SYSTEMS ANALYSIS & DESIGN

JACKSON ALUNGA

JALUNGA@TUKENYA.AC.KE

INTRODUCTION

- The Rational Unified Process (RUP) is an iterative software development methodology developed by Rational Software Corporation (now part of IBM).
- It provides a structured approach to systems analysis and design, emphasizing a disciplined, well-organized development process.

THE RUP

- Software Development is a process of developing a software system from requirements (functional and non-functional).

- A software process provides a well-organized approach to assigning tasks and responsibilities to ensure the production of high-quality software within a predictable schedule / budget.

THE RUP

- The Rational Unified Process enhances team productivity, by providing every team member with easy access to a knowledge base with guidelines, templates and tool mentors for all critical development activities.
- The Rational Unified Process activities create and maintain models.
- The Rational Unified Process is a guide for how to effectively use the Unified Modeling Language (UML).

THE RUP

- The Rational Unified Process is supported by tools, which automate large parts of the process.

- The Rational Unified Process is a configurable process.

PROCESS AND A MODELING LANGUAGE

For the **Process**, we need **standards** for **artifacts** produced by workers in **roles** undertaking **work items** or **activities** during development

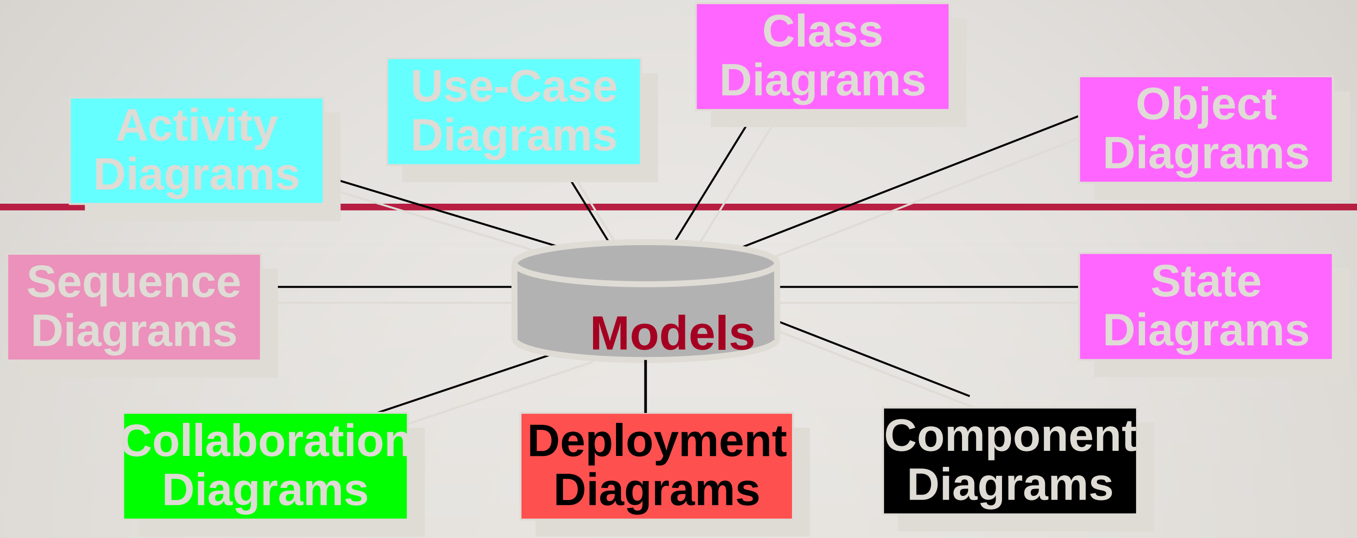
We need a modeling language:
We will use the **Unified Modeling Language**, (UML)

So, we are talking about a Unified Process (UP) and a modeling language (UML).

WHAT IS THE UML?

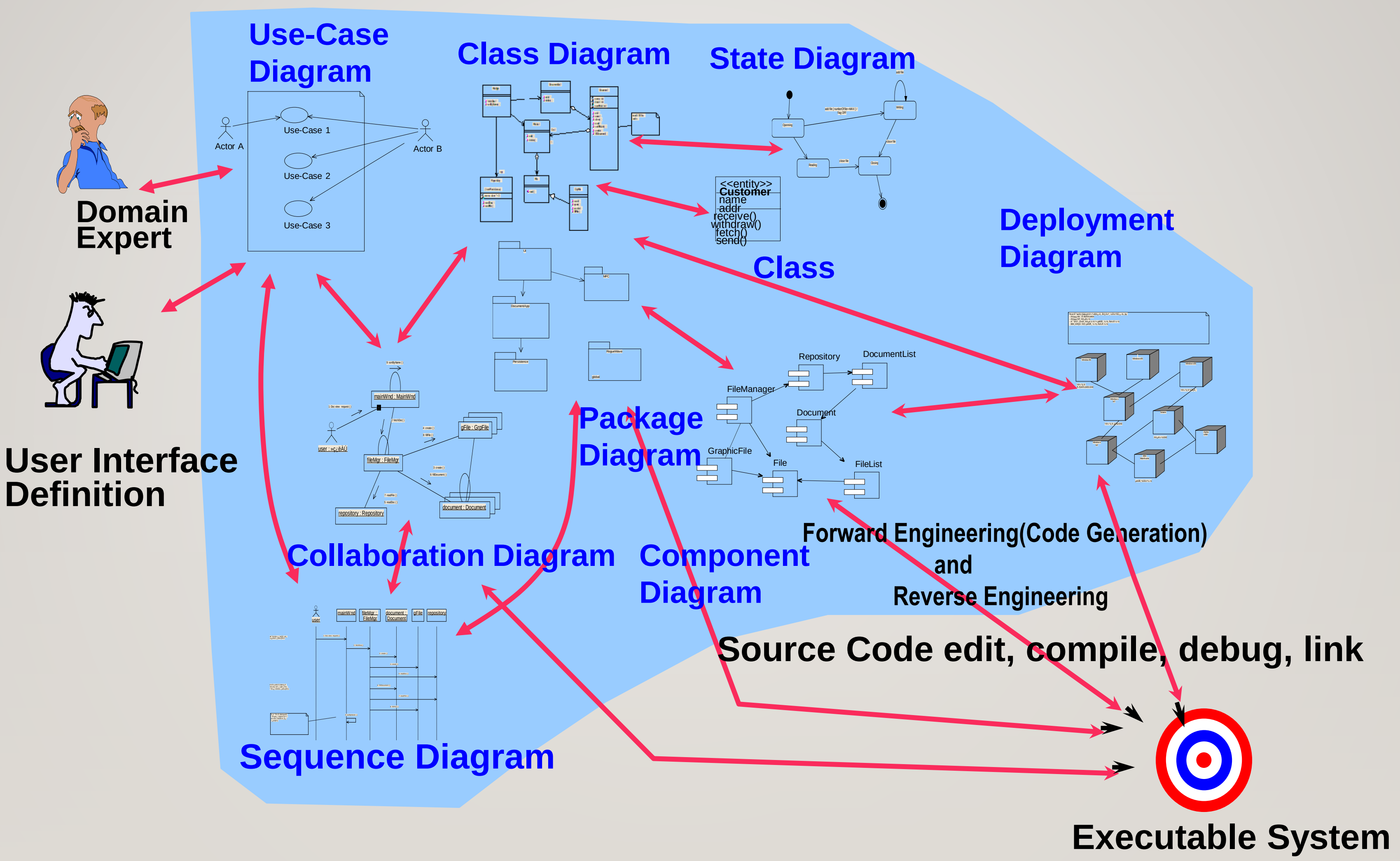
- The Unified Modeling Language (UML) is a language for
    - Specifying
    - Visualizing
    - Constructing
    - Documenting
- the artifacts of a software-intensive system

THE UML PROVIDES STANDARDIZED DIAGRAMS (VIEWS)



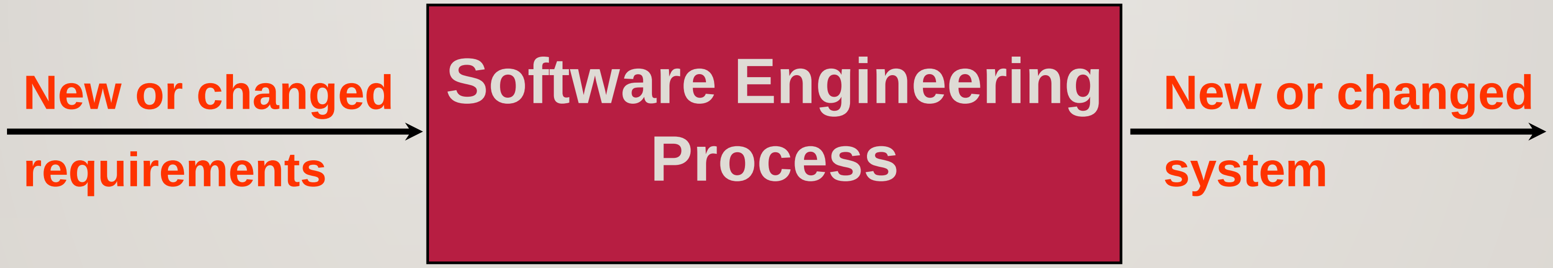
- In building **visual models**, many different diagrams are needed to represent different views of the system. (different views to **different stakeholders**).
- **Use Case Diagrams** (ahead) – illustrate user interactions with the application.
- **Activity Diagrams** illustrate the **flow of events** in a Use Case (all scenarios).
- **Class diagrams** represent logical **structure**, while
- **Interaction Diagrams** illustrate **behavior** (show how objects collaborate via message passing to provide features (responsibilities) of the objects..
- **Other diagrams** are used to illustrate other viewpoints; e.g. State Diagrams.

UML DIAGRAMS ARE KEY ARTIFACTS PRODUCED



WHAT IS A PROCESS?

- A process defines Who is doing What, When and How to reach a certain goal. In software engineering the goal is to build a software product or to enhance an existing one



We will use the UP - a generic process that uses UML as a modeling language.

The UP can be used for **any kind** of software system (information system, scientific or engineering-oriented system, etc.)

- The Rational Unified Process captures many of the best practices in modern software development in a form that is suitable for a wide range of projects and organizations.

- In next section, we describe the six fundamental best practices of the Rational Unified Process.

THE 6 BEST PRACTICES

- 1. Develop software iteratively
- 2. Manage requirements
- 3. Use component-based architectures
- 4. Visually model software
- 5. Verify software quality
- 6. Control changes to software

1. DEVELOP SOFTWARE ITERATIVELY

- Given today's sophisticated software systems, it is not possible to sequentially first define the entire problem, design the entire solution, build the software and then test the product at the end.
- An iterative approach is required that allows an increasing understanding of the problem through successive refinements, and to incrementally grow an effective solution over multiple iterations.
- 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 you 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.

2. 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.
- They provide coherent and traceable threads through both the development and the delivered system.

3. USE COMPONENT-BASED ARCHITECTURES

- The process focuses on early development and baselining of a robust executable architecture, prior to committing resources for full-scale development.
- It describes how to design a resilient architecture that is flexible, accommodates change, is intuitively understandable, and promotes more effective software reuse.
- The Rational Unified Process supports component-based software development.

- Components are non-trivial modules, subsystems that fulfill a clear function. The Rational Unified Process provides a systematic approach to defining an architecture using new and existing components.
- These are assembled in a well-defined architecture, either ad hoc, or in a component infrastructure such as the Internet, CORBA, and COM, for which an industry of reusable components is emerging.

4. VISUALLY MODEL SOFTWARE

- The process shows you how to visually model software to capture the structure and behavior of architectures and components.
- This allows you to hide the details and write code using "graphical building blocks." Visual abstractions help you communicate different aspects of your software; see how the elements of the system fit together; make sure that the building blocks are consistent with your code; maintain consistency between a design and its implementation; and promote unambiguous communication.
- The industry-standard Unified Modeling Language (UML), created by Rational Software, is the foundation for successful visual modeling.

5.VERIFY SOFTWARE QUALITY

- Poor application performance and poor reliability are common factors which dramatically inhibit the acceptability of today's software applications. Hence, 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.
- Quality assessment is built into the process, in all activities, involving all participants, using objective measurements and criteria, and not treated as an afterthought or a separate activity performed by a separate group.

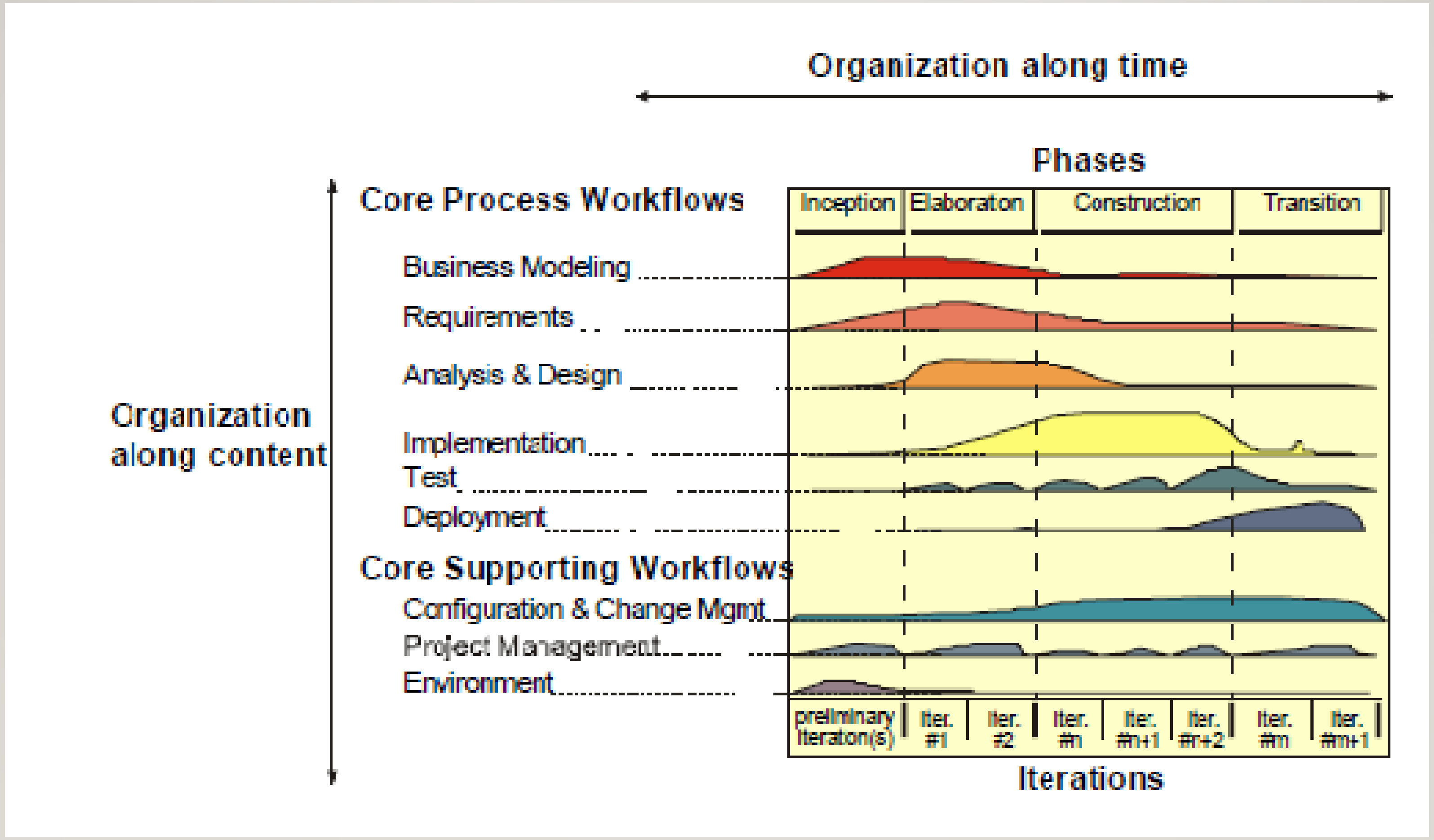
6. CONTROL CHANGES TO SOFTWARE

- The ability to manage change-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 you in 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.). And it brings a team together to work as a single unit by describing how to automate integration and build management.

TWO DIMENSIONS

- The process can be described in two dimensions, or along two axis:
- The horizontal axis represents time and shows the dynamic aspect of the process as it is enacted, and it is expressed in terms of cycles, phases, iterations, and milestones.
- The vertical axis represents the static aspect of the process: how it is described in terms of activities, artifacts, workers and workflows.

THE ITERATIVE MODEL GRAPH SHOWS HOW THE PROCESS IS STRUCTURED ALONG TWO DIMENSIONS

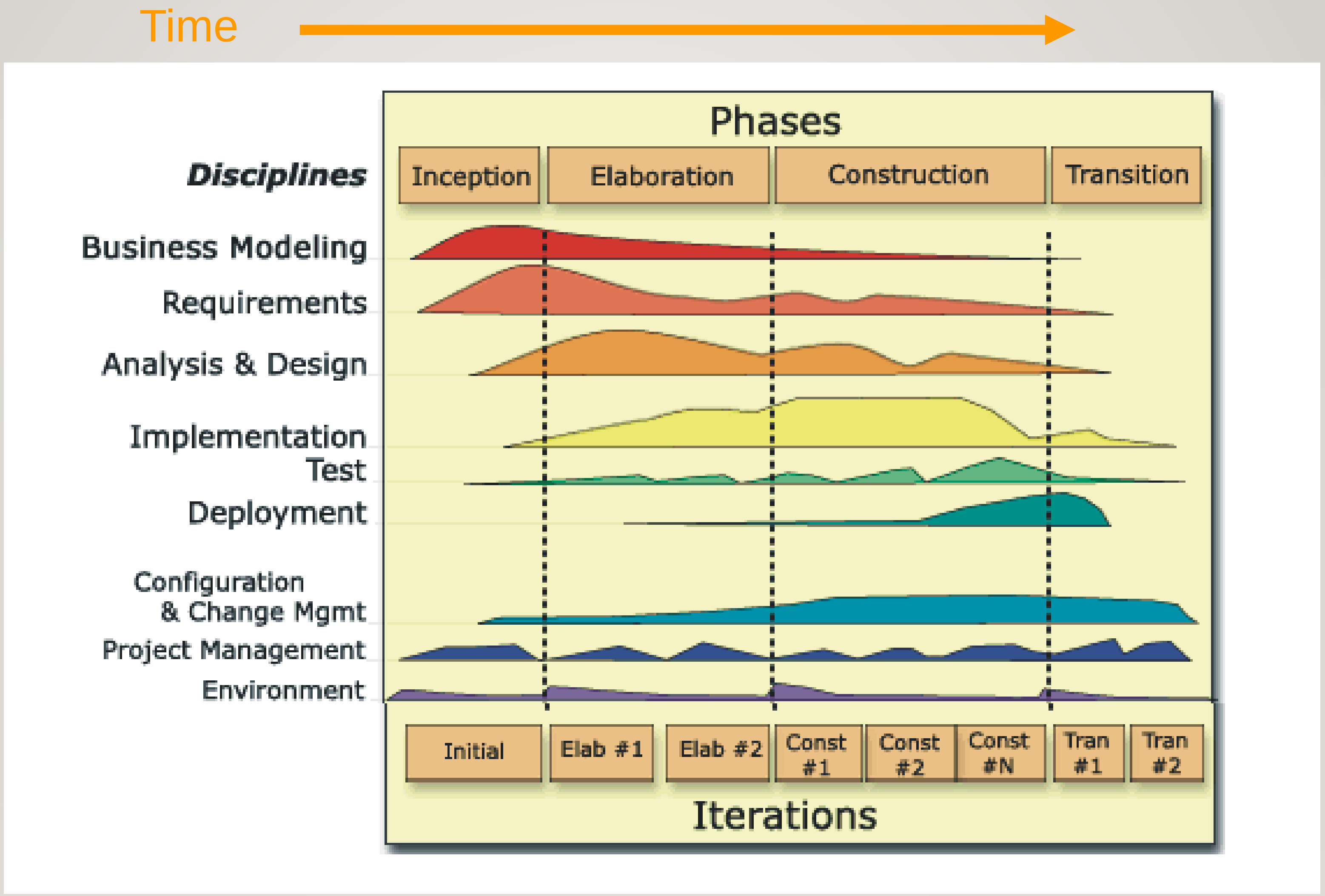


PHASES AND ITERATIONS - THE TIME DIMENSION

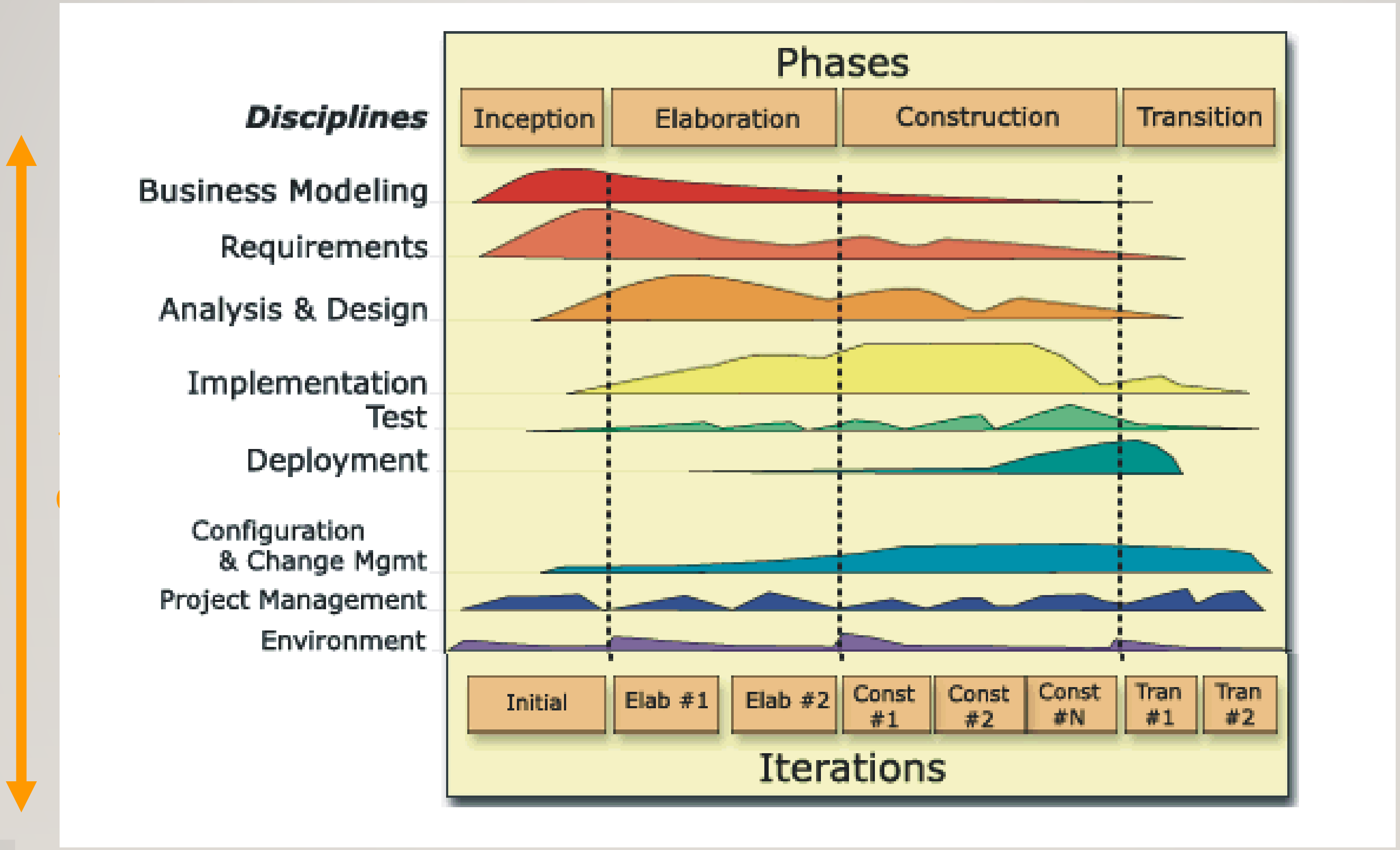
- The software lifecycle is broken into cycles, each cycle working on a new generation of the product.

- Rational Unified Process divides one development cycle in four consecutive phases
- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

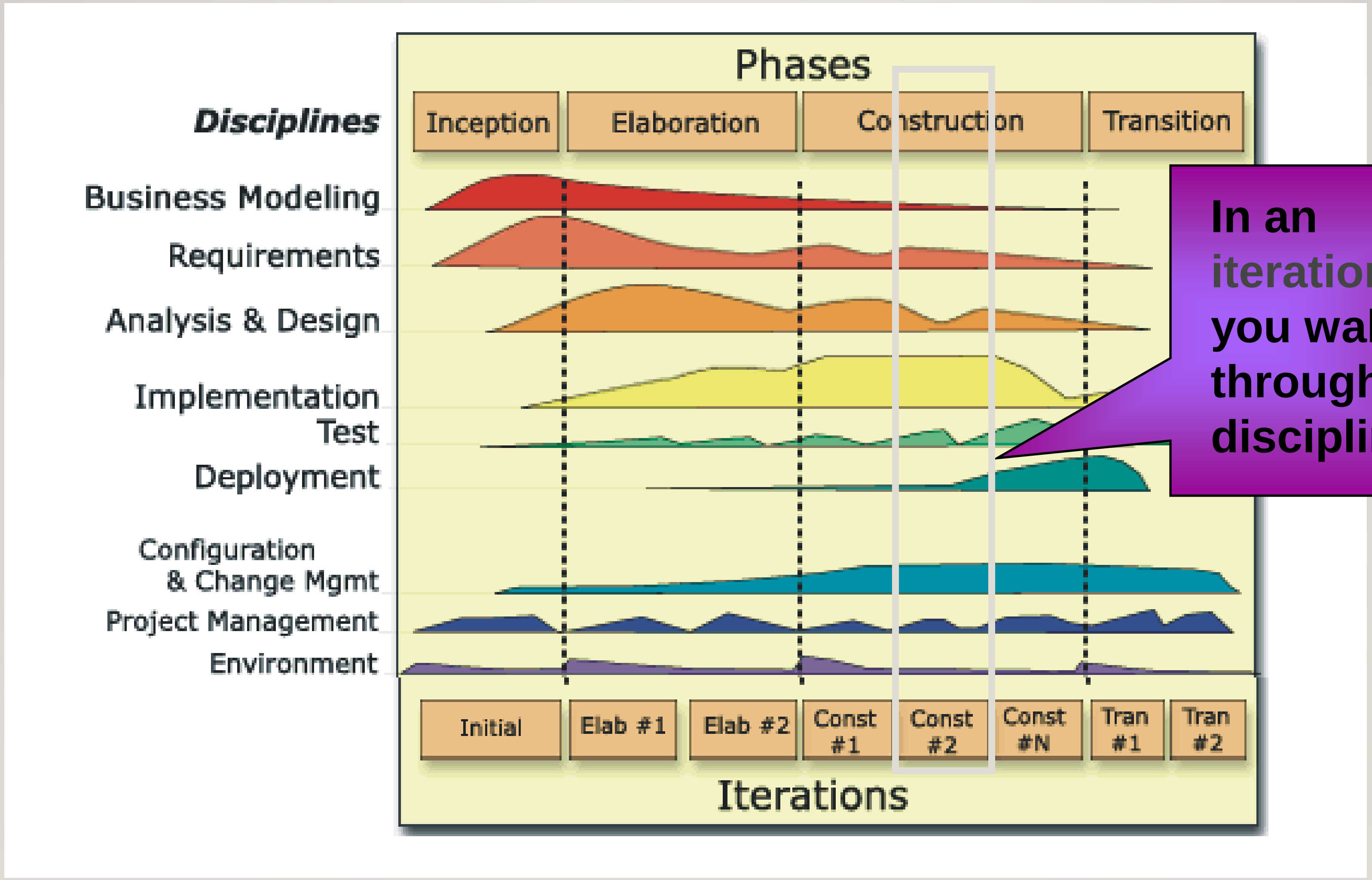
ORGANIZATION ALONG TIME



ORGANIZATION BASED ON CONTENT



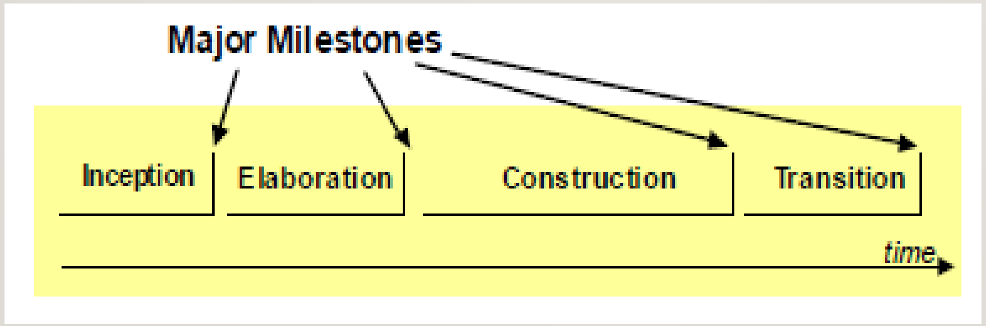
ONE ITERATION



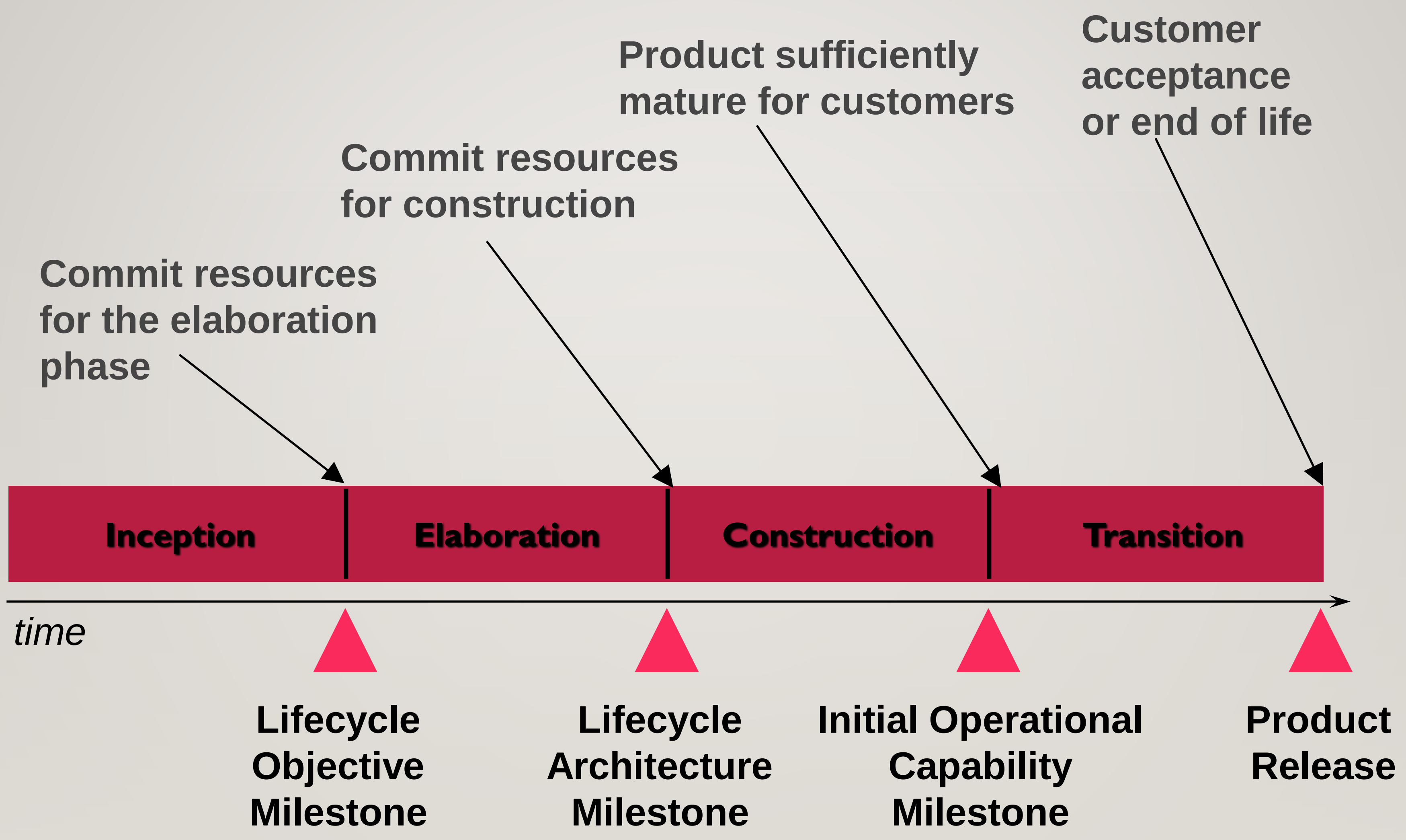
In an iteration, you walk through all disciplines.

MILESTONE

- Each phase is concluded with a well-defined milestone—a point in time at which certain critical decisions must be made, and therefore key goals must have been achieved.
- Phase and major milestone



MAJOR MILESTONES: BUSINESS DECISION POINTS


Customer acceptance or end of life
Product sufficiently mature for customers
Commit resources for construction
Commit resources for the elaboration phase
Inception
Elaboration
Construction
Transition
time
Lifecycle Objective Milestone
Lifecycle Architecture Milestone
Initial Operational Capability Milestone
Product Release

INCEPTION PHASE

- During the inception phase, you establish the business case for the system and delimit the project scope.
- To accomplish this you must identify all external entities with which the system will interact (actors) and define the nature of this interaction at a high-level. This involves identifying all use cases and describing a few significant ones.

INCEPTION PHASE: OBJECTIVES

- Prepare supporting environment for project
- Initial use case model (10%-20% complete)
- Establish project scope and boundary conditions
- Determine the use cases and primary scenarios that will drive the major design trade-offs
- Demonstrate a candidate architecture against some of the primary scenarios
- Estimate the overall cost and schedule
- Identify potential risks (the sources of unpredictability)

ELABORATION PHASE

- The overriding goal of the elaboration phase is to analyze the problem domain, establish a architectural foundation, develop the project plan, and eliminate the project’s high-risk elements.

ELABORATION PHASE: OBJECTIVES

- Refine support environment
- Define, validate and baseline the architecture as rapidly as is practical
- Baseline the vision
- Baseline a detailed plan for the construction phase
- Demonstrate that the baseline architecture will support the vision at a reasonable cost in a reasonable period of time
- Requirements Analysis and Capture
  - Use Case Analysis
    - Use Cases (80% written and reviewed by end of phase)
    - Use Case Model (80% done)
    - Scenarios
      - Sequence and Collaboration Diagrams
      - Class, Activity, Component, State Diagrams

CONSTRUCTION PHASE

- The overriding goal of the construction phase is to develop all remaining components and features and integrate them into the product.

CONSTRUCTION PHASE: OBJECTIVES

- Completing the software product for transition to production
- Minimizing development costs by optimizing resources and avoiding unnecessary scrap and rework
- Achieving adequate quality as rapidly as is practical
- Achieving useful versions (alpha, beta, and other test releases) as rapidly as possible

TRANSITION PHASE

- The overriding goal of the transition phase is to move the product to the user community.

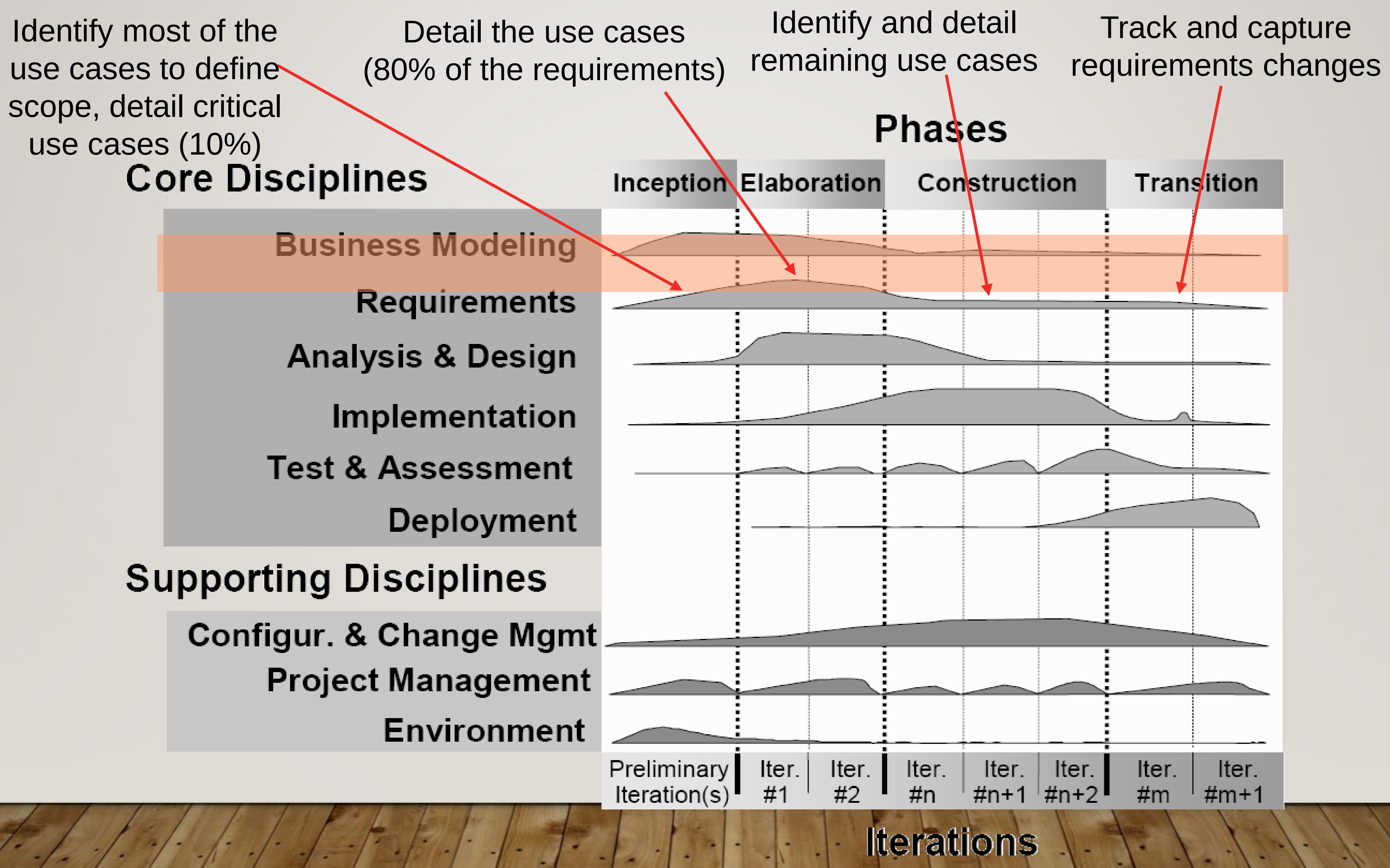
TRANSITION PHASE: OBJECTIVES

- 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 possible
  - Development team begins to shrink
  - Control is moved to maintenance team
  - Alpha, Beta, and final releases
  - Software updates
  - Integration with existing systems (legacy, existing versions…)

TRANSITION PHASE

- This phase can range from being very simple to extremely complex, depending on the type of product.
- For example, a new release of an existing desktop product may be very simple, whereas replacing a nation's air traffic control system would be very complex.

DISCIPLINES, PHASES, AND ITERATIONS



ITERATIONS

- Each phase in the Rational Unified Process can be further broken down into iterations.
- An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows incrementally from iteration to iteration to become the final system .

BENEFITS OF AN ITERATIVE APPROACH

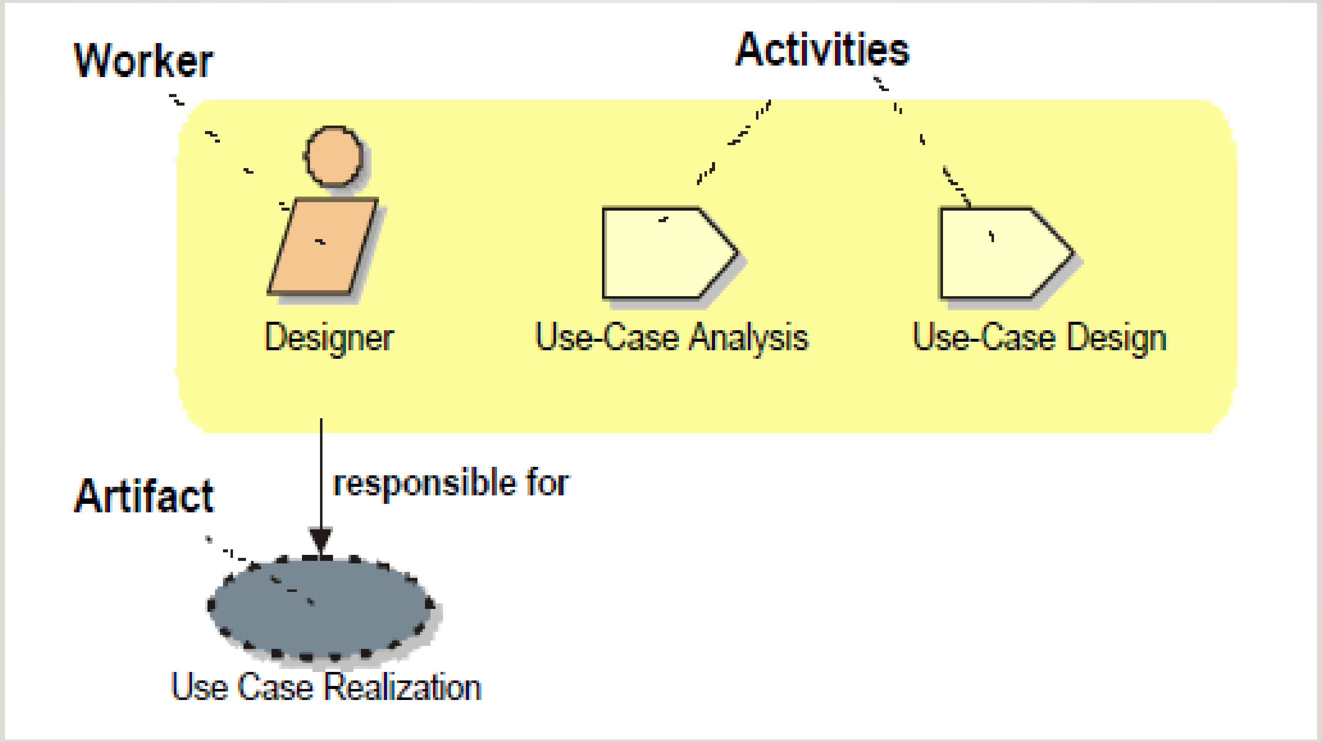
- Compared to the traditional waterfall process, the iterative process has the following advantages:
- Risks are mitigated earlier
- Change is more manageable
- Higher level of reuse
- The project team can learn along the way
- Better overall quality

STATIC STRUCTURE OF THE 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’

ACTIVITIES, ARTIFACTS, AND WORKERS

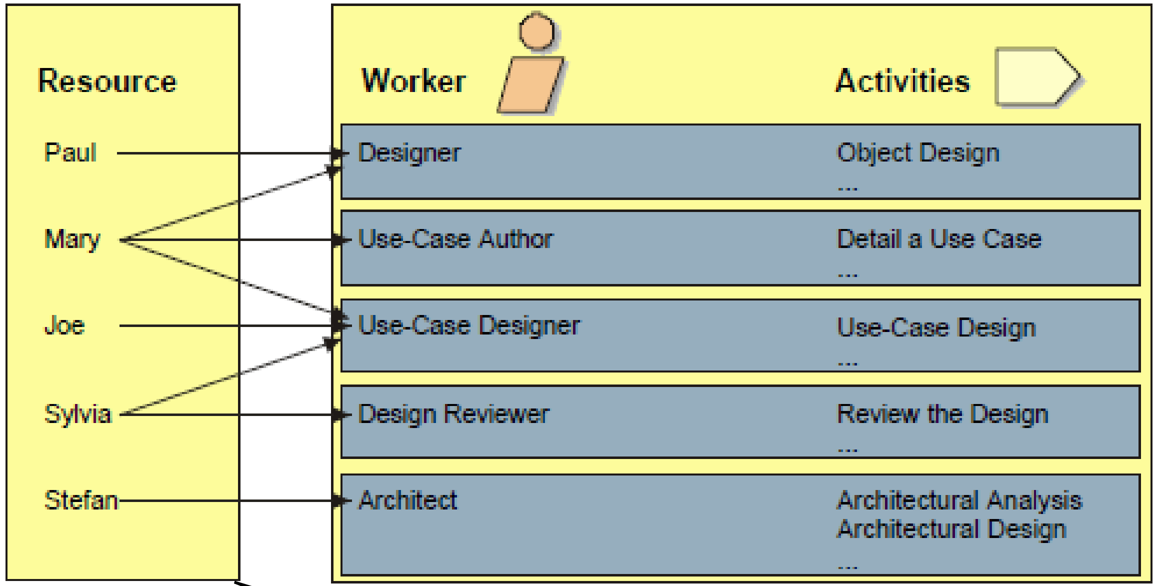
-



WORKER

- A worker defines the behavior and responsibilities of an individual, or a group of individuals working together as a team.
- You could regard a worker as 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.

PEOPLE AND WORKERS



People and Workers.

Each individual in the project is assigned to one or several roles

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.
- 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.

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

ARTIFACT

- 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 SHAPES OR FORMS:

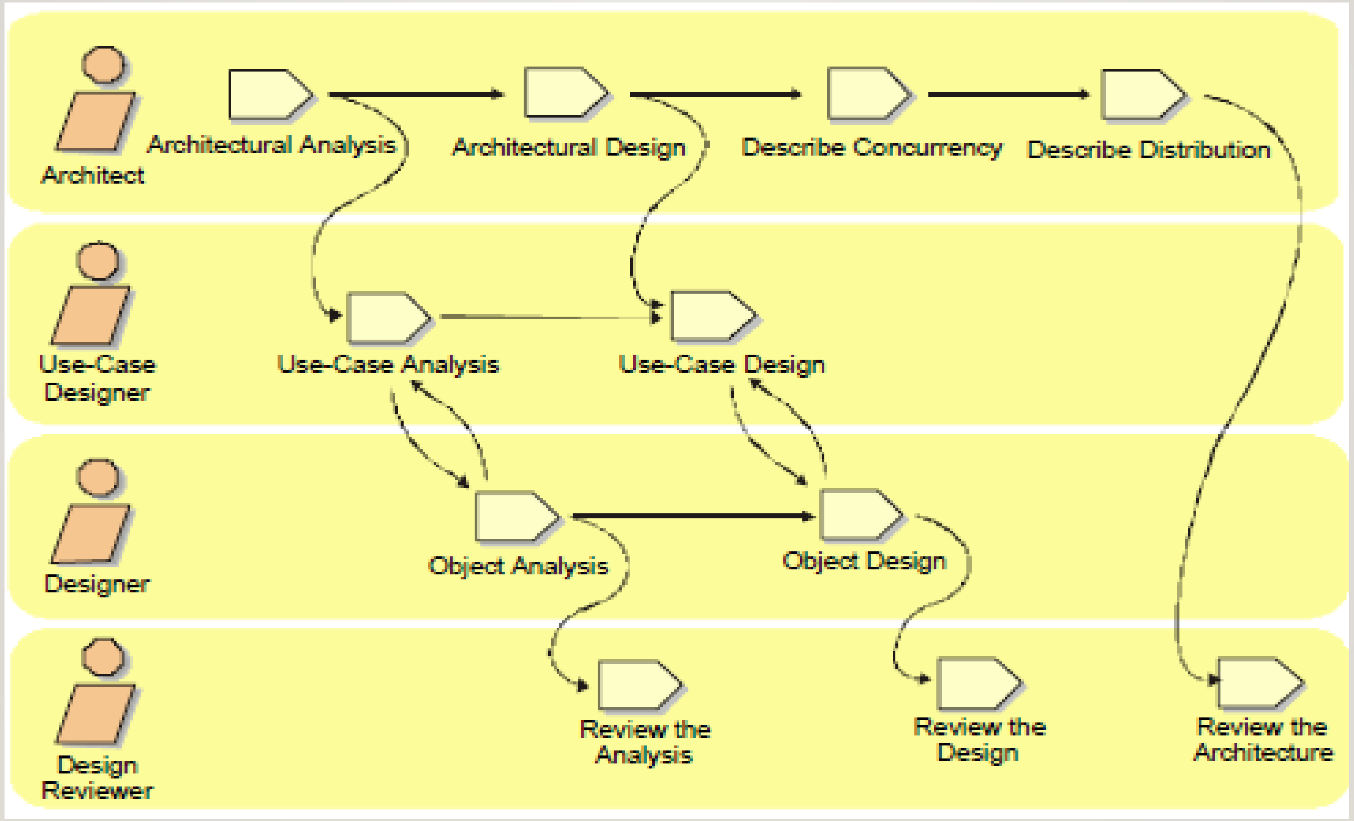
- Artifacts may take various shapes or forms:
- A model, such as the Use-Case Model or the Design Model
- A model element, i.e. an element within a model, such as a class, a use case or a subsystem
- A document, such as Business Case or Software Architecture Document
- Source code

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.
- We use a form of activity diagrams in this white paper.

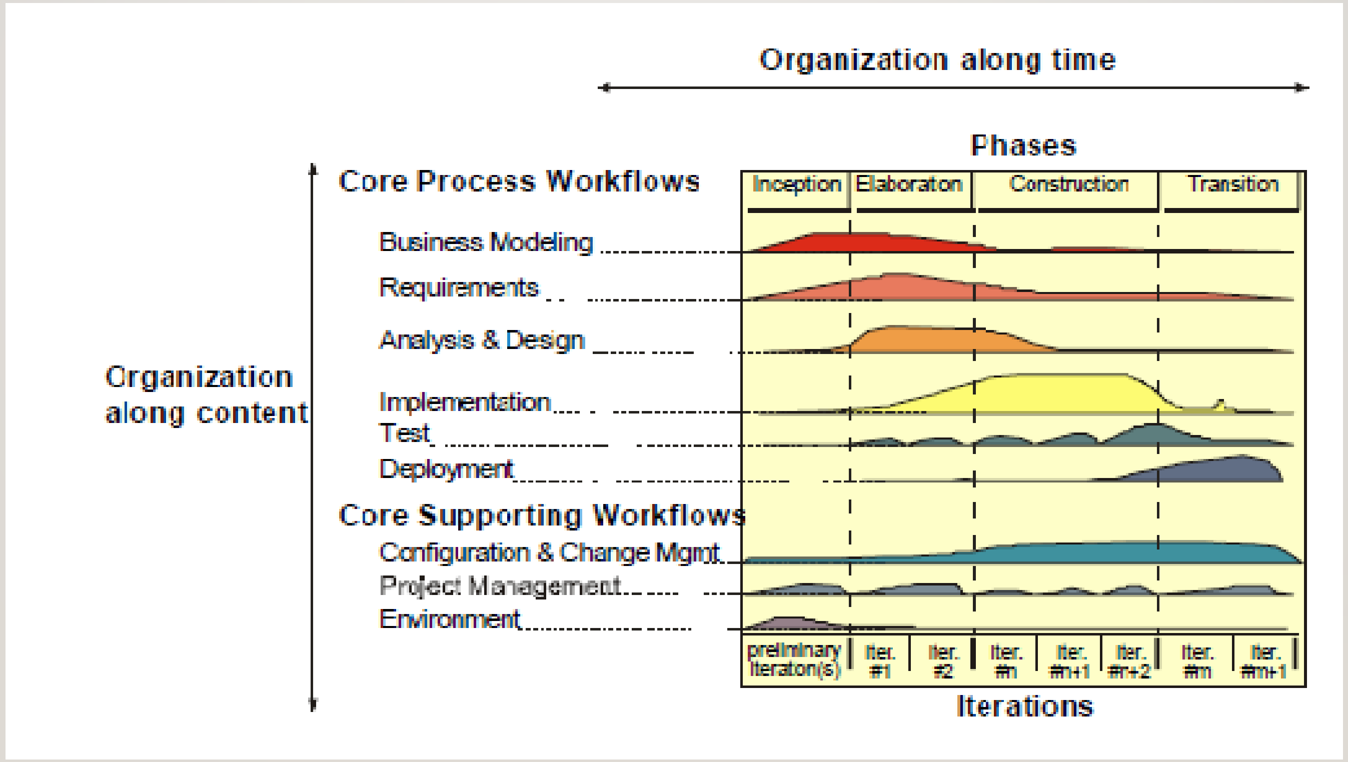
EXAMPLE OF WORKFLOW

•



CORE WORKFLOWS

- There are nine core process workflows in the Rational Unified Process.



WORKFLOWS

- The core process workflows are divided into six core “engineering” workflows:
- 1. Business modeling workflow
- 2. Requirements workflow
- 3. Analysis & Design workflow
- 4. Implementation workflow
- 5. Test workflow
- 6. Deployment workflow

- And three core “supporting” workflows:
- 1. Project Management workflow
- 2. Configuration and Change Management workflow
- 3. Environment workflow

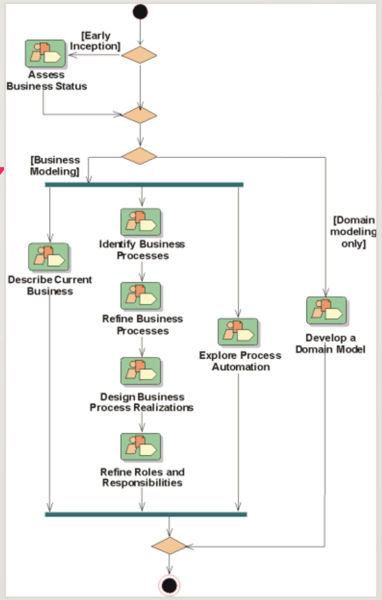
WORKFLOWS

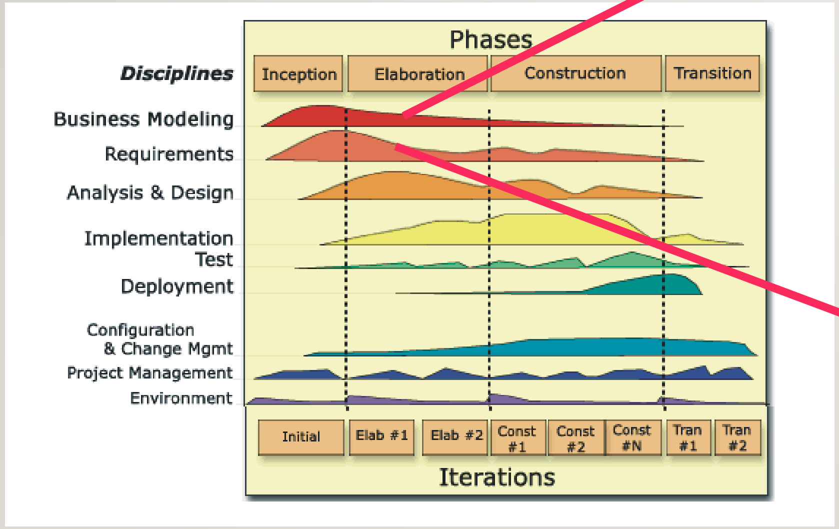
- Although the names of the six core engineering workflows may evoke the sequential phases in a traditional waterfall process, we should keep in mind that the phases of an iterative process are different and that these workflows are revisited again and again throughout the lifecycle.
-
- The actual complete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

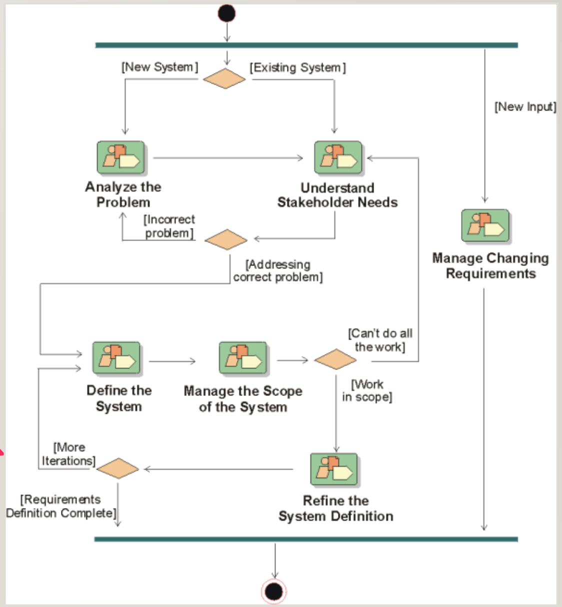
WORKFLOWS GUIDE ITERATIVE DEVELOPMENT

Business Modeling:

Workflow Details


[Early Inception]
Assess Business Status
[Business Modeling]
[Domain modeling only]
Identify Business Processes
Describe Current Business
Refine Business Processes
Explore Process Automation
Develop a Domain Model
Design Business Process Realizations
Refine Roles and Responsibilities




[New System]
[Existing System]
[New Input]
Analyze the Problem
Understand Stakeholder Needs
[Incorrect problem]
[Addressing correct problem]
Manage Changing Requirements
[Can't do all the work]
Define the System
Manage the Scope of the System
[Work in scope]
[More Iterations]
[Requirements Definition Complete]
Refine the System Definition

Requirements:

Workflow Details

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 that the output from business engineering is not 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.

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.
- The use-case description shows how the system interacts step by step with the actors and what the system does.

ANALYSIS & DESIGN

- The goal of the Analysis & Design workflow is to show how the system will be realized in the implementation phase.
- You want to build a system that:
- Performs—in a specific implementation environment—the tasks and functions specified in the use case descriptions.
- Fulfills all its requirements.
- Is structured to be robust (easy to change if and when its functional requirements change).

- 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.

IMPLEMENTATION

- The purpose of implementation are:
- To define the organization of the code, in terms of implementation subsystems organized in layers.
- To implement classes and objects in terms of components.
- 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.
- For example, a subsystem can be created as a directory or a folder in a file system, or a subsystem in Rational/Apex for C++ or Ada, or packages using Java.

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.

- 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.
- Test 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.

TEST AUTOMATION

- Strategies for when and how to automate test are described.
- Test automation is especially important using an iterative approach, to allow regression testing at the end of each iteration, as well as for each new version of the product.

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.

- 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.

PROJECT MANAGEMENT

- Software Project Management is the art of balancing competing objectives, managing risk, and overcoming constraints to deliver, successfully, a product which meets the needs of both customers (the payers of bills) and the users.