AGILE-PARADIGM SHIFT IN SDLC

Session - 11

Agile Process Models - II

Objectives

- Describe FDD
- Describe different FDD practices and principles
- List the processes under FDD
- Describe the various tools used under LSD
- List the principles under LSD
- Describe the philosophies of AUP
- List the AUP principles
- Explain the phases and processes under AUP
- List the strengths and weaknesses of AUP

Feature Driven Development (FDD)

- FDD is a type of software development methodology.
- Ideal for use by big teams, who undertake assignments involving object-oriented technology.
- FDD is a highly adaptive software development process, as:

  - Is highly and short iterative
  - Pays attention to quality, in all the steps
  - Provides working results at the end of each step
  - Is favored by managers, developers, and users

FDD – Practices [1-5]

- FDD is a process which is a combination of a number of best practices.
- Some of the practices are explained as follows:

Domain Object Modeling

- Involves creating class diagrams.

UML in Color

- Regular UML with color-coded classes defines the UML in Color process.
- Classes are classified under different color categories.

FDD – Practices [2-5]

- Each color has a different meaning:

Yellow

- Denotes the role assumed by a person in an organization. A user can have many different roles.

Blue

- This description helps classify or label an object, providing extensive details.

Green

- Stands for a party, place or thing, with something unique or particular identifiable about it.

Pink

- Refers to a point in time or an interval of time, which can be traced.

FDD – Practices [3-5]

Developing by Feature

- A complex function, which spans over two weeks, is broken down into smaller functions, which can be completed within two weeks. The smaller functions are called Features.

- The feature naming template is as follows:

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

- Some examples of Features are as follows:
  - Calculation of [action] the total [result] of a sale [object]
  - Evaluation of [action] the fulfillment timeliness [result] of a sale [object]
  - Calculation of [action] the total purchases made [result] by a customer [object]

FDD – Practices [4-5]

Class (Code) Ownership

- Class code ownership represents the person who has the ultimate responsibility for the contents of a class.

Feature Team

- Since the implementation of a feature could involve multiple classes, it could have multiple class owners.

Inspections

- This practice aims at publicizing good practices, conventions, and development culture.

Regular Build Schedule

- At regular points of time, a system is built using the source code of the completed features as well as the components and libraries it is dependent on.

FDD – Practices [5-5]

Configuration Management

- The main purpose of this best practice is the identification of the latest versions of the completed source code files.

Progress Reporting

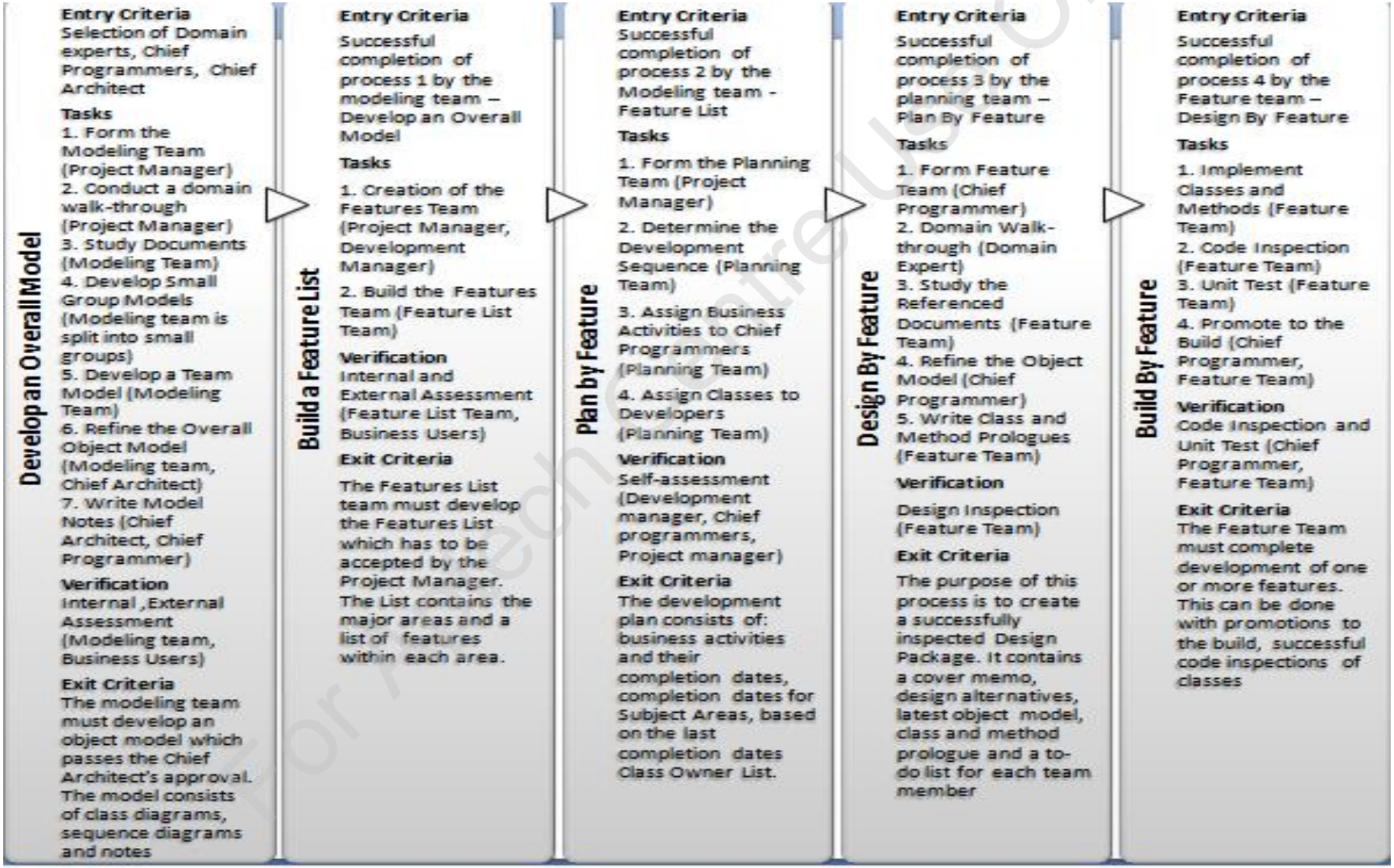
- Frequent reporting is an essential part of the process as it helps identify the status of completion of the project.

FDD Process [1-6]

- There are five processes under FDD are as follows:

FDD Process [2-6]

- Figure shows the five FDD processes.



FDD Process [3-6]

- Table shows five processes used under FDD.

	Process #1: Develop an Overall Model	Process #2: Build a Feature List	Process #3: Plan by Feature	Process #4: Design by Feature	Process #5: Build by Feature
Entry Criteria	The client collects all the requirements and decides what he really needs.	Selection of Domain Experts, Chief Programmers, and the Chief Architect has been completed.	Completion of the 'Build a Features List' process.	Completion of the Planning process.	Completion of the Design by Feature process—the conclusion is based on the successful inspection of the design package.

FDD Process [4-6]

	Process #1: Develop an Overall Model	Process #2: Build a Feature List	Process #3: Plan by Feature	Process #4: Design by Feature	Process #5: Build by Feature
Tasks	Form the Modeling Team Domain Walk-through Study Documents Develop the Model Refine the Overall Object Model Write Model Notes	Form the Features List Team Build Features List	Form the Planning Team Determine the Development Sequence Allocate Business Activities to Chief Programmers Assign Classes to Developers	Form Feature Team Domain Walk-through Study the Referenced Documents Refine the Object Model Write Class and Method Prologues	Implement Classes and Methods Code Inspection Unit Test Promote to the Build

FDD Process [5-6]

	Process #1: Develop an Overall Model	Process #2: Build a Feature List	Process #3: Plan by Feature	Process #4: Design by Feature	Process #5: Build by Feature
Verification	Internal or self-assessment is achieved. External assessment is made by referring back to the business (users) for ratification or clarification of issues that affects the model.	Two types of assessments are conducted – an internal and an external assessment.	A self-assessment is conducted.	The chief programmer decides whether to have a design inspection with the feature-team or any other project members. Once accepted, a 'To-Do' list is generated per affected class.	The verification of the output of this process can be traced to a successful code inspection and the successful completion of unit test.

FDD Process [6-6]

	Process #1: Develop an Overall Model	Process #2: Build a Feature List	Process #3: Plan by Feature	Process #4: Design by Feature	Process #5: Build by Feature
Exit Criteria	Class diagrams, Sequence Diagrams Notes substantiating the selection of a particular model shape.	A list of the Subject Areas. The Features which satisfy the criteria fixed in each Business Activity.	A schedule of business activities and their completion dates. The chief programmers assigned to each business activity. Assignment of completion dates for Subject Areas. The Class Owner List .	A Cover Sheet or memo. The documents and all related confirmation memos. The Sequence Diagram. The Design alternatives. The new/updated classes, methods, and attributes of the Object Model. The <your tool> output generated for the class and method prologues produced or altered due to this design. Calendar/To-Do task-list entries.	Successful code inspection of classes and/or methods. Classes promoted to the Build. The completion of a client-valued function, otherwise known as Feature.

Roles in FDD

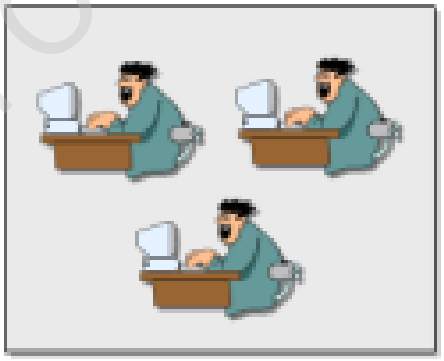
- Figure shows the key roles in FDD.

Project Manager

Domain Expert

Development Manager

Chief Architect



Chief Programmers

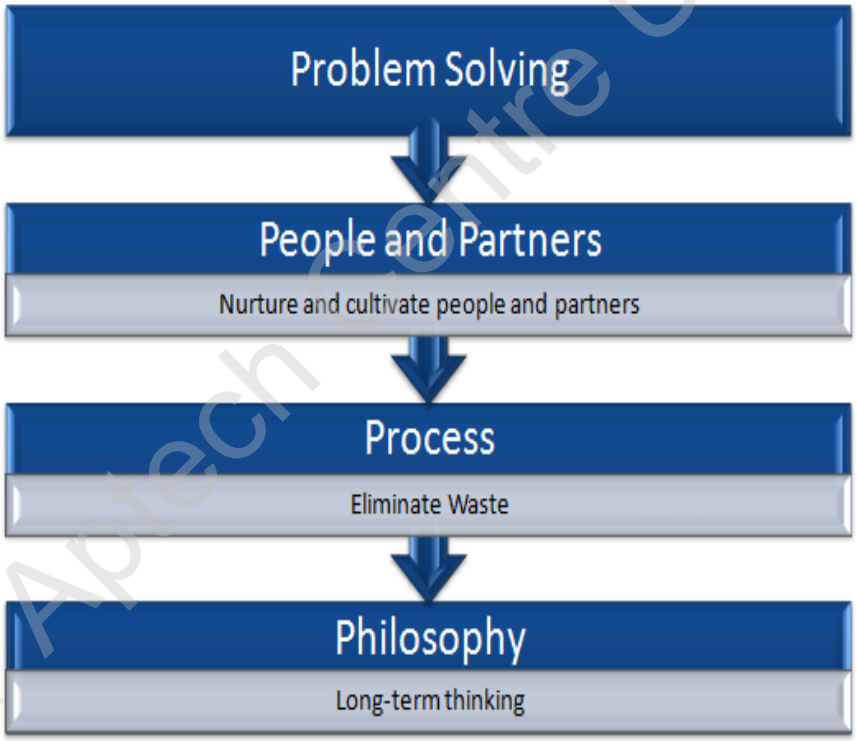
Class Owners

Need of FDD

- It shows that a project is an iterative and incremental type.
- It also explains what the agile-like methodologies can scale.
- It demonstrates how planning and a clear understanding of the domain at the beginning of the project, helps in drafting a plan to see it through completion
- It also helps work around '**analysis and design paralyses**'.
- It is usually used for large projects.

Lean Software Development (LSD)

- LSD is a movement dedicated to reducing errors and wasted time and for maximizing education and efficiency.
- Figure shows a high-level overview of LSD.



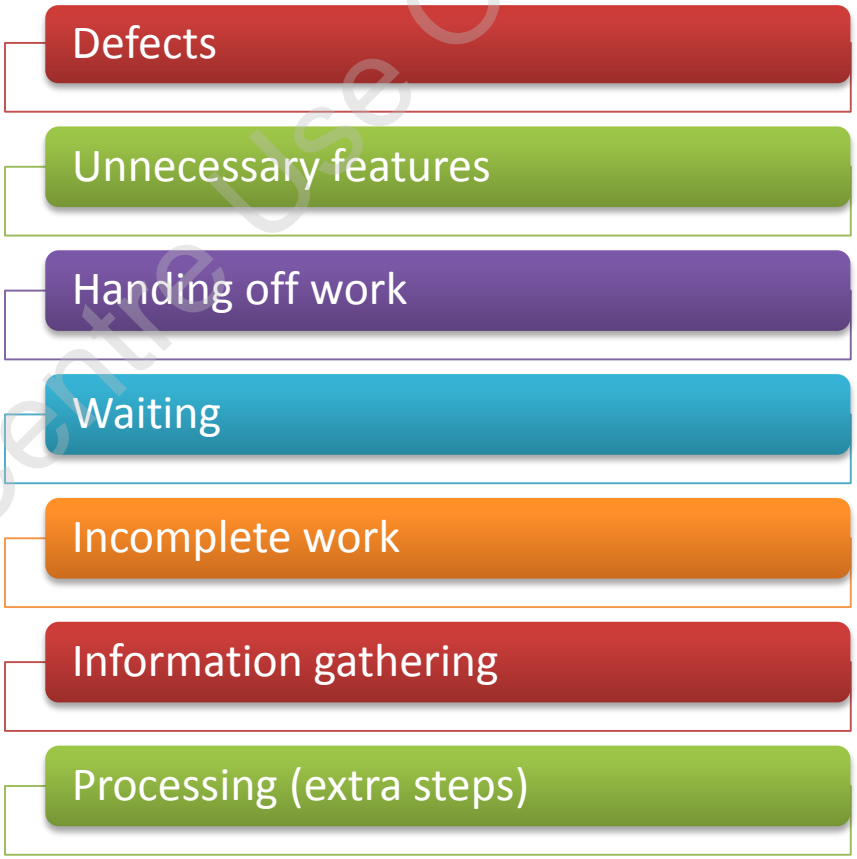
Principles of LSD [1-11]

- Seven key principles that form the backbone of LSD are as follows:

Principles of LSD [2-11]

Waste Elimination

- Anything that fails to bring any additional value to the customer can be defined as **'Waste'**.
- Several tools have been designed to eliminate the waste:
  - **Eliminate Waste by Seeing Waste (Tool #1)**
  - **Eliminate Waste by Value Stream Mapping (Tool #2)**
- Figure shows the seven types of wastes listed under software development.



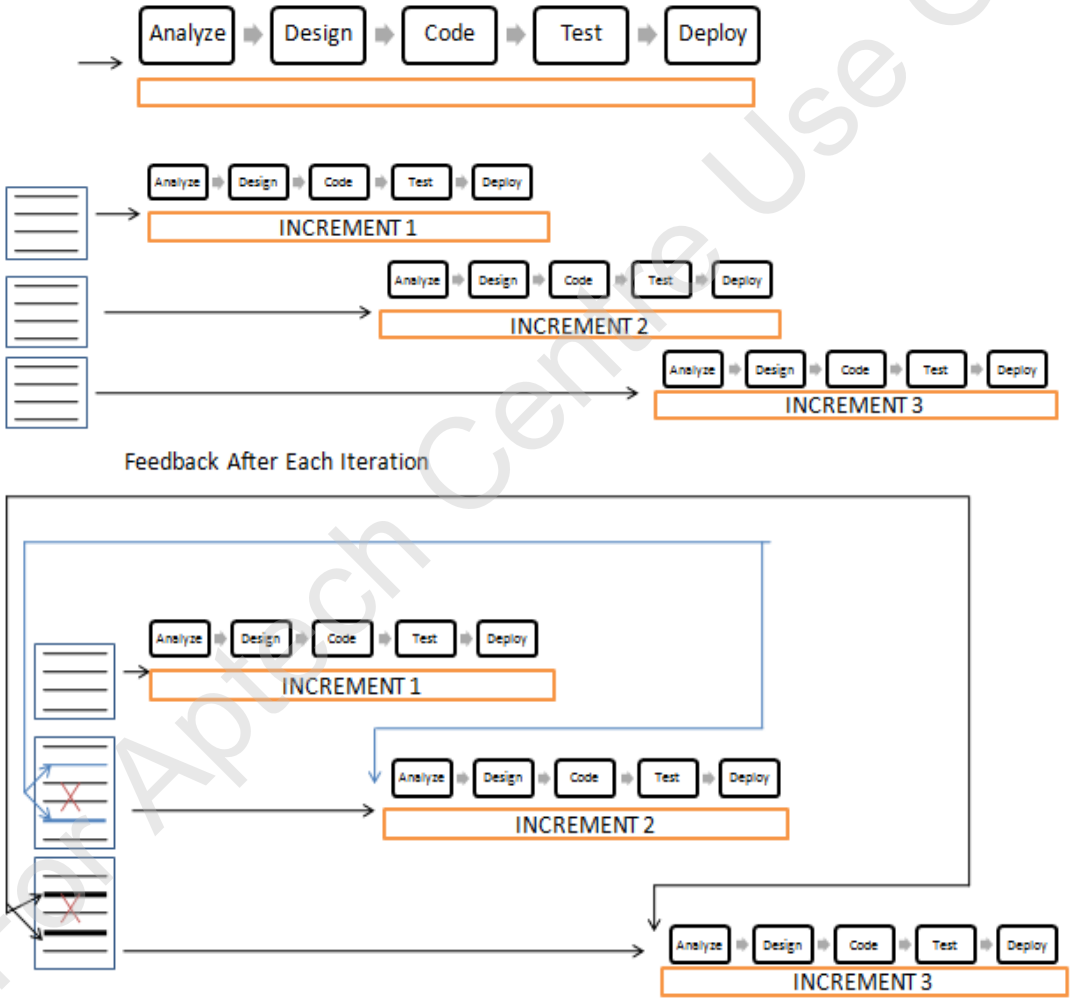
Principles of LSD [3-11]

Encourage Learning

- Provide all the tools required to have all the resources, working on a project, from comprehensive code reviews to documentation.
  - **Encouraging Learning with Feedback (Tool #3)**
  - **Encouraging Learning with Iterations (Tool #4)**

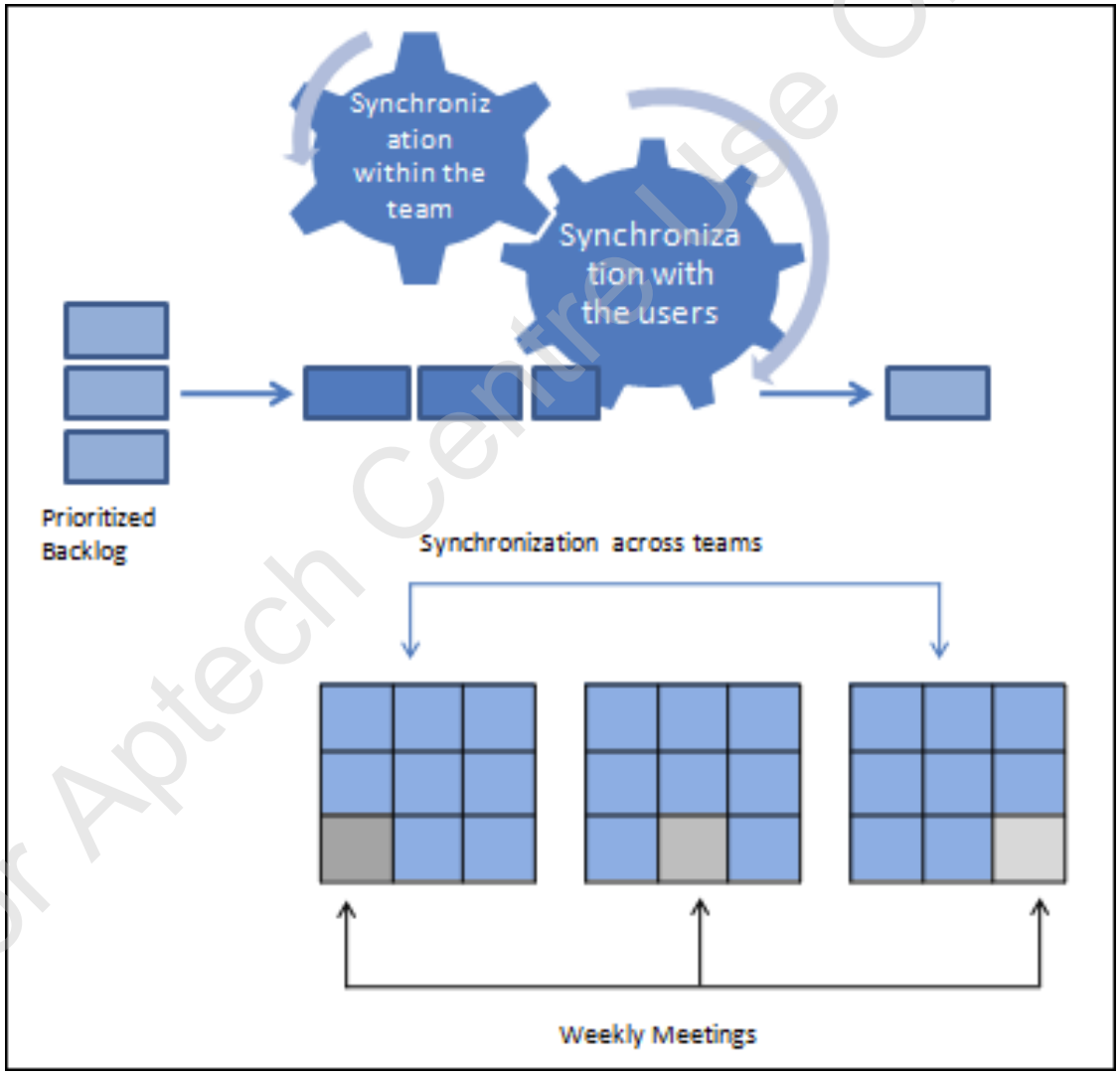
Principles of LSD [4-11]

- Figure shows the iterative and incremental approach.



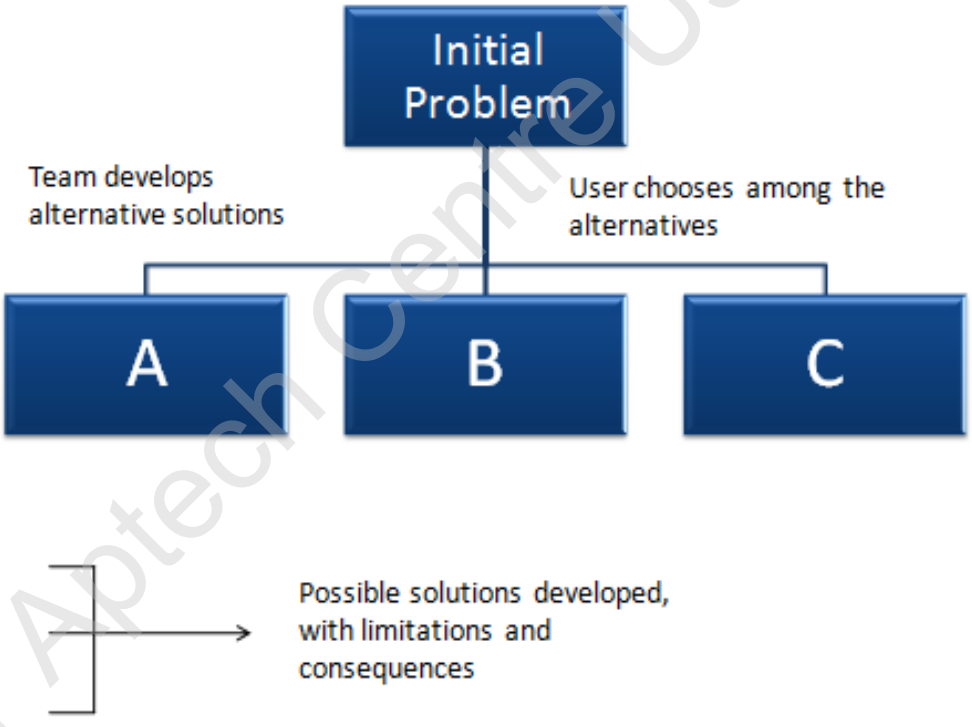
Principles of LSD [5-11]

- Figure shows **Encouraging Learning with Synchronization (Tool #5)**.



Principles of LSD [6-11]

- **Encouraging Learning with Set-Based Development (Tool #6)**
  - Figure shows the set-based developments which can bring about efficient and faster solutions.



Principles of LSD [7-11]

Late Decision-Making

- This process refers to postponing decision-making for as long as possible, especially decisions that will be difficult or impossible to take back.
  - **Decide as Late as Possible with Options Thinking (Tool #7)**
  - **Decide as Late as Possible with the Last Responsible Moment (Tool #8)**
  - **Decide as Late as Possible with Making Decisions (Tool #9)**

Principles of LSD [8-11]

Fast Delivery

- A fast delivery also translates to lesser time for changes suggested by users and managers.
  - **Deliver as Fast as Possible with Pull Systems (Tool #10)**
  - **Deliver as Fast as Possible with Queuing Theory (Tool #11)**
  - **Deliver as Fast as Possible – Cost of Delay (Tool #12)**

Principles of LSD [9-11]

Team Empowerment

- Lean team management requires the managers to ensure the programmers are not overloaded. They should also be able to establish their own reasonable workflow.
  - **Empower Team by Self-Determination (Tool #13)**
  - **Empower Team with Motivation (Tool #14)**
  - **Empower Team with Leadership (Tool #15)**
  - **Empower the Team with Expertise (Tool #16)**

Principles of LSD [10-11]

Built-In Integrity

- The main aim of the team should be to produce high-quality code.
  - **Build Integrity In – Perceived Integrity (Tool #17)**
  - **Build Integrity In – Conceptual Integrity (Tool #18)**
  - **Build Integrity In – with Refactoring (Tool #19)**
  - **Build Integrity In – with Testing (Tool #20)**

Principles of LSD [11-11]

Vision of the Big Picture

- The Lean development teams ought to view the project as a whole, not as separate segments. The entire project should be kept in mind while setting goals and mini-goals.
  - **See the Whole with Measurements (Tool #21)**
  - **See the Whole – Contracts (Tool #22)**

Agile Unified Process (AUP)

- It is a simplified version of the Rational Unified Process (RUP).
- It illustrates a simple and straightforward approach
- It is used to developing business application software using agile techniques and concepts.
- It conforms to the RUP.

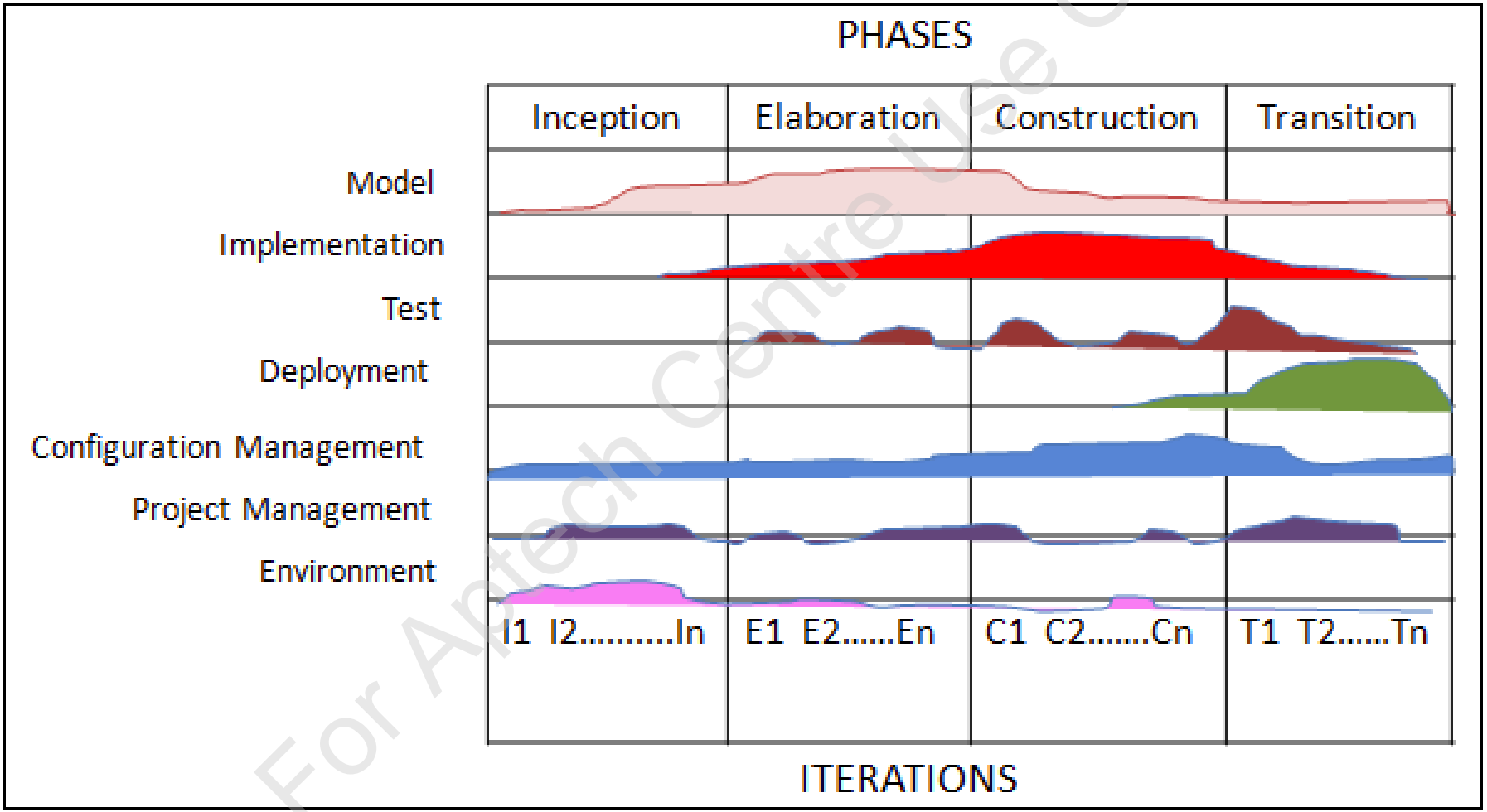
AUP Principles

AUP Principles

- Your staff knows what they are doing
- Simplicity
- Agility
- Focus on high-value activities
- Tool independence
- Tailor this product to meet your own needs

AUP Life Cycle

- Figure shows the life cycle phases of AUP.



AUP: Process - Phases

- The overall development cycle consists of four phases:

Inception

- Outline the scope of the project, identify potential architecture, obtain initial funding, and gain stakeholder acceptance.

Elaboration

- The goal is to prove the system architecture.

Construction

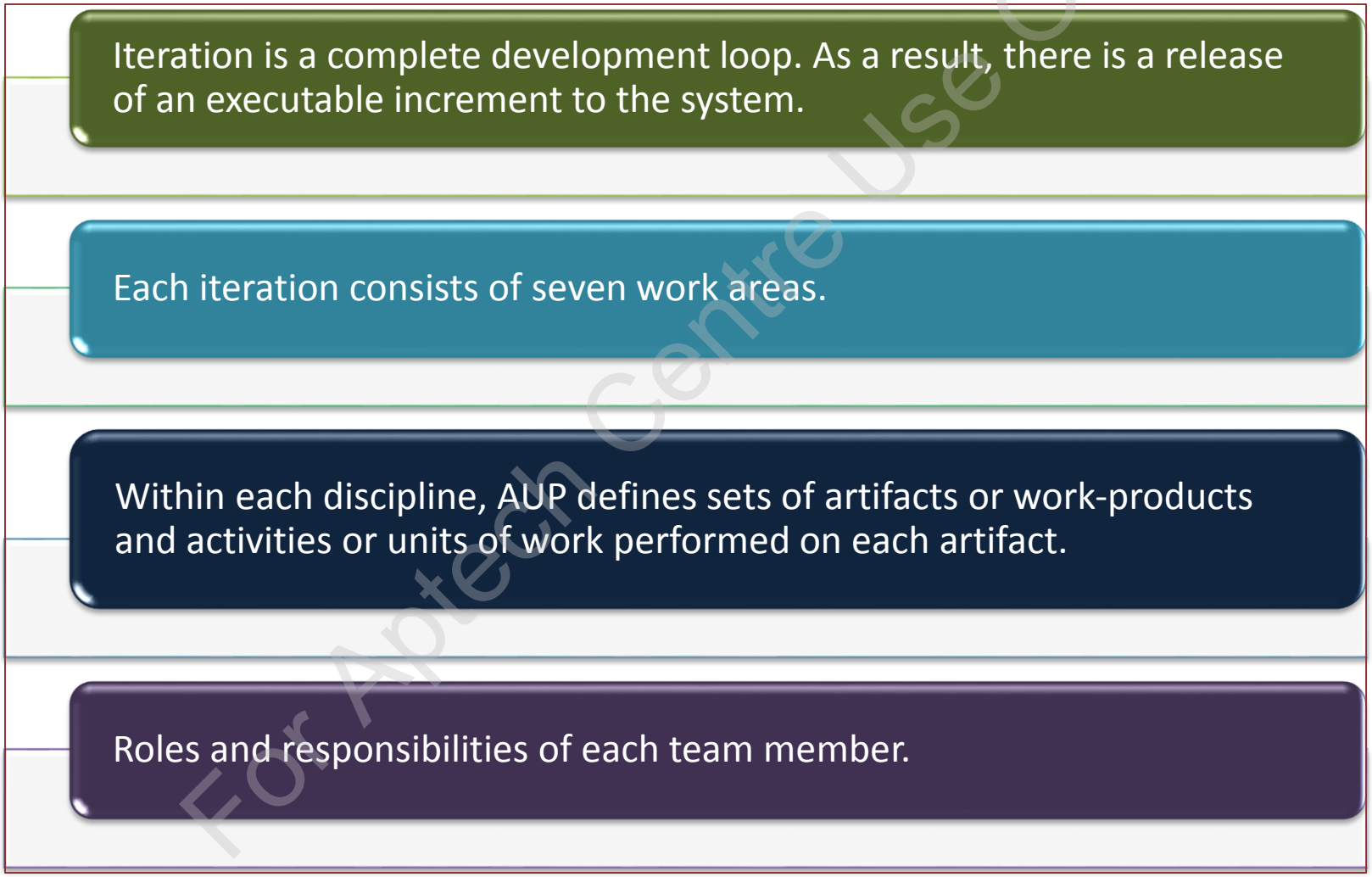
- The main objective is to develop working software in a frequent, incremental manner.
- This should meet the most critical requirements of project stakeholders.

Transition

- To validate and deploy the system into the production environment is the main aim of this function.

AUP: Process – Iterations and Disciplines

- Each phase can be further broken down into iterations:



AUP: Process – Disciplines

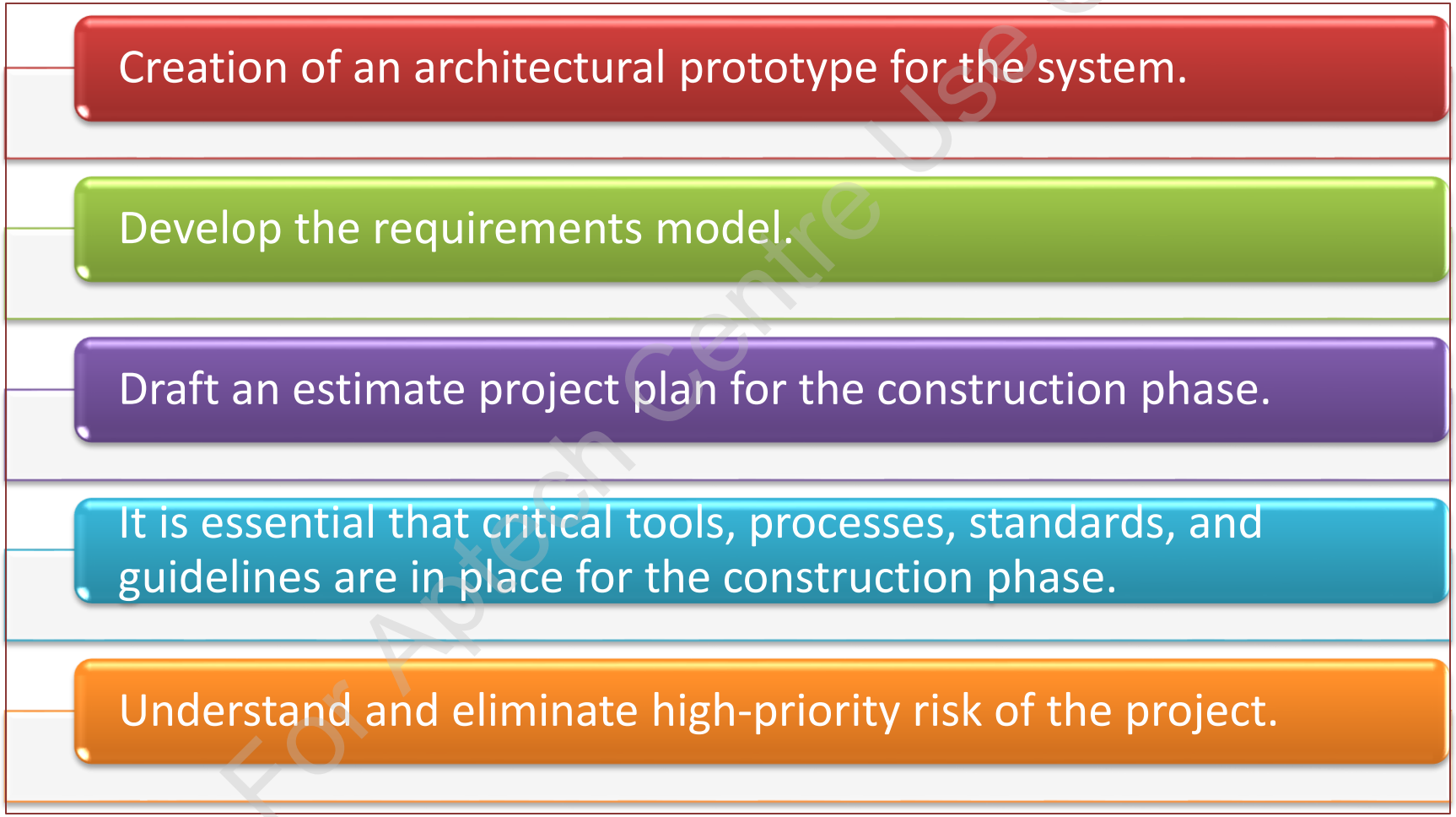
- Seven disciplines performed during each iteration:
  - Model
  - Implementation
  - Test
  - Deployment
  - Configuration Management
  - Project Management
  - Environment

AUP: Process – Inception Phase

- Following are the tasks in the Inception phase:

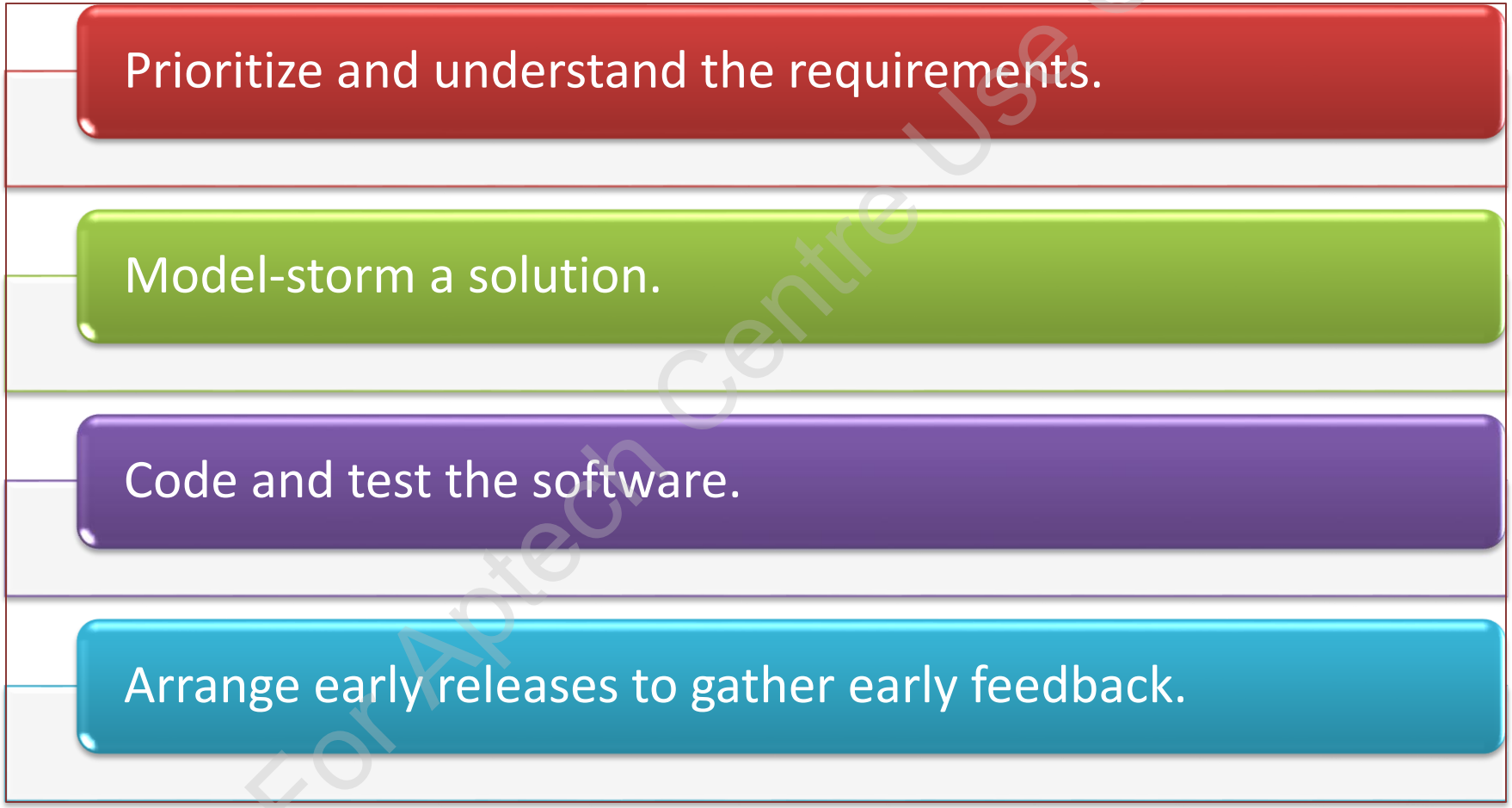
AUP: Process – Elaboration Phase

- Tasks that are part of the Elaboration phase are as follows:



AUP: Process – Construction Phase

- In the Construction phase, the following tasks are important:



AUP: Process – Transition Phase

- The main tasks in the Transition phase are as follows:

AUP: Process – Model Discipline

This encompasses RUP's Business Modeling, Requirements, and Analysis and Design disciplines.

Agility is observed by creating models which meet the bare minimum requirements.

AUP: Advantages and Disadvantages 1-2

- **Advantages:**
  - Iterative-incremental process.
  - Based on system architecture.
  - Based on structural, functional, and behavioral modeling – both logical and physical, of the problem domain and the system.
  - Based on system functionality, described in use-cases.
  - Can be traced to requirements through use cases.
  - Design-based development.
  - Iterative development engine governed by planning/reviewing.
  - Though limited, seamlessness can be observed due to use-case based activities and design-based development.
  - Risk-based process.
  - Facility to add formal features through UML/OCL.
  - Addresses configurability and flexibility.

AUP: Advantages and Disadvantages 2-2

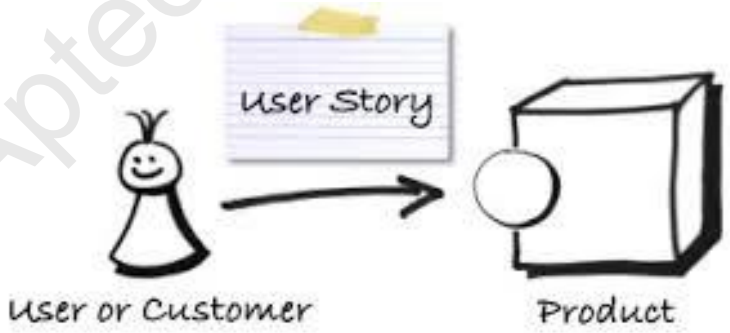
- **Disadvantages:**
  - It is possible that modeling can endanger agility if the limits are not observed precisely.
  - The list of AUP models that are produced, as a minimum, is extensive.
  - Tackling model inconsistencies is not addressed, explicitly.

User Stories [1-2]

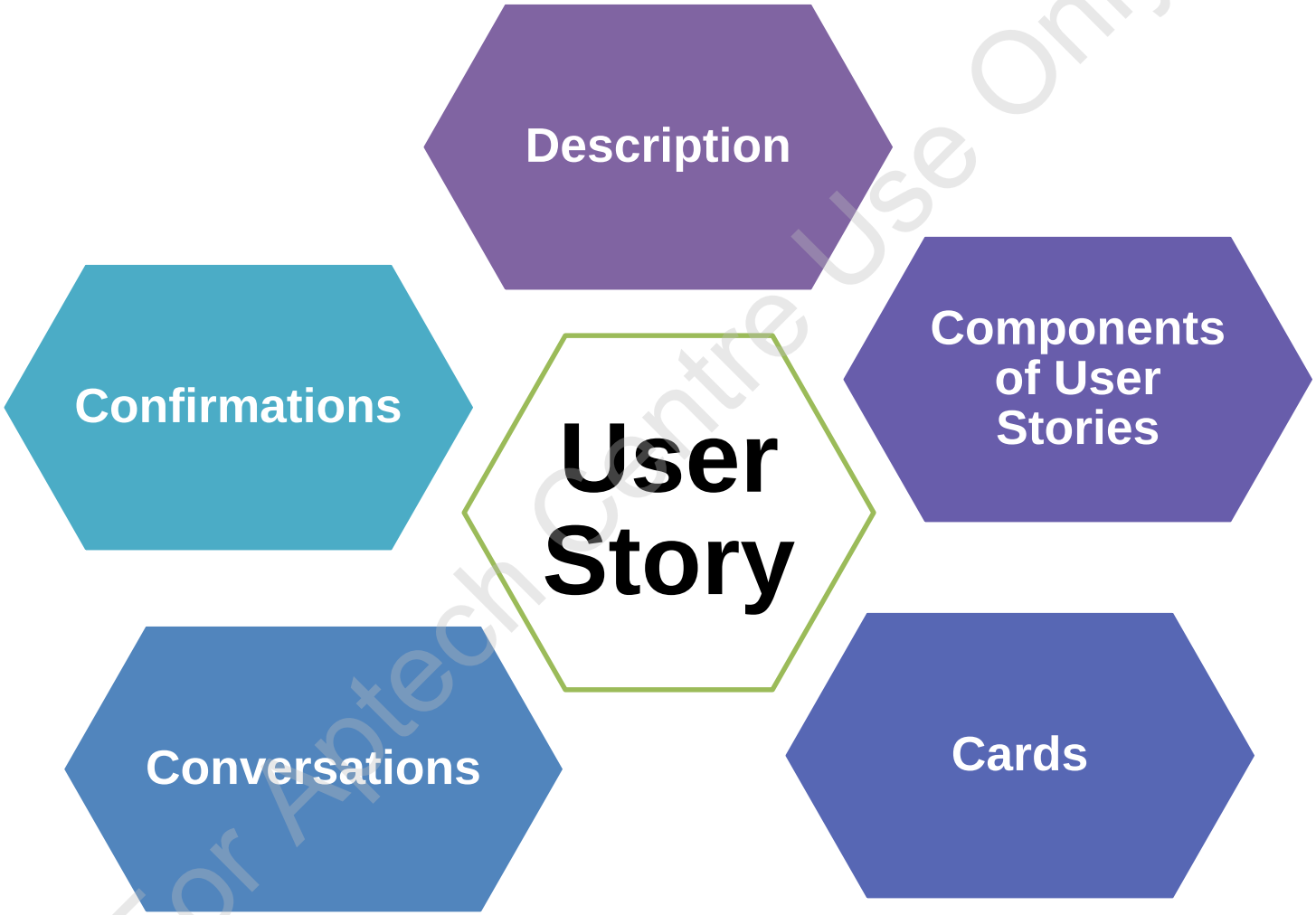
- User Stories can be described as easy and simple articles.
- Describe the wish list of a new feature, from the viewpoint of a user.
- Represented in the following template:

  As a <type of user>, I want <some goal> so that <some reason>

- User stories are not formally documented, but stored in small containers, such as shoe-boxes.
- This method puts the emphasis on discussions.
- This way, it shifts the importance from writing features down to open dialogue which yields better results.



User Stories [2-2]



Agile Best Practices

Summary

- FDD is highly adaptive software that is iterative, pays attention to quality and provides working results at each step.
- There are five processes under FDD that are Develop an overall model, Build feature list, Plan by feature, Design by feature, and Build by feature.
- LSD is dedicated to reducing errors and wasted time while maximizing education and efficiency.
- There are seven principles under Lean Software Development namely, Waste elimination, Encouraging learning, Late decision-making, Fast delivery, Team empowerment, Built-in integrity, and Vision of the big picture.
- AUP is a simple and straightforward approach for developing business application software using agile techniques and concepts.
- There are four phases under AUP namely, Inception, Elaboration, Construction, and Transition.
- Some of the best practices under Agile are namely, Strategic planning, Organizational commitment and collaboration, Preparation, Execution, and Evaluation.