UNIT V

Trends in Software Engineering

Agile Management and Practices

Agile Practices: Agile manifesto, The Fundamentals of Agile Software Development, Aspects of Agile Approaches, Practices and Processes, Techniques in Agile Projects, Extreme Programming,

Devops: Introduction-Definition, Devops Tool chain, Why Devops?, Goals, Benefits, Relationship to Agile and Devops (continuous delivery), Devop Tools.

Role of Software engineering in IoT applications, data science applications, cloud computing and cyber security applications.

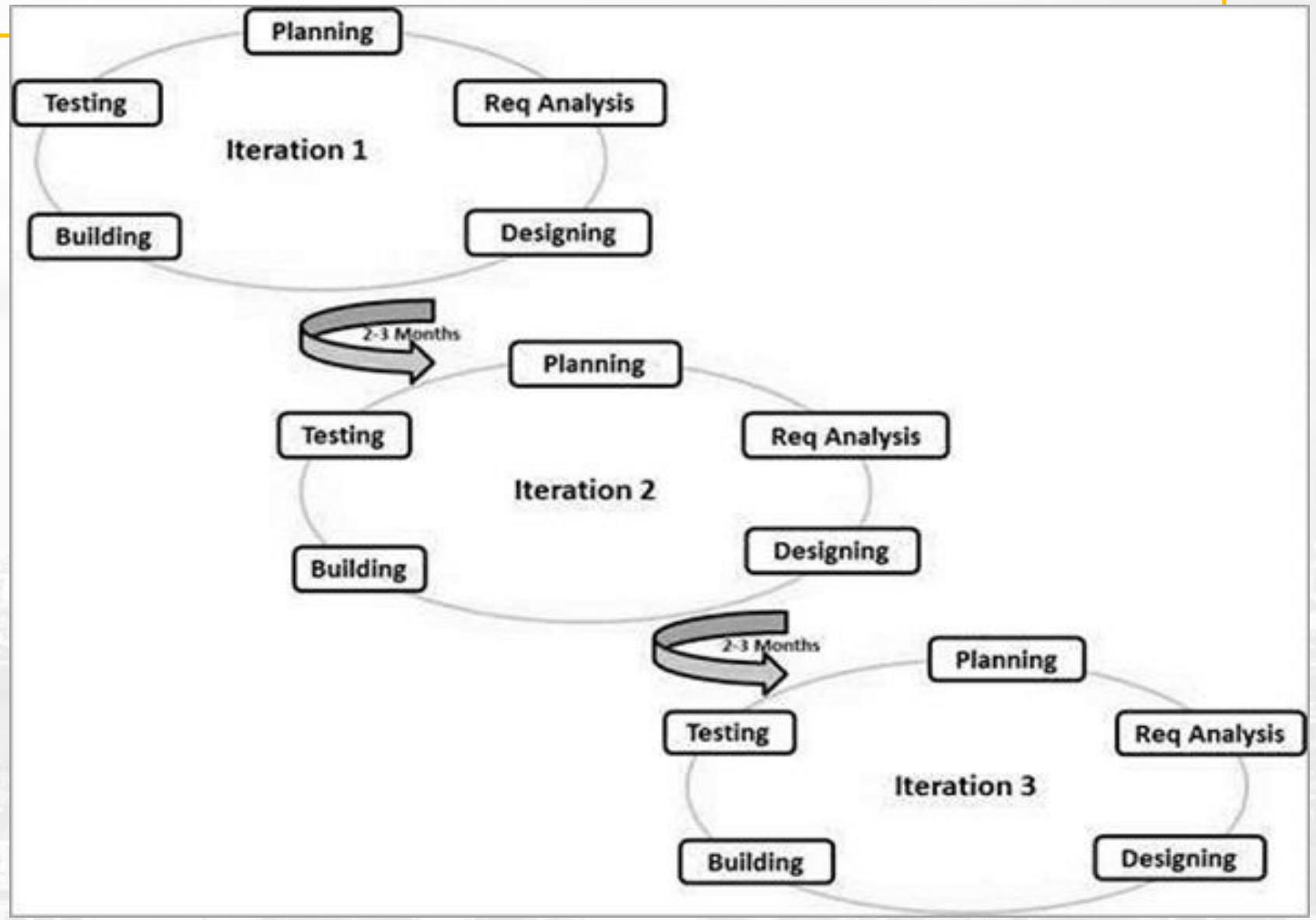
Agile Software Development

- Agile software development is a methodology that refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.
- It helps teams provide quick and unpredictable responses to the feedback they receive on their project.
- It creates opportunities to assess a project's direction during the development cycle. Teams assess the project in regular meetings called sprints or iterations.

Agile SDLC Model

Agile is both incremental and iterative. They are iterative in that they plan for the work of one iteration to be improved upon in subsequent iterations.

They are incremental because completed work is delivered throughout the project.



Waterfall vrs Agile

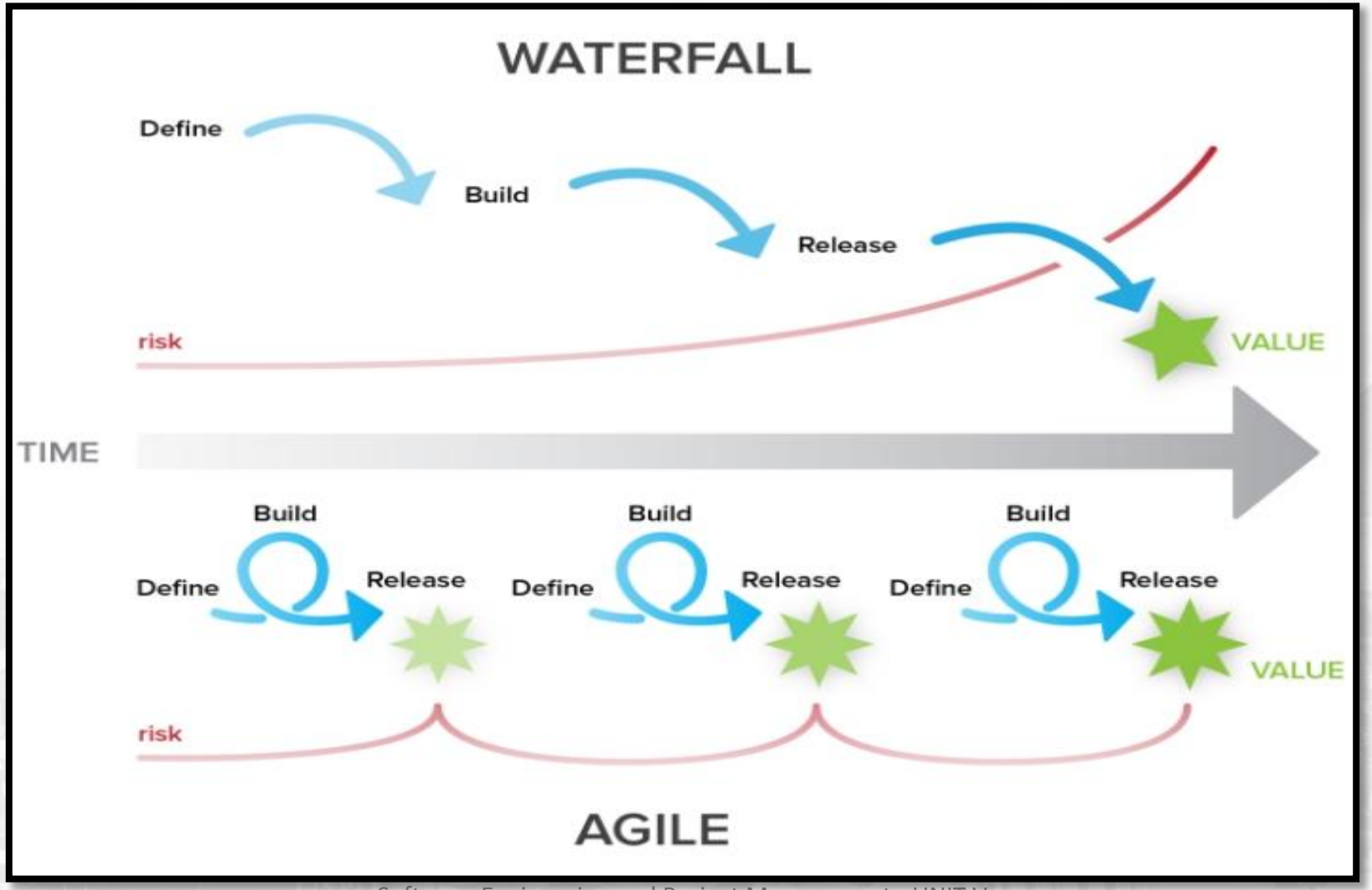
- **Waterfall** is a structured software development methodology so most times it can be quite rigid. ... Software development will be completed as one single project.
- **Agile** is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed.

Traditional Process model vs Agile

Waterfall vs. Agile

Traditional	Agile
Detailed plan	Dynamic plan
Sequential phases conducted by documentation Errors may be magnified	Iterative phases conducted by product delivery (by prioritizing) Errors may be minimized
Functionality at the end One Long period of time	Functionality after each iteration: Increments of the system every thirty days or less Short periods of time
Project Manager	Team self management
Requirements Analysis	Customers working with development team (every iteration)
Reports (Docs)	Communication

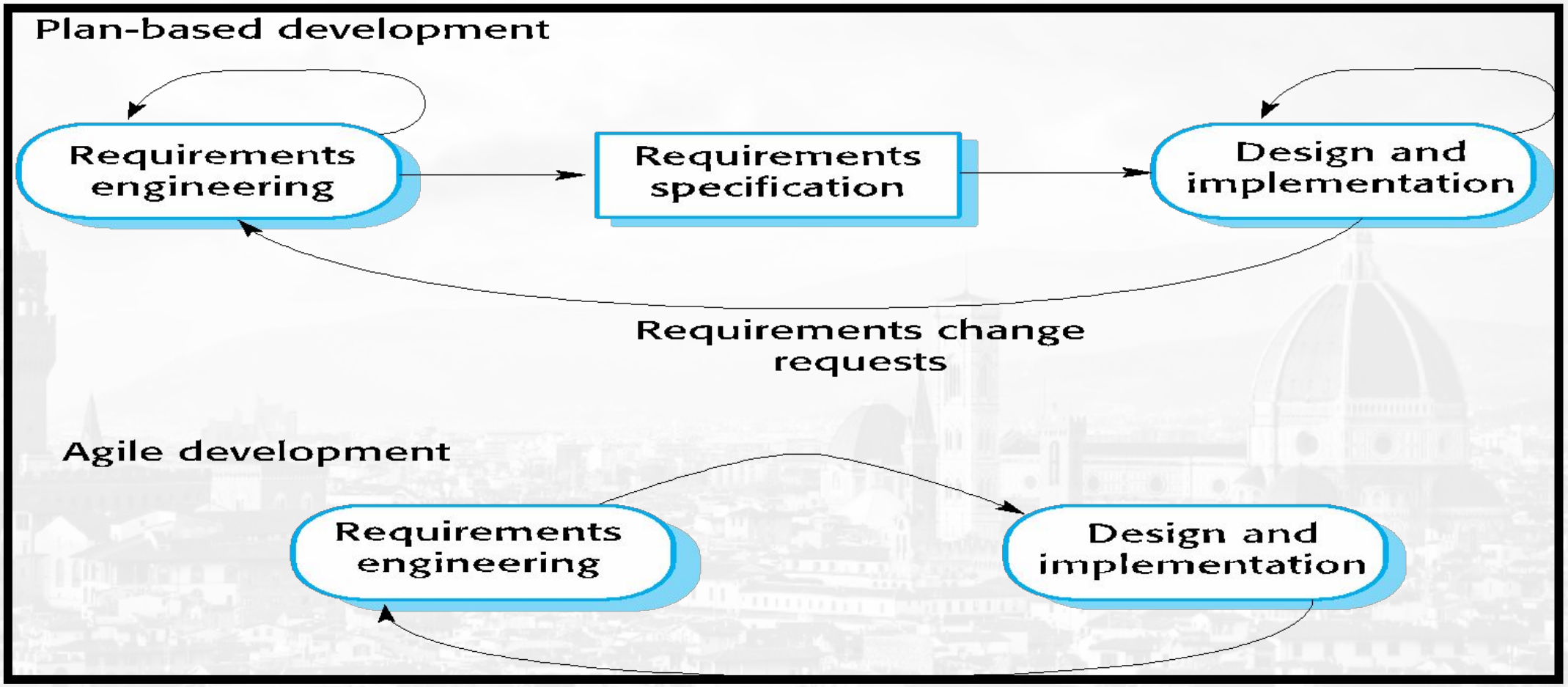
Waterfall model vs Agile



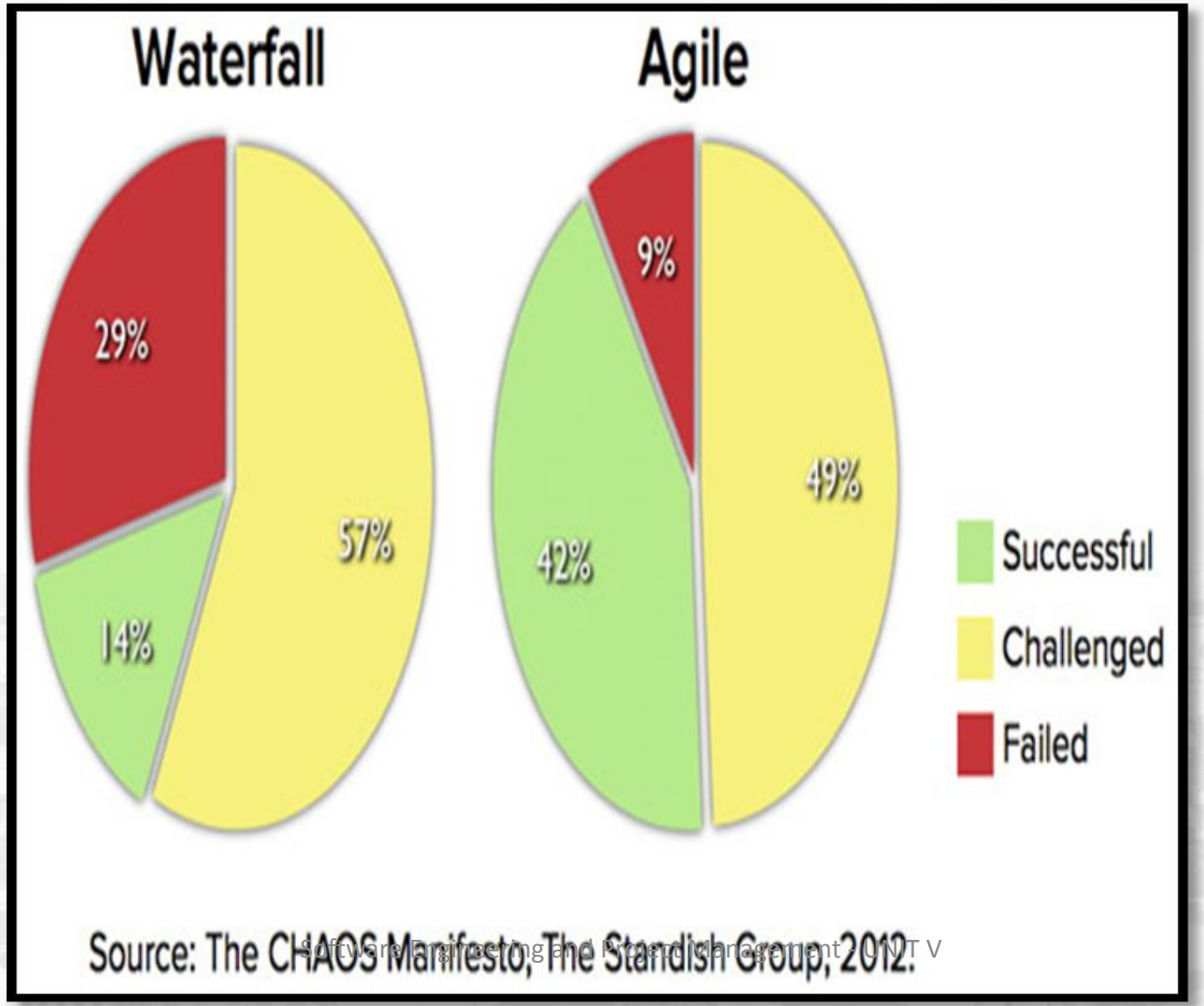
Plan-driven and agile development

- **Plan-driven development**
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
  - Not necessarily waterfall model – plan-driven, incremental development is possible
  - Iteration occurs within activities.
- **Agile development**
  - Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Plan-driven and agile specification



Comparison of failure rate



Agile Manifesto

The four core values of Agile software development as stated by the Agile Manifesto are:

Individuals and Interactions over Processes and Tools

- People make biggest impact on success
  - Process and environment help, but will not create success.
- Strong individuals not enough without good team interaction.
  - Individuals may be stronger based on their ability to work on a team.
- Tools can help, but bigger and better tools can hinder more than help
  - Simpler tools can be better

Working Software over Comprehensive Documentation

- Documentation important, but too much is worse than too little
  - Long time to produce, keep in sync with code
  - Keep documents short and salient
- Focus effort on producing code, not descriptions of it
  - Code should document itself
  - Knowledge of code kept within the team
- *Produce no document unless its need is immediate and significant.*

Responding to Change over Following a Plan

- Agile Development is focused on quick responses to change and continuous development.
- Changes due to Environment, requirements, and estimates of work are inevitable.
- Planning out a whole project doesn't hold up
  - Changes in shape, not just in time
- Keep planning realistic
  - Know tasks for next couple of weeks
  - Rough idea of requirements to work on next few months
  - Vague sense of what needs to be done over year

Customer Collaboration over Contract Negotiation

- As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- Get regular customer feedback

Agile method is suitable if …

- Product development where a software company is developing a small or medium-sized product for sale.
- Customized system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.
- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.

Principles in Agile Projects

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.

Principles in Agile Projects

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.

Purpose of Agile Manifesto and Principles

- Proponents of Agile methodologies say the four values outlined in the Agile Manifesto promote a software development process that focuses on quality by creating products that meet consumers' needs and expectations.
- The 12 principles are intended to create and support a work environment that is focused on the customer, that aligns to business objectives, and that can respond and pivot quickly as user needs and market forces change

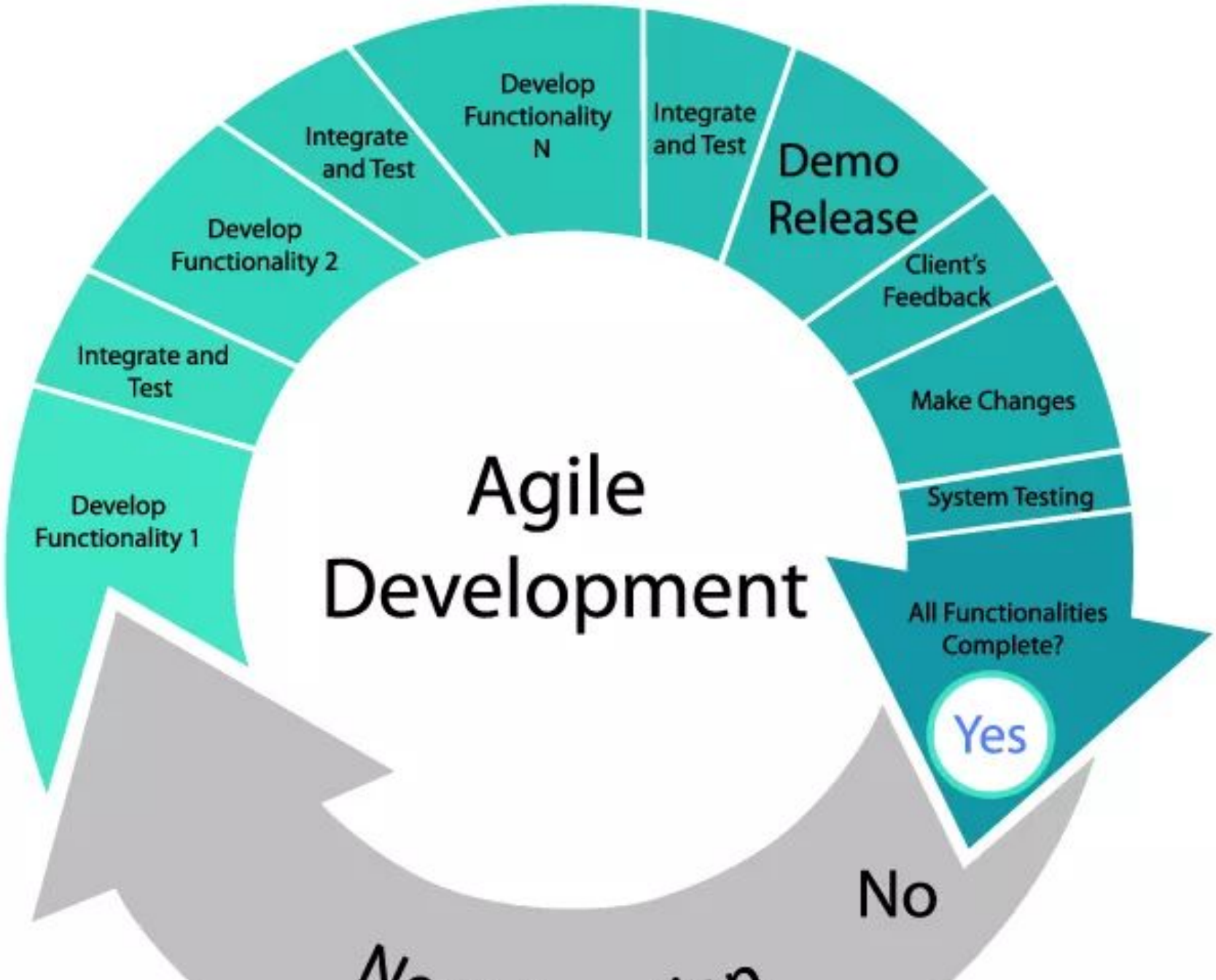
Agile methods and software maintenance

- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues:
  - Are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
  - Can agile methods be used effectively for evolving a system in response to customer change requests?
- Problems may arise if original development team cannot be maintained.

An Agile Practices and Processes

- Is driven by customer descriptions of what is required (scenarios). Some assumptions:
  - Recognizes that plans are short-lived (some requirements will persist, some will change. Customer priorities will change)
  - Develops software iteratively with a heavy emphasis on construction activities (design and construction are interleaved, Design models are proven as they are created.)
  - Analysis, design, construction and testing are not predictable.
- Thus has to Adapt as changes occur due to unpredictability
- Delivers multiple 'software increments', deliver an operational prototype or portion of an OS to collect customer feedback for adaption.

cesses


Develop Functionality 1
Integrate and Test
Develop Functionality 2
Integrate and Test
Develop Functionality N
Integrate and Test
Demo Release
Client's Feedback
Make Changes
System Testing
All Functionalities Complete?
Yes
No
Agile Development

Agile Software Development Processes

- **Agile software development** is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.

- Software developed during one unit of time is referred to as an iteration, which may last from one to four weeks.

- Agile methods also emphasize working software as the primary measure of progress

Agile Software Development Processes

- **Characteristics of Agile Software Development**

- Light Weighted methodology
- Small to medium sized teams
- Vague and/or changing requirements
- Vague and/or changing techniques
- Simple design
- Minimal system into production

Aspects of Agile Approaches

1. Collaborative User Story creation
2. Retrospective
3. Continuous Integration
4. Iteration and release planning

Collaborative User Story creation

- User Stories adhere to a specific, predefined structure, and are thus a simplistic way of documenting the requirements, and desired end-user functionality.
- A User Story tells you three things about the requirement; **Who**, **What**, and **Why**.
- The requirements expressed in User Stories are short, simple, and easy-to-understand statements.

Format for creation of User Story

- A useful format for creating a User Story is the following:
  "As a *<role>,* I want *<goal/ desire>* so that *<benefit>".*
  Here is an example using the format:
  As a Database Administrator, I should be able to revert a selected number of database updates so that the desired version of the database is restored.

Retrospectives

The Retrospect Sprint Meeting is an important element of the ‘inspect-adapt’ Scrum framework and it is the final step in a Sprint.

Some facts and some techniques are as follows:

- Meet at the end of each iteration
  - What worked and what didn’t work so well
  - How to improve and how to retain success
- Discussed on process, people, organizations, relationships, tools.
- Follow through is required.
- Essential to self-organization and continual improvement
- Address on: test effectiveness/ efficiency, test case quality, team satisfaction, and testability issues.

Continuous Integration

- This software engineering practice used for merging all developer working copies with a shared mainline multiple times a day. Some key points:

Following steps are follows in integration process:

- Merge changes and integrate all code daily (or more often) for quick discovery of defects
- Developers code, debug, and check-in
- Continuous Integration automatically:
  - Static code analysis
  - Compile
  - Unit test (functional/ non-functional)
  - Deploy
  - Integration test (functional/ non-functional)
  - Report
- Automated Continuous Integration, regression testing, and feedback
- Manual testing of new features, changes, and defect fixes

Iteration and release planning

Iteration and release planning sessions are conducted to develop a Release Plan.

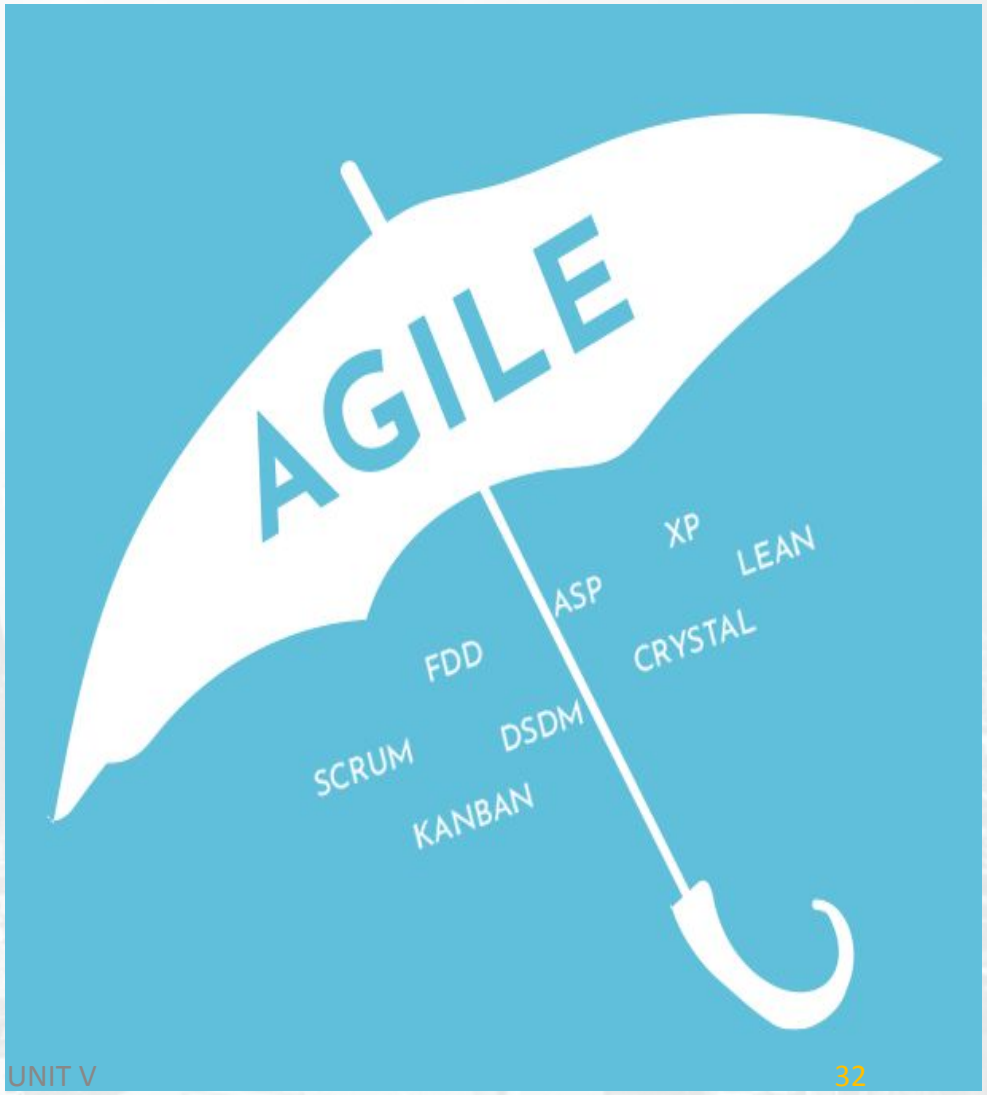
- The plan defines when various sets of usable functionality or products will be delivered to the Customer.
- The major objective of Team to have an overview of the releases and delivery schedule for the product they are developing, so that they can align with the expectations of the Product Owner and relevant Stakeholders (primarily the project sponsor).

Release Planning

- Planning is an on-going activity
- For Agile lifecycles: release and iteration planning
- Release planning
  - Define product backlog (User Stories for release)
  - Refine user epics
  - Basis of test approach and test plan for release
  - Identify project/ quality risks, estimate effort
- Testers:
  - Define testable User Stories
  - Participate in project and quality risk analyses
  - Estimate test effort
  - Plan testing for release
- Release plans may change during project due to internal and external factors

Agile Methodology / Process Models

- There are several agile methodologies; all share similar philosophies, characteristics, and practices.
- However, from the point of implementation each agile has its own practices, terminology, and tactics.
- Some of the main agile software development methodology components include:



Agile Methodology/ Process Models

1. Extreme Programming XP
2. Scrum
3. Dynamic Systems Development Method (DSDM)
4. Agile Modeling
5. Agile Unified Process

1. Extreme programming

- One of the most well-known agile methods
- 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.

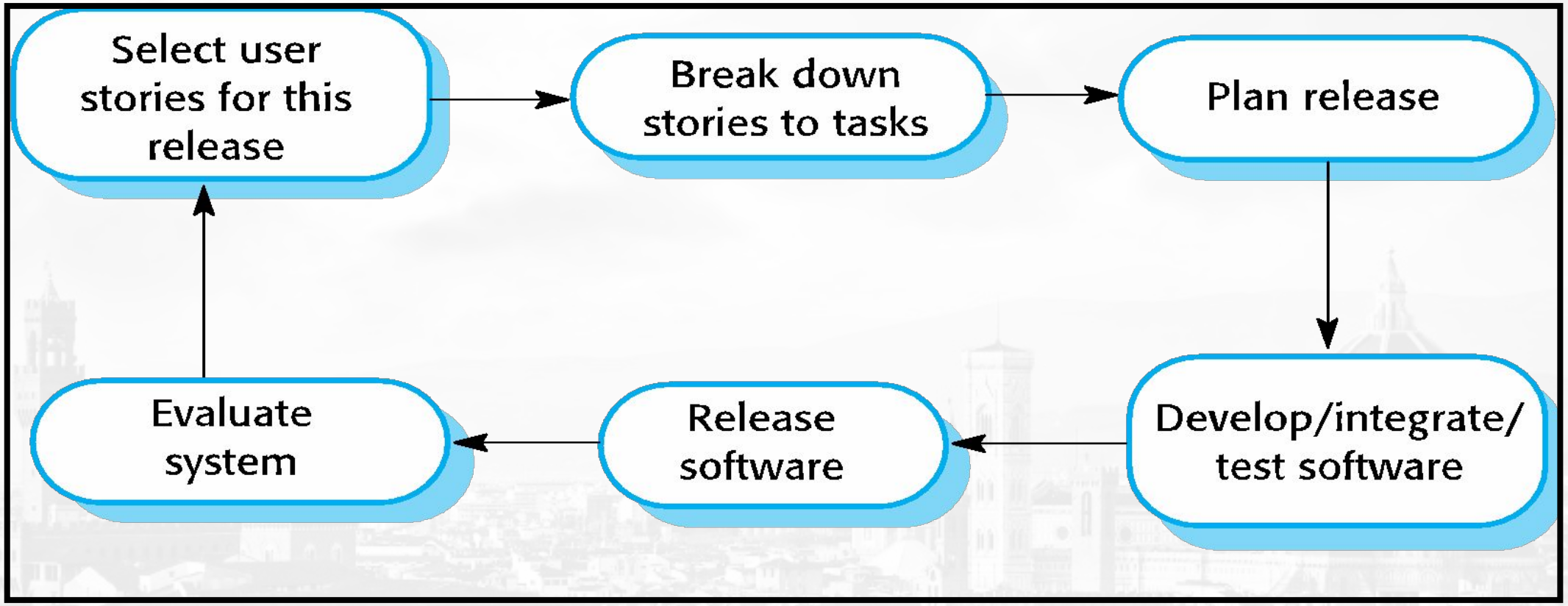
XP is appropriate where

- The general characteristics where XP is appropriate were :
  - Dynamically changing software requirements
  - Risks caused by fixed time projects using new technology
  - Small, co-located extended development team
  - The technology you are using allows for automated unit and functional tests

5 values of XP are

- Communication : to transfer knowledge from one team member to everyone else on the team.
- Simplicity : Simplicity means “what is the simplest thing that will work?” This helps to avoid waste and do only absolutely necessary things
- Feedback : Through constant feedback, teams can identify areas for improvement and revise their practices.
- Courage: You need courage to raise organizational issues , to stop doing something that doesn’t work , and need courage to accept and act on feedback, even when it’s difficult to accept.
- Respect : The members of your team need to respect each other in order to communicate with each other and to work together to identify simple designs and solutions.

The extreme programming release cycle



A user story is an informal, natural language description of one or more features of a software system. User stories are often written from the perspective of an end user or user of a system

Extreme programming practices (a)

Principle or practice	Description
Incremental planning	Recorded stories on story cards and their relative priority. The developers break these stories into development 'Tasks'.
Small releases	Set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

Extreme programming practices (b)

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

2. SCRUM

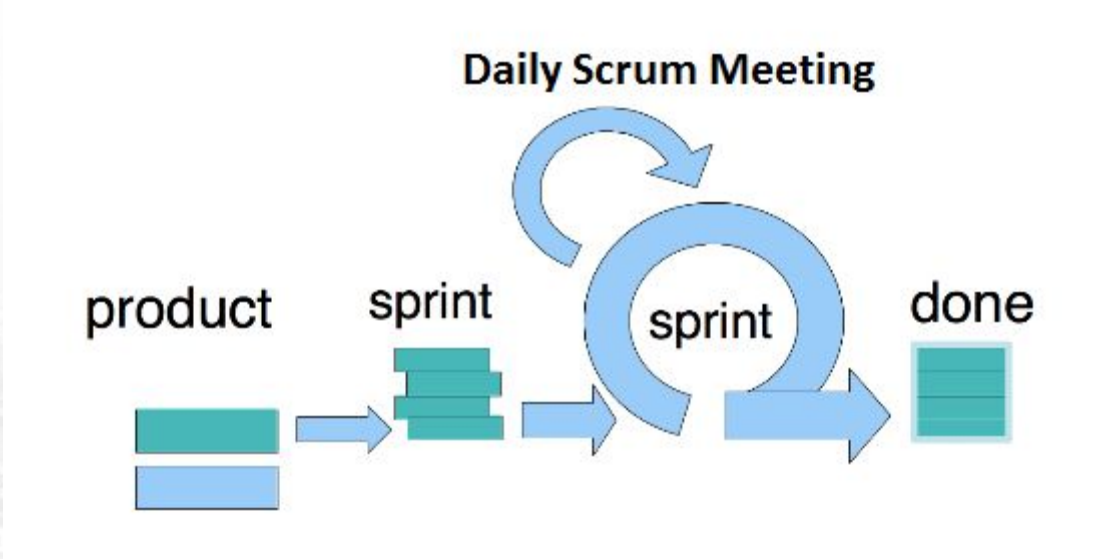
- Scrum is an Agile Software Development Process.
- Scrum is not an acronym
- Name taken from the **sport of Rugby**, where everyone in the team pack acts together to move the ball down the field
- Analogy to development is the team works together to successfully develop quality software.
- Scrum focuses on the entire organization for its implementation to be a success.

SCRUM

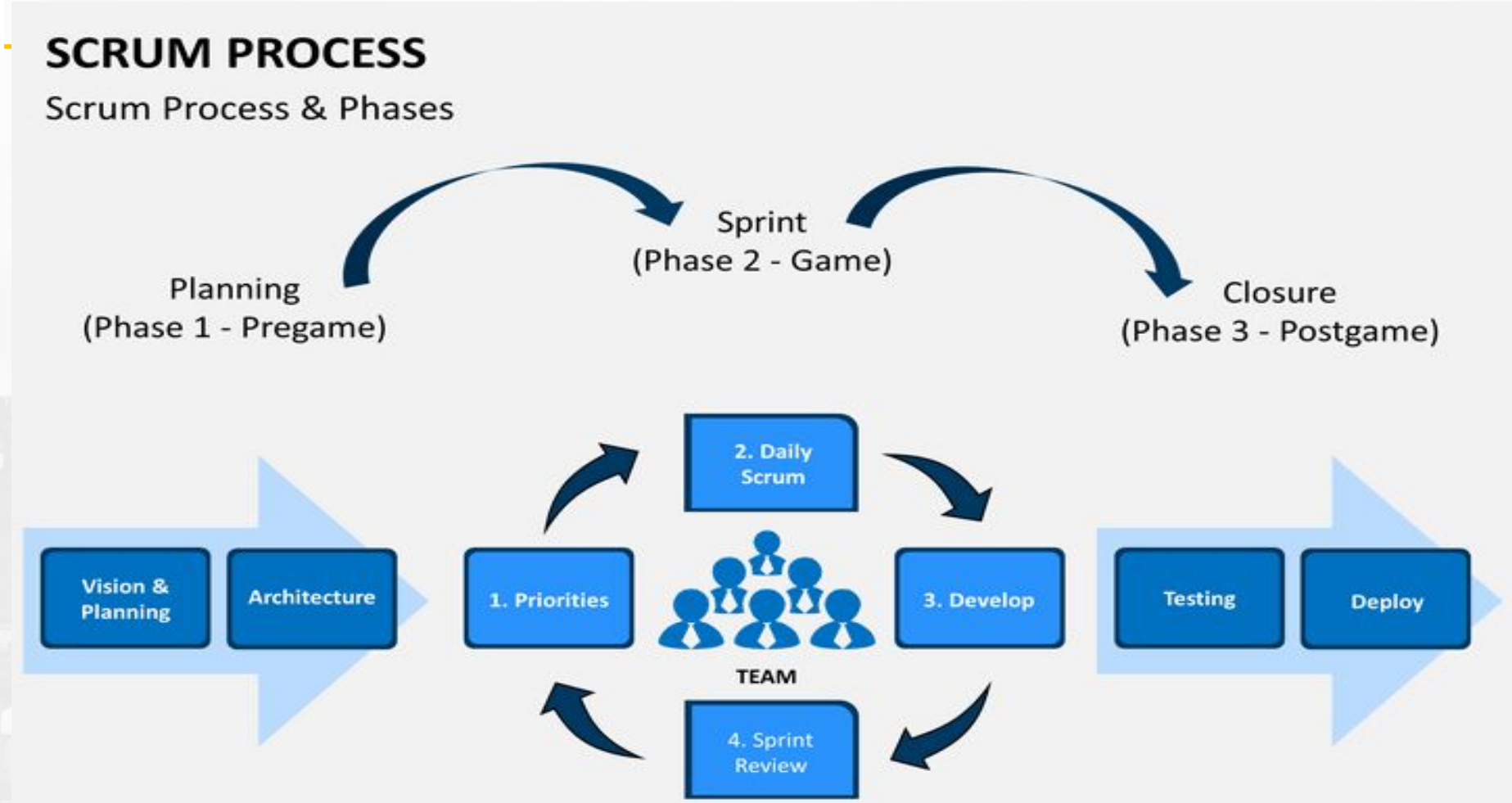
- **Scrum principles include:**
  - **Quality work:** empowers everyone involved to be feel good about their job.
  - **Assume Simplicity:** Scrum is a way to detect and cause removal of anything that gets in the way of development.
  - **Embracing Change:** Team based approach to development where requirements are rapidly changing.
  - **Incremental changes:** Scrum makes this possible using sprints where a team is able to deliver a product (iteration) deliverable within 30 days.

Sprints in SCRUM

- A **sprint** is a short, time-boxed period when a **scrum** team works to complete a set amount of work. **Sprints** are the very heart of **scrum** and **agile** methodologies.
- With Scrum, a product is built in a series of iterations called sprints that break down big, complex projects into bite-sized pieces.
- The goal of this meeting is to surface any blockers and challenges that would impact the teams ability to deliver the sprint goal.



Scrum Methodology- Scrum Phases



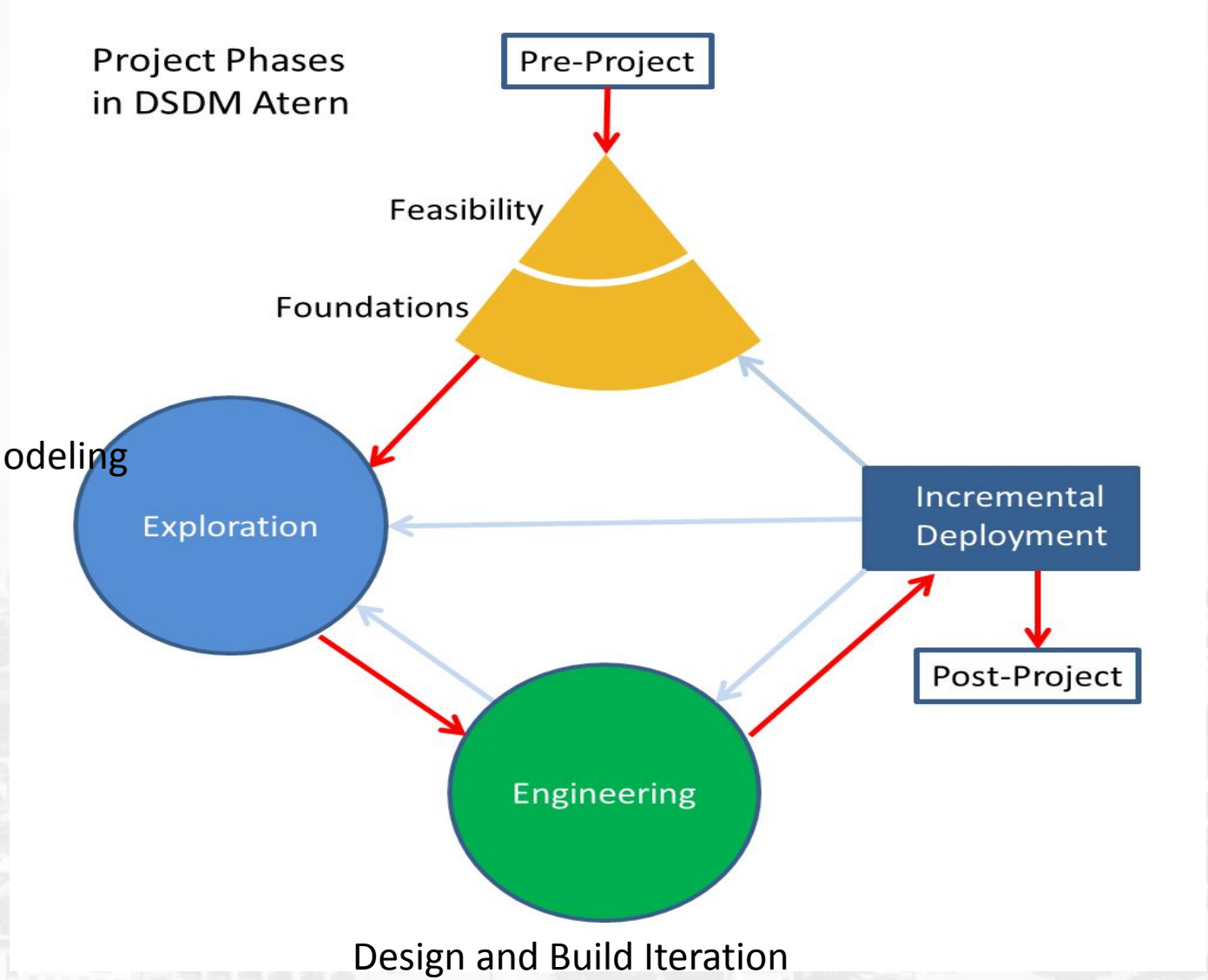
Difference Between Scrum vs XP

Scrum	XP
• Changes in sprint are not allowed • Once tasks for a certain sprint are set, the team determines the sequence in which they will develop the backlog items • The Scrum Master is responsible for what is done in the sprint, including the code that is written • The validation of the software is completed at the end of each sprint, at Sprint Review	• As long as the team hasn't started working on a particular feature, a new feature, of equivalent size can be swapped into the interation in exchange for an un-started feature • Tasks are taken in a strict priority order • Developers can modify or refactor parts of code as the need arises • The software needs to be validated at all time, to the extent that tests are written prior to the actual software

3. Dynamic System Development Method (DSDM)

- It is an iterative, incremental approach that is largely based on the RAD
- The method provides a four-phase framework consisting of:
  - Feasibility and business study
  - Functional model / prototype iteration
  - Design and build iteration
  - Implementation
- Eliminate problems of:
  - Going over-budget
  - Missing deadlines
  - Users not involved
  - Management not committed

DSDM Development Process



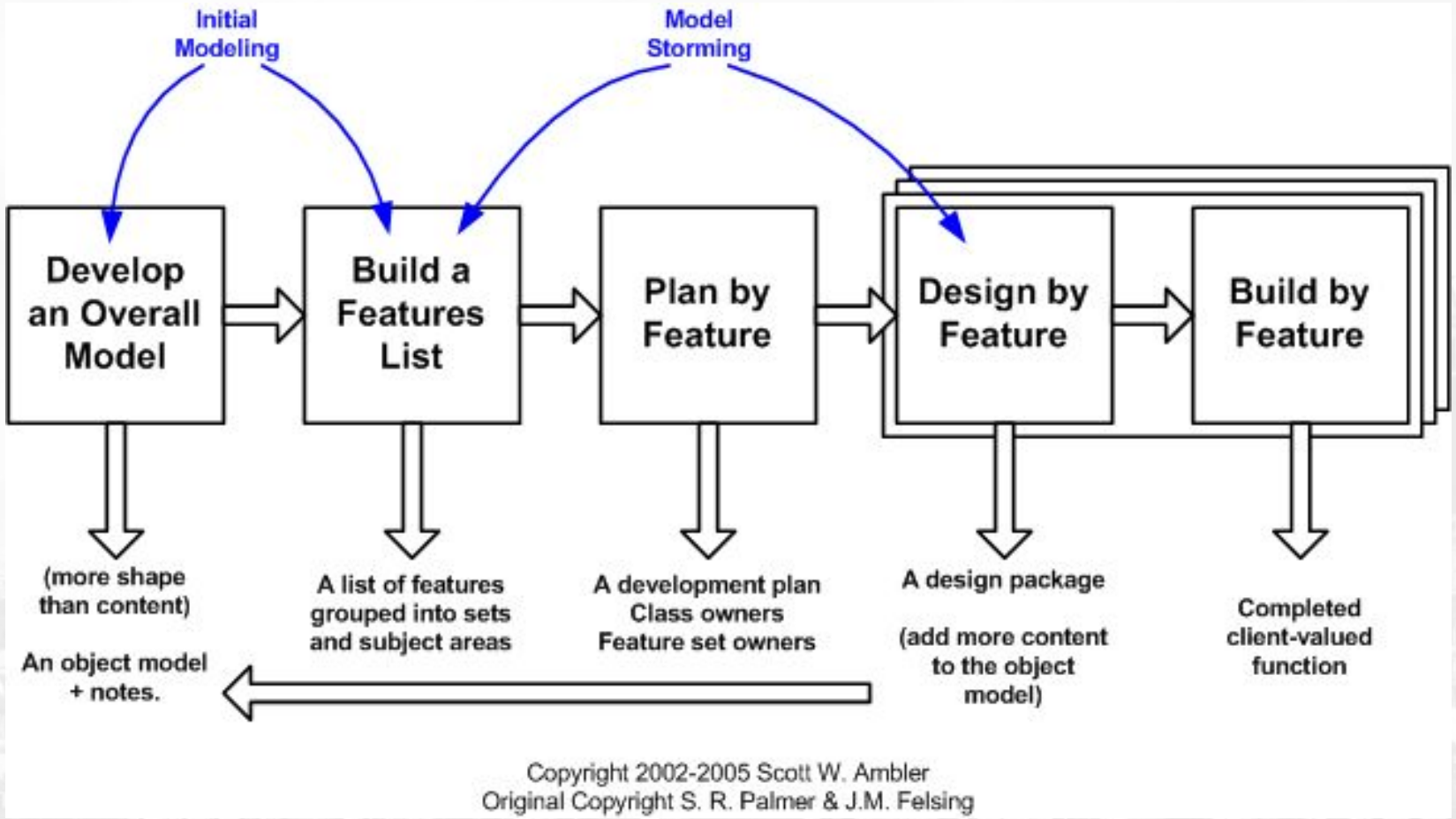
DSDM 9 Principles

1. Active user involvement
2. Teams must be empowered to make their own decisions.
3. Frequent releases more important than maximizing quality.
4. Primary criteria for deliverables is meeting business needs.
5. Iterative development is essential to reach correct solution.
6. Any change during development can be reversed.
7. The most high level requirements should be unchangeable.
8. Testing shall occur throughout the lifecycle of the project.
9. All stakeholders must cooperate and communicate.

4. Feature Driven Development FDD

- FDD organizes software development around making progress on features.
- A feature is a small, client-valued function expressed in the form <action><result><object>.
- That project lifecycle looks like this:
  - Develop an overall model
  - Build a features list
  - Plan by feature
  - Design by feature
  - Build by feature

FDD Lifecycle



FDD – Website development

- Feature Driven Development stresses in creating shorter iterations of functionality, with each functionality catering to certain features in the website.
- Best suitable as functionality of complex web-projects grows up

Strengths and Weaknesses

FDD's strengths include:

- Simple five-step process allows for more rapid development
- Allows larger teams to move products forward with continuous success
- Leverages pre-defined development standards, so teams are able to move quickly

FDD's weaknesses include:

- Does not work efficiently for smaller projects
- Less written documentation, which can lead to confusion
- Highly dependent on lead developers or programmers

5. Agile Unified Process (AUP)

AUP describes a simple, easy to understand approach to developing application using agile techniques and concepts yet still remaining true to the RUP.

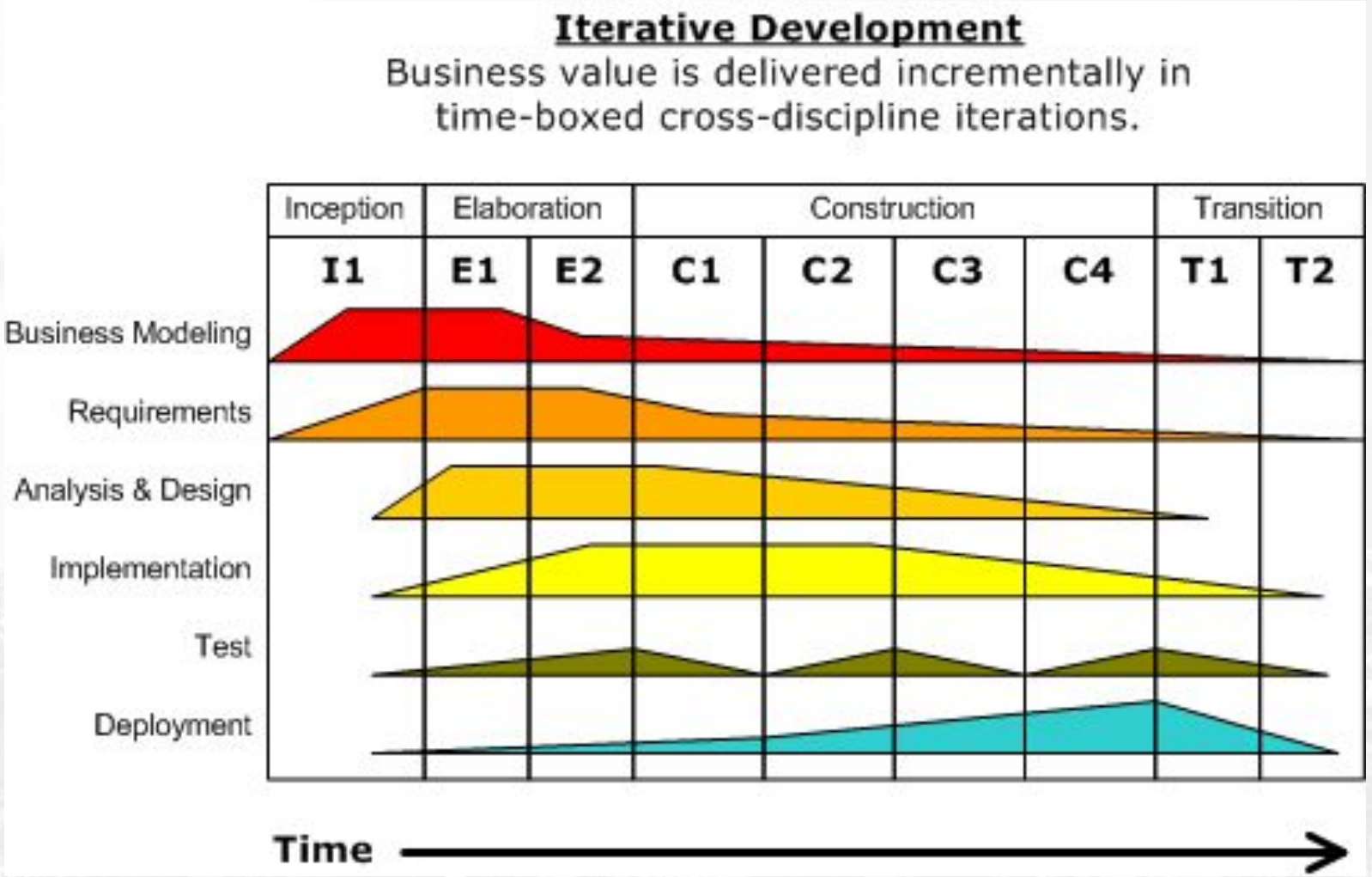
AUP has 7 phases :

- **Model**.
- **Implementation**.
- **Test**.
- **Deployment**.
- **Configuration Management**.
- **Project Management**.
- **Environment**.

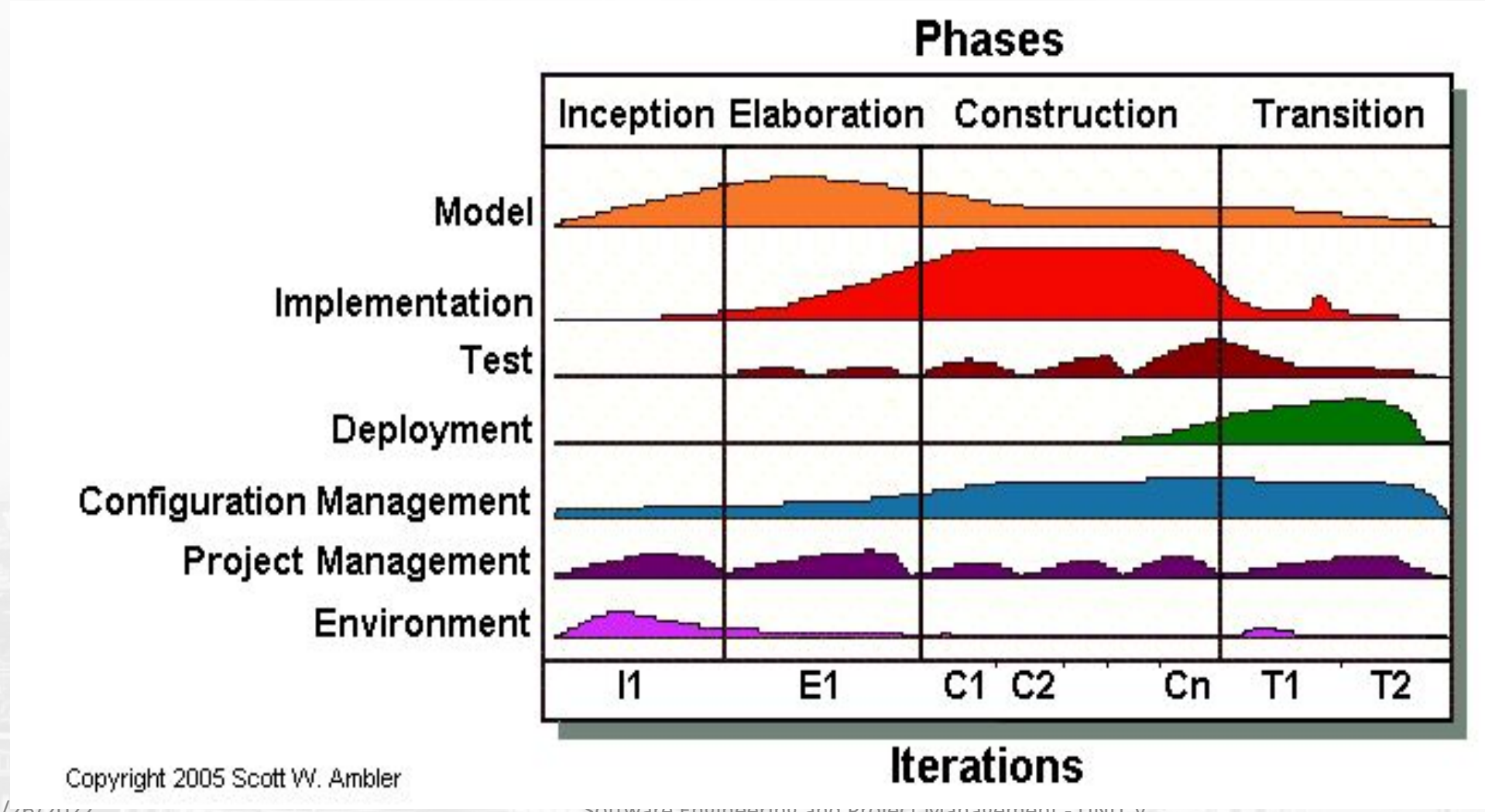
RUP (Revision)

The unified process model is an iterative, incremental and use-case driven approach for **developing** software.

It consists of four phases viz. **inception**, **elaboration**, **construction** and **transition**. This process treats software **development** as a 'unification' of mini iterations.



AUP


Phases
Inception
Elaboration
Construction
Transition
Model
Implementation
Test
Deployment
Configuration Management
Project Management
Environment
I1
E1
C1
C2
Cn
T1
T2
Iterations
Copyright 2005 Scott W. Ambler

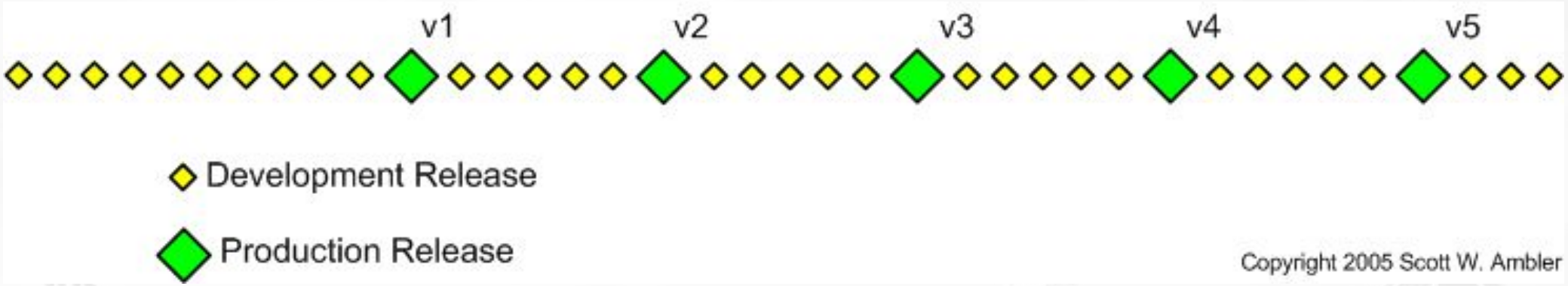
AUP : serial nature

- The serial nature of Agile UP is captured in its four phases :
  - Inception. The goal is to identify the initial scope of the project, a potential architecture for your system, and to obtain initial project funding and stakeholder acceptance.
  - Elaboration. The goal is to prove the architecture of the system.
  - Construction. The goal is to build working software on a regular, incremental basis which meets the highest-priority needs of your project stakeholders.
  - Transition. The goal is to validate and deploy your system into your production environment.

Iterative in AUP

- The disciplines are:
  - **Model**. understand the business of the organization, the problem domain being addressed by the project, and to identify a viable solution to address the problem domain.
  - **Implementation**. : transform your model(s) into executable code and to perform a basic level of testing, in particular unit testing.
  - **Test**. : perform an objective evaluation to ensure quality. This includes finding defects, validating that the system works as designed, and verifying that the requirements are met.
  - **Deployment**. The goal of this discipline is to plan for the delivery of the system and to execute the plan to make the system available to end users.
  - **Configuration Management**. The goal of this discipline is to manage access to your project artifacts. This includes not only tracking artifact versions over time but also controlling and managing changes to them.
  - **Project Management**. The goal of this discipline is to direct the activities that takes place on the project. This includes managing risks, and coordinating with people and systems to be sure that it is delivered on time and within budget.
  - **Environment**. The goal of this discipline is to 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.

Delivering Incremental Releases Over Time



Instead of the "big bang" approach where you deliver software all at once you instead release it into production in portions (e.g. version 1, then version 2, and so on).

Agile Unified Process Goals

- Stakeholders focus on the problem, solution, and constraints (visioning workshops).
- Users focus on their needs and product features at a high level (requirements workshops).
- The stakeholders and users focus on establishing the business justification for the product and project.
- The users and software development team focus on establishing the technology justification for the product and project.
- The team (software development team, stakeholders, and users) focus on prioritizing risks and opportunities (road-mapping workshops).
- The team focuses on relating product features to Execute (Innovation) goals and objectives in the project roadmap (road-mapping workshops).
- The team establishes the development infrastructure.

Agile Recommendations

- If you are a lone programmer, use the XP practices that are suitable for you.
- If you are a small team (two to eight people), consider using XP and introducing XP practices in small bunches, each bunch every two to three weeks.
- If you are in charge of a bigger team (eight to 20), consider using Scrum or Crystal as a high-level process "container," together with the XP practices that you deem sensible to use in your project.
- If you don't want to give up what you were taught (and maybe you are doing well) about OO modeling or code ownership, consider FDD, regardless of the size of your team.
- If the team is even bigger than 20 people, again consider Scrum, Crystal, or FDD, depending on your needs and preferences.

Benefits of Using Agile

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

Problems with agile methods

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

Popular Agile Tools

- **ActiveCollab.** An affordable tool for small businesses, ActiveCollab is easy to use. This software development aid requires little training and provides excellent support.
- **Agilo for Scrum.** Stakeholders get updated automatically on the project's progress with Agilo for Scrum. Features sprint reports and burn down charts for better data mining.
- **Atlassian Jira + Agile.** This powerful project management tool facilitates development by incorporating Scrum, Kanban, and customizable workflows.

Summary

- Agile models are based on iterative software development.
- An independent working module is built after the completion of iteration. Iteration should not consume more than two weeks to complete a code. Agile methodologies invite the developers to get involved in testing, rather than a separate quality assurance team.
- Agile methodologies are suitable in changing environments because of new practices and principles that enable a team to develop a product in short duration.

BOOKS

Books:

1. The Art of Agile Development James Shore, Shane Warden "O'Reilly Media, Inc." 3rd Edition, 2010