Rational Unified Process

Rational Unified Process (RUP)

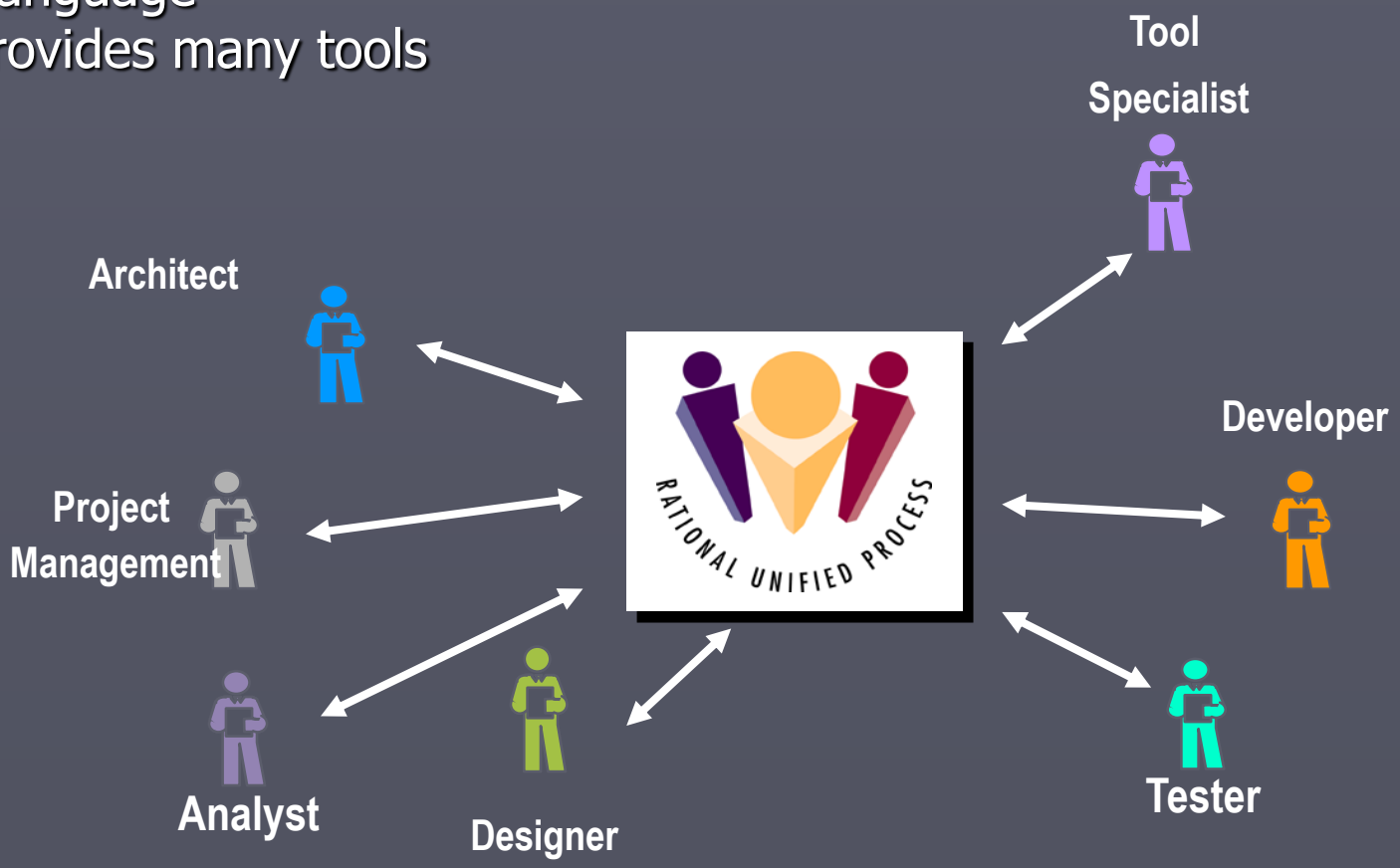
- Introduction
- Phases
- Core Workflows
- Best Practices

Team-Unifying Approach

The RUP unifies a software team by providing a common view of the development process and a shared vision of a common goal

Increased Team Productivity

- knowledge base of all processes
- view of how to develop software
- modeling language
- Rational provides many tools



Rational Unified Process (RUP)

time

content

Phases

Process Workflows

Business Modeling

Requirements

Analysis & Design

Implementation

Test

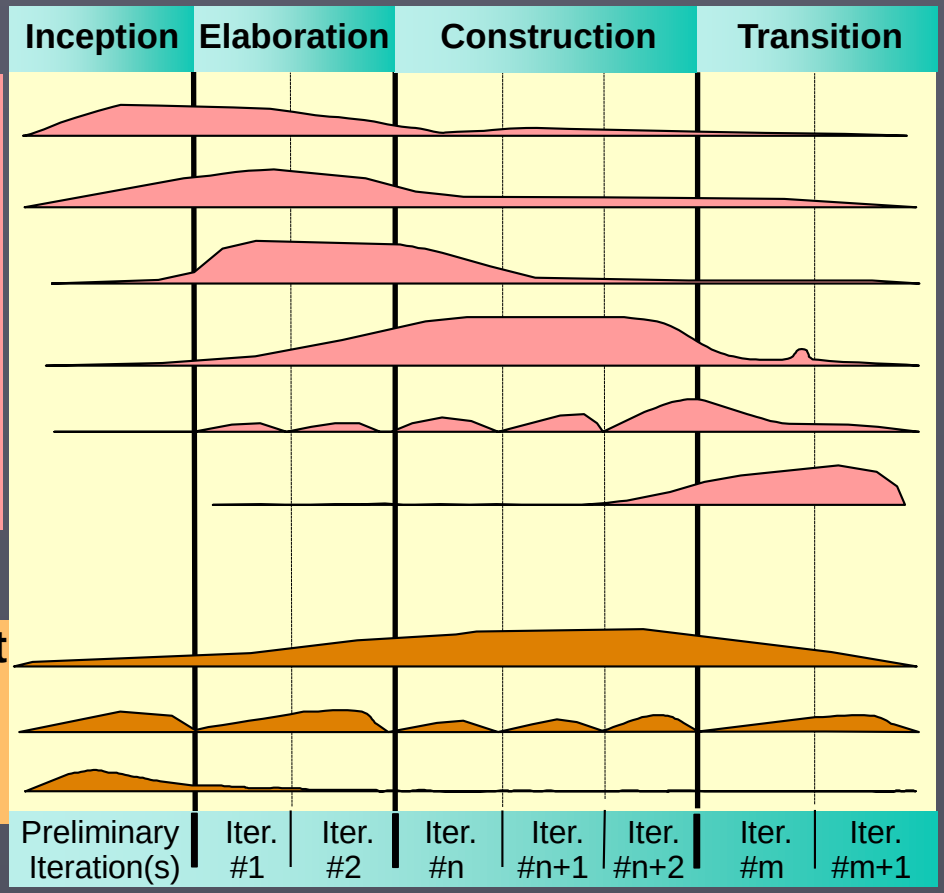
Deployment

Supporting Workflows

Configuration & Change Mgmt

Project Management

Environment

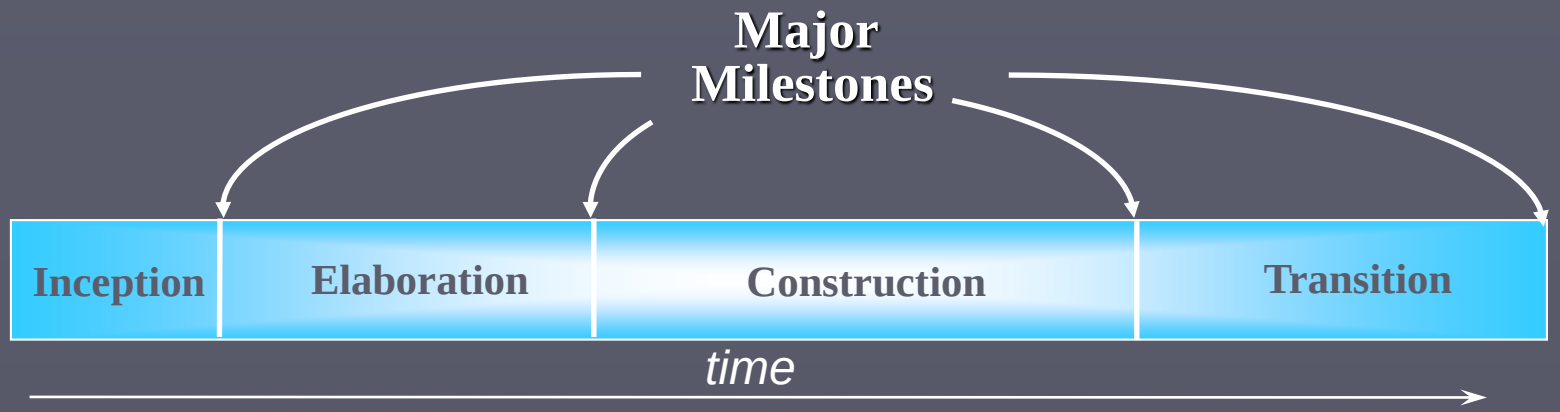


Iterations

Rational Unified Process (RUP)

- Introduction
- Phases
- Core Workflows
- Best Practices

Phases in the Process



The Rational Unified Process has four phases:

- Inception - Define the scope of project
- Elaboration - Plan project, specify features, baseline architecture
- Construction - Build the product
- Transition - Transition the product into end user community

Inception Goals

- Establishing the project's software scope and boundary conditions, including:
  - an operational vision
  - acceptance criteria
  - what is intended to be in the product
  - what is not.
- Discriminating
  - the critical use cases of the system
  - the primary scenarios of operation that will drive the major design trade-offs.
- Exhibiting, and maybe demonstrating, at least one candidate architecture against some of the primary scenarios

Inception Goals (Cont.)

- Estimating
  - the overall cost
  - and schedule for the entire project
  - and more detailed estimates for the elaboration phase that will immediately follow
- Estimating potential risks (the sources of unpredictability)
- Preparing the supporting environment for the project.

Inception Essential Activities

- Formulating the scope of the project.
- Planning and preparing a business case.
- Synthesizing a candidate architecture.
- Preparing the environment for the project.
- …

Inception Artifacts

- **Vision:** The project's core requirements, key features, and main constraints are documented. Stakeholders …
- **Glossary:** defines important terms used by the project.
- **Business Case:** provides the necessary information from a business standpoint to determine whether or not this project is worth investing in.
- **Software Development Plan**: all information required to manage the project. (Risk, time and durations, needed tools, changes, documentations)
- **Use-case model:** a model of the system's intended functions and its environment, and serves as a contract between the customer and the developers.

Elaboration Goals

- To ensure stability of:
  - Architecture;
  - Requirements;
  - Plans.
- To be able to predictably determine:
  - Cost;
  - Schedule.
- To address all significant risks of the project and to ensure all of them will be mitigated.
- To establish a baseline architecture
  - Derived from addressing the architectural significant scenarios

Elaboration Goals (Cont.)

- To produce an evolutionary prototype
- Verify baseline architecture
  - Demonstrate that the architecture will support requirements of the system at a reasonable cost and time.
- To establish a supporting environment.

Elaboration Activities

- Defining, validating the baseline architecture.
- Refining the Vision.
- Creating detail of iteration plans for the construction phase.
- Refining the development case and putting in place the development environment
- Refining the architecture and selecting components.

Elaboration Artifacts

- **Software Architecture Document**: provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system.
- **Prototypes**: One or more executable architectural prototypes have been created to explore critical functionality and architecturally significant scenarios.
- **Design model:** an object model describing the realization of use cases, and serves as an abstraction of the implementation model and its source code.
- **Data model:** a subset of the implementation model which describes the logical and physical representation of persistent data in the system.
- Testing Mechanisms and refining previous Iteration's artifacts.

Construction Goals

- Completing the analysis, design, development and testing of all required functionality.
- Achieving useful versions (alpha, beta and other test releases)
- Achieving adequate quality as rapidly as practical
- To decide if the software, the sites and the users are all ready for the application to be deployed.
- Minimizing development costs by optimizing resources and avoiding unnecessary scrap and rework.
- To achieve some degree of parallelism in the work of development teams.

Construction Activities

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

Construction Artifacts

- **The System**: The executable system itself, ready to begin "beta" testing.
- **Training materials:** the material that is used in training programs or courses to assist the end-users with product use, operation and/or maintenance.
- Testing results and refining previous Iteration's artifacts.

Transition Goals

- Beta testing to validate the new system against user expectations
- Beta testing and parallel operation relative to a legacy system that it is replacing
- Training of users and maintainers
- Roll-out to the marketing, distribution and sales forces
- Tuning activities such as bug fixing, enhancement for performance and usability

Transition Goals (Cont.)

- Achieving user self-supportability
- Achieving stakeholder concurrence that deployment baselines are complete

Transition Activities

- Executing deployment plans
- Finalizing end-user support material
- Testing the deliverable product at the development site
- Creating a product release
- Getting user feedback
- Fine-tuning the product based on feedback
- Making the product available to end users

Transition Artifacts

- **Product**.
- **Release Notes:** identify changes and known bugs in a version of a build or deployment unit that has been made available for use.
- **Installation Artifacts:** refer to the software and documented instructions required to install the product.
- **End-User Support Material:** Materials that assist the end-user in learning, using, operating and maintaining the product.
- Testing results and refining previous iteration's artifacts.

Rational Unified Process (RUP)

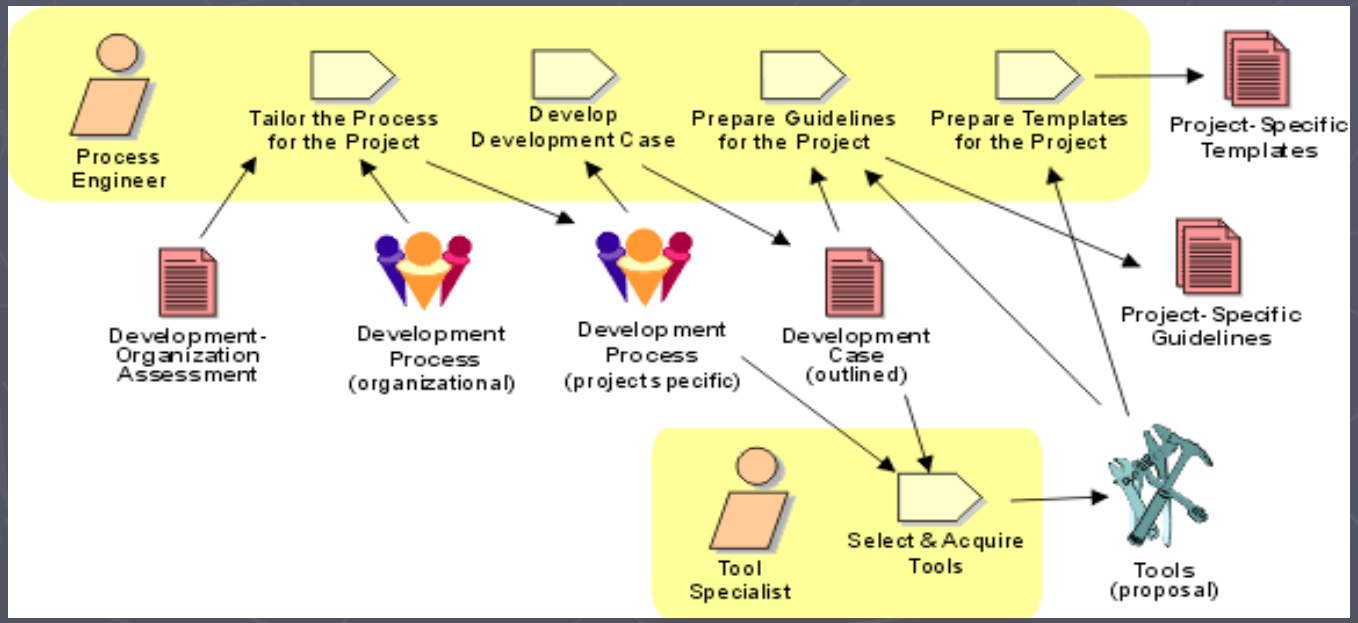
- Introduction
- Phases
- Core Workflows
- Best Practices

What is a workflow?

- A set of activities that is performed by the various roles in a project
- Describes a meaningful sequence of activities that produce a useful result (an artifact)
- Shows interaction between roles

Workflows - 3 key elements

- Three key elements of each workflow:
  - Artifacts
  - Roles
  - Activities
- Workflow Detail: Prepare Environment for Project



Artifacts

- A piece of information that:
  - Is produced, modified, or used by a process
  - Defines an area of responsibility
  - Is subject to version control.
- An artifact can be a *model*, a *model element* or a *document*. A document can enclose other documents.

Roles

- Represent a role that an individual may play in the project
- Responsible for producing artifacts
- Distinct from actors

Activities

- Tasks performed by people representing particular roles in order to produce artifacts

Brief summary of process workflows

- Business Modelling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment

Business Modelling

- Understand structure & dynamics of organization in which system is to be deployed
- Understand current problems in the target organization & identify improvement potential
- Ensure customers, end users & developers have common understanding of target organisation
- Derive system requirements to support target organisation

Business Modelling Artifacts

- Business Vision
- Business Architecture Document
- Supplementary Business Specifications
- Business Rules (either as a document and/or as elements in the Business Analysis Model)
- Business Glossary

Requirements

- To establish and maintain agreement with the customers and other stakeholders on what the system should do.
- To provide system developers with a better understanding of the system requirements.
- To define the boundaries of (delimit) the system.
- To provide a basis for planning the technical contents of iterations.
- To provide a basis for estimating cost and time to develop the system.
- To define a user-interface for the system, focusing on the needs and goals of the users.

Requirements Artifacts

- Glossary
- Use-case model
- Vision document
- User-interface prototype

Analysis & Design

- Transform requirements into a design of the system
- Evolve a robust architecture for the system
- Adapt design to match the implementation environment, designing it for performance

Analysis & Design Artifacts

- Software architecture document
- Design model
- Data model

Implementation

- Define organization of the code, in terms of implementation subsystems organized in layers
- Implement classes & objects in terms of components
- Test developed components as units
- Integrate results into an executable system

Implementation Artifacts

- Build
- Implementation model

Test

- Verify interaction between objects
- Verify proper integration of all components of the software
- Verify that all requirements have been correctly implemented
- Identify & ensure defects are addressed prior to deployment

Test Artifacts

- Test Plan
- Test Results
- Test_Ideas List
- Test Evaluation Summary

Deployment

- Provide custom installation
- Provide shrink wrap product offering
- Provide software over internet

Deployment Artifacts

- Deployment Plan
- End-user support material

Brief summary of supporting workflows

- Configuration & Change Management
- Project Management
- Environment

Configuration & Change Management

- Supports development methods
- Maintains product integrity
- Ensures completeness & correctness of configured product
- Provides stable environment within which to develop product
- Restricts changes to artifacts based on project policies
- Provides an audit trail on why, when & by whom any artifact was changed

Configuration & Change Management

Artifacts

- Change requests
- Configuration Management Plan
- Workspace

Project Management

- A framework for managing software-intensive projects
- Practical guidelines for planning, staffing, executing & monitoring projects
- A framework for managing risk

Project Management Artifacts

- Business case
- Iteration plan
- Iteration assessment
- Risk list
- Software development plan

Environment

- Design, implement and manage the project's required technical environments
- Define the technical architectures for the development, system validation, testing & staging/release management environments
- When possible, standard architectural models for given types of platforms should be utilized when defining the production environment

Environment Artifacts

- Development case
- Development infrastructure

Bringing It All Together...

In an iteration, you walk through all workflows

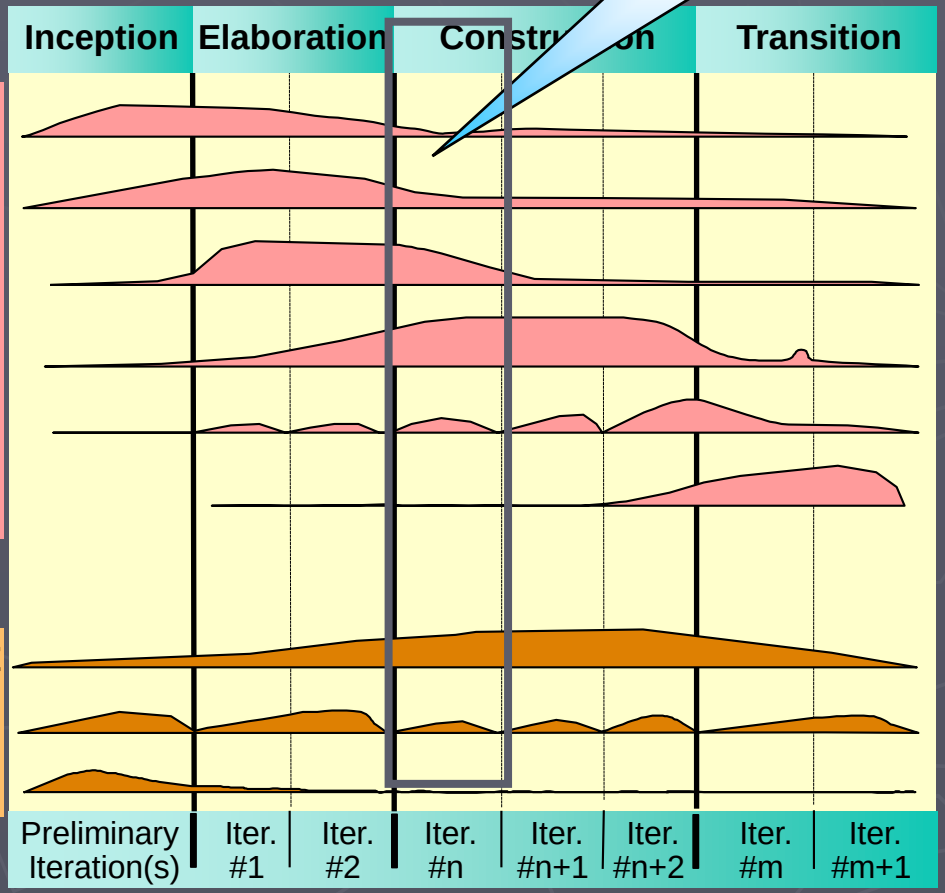
Phases

Process Workflows

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

Supporting Workflows

- Configuration & Change Mgmt
- Project Management
- Environment



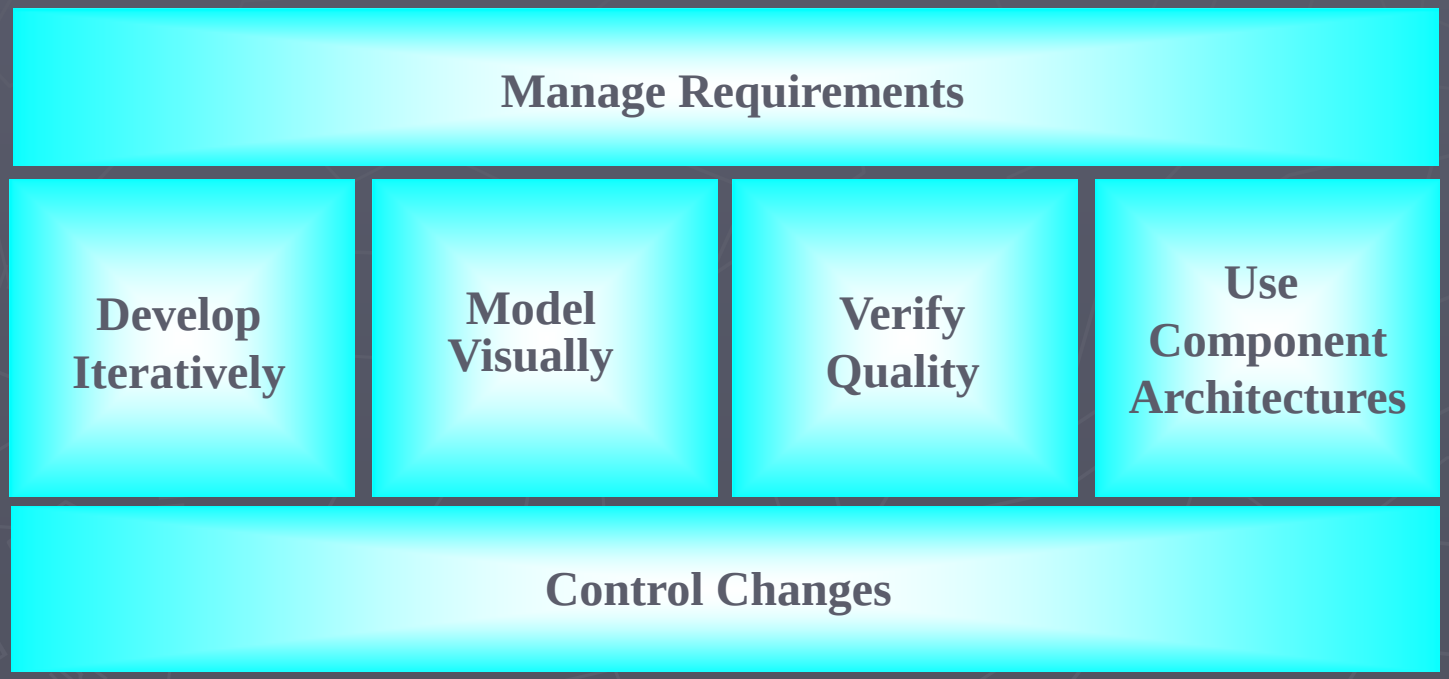
Iterations

Rational Unified Process (RUP)

- Introduction
- Phases
- Core Workflows
- Best Practices

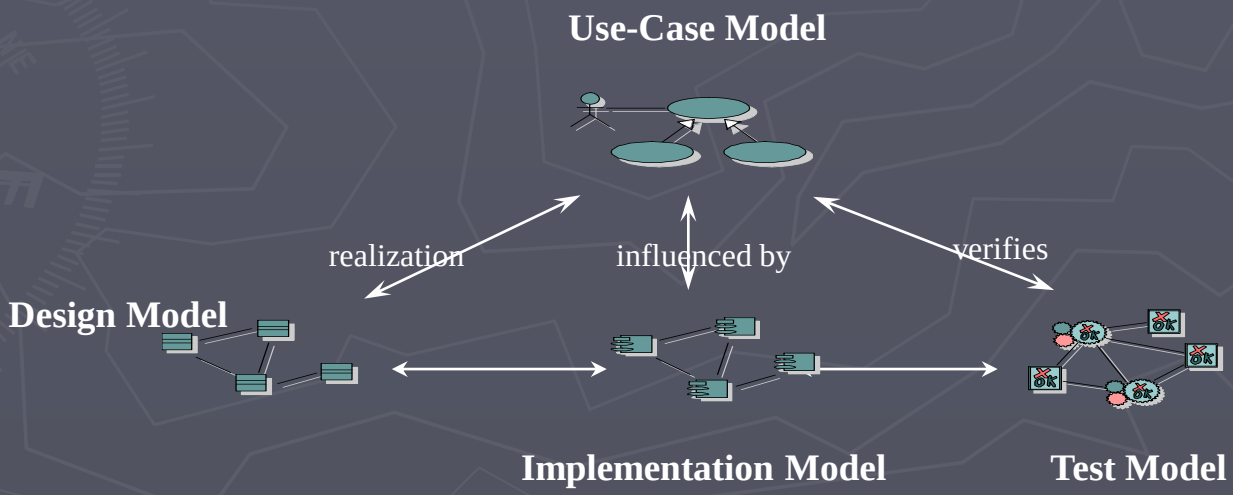
Rational Unified Process

Describes the effective implementation of key "Best Practices"



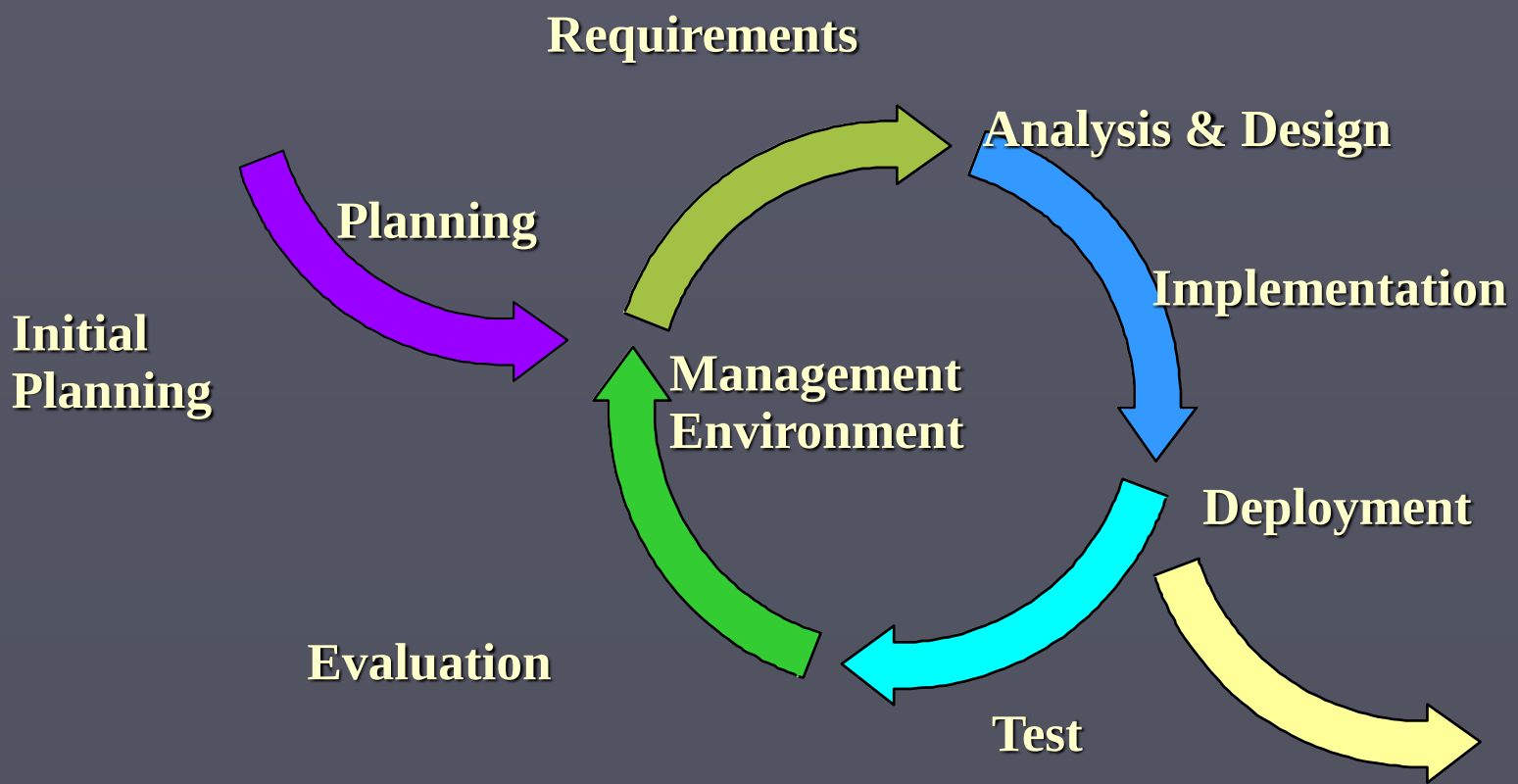
1. Manage Your Requirements

- Elicit, organize, and document required functionality and constraints
- Track and document tradeoffs and decisions
- Business requirements are easily captured and communicated through use cases
- Use cases are important planning instruments

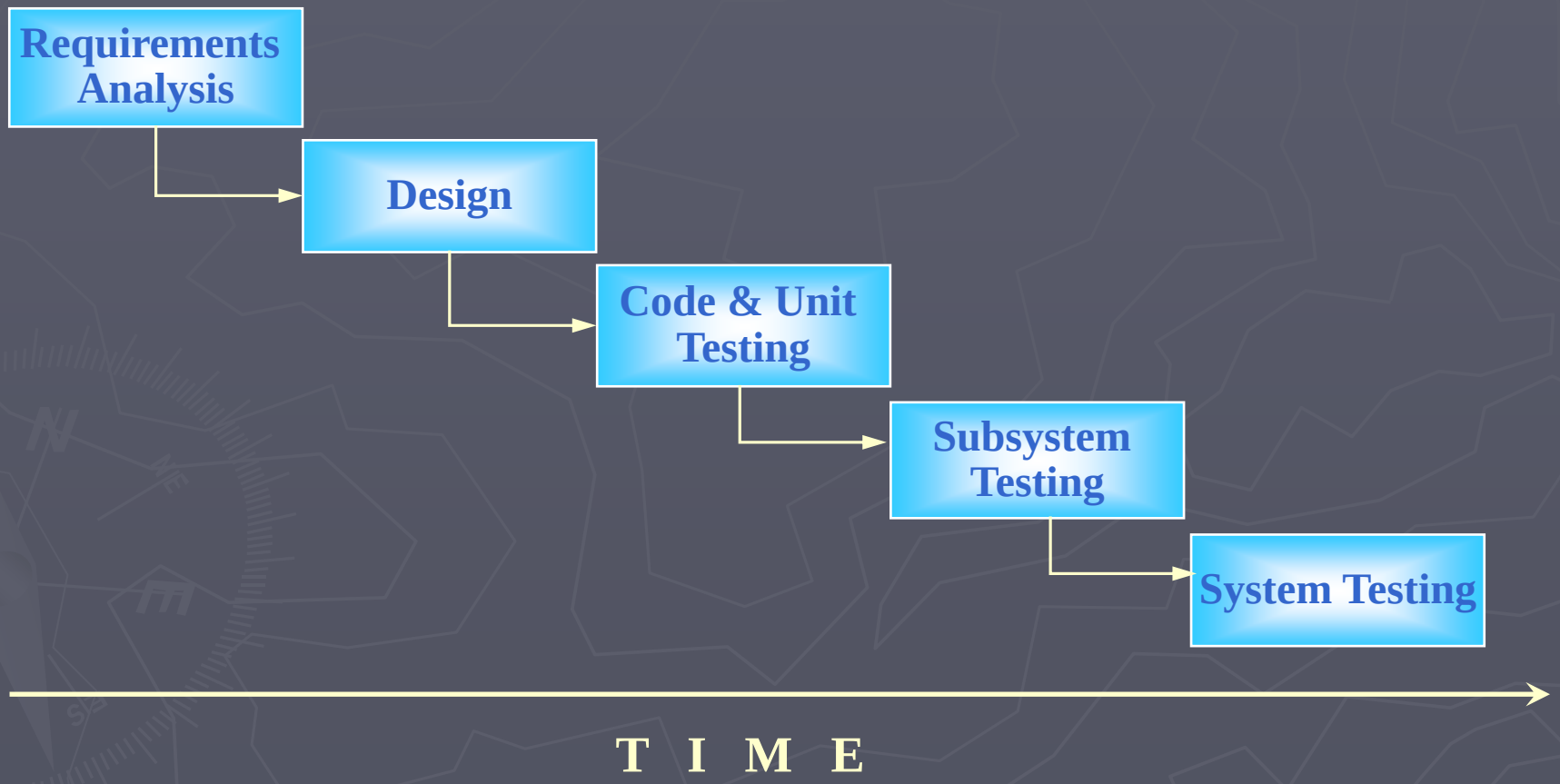


2. Develop Software Iteratively

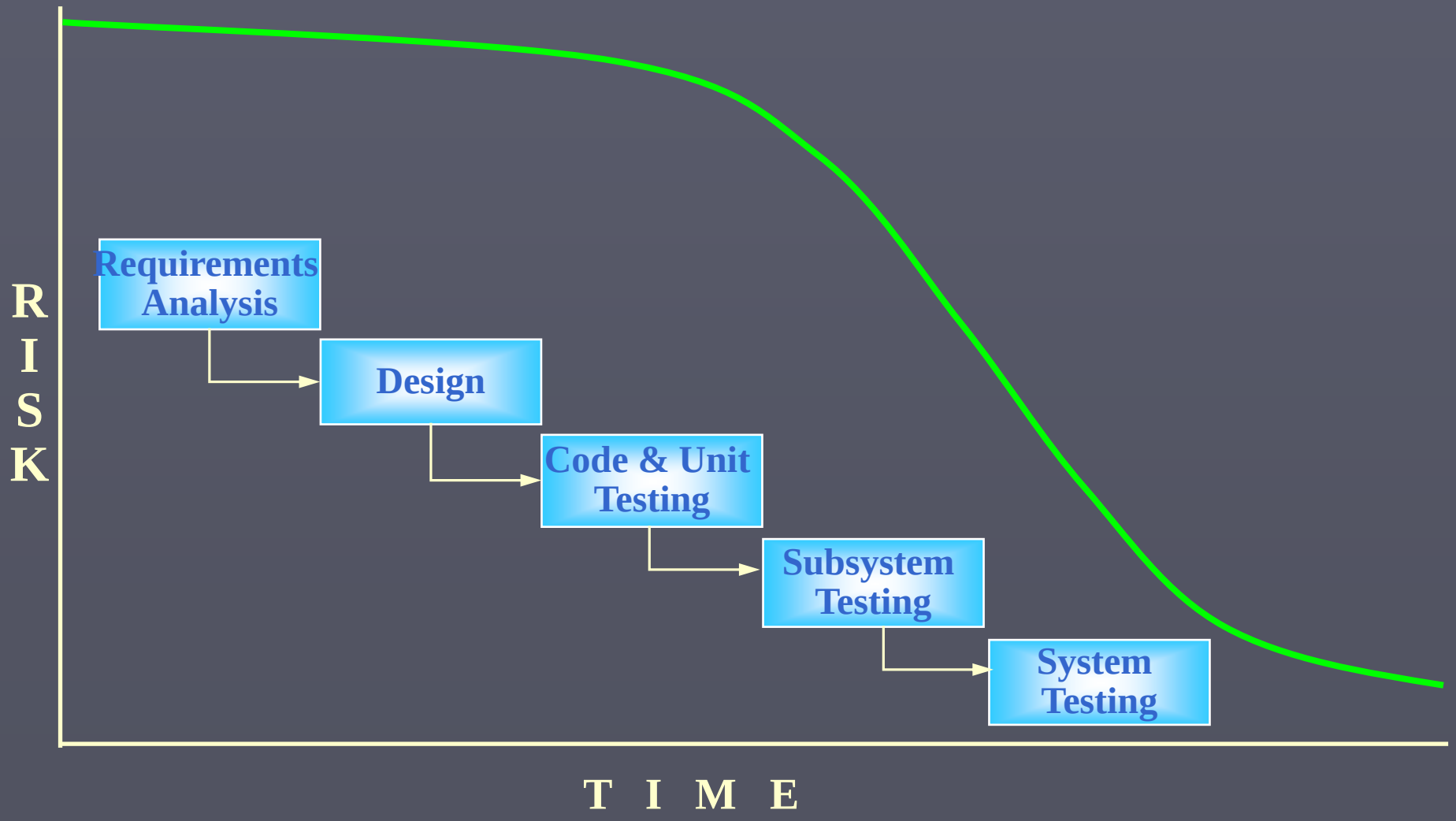
- An initial design will likely be flawed with respect to its key requirements
- Late-phase discovery of design defects results in costly over-runs and/or project cancellation



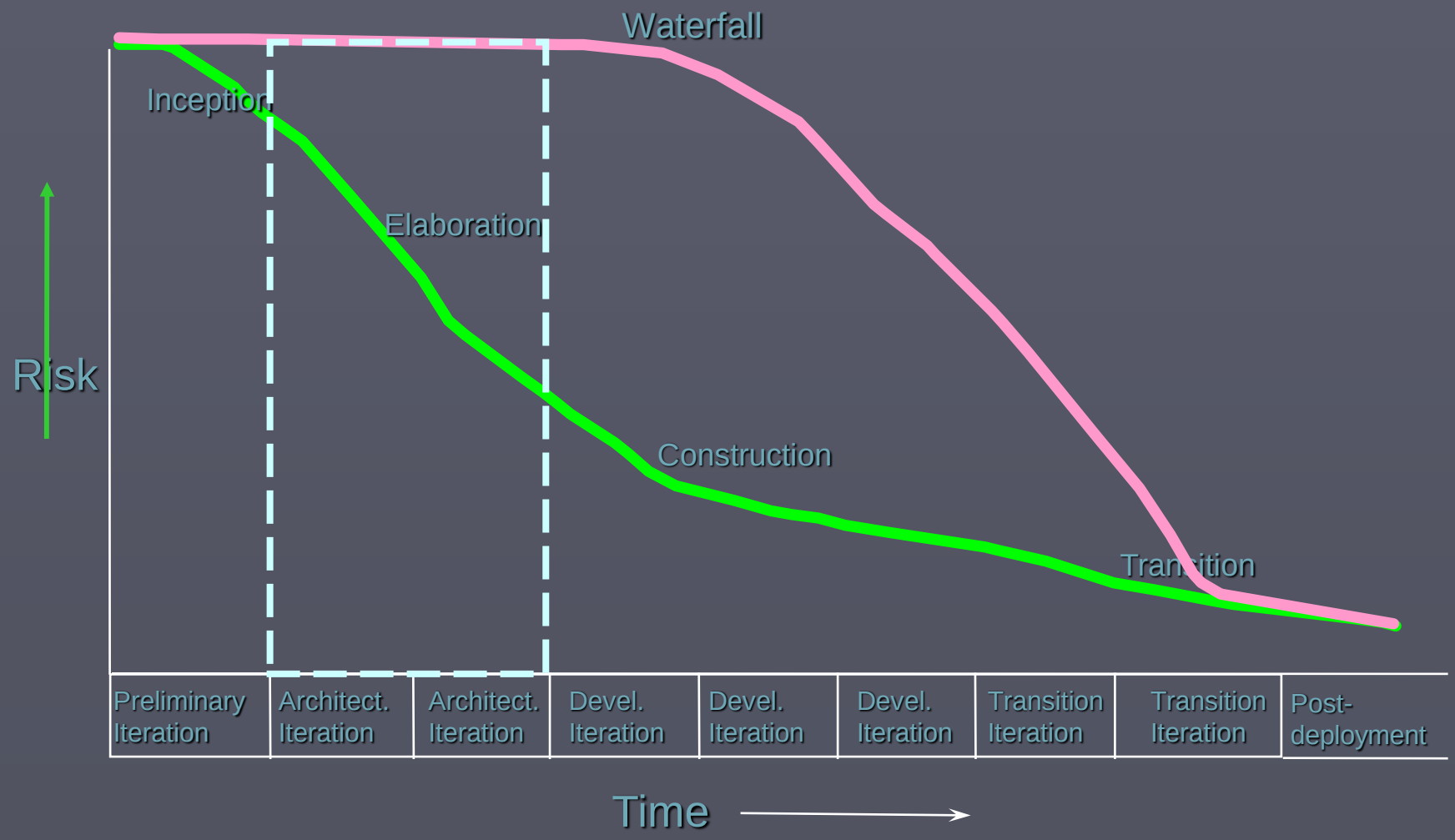
Waterfall Development



Waterfall Development: Risk vs. Time



Risk Profile of an Iterative Development

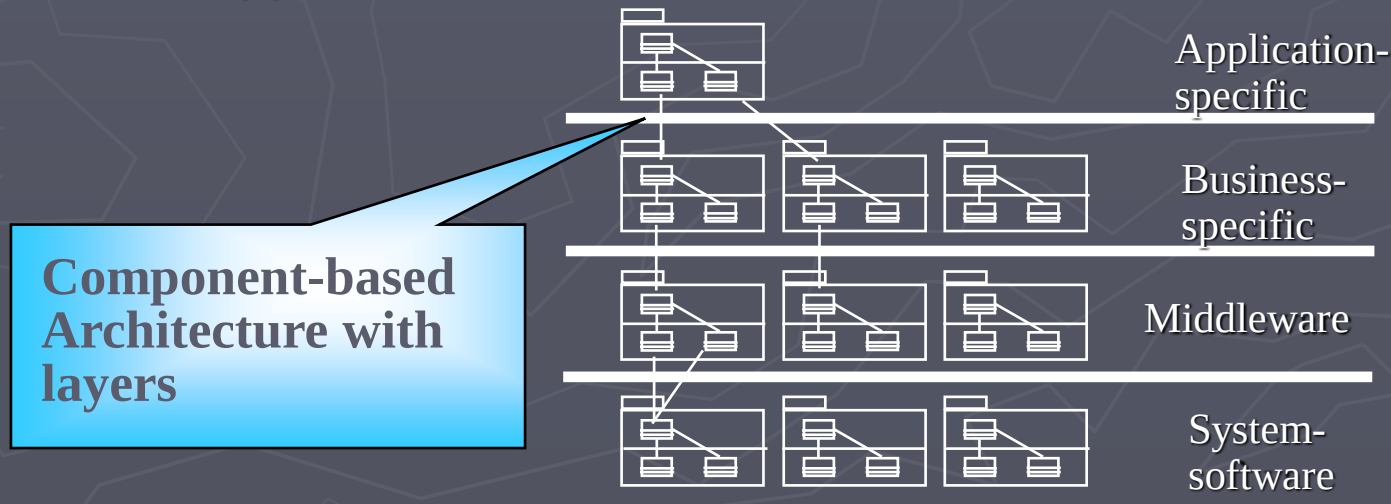


Iterative Development Characteristics

- Critical risks are resolved before making large investments
- Initial iterations enable early user feedback
- Testing and integration are continuous
- Objective milestones provide short-term focus
- Progress is measured by assessing implementations
- Partial implementations can be deployed

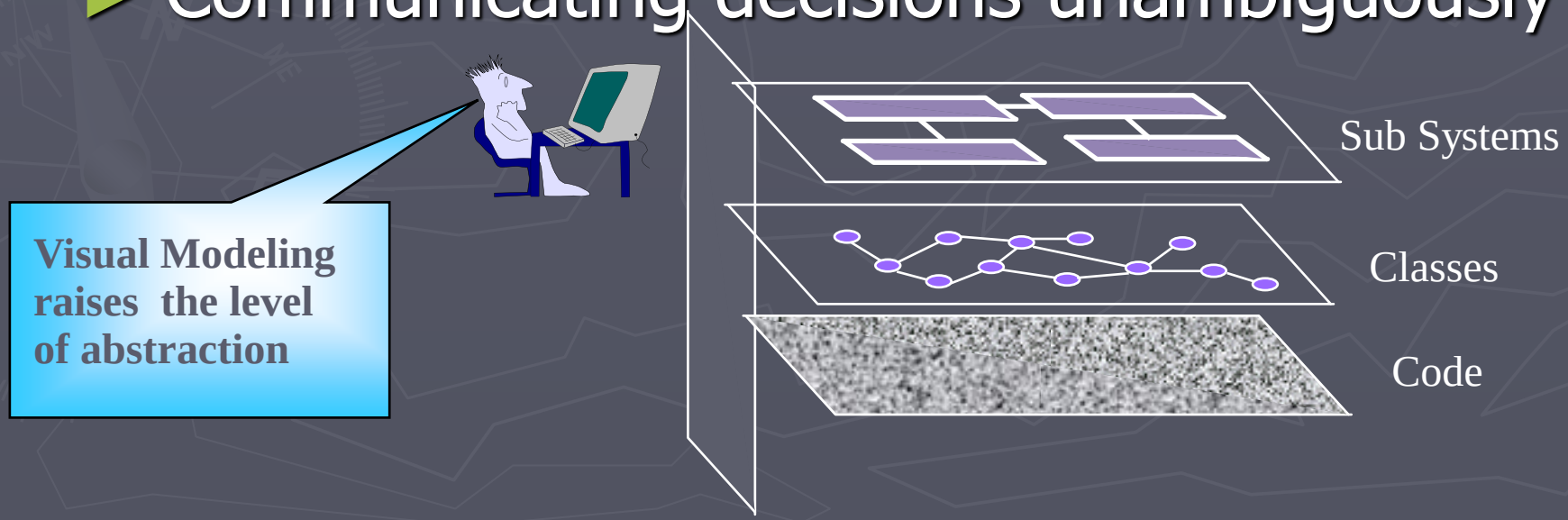
3. Employ Component-based Architecture

- Component Architectures are based on independent, replaceable, modular components, they help to manage complexity and encourage re-use.
  - The application is **built from discrete executable components** which are **developed relatively independently of one another**, potentially by different teams
  - Assembly components may be shared between applications, creating opportunities for **reuse**, but also creating **inter-project dependencies**.
  - component-based applications tend to be **distributed**.



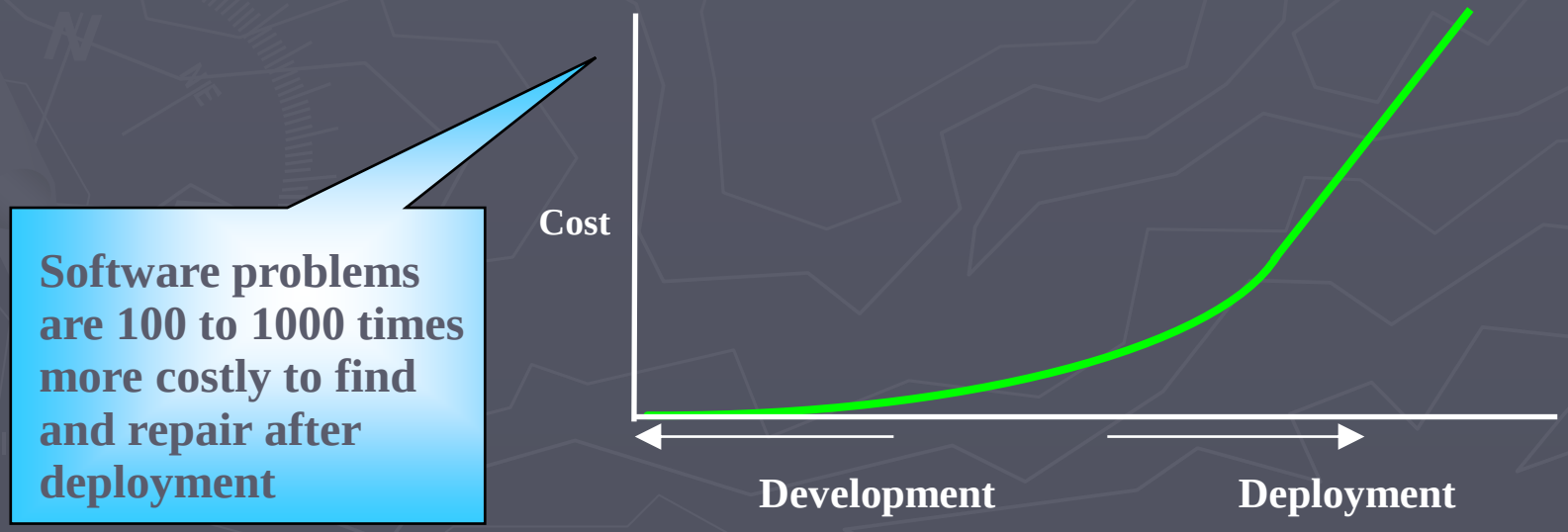
4. Model Software Visually

- Aiding understanding of complex systems
- Exploring and comparing design alternatives at a low cost
- Forming a foundation for implementation
- Capturing requirements precisely
- Communicating decisions unambiguously



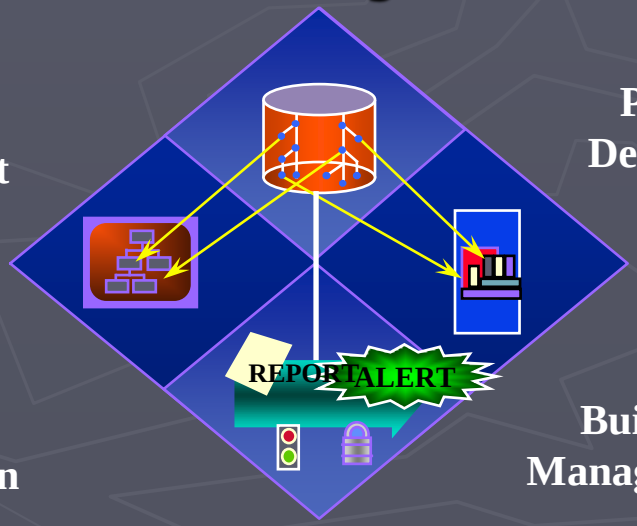
5. Verify Software Quality

- Create tests for each key scenario to ensure that all requirements are properly implemented
- Verify software performance
- Verify software reliability
- Test every iteration



6. Control Changes to Software

- Control, track and monitor changes to enable iterative development
- Establish secure workspaces for each developer
  - Provide isolation from changes made in other workspaces
  - Control all software artifacts - models, code, docs, etc.
- Automate integration and build management



CM is more than just check-in and check-out

Summary: Best Practices of Software Engineering

- The result is software that is
  - On Time
  - On Budget
  - Meets Users Needs