OBJECT ORIENTED SOFTWARE DEVELOPMENT

Subash Manandhar

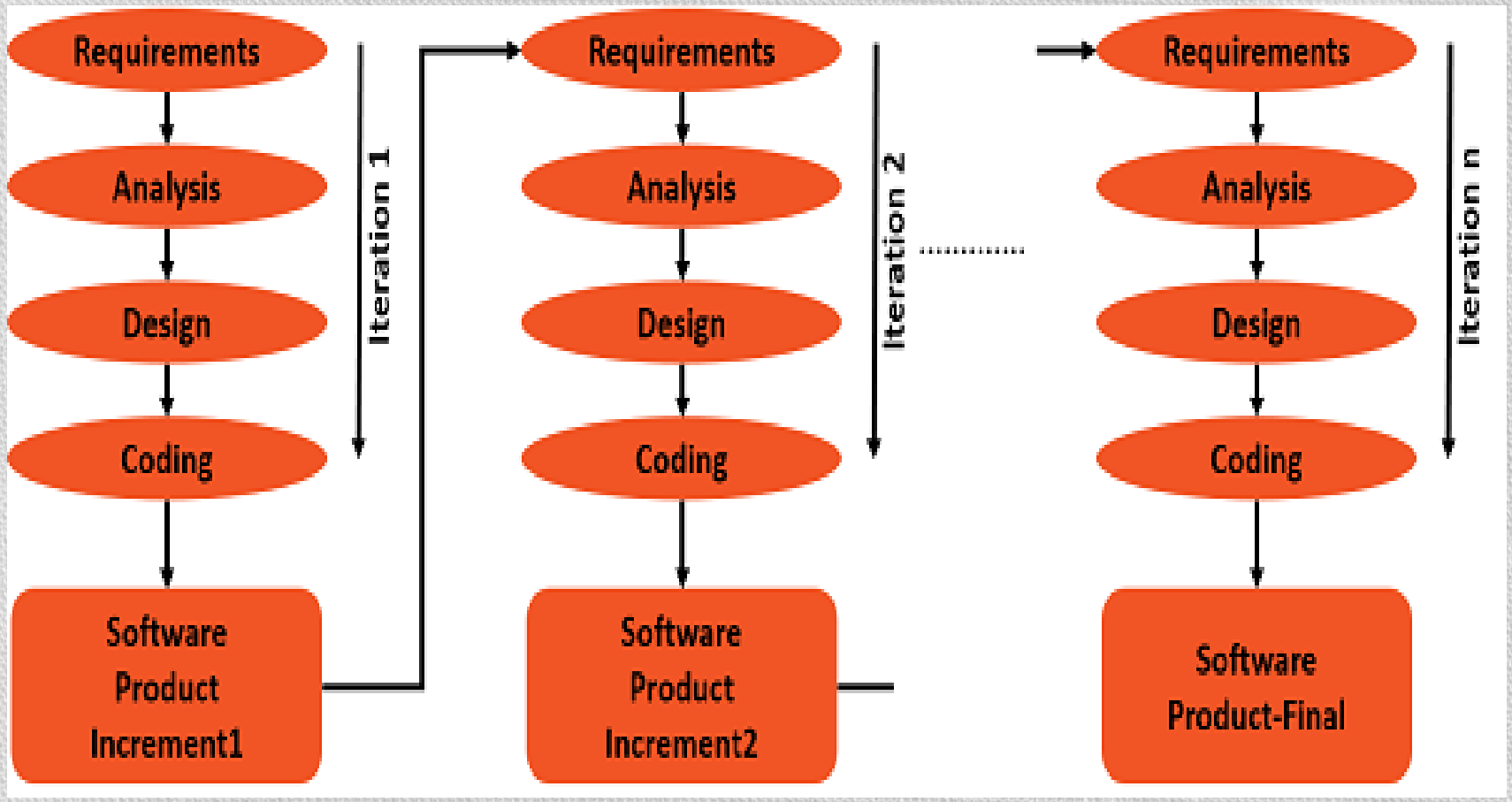
Chapter 2 – Iterative and Incremental Development

- Iterative and incremental software development is a method of software development that is modeled around a gradual increase in feature additions and a cyclical release and upgrade pattern.
- The outcome of the subsequent iteration is an enhanced working increment of the product. This is repeated until the product accomplishes the required functionalities.
- Iterative and incremental software development begins with planning and continues through iterative development cycles involving continuous user feedback and the incremental addition of features concluding with the deployment of completed software at the end of each cycle.
- It is one of the methodologies of Agile software development, rational unified process and extreme programming.
- ensures that the customer gets the desired feature/product with the desired level of quality.

Chapter 2 – Iterative and Incremental Development

- Iterative and incremental development is a discipline for developing systems based on producing deliverables.
- In incremental development, different parts of the system are developed at various times or rates and are integrated based on their completion.
- In iterative development, teams plan to revisit parts of the system in order to revise and improve them. User feedback is consulted to modify the targets for successive deliverables.
- Iterative and incremental software development came about in response to flaws in the waterfall model, a sequential design process in which progress flows steadily downwards.
- It differs from the waterfall model because it is cyclical rather than unidirectional, offering a greater ability to incorporate changes into the application during the development cycle.

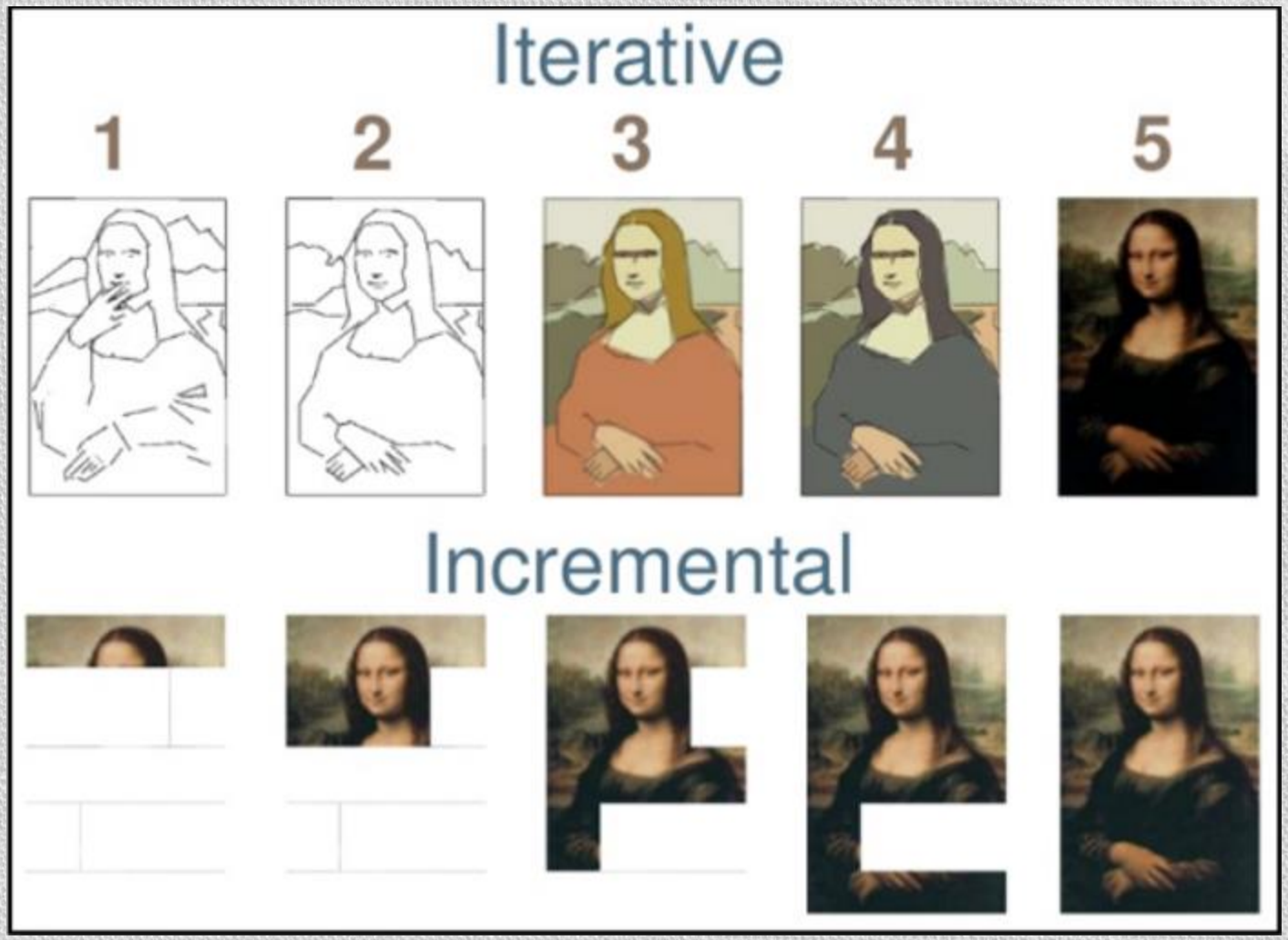
Chapter 2 – Iterative and Incremental Development



Chapter 2 – Iterative and Incremental Development

- **When to use iterative and incremental model?**
  - Most of the requirements are known up-front but are expected to evolve over time.
  - The requirements are prioritized.
  - There is a need to get the basic functionality delivered fast.
  - A project has lengthy development schedules.
  - A project has new technology.
  - The domain is new to the team.

Chapter 2 – Iterative and Incremental Development



Chapter 2 – Iterative and Incremental Development

- **e.g. Adding filter for searching restaurants**
- **Iteration 1:**
  - Provide the user with an option to search restaurants based on location
  - User can type in location to get a list of all the restaurants at that particular location. This is primary as user would generally look for places close to their location.
- **Iteration 2:**
  - Provide the user with an option to search restaurants based on types of dishes
  - User can search by the type of dishes that they like for eg: Pizza, Dosa etc.. This aids them to get a listing of all the restaurants that are serving the dishes of choice
- **Iteration 3:**
  - Provide the user with an option to search restaurants which provide home delivery

Chapter 2 – Iterative and Incremental Development

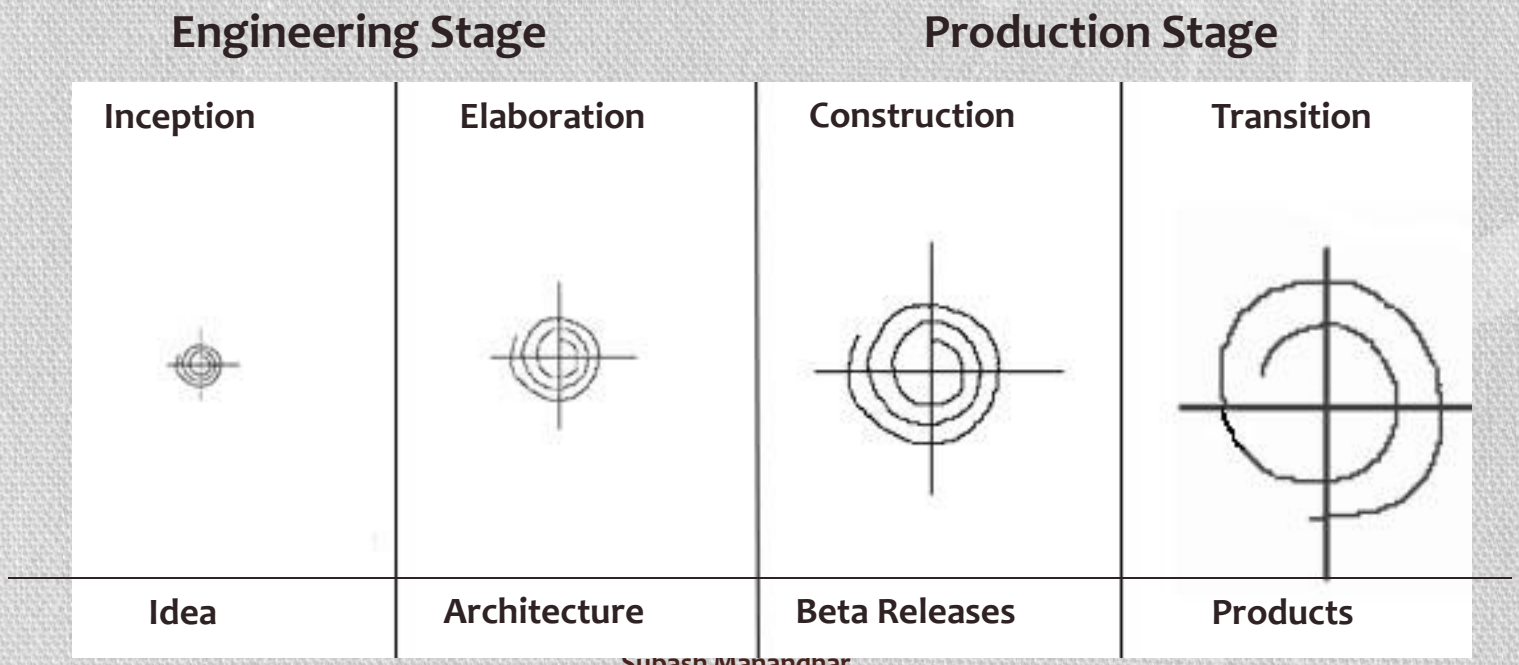
- **e.g. Online ordering of food**
- **Iteration 1(Increment 1):**
  - Provide support from website to order for food and pay online. This can have the home delivery support as well.
- **Iteration 2(Increment 2):**
  - Provide/implement an Android App and wallet integration. Also, show/provide recommendations based on user history
- **Iteration 3(Increment 3):**
  - Provide/implement IOS based App and tracking of order and notifications.

Chapter 2 – Iterative and Incremental Development

- **Phases**
  - **Inception Phase:**
    - Deals with the scope of the project, requirements and risks at higher levels
  - **Elaboration Phase:**
    - Delivers working architecture that moderates risks identified in the inception phase and satisfies nonfunctional requirements
  - **Construction Phase:**
    - Fills in architecture components incrementally with production-ready code, which is produced through the analysis, implementation, design and testing of functional requirements
  - **Transition Phase:**
    - Delivers the system to the production operating environment

Chapter 2 – Iterative and Incremental Development

- **Life Cycle Phase**
  - **The engineering stage**
    - driven by smaller teams doing design and synthesis activities.
  - **The production stage**
    - driven by larger teams doing construction, test, and deployment activities



Chapter 2 – Iterative and Incremental Development

Inception Phase

Overriding goal – to achieve concurrence among stakeholders on the life-cycle objectives

Essential activities :

- *Formulating the scope of the project* (capturing the requirements and operational concept in an information repository)
- *Synthesizing the architecture* (design trade-offs, problem space ambiguities, and available solution-space assets are evaluated)
- *Planning and preparing a business case* (alternatives for risk management, iteration planes, and cost/schedule/profitability trade-offs are evaluated)

Chapter 2 – Iterative and Incremental Development

Elaboration Phase

During the elaboration phase, an executable architecture prototype is built

Essential activities :

- *Elaborating the vision* (establishing a high-fidelity understanding of the critical use cases that drive architectural or planning decisions)
- *Elaborating the process and infrastructure* (establishing the construction process, the tools and process automation support)
- *Elaborating the architecture and selecting components* (lessons learned from these activities may result in redesign of the architecture)

Chapter 2 – Iterative and Incremental Development

Construction Phase

During the construction phase :

All remaining components and application features are integrated into the application

All features are thoroughly tested

Essential activities :

- *Resource management, control, and process optimization*
- *Complete component development and testing against evaluation criteria*
- *Assessment of the product releases against acceptance criteria of the vision*

Chapter 2 – Iterative and Incremental Development

Transition Phase

The transition phase is entered when baseline is mature enough to be deployed in the end-user domain

This phase could include beta testing, conversion of operational databases, and training of users and maintainers

Essential activities :

- *Synchronization and integration of concurrent construction into consistent deployment baselines*
- *Deployment-specific engineering* (commercial packaging and production, field personnel training)
- *Assessment of deployment baselines against the complete vision and acceptance criteria in the requirements set*

Chapter 2 - Iterative and Incremental Development

- **Unified Software Development Process:**
  - The Unified Software Development Process or Unified Process is a popular iterative and incremental software development process framework.
  - The best-known and extensively documented refinement of the Unified Process is the Rational Unified Process (RUP).
  - Other examples are OpenUP and Agile Unified Process.
  - The Unified Process is not simply a process, but rather an extensible framework which should be customized for specific organizations or projects.
  - The *Rational Unified Process* is, similarly, a customizable framework.
  - As a result it is often impossible to say whether a refinement of the process was derived from UP or from RUP, and so the names tend to be used interchangeably.

Chapter 2 - Iterative and Incremental Development

- **Unified Software Development Process:**
  - The Unified Process (UP) is a use-case-driven, architecture-centric, iterative and incremental development process framework that leverages the Object Management Group's (OMG) UML and is compliant with the OMG's SPEM.
  - The process consists of four main phases: ***inception***, ***elaboration***, ***construction*** and ***transition***.
  - Each of these contains one or more iterations across five core workflows: ***requirements***, ***analysis***, ***design***, ***implementation*** and ***test***.
  - The UP is broadly applicable to different types of software systems , including small-scale and large-scale projects having various degrees of managerial and technical complexity, across different application domains and organizational cultures.
  - The UP is an "idea," a process framework that provides an infrastructure for executing projects but not all of the details required for executing projects; essentially, it is a software development process framework, a lifecycle model involving context, collaborations, and interactions.

Chapter 2 - Iterative and Incremental Development

- **Unified Software Development Process:**
  - The Unified Process is a design framework which guides the tasks , people and products of the design.
  - It is a framework because it provides the inputs and outputs of each activity, but does not restrict how each activity must be performed.
  - It is called a process because its primary aim is to define:
    - Who is doing what.
    - When they do it.
    - How to reach a certain goal (i.e. each activity).
    - The inputs and outputs of each activity.

Chapter 2 - Iterative and Incremental Development

- **Unified Software Development Process:**
- The Unified Process actually comprises low-level activities (such as finding classes).
- which are combined together into disciplines (formerly known as workflows) which describe how one activity feeds into another).
- These disciplines are organized into iterations. Each iteration identifies some aspect of the system to be considered. How this is done is considered in more detail later.
- Iterations themselves are organized into phases. Phases focus on different aspects of the design process, for example requirements, analysis, design and implementation.
- In turn phases can be grouped into cycles . Cycles focus on the generation of successive releases of a system (for example, version 1.0, version 1.1 etc.).

Chapter 2 - Iterative and Incremental Development

- **Unified Software Development Process:**

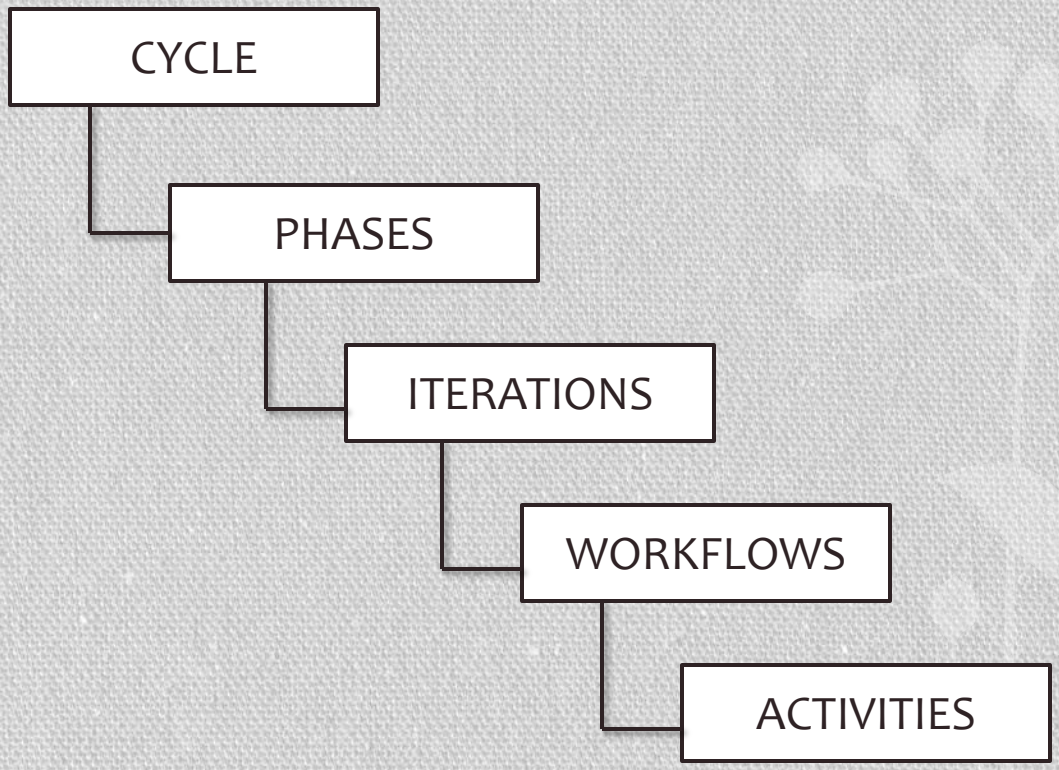


Fig : Key building blocks of the Unified Process.

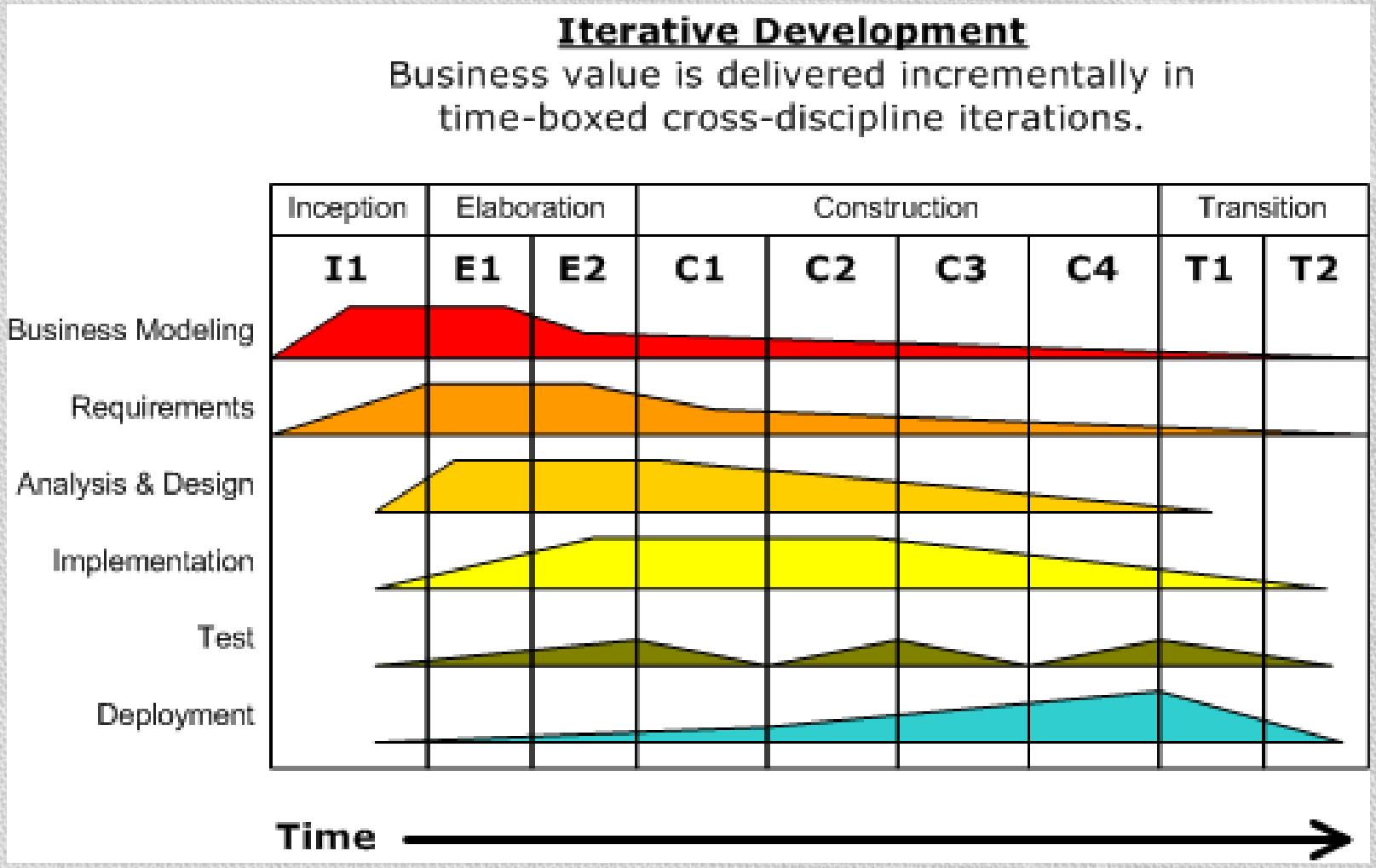
Chapter 2 - Iterative and Incremental Development

- **Characteristics of Unified Software Development Process:**
  - **Iterative and Incremental**
    - The Unified Process is an iterative and incremental development process.
    - The Elaboration, Construction and Transition phases are divided into a series of time boxed iterations.
    - The Inception phase may also be divided into iterations for a large project.
    - Each iteration results in an *increment*, which is a release of the system that contains added or improved functionality compared with the previous release.
    - Although most iterations will include work in most of the process disciplines (*e.g.* Requirements, Design, Implementation, Testing) the relative effort and emphasis will change over the course of the project.

Chapter 2 - Iterative and Incremental Development

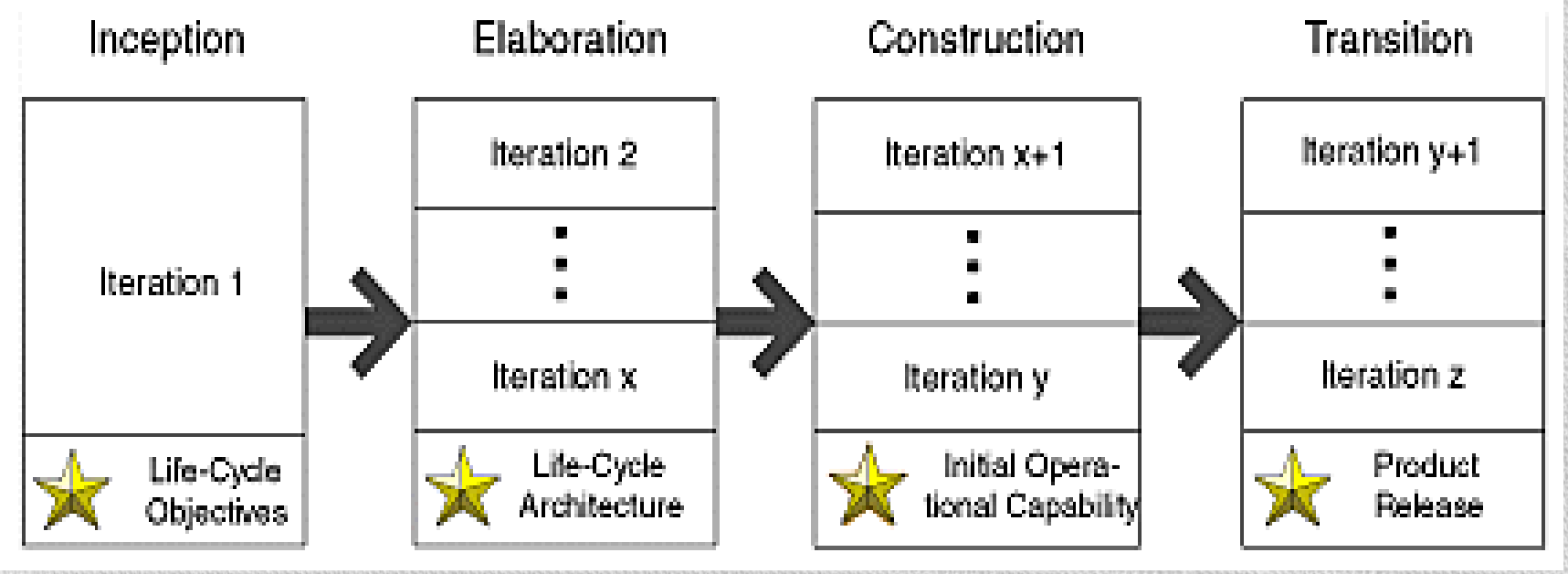
- **Characteristics of Unified Software Development Process:**
  - **Architecture Centric**
    - The Unified Process insists that architecture sit at the heart of the project team's efforts to shape the system.
    - Since no single model is sufficient to cover all aspects of a system, the Unified Process supports multiple architectural models and views.
    - One of the most important deliverables of the process is the executable architecture baseline which is created during the Elaboration phase.
    - This partial implementation of the system serves to validate the architecture and act as a foundation for remaining development.
  - **Risk Focused**
    - The Unified Process requires the project team to focus on addressing the most critical risks early in the project life cycle.
    - The deliverables of each iteration, especially in the Elaboration phase, must be selected in order to ensure that the greatest risks are addressed first.

Chapter 2 - Iterative and Incremental Development



Chapter 2 - Iterative and Incremental Development

- **Life Cycle of UP**
  - The lifecycle describes the time dimension of project, that is, how a project is divided into phases and iterations.
  - It divides a project into four **phases: Inception, Elaboration, Construction,** and **Transition,** each ending with a well-defined milestone.
  - Each phase has specific objectives.



Chapter 2 - Iterative and Incremental Development

- **Inception:**
  - Establish the scope of the system, including a good understanding of what system to build, by reaching a high-level understanding of the requirements.
  - Mitigate many of the business risks and produce the business case for building the system and a vision document to get buy-in from all stakeholders on whether or not to proceed with the project.
  - This is similar to what many agile processes refer to as *Iteration 0*.

Chapter 2 - Iterative and Incremental Development

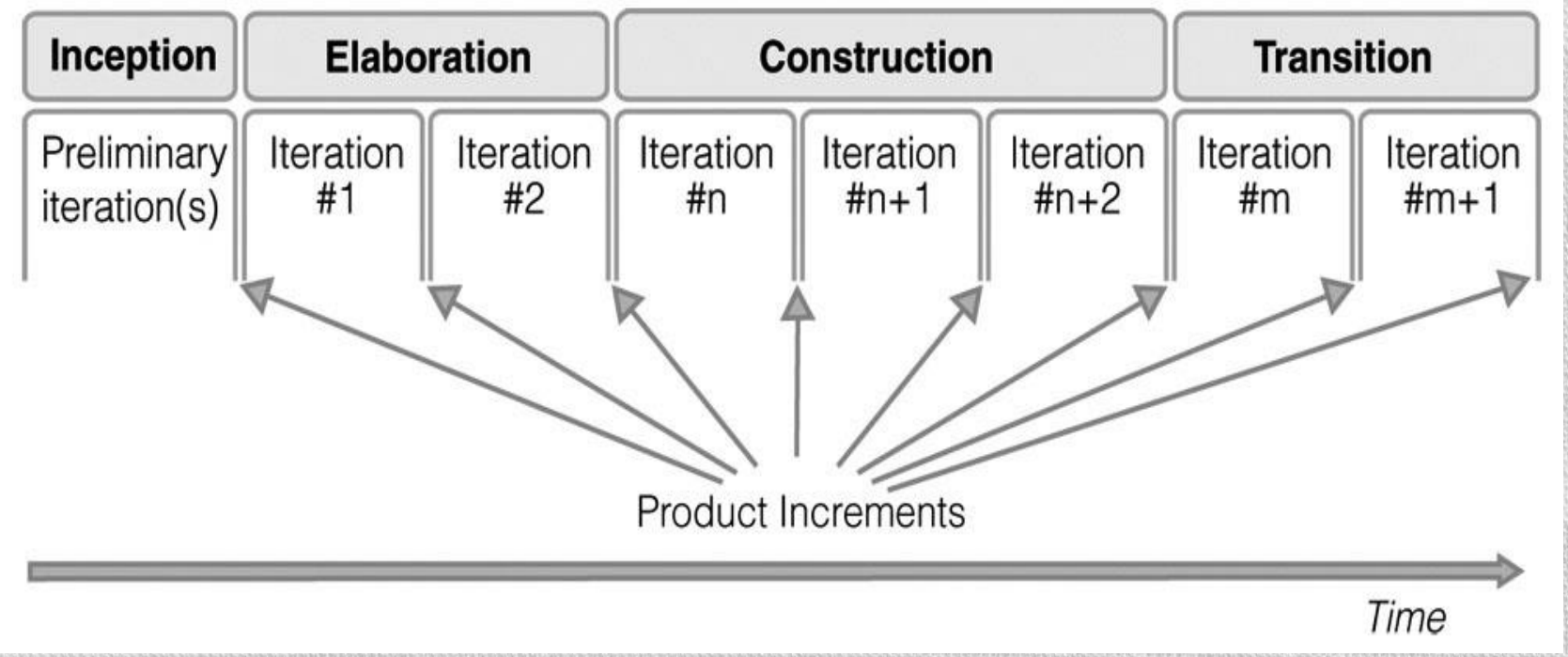
- **Elaboration:**
  - Reduce major risks to enable cost and schedule estimates to be updated and to get buy-in from key stakeholders.
  - Mitigate major technical risks by taking care of many of the most technically difficult tasks.
  - Design, implement, test, and baseline an executable **architecture,** including subsystems, their interfaces, key components, and architectural mechanisms such as how to deal with interprocess communication or persistency.
  - Address major business risks by defining, designing, implementing, and testing key capabilities, which are validated with the customer.
  - Do not define and analyze all requirements at this time, as doing so would lead to waterfall development.
  - Detail and analyze only the requirements required to address the above risks.

Chapter 2 - Iterative and Incremental Development

- **Construction:**
  - Undertake a majority of the implementation as you move from an executable architecture to the first operational version of your system.
  - Deploy several internal and alpha releases to ensure that the system is usable and addresses user needs.
  - End the phase by deploying a fully functional beta version of the system, including installation and supporting documentation and training material (although the system will likely still require fine-tuning of functionality, performance, and overall quality).
- **Transition:**
  - Ensure that software addresses the needs of its users by testing the product in preparation for release and making minor adjustments based on user feedback.
  - At this point in the lifecycle, user feedback focuses mainly on fine-tuning, configuration, installation, and usability issues; all the major structural issues should have been worked out much earlier in the project lifecycle.

Chapter 2 - Iterative and Incremental Development

- **Life Cycle of UP**



Chapter 2 - Iterative and Incremental Development

- **Inception Phase:**
  - The outcome of the inception phase is:
    - A vision document: a general vision of the core project's requirements, key features, and main constraints.
    - A initial use-case model (10% -20%) complete).
    - An initial project glossary (may optionally be partially expressed as a domain model).
    - An initial business case, which includes business context, success criteria (revenue projection, market recognition, and so on), and financial forecast.
    - An initial risk assessment.
    - A project plan, showing phases and iterations.
    - A business model, if necessary.
    - One or several prototypes.

Chapter 2 - Iterative and Incremental Development

- **Inception Phase:**
  - At the end of the inception phase is the first major project milestone: the Lifecycle Objectives Milestone.
  - The evaluation criteria for the inception phase are:
    - Stakeholder concurrence on scope definition and cost/schedule estimates.
    - Requirements understanding as evidenced by the fidelity of the primary use cases.
    - Credibility of the cost/schedule estimates, priorities, risks, and development process.
    - Depth and breadth of any architectural prototype that was developed.
    - Actual expenditures versus planned expenditures.

Chapter 2 - Iterative and Incremental Development

- **Elaboration Phase:**
  - The outcome of the elaboration phase is:
    - A use-case model (at least 80% complete) — all use cases and actors have been identified, and most usecase descriptions have been developed.
    - Supplementary requirements capturing the non functional requirements and any requirements that are not associated with a specific use case.
    - A Software Architecture Description.
    - An executable architectural prototype.
    - A revised risk list and a revised business case.
    - A development plan for the overall project, including the coarse-grained project plan, showing iterations" and evaluation criteria for each iteration.
    - An updated development case specifying the process to be used.
    - A preliminary user manual (optional).

Chapter 2 - Iterative and Incremental Development

- **Elaboration Phase:**
  - At the end of the elaboration phase is the second important project milestone, the Lifecycle Architecture Milestone.
  - The main evaluation criteria for the elaboration phase involves the answers to these questions:
    - Is the vision of the product stable?
    - Is the architecture stable?
    - Does the executable demonstration show that the major risk elements have been addressed and credibly resolved?
    - Is the plan for the construction phase sufficiently detailed and accurate? Is it backed up with a credible basis of estimates?
    - Do all stakeholders agree that the current vision can be achieved if the current plan is executed to develop the complete system, in the context of the current architecture?
    - Is the actual resource expenditure versus planned expenditure acceptable?

Chapter 2 - Iterative and Incremental Development

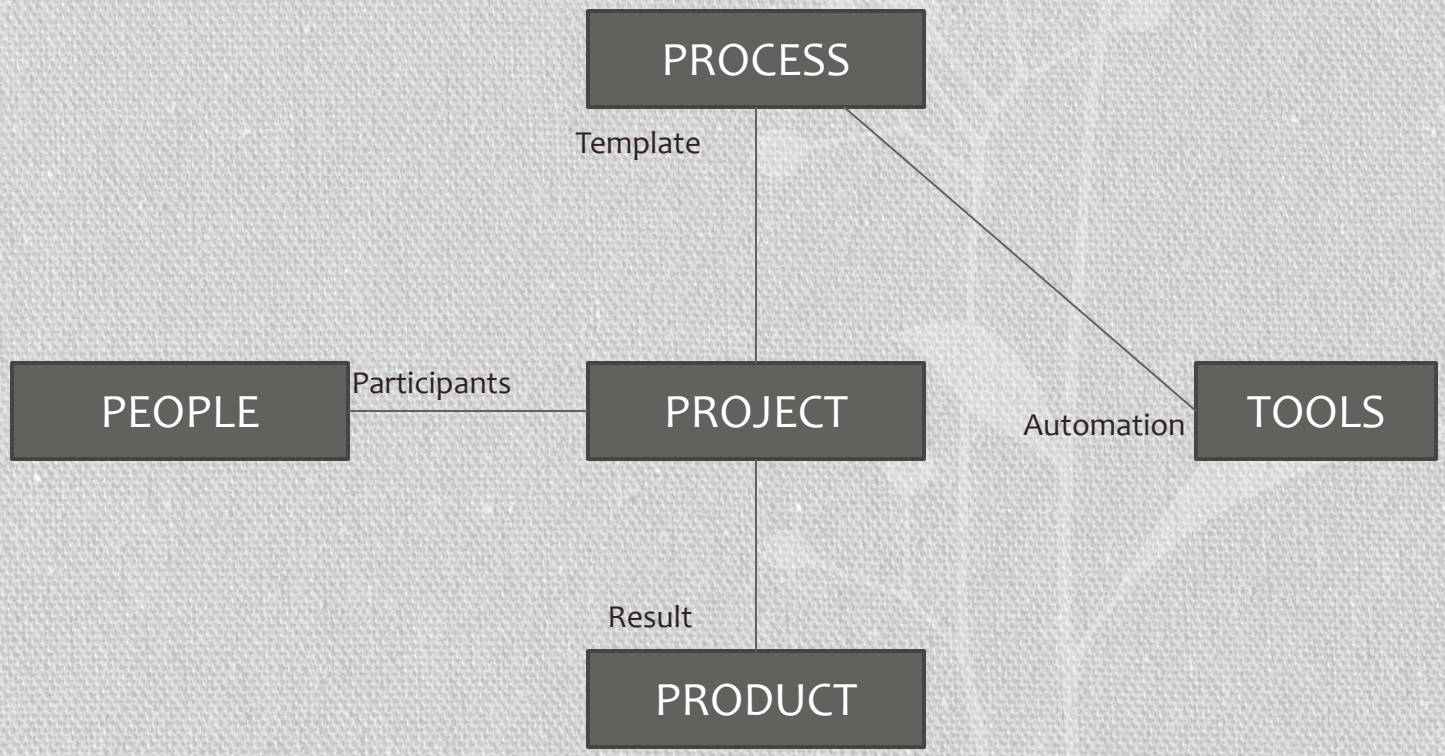
- **Construction Phase:**
  - The outcome of the construction phase is a product ready to put in hands of its end-users.
  - At minimum, it consists of:
    - The software product integrated on the adequate platforms.
    - The user manuals.
    - A description of the current release.
  - At the end of the construction phase is the third major project milestone (Initial Operational Capability Milestone).
  - The evaluation criteria for the construction phase involve answering these questions:
    - • Is this product release stable and mature enough to be deployed in the user community?
    - • Are all stakeholders ready for the transition into the user community?
    - • Are the actual resource expenditures versus planned expenditures still acceptable?

Chapter 2 - Iterative and Incremental Development

- **Transition Phase:**
  - The primary objectives of the transition phase include:
    - Achieving user self-supportability
    - Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision
    - Achieving final product baseline as rapidly and cost effectively as practical
  - At the end of the transition phase is the fourth important project milestone, the Product Release Milestone.
  - The primary evaluation criteria for the transition phase involve the answers to these questions:
    - Is the user satisfied?
    - Are the actual resources expenditures versus planned expenditures still acceptable?

Chapter 2 - Iterative and Incremental Development

- **Four *P*s in software development**
  - People
  - Project
  - Product
  - Process



Chapter 2 - Iterative and Incremental Development

- **Four *P*s in software development**
- **People:**
  - The architects, developers, testers, and their supporting management, plus users, customers, and other stakeholders are the prime movers in a software project.
- **Project:**
  - The organizational element through which software development is managed.
  - Outcome of project is a released product.
- **Product:**
  - Artifacts that are created during the life of the project such as models, source code, executables and documentations
- **Process:**
  - Definition of complete set of activities needed to transform user's requirements into a product.
  - Is a template for creating projects.
- **Tools:**
  - Software that is used to automate the activities defined in process.

Chapter 2 - Iterative and Incremental Development

- **Four *P*s in software development**
- **People:**
  - Throughout the entire life cycle of software development, people are involved in one or another way.
  - So the process that guides this development must be people oriented.
    - Development process affect people
      - project feasibility
      - Risk management
      - Team structure
      - Project schedule
      - Project understandability
      - Sense of accomplishment
    - Roles will change
      - Uniform development process enables developers to build better software in terms of time to market, quality and cost, also to select an suitable architecture.
      - Here, developers will find themselves working with many other developers.
      - Providing guidance will result in developers "working smarter".
      - To develop more effective software product, choosing right people make them effective and allow them to do what human can.

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Project:**
  - Results in a new release of a product.
  - First project in life cycle develops and release initial system.
  - Successive project cycles extend the life of the system over many releases.
  - Throughout life cycle, a project team has to be concerned with change , iterations and the organizational pattern with in which the project is conducted.
    - A sequence of change
      - Every cycle, phase and every iteration changes the system from one thing to something else.
      - Each cycle leads to a ***release*** and changes continues for ***generations***.
    - A series of iterations
      - With in each phase of cycle, series of iterations of activities are carried.
      - Each iteration implements set of use case or mitigates some risks.
      - An iteration can be thought as ***miniproject.***
    - An organizational pattern
      - There is a pattern within which people executes a project.
      - The pattern indicates the type of workers the project needs and the artifacts with which it is to work.

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Product:**
  - Is whole software system, not just code.
  - Software can be described as a binary form that can be read and understood by a computer.
  - It is a description that is written by programmers that can be read and understood by a compiler.
  - Software design in terms of subsystems, classes, interaction diagrams and other artifacts and also requirements, testing, sales etc can be viewed as part of system.
  - So, a system is all the artifacts that it take to represent it in machine or human readable form to the machines, the workers, and the stakeholders

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Product:**
  - Artifacts is a general term for any kind of information created , produced , changed or used by workers in developing system.
  - E.g. UML diagrams, prototypes, components, test plans and test procedures.
  - Two types:
    - Engineering artifacts
      - Created during various phases of process.
    - Management artifacts
      - Have a short lifetime.
      - Like business case, development plans etc.
      - Also include specifications of the development environment.

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Product:**
  - Model is one of artifacts employed in Unified Process.
  - Building a system is thus a process of model building using different models to describe all the different perspective of the system.
  - UP provides a carefully selected set of models with which to start to satisfy those worker's need for information.
  - Model is an abstraction of a system that the architects and developers build.
  - Each model is a self contained view of the system.
  - Most engineering models are defined by a carefully selected subset of UML.
  - Like use case model consisting of use cases and actors.
  - There is a hierarchy of elements in model.
  - The system is not just collection of models but also the relationships between them as well.
  - Trace relationship between elements in different models add no semantic information to help understand the related model themselves, they just connect the models.

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Process:**
  - In UP process refers to a concept that works as a template that can be reused by creating instances of it.
  - Is compare to class in object oriented paradigm from which objects can be created.
  - Here, process instance represents *project*
  - *Software development process is a definition of complete set of activities needed to transform user's requirements into a consistent set of artifacts that represents a software product and later to transform changes in those requirements into a new, consistent set of artifacts.*
  - The value added result of the process is a consistent set of artifacts.
  - A process is a definition of a set of activities ,not their execution.

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Process:**
  - We can describe a process in terms of workflows, where a workflow is a set of activities.
  - In UML, a workflow is a stereotype collaboration in which workers and artifacts are participants.
  - E.g. requirement workflow
    - Here, workers are: system analyst, architect , use case specifier and user interface designer .
    - Artifacts are: use case models, use cases and others.
  - No single software development process can be applied everywhere.
  - Unified process is a generic process that is process framework.
  - UP may be specialized to fit different applications and organizational needs.

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Process:**
  - It is also desirable with in organization, the process be fairly consistent.
  - This consistency will allow components to be used interchangeably, people and managers to transfer between projects readily and accomplishment metrics to be comparable.
  - Factors that influence how process will differ:
    - *Organizational factors*
    - *Domain factors*
    - *Life cycle factors*
    - *Technical factors*

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

- **Process:**
  - Benefits:
    - Everyone on the development team can understand what to do.
    - Developers can better understand what other developers are doing.
    - Supervisors and Managers even those who cannot read code, can, thanks to architectural drawing, understand what developers are doing.
    - Developers , supervisors and managers can transfer between projects or divisions without having learn a new process.
    - Training can be standardized within a company.
    - The course of software development is repeatable.

Chapter 2 - Iterative and Incremental Development

Four *P*s in software development

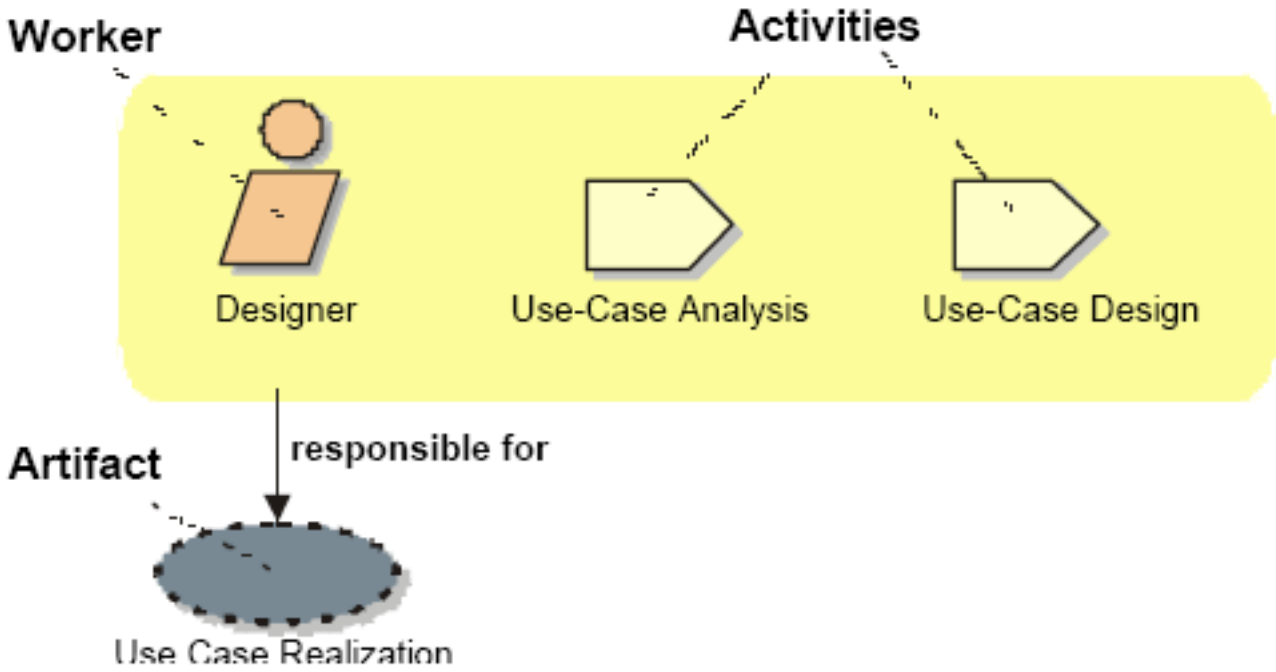
- **Tools:**
  - Tools are integral to the process.
  - Are good at automating repetitive tasks, keeping things structured, managing large amount of information and guiding along a particular development path.
  - The tools that implement an automated process should be easy to use.
  - There has to be balance between process and tools.
  - On one hand ,process drives tool development.
  - On the other hand, tools guide process development.
  - *Successful development of process automation (tools) cannot be achieved without the parallel development of the process framework in which the tools are to function.*

Chapter 2 - Iterative and Incremental Development

- **Static structure of process:**
  - A process describes *who* is doing *what*, *how*, and *when*. The Rational Unified Process is represented using four primary modeling elements:
    - Workers, the 'who'
    - Activities, the 'how'
    - Artifacts, the 'what'
    - Workflows, the 'when'

Chapter 2 - Iterative and Incremental Development

- **Workers , Activities and artifacts**



Workers, activities and artifacts.

Chapter 2 - Iterative and Incremental Development

- **Workers:**
  - A ***worker*** defines the behavior and responsibilities of an individual, or a group of individuals working together as a team.
  - Worker is like a "hat" an individual can wear in the project.
  - One individual may wear many different hats.
  - This is an important distinction because it is natural to think of a worker as the individual or team itself, but in the Unified Process the worker is more the role defining how the individuals should carry out the work.
  - The responsibilities we assign to a worker includes both to perform a certain set of activities as well as being owner of a set of artifacts.

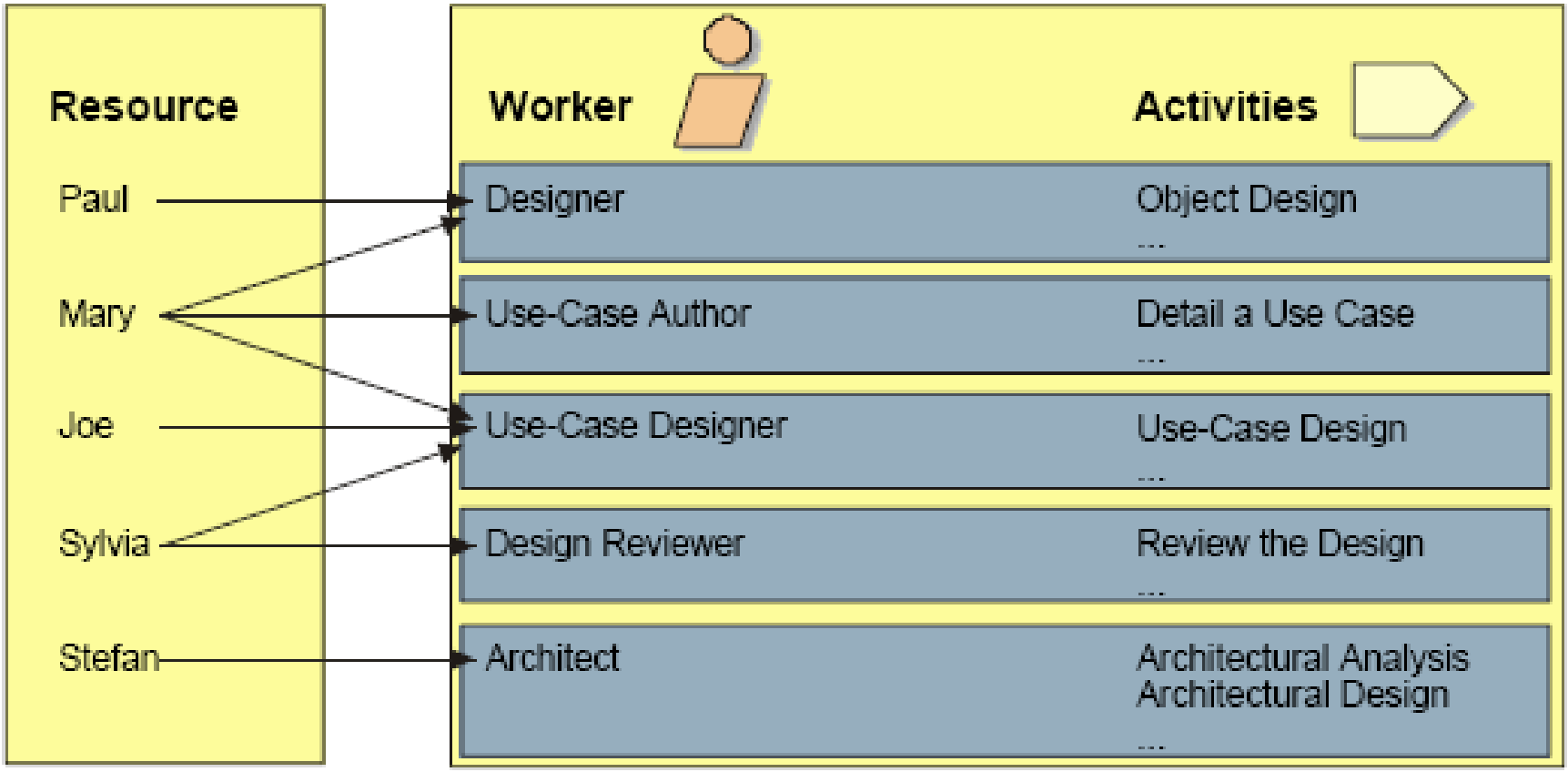
Chapter 2 - Iterative and Incremental Development

- **Activity:**
  - An **activity** of a specific worker is a unit of work that an individual in that role may be asked to perform.
  - The activity has a clear purpose, usually expressed in terms of creating or updating some artifacts, such as a model, a class, a plan.
  - Every activity is assigned to a specific worker.
  - The granularity of an activity is generally a few hours to a few days, it usually involves one worker, and affects one or only a small number of artifacts.
  - An activity should be usable as an element of planning and progress; if it is too small, it will be neglected, and if it is too large, progress would have to be expressed in terms of an activity's parts.

Chapter 2 - Iterative and Incremental Development

- **Activity:**
- Example of activities:
  - **Plan an iteration,** for the Worker: Project Manager
  - **Find use cases and actors,** for the Worker: System Analyst
  - **Review the design,** for the Worker: Design Reviewer
  - **Execute performance test,** for the Worker: Performance Tester

Chapter 2 - Iterative and Incremental Development



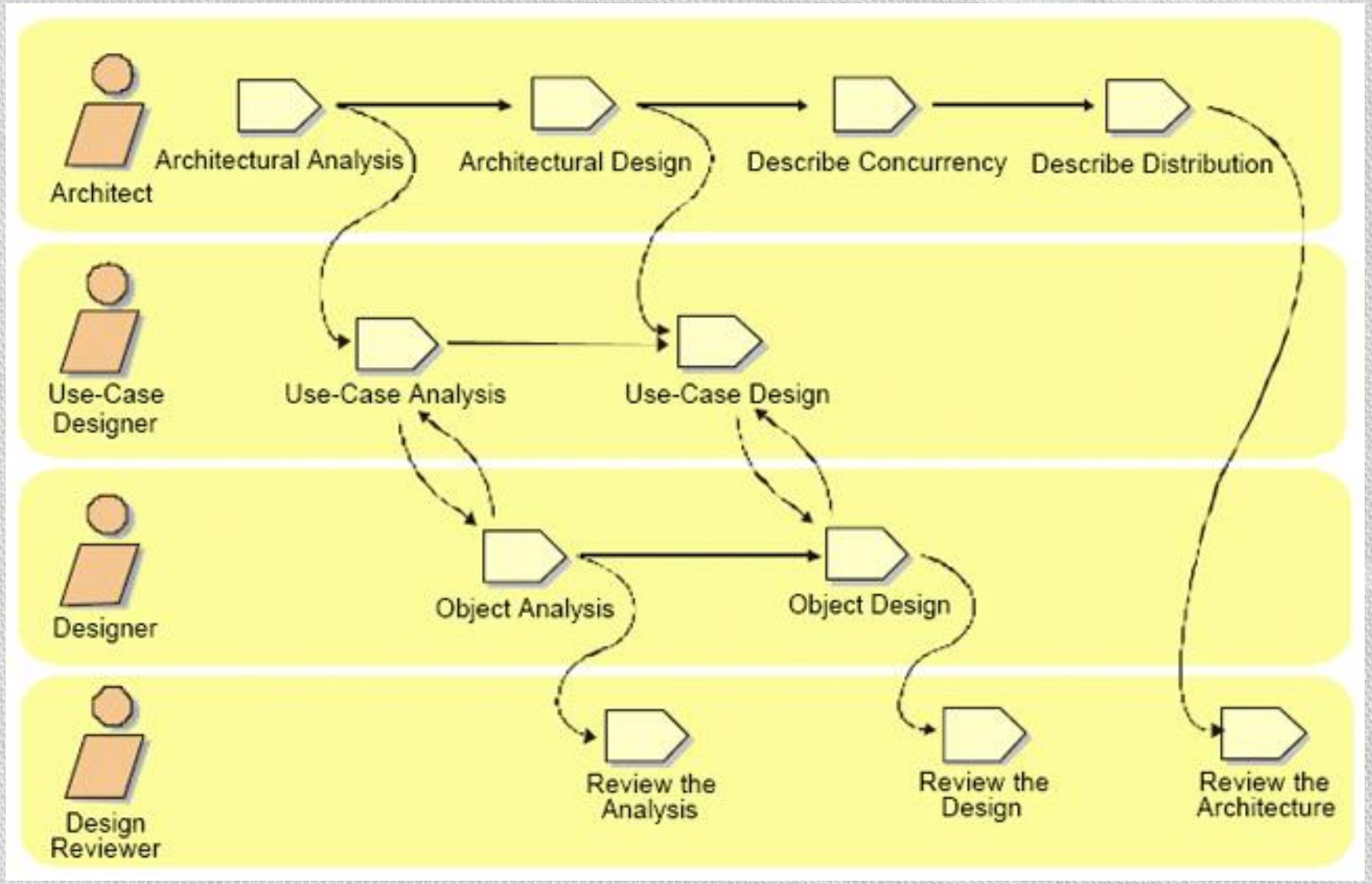
Chapter 2 - Iterative and Incremental Development

- **Artifacts:**
  - An artifact is a piece of information that is produced, modified, or used by a process.
  - Artifacts are the tangible products of the project, the things the project produces or uses while working towards the final product.
  - Artifacts are used as input by workers to perform an activity, and are the result or output of such activities.
  - In object-oriented design terms, as activities are operations on an active object (the worker), artifacts are the parameters of these activities.
  - Artifacts may take various shapes or forms:
  - A model, such as the Use-Case Model or the Design Model
  - A model element, i.e. an element within a model, such as a class, a use case or a subsystem
  - A document, such as Business Case or Software Architecture Document
  - Source code
  - Executables

Chapter 2 - Iterative and Incremental Development

- **Workflows:**
  - A mere enumeration of all workers, activities and artifacts does not quite constitute a process.
  - We need a way to describe meaningful sequences of activities that produce some valuable result, and to show interactions between workers.
  - A *workflow* is a sequence of activities that produces a result of observable value.
  - In UML terms, a workflow can be expressed as a sequence diagram, a collaboration diagram, or an activity diagram.
  - Note that it is not always possible or practical to represent all of the dependencies between activities.
  - Often two activities are more tightly interwoven than shown, especially when they involve the same worker or the same individual.
  - People are not machines, and the workflow cannot be interpreted literally as a program for people, to be followed exactly and mechanically.

Chapter 2 - Iterative and Incremental Development

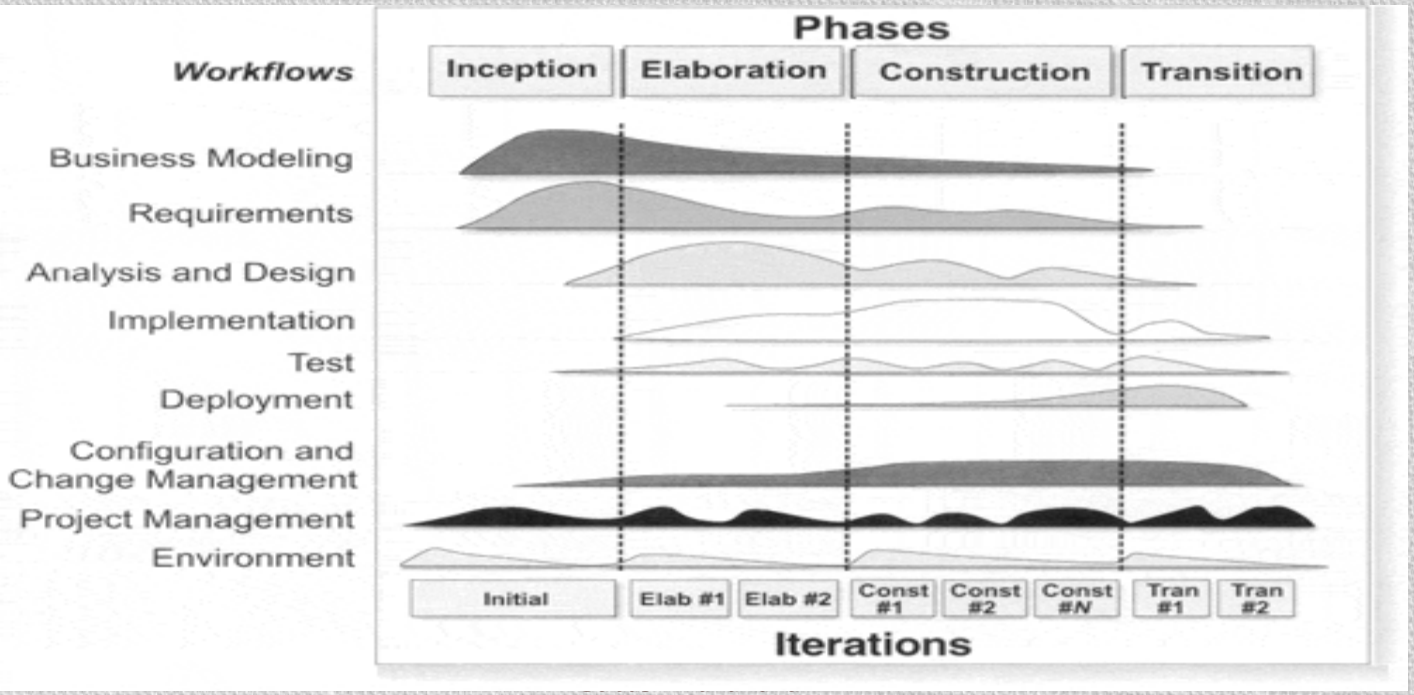


Chapter 2 - Iterative and Incremental Development

- **Core Workflows:**
  - There are nine *core process workflows* in the Rational Unified Process, which represent a partitioning of all workers and activities into logical groupings.
  - The core process workflows are divided into six core "engineering" workflows:
    - Business modeling workflow
    - Requirements workflow
    - Analysis & Design workflow
    - Implementation workflow
    - Test workflow
    - Deployment workflow
  - And three core "supporting" workflows:
    - Project Management workflow
    - Configuration and Change Management workflow
    - Environment workflow

Chapter 2 - Iterative and Incremental Development

- **Core Workflows:**
  - Although the names of the six core engineering workflows may evoke the sequential phases in a traditional waterfall process, we should keep in mind that the phases of an iterative process are different and that these workflows are revisited again and again throughout the lifecycle.
  - The actual complete workflow of a project interleaves these nine core workflows, and repeats them with various emphasis and intensity at each iteration.



Chapter 2 - Iterative and Incremental Development
Core Workflows

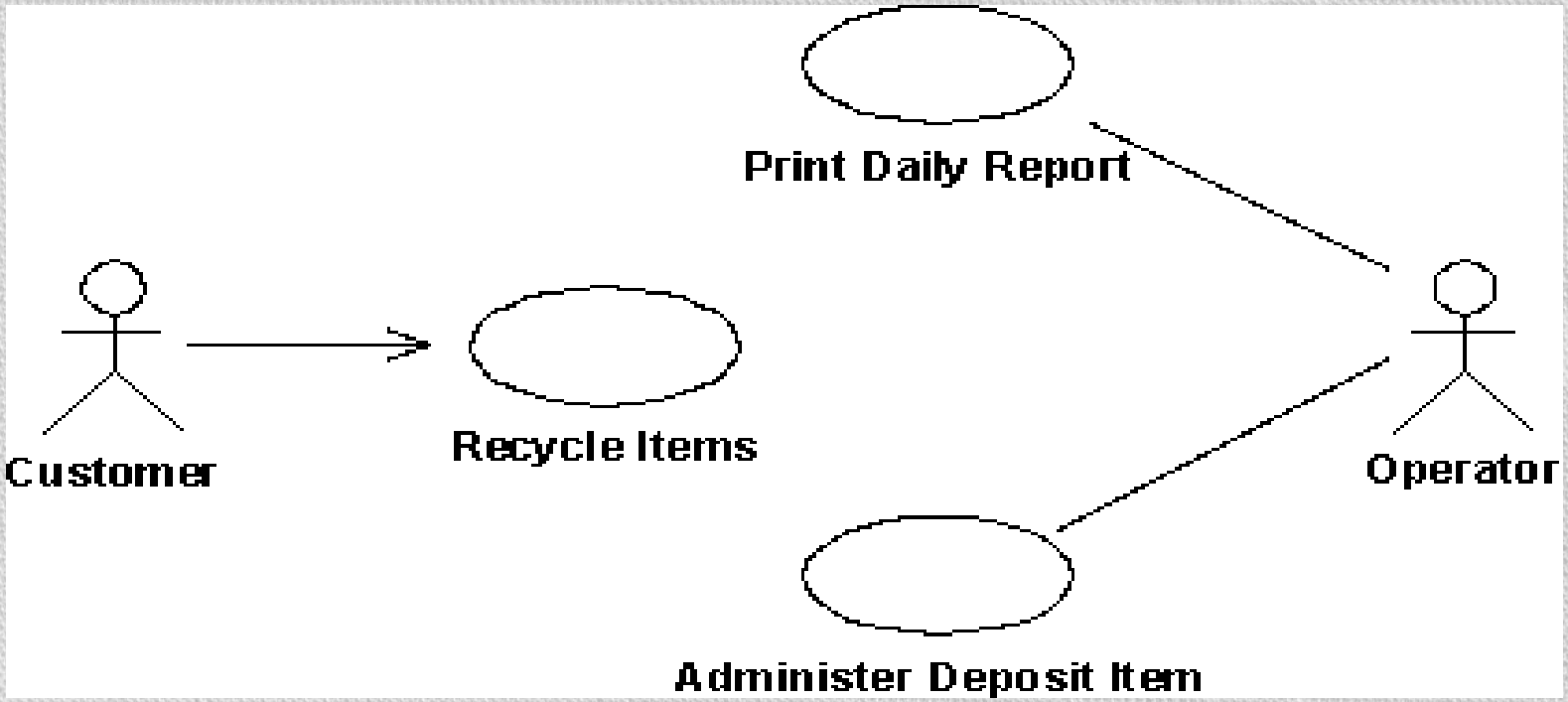
- **Business Modeling:**
  - One of the major problems with most business engineering efforts, is that the software engineering and the business engineering community do not communicate properly with each other.
  - This leads to the output from business engineering is not being used properly as input to the software development effort, and vice-versa.
  - The Rational Unified Process addresses this by providing a common language and process for both communities, as well as showing how to create and maintain direct traceability between business and software models.
  - In Business Modeling we document business processes using so called business use cases.
  - This assures a common understanding among all stakeholders of what business process needs to be supported in the organization.
  - The business use cases are analyzed to understand how the business should support the business processes.
  - This is documented in a business object-model.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Requirements:**
  - The goal of the Requirements workflow is to describe *what* the system should do and allows the developers and the customer to agree on that description.
  - To achieve this, we elicit, organize, and document required functionality and constraints; track and document tradeoffs and decisions.
  - A Vision document is created, and stakeholder needs are elicited.
  - *Actors* are identified, representing the users, and any other system that may interact with the system being developed.
  - *Use cases* are identified, representing the behavior of the system.
  - Because use cases are developed according to the actor's needs, the system is more likely to be relevant to the users.

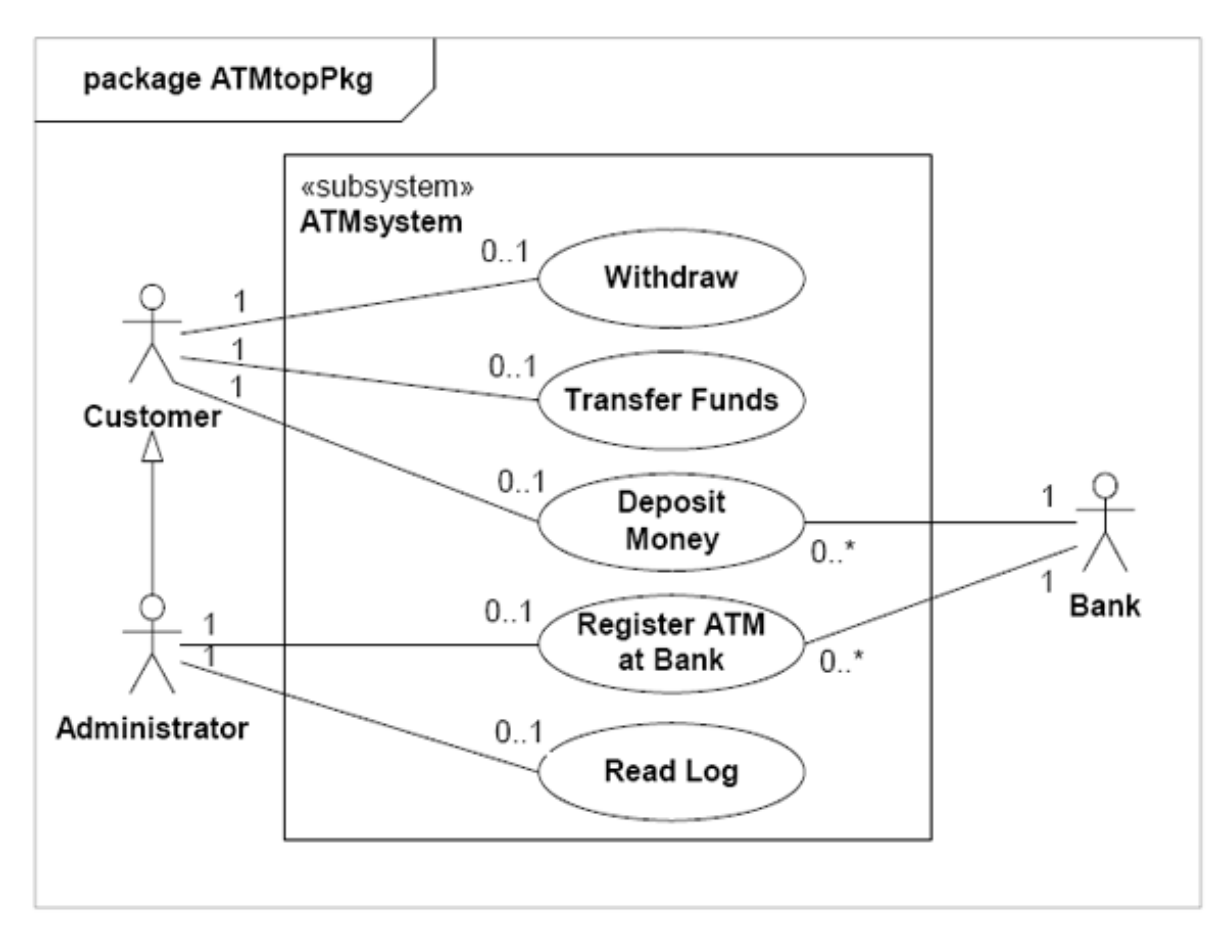
Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Requirements:**
- The following figure shows an example of a use-case model for a recycling-machine system.



Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Requirements:**



Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Requirements:**
  - Each use case is described in detail.
  - The *use-case description* shows how the system interacts step by step with the actors and what the system does.
  - Non-functional requirements are described in *Supplementary* Specifications.
  - The use cases function as a unifying thread throughout the system's development cycle.
  - The same use-case model is used during requirements capture, analysis & design, and test.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Analysis and Design:**
  - The goal of the Analysis & Design workflow is to show *how* the system will be *realized* in the implementation phase.
  - Analysis & Design results in a *design model* and optionally an *analysis model*.
  - The design model serves as an abstraction of the source code; that is, the design model acts as a 'blueprint' of how the source code is structured and written.
  - The design model consists of design classes structured into design packages and design subsystems with well defined interfaces, representing what will become components in the implementation.
  - It also contains descriptions of how objects of these design classes collaborate to perform use cases.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Analysis and Design:**
  - The design activities are centered around the notion of *architecture.*
  - The production and validation of this architecture is the main focus of early design iterations.
  - Architecture is represented by a number of architectural views .
  - These views capture the major structural design decisions.
  - In essence, architectural views are abstractions or simplifications of the entire design, in which important characteristics are made more visible by leaving details aside.
  - The architecture is an important vehicle not only for developing a good design model, but also for increasing the quality of any model built during system development.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Implementation:**
  - The purpose of implementation is:
    - To define the organization of the code, in terms of implementation subsystems organized in layers.
    - To implement classes and objects in terms of components (source files, binaries, executables, and others).
    - To test the developed components as units.
    - To integrate the results produced by individual implementers (or teams), into an executable system.
  - The system is realized through implementation of components.
  - The Rational Unified Process describes how you reuse existing components, or implement new components with well defined responsibility, making the system easier to maintain, and increasing the possibilities to reuse.
  - Components are structured into Implementation Subsystems.
  - Subsystems take the form of directories, with additional structural or management information.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Test:**
  - The Rational Unified Process proposes an iterative approach, which means that you test throughout the project.
  - This allows you to find defects as early as possible, which radically reduces the cost of fixing the defect.
  - Tests are carried out along three quality dimensions reliability, functionality, application performance and system performance.
  - For each of these quality dimensions, the process describes how you go through the test lifecycle of planning, design, implementation, execution and evaluation.
  - Strategies for when and how to automate test are described.
  - Test automation is especially important using an iterative approach, to allow regression testing at then end of each iteration, as well as for each new version of the product.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Test:**
  - The purposes of testing are:
    - To verify the interaction between objects.
    - To verify the proper integration of all components of the software.
    - To verify that all requirements have been correctly implemented.
    - To identify and ensure defects are addressed prior to the deployment of the software.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Deployment:**
  - The purpose of the deployment workflow is to successfully produce product releases, and deliver the software to its end users.
  - It covers a wide range of activities including:
    - Producing external releases of the software.
    - Packaging the software.
    - Distributing the software.
    - Installing the software.
    - Providing help and assistance to users.
    - In many cases, this also includes activities such as:
    - Planning and conduct of beta tests.
    - Migration of existing software or data.
    - Formal acceptance.
  - Although deployment activities are mostly centered around the transition phase, many of the activities need to be included in earlier phases to prepare for deployment at the end of the construction phase.
  - The Deployment and Environment workflows of the Rational Unified Process contain less detail than other workflows.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Project Management:**
  - Software Project Management is the art of balancing competing objectives, managing risk, and overcoming constraints to deliver, successfully, a product in which meets the needs of both customers (the payers of bills) and the users.
  - The fact that so few projects are unarguably successful is comment enough on the difficulty of the task.
  - This workflow focuses mainly on the specific aspect of an iterative development process.
  - Goal with this section is to make the task easier by providing:
    - A framework for managing software-intensive projects.
    - Practical guidelines for planning, staffing, executing, and monitoring projects.
    - A framework for managing risk.
  - It is not a recipe for success, but it presents an approach to managing the project that will markedly improve the odds of delivering successful software.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Configuration and Change Management:**
  - describe how to control the numerous artifacts produced by the many people who work on a common project.
  - Control helps avoid costly confusion, and ensures that resultant artifacts are not in conflict due to some of the following kinds of problems:
    - Simultaneous Update =>When two or more workers work separately on the same artifact, the last one to make changes destroys the work of the former.
    - Limited Notification => When a problem is fixed in artifacts shared by several developers, and some of them are not notified of the change.
    - Multiple Versions => Most large programs are developed in evolutionary releases. One release could be in customer use, while another is in test, and the third is still in development. If problems are found in any one of the versions, fixes need to be propagated between them. Confusion can arise leading to costly fixes and re-work unless changes are carefully controlled and monitored.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Configuration and Change Management:**
  - This workflow provides guidelines for managing multiple variants of evolving software systems, tracking which versions are used in given software builds, performing builds of individual programs or entire releases according to user-defined version specifications, and enforcing site-specific development policies.
  - This workflow also covers change request management, i.e. how to report defects, manage them through their lifecycle, and how to use defect data to track progress and trends.

Chapter 2 - Iterative and Incremental Development
Core Workflows

- **Environment:**
  - The purpose of the environment workflow is to provide the software development organization with the software development environment—both processes and tools—that are needed to support the development team.
  - This workflow focuses on the activities to configure the process in the context of a project.
  - It also focuses on activities to develop the guidelines needed to support a project.
  - A step-by-step procedure is provided describing how you implement a process in an organization.

Chapter 2 - Iterative and Incremental Development

- **Effective Deployment of 6 Best Practices:**
  - Develop software iteratively
  - Manage requirements
  - Use component-based architectures
  - Visually model software
  - Verify software quality
  - Control changes to software

Chapter 2 - Iterative and Incremental Development

- **Develop software iteratively**
  - The Rational Unified Process supports an iterative approach to development that addresses the highest risk items at every stage in the lifecycle, significantly reducing a project's risk profile.
  - This iterative approach helps attack risk through demonstrable progress frequent, executable releases that enable continuous end user involvement and feedback.
  - Because each iteration ends with an executable release, the development team stays focused on producing results, and frequent status checks help ensure that the project stays on schedule.
  - An iterative approach also makes it easier to accommodate tactical changes in requirements, features or schedule.
- **Manage requirements**
  - The Rational Unified Process describes how to elicit, organize, and document required functionality and constraints; track and document tradeoffs and decisions; and easily capture and communicate business requirements.
  - The notions of use case and scenarios prescribed in the process has proven to be an excellent way to capture functional requirements and to ensure that these drive the design, implementation and testing of software, making it more likely that the final system fulfills the end user needs.

Chapter 2 - Iterative and Incremental Development

- **Use component-based architectures**
  - The Rational Unified Process supports *component-based software development*.
  - The Rational Unified Process provides a systematic approach to defining an architecture using new and existing components.
  - It describes how to design a resilient architecture that is flexible, accommodates change, is intuitively understandable, and promotes more effective software reuse.
- **Visually model software**
  - The process shows how to visually model software to capture the structure and behavior of architectures and components.
  - Visual abstractions help to communicate different aspects of software; see how the elements of the system fit together; make sure that the building blocks are consistent with code; maintain consistency between a design and its implementation; and promote unambiguous communication.
  - Unified Modeling Language (UML), created by Rational Software, is the foundation for successful visual modeling.

Chapter 2 - Iterative and Incremental Development

- **Verify software quality**
  - Quality should be reviewed with respect to the requirements based on reliability, functionality, application performance and system performance.
  - The Rational Unified Process assists you in the planning, design, implementation, execution, and evaluation of these test types.
- **Control changes to software**
  - The ability to manage change is making certain that each change is acceptable, and being able to track changes is essential in an environment in which change is inevitable.
  - The process describes how to control, track and monitor changes to enable successful iterative development.
  - It also guides how to establish secure workspaces for each developer by providing isolation from changes made in other workspaces and by controlling changes of all software artifacts (e.g., models, code, documents, etc.).

Chapter 2 - Iterative and Incremental Development

- **Models Evolution Through Iterations**
  - Design in Unified Process proceeds through a series of cycles, each of which has following phases:
    - Inception
    - Elaboration
    - Construction
    - Transition
  - Within these phases we may go through a number of iterations, each involving the normal forms of workflow activity(requirements , analysis , design , implementation , testing).
  - A principal product of the Unified Process is a series of models, each appropriate to a key stage in system design.
  - Since many different models are produced, each for a different design purpose but all related to the same system, we need some common point of anchorage.
  - This is provided by a use case model.
  - Following figure shows a typical arrangement, in which five models appropriate to specific design activities all are rooted in the same use case model.

Chapter 2 - Iterative and Incremental Development

- **Models Evolution Through Iterations**

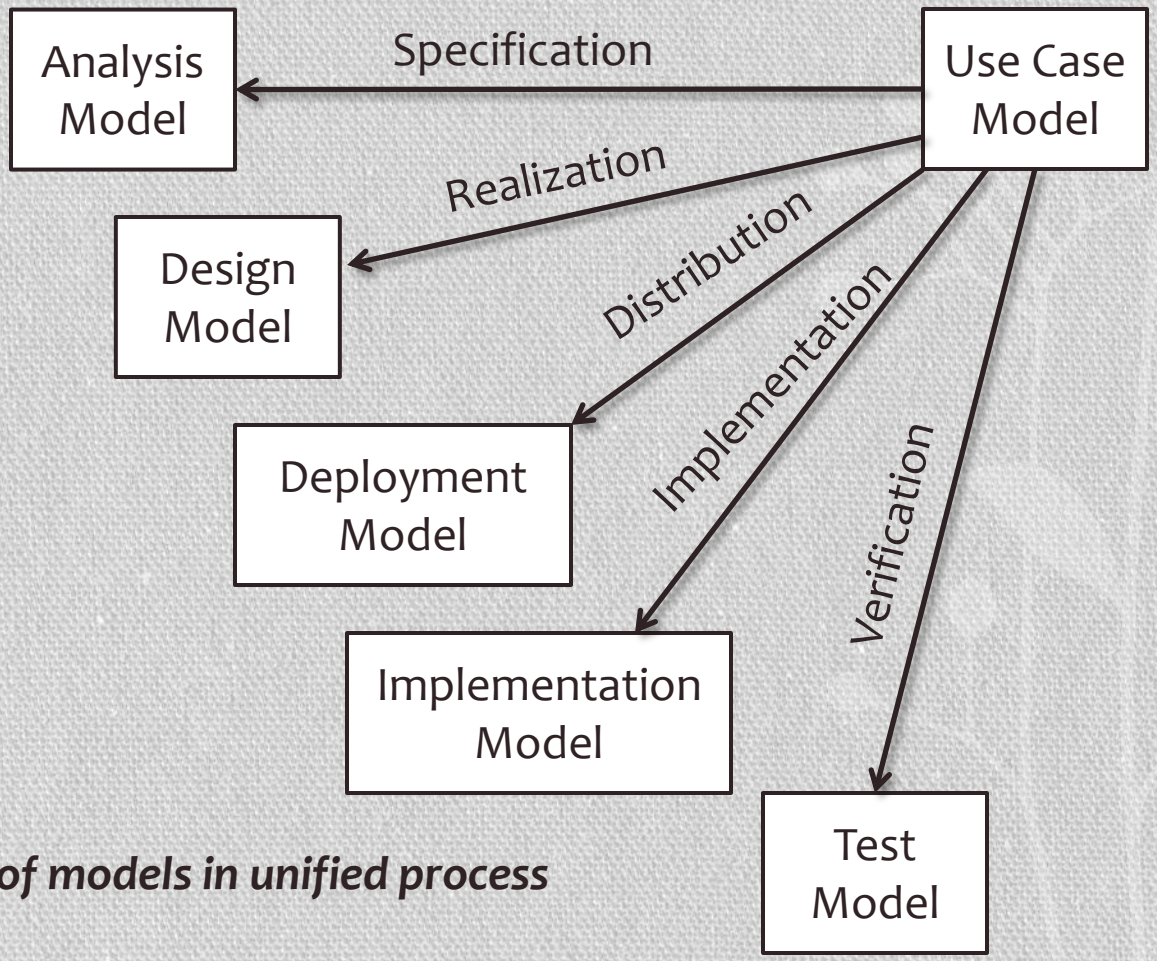


Fig: Series of models in unified process

Chapter 2 - Iterative and Incremental Development

- **Models Evolution Through Iterations**
  - The purpose of a use case is to describe the functionality required of the system from the point of view of those concerned with its operation.
  - The way a use case does this is by specifying a sequence of actions, including variants, that the system can perform and that yield an observable result of value to some actor.
  - In the Unified Process, this drives:
    - Requirements capture.
    - Analysis and design of how system realises use cases.
    - Acceptance/system testing.
    - Planning of development tasks.
    - Traceability of design decisions back to use cases.

Chapter 2 - Iterative and Incremental Development

- **Models Evolution Through Iterations**
  - **Use Case Model**
    - *Use case diagram*
  - **Analysis Model**
    - *Collaboration diagram and a flow of event description*
  - **Design Model**
    - *Class diagram, Sequence diagram, State chart diagram*
  - **Deployment Model**
    - *Deployment diagram*
  - **Implementation Model**
    - *Component diagram*
  - **Test Model**
    - *Test case*

Chapter 2 - Iterative and Incremental Development

- **Example**
  - consider part of the design for a cash dispenser system.
  - Take initial use-case diagram of figure 1.

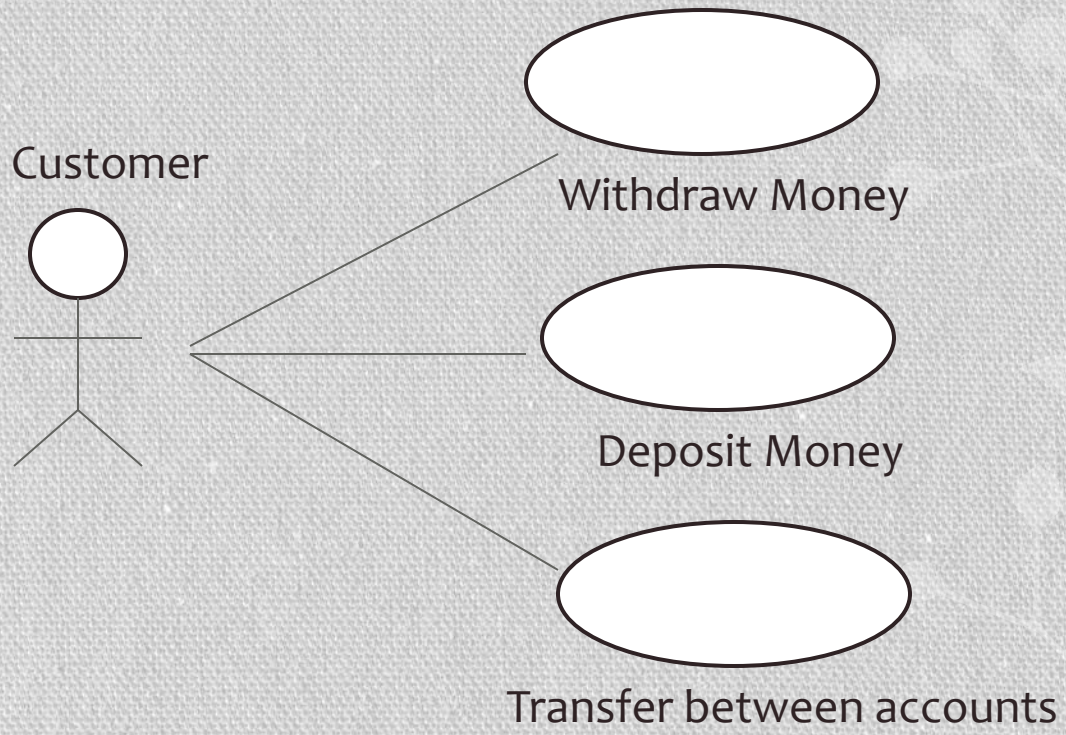


fig 1: Initial Use case diagram

Chapter 2 - Iterative and Incremental Development

- The actor here is a customer and the actions are withdrawing money, depositing money and transferring between accounts.
- Now, concentrate on withdrawing money.
- Which are the main top-level elements (classes in object oriented design) necessary in our system to perform this action?
- To depict this we produce an analysis model as shown in fig. 2.

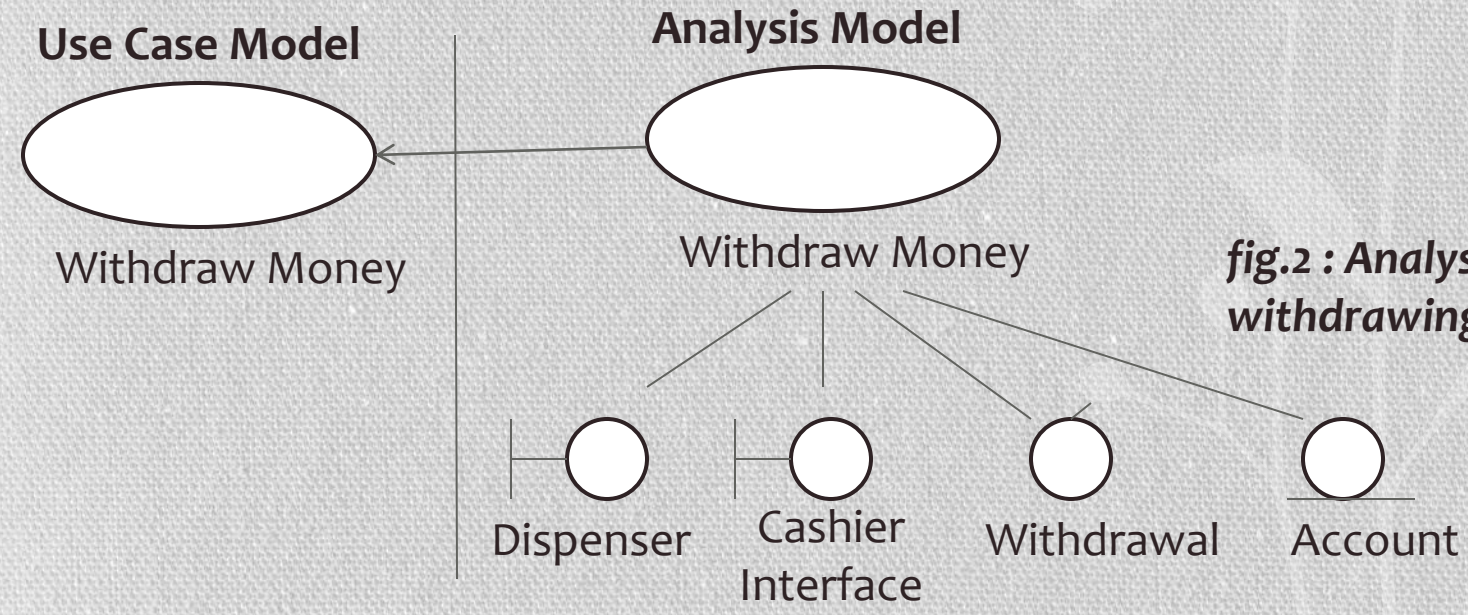


fig.2 : Analysis Model for withdrawing money

Chapter 2 - Iterative and Incremental Development

- Above fig. shows four analysis classes: a dispenser interface; a cashier interface; a process for making withdrawals; and an account database.
- Here, interfaces, processes and databases are distinguished using different symbols.
- the analysis model is linked to the use case model so as to have traceability back to use case.
- Now, we will show how they interact, with each other and with the customer (the actor in our use case model).
- fig. 3 shows the collaboration model.
  - Customer interaction is with the cashier interface (identifying the customer) and the dispenser (passing out money).
  - Both of these interact with the withdrawal process which in turn interacts with the account database.

Chapter 2 - Iterative and Incremental Development

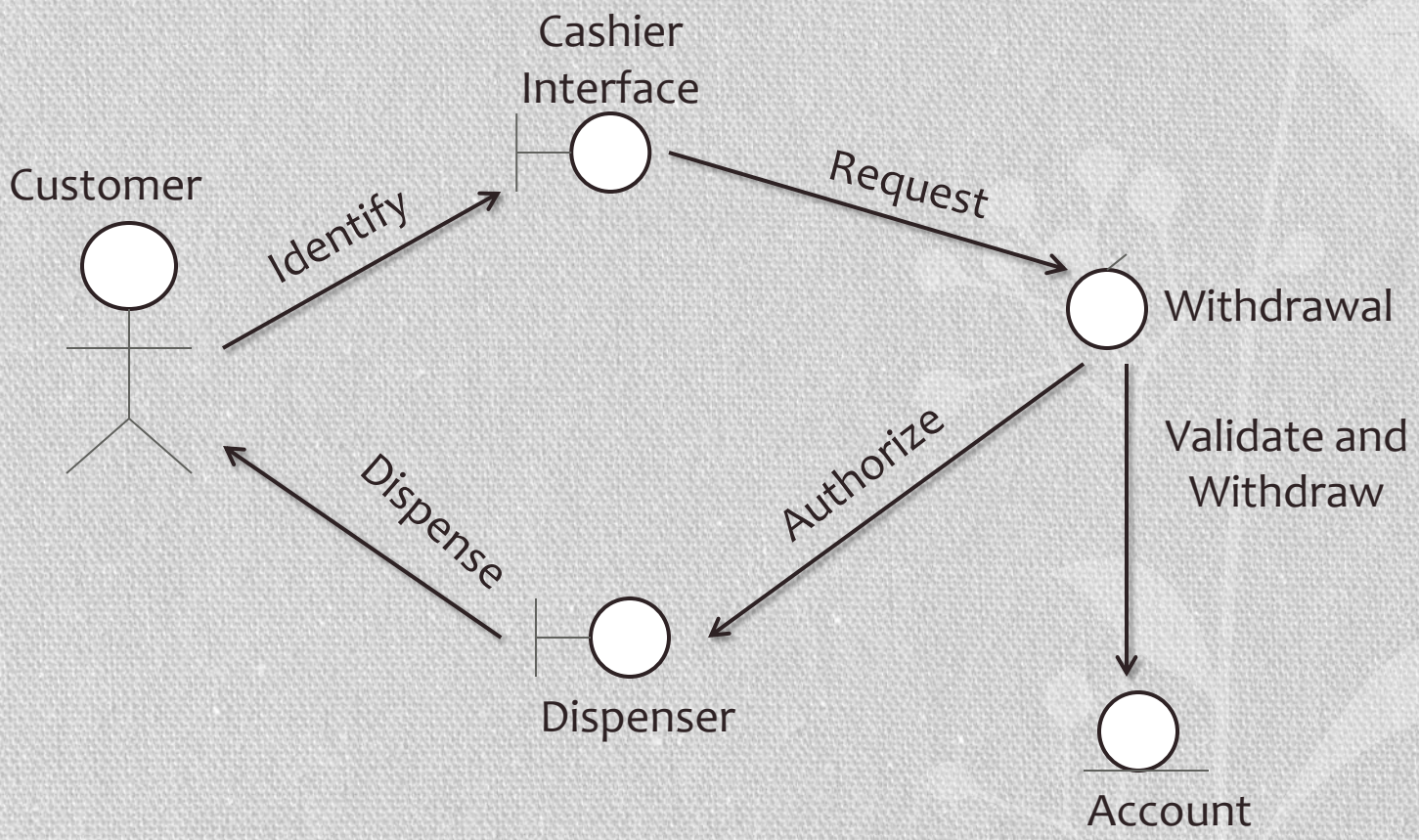


fig.3 : Collaboration Model for withdrawing money

Chapter 2 - Iterative and Incremental Development

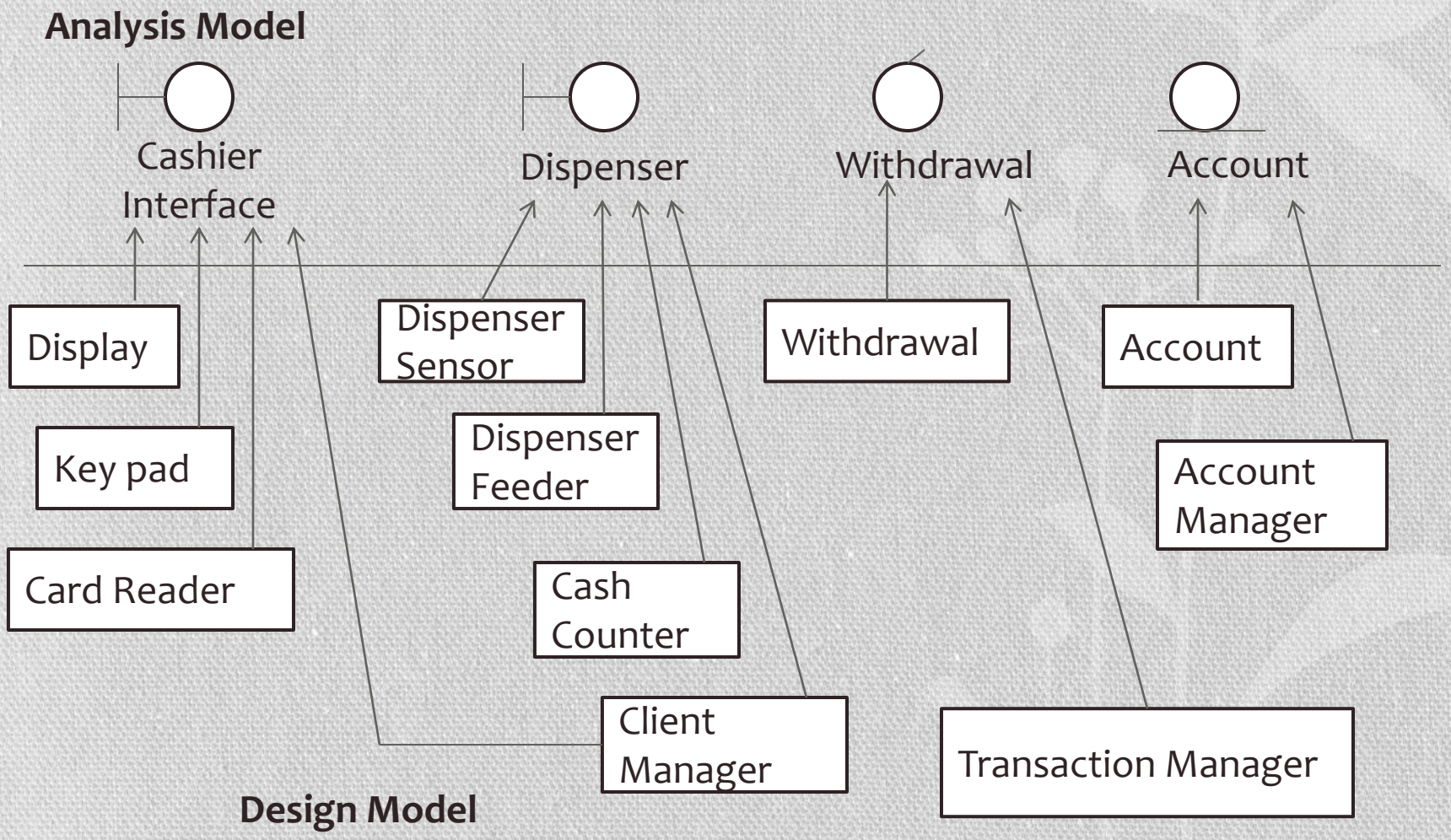


fig. 4 : Design classes introduced for analysis classes

Chapter 2 - Iterative and Incremental Development

- fig. 4 decomposes each of our analysis classes into groups of design classes.
  - Like the cashier interface needs a display, keypad and card reader.
- As shown in fig. 5 design classes will interact.
  - Like the card reader obtains information from the customer and supplies information to the client manager.
- Finally, the sequencing of events for interactions permitted in fig. 5 is visually represented through fig. 6.
  - The temporal sequence of events is read from the top of the diagram and each directed link shows the interaction between design classes.
  - Thus the sequence starts with the customer inserting a card in the card reader; then the card reader informs the client manager that the card has been inserted; then the client manager contacts the display to request a personal identication number; and so on.

Chapter 2 - Iterative and Incremental Development

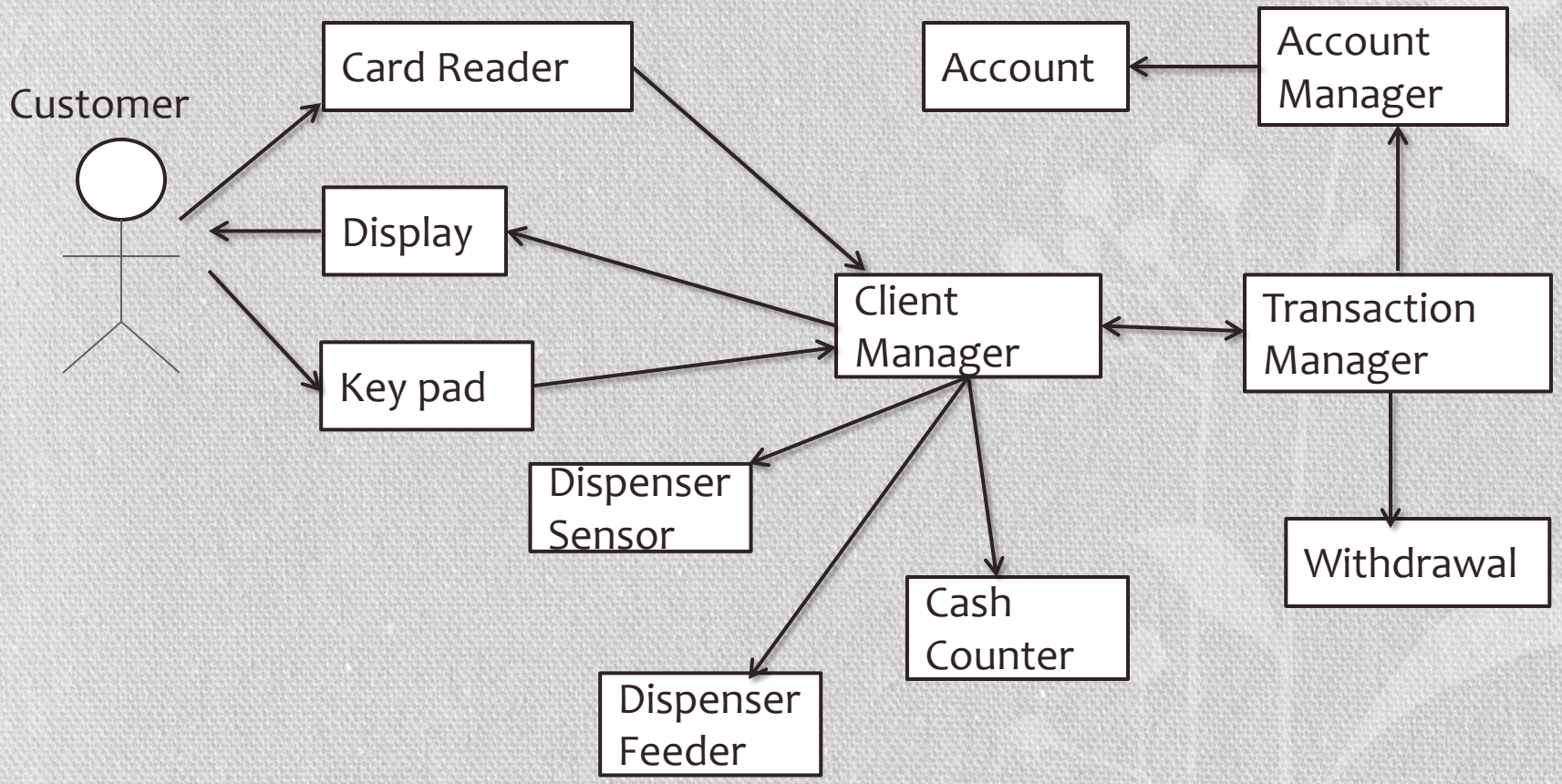


fig. 5 : Interactions between design classes

Chapter 2 - Iterative and Incremental Development

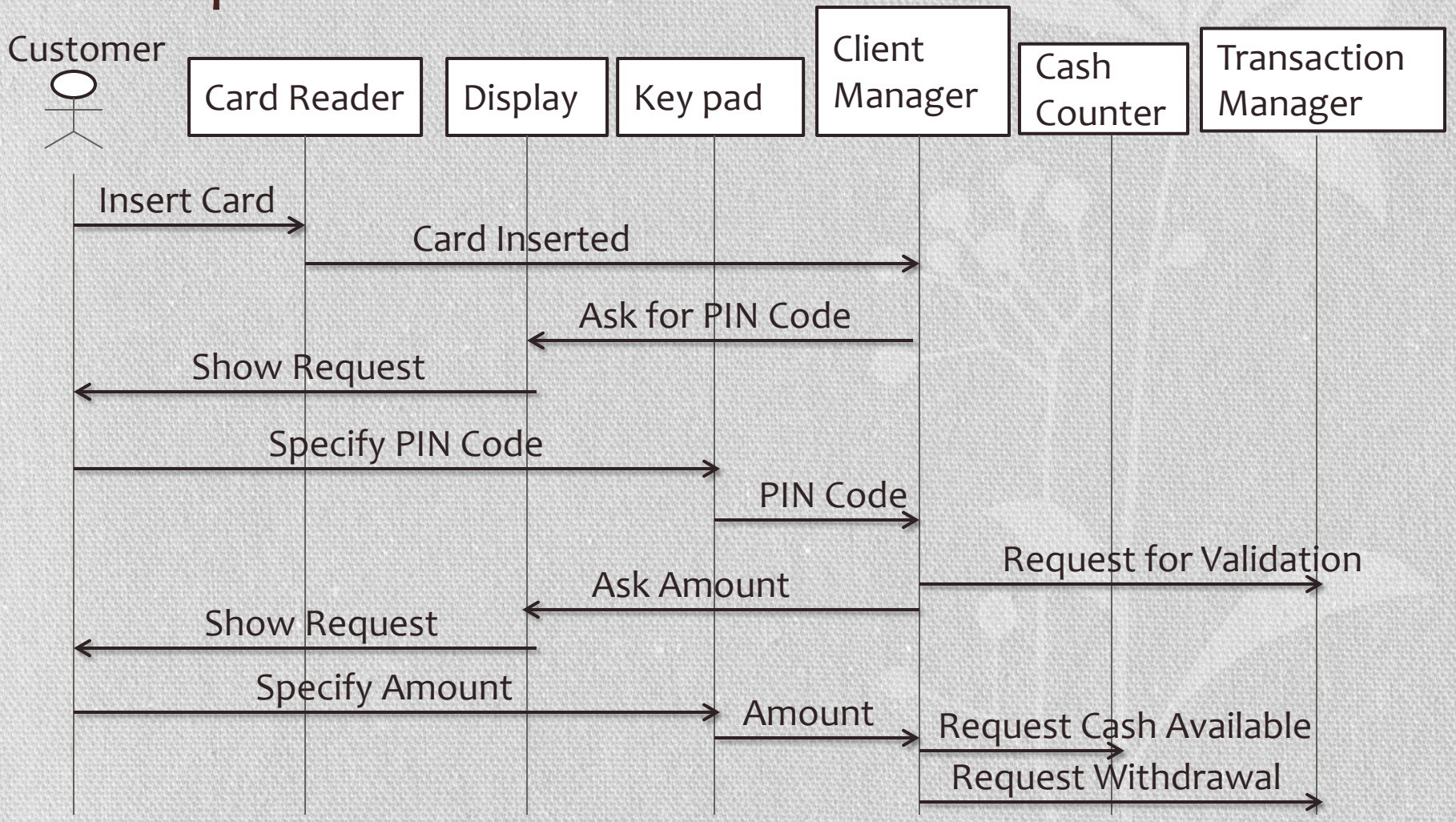


fig. 6 : Sequence Diagram