

---

## ■ Agile Development

*Slide Set to accompany*

*Software Engineering: A Practitioner's Approach, 8/e*  
by Roger S. Pressman and Bruce R. Maxim

Slides copyright © 1996, 2001, 2005, 2009, 2014 by Roger S. Pressman

***For non-profit educational use only***

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach, 8/e*. Any other reproduction or use is prohibited without the express written permission of the author.

All copyright information MUST appear if these slides are posted on a website for student use.

# The 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 et al***

# What is “Agility”?

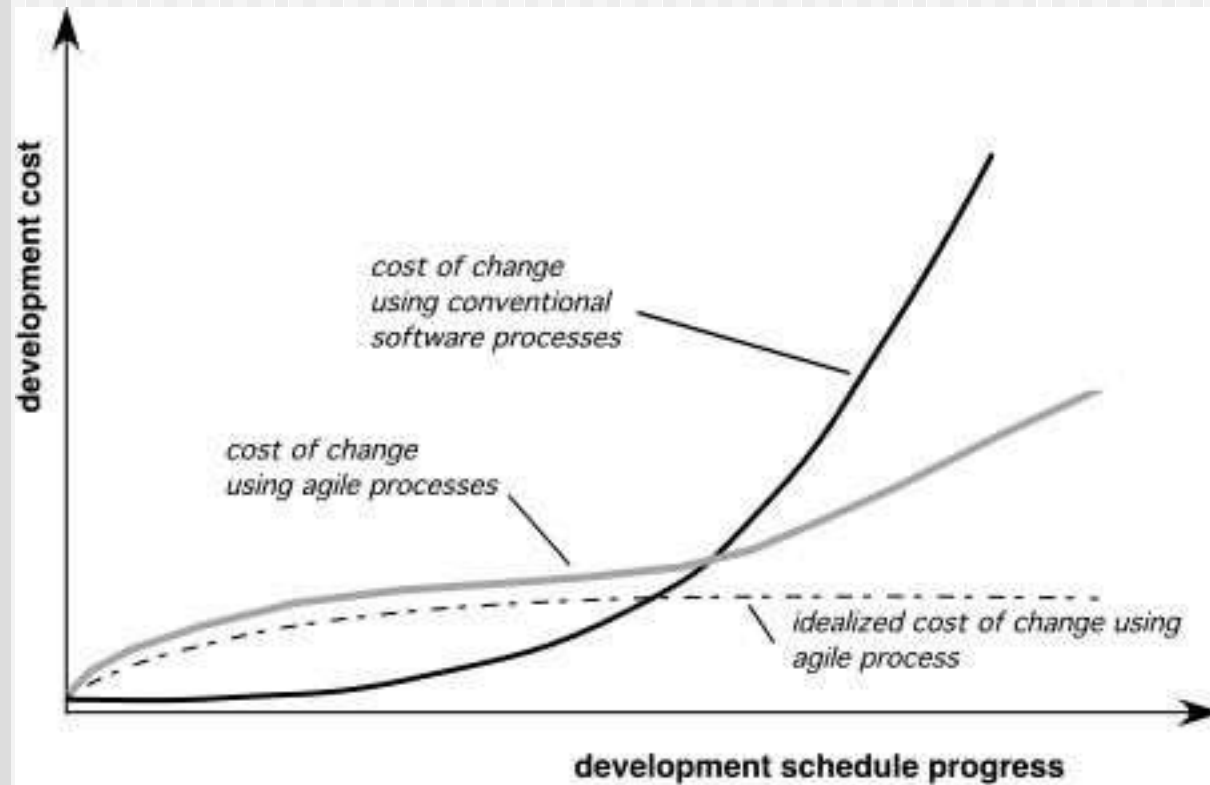
---

- 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

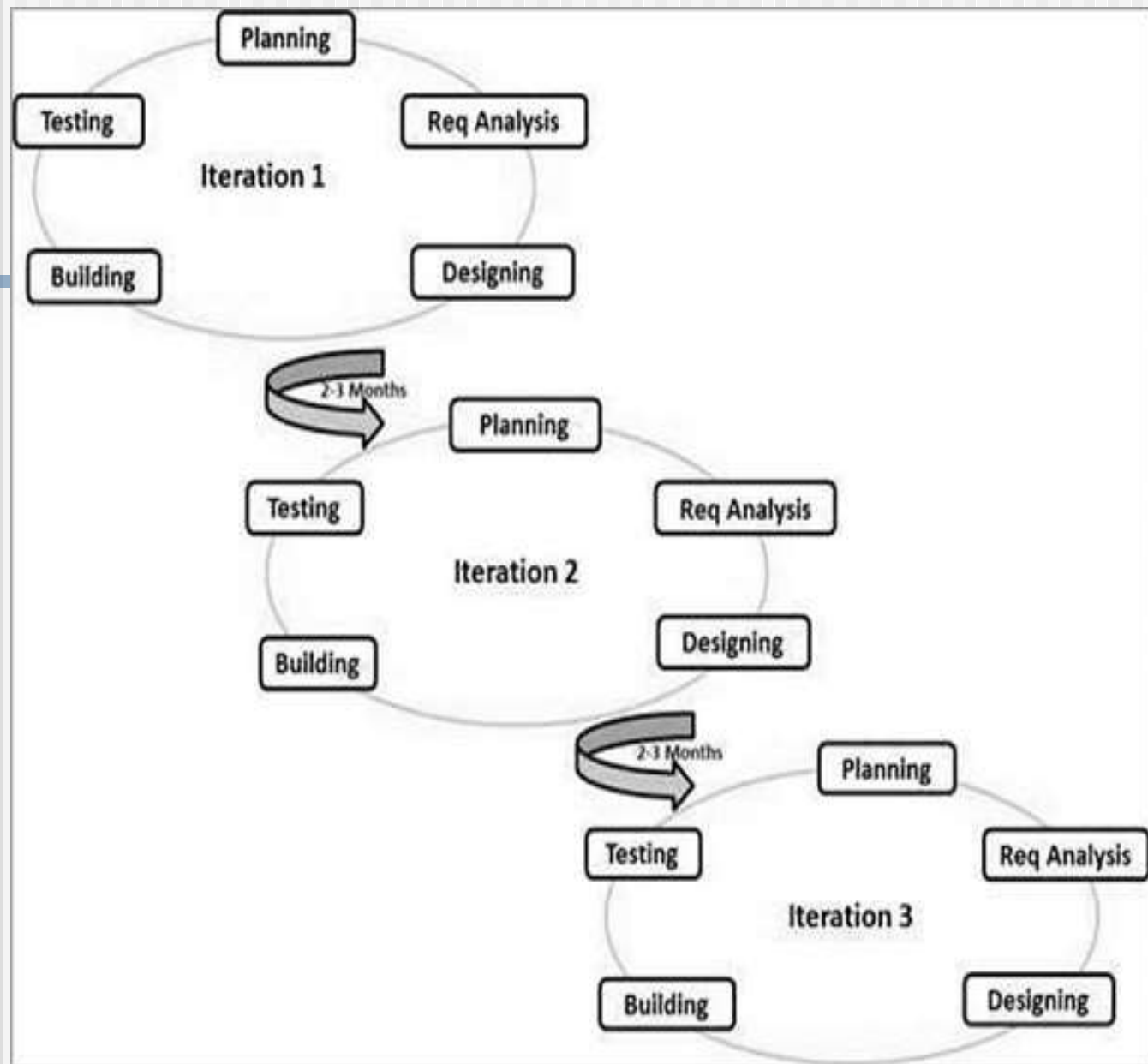
# Agility and the Cost of Change



# 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



# Agility Principles - I

---

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.

# Agility Principles - II

---

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.



# Human Factors

---

- key traits must exist among the people on an agile team and the team itself:
  - **Competence.**
  - **Common focus.**
  - **Collaboration.**
  - **Decision-making ability.**
  - **Fuzzy problem-solving ability.**
  - **Mutual trust and respect.**
  - **Self-organization.**

# Agile Methodologies

- 1 Extreme Programming (XP)
- 2 Adaptive Software Development (ASD)
- 3 Dynamic Systems Development Method (DSDM)
- 4 Scrum
- 5 Crystal
- 6 Feature Driven Development (FDD)
- 7 Lean Software Development(LSD)

# Extreme Programming (XP)

---

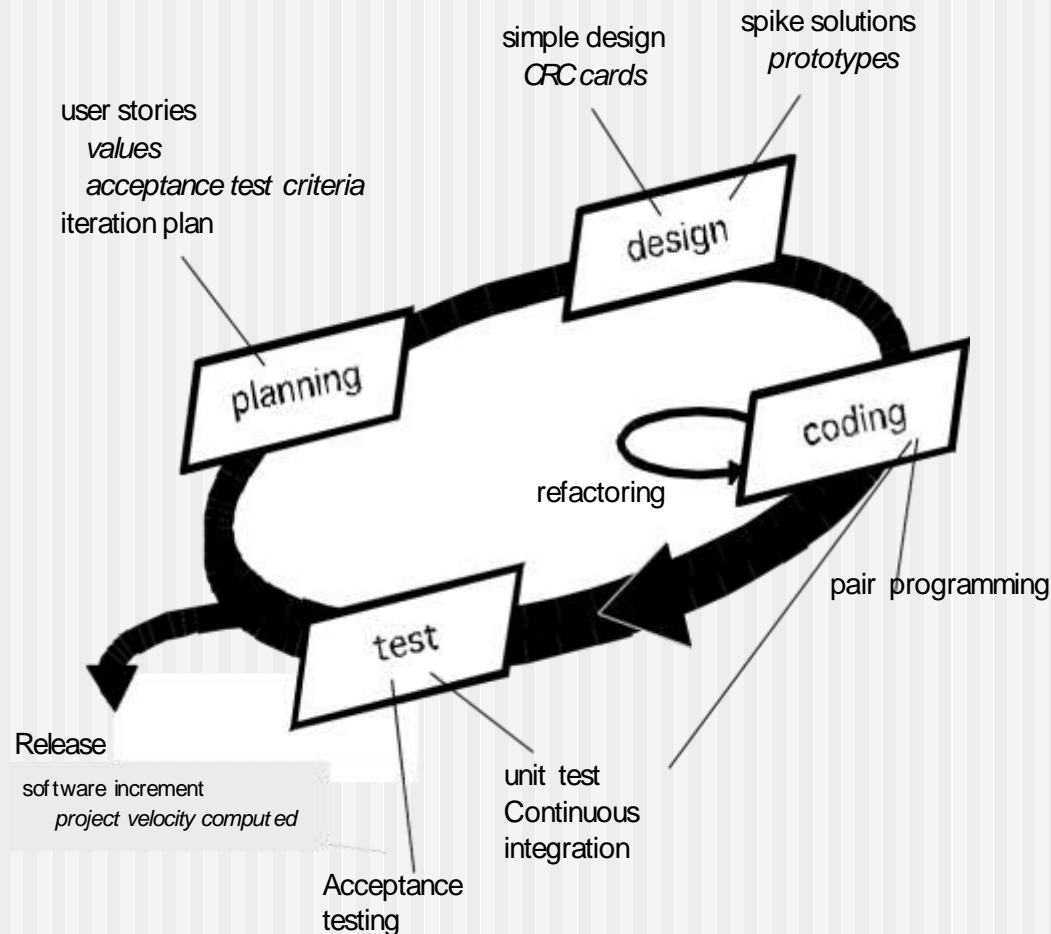
- The most widely used agile process, originally proposed by Kent Beck
- XP Planning
  - Begins with the creation of “user stories”
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment “project velocity” is used to help define subsequent delivery dates for other increments

# Extreme Programming (XP)

---

- XP Design
  - Follows the KIS principle
  - Encourage the use of CRC cards (see Chapter 8)
  - For difficult design problems, suggests the creation of “spike solutions”—a design prototype
  - Encourages “refactoring”—an iterative refinement of the internal program design
- XP Coding
  - Recommends the construction of a unit test for a store *before* coding commences
  - Encourages “pair programming”
- XP Testing
  - All unit tests are executed daily
  - “Acceptance tests” are defined by the customer and executed to assess customer visible functionality

# Extreme Programming (XP)



These slides are designed to accompany *Software Engineering: A Practitioner's Approach*, 8/e (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

# Industrial XP (IXP)

---

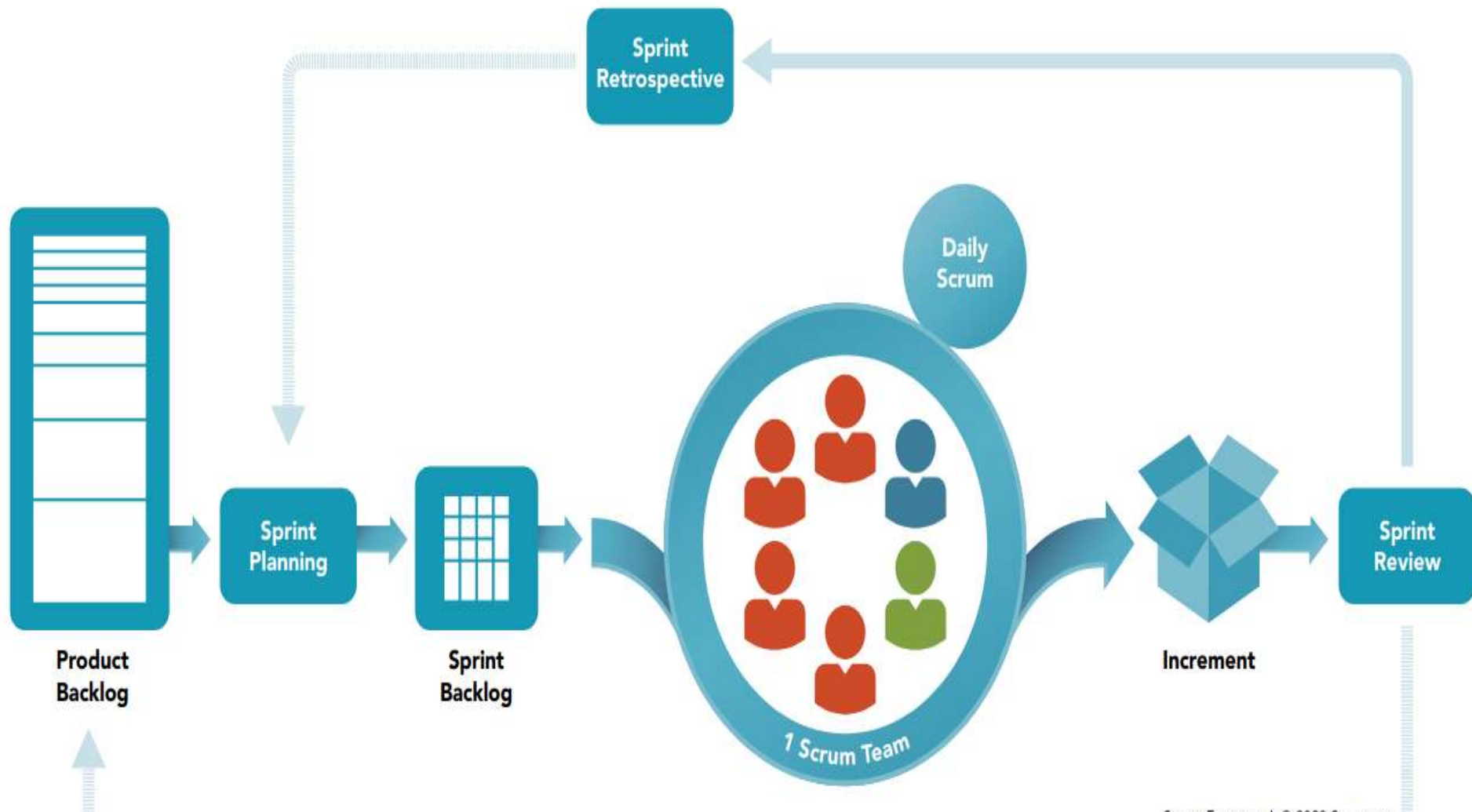
- IXP has greater inclusion of management, expanded customer roles, and upgraded technical practices
- IXP incorporates six new practices:
  - Readiness assessment
  - Project community
  - Project chartering
  - Test driven management-Creating test cases before developing the actual code.
  - Retrospectives
  - Continuous learning

# 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

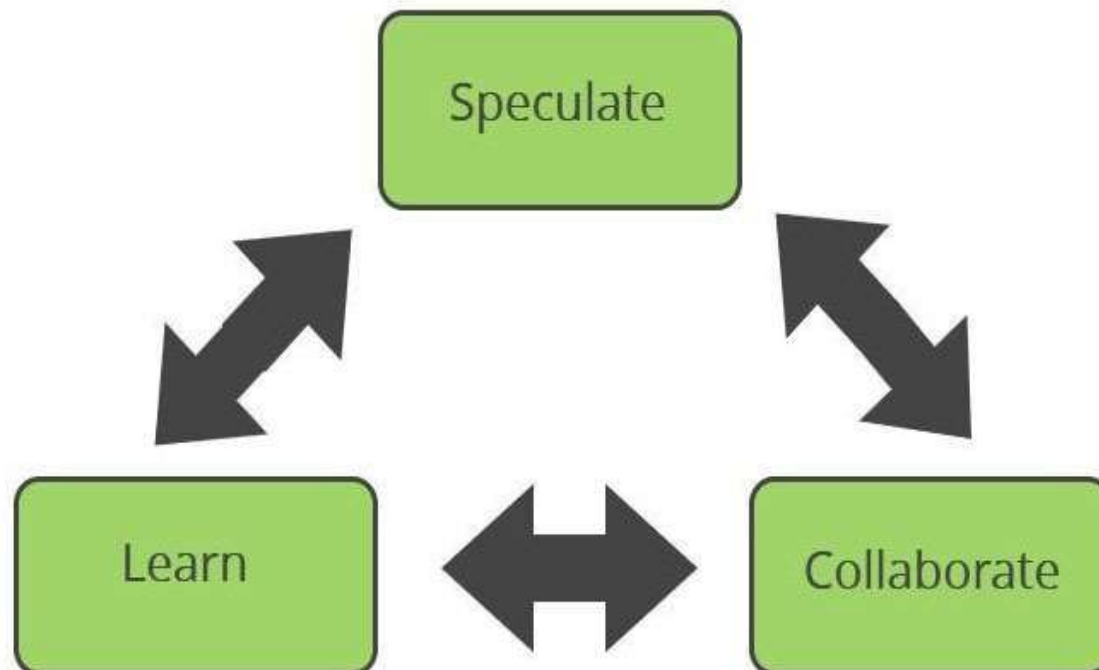
# SCRUM





# Adaptive Software Development (ASD)

## Adaptive Software Development (ASD)



# Adaptive Software Development (ASD)

**This model used to create complex software**

**1. Speculation**

**2. Collaboration**

**3. Learning**

# Adaptive Software Development (ASD)

## 1. Speculation:

- > Adaptive cycle planning
- > The project plan uses project initiation information like project requirements, user needs, customer mission statement, etc,
  - > Fix delivery date, budgets.
  - > Make release plan (ie weekly, monthly etc..)

## 2. Collaboration

The collaborate phase is the phase in which developers actually perform the development work.

This phase is divided into cycles, and each one of them delivers value to the customer.

# Adaptive Software Development (ASD)

## 3. Learning:

- > Learning new skills and technology
- > The workers may have a overestimate of their own understanding of the technology which may not lead to the desired result.
- > Learning helps the workers to increase their level of understanding over the project.
- > The feedback from the customer is taken.

# Lean Software Development(LSD)



# Lean Software Development(LSD)

## **Eliminate waste**

One of the key elements of practicing Lean is to eliminate anything that does not add value to the customer.

- 1 Partially done work
- 2 Extra work but does not add any value(Unused documentation)
- 3 Extra features that are not required.

**Empower the Team.** Software development is a process of mental work, so treat people as competent, motivated professionals, rather than professionals with narrow skills in writing code or drawing diagrams. In order for people to take responsibility, be motivated and work as a solid team, they should be aware of their contribution to the product being developed. It is necessary to create conditions in which each person can be focused working on the current business task. Trust your team and respect it. The human factor is one of the most important elements in successful software development.

# Lean Software Development(LSD)

## **Deliver Fast**

This is the basis of iterative development. The faster you show your groundwork to the customer, the sooner you will get his feedback, thus, he will receive the product with the necessary improvements much sooner.

## **Amplify Learning.**

In order for the team to develop a system that will bring business-value to the customer, they must have a wide range of skills. The team has to accumulate knowledge and share it, for example, in the form of a review at the end of the iteration.

## **Optimize the Whole**

The main way to solve problems is to break them down into smaller issues and consistently eliminate the causes of their occurrence. But in order to see the root of the problem, the team should have a good overall understanding of the current development process, the concept and strategy of the product being developed.

# Lean Software Development(LSD)

## **Build quality in**

Every team wants to build quality into their work.

Pair programming: Avoid quality issues by combining the skills and experience of two developers instead of one

Test-driven development: Writing criteria for code before writing the code to ensure it meets business requirements

Incremental development and constant feedback



# Lean Software Development(LSD)

## **Defer Decision**

To defer commitment means to:

Not plan (in excessive detail) for months in advance

Not commit to ideas or projects without a full understanding of the business requirements

Constantly be collecting and analyzing information regarding any important decisions

# Dynamic Systems Development Method

---

The University of Cambridge was the birthplace of DSDM's concept. This university has worked on dynamic systems development projects since 1972 under the name Concurrent Contextual Design.

In 1987, they developed their own “methodology”, dynamic systems development method (DSDM), **which aimed to shorten the time needed for software production.**

# Dynamic Systems Development Method

## **Pre-project phase**

---

Here, we pick out the projects to work on, register the budget or funds required (and available), and establish project commitment.

## **Project life cycle**

Project lifecycle covers the stages of feasibility and market research, functional model and prototype iteration, design and structure iteration, and execution.

## **Post project phase**

The objective of this phase is to ensure that the product is working efficiently and as per user and company's requirements. This stage requires maintenance, rectifications, and performance enhancement.

# Project Life Cycle Phase of DSDM

---

**Feasibility and market research:** It is a study of necessary success requirements and the constraints of the application

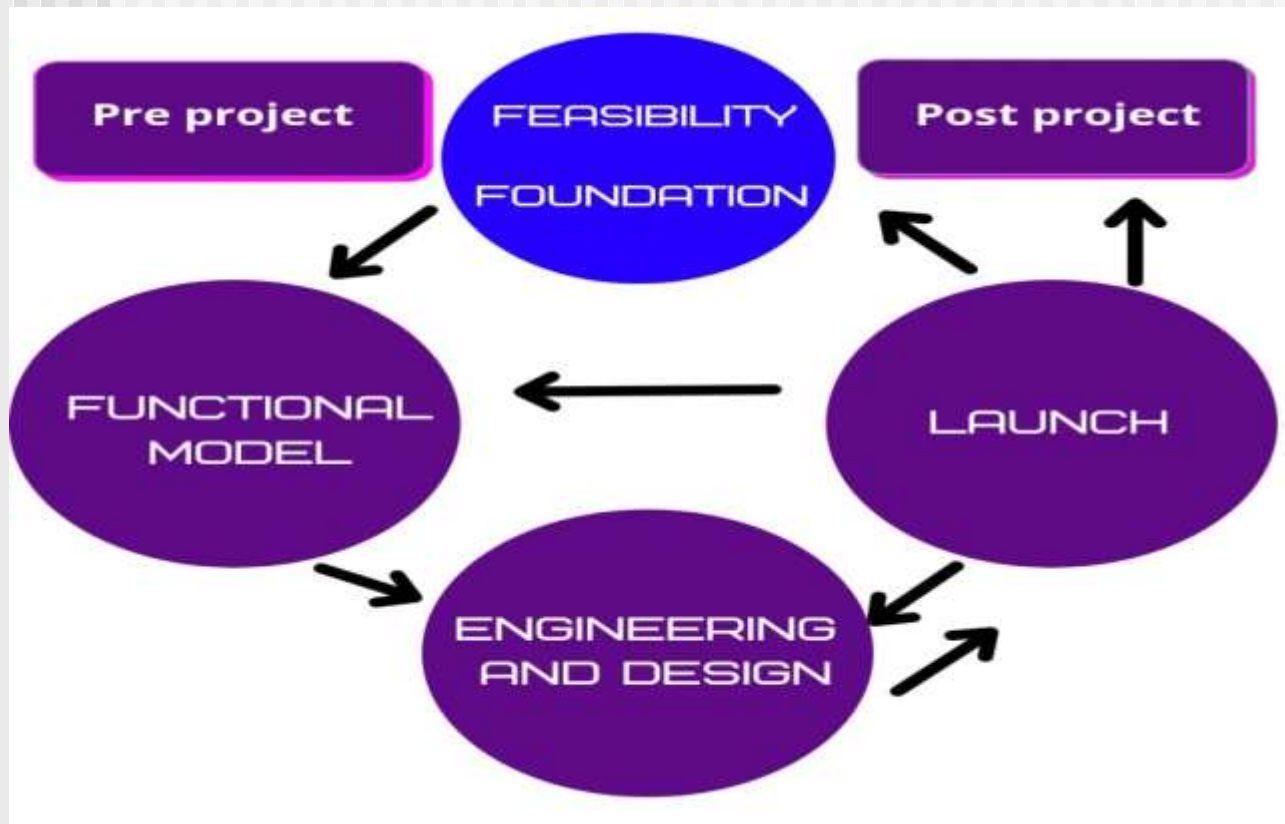
**Functional model and prototype iteration:** This step involves the production of operational models and prototypes at many levels of demonstrations and trials to collect feedback. The iterative testing helps in rectifying the prototype at every level. This way, the prototypes move forward to develop into the actual deliverable product.

**Design and structure iteration:** To ensure that the built functions are by the user's expectations and ease.

**Implementation:** The product is made available to the end-users

# Dynamic Systems Development Method

---



# Principles of DSDM

---

DSDM has eight core principles as follows:

Focus on market needs

Deliver on time

Collaboration

Quality

Build incrementally from firm foundations

Develop iteratively

Communicate continuously and clearly

Demonstrate control: The project manager and team leader should make their plans and progress visible to all and focus on successful delivery.

# Crystal

## Properties of Crystal Agile Framework :

### Frequent Delivery-

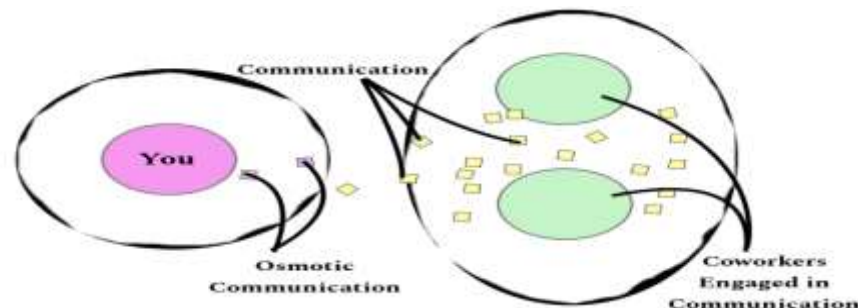
It allows you regularly deliver the products, test code to real users. Without this, you might build a product that nobody needs.


### Reflective Improvement-

No matter how good you have done or how bad you have done. Since there are always areas where the product can be improved, so the teams can implement to improve their future practices.

### Osmotic Communication-

means that information flows into the background hearing of members of the team, so that they pick up relevant information as though by osmosis.





“osmosis” in a biology class. It’s the process by which water molecules pass through a semipermeable membrane to equalize the concentration of solutes on either side.

Osmotic communication is similar in describing the absorption of information across a barrier.

Osmotic communication occurs whenever team members are physically co-located with one another. For instance, if you overhear a colleague talking about a new client project they’re working on, you’re engaging in osmotic communication.



# Osmotic Communication



Osmotic communication occurs when information is transmitted so that team members can pick up pertinent information by osmosis. Usually, this is accomplished by placing them in the same space.

# Crystal

## Personal Safety-

There are no bad suggestions in a crystal team, team members should feel safe to discuss ideas openly without any fear.

## Focus-

Each member of team knows exactly what to do, which enables them to focus their attention. This boosts team interaction and work towards the same goal.

**Customer involvement** – The framework emphasizes on involving customers in the development process, promoting customer satisfaction, and delivering products that meet their needs.

## Technical tooling-

It contains very specific technical tools which to be used by software development team during testing, management and configuration. These tools make it enable for the team to identify any error within less time.

# Crystal

Crystal family consists of many variants like [Crystal Clear](#), [Crystal Yellow](#), [Crystal Red](#), [Crystal Sapphire](#), [Crystal Red](#), [Crystal Orange Web](#), [Crystal Diamond](#).

## [Crystal Clear-](#)

The team consists of only 1-6 members that is suitable for short-term projects where members work out in single workspace.

## [Crystal Yellow-](#)

It has a small team size of 7-20 members, where feedback is taken from Real Users. This variant involves automated testing which resolves bugs faster and reduces use of too much documentation.

## [Crystal Orange-](#)

It has a team size of 21-40 members, where team is split according to their functional skills. Here the project generally lasts for 1-2 years and the release is required every 3 to 4 months.

# Crystal

## Crystal Orange Web-

It has also the team size of 21-40 members where the projects that have a continually evolving code base that is being used by the public.

## Crystal Red-

The software development is led by 40-80 members where the teams can be formed and divided according to requirements.

## Crystal Maroon-

It involves large sized projects where team size is of 80-200 members where methods are different and as per the requirement of the software.

## Crystal Diamond & Sapphire-

This variant is used in large projects .

# Feature Driven Development (FDD)

FDD was designed to follow a five-step development process, built largely around discrete “feature” projects. That project lifecycle looks like this:

Develop an overall model

Build a features list

Plan by feature

Design by feature

Build by feature



# Feature Driven Development (FDD)

## Develop An Overall Model:

The client and the development team make an overall model. Detailed domain models are created and then these models are progressively merged into the overall model.

## Develop a Feature List

Next up is the feature list — all the functionality the product needs to offer.

# Feature Driven Development (FDD)

## Plan by Feature

The third step is where you plan the order in which to develop features and feature sets.

Factors to consider, are:

Which features your customer's users need most urgently.

Which features will deliver the most value soonest.

Development time estimates

Risks involved with features — for example , using new technology, learning new frameworks and libraries.

Available people, their skills and expertise.

Workload.

# Feature Driven Development (FDD)

## Design by Feature

Each feature team works on the detailed design of the features assigned to them for the current iteration

## Build by Feature

Each feature team then works to turn their design into working software, test it, and gather feedback from domain experts to verify that the feature works as intended



# AGILE MODELING

Agile Modeling (AM) is a practice-based methodology for effective modeling and documentation of software-based systems.

Agile Modeling (AM) is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light-weight manner.

Agile modeling helps developers create a customized software development process that fulfills their development needs yet is flexible enough to adjust to future situations.

# Modeling principles

Model with a purpose

Use multiple models

Travel light- have sufficient documentation about the models you're developing

Incremental Change.

Know the models and the tools you use to create them

# Modeling principles

Adopt Simplicity-Keep the models as straightforward and

simpli|learn

## Stages of Agile Modelling





## Agile Unified Process Model

Agile unified process (AUP) is a simplified version of the rational unified process (RUP) developed by Scott Ambler.[1]

It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts



Unlike the RUP, the AUP has only seven disciplines[citation needed]:

**Model.** Understand the business of the organization, the problem domain being addressed by the project, and identify a viable solution to address the problem domain.


**Implementation.** Transform model(s) into executable code and perform a basic level of testing, in particular unit testing.



Unlike the RUP, the AUP has only seven disciplines[citation needed]:


**Test:** Perform an objective evaluation to ensure quality. This includes finding defects, verifying that the system works as designed, and validating that the requirements are met.

**Deployment.** Plan for the delivery of the system and to execute the plan to make the system available to end users.



Configuration management. Manage access to project artifacts. This includes not only tracking artifact versions over time but also controlling and managing changes to them.

Project management. Direct the activities that take place within the project. This includes managing risks, directing people (assigning tasks, tracking progress, etc.), and coordinating with people and systems outside the scope of the project to be sure that it is delivered on time and within budget.



**Environment:** Support the rest of the effort by ensuring that the proper process, guidance ( standards and guidelines), and tools (hardware, software, etc.) are available for the team as needed.



# Selection of an appropriate Project Approach- Choice of process Models.

## LOOK AT RISK AND UNCERTAINTIES

- are requirement well understood?
- are technologies to be used well understood?

## LOOK AT THE TYPE OF APPLICATION BEING BUILT

- information system? Embedded system?

## CLIENTS OWN REQUIREMENTS

- need to use a particular model