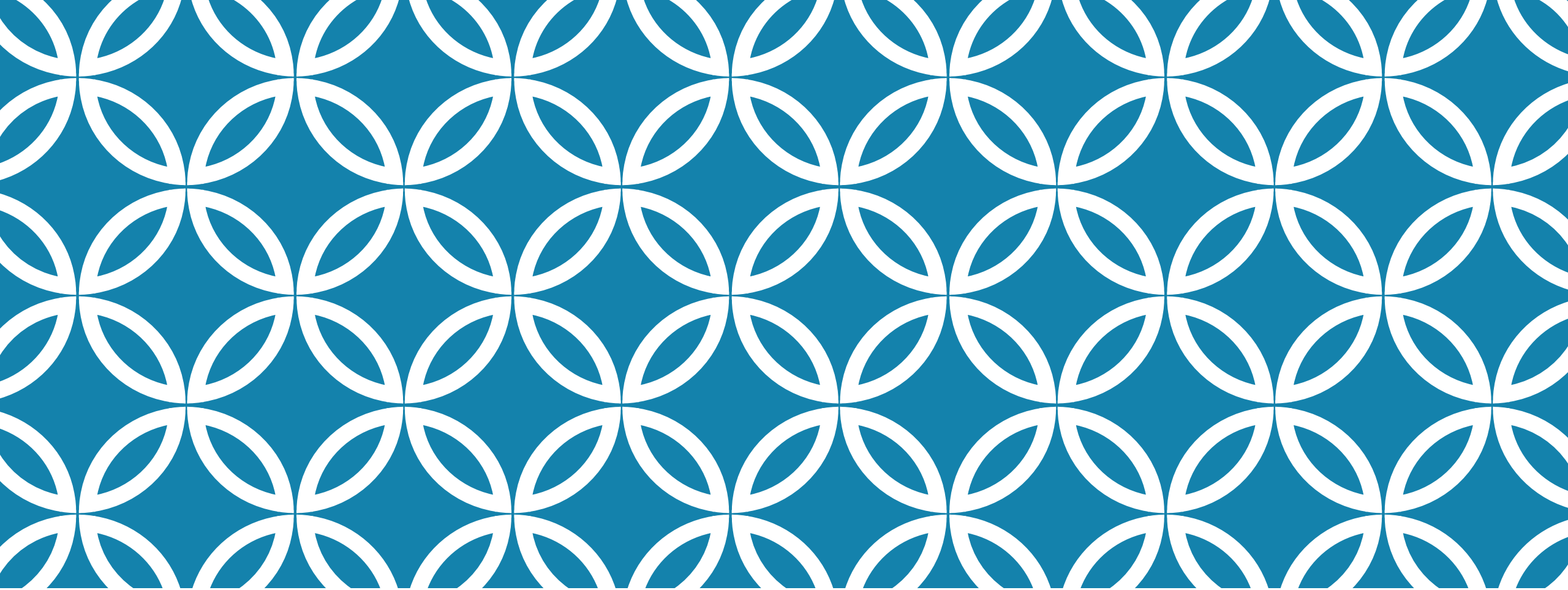


RATIONAL UNIFIED PROCESS

An Introduction

INTRODUCTION TO RATIONAL UNIFIED PROCESS

Overview

- Gain an introductory knowledge of the Rational Unified Process
  - The four phases of the Unified Process
  - The disciplines of the Unified Process
  - An overview of implementing the Unified Process
  - Best Practices

Be dazzled by pictures, graphs, and exercises.

WHAT IS THE RUP?

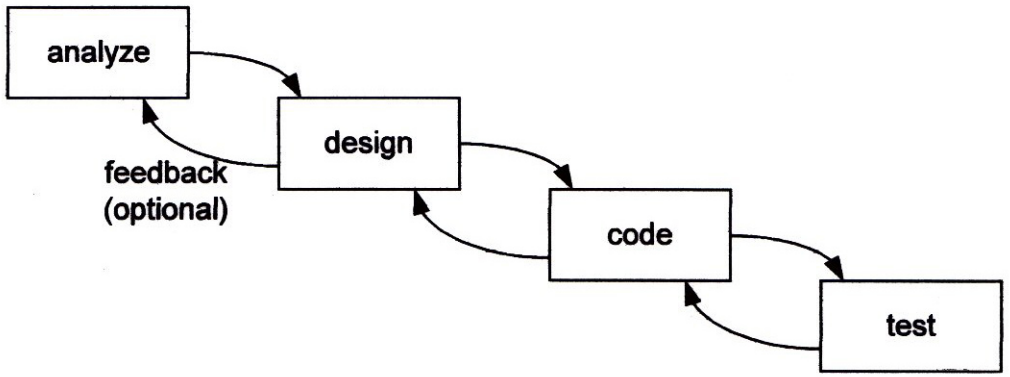
Software development process.

- Disciplined approach to assigning tasks and responsibilities
- Generic process framework
- Utilizes UML
- Use case driven
  - Architecture-centric
  - Iterative

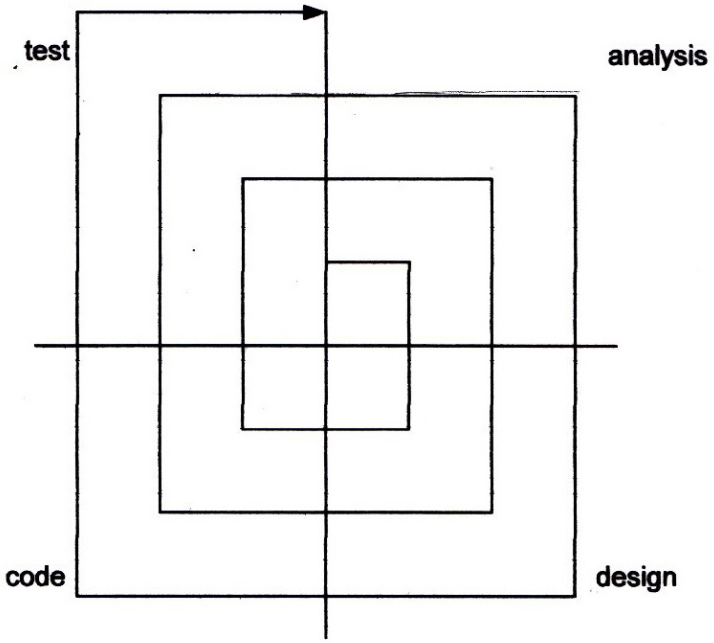
WATERFALL VS SPIRAL

An

- Ea



The Waterfall



The Spiral

The waterfall process was popular during the days of small projects

Each step had to be completed before progressing to the next.

The first iterate process was the Spiral, which iterates the entire waterfall process.

DEVELOPMENT RISKS

Initial designs usually have flaws

Flaws are more easily discovered in late steps of traditional processes (during system test, integration, delivery)

The cost of flaws increases as development progresses

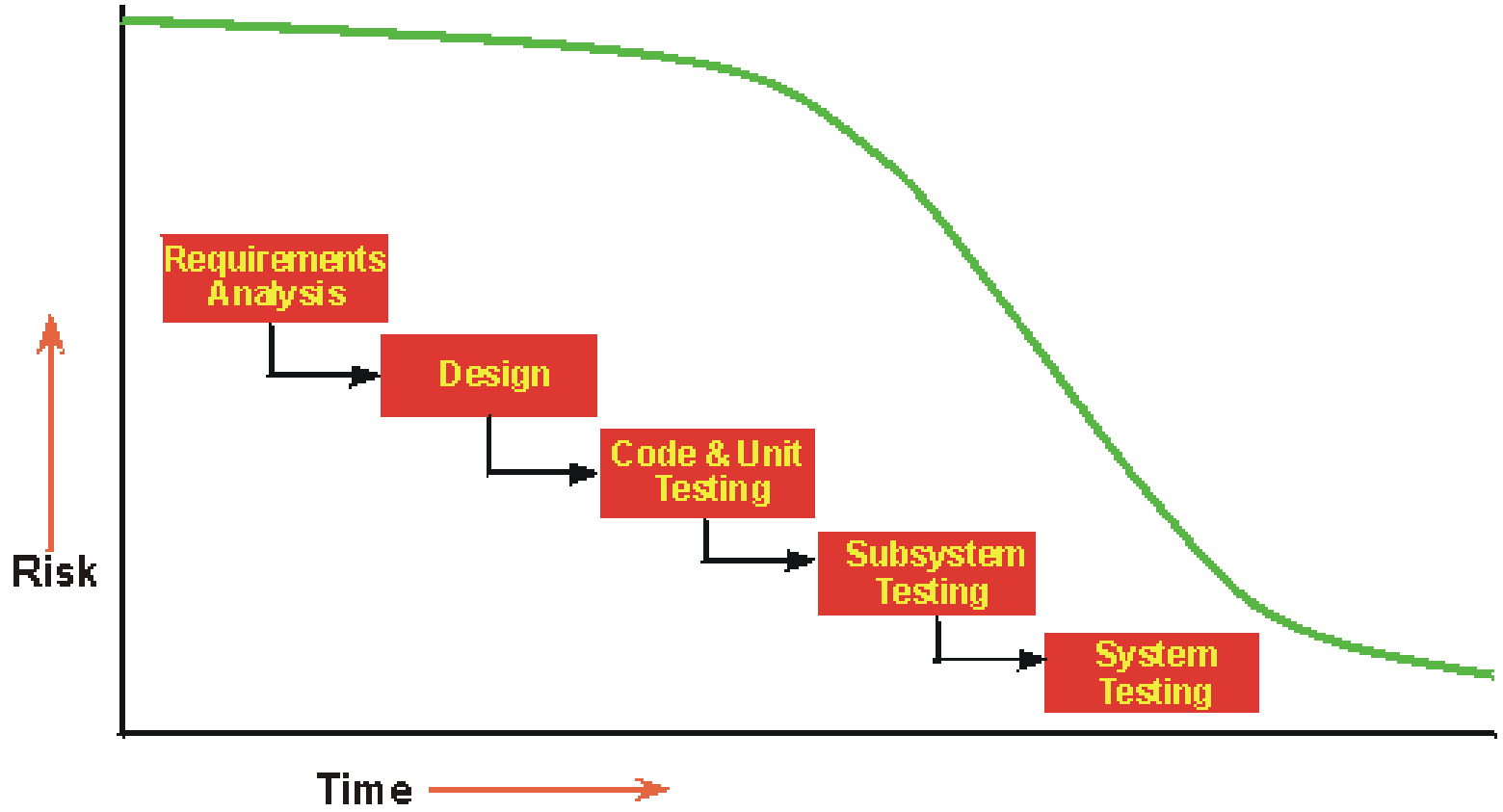
MITIGATING RISKS

In the waterfall approach most risks are discovered and addressed during integration.

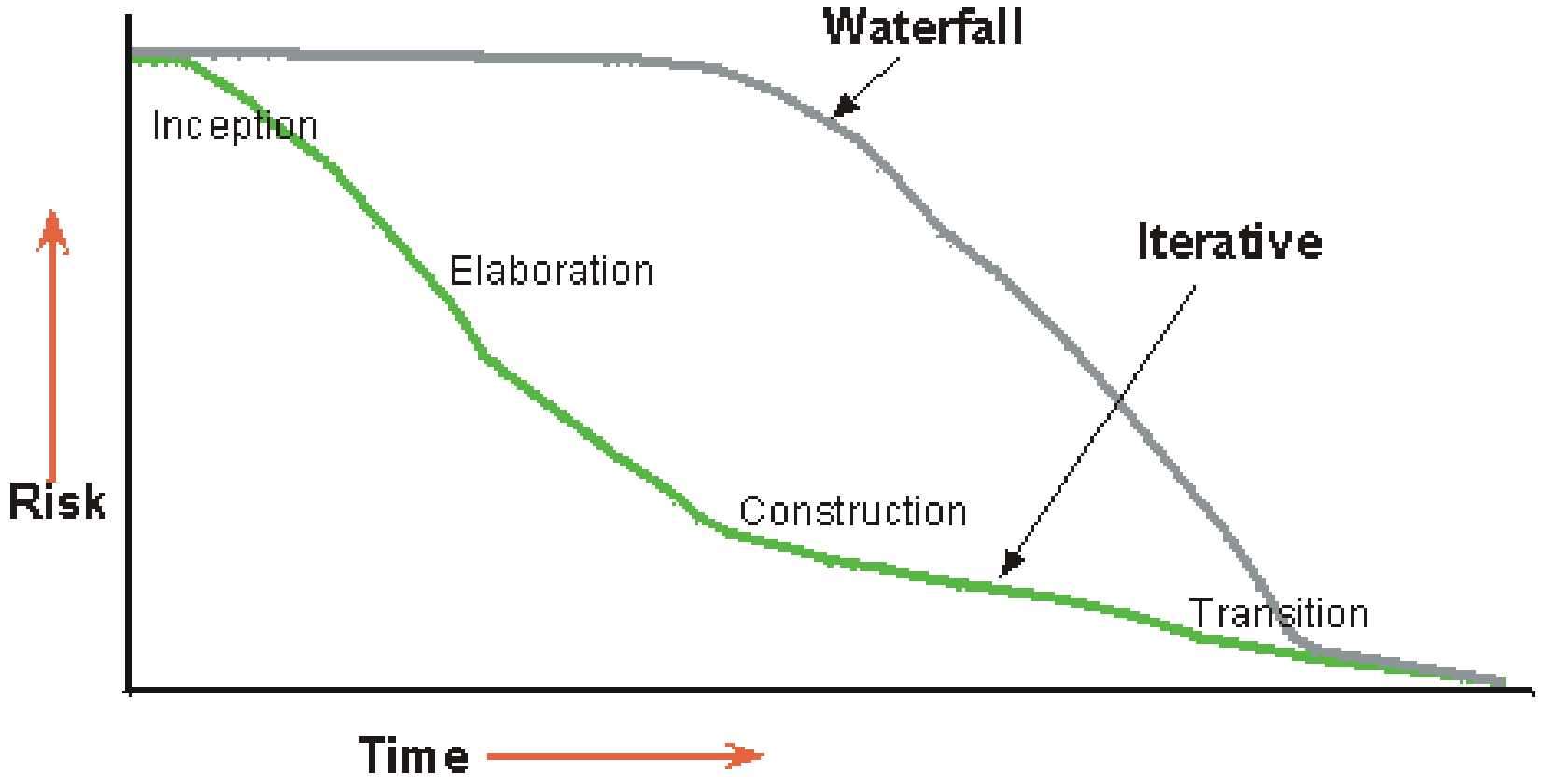
With iterative development, integration progresses continuously, rather than all at the very end. Risks are dealt with early and continuously.

High risk activities are done first.

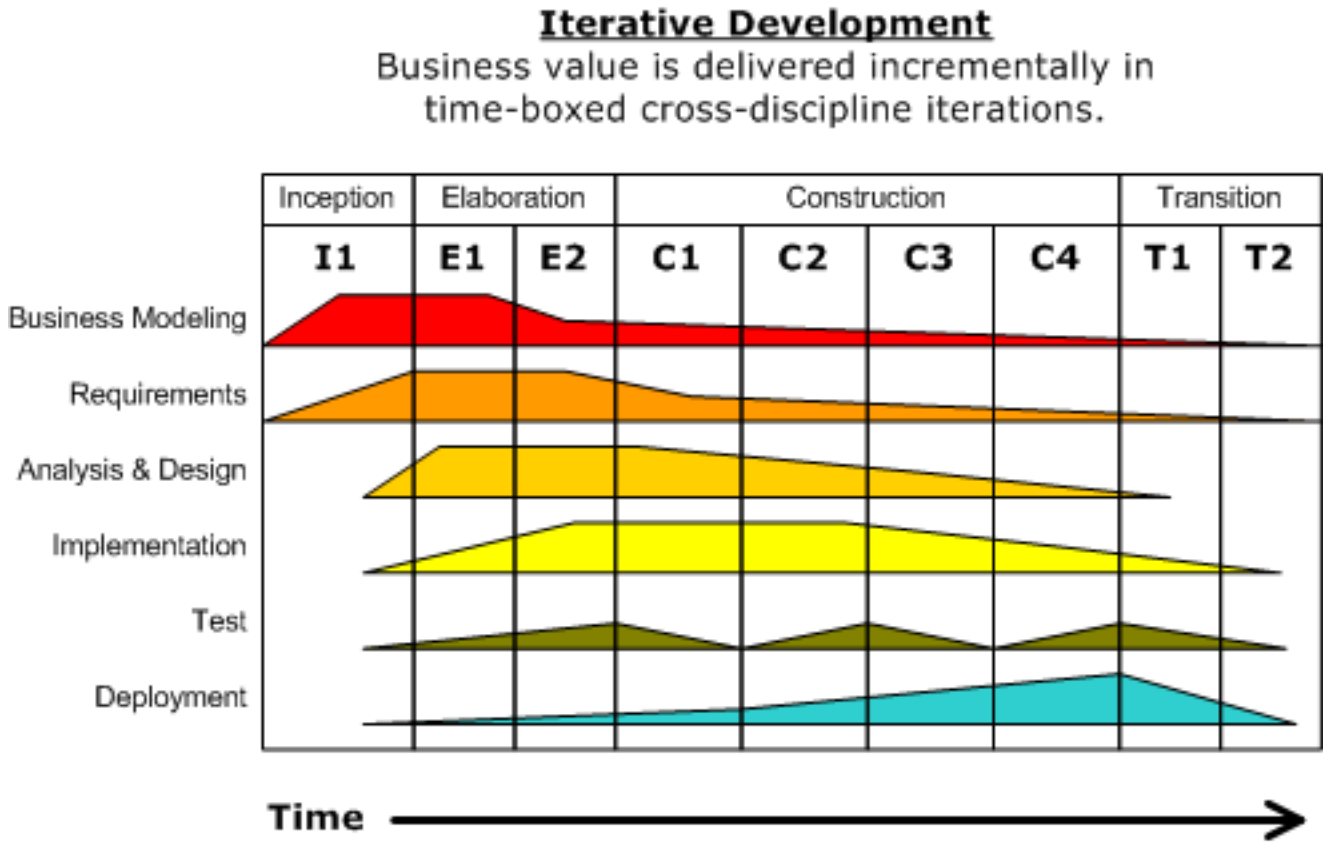
RISK AND THE WATERFALL APPROACH



ITERATIVE VERSUS WATERFALL



PHASES OF DEVELOPMENT



RUP IS USE CASE DRIVEN

We must know what prospective users want and need

- Example: ATM transaction
  - A person inserts the card, replies to a series of questions on the screen, and receives cash.
  - The machine, in response to the card and answers, performs a series of actions
  - The sequence creates a use case diagram or model.

Use cases capture functional requirements

All of the use cases create form a use case model, which describes the complete functionality of the system.

USE CASES IN THE PROCESS

Use case strategy can answer:

- What is the system supposed to do for each user?
- This strategy drives our focus on Value to the user

Use cases drive the development process

- Developers create design and implementation models
  - Testers test the implementation
  - Ensure components model the use cases correctly
- Use cases developed in tandem with system architecture.
  - Use cases drive architecture
  - Architecture influences the selection of the use cases.

RUP IS ARCHITECTURE-CENTRIC

The architecture grows out of the need of the enterprise as reflected by the use cases

Other influences:

- Platform
- Framework
- Deployment Considerations
- Legacy systems
- Nonfunctional requirements

Architecture is a high level view of the whole design.

ARCHITECTURE AND USE CASE "BALANCE"

Every product has function (use case) and form (architecture).

- Use cases must fit into the architecture
- Architecture must be scalable
- Both must evolve in parallel

RUP IS ITERATIVE AND INCREMENTAL

Development divided in to slices, or mini projects

- The mini project is an iteration that results in an increment
- Iterations → steps in a workflow
- Increments → growth in the product

Developers select what is implemented based on:

- The iteration deals with a group of use cases that together extend the usability of the product as developed
- The iteration deals with the most important risks
- Successive iterations built upon development artifacts from state at end of previous iteration

MINI PROJECT

Continues through consequent development work

- Analysis, design, implementation, test

Iterations not always additive in the early phases

- May involve replacing a superficial design with a more sophisticated one
- Can be reductive

BENEFITS OF ITERATIVE APPROACH

Easily controlled in terms of time, space, resource, as well as implementation

- Risks mitigated earlier
  - Reduces the risk of cost if repeated
- Accommodation of changing requirements
- Reusability
- Product is improved *and* refined
- Reduces the risk of time to market
- Speeds the development effort
- Acknowledges user needs

BENEFITS OF ITERATIVE APPROACH

Accommodating Changes

- With iteration, changes are much more easily incorporated into the project because the project is continuously changing anyway
- Changing requirements is ***inevitable***

Higher Quality

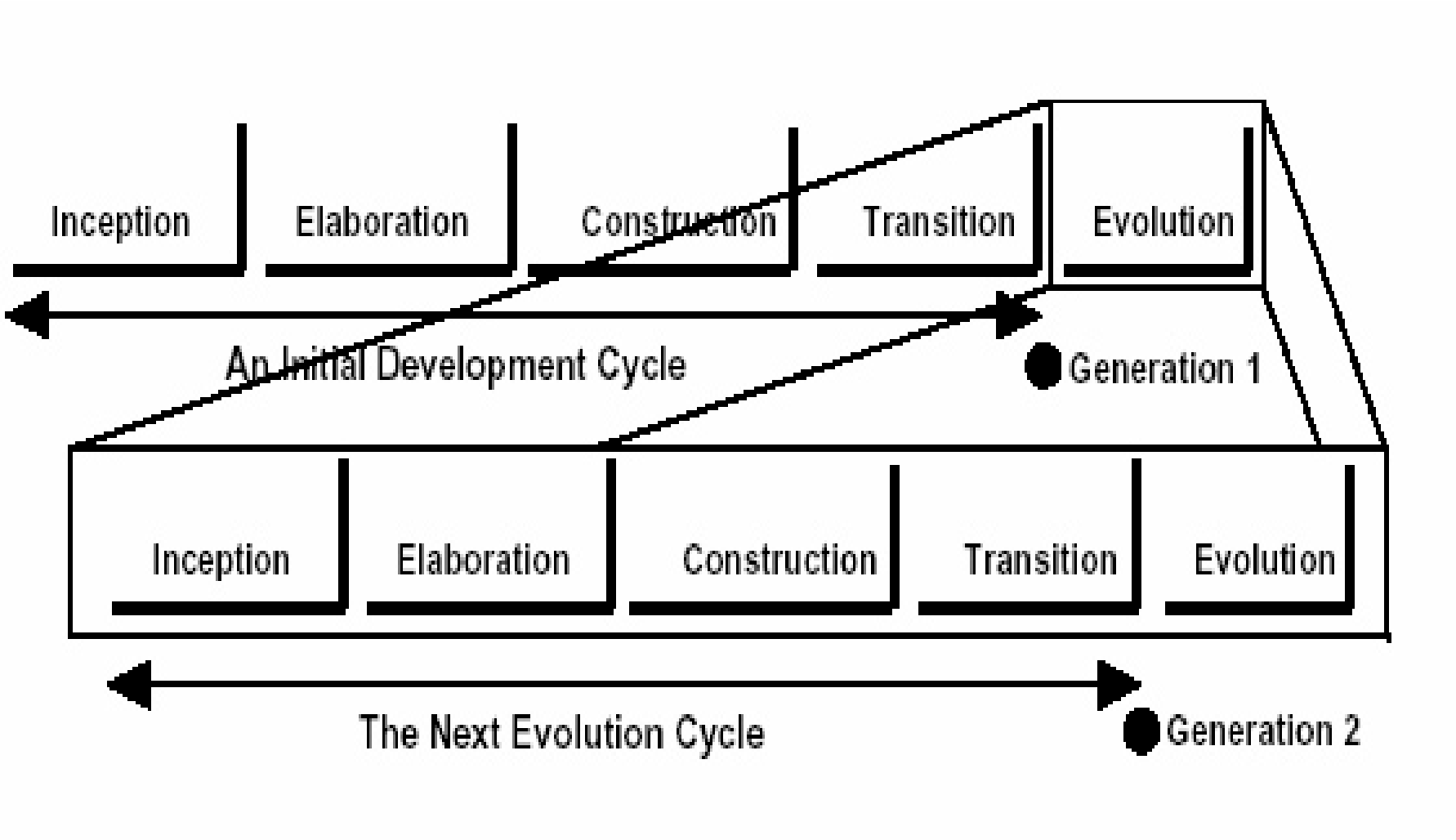
- Early flaws are corrected early
- Performance bottlenecks are discovered early
- Non-critical flaws (otherwise skipped) can be corrected later in development

BENEFITS OF ITERATIVE APPROACH

Learning and Improving

- Each workflow gets involved early and continuously contributes to the project
- Managing the need for additional personnel is easier
- Developer skills are honed as the project progresses

INTRODUCTION TO RATIONAL UNIFIED PROCESS



ORGANIZATION AND STRUCTURE

Before the Inception Phase Begins

- If your company develops software for resale, then a good start on the inception phase is available, driven by the marketing research
- Software departments that produce systems for internal use:
  - Typically an internal department has a need,
    - Makes a case to management
    - Provides the software group with very little information during inception
  - Management decides on an enterprise wide system
    - Establishes a business engineering group to determine what it should support
- Outsourced software development companies respond to an RFP

INCEPTION ENTRY

Criteria

- The expression of a need
  - an original vision
  - a legacy system
  - an RFP (request for proposal)
  - the previous generation and a list of enhancements
  - some assets (software, know-how, financial assets)
  - a conceptual prototype, or a mock-up
  - A re-engineering effort is needed

THE INCEPTION PHASE

The goal of the inception phase is to make a business case to the extend needed to justify launching the project.

During the inception phase, a business case for the system is established and we delimit the project scope.

- We 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.
  - The business case includes success criteria, risk assessment, and estimate of the resources needed, and a phase plan showing dates of major milestones.

THE INCEPTION PHASE

A beginning step is to delimit the scope of the proposed system

- Needed to discern what the project entails
  - To understand what the architecture has to cover
    - Just enough architecture to ensure it can support the system scope
  - To define the limits we will prospect for critical risks
  - Define boundaries for cost, schedule, R.O.I. estimates
- We are also trying to avoid one of those brilliant, famous, career-ending fiascos like the Mars Lander

THE INCEPTION PHASE

The outcome of the inception phase is:.

- An initial use-case model (10% -20%) complete).
- An initial project glossary (Data Dictionary)
- 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.

THE INCEPTION PHASE

Success criteria

- A revenue projection
- An initial risk assessment
- An estimate of the resources required to complete the elaboration phase.
- Optionally at the end of the inception phase, we may have:
  - An initial domain analysis model (~10%-20% complete)
    - Identifying the top key use cases, and sufficient to drive the architecture effort.
  - An initial architectural prototype
    - which at this stage may be a throw-away prototype.

THE INCEPTION PHASE

Exit criteria

- An initial business case containing at least:
  - A clear formulation of the product vision —the core requirements:
    - In terms of functionality
    - Scope
    - Performance
    - Capacity
    - Technology base

THE "GO-NO GO" SCENARIO

At the end of the inception stage, a decision is made.

- Go forward with the project (GO)
- End the project for whatever reason (NO)
  - Too much risk
  - No market
  - Not enough R.O.I.

DECISIONS

A “go” decision leads to

- Further research
- Requirements gathering
- Use case development

Several “go-no go” in early stages of the project.

- Always a series of review meetings throughout the course of a project
- As more information is gathered
  - feasibility becomes clearer
  - risks more obvious
  - needs more clearly defined.

A “Go” decision invokes the next stage, Elaboration

ELABORATION PHASE

The purpose of this phase:

- Thoroughly analyze the problem domain
- Define and stabilize the architecture
- Address the highest risk elements of the project
  - Enable production of comprehensive plan for next two phases
  - Produce an initial set of requirements
- Evaluation criteria for at least the first construction iteration

ELABORATION PHASE

Purpose

- Develop a baseline software architecture
- Discover the resources necessary to develop and deploy the product
  - People
  - Tools
- Develop a schedule (Gantt Chart)
- Determine a resolution of the risks sufficient to make estimates
  - Cost
  - Schedule
  - Quality

ELABORATION PHASE

Executable architectural prototype is built

- In one or more iterations depending on the scope, size, risk, etc
- Addresses the top key use cases identified in the inception phase
- Addresses the top technical risks of the project.

Evolutionary prototype

- Production quality code
- Becomes the architectural baseline
  - No throw-away prototypes
  - Refinements of the requirements feasibility
  - Human-interface studies, demonstrations to investors, etc.

ELABORATION PHASE

Entry criteria:

- The products and artifacts described in the exit criteria of the Inception phase.
- The plan approved by:
  - Project management
  - Funding approved
  - The resources required for the elaboration phase have been allocated.

ELABORATION PHASE

Exit criteria

- A detailed software development plan containing:
  - Updated risk assessment
  - Management plan
  - Staffing plan
  - Phase plan
  - Iteration plan
  - Development environment and other tools required

ELABORATION PHASE

Exit criteria

- Test plan
- Baseline vision
- Objective, measurable evaluation criteria
- Domain analysis model (80% complete),
- Software architecture description
- Executable architecture baseline.

ELABORATION PHASE

Another “go – no go” Decision

- To actually invest and build the product
- Or bid for the complete development of the contract.
- The plans produced must be detailed
- The risks sufficiently mitigated
- Decision Makers must have enough information to be able to determine with accuracy, the cost and schedule for the completion of the development.

THE CONSTRUCTION PHASE

Remaining components and features developed, integrated into product and tested

- Manufacturing process:
  - emphasis is placed on managing resources and controlling operations
  - costs, schedules, and quality
  - management transitions from the IP to products

THE CONSTRUCTION PHASE

In large projects, parallel construction increments can be spawned:.

- Significantly accelerate the availability of deployable releases
- Also increase the complexity of resource management and workflow synchronization.
- A robust architecture and an understandable plan are highly correlated.
- At minimum, it consists of:
  - The software product integrated on the adequate platforms.
  - The user manuals.
  - A description of the current release.

THE CONSTRUCTION PHASE

At the end of the construction phase is the third major project milestone

- decide if the software, the sites, and the users are ready to go operational
  - This release is often called a “beta” release.

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?
- Transition may have to be postponed by one release if the project fails to reach this milestone.

THE TRANSITION PHASE

Focuses on the activities required to place the software into the hands of the users.

- Typically, this phase includes several iterations, including:
  - Beta releases
  - General availability releases
  - Bug-fix es
  - Enhancement releases
  - Developing user-oriented documentation
  - Training users
  - Supporting users in their initial product use
  - Reacting to user feedback

THE TRANSITION PHASE

The primary objectives of the transition phase include:

- Achieving user self-supportability
- Achieving stakeholder concurrence:
  - That deployment baselines are complete
  - Consistent with the evaluation criteria of the vision
- Achieving final product baseline as rapidly and cost effectively as practical

THE TRANSITION PHASE

This phase can range from being very simple to extremely complex

- For example:
  - new release of an existing desktop product may be very simple
  - replacing a nation's air-traffic control system would be very complex.

- At the end of the transition phase is the fourth important project milestone
  - Product Release Milestone.
    - You decide if the objectives were met
    - If you should start another development cycle.
  - In some cases, this may coincide with the end of the inception phase for the next cycle.

THE TRANSITION PHASE

The primary evaluation criteria for the transition phase:

- Is the user satisfied?
- Are the actual resources expenditures versus planned expenditures still acceptable?

RUP CORE WORKFLOWS

The core process workflows are divided into 6 “engineering” workflows

- Business Modeling
- Requirements
- Analysis and Design
- Implementation
- Testing
- Deployment

There are also 3 “supporting” workflows

- Configuration and Change Management
- Project Management
- Environment

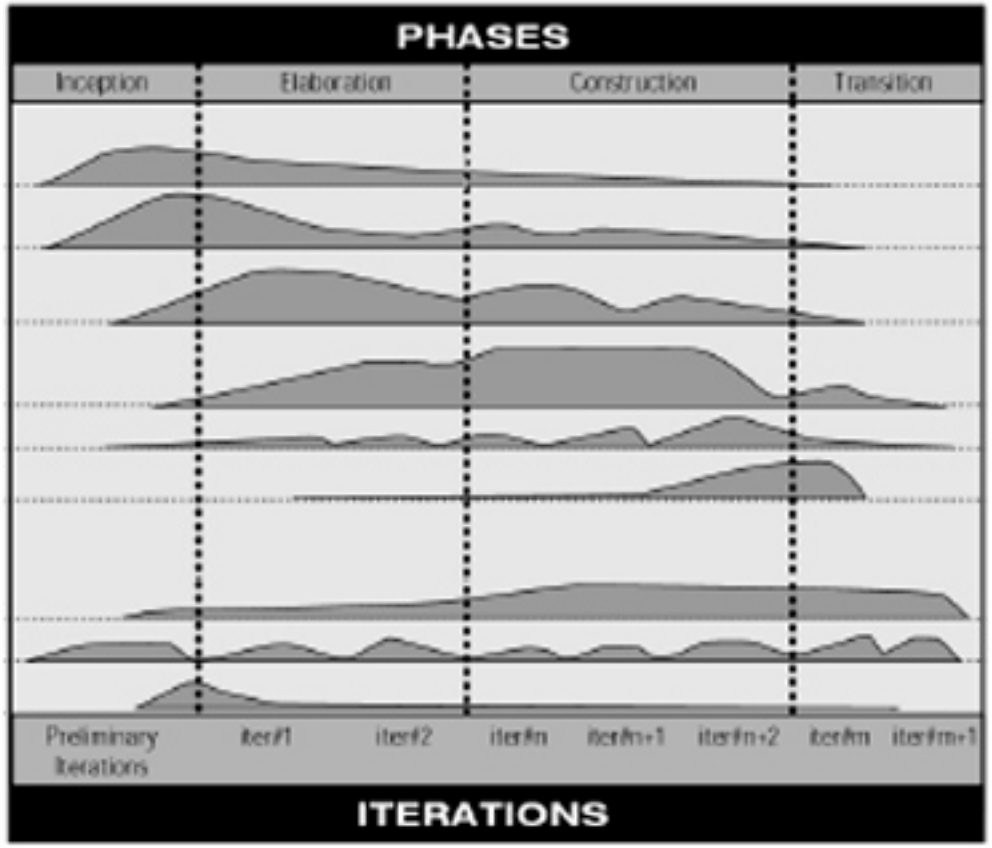
CORE WORKFLOWS

Core Process

- Business Modeling
- Requirements
- Analysis /Design
- Implementation
- Test
- Deployment

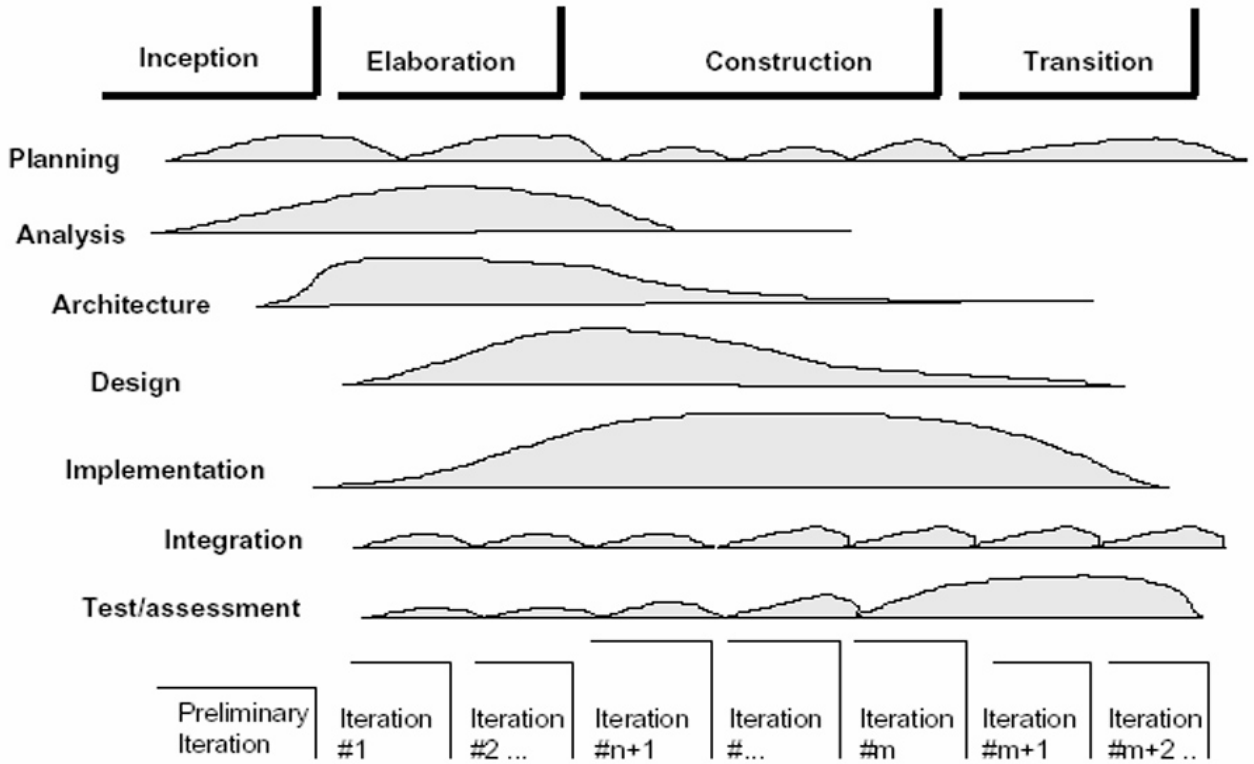
Core Supporting

- Configuration/Change Mgmt
- Project Management
- Environment



The nine core process workflows

ORGANIZATION BY DISCIPLINE



This change of focus explains also that, although all structured in the same way, the exact nature and contents of the iterations evolves over time.

WORKFLOWS AND MODELS

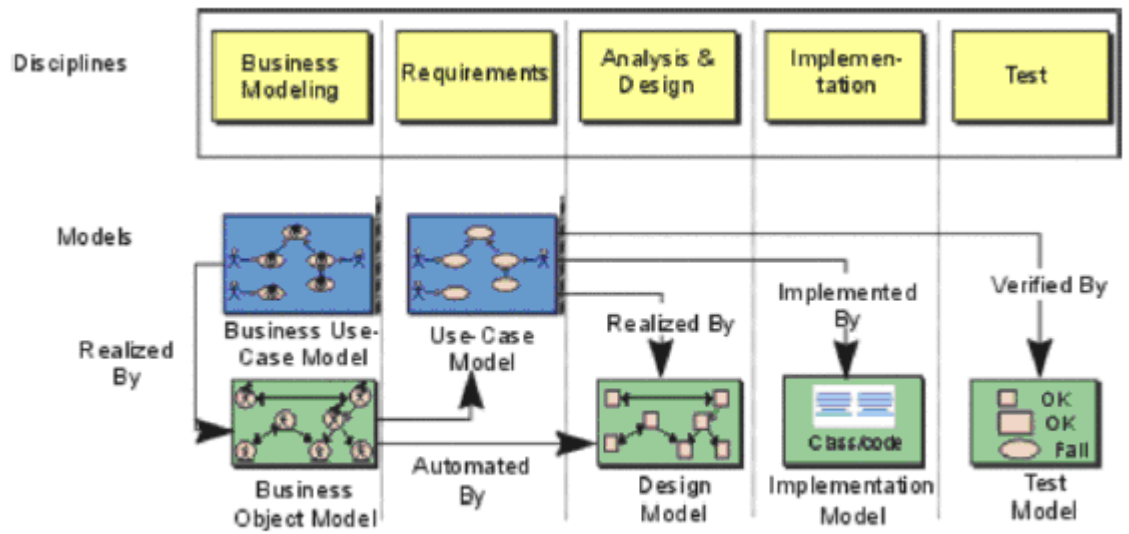


Figure 4. Each core process workflow is associated with a particular set of models.

BUSINESS MODELING WORKFLOW

A major problem with many business engineering efforts, is the lack of communication between Software Engineering and Business Engineering.

RUP addresses this by providing a common language and process for both communities.

- Also by showing how to create and maintain direct traceability between business and software models.

In Business Modeling we document business processes using business use cases.

- This assures a common understanding among all stakeholders of what business process needs to be supported in the organization.

REQUIREMENTS WORKFLOW

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.

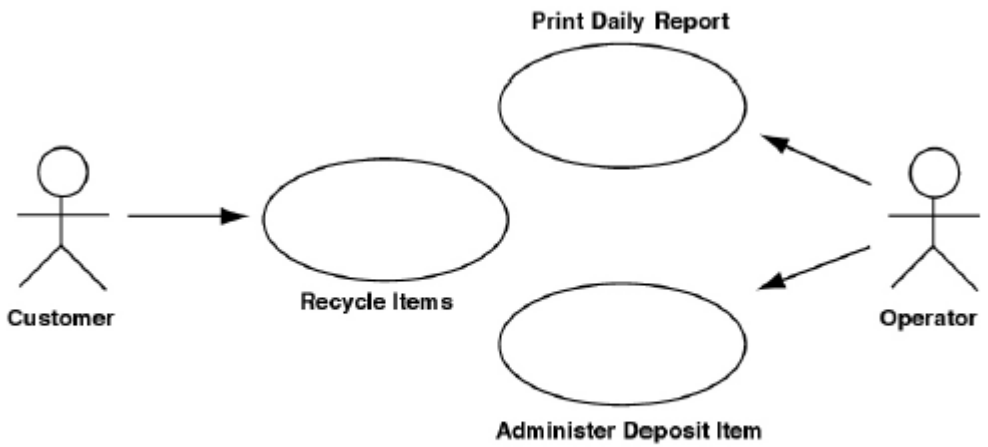
REQUIREMENTS WORKFLOW

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.

REQUIREMENTS WORKFLOW



An example of use-case model with actors and use cases.

ANALYSIS AND DESIGN WORKFLOW

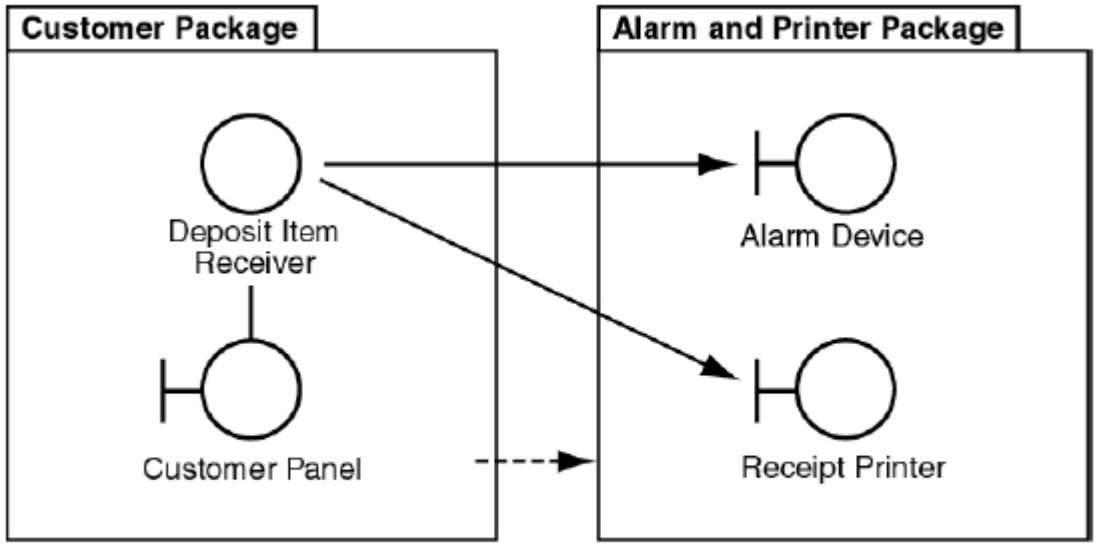
The Analysis & Design workflow is to show *how* the system will be *realized* in the implementation phase. You want to build a system that:

- 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.

ANALYSIS AND DESIGN WORKFLOW



Part of a design model with communicating design classes, and package group design classes.

IMPLEMENTATION WORKFLOW

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.

- RUP describes how you reuse existing components, or implement new components with well defined responsibility
  - This makes the system easier to maintain, and increasing the possibilities to reuse.

TEST WORKFLOW

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 RUP 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 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.

DEPLOYMENT WORKFLOW

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.

While 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.

PROJECT MANAGEMENT WORKFLOW

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 and the users.

- This workflow focuses mainly on the specific aspect of an iterative development process. Our 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.

CONFIGURATION AND CHANGE MANAGEMENT WORKFLOW

This workflow describes how to control the numerous artifacts produced.

- 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.

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.

ENVIRONMENT WORKFLOW

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.

BEST PRACTICES

Develop Iteratively

- Manage Requirements
- Use Component Architectures
- Model Visually
- Continuously Verify Quality
- Manage Change

BEST PRACTICES

BEST PRACTICES

What is Iterative Development?

- A series of small steps in the total project development
- Each iteration develops a portion of the final project
- Each iteration involves each of the traditional development steps (requirements, design, code, test, etc.)

BEST PRACTICES

Manage Requirements

What is Requirements Management?

- Finding, documenting, and organizing changing requirements
- Ensuring descriptions are clear
- Including traceability

BEST PRACTICES

Why are Requirements Difficult?

- Requirements are not always obvious
- Requirements not always clearly expressed
- Requirements are at different levels of detail and are of different types
- Requirements are intertwined
- Requirements are connected to many key people, so must be managed by cross-functional people
- Requirements change

BEST PRACTICES

What is Needed for Requirements Management?

- Problem analysis
- Understanding of stakeholder needs
- Definition and refinement of system
- Management of scope of project

BEST PRACTICES

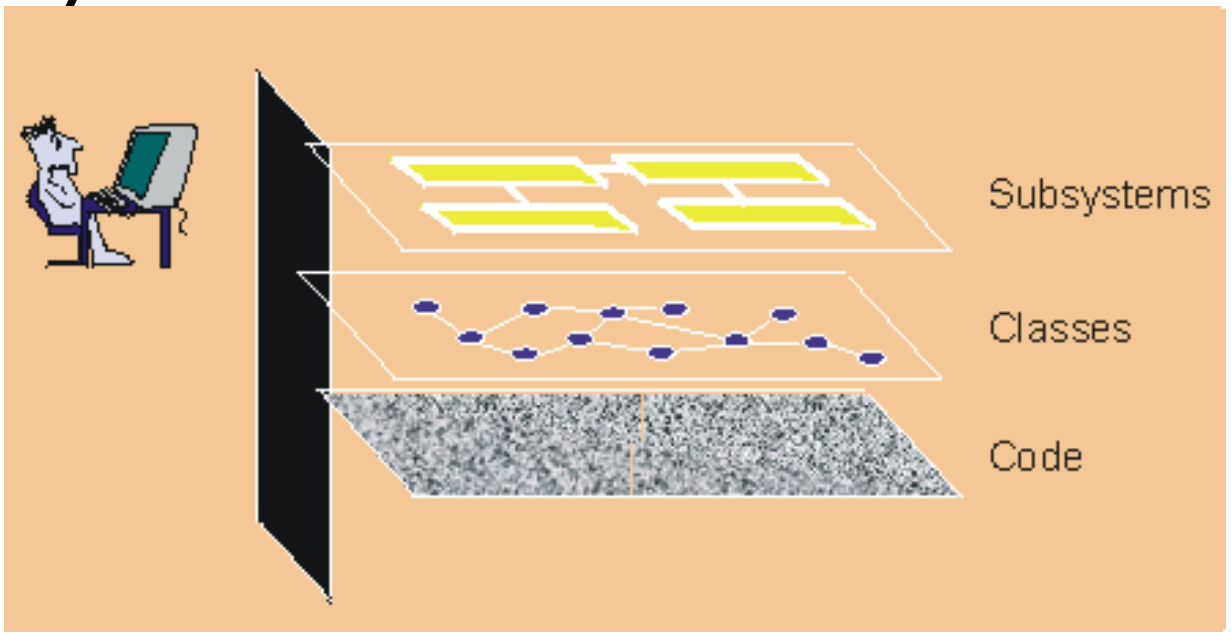
Use Component Architectures

- Components
  - Cohesive groups of code
  - Well defined interfaces
  - Provide strong encapsulation
  - Replaceable
- Reduces size and complexity of the project
  - Robust
  - Resilient

BEST PRACTICES

Model Visually

- Use UML!



BEST PRACTICES

What is Visual Modeling?

- Combination graphical and textual notations
- Defined syntax and semantics
- Reduced ambiguity
- Easy understandability

BEST PRACTICES

Why Do We Model?

- Simplification of view
- Uniqueness of perspectives
- Simple and clear communication
- Easy comparison with other designs
- Assists understanding complex systems

BEST PRACTICES

Continuously Verify Quality

- How is Quality Verified?
  - All the artifacts in a project must be continually evaluated and assessed (example: 70% complete, 50% correct, etc.)
  - All attributes of quality must be assessed
  - Quality must be ensured by all and not simply the responsibility of one specific group
- Quality Attributes (examples) Completeness
  - Consistency
  - Simplicity
  - Expandability
  - Modularity
  - Error Tolerance

BEST PRACTICES

Misconceptions regarding Quality

- Quality can be added
- Quality is an attribute, dimension, and means the same thing to all
- Quality is a natural by-product of development
- We did it this way last time and everyone was happy
- We'll fix that later
- You wanted it to work???

BEST PRACTICES

Manage Change

- Coordinating Activities and Artifacts
  - Having repeatable and agreed-upon procedures to manage change
  - Active discovery and reactions to problems
  - Allocation of resources based on priorities and risks
- Coordinating Iterations and Releases
  - Maintaining a baseline at each iteration
  - Maintaining traceability of elements within and between multiple releases

BEST PRACTICES

Controlling Change

- Workflow of requirements change is defined and repeatable
- Clear communication of change request
- Change rate statistics are tracked
- All artifacts are available in workspaces
- Maintenance of change is robust and flexible

CONCLUSION

What did we learn?

- UML is graphic
- RUP is iterative
- Modeling is a *GOOD* thing
- Quality counts and extends throughout the process