الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

كلية الهندسة
FACULTY OF ENGINEERING

Agile Software Development

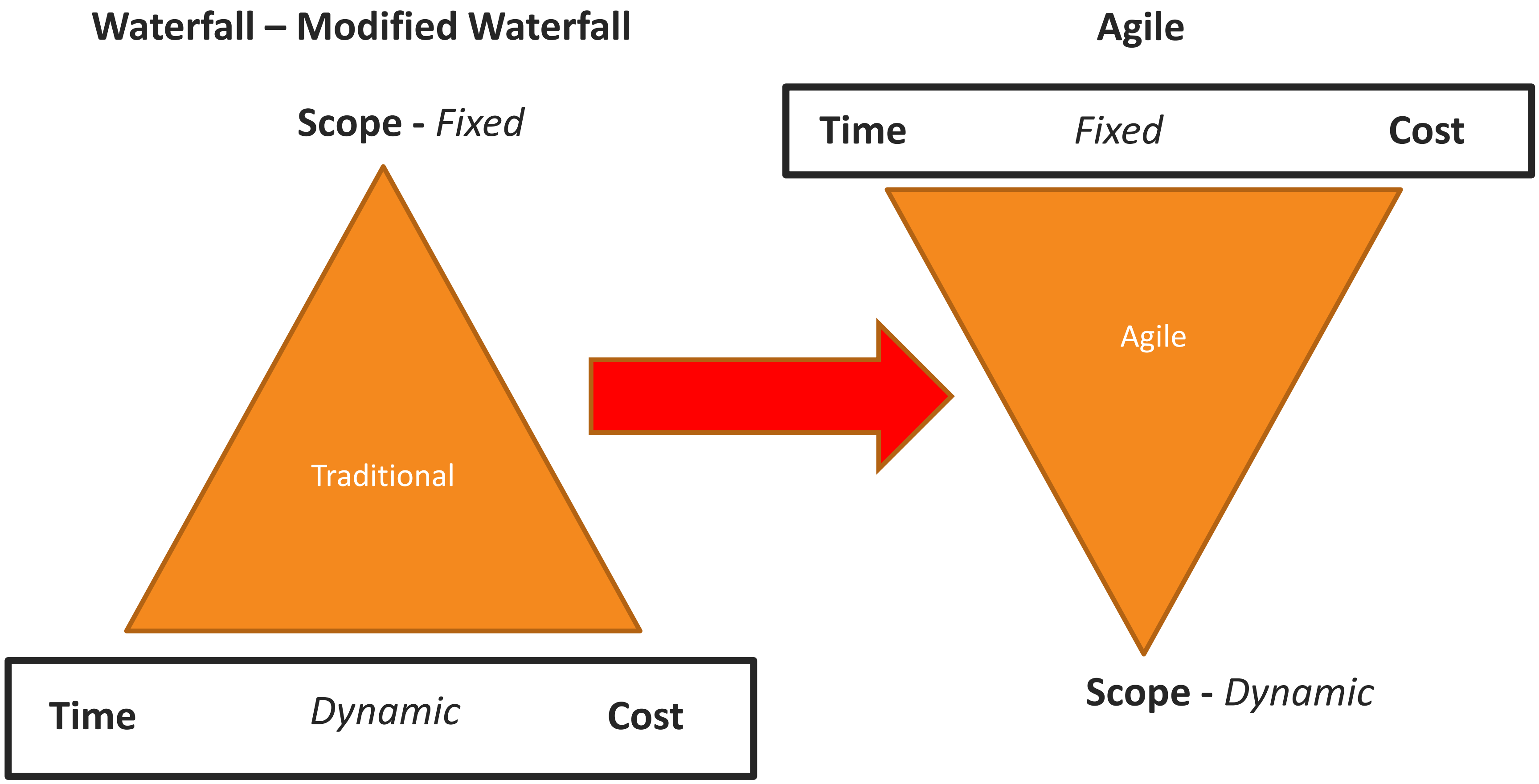
Instructor: Dr. Mouhib Alnoukari

4 Agile Software Process

What is an Agile Project?

- Simple answer: A Project that uses Agile Principles and Practices
- But what are those characteristics?
  - Being able to measure ROI early and often in the project
  - Transparent to the organization – high visibility of project progress and issues
  - Continuous involvement of stakeholders (customers) in throughout the project
  - The business owner is empowered to make decisions to meet goals
  - Adaptive – able to change to meet needs and requirements as they present themselves
  - ***Always looking for ways to reduce waste and target the minimum set of customer value***
    - ***Reduce waste of both the product and the process***

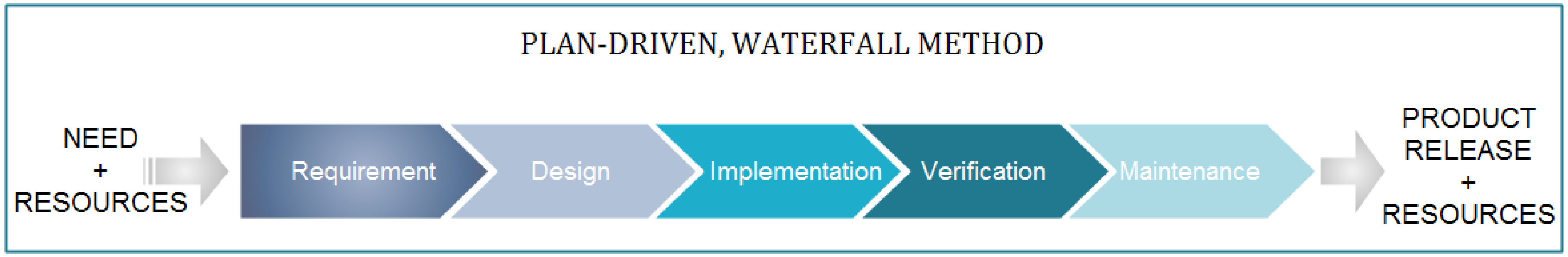
The Iron Triangle – Cost, Time and Scope Relationship

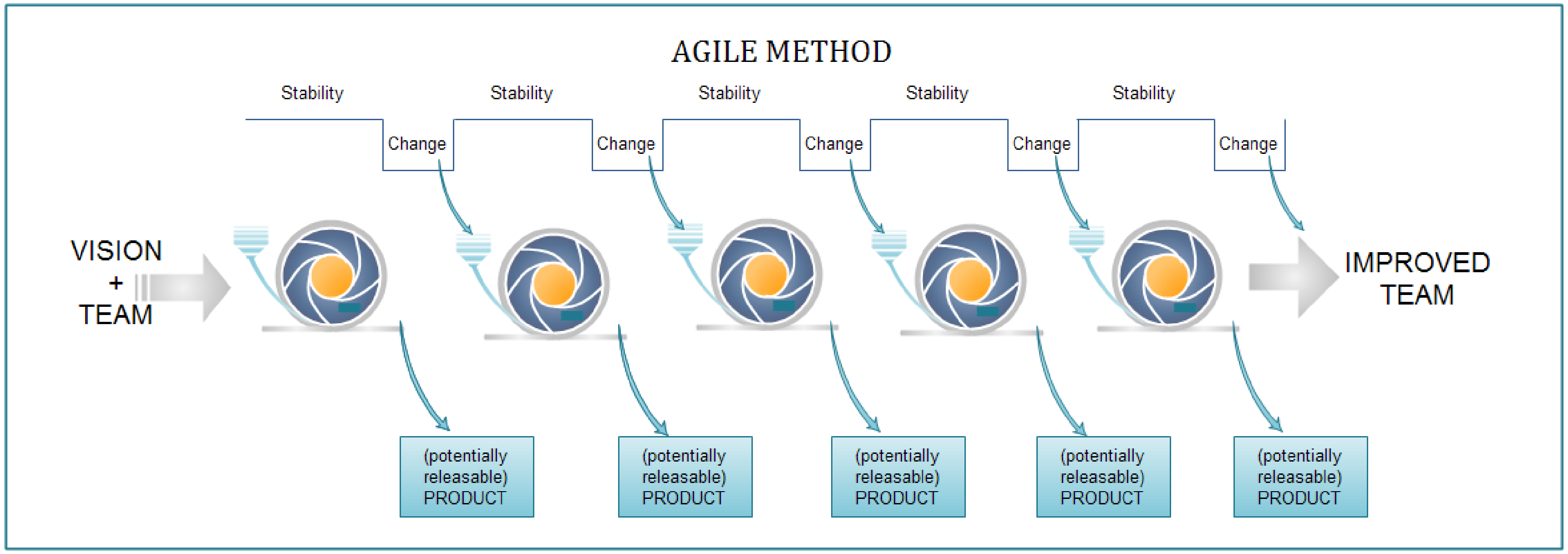


An Agile Process

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

Plan-driven and agile specification




AGILE METHOD
Stability
Change
Stability
Change
Stability
Change
Stability
Change
Stability
Change
VISION + TEAM
IMPROVED TEAM
(potentially releasable) PRODUCT
(potentially releasable) PRODUCT
(potentially releasable) PRODUCT
(potentially releasable) PRODUCT
(potentially releasable) PRODUCT

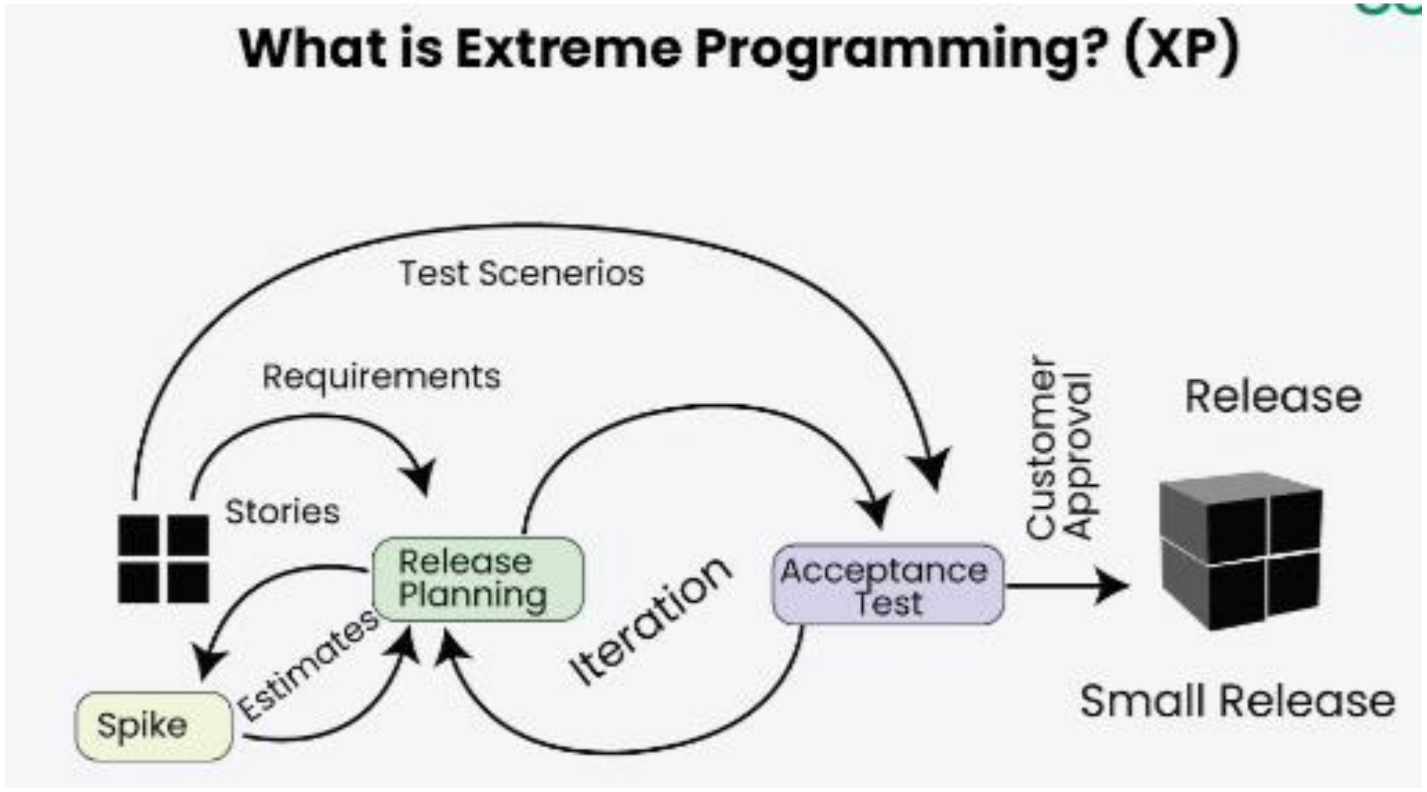
Problems with agile methods

- It can be difficult to keep the interest of customers / users who are involved in the process.
- Team members may be unsuited to the intense involvement that characterizes agile methods.
- Prioritizing changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.
- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.
- Less emphasis on documentation - harder to maintain when you get a new team for maintenance

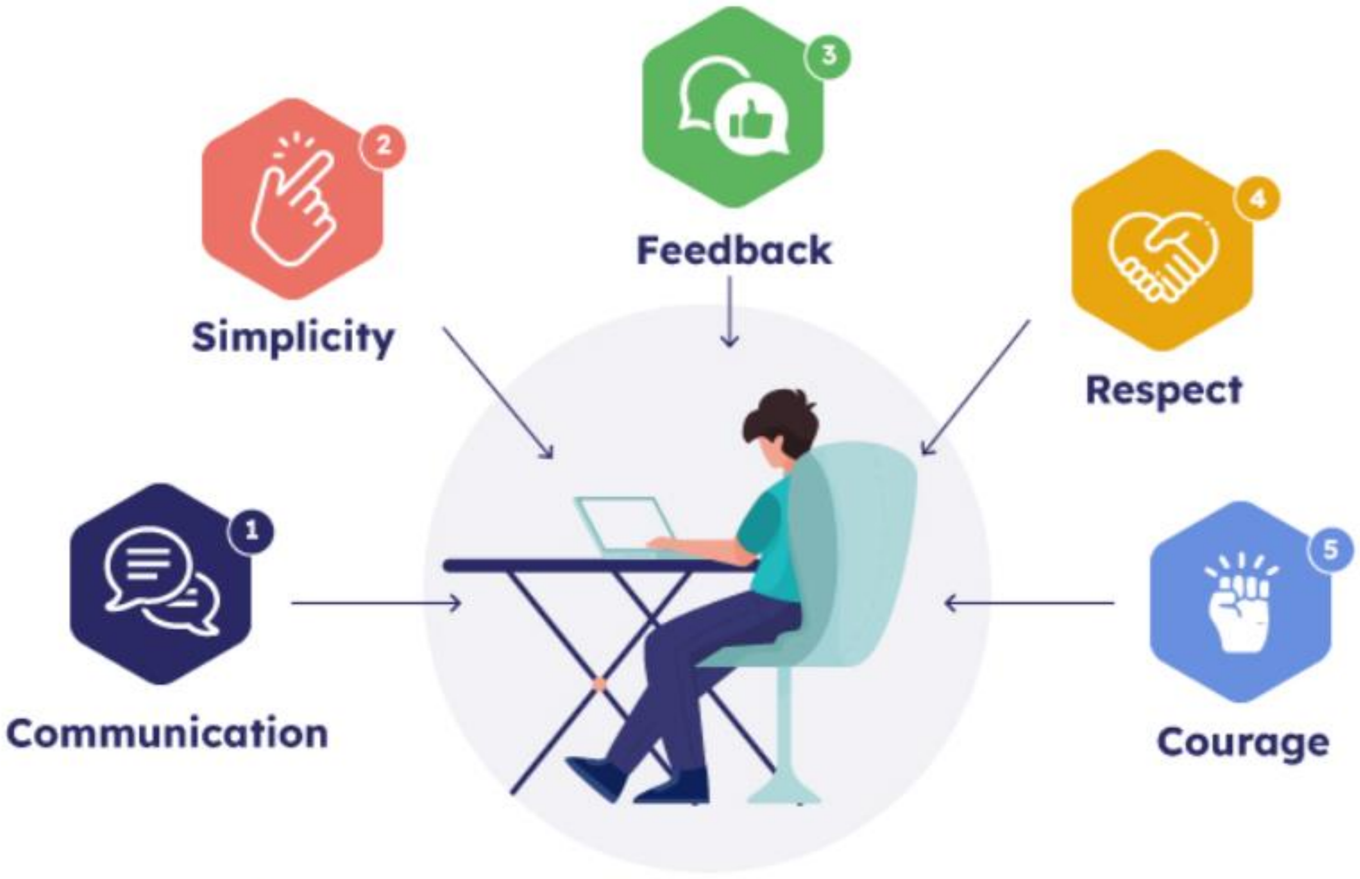
Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck 2004.
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
    - New versions may be built several times per day;
    - Increments are delivered to customers every 2 weeks;
    - All tests must be run for every build and the build is only accepted if tests run successfully.
- Customer involvement means full-time customer engagement with the team. - Specifications through user stories broken into tasks
- People not process : pair programming, collective ownership and a process that avoids long working hours.
- Regular system releases. - release set of user stories
- Maintaining simplicity through constant refactoring of code.

Extreme Programming (XP)



XP Values


3
Feedback
2
Simplicity
4
Respect
1
Communication
5
Courage

XP Values - Communication

- Everyone is a member of the team: project managers, developers, testers. They constantly communicate with each other and with the customer. There’s a daily standup meeting for status updates. The aim is to avoid surprises and identify issues quickly.

- The customer gets frequent updates on progress. This is a consistent practice throughout the entire development cycle. Customers find out from the beginning what’s happening, not just when there’s something ready for them to test.

- Communication may take the form of “pair programming,” one of XP’s most controversial ideas. The idea is that two people work at the same computer, checking each other’s work. Making a success of this requires the right personalities.

XP Values - Simplicity

- Keeping the steps simple is one of XP’s most distinctive features.
- A project starts with the simplest design possible.
- The process starts with a few lines of code that do something.
- Tasks are broken into small pieces, and functionality is added a bit at a time.
- Instead of submitting a requirements document, the customer provides user stories. They don’t have to be full technical descriptions, just to say what needs to get done.
- The developers will work from that to create something. If the customer doesn’t like it, they’ll talk about what needs changing.

XP Values - Feedback

- Communication and feedback are closely related.
- The team is always focused on the customer’s needs. It provides usable code as frequently as possible and responds to customer reactions.
- Instead of finding out months later that it should have done something differently, it gets feedback within days or even the same day.
- Since each piece is kept simple, changing how it works isn’t onerous.

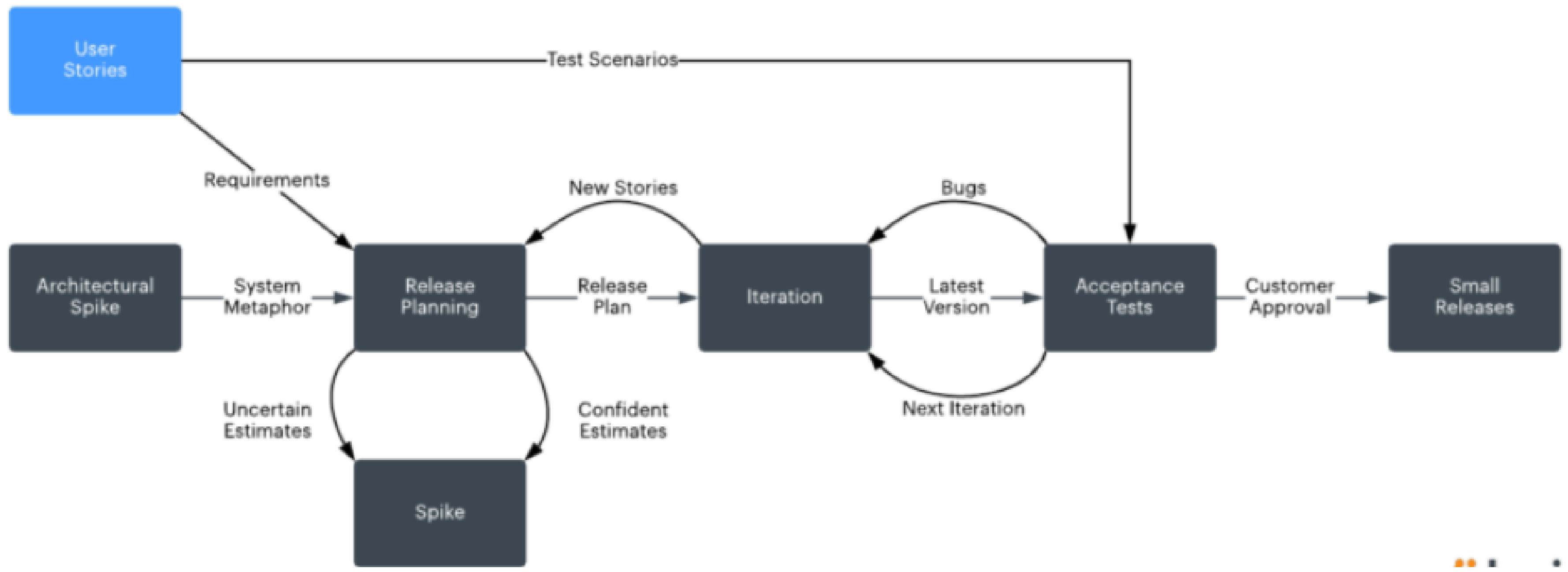
XP Values - Respect

- Everyone on the team is a contributor. Everyone gets listened to. Ideas get a hearing, even if they may seem silly at first. Ideas are evaluated by their worth, not by the seniority of the person presenting them.
- The customer gets respect too. This can be hard when the demands seem arbitrary and excessive, but the tight feedback cycle helps avoid stress.
- Requests for change will come early in the cycle, before the team has invested too much effort and pride in making things work a certain way.

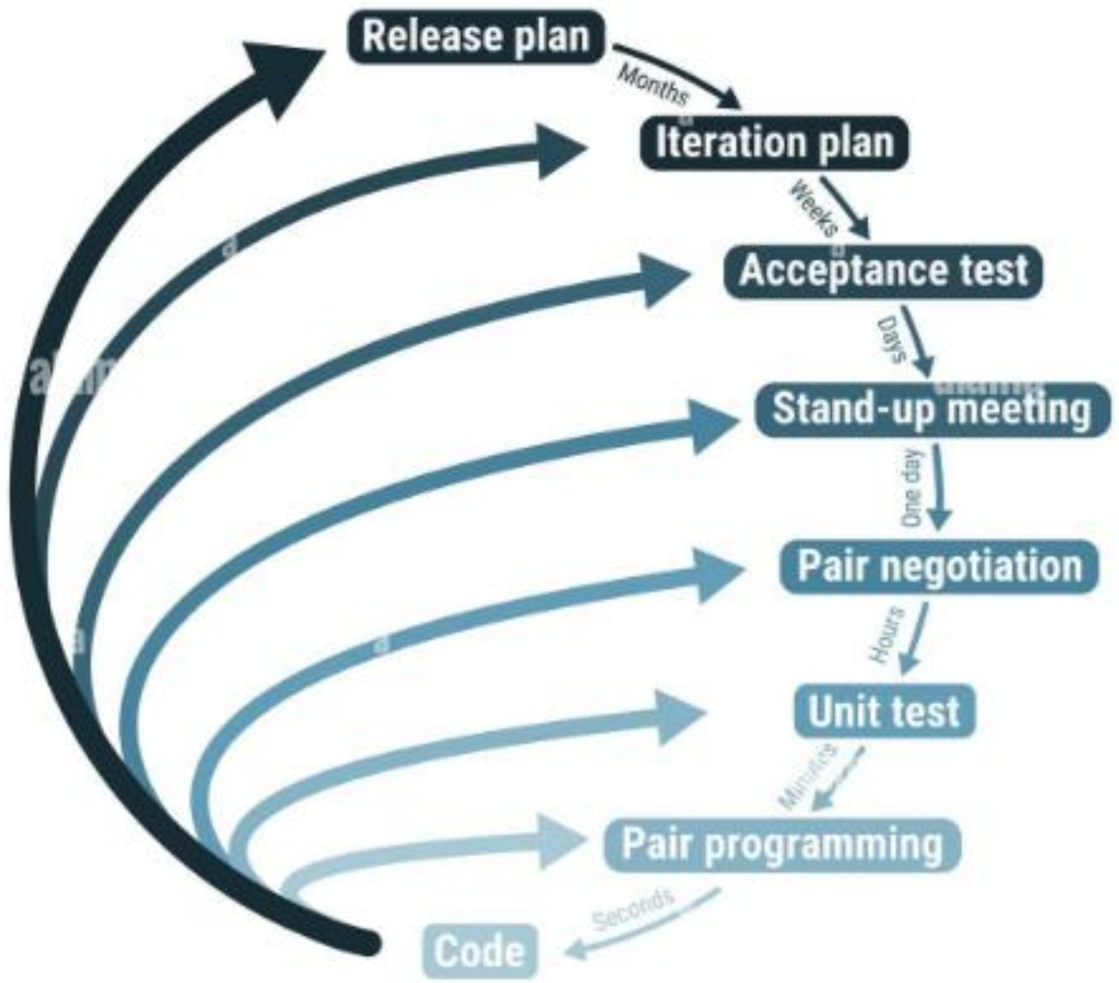
XP Values - Courage

- Respect and courage on a team go hand in hand. Everyone takes responsibility, and people don’t fake progress.
- If there’s a problem, the people responsible own up to it. They don’t get blasted, but they’re expected to do what’s necessary to fix it. This can include asking for help.
- Courage can mean discarding a bad idea or a bad piece of code.
- The aim is to meet the customer’s needs, not to prop up sunk costs.
- Trying out a new approach is sometimes the best way to deal with a persistent problem.

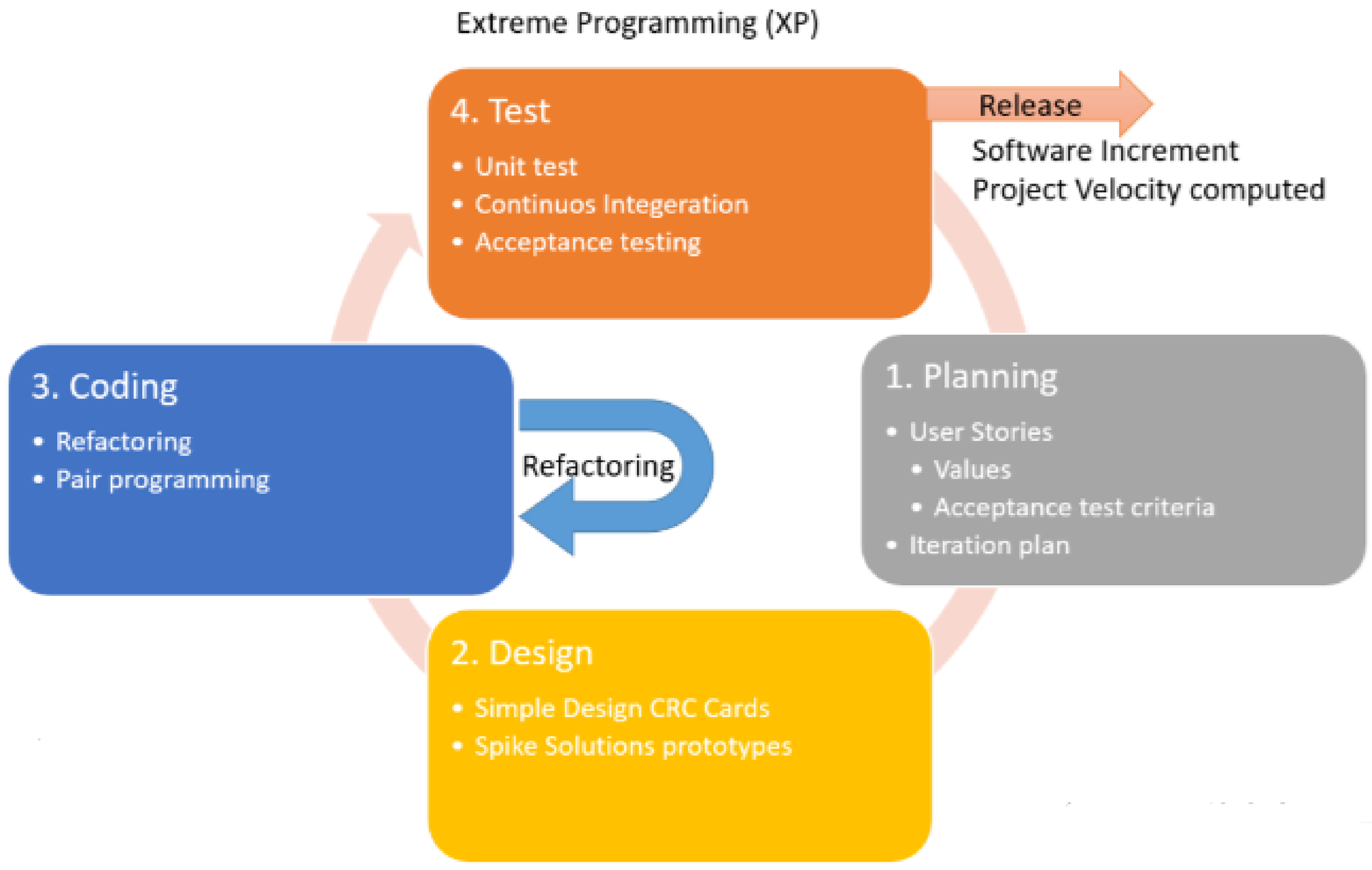
XP Methodology



Planning/feedback Loops



The XP Process



Extreme Programming (XP) – Pair Programming

- ✧ In XP, programmers work in pairs, sitting together to develop code.
- ✧ Common ownership
- ✧ Knowledge spread
- ✧ Informal review
- ✧ Refactoring
- ✧ Similar output to two people coding

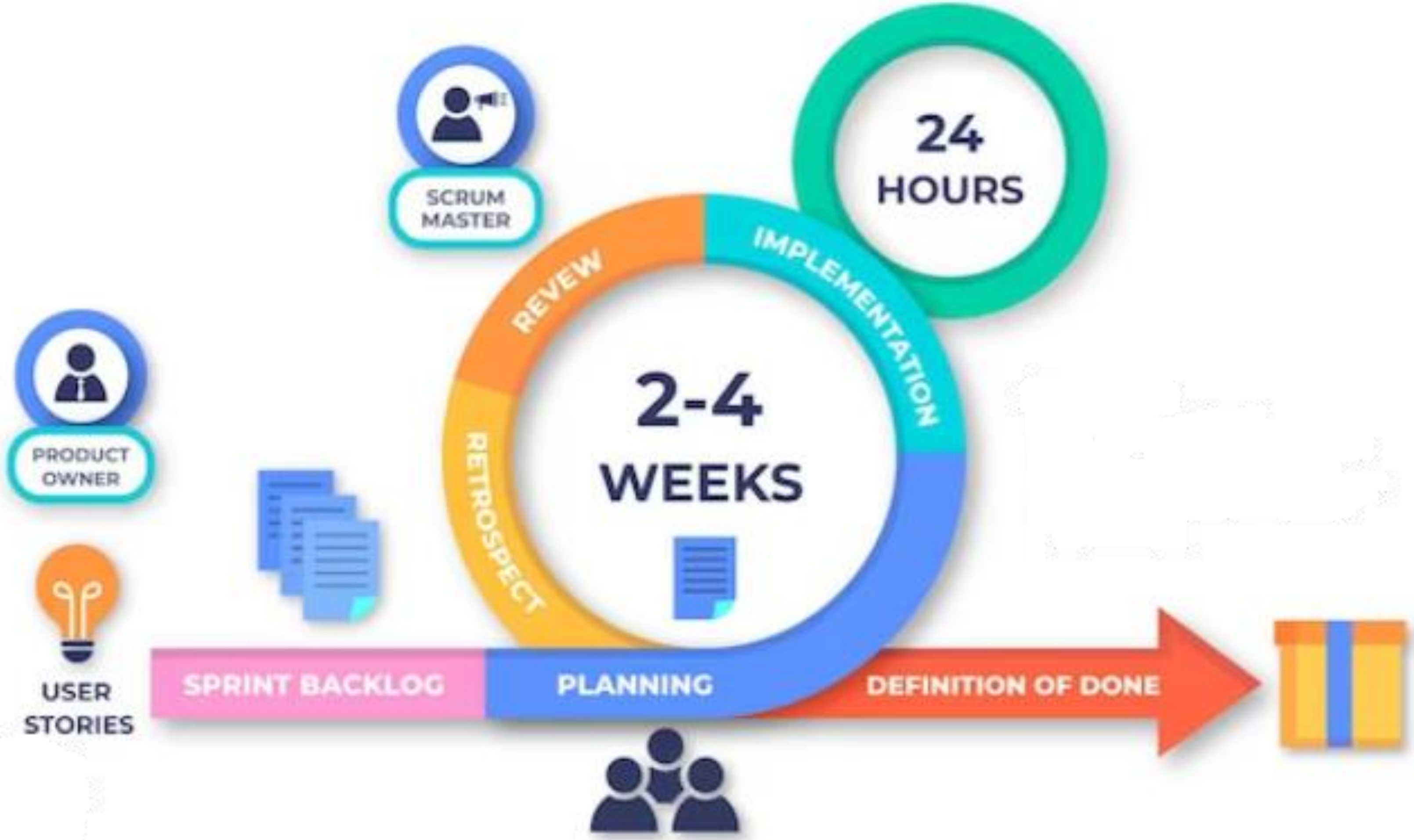
Scrum

- Originally proposed by Schwaber and Beedle
- Scrum—distinguishing features
  - Development work is partitioned into “packets”
  - Testing and documentation are on-going as the product is constructed
  - Work occurs in “sprints” and is derived from a “backlog” of existing requirements
  - Meetings are very short and sometimes conducted without chairs
  - “demos” are delivered to the customer with the time-box allocated

Scrum

- Project Manager's job: - Deliver needed system on time within budget
- The Scrum approach - manage the iterations
- There are three phases in Scrum.
  - Outline planning phase - general picture and architecture
  - Sprint cycles releasing increments of the system.
  - The project closure phase - final delivery, documentation and review of lessons learned.

Scrum – The Process



Scrum - The Sprint cycle

- ✧ Every 2–4 weeks (a fixed length).

1) Project team with customer: Look at product backlog - select stories to implement

2) Implement with all customer communication through scrum master
   - ✧ Scrum master has project manager role during sprint
   - ✧ Daily 15 min meetings
     - ✧ Stand up often
     - ✧ Team presents progress and impediments
     - ✧ Scrum master tasked with removing impediments

3) Review system release with user

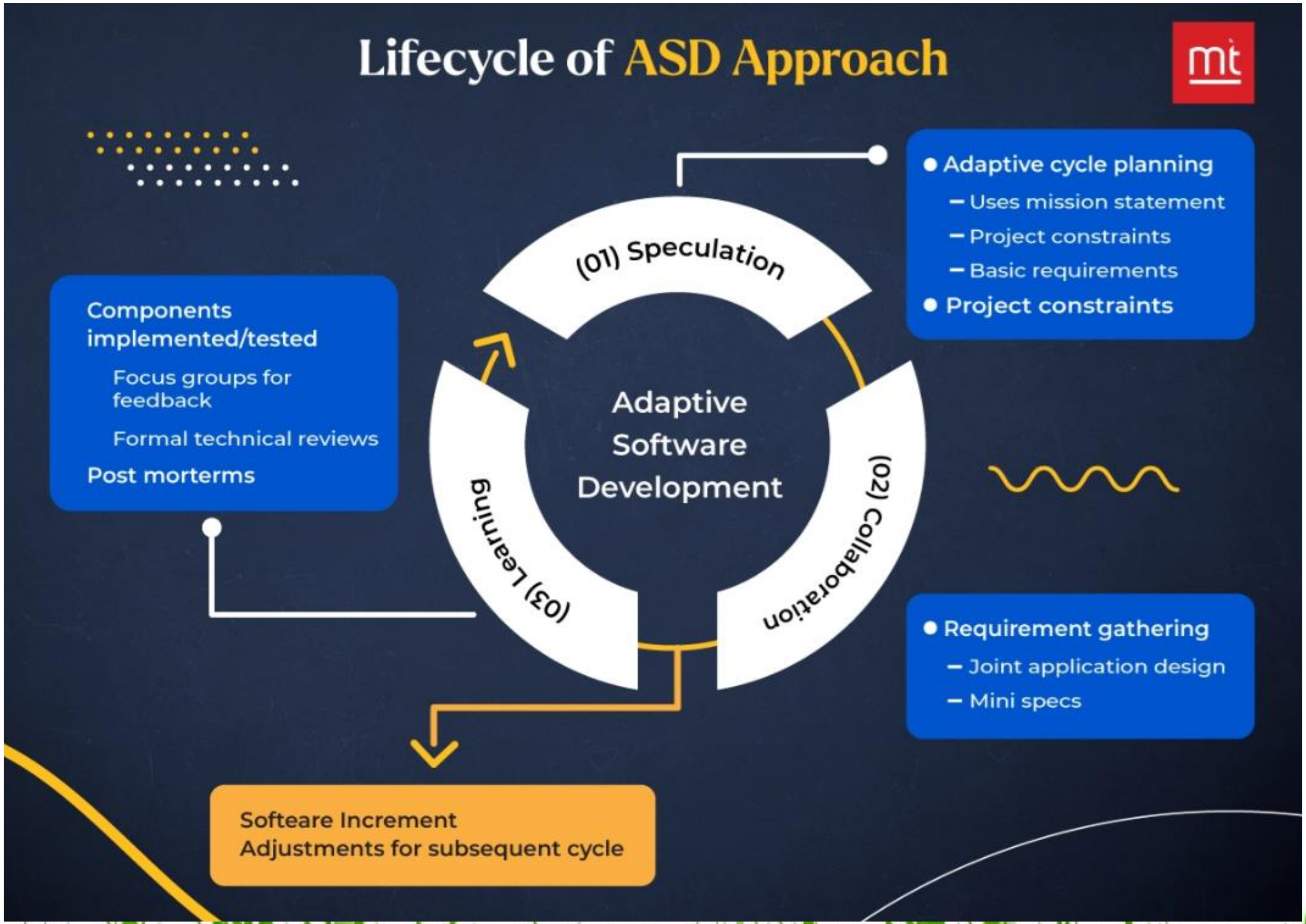
Scrum - Benefits

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Adaptive Software Development

- Originally proposed by Jim Highsmith
- ASD — distinguishing features
  - Mission-driven planning
  - Component-based focus
  - Uses “time-boxing”
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes “learning” throughout the process

Adaptive Software Development



Three Phases of ASD

1. Speculation:

– Project is initiated and adaptive cycle planning is conducted.

– Adaptive cycle planning uses project initiation information- the customer's mission statement, project constraints (e.g. delivery date), and basic requirements to define the set of release cycles (increments) that will be required for the project.

– Based on the information obtained at the completion of the first cycle, the plan is reviewed and adjusted so that planned work better fits the reality.

Three Phases of ASD

2. Collaborations are used to multiply their talent and creative output beyond absolute number . It encompasses communication and teamwork, but it also emphasizes individualism, because individual creativity plays an important role in collaborative thinking.

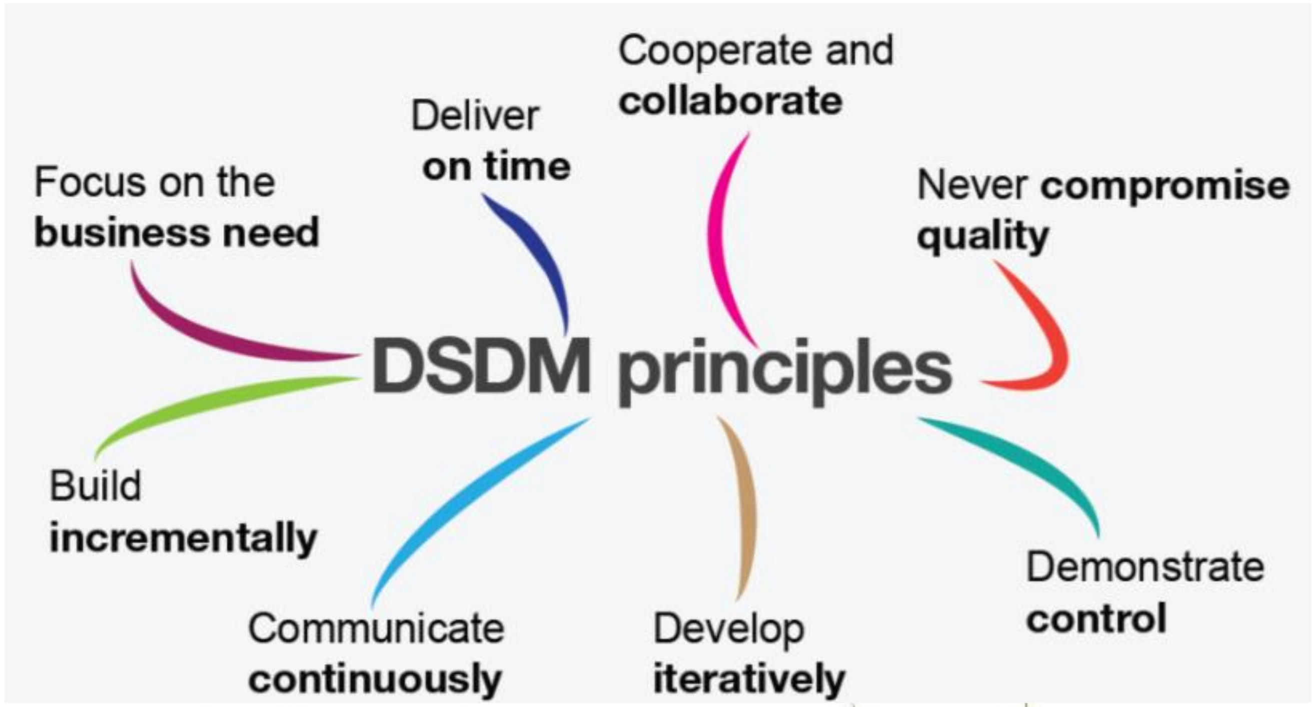
Three Phases of ASD

3. Learning:

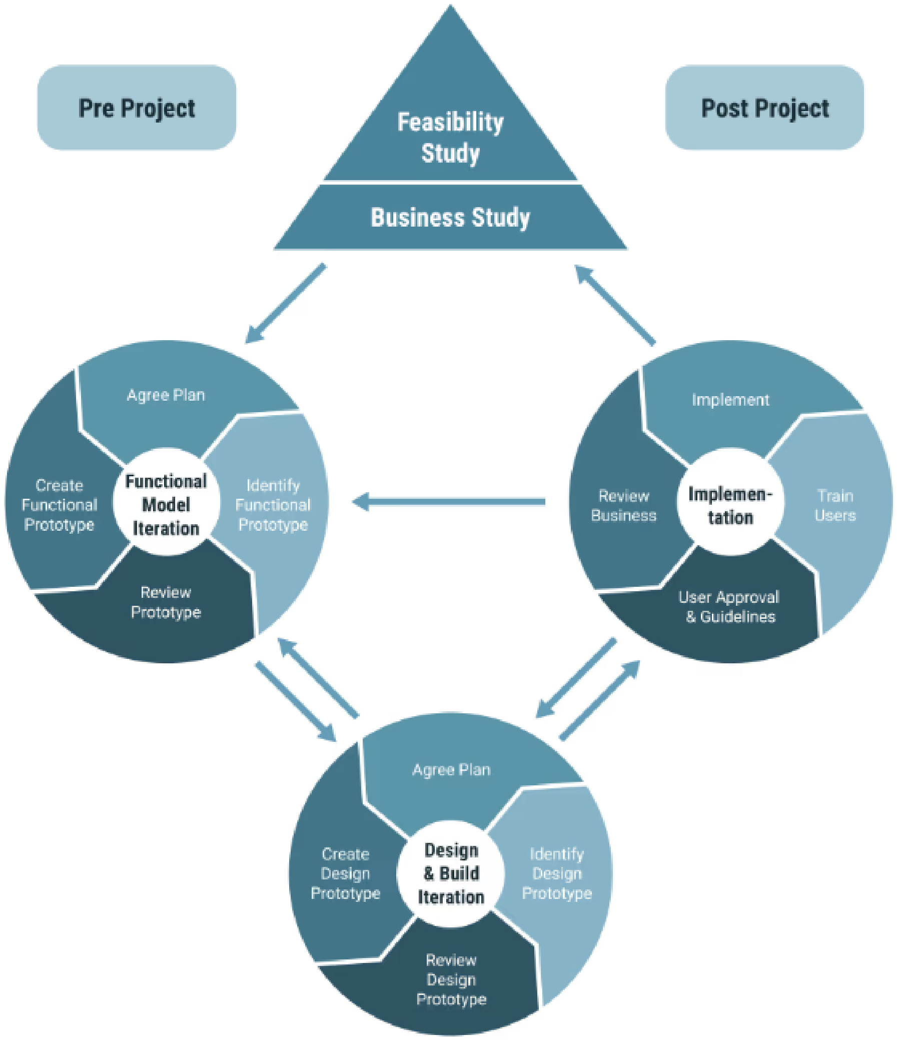
- As members of ASD team begin to develop the components, the emphasis is on “learning”.
- Highsmith argues that software developers often overestimate their own understanding of the technology, the process, and the project and that learning will help them to improve their level of real understanding.
- Three ways:
  - focus groups,
  - technical reviews
  - project postmortems.

Dynamic Systems Development Method

- Promoted by the DSDM Consortium (www.dsdm.org)
- DSDM—distinguishing features
  - Similar in most respects to XP and/or ASD


DSDM principles
Focus on the **business need**
Deliver **on time**
Cooperate and **collaborate**
Never **compromise quality**
Build **incrementally**
Communicate **continuously**
Develop **iteratively**
Demonstrate **control**

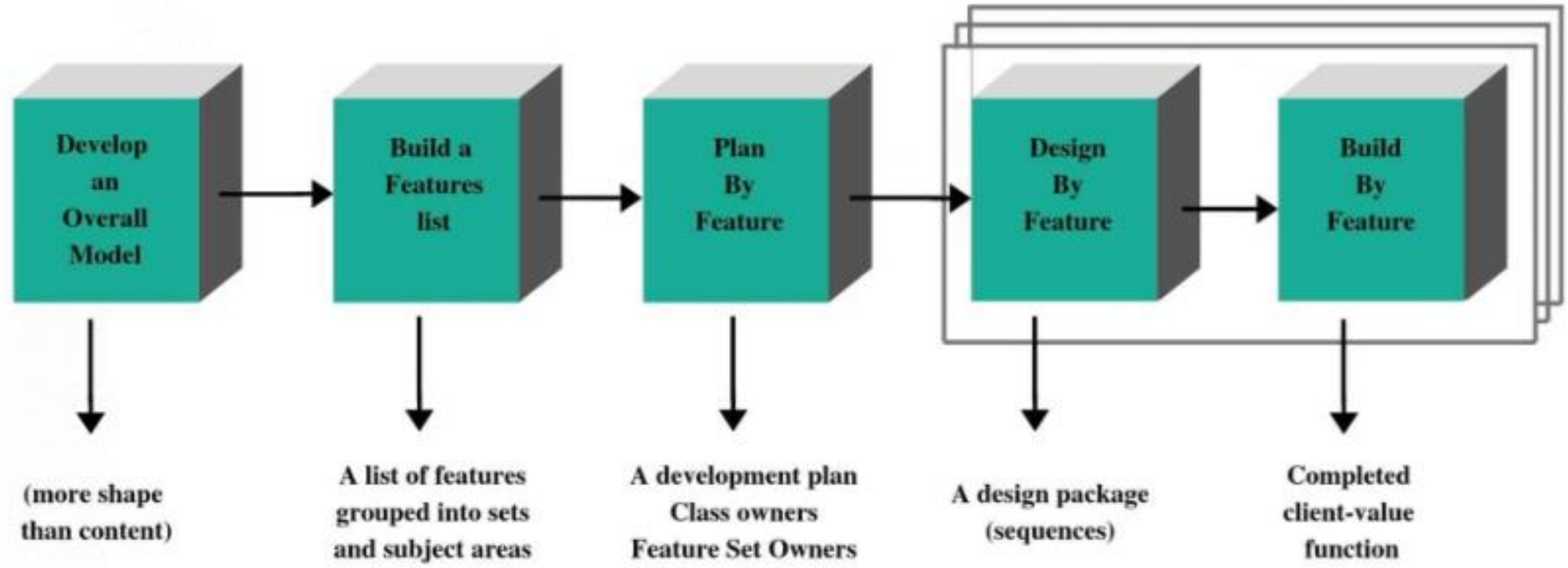
Dynamic Systems Development Method



Feature Driven Development

- Originally proposed by Peter Coad et al
- FDD—distinguishing features
  - Emphasis is on defining "features"
    - a *feature* "is a client-valued function that can be implemented in two weeks or less."
  - Uses a feature template
    - <action> the <result> <by | for | of | to> a(n) <object>
  - A features list is created and "plan by feature" is conducted
  - Design and construction merge in FDD

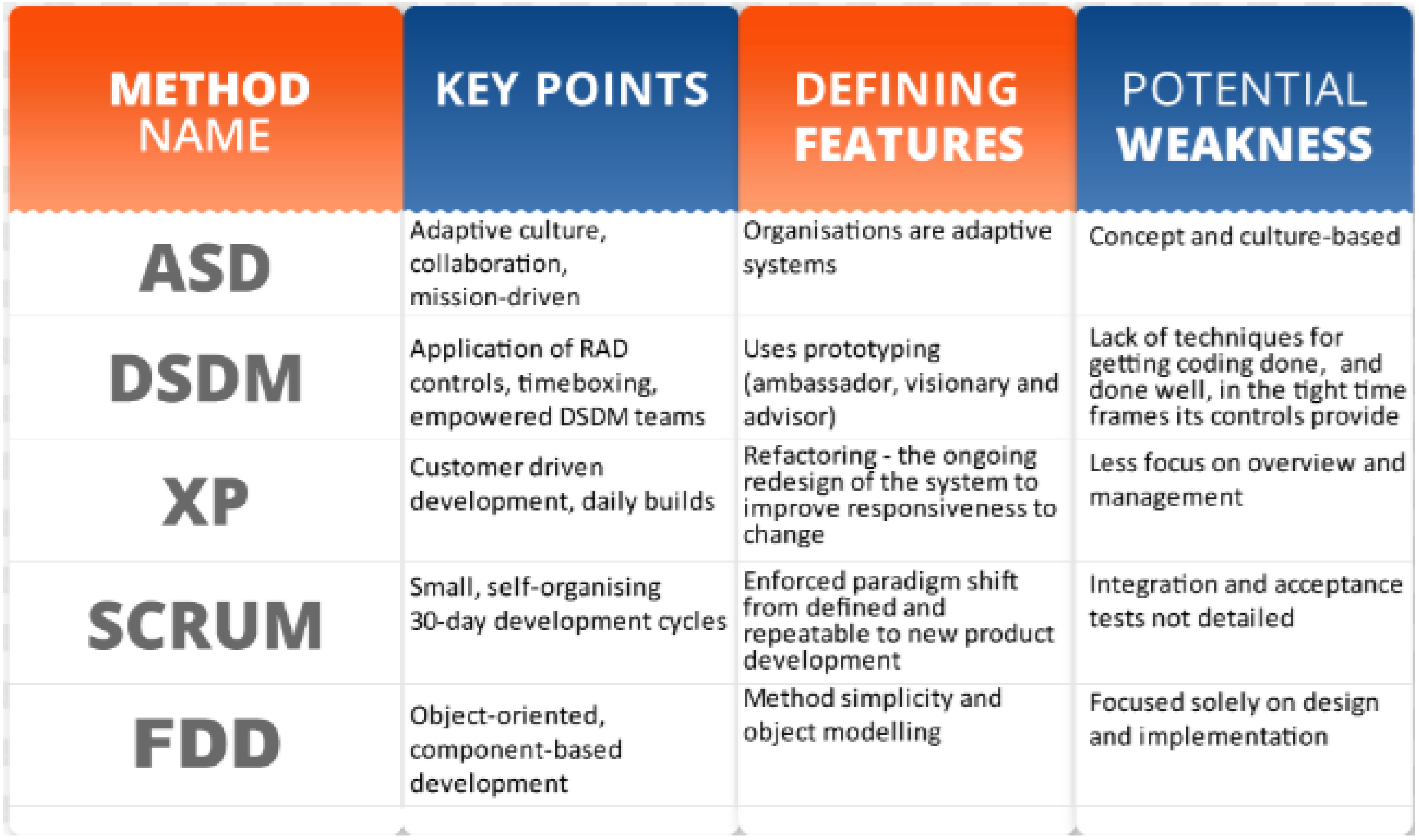
Feature Driven Development



Feature Driven Development Process

1. Creating a Domain object model using object modeling with their domain experts.
2. Developers use the information from Object Modeling and other activities and goes on to create a feature list.
3. A rough plan is prepared based on features and roles, and responsibilities are assigned.
4. Small features or features of feature, whatever takes less than two weeks, are taken up.
5. Designing and coding for each feature happen. It is the construction phase for that feature.

Feature Driven Development



METHOD NAME	KEY POINTS	DEFINING FEATURES	POTENTIAL WEAKNESS
ASD	Adaptive culture, collaboration, mission-driven	Organisations are adaptive systems	Concept and culture-based
DSDM	Application of RAD controls, timeboxing, empowered DSDM teams	Uses prototyping (ambassador, visionary and advisor)	Lack of techniques for getting coding done, and done well, in the tight time frames its controls provide
XP	Customer driven development, daily builds	Refactoring - the ongoing redesign of the system to improve responsiveness to change	Less focus on overview and management
SCRUM	Small, self-organising 30-day development cycles	Enforced paradigm shift from defined and repeatable to new product development	Integration and acceptance tests not detailed
FDD	Object-oriented, component-based development	Method simplicity and object modelling	Focused solely on design and implementation

Next Lecture

- SCRUM