# Transition from Traditional Waterfall to Agile Software Development

- The Shift from prescriptive software development to the Agile software Development
- The Essence of Agile philosophy
- A brief overview of popular agile methods namely, Scrum, Extreme Programming and Feature Driven Development (FDD) followed by a comparative analysis

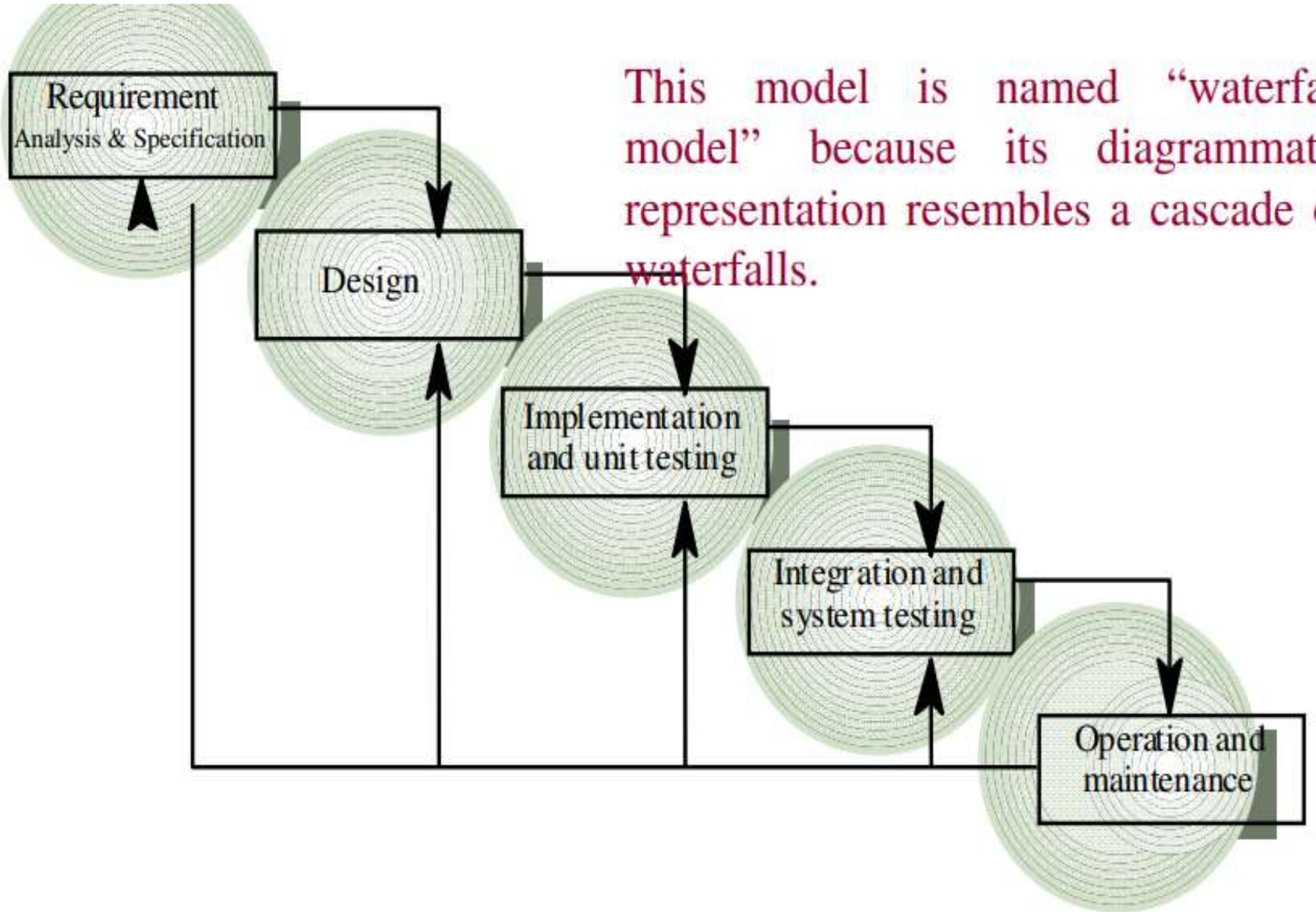
# Shift From Prescriptive Software Development To The Agile Software Development

# Shortcomings of Waterfall Model

- Deceptively simply “requirements box” of the waterfall model
- An ineffective treatment of requirements led to “challenged” Projects
  - Lack of User input: 13% of all Projects
  - Requirements and specifications: 12% of all projects
  - Changing Requirements and Specifications 12% of all projects

Source: Leffingwell, “Agile Software Requirements, Lean Requirements Practices for Teams, Programs and he Enterprises”

# Waterfall model

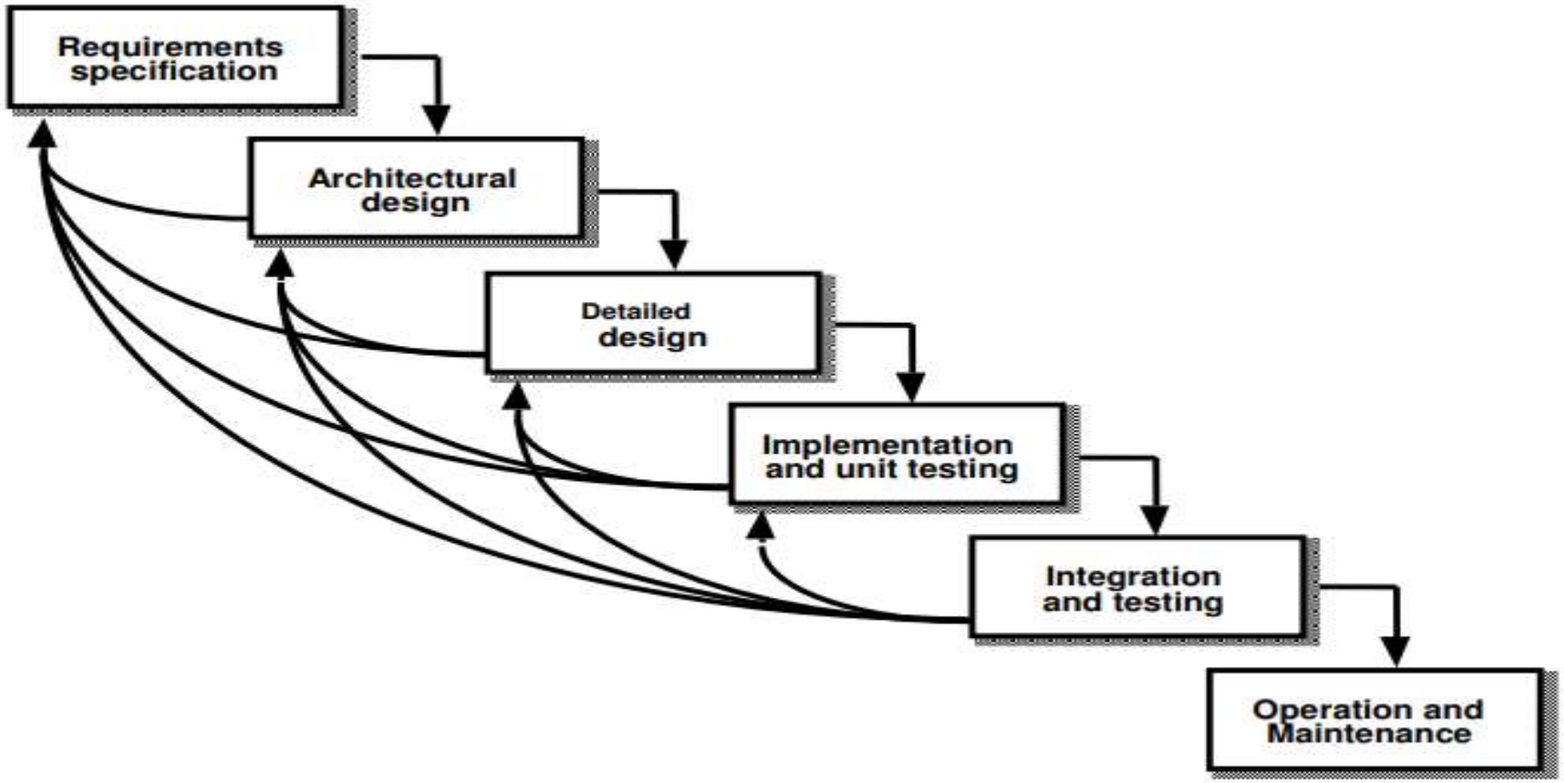



This model is named "waterfall model" because its diagrammatic representation resembles a cascade of waterfalls.

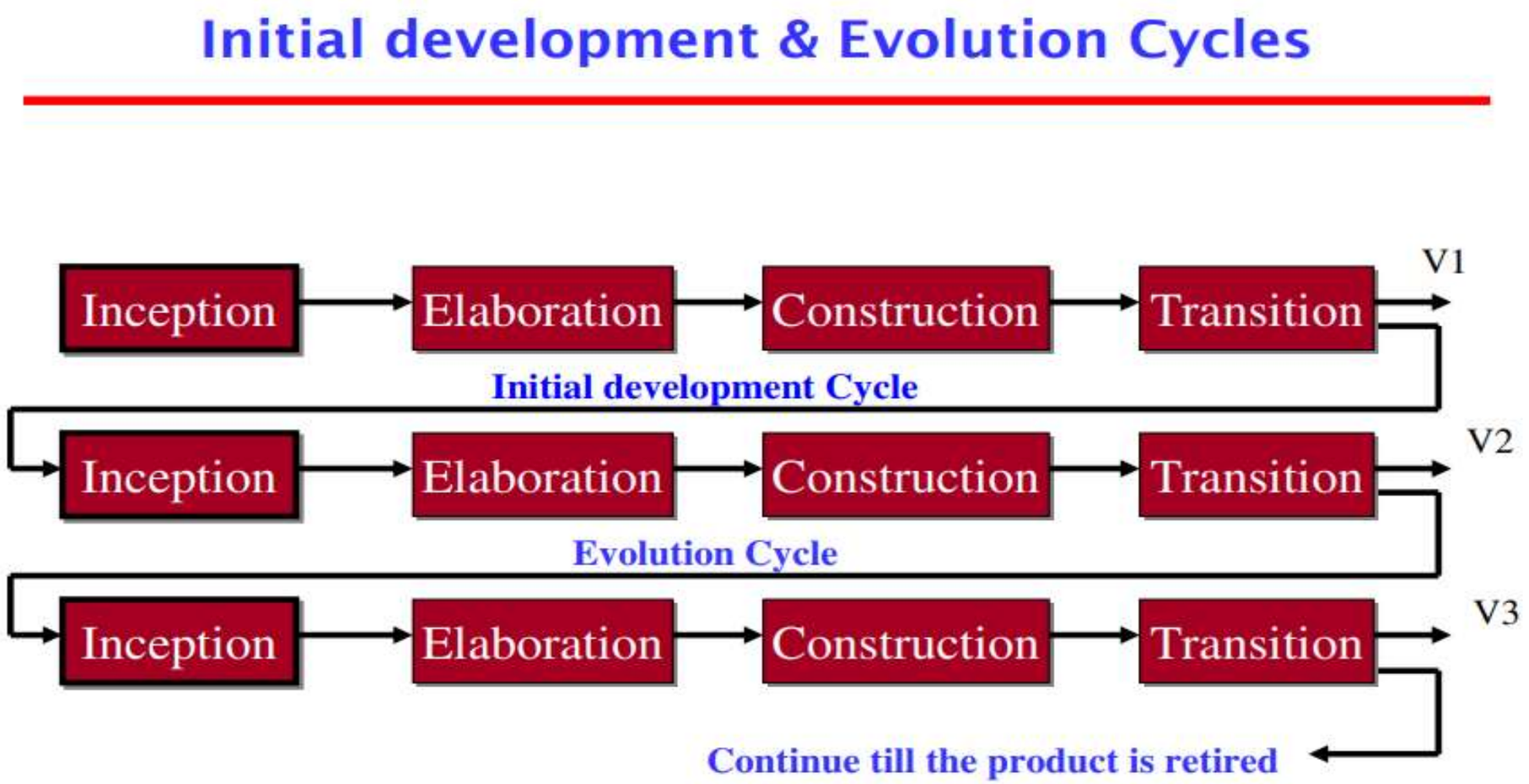
# Giant step forward for the industry

- **Emergence of Iterative and Incremental Approach in 1980s and 1990s**
  - **Rapid Development of understanding via experimental discovery (Spiral)**
  - **Rapid build of models, prototypes and initial systems using more advanced tools (RAD)**
  - **Iterative and incremental development of ever larger and more complex systems (RUP)**

# Iterative and Enhancement Model



# Rational unified process (RUP)



# Adaptive (Agile) Processes

- Starting in the late 1990s and through the current decade,
- An explosion of lighter-weight and ever-more-adaptive models
- Based on a different economic foundation
- more cost effective to write the code quickly, have it evaluated by customers in actual use, be “wrong” (if necessary), and quickly refactor it

# Agile Manifesto

- **Individual and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan

# Agile Principles

- Customer satisfaction by early and continuous delivery of valuable software.
- Welcome changing requirements, even in late development.
- Working software is delivered frequently (weeks rather than months).
- Close daily cooperation between business people and developers.
- Projects are built around motivated individuals, who should be trusted.

# Some Popular Agile Methodologies

- Scrum
- Extreme Programming (XP)
- Feature Driven Development(FDD)
- Lean/ Kanban

**The most widely adopted agile methods are Scrum and XP.**

# Agile Methods

1. SCRUM
2. Extreme Programming XP
3. Feature Driven Development FDD

# Framework

## Roles

- Product owner
- Scrum master
- Team

## Events

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

## Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

# Key Practices

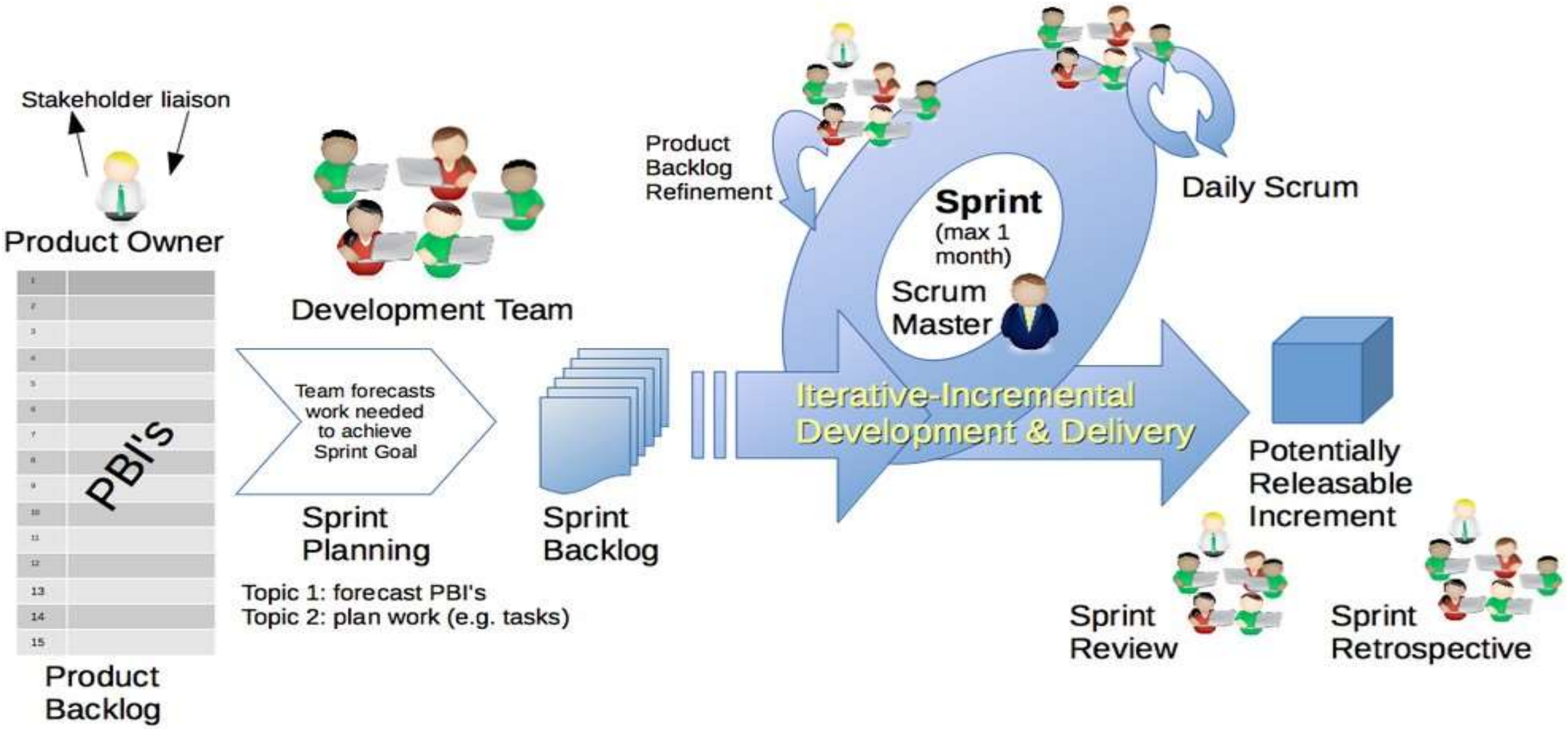
- Work is done in “sprints” which are time boxed durations of a fixed 30 days or fewer duration
- Work within a sprint is fixed
- All work to be done is characterized by a product backlog, which includes new requirements to be delivered, the defect workload and the infrastructure and design activities.

# Key Practices

- A scrum master mentors the empowered self organizing and self accountable teams that are responsible for delivery of successful outcomes at each sprint.
- Product Owner plays the role of customer proxy
- A daily Stand-up meeting is a primary communication method

As . (role), want (feature)

# Workflow


Stakeholder liaison
Product Owner
Development Team
Product Backlog Refinement
Sprint (max 1 month)
Scrum Master
Daily Scrum
PBI's
Team forecasts work needed to achieve Sprint Goal
Sprint Planning
Sprint Backlog
Iterative-Incremental Development & Delivery
Potentially Releasable Increment
Topic 1: forecast PBI's
Topic 2: plan work (e.g. tasks)
Product Backlog
Sprint Review
Sprint Retrospective

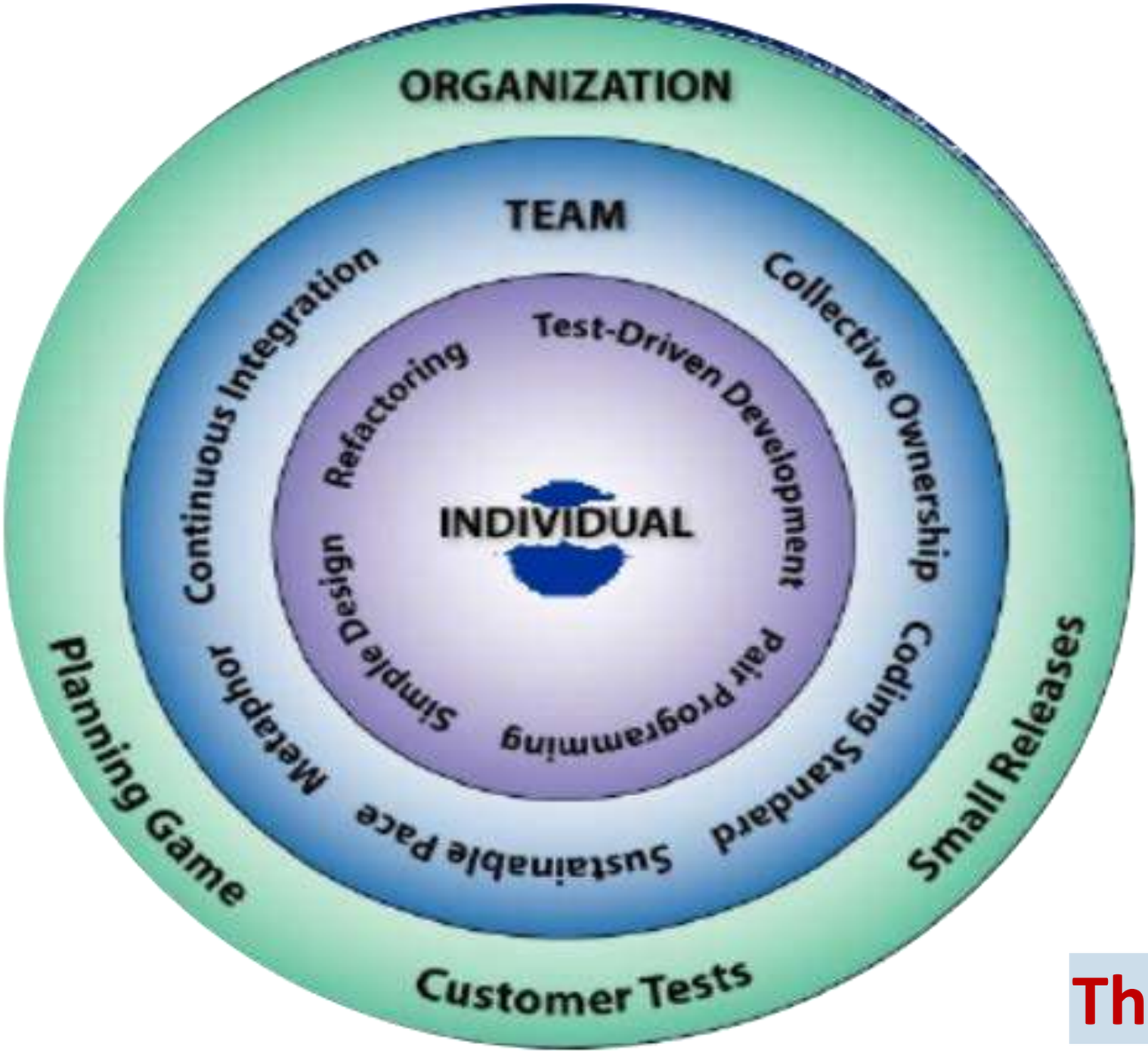
# Agile Methods

1. SCRUM
2. Extreme Programming XP
3. Feature Driven Development FDD

# Extreme Programming (XP)

- A team of eight to ten members work at one location with customer representative on-site
- Development occurs in frequent builds or iterations , which may or may not be releasable
- Requirements are specified as user stories, each a chunk of new functionality the user requires
- Programmers work in pairs, follow strict coding standards and do their own unit testing, customers participate in acceptance testing.

- XP, originally described by Kent Beck.




# The 12 key Practices

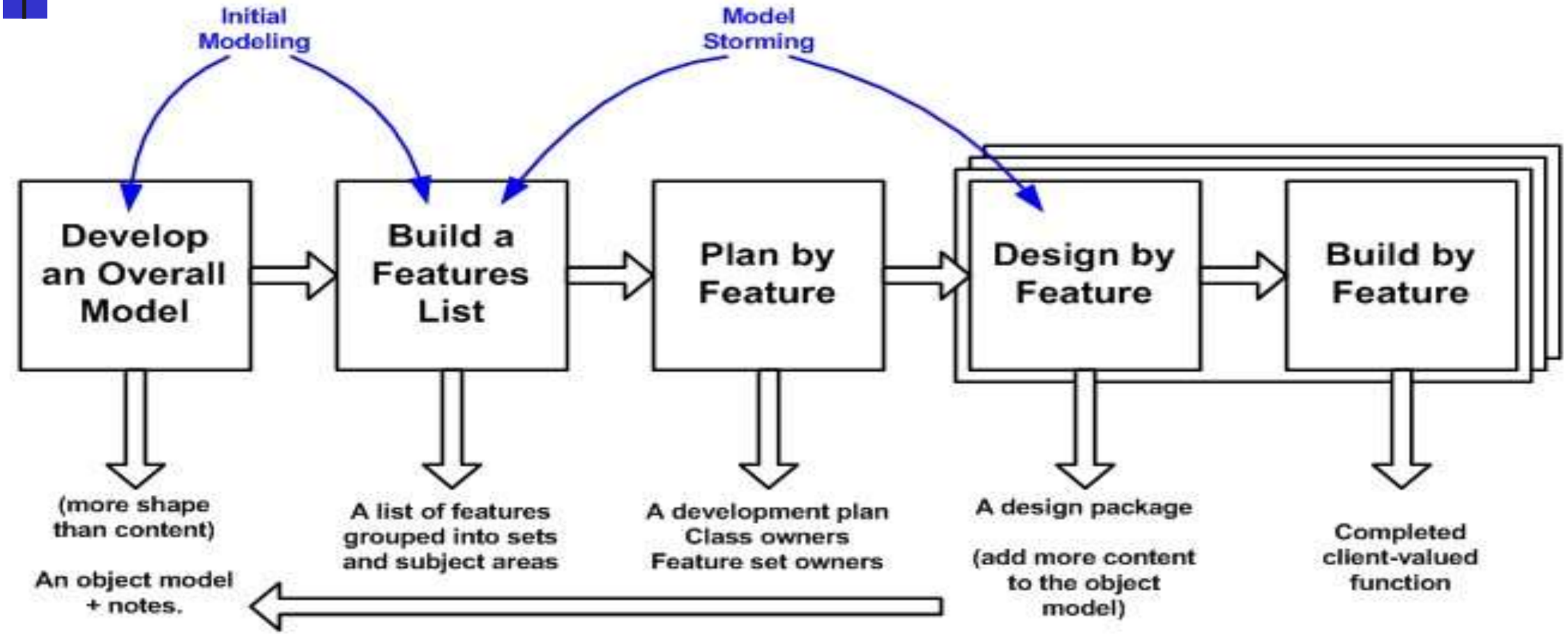
# Agile Methods

1. SCRUM
2. Extreme Programming XP
3. **Feature Driven Development FDD**

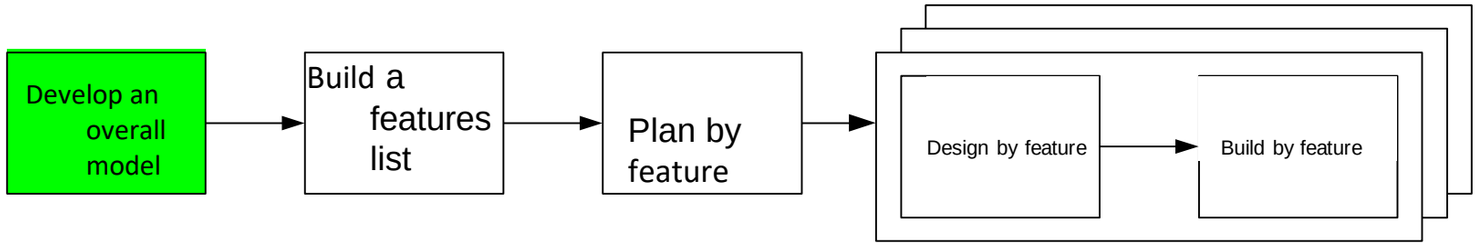
# Feature Driven Development Process

- Process #1: Develop an Overall Model
- Process #2: Build a Features List
- Process #3: Plan By Feature
- Process #4: Design By Feature
- Process #5: Build By Feature

# Feature Driven Development Process



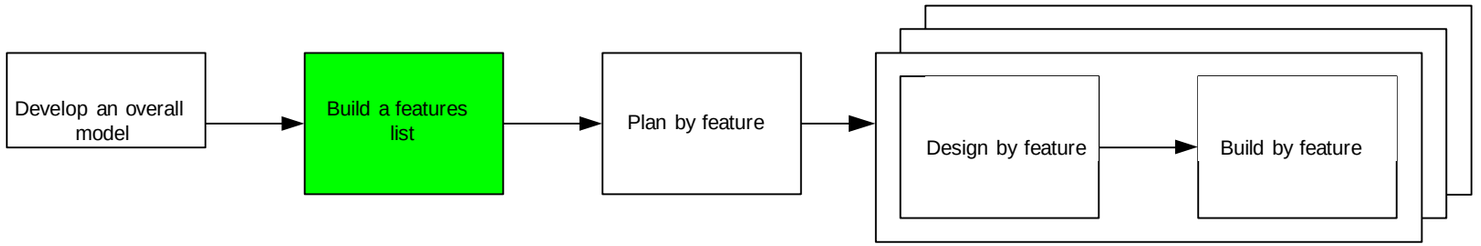
# Process #1: Develop an Overall Model




**Who? Domain experts, chief architect, chief programmers.**

- Establishes the shape of the system
- Defines classes, how classes related to each other
- Creates the base object model
- Includes internal and external reviews, model notes
- Goal - for team members to gain a good, shared understanding of the problem domain and build a foundation

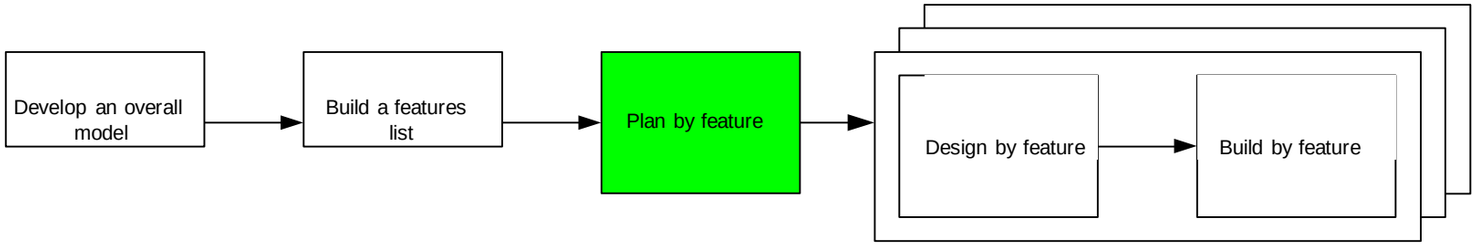
# Process #2: Build a Features List




*Who? Feature List Team: domain experts, chief programmers, chief architect*

- ***Functional decomposition of model developed in step 1***
- ***Subject area to business activity to business activity step***
- ***Feature is a business activity step, customer centric not technology centric***
- ***Nomenclature: <action> <result> <object>***
- ***"Generate an account number for the new customer***

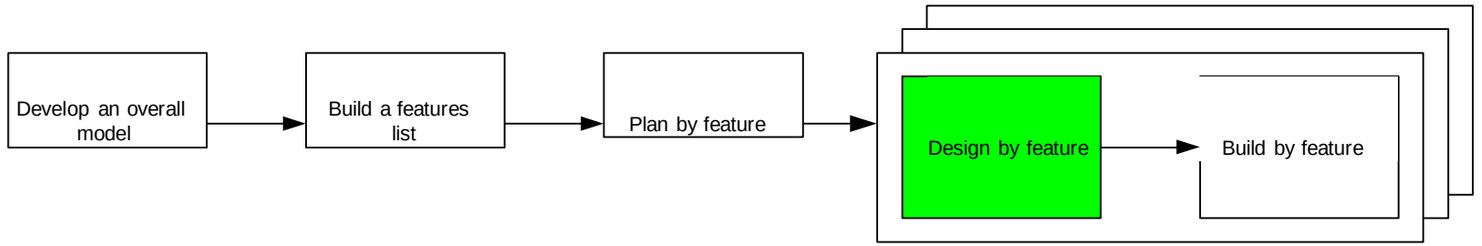
# Process #3: Plan By Feature




**Who? The Planning Team: the project manager, the development manager, and chief programmers.**

- Group features into feature sets (one or more business activities)
- Prioritize based on customer need
- Establish completion dates (MM/YYYY)
- Assign responsibilities to team members
  - Determine Class Owners
  - Assign feature sets to chief programmers

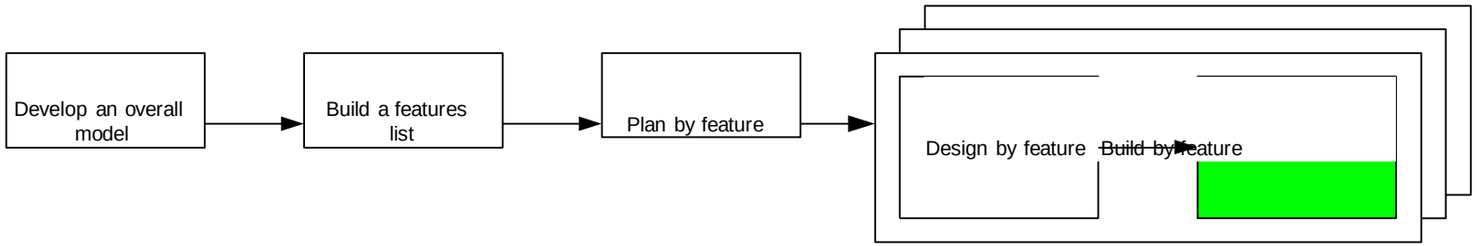
# Process #4: Design By Feature




**Who? The Feature Team: chief programmer, class owners**

- Form Feature Teams
- Team members collaborate on the full low level analysis and design
- Certain features may require teams to bring in domain experts
- Teams need to update the model artifact to support their changes
- Fleshes out class and object design, create sequence diagrams as necessary

# Process #5: Build By Feature




**Who? Class owners, chief programmers**

- Implement
- Code inspection
- Unit test
- Promote to build

# Comparison Between Different Agile Methods

| Characteristics | XP | Scrum | FDD |
|---|---|---|---|
| Approach | Iterative increments | Iterative increments | Iterative |
| Time Period for Iteration | 1 to *6* weeks | 2 to 4 week s | 2 days to 2 weeks |
| Team Size | small teams (<10 members) | All Sizes | Many members and more than one team |
| Suitable Project Size | For smaller Project | all types o f projects | m ore complex projects |
| User involvement | user highly involved | Involvement through the product owner | Involvement through rep ort s |
| Documentation | only basic documentation | only basic documentation | Documentation is importan t |
| Major Practices | User Stories, Test driven development, Refactoring, Pair Programming, strict practices for coding | Sprint, Product and sprint Backlog, Scrum meetings | UML Diagrams |

# Thanks

# COMPARATIVE STUDY

| Characteristics | XP | Scum | DSDM | FDD |
|---|---|---|---|---|
| Approach | Iterative increments | Iterative increments | Iterative | Iterative |
| Time Period for Iteration | 1 to 6 weeks | 2 to 4 weeks | 80% solution in 20 % total time | 2 days to 2 week |
| Team Size | small teams (<20 members) | All Sizes | all sizes independent teams | many members and more than one team |
| Suitable Project Size | For smaller Project | all types of projects | all types of projects | more complex projects |
| User involvement | user highly involved | involvement through the product owner | involvement through frequent releases | involvement through reports |
| Documentation | only basic documentation | only basic documentation | Documentation exist | Documentation is important |
| Major Practices | User Stories, Test driven development, Refactoring, Pair Programming | Sprint, Product and sprint Backlog, Scrum meetings | Prototyping, feasibility and business study | UML Diagrams |

# CASE STUDY

- **The case study was performed in controlled environment – in a course called Application Development—a four credit hours course at UTM.**
- **The course began by splitting up the class to seven software development groups and two groups were chosen to implement the existing process of FDD.**

**Two basic goals of this experiment of case student were**

**1 To introduce students to a variety of software processes in Agile.**

**2 To get students to adapt security in their system in FDD manners.**

- **We wanted to investigate that by adding security elements in the system while applying FDD processes, whether it was possible to handle changes**

# Introduction

- In the case study, the students who did not have any experience or skills in implementing security element in the system, managed to learn new skill in a semester.
- Later we shall see, how the student were assigned tasks on managing the system development process in addition security elements that have been added as late requirements.
- The main focus shall be on how the course gets students to think about how they
  manage when new security requirement when they are in
  FDD process in developing their system.

# Overview

- The course began by introducing a variety of software processes in Agile to students. Then we chose two groups to apply FDD while other 5 groups applied other Agile
  software process such as Scrum and XP.
- After that, a two week introduction to FDD was conducted. Then, the students started the documentation as advised. After that the course continued with implementing the design.
- While the students were busy doing the project, the lecturer suddenly changed the requirements during second month.
- The tasks list was to implement security element like SQL injection, encryption, session management and others that relate towards their system.
- At the end the students compared the estimated date of completion of the system with the previous date of completion set during the first week of course. They had to check whether they overrun the dateline or not and if they overrun the dateline, what were the reasons that they overboard the dateline

# IMPLEMENTING FDD

- **As the class project started, the students were divided to seven groups. Two groups were given the task of completing their system project using FDD model and there were guided with all the phases of FDD**

**(A) Develop an Overall Model Phase**

In this phase the students were familiarized with this agile model. They needed to perform their requirement and made it into a overall model for the whole system where they did using Enterprise Architecture (EA) tool. EA tool helped them create a whole model to depict the whole modules of the system. Both team managed to come out with 5 modules.

# IMPLEMENTING FDD

**(B) Build a Feature List**

- This is the next phase after finishing the overall model for the system. They need to identify the features that listed under the module.
- For instance, the system has a module of Quota Package. Therefore the features are Apply Quota, View List, Approve Quota, Reject Quota, Reject Quota and Update Quota List.
  The task is that one group must have at least 20 features in their project.
- This phase also have been done using EA tool where each features are noted in use case diagram.

# IMPLEMENTING FDD

**(C) Plan by Feature**

- This third phase is the most crucial phase where all the planning of the project started here. The student must really need to prepare the planning documentation for each module that they have created. Each module must has estimation time to be completed.
- As far as concern, feature sets with completion dates, Chief Programmers assigned to feature sets, a list of classes and the developers that own them must be listed in their planning documentations. However each team has different kind of planning documentation. One of the group must do the planning using gantt chart where else the other group used mind maps.
- The main reason why this method was used to determine whether by using gantt chart or by using mind map could help to see the overall progress of the development better.

# IMPLEMENTING FDD

**(D) Design by Feature**

- This phase required the students to be more precise on how they going to design their system by feature. Each feature must have a sequence diagram and class diagram to picture their system looks like.
- This phase is also important to show to their customer how the system operates so that any confusion or disagreement could be settled before the developers starting developing the system.

**(E) Build by Feature**

- This phase is the last phase where the technical part comes in action. After a thorough check on the design, they started developing the system by module one after another.
- The lecturer has appointed one to two weeks to complete one module. Even though both teams were using FDD, but they were using different programming language and tools to complete their system.
- While completing the system they will also update their gantt chart or their mind maps.

# IMPLEMENTING FDD

**RECOMMENDATIONS**

- **This is the part where the students that have complete their phases in the FDD to voice out their opinion about the weaknesses that they find in the existing FDD model to adapt with security feature.**

- **A set of questionnaire have been distributed to the FDD members to ask them about their opinion throughout the implementation of system. The questions that been asked are as follows:**

1) Through your experience in FDD process development, explain briefly weaknesses in FDD modelling.

1) When the lecturer asked you to add security features/elements inside your system, do you

# IMPLEMENTING

TABLE I: FEEDBACKS FROM QUESTION 1

| Feedbacks | No. of Students |
|---|---|
| FDD have less iteration between the phases and stakeholders have the right to add more requirements while the implementation in progress | 1 |
| Make sure everything run smoothly; the entire team member must have all the skills. | 2 |
| There are not specific roles for each team members | 2 |
| There are too much feature that hold by different member that will be very hard to synchronize between the features | 2 |
| That is hard to update the documentation until the final stage | 3 |
| Lacking in communication between the class owner. | 3 |
| Could not handle sudden changes and it will consume time | 5 |
| FDD requires too much documentation before the real coding starts | 7 |

# IMPLEMENTING

TABLE II: FEEDBACKS FROM QUESTION

| Feedbacks | No. of Students |
|---|---|
| No | |
| The security feature is necessary to be considered as part of the requirements. | 2 |
| Yes | |
| Consume time since they are not prepared the right tool that they are using can be easily create the security features | 1 |
| They need time to learn how to implement the security feature inside the system | 1 |
| It will slow down since they need time to adapt the changes of the addition on security features | 2 |
| Deviate from the original schedule. | 3 |

# IMPLEMENTING

TABLE III: FEEDBACKS FROM QUESTION 3

| Feedbacks | No. of Students |
|---|---|
| Consider security features as the functional requirement that must be treated fairly like other requirements of the system | 1 |
| Lessen the unnecessary documentations | 1 |
| Any additional changes on the system especially in security feature must be made parallel with the main system development progress. | 1 |
| The team must always report to each other to keep track the progress among each other | 2 |
| Must use or specify tools that can be used to create any security features | 2 |
| Add security specialist among the roles in FDD | 2 |
| Expose the team members with the security knowledge | 3 |
| Any requirements that included security features must be clearly stated | 4 |
| Must be stated in early stage of the development process | 4 |
| The existing FDD modeling must add more iteration between the phases. | 5 |

# IMPLEMENTING

TABLE IV: FEEDBACKS FROM QUESTION 4

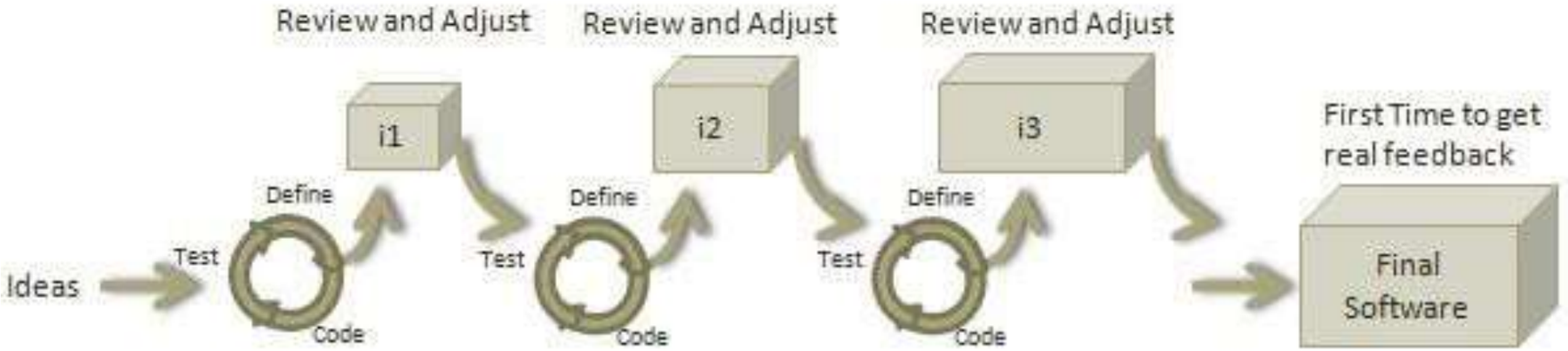
| Feedbacks | No. of Students |
|---|---|
| Mind map is better than gantt chart because gantt chart: | 7 |
| Do not show individual progress | |
| Do not show the detail of the whole project in one shot | |
| Does not show relation between the task | |
| Does not show the whole progress of the development | |
| Hard to understand the progress | |
| Only focus on duration of the task and dateline | |
| Need to weekly update the progress | |
| Only show the progress in parallel | |
| Gantt chart is better that mind map because mind map: | 1 |
| Do not specify the date | |
| Day and time and have too much information in one short that need time to understand the whole picture. | |

# CONCLUSION AND FUTURE WORKS

- Based on our observations, we can say that it is possible that FDD could manage well with security of security of software if this model is carefully analyzed and implemented with some of the security practices, plan the right tool, add more iterations between the phases of FDD and use mind map tool, to get a clearer view of the whole progress in the planning phase.
- We noticed that, even if the students that were still new in developing a secure system and never heard off or familiarized with the secure software development could pull off with little difficulty. However, we noticed that FDD needs to be remodeled to clearly define the process

# TRADITIONAL VS AGILE


**Traditional Method**



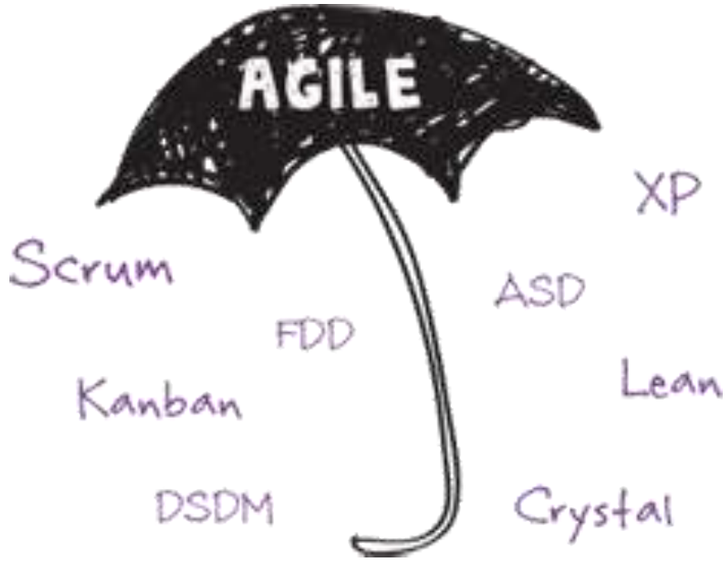

**Agile Method**

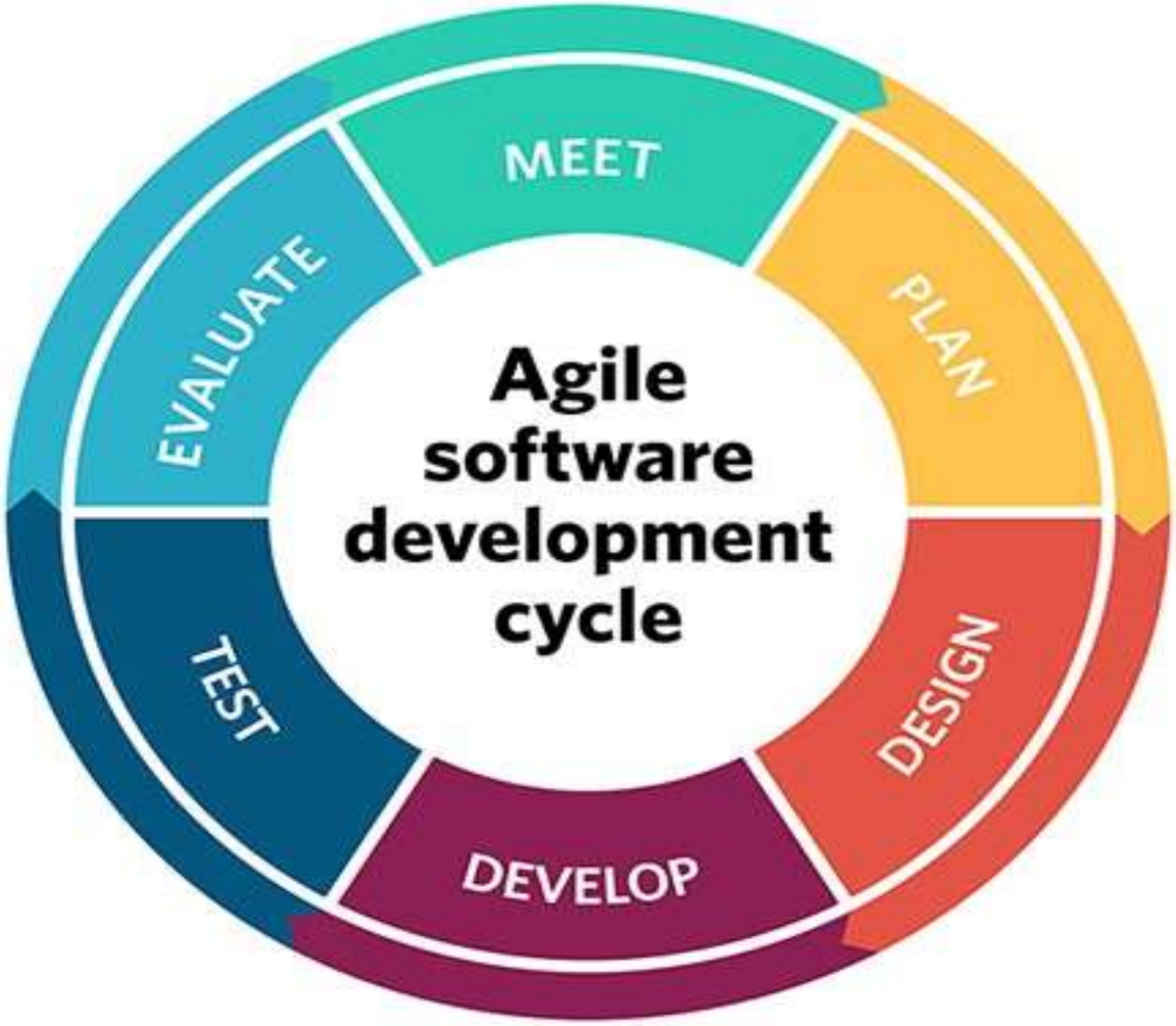
# TRADITIONAL VS AGILE APPROACH

| Traditional Method | Agile Method |
|---|---|
| Sequentially from start point to end point | Incremental and iterative approach |
| Design process is not broken into an individual models | Broken into individual models that designers work on |
| Customer can only see the product at the end of the project | Make early decision and changes to the project |
| Only at the end, the whole product is tested. Start again if error found in middle | Error can be fixed in the middle of the project. |
| Development process is phased. | Project is executed in short (2-4) weeks iterations. |
| Testers work separately from developers | Testers and developers work together |
| Documentation is a top priority | Documentation attends less priority than software development |

# WHAT IS AGILE SOFTWARE DEVELOPMENT METHODOLOGY?

- An incremental, iterative approach to software development
- These methods are open to changes in requirements over time and encourage constant feedback from customers.




Agile software development cycle
MEET
PLAN
DESIGN
DEVELOP
TEST
EVALUATE

# 12 PRINCIPLES

- ***Customer satisfaction***
- ***Welcome Change***
- ***Deliver a Working Software***
- ***Collaboration***
- ***Motivation***
- ***Face-to-face Conversation***
- ***Measure the Progress as per the Working Software***
- ***Maintain Constant Pace***
- ***Monitoring***
- ***Simplicity***
- ***Self-organized Teams***
- ***Review the Work Regularly***

# 4 VALUES

- **Individuals and Interactions more than processes and tools**

**the team is more important than the tools**

- **Working Software more than comprehensive documentation**

**It is vital that the application works**

- **Customer Collaboration more than contract**

**negotiation  The client should be involved in development**

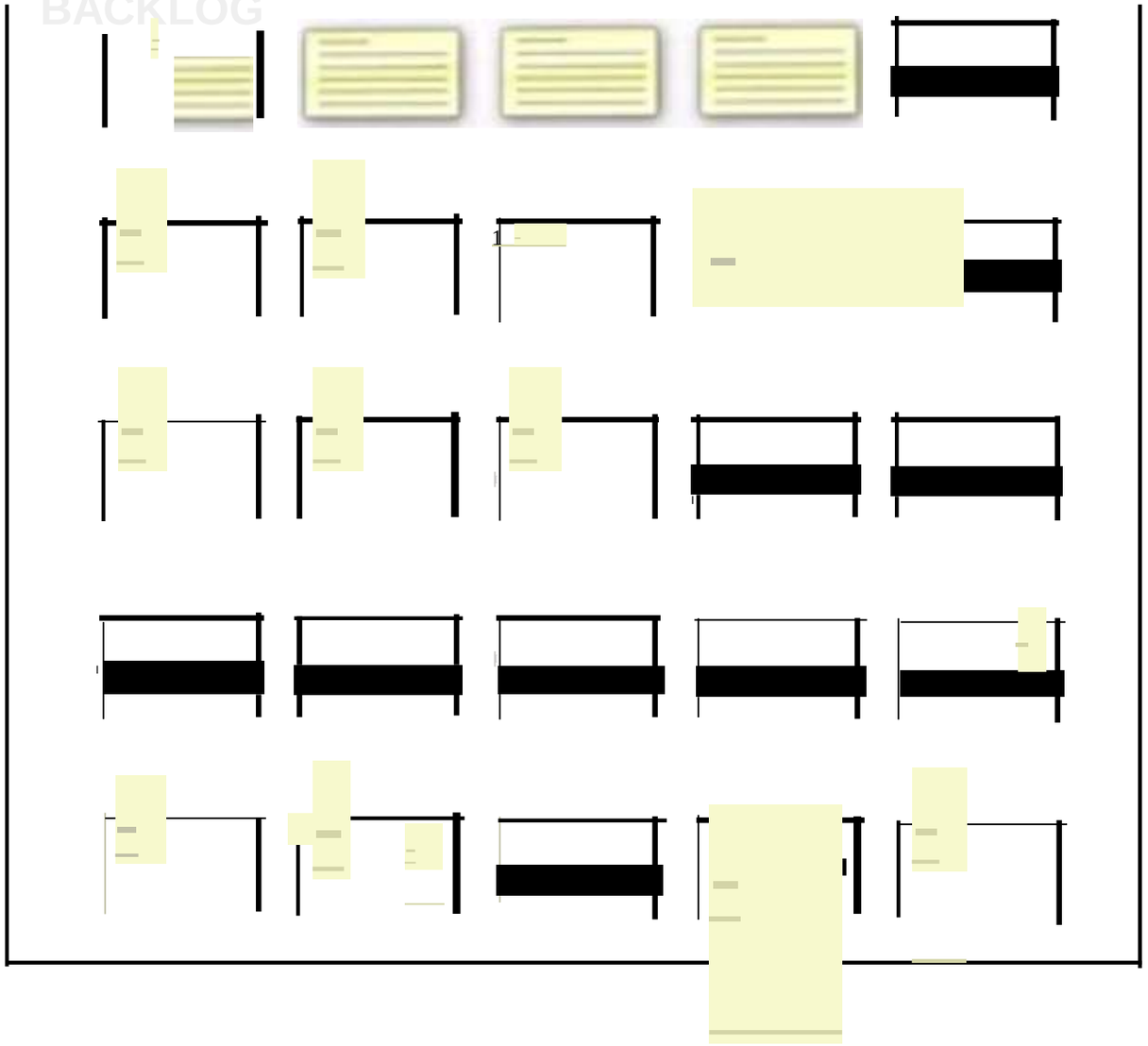
- **Responding to Change more than following a plan**

**Initial planning and structure of the software must be flexible to allow changes**

# Why scrum?

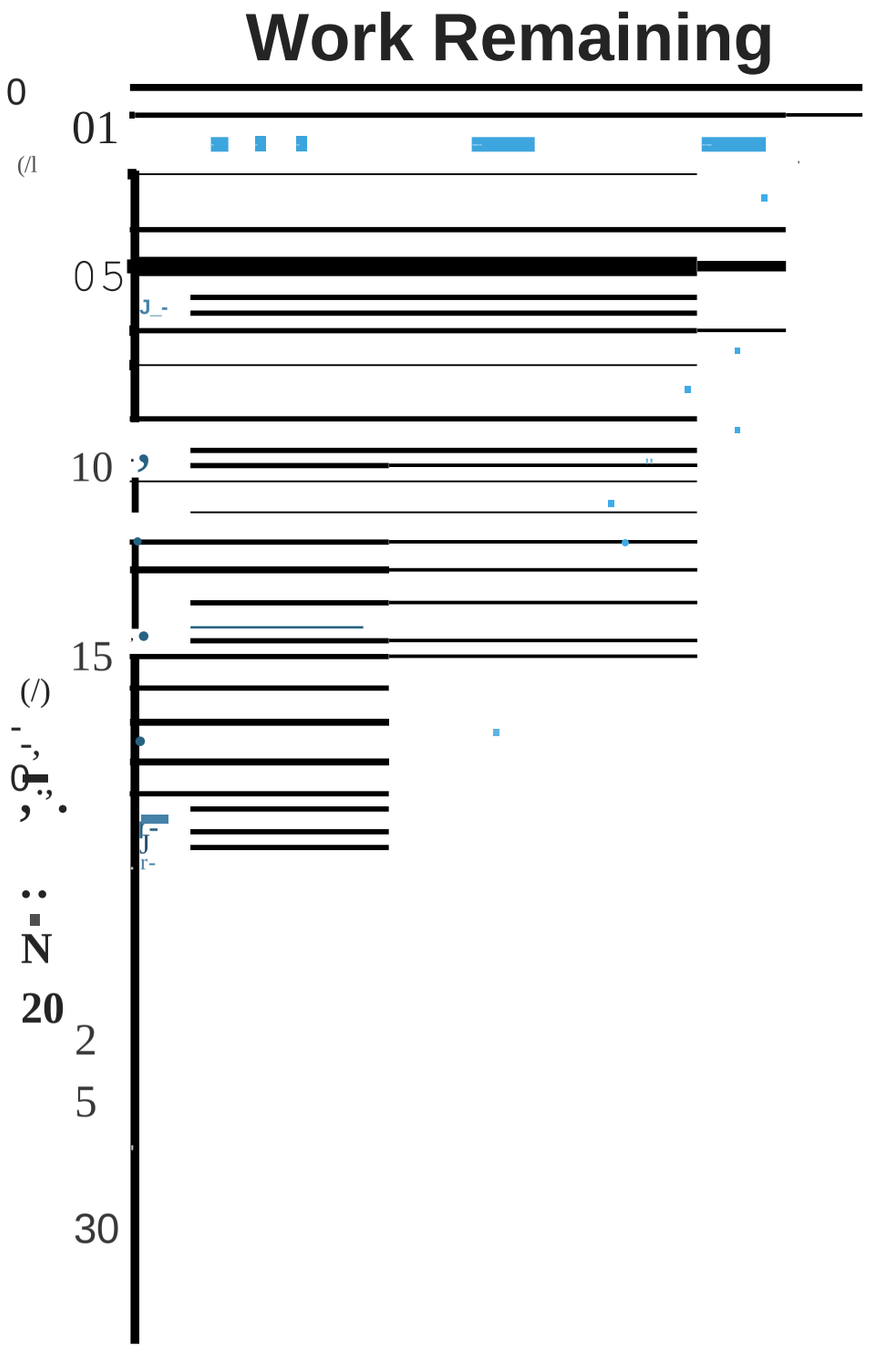
- Welcome changing requirements, even late in development
- Deliver potential working software early
- Early visibility of business
- Easy to keep track of project progress
- Self organizing teams

# PRODUCT BACKLOG



## PRODUCT BACKLOG

# Sprint Backlog


Work Remaining
0
01
05
10
15
20
2
5
30

# Twelve Supporting Practices

- Planning Game
- Small Releases
- Customer Acceptance
- Tests  Simple Design
- Pair Programming
- Test-Driven
- Development  Re-
- factoring
- Continuous Integration
- Collective Code Ownership
- Coding Standards
- Metaphor

Sustainable Pace

# Analysis & Design

The main goal of the analysis & design discipline is to show how the system will be realized in the implementation phase.

**The purpose of analysis and design is as follows:**

- **To translate the requirements into a specification that describes how to implement the system**
- **To establish robust architecture so that you can design a system that is easy to understand, build, and evolve**

# How Do I Start This XP Thing?

- Start out collecting user stories and conducting spike solutions for things that seem risky. Spend only a few weeks doing this.
- Then schedule a release planning meeting. Invite customers, developers, and managers to create a schedule that everyone agrees on.
- Begin your iterative development with an iteration planning meeting.
- Now you're started.

# Usual Type Of Projects XP Receive

- Usually projects come looking for a new methodology like XP only after the project is in trouble.

- In this case the best way to start XP is to take a good long look at your current software methodology and figure out what is slowing you down. Add XP to this problem first.

# When should Extreme Programming be Used?

- Can be used in the problems domains where requirements change.
- XP was also set up to address the problems of project risk.
- The XP practices are set up to mitigate the risk and increase the likelihood of success.
- XP projects unanimously report greater programmer productivity when compared to other projects within the same corporate environment. But this was never a goal of the XP methodology. The real goal has always been to deliver the software that is needed when it is needed. If this is what is important to your project it may be time to try XP.

# EXTREME PROGRAMMING (XP)

# What is scrum?

- Agile way of Project Management
- A team based collaborative approach
- Iterative & incremental development
- Always focus to deliver business value

# Meetings

## The sprint review

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Whole team participates

## Sprint retrospective

- Periodically take a look at what is and is not working
- Typically around 30 minutes
- Done after every sprint
- Whole team participates including Scrum Master, Product Owner, Team and possibly customers

# Extreme Programming

**It is based on four simple values**

- **Simplicity,**
- **Communication,**
- **Feedback**
- **Twelve supporting practices.**

# Criticisms Faced By The Methodology

- A methodology is only as effective as the people involved, XP can't solve it.
- Lack of structure and necessary documentation.
- Requires too much cultural change to adopt.
- Can be very inefficient; if the requirements for one area of code change through various iterations, the same programming may need to be done several times over.

# FDD HISTORY

- FDD was first introduced to the world in 1999 via the book Java Modeling In Color with UML, a combination of the software process followed by Jeff DeLuca's company and Peter Coad's concept of features

- FDD was first applied on a 15 month, 50-person project for a large Singapore bank in 1997

# SCRUM *V/S* FEATURE DRIVEN DEVELOPMENT

1. Scrum does not specify any particular engineering practice but feature driven development specify specific engineering practices i.e design/code inspection, tests.

1. Scrum has shorter feedback loops, but feature driven development has longer feedback loop.

1. Scrum has self-organizing teams but feature driven development team have recognized roles i.e project manager, chief architect, development manager, chief programmer, class owner and domain expert.

# PROJECT SCOPE AND PLAN

- The project scope and plan is simply and quickly created by manipulating the cards by hand.
- There are three levels of plans. One which looks a few months into the future and groups stories into larger deployments. Another plan for stories in the next iteration. And finally a break down of stories into tasks with developer sign up for the current iteration.
- Plans are considered temporary artifacts in XP.
- You will be expected to re-create your plans before they expire. Every time the customer gains insight you will make a new plan. Every time you slip or get ahead of your schedule you will make a new plan.

# User Participation

- In XP, the “Customer” works very closely with the development team to define and prioritize granular units of functionality referred to as “User Stories”.

# Roles

## Product Owner

- Define the features of the product
- Makes scope vs. schedule decisions
- Responsible for achieving financial goals of the project
- Prioritize the product backlog
- Adjust features and priority every sprint, as needed
- Accept or reject work results

## Scrum Master

- Coaches the team to their best possible performance
- Helps improve team productivity in any way possible
- Enable close cooperation across all roles and functions
- Shield the team from external interference

# The Team

- Typically 5-9 people
- Cross-functional
- Programmer, tester, designers, etc.
- Members should be full-time
- Teams are self-organizing
- Membership should change only between sprints

# Product Backlog

**It is the list of all the features(user stories) that we wish to include in our product. It consists of**

- **Features**
- **Bug fix**
- **Non functional requirements and many more**

**It is the role of product owner to prioritize product backlog**

# Sprint Planning

**At the beginning of a sprint, the scrum team holds a sprint planning event to:**

- **Communicate the scope of work that is intended to be done during that sprint**
- **Select product backlog items that can be completed in one sprint**
- **Prepare a sprint backlog that includes the work needed to complete the selected product backlog items**

# Burndown Charts

- The sprint burn-down chart is a public displayed chart showing remaining work in the sprint backlog.
- Updated every day, it gives a simple view of the sprint progress. It also provides quick visualizations for reference.
- The horizontal axis of the sprint burn-down chart shows the days in a sprint, while the vertical axis shows the amount of work remaining each day.
- During sprint planning the ideal burndown chart is plotted. Then, during the sprint, each member picks up tasks from the sprint backlog and works on them.
- At the end of the day, they update the remaining hours for tasks to be completed. In such a way, the actual burndown chart is updated day by day.

# Sprint Backlog

- The sprint backlog is the list of work the development team must address during the next sprint.
- The list is derived by the scrum team progressively selecting product backlog items in priority order from the top of the product backlog until they feel they have enough work to fill the sprint.
- The sprint backlog is the property of the development team, and all included estimates are provided by the development team.

# Meetings

## The Daily Scrum

- It is a stand up meeting that  happens daily for 15 minutes
- All members are invited
- Only team members, Scrum Master,  product owner, can talk
- Helps avoid other  unnecessary  meetings


## These 3 main questions are answered

- What did you do yesterday?
- What will you do today?
- Is anything in your way?

# FEATURE-DRIVEN DEVELOPMENT

- FDD is a client-centric, architecture-centric, and pragmatic software process.
- Model-driven, short-iteration process.
- Lightweight Agile method for developing software.
- Practices are all driven from a client-valued functionality (feature) perspective
- This method is focused around "designing & building" features
- Main purpose is to deliver tangible, working software repeatedly in a timely manner

# WHAT IS FEATURE ?

- As the name implies, Features are an important aspect of FDD.
- A feature is a small, client-valued function
- Features are to be “small” in the sense they will take no more than two weeks to complete
- Features that appear to take longer are to be broken up into a set of smaller
- features
- Feature naming template:

<action> the <result> <by|for|of|to> a(n) <object>

Examples: Calculate the total of a sale

Validate the password of a user

# FEATURE DRIVEN DEVELOPMENT

- 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, “useful in the eyes of the client” results.
- The advantage of using FDD is that it is scalable even to large teams due to the concept of ‘just enough design initially’ (JEDI)
- It is used in enterprise projects as well as web projects

# SIX PRIMARY ROLES

- Project Manager- Administrative lead, Operates project system(eg. Togethersoft control center) Shields participants from external distractions
- Chief Architect- Responsible for the overall design, Runs design workshops, Steers project through technical obstacles.
- Development Manager-Leads day to day development activities, Resolves resource conflicts
- Chief Programmers- Experienced developers, Leads smaller teams of individual developers
- Class Owners- Individual developers, Design, code, test and document features
- Domain Experts- Users, clients, sponsors, etc., Knowledge base for developers

# CLASS OWNERSHIP

- Class assigned to specific developer.
- Class owner responsible for all changes in implementing new features
- An individual will take on one or more roles on a project.

# Reporting

- FDD emphasizes the ability to provide accurate, meaningful, and timely progress information to all stakeholders within and outside the project.
- For accurate state reporting and keeping track of the software development project it is however important to mark the progress made on each feature
- The first three milestones are completed during the Design By Feature activity, the last three are completed during the Build By Feature activity

| Domain Walkthrough | Design | Design Inspection | Code | Code Inspection | Promote To Build |
|---|---|---|---|---|---|
| 1% | 40% | 3% | 45% | 10% | 1% |

# BEST PRACTICES IN FDD

- Domain object modelling, the creation of a high-level class diagram and supporting artifacts that describes the problem domain.
- Developing by feature function that is too complex further decomposed into smaller functions until each sub-problem is small enough to be called a feature
- Individual class ownership distinct pieces or grouping of code are assigned to a single owner.
- Owner is responsible for the consistency, performance, and conceptual integrity of the class.
- Developers work together in feature teams.
- A feature team is a small, dynamically formed team that develops a small activity

# BEST PRACTICES IN FDD cont..

- Inspections are an important aspect of FDD to ensure good quality design and code, primarily by detection of defects.
- FDD also insists on regular builds, similar to XP, ensure there is always an up-to-date system that can be demonstrated to the client and helps highlighting integration errors of source code for the features early.
- Configuration management.
  - Configuration management helps with identifying the source code for all features that have been completed to date and to maintain a history of changes to classes as feature teams enhance them.
- Reporting/Visibility of results, similar to XP
  - By frequent, appropriate, and accurate progress reporting at all levels inside and outside the project, based on completed work, managers are helped at steering a project correctly.

# COMPARATIVE ANALYSIS OF AGILE SOTWARE METHODS

## SCRUM *V/S* EXTREME XP

1. Scrum does not recommend any of engineering customs but in XP engineering. Customs like-
    - ❖ test driven development;
    - ❖ refactoring;
    - ❖ pair programming;
    - ❖ automated testing are supported by XP.

1. In scrum , customer stories are accumulated in (i) product backlog (ii) sprint backlog. But XP stores the stories in “parking lot”.

# SCRUM vs FDD

| SCRUM | FDD |
| --- | --- |
| Does not specify any particular engineering practice | Specific engineering practices i.e. design/code inspections, tests |
| Focus on producing vertical slices of functionality acceptable to product owner | Domain driven |
| Shorter feedback loops | Longer feedback loop |
| Self-organizing teams | Feature team have recognized roles i.e. Project Manager, Chief Architect, Development Manager, Chief Programmer, Class Owner, and Domain Expert |
| Shared code ownership | Class ownership |