Agile Development – an overview

- Development guidelines
- Encourages
  - *customer satisfaction*
  - *Early delivery of Software increment*
  - *small & highly motivated team*
  - *minimal software engineering work products*
  - *overall development simplicity*
- How?
  - *Software engineers & other stake holders will work together as a agile team*

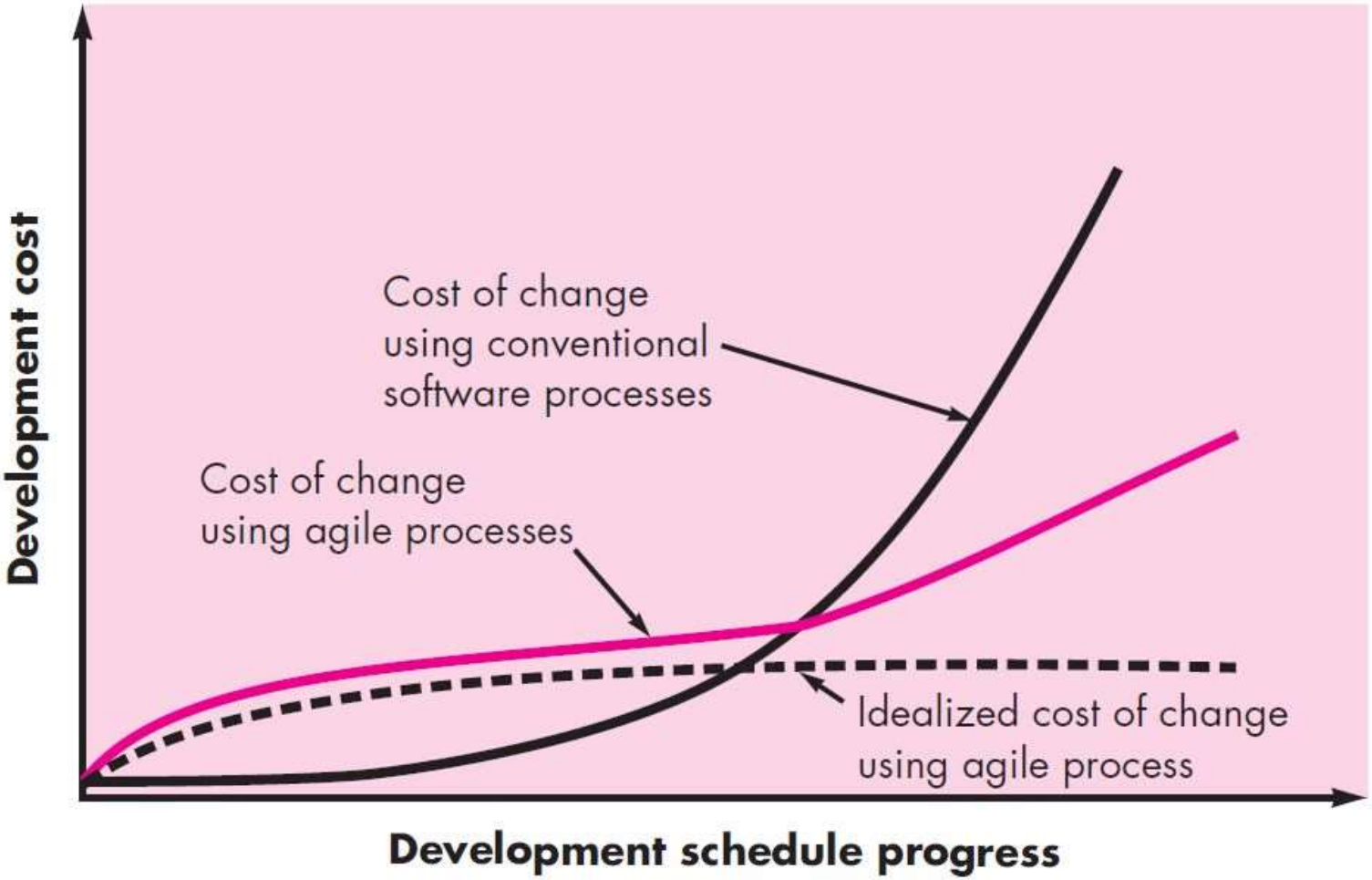
What is Agility?

- An agile team – able to appropriately respond to changes
- Changes might be
  - In the software being built
  - to the team members
  - Because of new technology
- Changes of any kind impact the software product development
- Support for changes should be built-in everything
- Team collaboration – is the core for the success of the project development
- The pervasiveness of change is the primary driver for agility

What is Agility?

- Encourages team structures and attitudes that make communication more facile
- Emphasizes rapid delivery of operational software
- De-emphasizes the importance of intermediate work products
- Adopts the customer as a part of the development team

Agility and the cost of change



What is an Agile process?

- Any agile software process is characterized in a manner that addresses a number of key assumptions:
  - Difficult to predict the change in nature of the requirements and changes in priority of the customer
  - As design and construction process are interleaved, it is difficult to predict how much design is necessary before construction is used to prove the design
  - Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.
- Software process that manages ***unpredictability*** – ***should be adaptable***

The Agility Principles

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

The Agility Principles

- ***Working software*** is the ***primary measure*** of progress.
- Agile processes promote ***sustainable development***. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- ***Continuous attention*** to ***technical excellence and good design*** enhances agility.
- ***Simplicity***—the art of maximizing the amount of work not done—is essential.
- The ***best architectures, requirements, and designs*** emerge from self– organizing teams.

Human Factors

- Competence – Skill & Knowledge
- Common focus - to deliver a working software increment to the customer within the time promised
- Collaboration
- Decision-making ability
- Fuzzy problem-solving ability
- Mutual trust and respect
- Self-organization

Extreme Programming (XP)

- Widely used agile software development
- **IXP** – Industrial Extreme Programming – refines XP and used in **Larger organizations**
- **XP values**
  - Communication
    - informal between the software engineers and the stake holders – metaphors – continuous feedback – reduce documentation
  - Simplicity
    - restricts developer to design immediate needs – then, refactored if required
  - Feedback
    - Software (Unit test) – Customer (User stories – acceptance testing) – Software team members (cost and schedule impact)

Extreme Programming (XP)

- **XP values**
  - Courage
    - strict adherence to certain XP practices – Discipline to design for today
  - Respect
    - Between members, between other stakeholders and team members, and indirectly, for the software itself

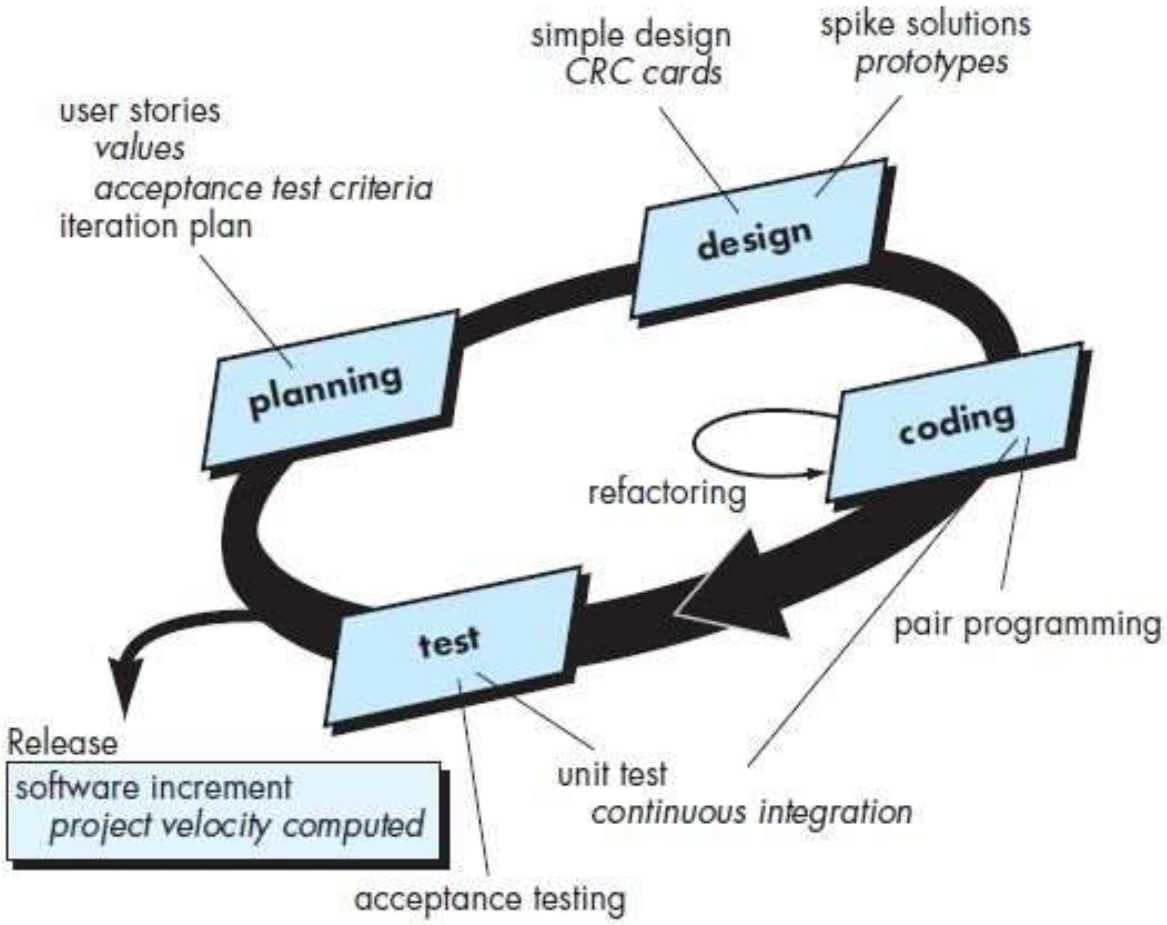
Extreme Programming (XP)

- **XP values**
  - Courage
    - strict adherence to certain XP practices – Discipline to design for today
  - Respect
    - Between members, between other stakeholders and team members, and indirectly, for the software itself

The XP Process

- An object-oriented approach as its preferred development paradigm
- Framework activities of XP process
    - Planning
    - Design
    - Coding
    - Testing

XP Process


simple design
CRC cards
spike solutions
prototypes
user stories
values
acceptance test criteria
iteration plan
design
planning
coding
refactoring
test
pair programming
Release
software increment
project velocity computed
unit test
continuous integration
acceptance testing

XP process – framework activities

- **Planning**
  - listening - a requirements gathering activity that enables the technical members of the XP team to understand the business context for the software
  - Listening leads to "User stories" - required output, features, and functionality for software to be built
  - an index card – valued by XP team – cost assigned (Development weeks)
  - After Commitment (agreement on stories to be included, delivery date, and other project matters)
    - all stories will be implemented immediately (within a few weeks)
    - the stories with highest value will be moved up in the

Planning

- **Project velocity** is the number of customer stories implemented during the first release
- It can be used to

  (1) help estimate delivery dates and schedule for subsequent releases

  (2) determine whether an overcommitment has been made for all stories across the entire development project.

*If an overcommitment occurs, the content of releases is modified or end delivery dates are changed.

Design

- Follows “Keep it simple” principle
- class-responsibility-collaborator Cards
- Spike solution - difficult problem
- Refactoring
  - “the process of changing a software system in such a way that it does not alter the external behaviour of the code yet improves the internal structure. It is a disciplined way to clean up code [and modify/simplify the internal design] that minimizes the chances of introducing bugs. In essence, when you refactor you are improving the design of the code after it has been written”

Coding

- Preliminary design work is done, the team **does *not*** move to code
- Develops a series of unit tests that will exercise each of the stories
- *pair programming*

Testing

- Unit tests
- Regression testing
- "universal testing suite" – Integration & Validation testing
- Acceptance tests

Industrial XP

- IXP is an organic evolution of XP.
- It is imbued with XP's minimalist, customer-centric, test-driven spirit
- Updates of IXP
  - Greater inclusion of management
  - Expanded role for customers
  - Upgraded technical practices

New practices of IXP

- **Readiness assessment**
- **Project community**
- **Project chartering**
- **Test-driven management**
- **Retrospectives – "**issues, events, and lessons-learned"
- **Continuous learning**

Readiness Assessment

(1) an **appropriate development environment** exists to support IXP

(2) the team will be populated by the **proper set of stakeholders**

(3) the organization has a distinct quality program and supports **continuous improvement**

(4) the organizational culture will support **the new values of an agile team**

(5) the **broader project community** will be populated appropriately.

The XP – Drawbacks

Requirements volatility

Conflicting customer needs

Informal communication

Lack of formal design

The XP – summary

Story Driven Development

Domain Driven Development

Pairing

Iterative Usability