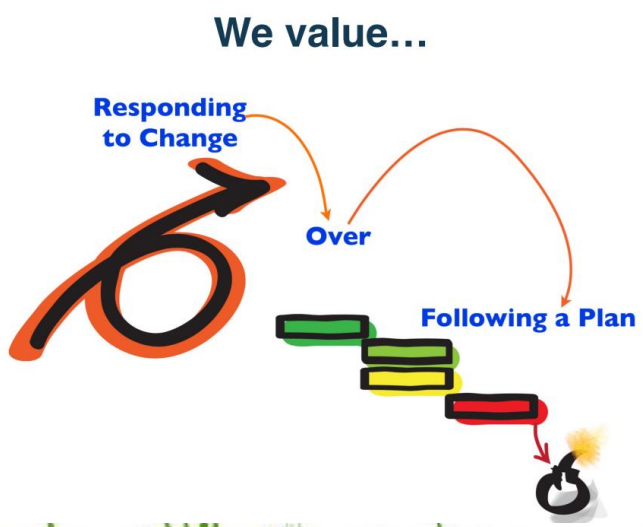
The Manifesto for Agile Software Development states:

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

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. 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.

Four Core Values of the Agile Manifesto



1. Individuals and Interactions over Processes and Tools

- This value emphasizes the people in the team and the interactions within them rather than a heavy-process mindset and an armory of project management tools.
  - The value doesn’t say that processes are an overhead or cannot help in delivering projects, nor does it say that Agile projects (of any scale) are completely devoid of processes and tools
  - This value states that processes and tools might be helpful, but without effective collaboration between individuals in an Agile project team delivery is not possible
  - Agile teams are empowered to tailor the process based on what they consider as essential in the specific context of their project (and the organization) and evolve the same through periodic introspection

Pros and Cons

Pros Individuals and Interactions Have High Value

- Communication is quick, clear and effective.
- Teamwork becomes strong as people work together.
- Development teams can self-organize.
- Development teams have more chances to innovate.
- Development teams can customize processes as necessary.

Cons:

- Development team members must have the capacity to be involved, responsible, and innovative.
- People may need to let go of ego to work well as members of a team.

Pros Processes and Tools Have High Value

- Processes are clear and can be easy to follow.
- Written records of communication exist

Cons:

- People may over-rely on processes instead of finding the best ways to create good products.
- One process doesn't fit all teams — different people have different work styles.
- One process doesn't fit all projects.
- Communication can be ambiguous and time-consuming.

2. Working software over comprehensive documentation

- This value reminds us that until the software is delivered to production, then it's carry on value to the customer.
- This has a few anticipated problems though, as mentioned below:
  - It is almost impossible for a person to initially specify exactly how a screen should look like or be perceived by an end user.
  - In reality, end users can change their minds frequently, so keeping up-to-date documentation is a challenging and costly affair.
  - If more some reason the project is terminated in the middle of the project, what will be left behind is a pile of documents that yields no value to the end user

Identifying Useful Documentation

Document	Does the Document Support Product Development?	Is the Document Barely Sufficient or Gold-Plated?
Project schedule created with expensive project management software, complete with Gantt Chart.	No. Start-to-finish schedules with detailed tasks and dates tend to provide more than what is necessary for product development. Also, many of these details change before you develop future features.	Gold-plated Although project managers might spend a lot of time creating and updating project schedules, project team members tend to want to know only key deliverable dates. Management often wants to know only whether the project is on time, ahead of schedule, or behind.
Requirements documentation.	Yes All projects have requirements — details about product features and needs. Development teams need to know those needs to create a product.	Possibly gold-plated; should be barely sufficient. Requirements documents can easily grow to include unnecessary details. Agile approaches provide simple ways to describe product requirements.

Identifying Useful Documentation (2)

Document	Does the Document Support Product Development?	Is the Document Barely Sufficient or Gold-Plated?
Weekly status report.	No. Documenting how you created a product can make future changes easier.	Gold-plated Knowing project status is helpful, but traditional status reports contain outdated information and are much more burdensome than necessary.
Product technical specifications.	Yes All projects have requirements — details about product features and needs. Development teams need to know those needs to create a product.	Possibly gold-plated; should be barely sufficient. Agile documentation includes just what it needs — development teams often don't have time for extra flourishes and are keen to minimize documentation.
On agile projects: - Documents are useful only if they support product development and are barely sufficient to serve for particular purpose (delivery, deployment, ...) - administrative paperwork relating to time, cost control, scope control, or reporting.		

3. Customer Collaboration over Contract Negotiation

- This value focuses on building a relationship based on trust that spans across organization boundaries between the customer and the vendor or service provider of the software
- The contracts are rigid, in the sense that both sides are coerced (or legally bound) to obey the elements of scope, time and cost
- Approvals of such change controls take a long time and at times, could make the progress frustratingly slow.
- Rather than negotiating with the customer on the scope and enforcing rigorous change management techniques, it is in the interest of both the customer and the vendor to collaborate, be flexible and drive toward a common goal.
- “Good collaboration can sometimes make a contract unnecessary” [Alistair Cockburn]

Hi.. I realized that we have to allow entry of negative interest rates too from the front end. Could you handle that ?

But you told us that interest rates can only be positive values ?

Yes, but that was like 6 months ago. The economic scenario has changed we didn't anticipate that earlier

Oops .. But that's too late.. We have already frozen the requirements document and started with implementation.

But I must have the change. Without that our software cannot be used by operations. What is the best you can do ?

I suspect there will be rework. So we must raise a change control record. We need 2 weeks to do a formal impact assessment.

Oh! That's a long time... Wondering how much it will cost ??

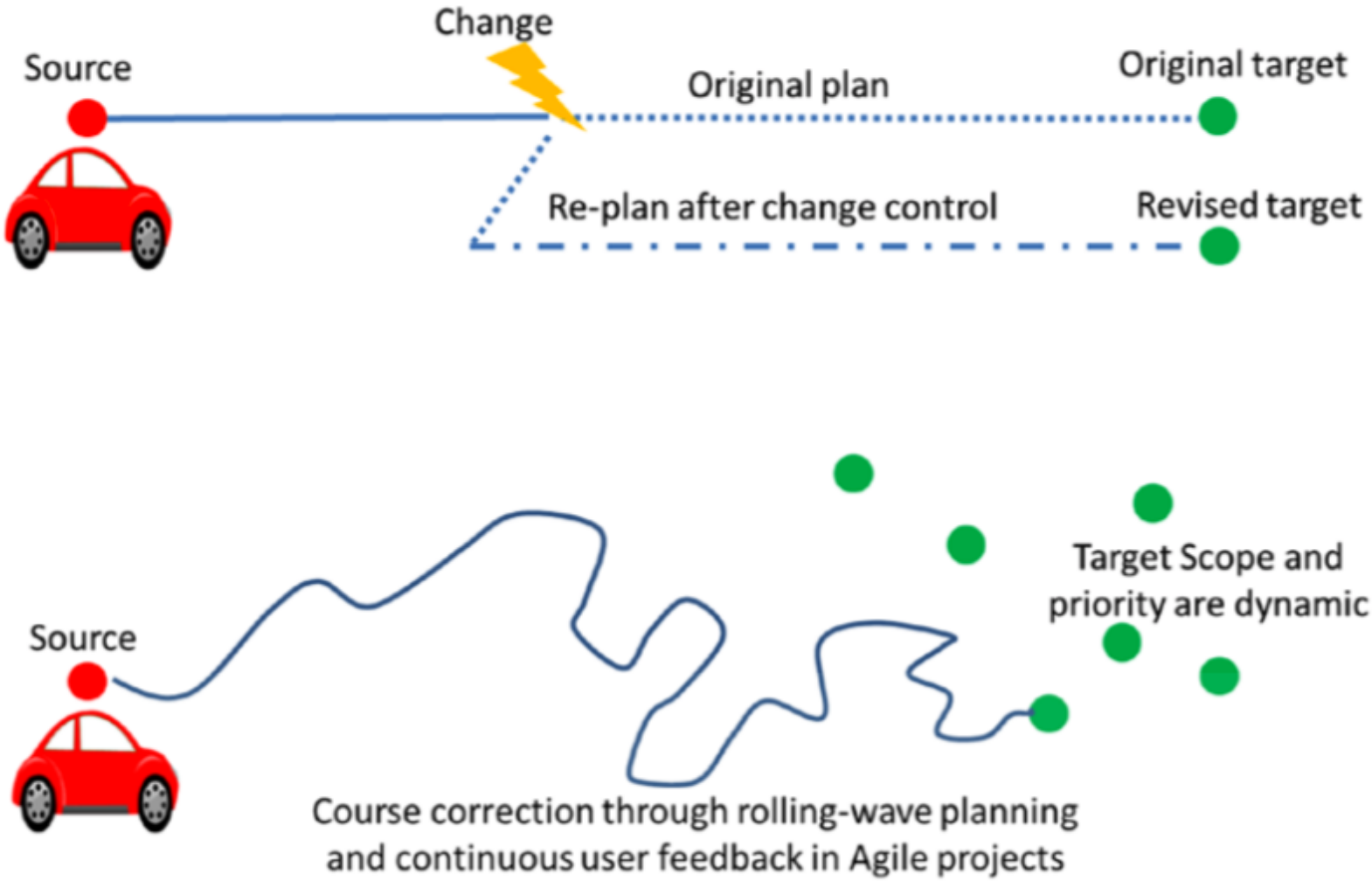
We will get back to you.

Hmm ... Its going to take long to approve the change control and then get this implemented

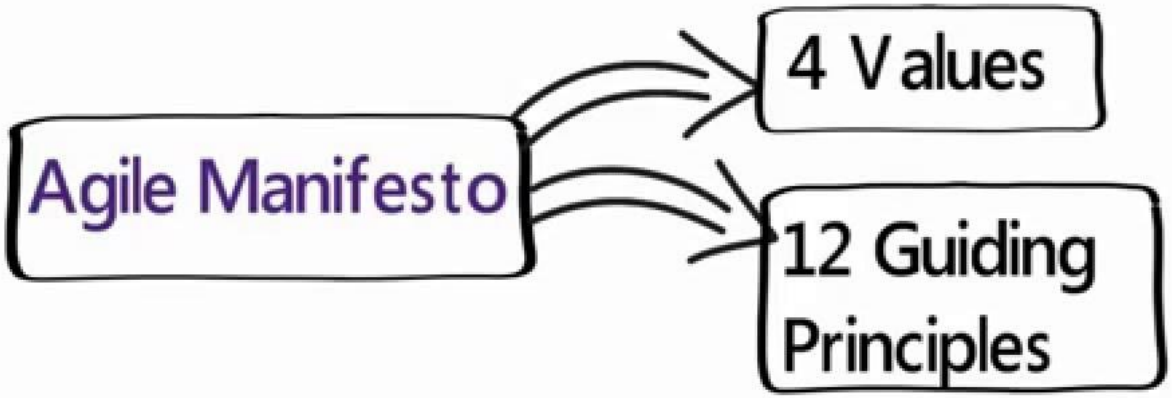
Yes, since we got to update the specifications and design documents and reissue for sign-off. I will try to get this expedited on a best-effort basis

4. Responding to Change over Following a Plan

- The value embodies the fact that Agile acknowledges and embraces change and not treat that as an exception.
- Traditional projects invest a substantial effort in developing a detailed project plan, consisting of detailed schedules, allocation and critical paths. And in the event of a change, re-planning and re-baselining needs to be done to make sure that the plan is up to date and reflects the changes.
- In the case of Agile projects, upfront detailed planning is considered counterproductive and inefficient because of the uncertainties involved. A fair amount of planning happens just in time or at the last responsible moment
- This means that the work items for the upcoming iteration is well detailed out and the others are left coarse-grained


Change
Source
Original plan
Original target
Re-plan after change control
Revised target
Target Scope and
priority are dynamic
Source
Course correction through rolling-wave planning
and continuous user feedback in Agile projects

Agile Principles



All Agile Methodologies MUST be based on these 12 Guiding Principles

The **12** Agile Principles

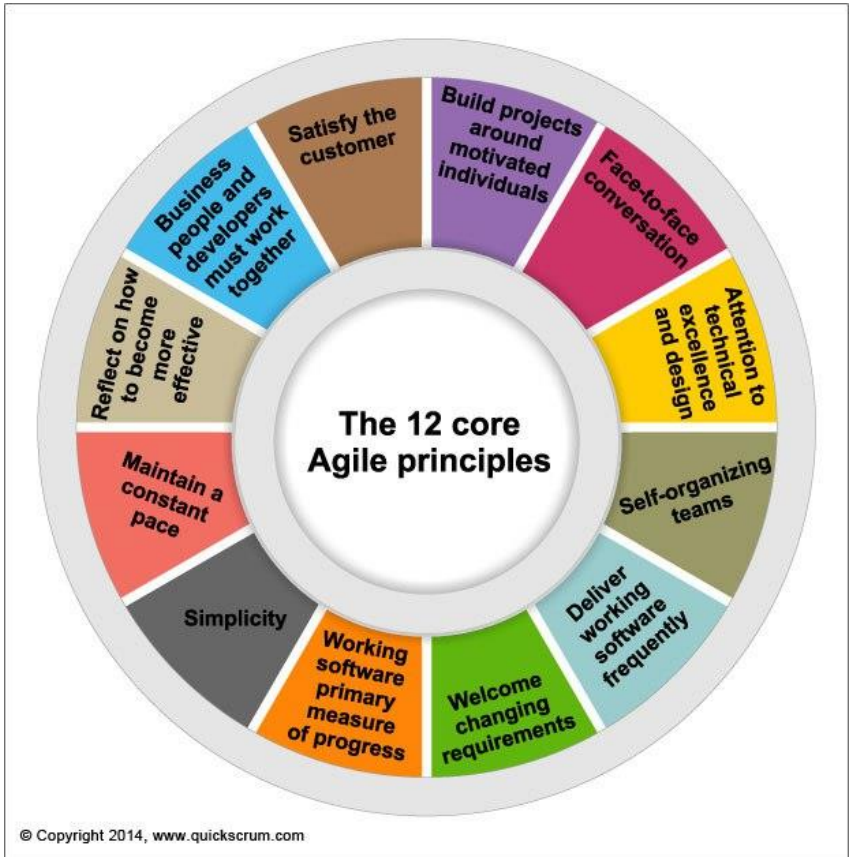
1. Our highest priority is to ***satisfy the customer*** through early and continuous ***delivery of valuable software***.
2. ***Welcome changing requirements***, even late in development. Agile processes harness change for the customer's competitive advantage.
3. ***Deliver working software frequently***, 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***.

The **12** Agile Principles (2)

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.
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***.

The **12** Agile Principles Explained

- These agile principles provide practical guidance for development teams.
- Another way of organizing the 12 principles is to consider them in the following four distinct groups:
  - Customer satisfaction
  - Quality
  - Communicate & Teamwork
  - Project management



The Agile principles of customer satisfaction

- Agile approaches focus on customer satisfaction, which makes sense. After all, the customer is the reason for developing the product in the first place.
  - principles 1, 2, 3, and 4 stand out for us
- How do you enact these principles?
  - Agile project teams include a product owner, a person who is responsible for ensuring translation of what the customer wants into product requirements
  - The product owner prioritizing product features (called Backlog) to the development team. The development team delivers the most valuable features on the list in short cycles of development, known as iterations or sprints.
  - The product owner has deep and ongoing involvement throughout each day to clarify priorities and requirements, make decisions, provide feedback, and quickly answer the many questions that pop up during a project.
  - Frequent delivery of working functionality allows the product owner and the customer to have a full sense of how the product is developing.

The Agile principles of customer satisfaction (2)

- As the development team continues to deliver complete, working, potentially shippable functionality every four to eight weeks or less, the value of the total product grows incrementally, as do its functional capabilities.
- The customer accumulates value for his or her investment regularly by receiving new, ready-to-use functionality throughout the project, rather than waiting until the end of what might be a long project for the first, and maybe only, delivery of releasable product features.

Customer Dissatisfaction and How Agile Might Help

Examples of Customer Dissatisfaction with Projects	How Agile Approaches Can Increase Customer Satisfaction
The product requirements were misunderstood by the development team.	Product owners work closely with the customer to define and refine product requirements and provide clarity to the development team. Agile project teams demonstrate and deliver working functionality at regular intervals. If a product doesn't work the way the customer thinks it should work, the customer is able to provide feedback at the end of the sprint, not before it's too late at the end of the project.
The product wasn't delivered when the customer needed it.	Working in sprints allows agile project teams to deliver high-priority functionality early and often.
The customer can't request changes without additional cost and time.	Agile processes are built for change. Development teams can accommodate new requirements, requirement updates, and shifting priorities with each sprint, offsetting the cost of these changes by removing the lowest-priority requirements — functionality that likely will never or rarely get used.

The Agile principles of quality

- An agile project team commits to producing quality in every product it creates — from development through documentation to integration and test results — every day
- Each project team member contributes his or her best work all the time.
- principles 1, 3, 4, 6, 7, 8, 9, and 12 stand out for us
- Agile approaches provide the following strategies for quality management:
  - Defining what done means at the beginning of the project and then using that definition as a benchmark for quality
  - Testing aggressively and daily through automated means
  - Building only the functionality that is needed when it's needed
  - Reviewing the software code and streamlining (refactoring)
  - Showcasing to stakeholders and customers only the functionality that has been accepted by the product owner
  - Having multiple feedback points throughout the day, iteration, and project

The Agile principles of communication and teamwork

- Teamwork is critical to agile projects.
- Creating good products requires cooperation among all the members of the project team, including customers and stakeholders.
- They know that meeting with users more often keeps them in sync, and actually prevents those changes.
- Agile approaches support team-building and teamwork, and they emphasize trust in self-managing development teams
- Principles 4, 5, 6, 8, 11, and 12 stand out for us as supporting team empowerment, efficiency, and excellence

The Agile principles of communication and teamwork (2)

- The following strategies promote effective teamwork:
  - Place the development team in the same location — this is called collocation
  - Put together a physical environment that's conducive for collaboration: a team room with whiteboards, colored pens, and other tactile tools for developing and conveying ideas to ensure shared understanding.
  - Create an environment where project team members are encouraged to speak their minds. When everyone is trusted and given a good working environment, they flourish.
  - Meet face-to-face whenever possible. Don't send an email if a conversation can handle the issue.
  - Get clarifications throughout the day as they're needed.
  - Encourage the development team to solve problems rather than having managers solve problems for the development team.

The Agile principles of project management

- An agile approach focuses on planning and executing the work to produce the best product that can be released.
- The approach is supported by communicating openly, avoiding distractions and wasteful activities, and ensuring that the progress of the project is clear to everyone.
- Principles 2, 8, and 10 stand out for us:
- Project management is facilitated by the following agile approaches:
  - Supporting the development team
  - Producing barely sufficient documents
  - Streamlining status reporting so that information is pushed out by the development team in seconds rather than pulled out by a project manager over a longer period of time

The Agile principles of project management (2)

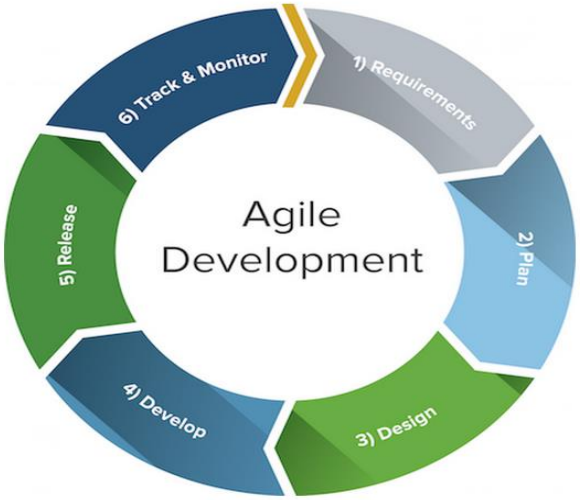
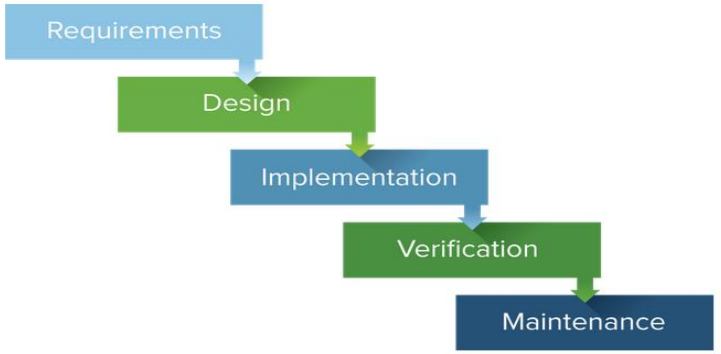
- Minimizing non-development tasks
- Setting expectations that change is normal and beneficial, not something to be feared or evaded
- Adopting a just-in-time requirements to minimize change disruption and wasted effort
- Collaborating with the development team to create realistic schedules, targets, and goals
- Protecting the development team from organizational disruptions that could undermine project goals by introducing work not relevant to the project objectives
- Understanding that an appropriate balance between work and life is a component of efficient development

Agile Project Management

- Agile project management is a style of project management that focuses on early delivery of business value, continuous improvement of the project's product and processes, scope flexibility, team input, and delivering well-tested products that reflect customer needs.

- Unlike more traditional project management techniques, which focus on achieving identifiable milestones using complex scheduling tools like Gantt charts, Agile management is an iterative, incremental process.

Agile Project Management

Agile	Waterfall
	
• Continuous cycles • Small, high-functioning, collaborative teams • Multiple methodologies • Flexible/continuous evolution • Customer involvement	• Sequential/linear stages • Upfront planning and in-depth documentation • Contract negotiation • Best for simple, unchanging projects • Close project manager involvement

Contrasting Historical Project Management with Agile Project Management

Traditional Project Management Tasks	Agile Approach to the Project Management Task
Create a fully detailed project requirement document at the beginning of the project. Try to control requirement changes throughout the project.	Create a product backlog — a simple list of requirements by priority. Quickly update the product backlog as requirements and priorities change throughout the project.
Conduct weekly status meetings with all project stakeholders and developers. Send out detailed meeting notes and status reports after each meeting.	The development team meets quickly, for no longer than 15 minutes, at the start of each day to coordinate and synchronize that day's work and any roadblocks. They can update the centrally visible burndown chart in under a minute at the end of each day.
Create a detailed project schedule with all tasks at the beginning of the project. Try to keep the project tasks on schedule. Update the schedule on a regular basis.	Work within sprints and identify only specific tasks for the active sprint.
Assign tasks to the development team.	Support the development team by helping remove impediments and distractions. On agile projects, development teams define and pull (as opposed to push) their own tasks.

Agile Principle over Practices

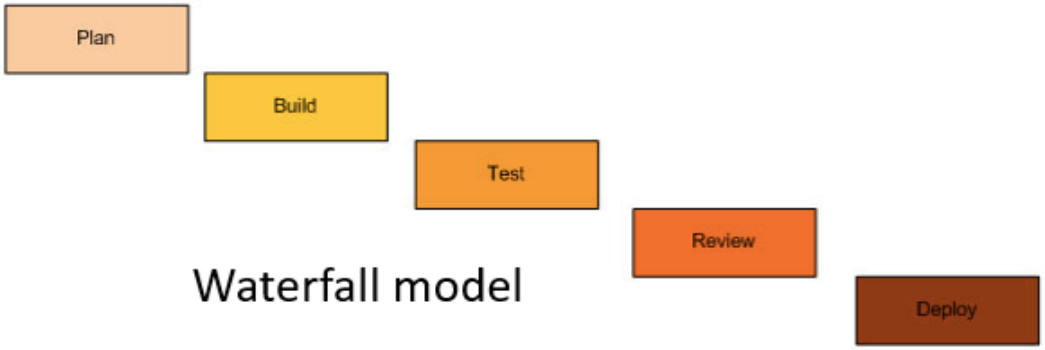
Some of the principles or culture changes that are the harder parts of Agile

- Communication
  - I often see people using the Scrum Master to deliver their messages to others.
  - Help your teams get into the habit of communicating face-to-face as their first option instead of as a last resort. Wasted time is not Agile.
- Collaboration:
  - Agile is all about people working together, talking together and coming up with great ideas together
  - Great Agile teams are not managed – they're encouraged to collaborate and figure things out themselves.
  - Often, the lesson is learned better by experience (good or bad) than by being told what to do.
  - Product owners to get involved with the project team regularly throughout the sprint

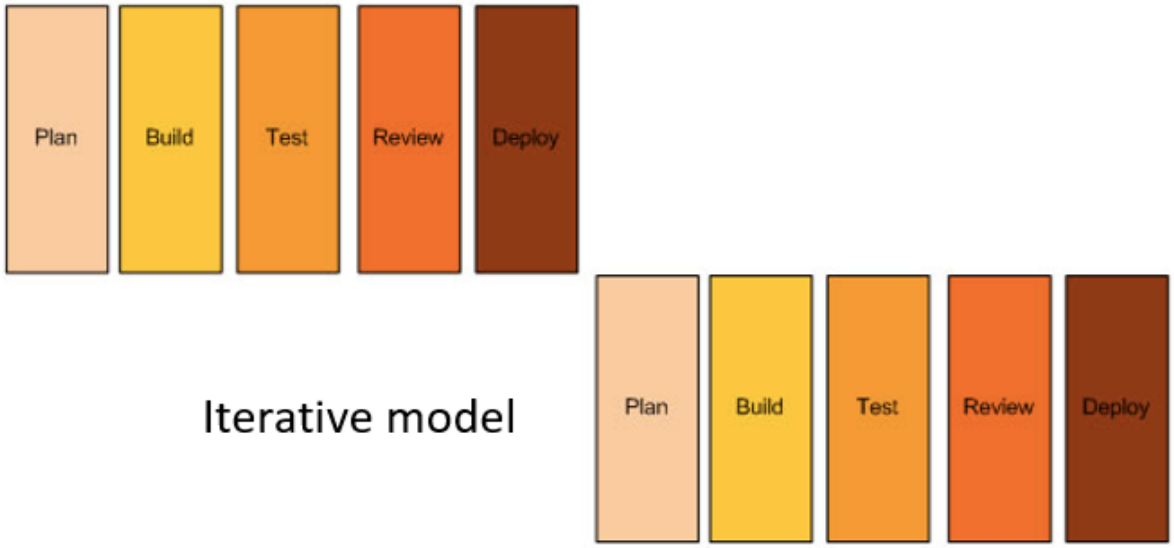
Agile Principle over Practices (2)

- Transparency
  - Transparency is just being honest – telling it like it is and sometimes being vulnerable
  - We can’t fix things if problems are swept under the rug.
  - encourage teams to make work more visible by putting large paper Scrum boards and burn-down charts on a wall and showcasing this information to everyone around them
  - Agile team members are reluctant to raise obstacles during stand-up meetings. Scrum Master is to remove obstacles so the team can focus on delivering software.
  - Reward those who speak up and raise obstacles in the first few sprints, especially the quieter people, by recognizing them in front of the team.
- Retrospection:
  - some Agile teams skip sprint retrospectives because they think they don’t have time or they don’t see the value of doing them
- Simplicity:
  - This can be one of the harder parts of Agile.
  - Artifacts should be minimized to only what’s really needed to get the job done.

From Traditional to Agile

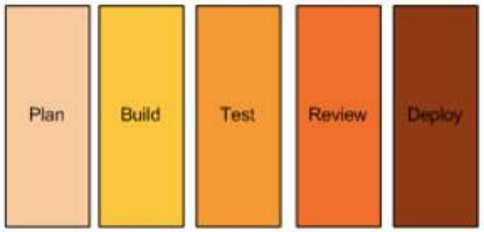


Waterfall model

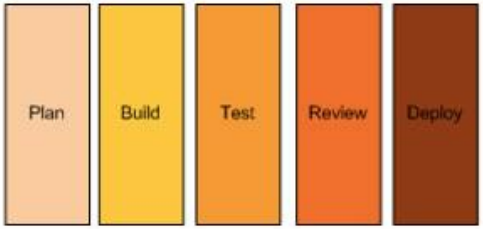


Iterative model

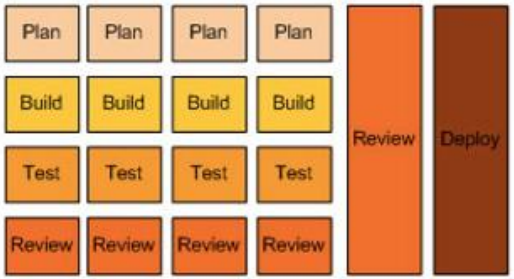
From Traditional to Agile



Iterative model



Agile methodology (Scrum)



Agile Vs. Traditional

Agile:

- Building off close collaboration with one customer.
- Getting a product with significant value out faster.
- Building with a small team working closely together.
- Iterative delivery by a single organization who come to have the knowledge in their heads.
- What's important can only be discovered incrementally.
- What's a "perfect example" of an Agile-friendly project?

Do it when the cost of rework is low. Dealing with things "*ad hoc*" mostly works. We call it "refactoring."

Non Agile:

- Building a general product to address a wide customer base.
- Building off standards and consistent design principles.
- Managing a large project with lots of interdependent pieces.
- Product releases over a long period of time, by rotating staff who rely on documentation.
- Goals and rules well known at the start.
- What's a "perfect example" of an Old-school-friendly project?

Do it when the cost of rework is high. Big surprises would be awful. We'd be starting over.

Agile Vs. Traditional

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual—favors specialization	Self-organizing teams—encourages role interchangeability
Communication	Formal	Informal
Customer's Role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

Next Lecture

- Agile Software Process

1. Agile Process & CMMI: رغد + محمد
2. Agile Software Project Management:حنان+سيدره
3. Agile Software Testing Tools:أحمد شيخة+فرح فارس
4. SCRUM successful case studies:محمد نور + ليلى+طيف
5. XP successful case studies:عبد الله + عامر خورشيد
6. Agile Software development and knowledge management: علي الأسود + وافي
7. Best Software environments for Agile dev.:كام+عامر
8. Agile Methodologies comparison:غالية+حنين
9. SCRUM & incremental comparision:رنيم+عمر