CHAPTER 2
AGILE DEVELOPMENT

TOPICS COVERED IN THIS CHAPTER:

- Agility and Agile Process model
- Extreme Programming
- Other process models of Agile Development

AGILITY:

- Agility is dynamic, content specific, change acceptance and growth oriented process.
- It encourages team structures and attitudes that make **communication** (among team members, between technologists and business people, between software engineers and their managers) more **simplistic**.
- It emphasizes **rapid delivery** of operational software.
- It recognizes that planning in an uncertain world has its limits and that a **project plan must be flexible.**

AGILE PROCESS:

- Any agile software process is characterized in a manner that addresses a number of **key assumptions**
  - It is **difficult to predict** in advance which **software requirements** will persist and which will change. It is equally difficult to predict how **customer priorities** will change as the project proceeds.
  - 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 as we might like.

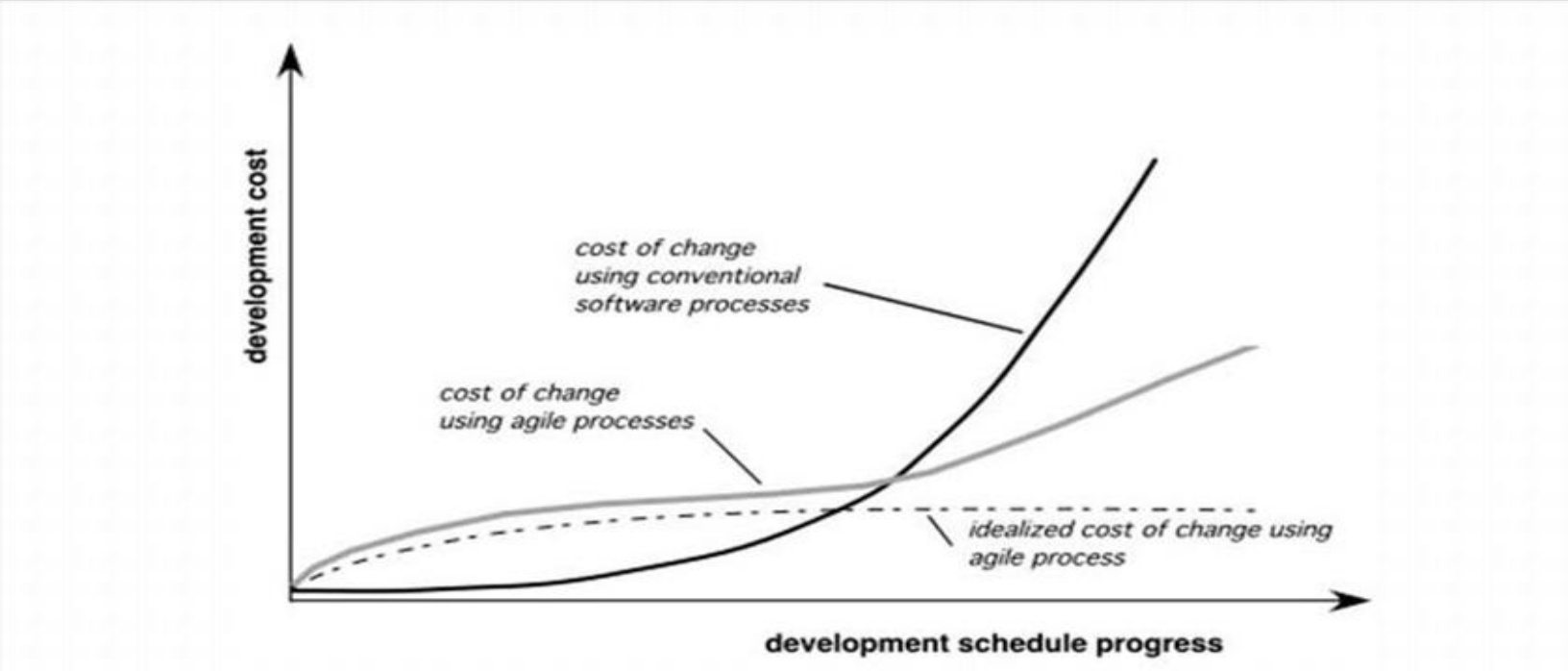
AGILITY PRINCIPLES:

- Highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
- **Welcome changing requirements**, even late in development.
- **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 information to and within a development team is **face-to-face conversation**.

CONT…

- **Working software** is the **primary measure** of progress.
- Agile processes promote **maintainable development**. The sponsors, developers and users should be able to **maintain a constant step of progress**.
- Continuous attention to **technical excellence and good design** enhances agility.
- At regular intervals, the team reflects on **how to become more effective**, then adjusts its behavior accordingly.

AGILITY AND COST OF CHANGE:



AGILE PROCESS MODELS:

- Extreme Programming (XP)
- Scrum
- Crystal
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)
- Agile Modeling (AM)

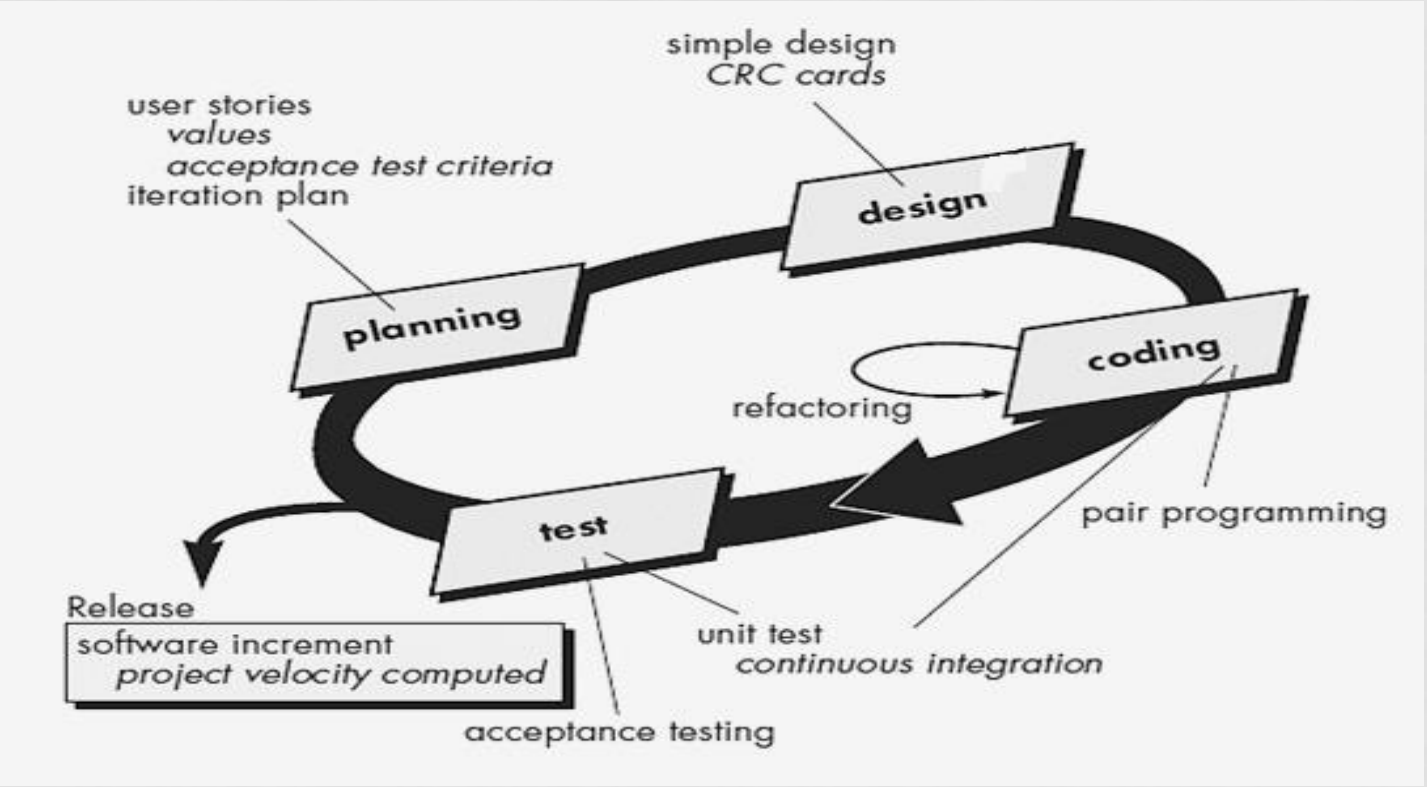
EXTREME PROGRAMMING (XP):

- It is most widely used agile process model.
- It defines **four** framework activities
  - Planning
  - Design
  - Coding
  - Testing.
- It is an example of how Agile can **heighten customer satisfaction**. Rather than deliver everything the customer could ever want in the future, it gives them what they need now, fast.
- XP is centered on **frequent releases** and **short development cycles.**

EXTREME PROGRAMMING (XP):

- **XP Values:**
  - **Communication**:
    - To achieve effective communication between software engineers and stakeholders, it emphasizes **informal communication, continuous feedback**.
  - **Simplicity**:
    - To achieve simplicity, it restricts developers to **design only for immediate needs**, rather than future needs.
  - **Feedback**:
    - It derives from **3 sources**: implemented software, customer, other software team members

CONT…


simple design
CRC cards
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

CONT…

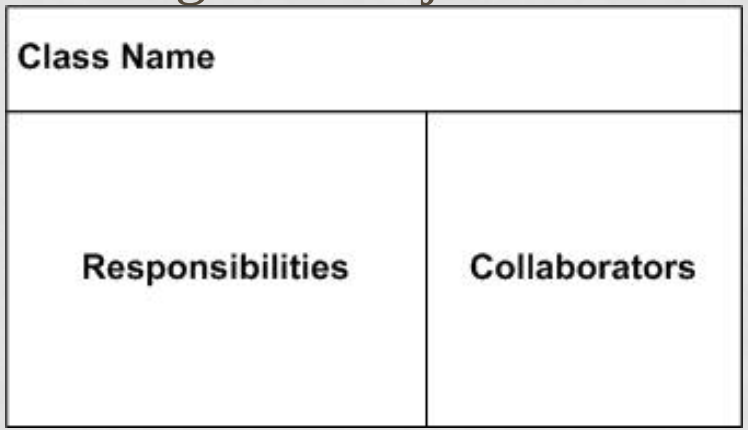
- **Planning:**
  - Begins with the creation of a set of stories (also called user stories)
  - Each story is written by the customer and is placed on an index card
  - The customer assigns a value (i.e. a priority) to the story
  - Agile team assesses each story and assigns a cost
  - Stories are grouped 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.

CONT…

- **Design:**
  - Follows the keep it simple principle
  - Encourage the use of CRC (class-responsibility-collaborator) cards
  - Encourages “refactoring”—an iterative refinement of the internal program design.

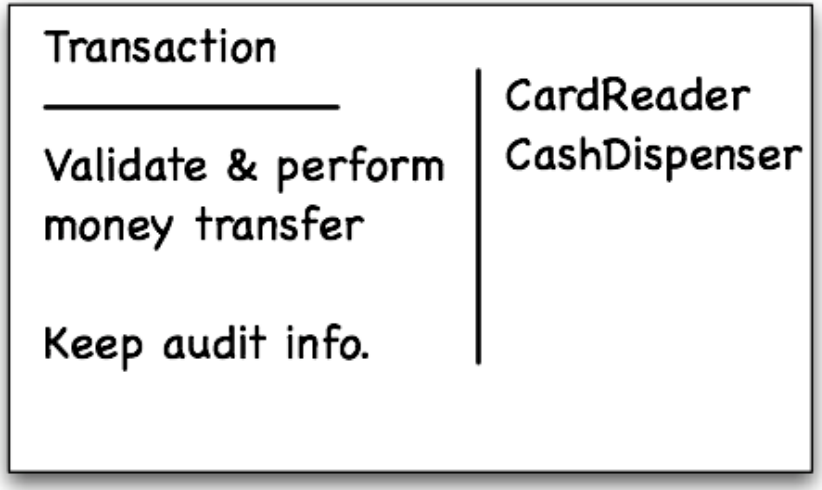
CONT…

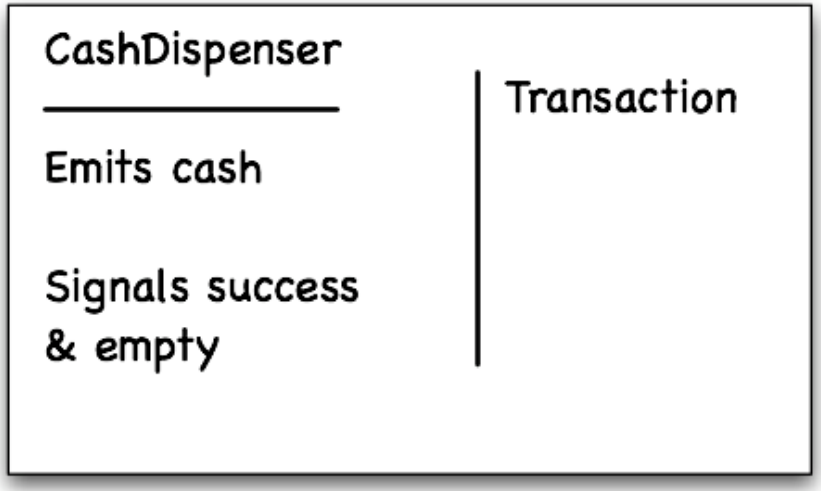
- **CRC Card:**
  - Brainstorming tool used in the design of object-oriented software
  - It includes
    - Class name
    - Responsibilities of the class
    - Collaborators
      - Other classes with which this class interacts to fulfill its responsibilities



CONT…

- **CRC Card (Example):**





CONT…

- **Coding:**
  - Recommends the construction of a series of **unit tests for each of the stories before coding**.
  - Encourages "**pair programming**"
    - Developers work in pairs, checking each other's work and providing the support to always do a good job
    - Mechanism for **real-time** problem solving and real-time quality assurance
    - Keeps the developers focused on the problem
  - Needs **continuous integration** with other portions (stories) of the s/w.

CONT…

- **Testing**:
  - Unit tests should be implemented using a framework to make **testing automated**.
  - **Integration and validation testing** can occur on a **daily** basis
  - Acceptance tests, also called customer tests, are specified by the customer and executed to assess customer visible functionality

SCRUM:

- Scrum principles are consistent with the agile strategy and are used to guide development activities within a process that incorporates the five framework activities:
  - Requirements
  - Analysis
  - Design
  - Evolution
  - Delivery

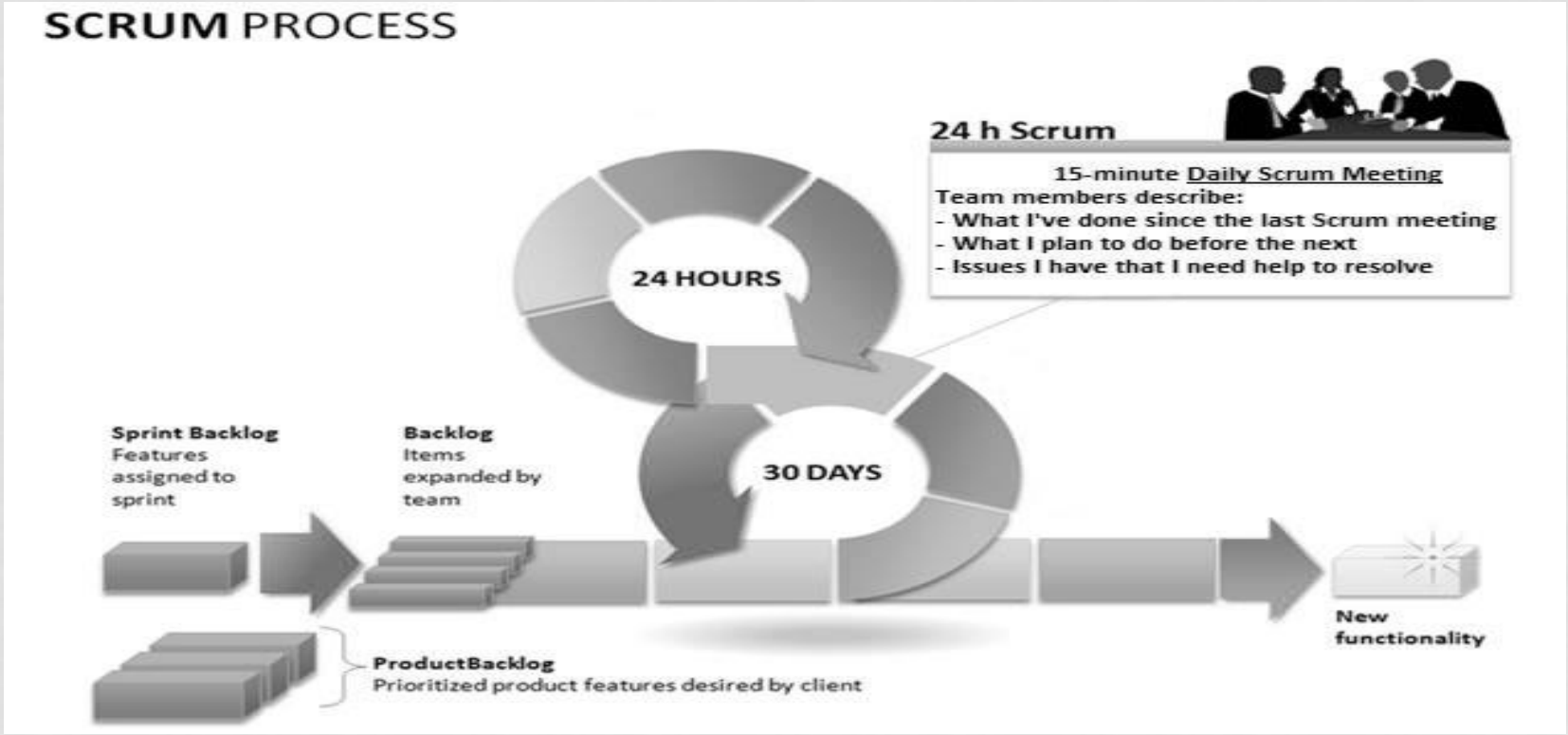
SCRUM:

- Scrum is a hands-on system consisting of simple interlocking steps and components:
  - A product owner makes a prioritized wish list known as a product **Backlog**.
  - The scrum team takes one small piece of the top of the wish list called a **Sprint** backlog and **plans** to implement it.
  - The team completes their sprint backlog task in a sprint (**a 2-4 week period**). They assess progress in a meeting called a daily **Scrum meeting**.
  - The **ScrumMaster** keeps the team focused on the goal.
  - At the sprint's end, the work is ready to ship or show. The team closes the sprint with a review that is called **Demos**, then starts a new sprint.

SCRUM:

- **Example of how Scrum works:**
  - Bill meets with a customer to discuss her company's needs. Those needs are the **product backlog**.
  - Bill chooses the most important tasks to work on in the next two weeks that is called **Sprint**.
  - His team meets in a daily scrum to target work for the day ahead and address **roadblocks**.
  - At the end of the sprint, Bill delivers the work, reviews the backlog and sets the goal for the next sprint.
  - The cycle repeats until the software is complete.

CONT…

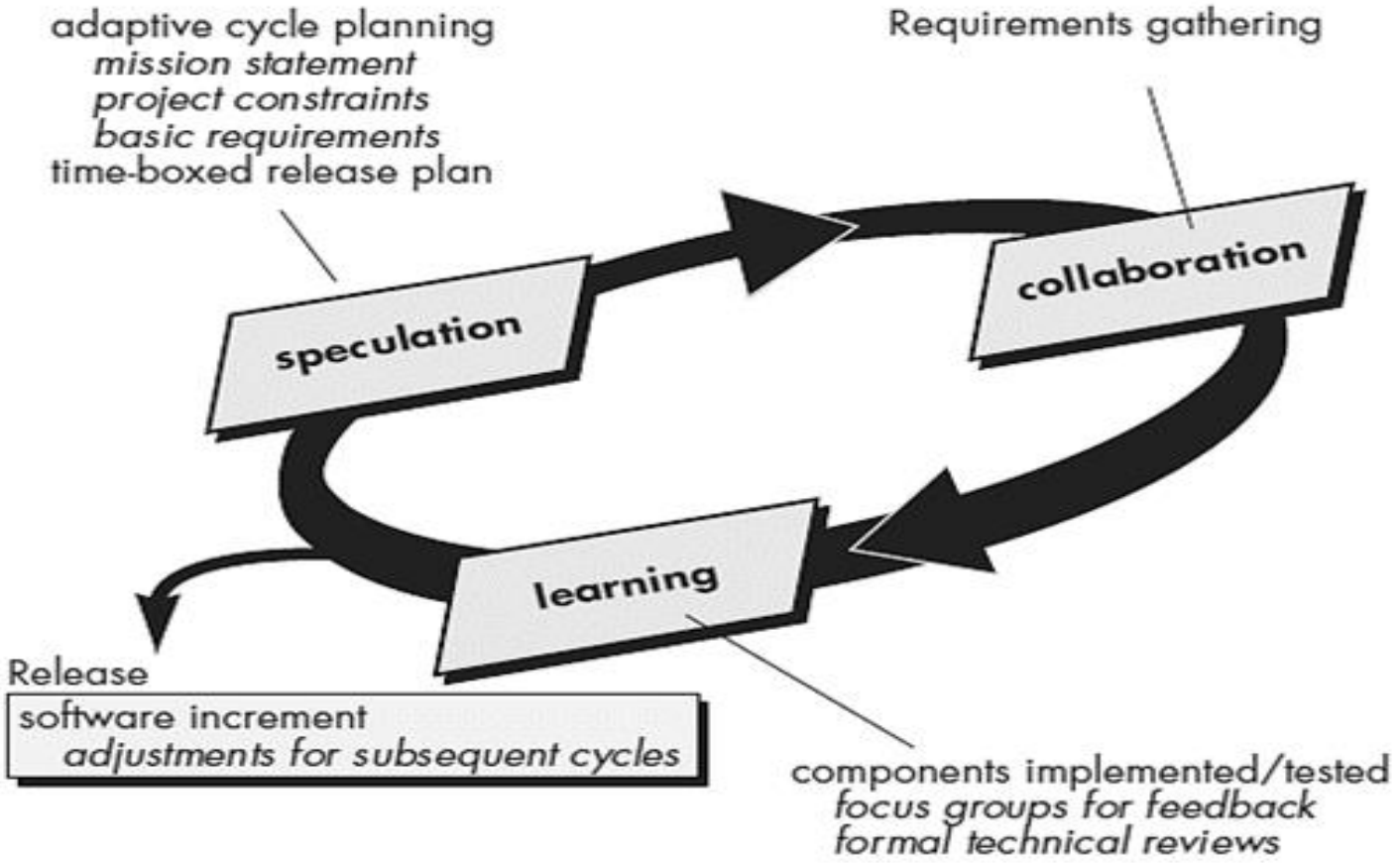

SCRUM PROCESS
24 h Scrum
15-minute Daily Scrum Meeting
Team members describe:
- What I've done since the last Scrum meeting
- What I plan to do before the next
- Issues I have that I need help to resolve
24 HOURS
30 DAYS
Sprint Backlog
Features
assigned to
sprint
Backlog
Items
expanded by
team
New
functionality
ProductBacklog
Prioritized product features desired by client

CRYSTAL:

- It is one of the most adaptable approaches to software development.
- Crystal family addresses the realization that each project may require a **set of policies**, **practices** and **processes** in order to meet the **project's unique characteristics**.
- Key principles of Crystal include **teamwork, communication and simplicity as well as reflection to frequently adjust** and **improve** the process.
- Crystal promotes **early, frequent delivery of working software**, high **user involvement**, **adaptability**.

ADAPTIVE SOFTWARE DEVELOPMENT (ASD):

- It is a
complex
- It focus
- It incorp
  - Spec
  - Colla
  - Learn



CONT…

- **Speculation**:
  - The project is initiated and **adaptive cycle planning** is conducted.
  - Adaptive cycle planning uses project **initiation information** such as customer's **mission statements**, project **constraints** (delivery date) and basic **requirements** to define set of release cycle (**increments**).
  - Based on information obtained at the completion of the first cycle, the plan is **reviewed and adjusted** for the plan of next cycle.

CONT…

- **Collaboration**:
  - It encompasses **communication** and **teamwork**, but it also focus on **individual creativity**.
  - Collaboration weights speculation in that a **project manager** plans the work between the **predictable parts** of the environment and **adapts to the uncertainties** of various factors such as stakeholders, requirements, software vendors, technology, etc.

CONT…

- **Learning**:
  - "Learning" cycles challenge all stakeholders and project team members.
  - Based on **short iterations** of design, build and testing, knowledge accumulates from the **small mistakes** due to **false assumptions**, **poorly stated requirements** or **misunderstanding of the stakeholders' needs**.
  - **Correcting** those mistakes through learning cycles leads to greater **positive experience**.
  - It will help them to improve their level of real understanding.

DYNAMIC SYSTEMS DEVELOPMENT METHODS (DSDM):

- This method is an agile software development approach that "**provides a framework** for building and maintaining systems which **meet tight time constraints** through the use of **incremental** prototyping in a **controlled project environment**"
- It is an **iterative software process** in which each iteration follows the **80 percent rule** i.e only enough work is required for each increment to facilitate movement to the next increment.
- The remaining detail can be completed later when more business requirements are known or changes have been requested and accommodated.

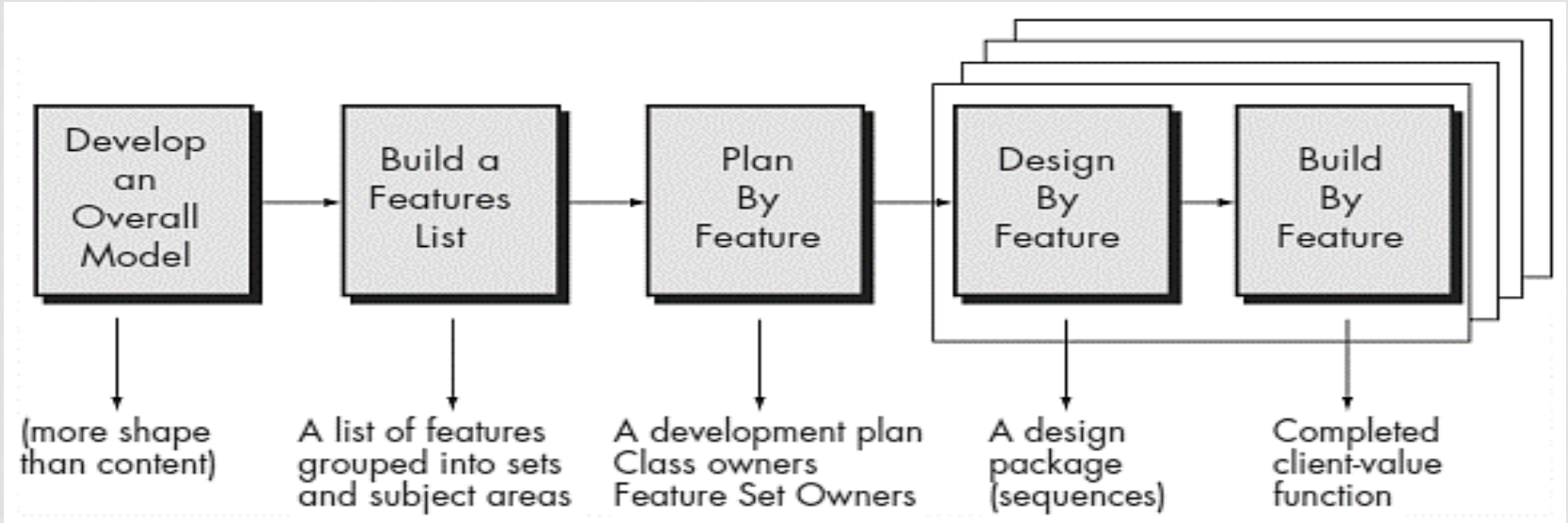
CONT…

- DSDM life cycle that defines different life cycle activities:
  - **Feasibility study:**
    - It establishes the basic **business requirements and constraints** associated with the application to be built and then assesses whether the application is a **feasible** for the DSDM process.
  - **Business study:**
    - It establishes the **functional and information requirements** that will allow the application to provide **business value**.
    - It defines the basic **application architecture** and identifies the **maintainability requirements** for the application.

CONT…

- **Functional model iteration:**
  - It produces a **set of incremental prototypes** that **demonstrate functionality** for the customer.
- **Design and build iteration:**
  - It **revisits prototypes** built during functional model iteration to ensure that each has been engineered in a manner that will enable it to provide operational business value for end users.
- **Implementation:**
  - It places the **latest** software increment into the **operational environment**.

FEATURE DRIVEN DEVELOPMENT (FDD):



CONT…

- FDD is a **model-driven**, short-iteration process.
- It **begins** with establishing an overall **model shape**.
- Then it continues with a series of **two-week** "**design by feature, build by feature**" iterations.
- The features are **small**.
- It recommends specific **programmer practices** such as "**Regular Builds**".
- Unlike other agile methods, FDD describes **specific**, very **short phases** of work, which are to be accomplished **separately per feature**.
- It includes **Domain Walkthrough**, **Design**, **Design Inspection**, **Code**, **Code Inspection** and **Encourage to Build**.

CONT…

- FDD designs the rest of the development process around feature delivery using the following practices:
  - Domain Object Modeling
  - Developing by Feature
  - Feature Teams
  - Inspections
  - Configuration Management
  - Regular Builds
  - Visibility of progress and results

AGILE MODELING (AM):

- It is a **practice-based** methodology for **effective** modeling and documentation of **software-based systems**.
- It is a collection of **values**, **principles** and **practices** for modeling software that can be applied on a software development project in an effective manner.
- It suggests a wide array of "**core**" and "**supplementary**" modeling principles.

AGILE MODELING (AM):

- **Use multiple models**
  - There are many **different models and notations** that can be used to describe software.
  - It suggests that to provide **needed insight**, each model should present a **different aspect** of the system and only those models that provide **value** to their intended audience should be used.

CONT…

- **Travel light**
  - As software engineering work proceeds, keep only **those models that will provide long-term value**.
- **Content is more important than representation.**
- **Know the models and the tools you use to create them**
  - Understand the **strengths and weaknesses** of each **model** and the **tools** that are used to create it.
- **Adapt locally**
  - The modeling approach should be **adapted to the needs of the agile team**.