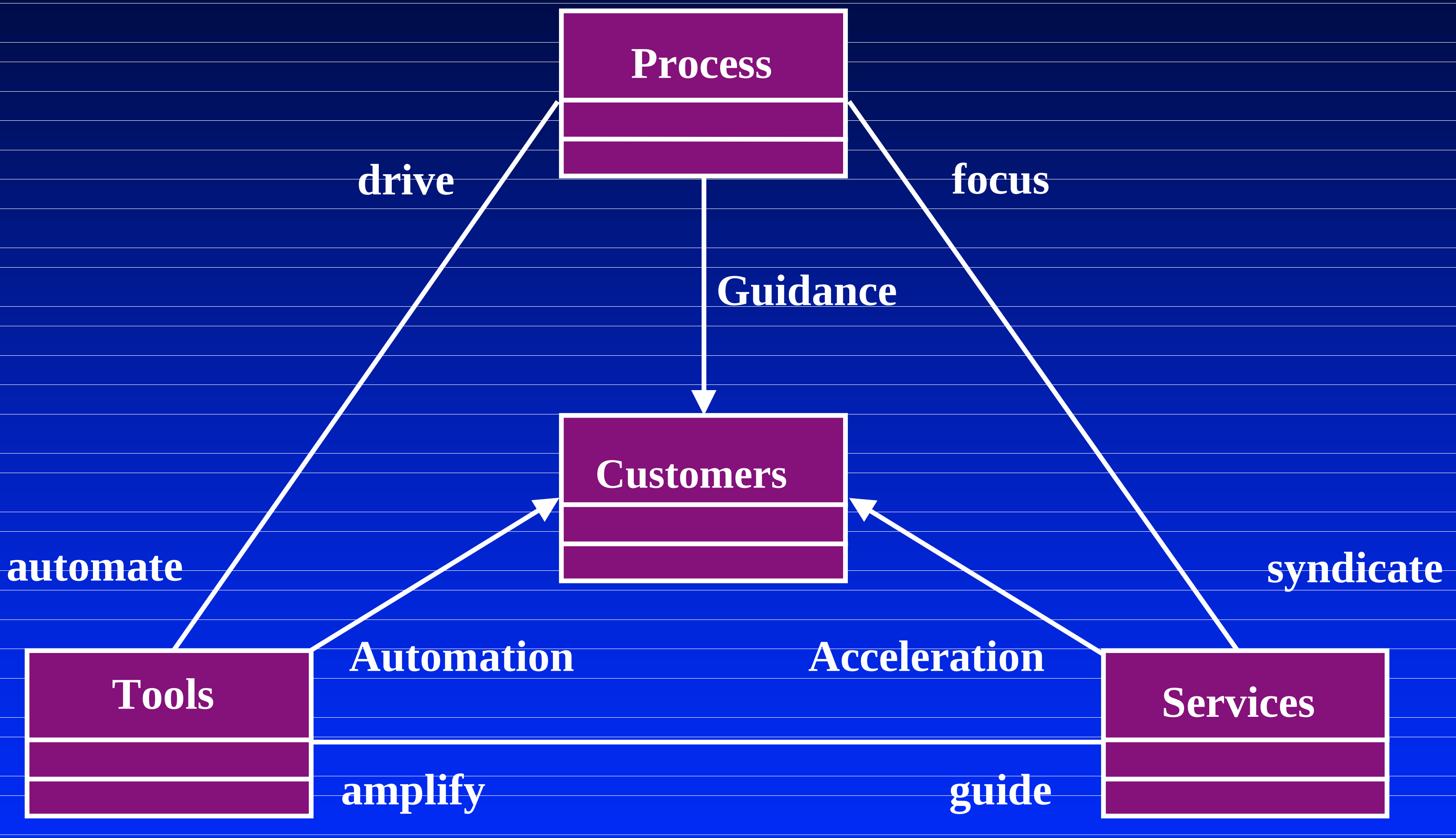
Rational Unified Process

**Best Practices for
Software Development Teams**

Agenda

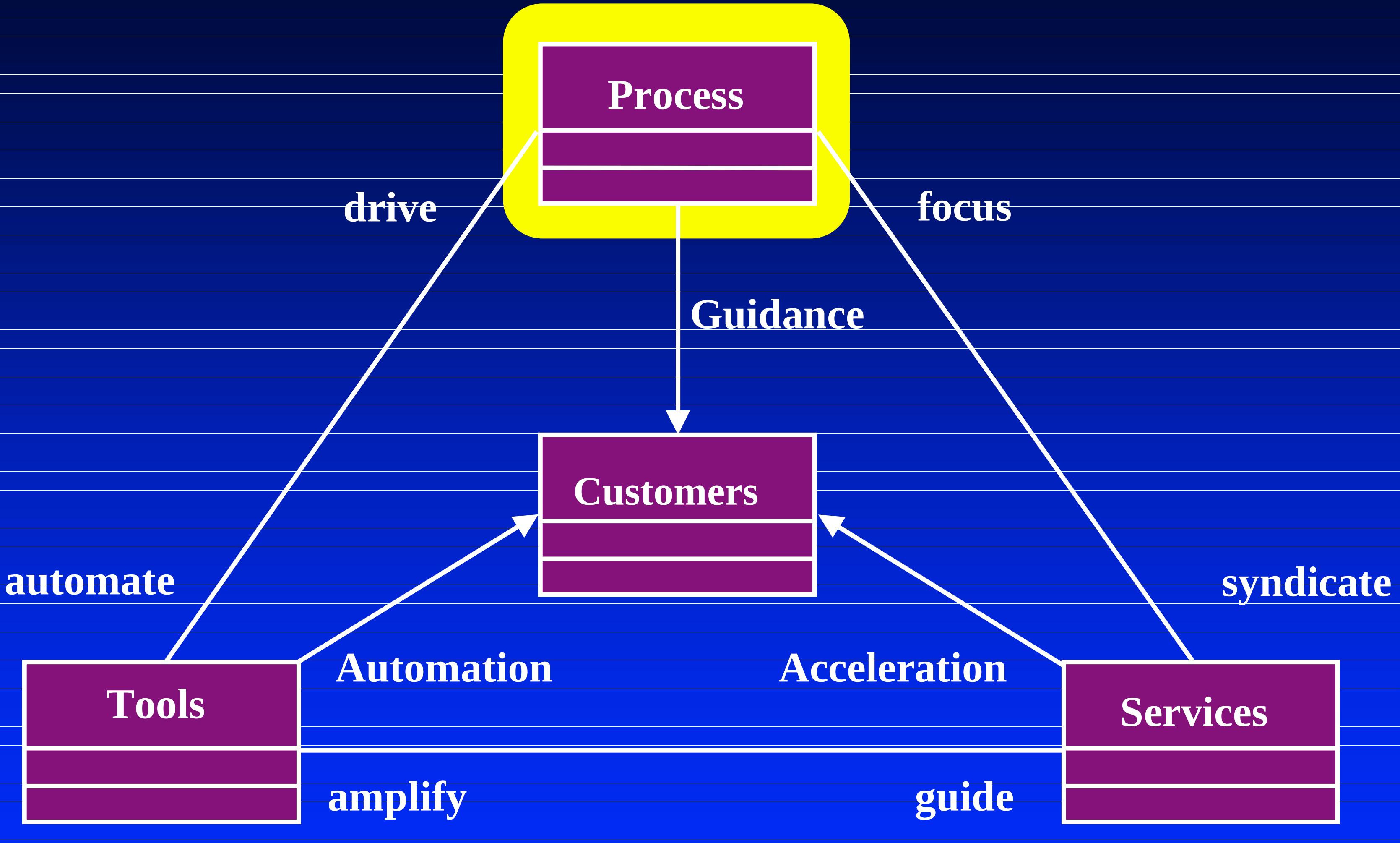
- ➔ What is the Rational Unified Process
- ◆ Implementing Best Practices
- ◆ Phases, Iterations, Workflows and Activities
- ◆ The Product
- ◆ Implementing the Rational Unified Process
- ◆ Software Economics
- ◆ Case Study: Skandia IT
- ◆ Summary

Rational's Strategy



R

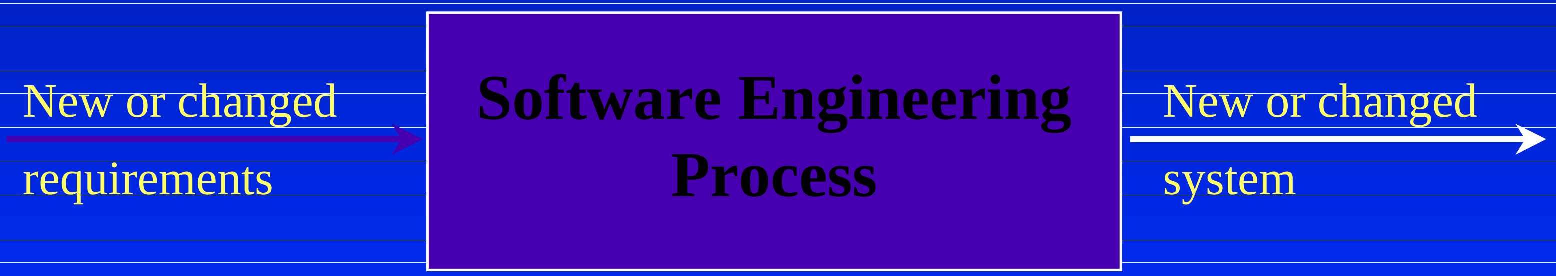
Rational's Strategy



R

What Is a Process?

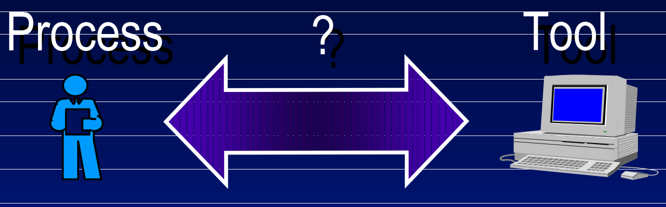
A process defines **Who** is doing **What**, **When** and **How** to reach a certain goal. In software engineering the goal is to build a software product or to enhance an existing one



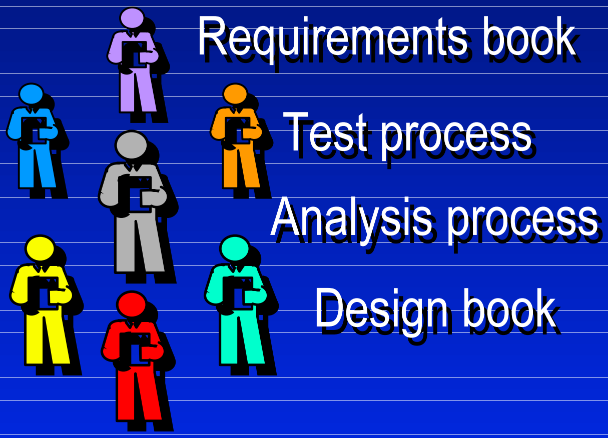
R

The Problem...

- Processes are not linked properly to tools, or are not properly automated



- If process is used, different functional teams use normally inconsistent processes and modeling languages



- Most software projects use no well-defined process. Instead team members (re-)invent process as they go

R

Problems addressed by RUP

- No repeatable process - results are lacking, unpredictable, and highly dependent on heroic programmers
- Software that poorly fits user needs
- Inability to deal with changing requirements
- Tedious and expensive testing procedures
- Discovery of serious flaws too late in the project
- Software that's hard to maintain and extend

Rational Unified Process (RUP)

- Unifies best practices from many disciplines into a consistent full lifecycle process
- Premier process for the UML, developed by the company that brought you the UML
- Online mentor integrated with and supported by Rational tools
- Applicable to a wide variety of applications and industries

Decrease Time to Market

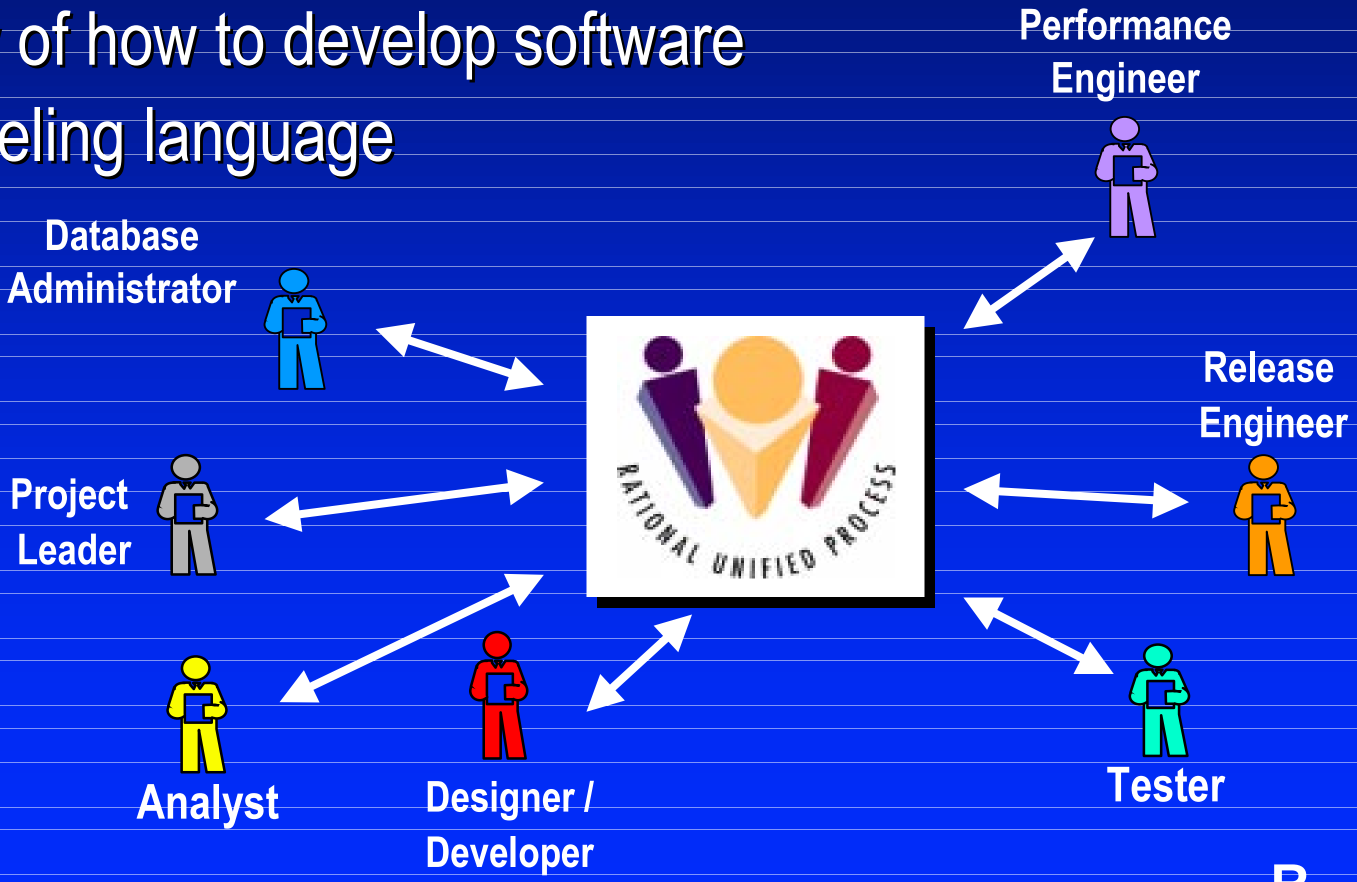
Increase Predictability

R

Increased Team Productivity

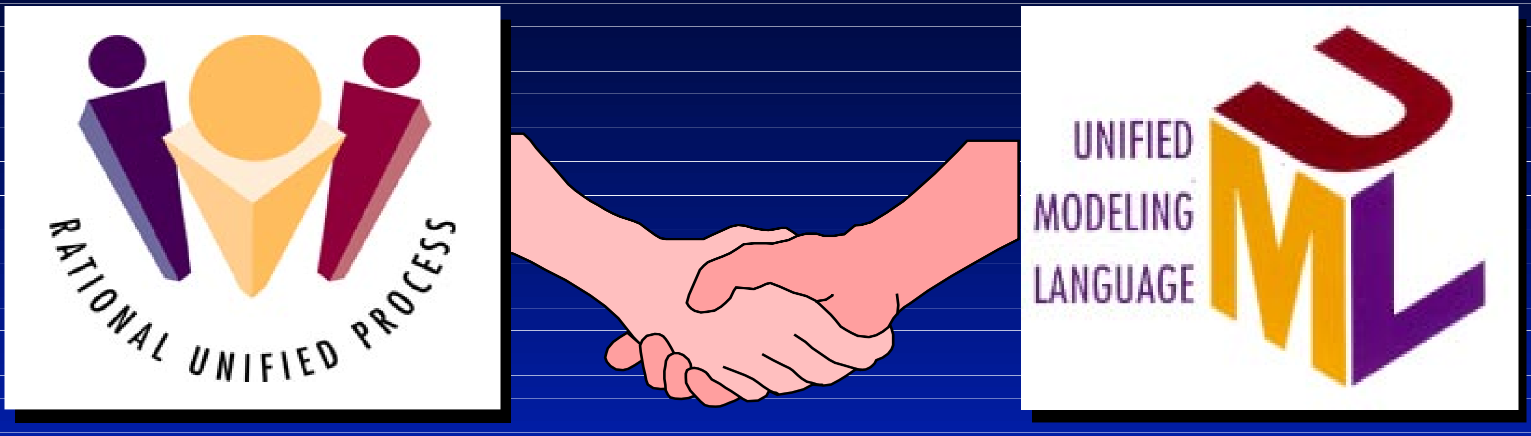
All team members share

- 1 knowledge base
- 1 process
- 1 view of how to develop software
- 1 modeling language



R

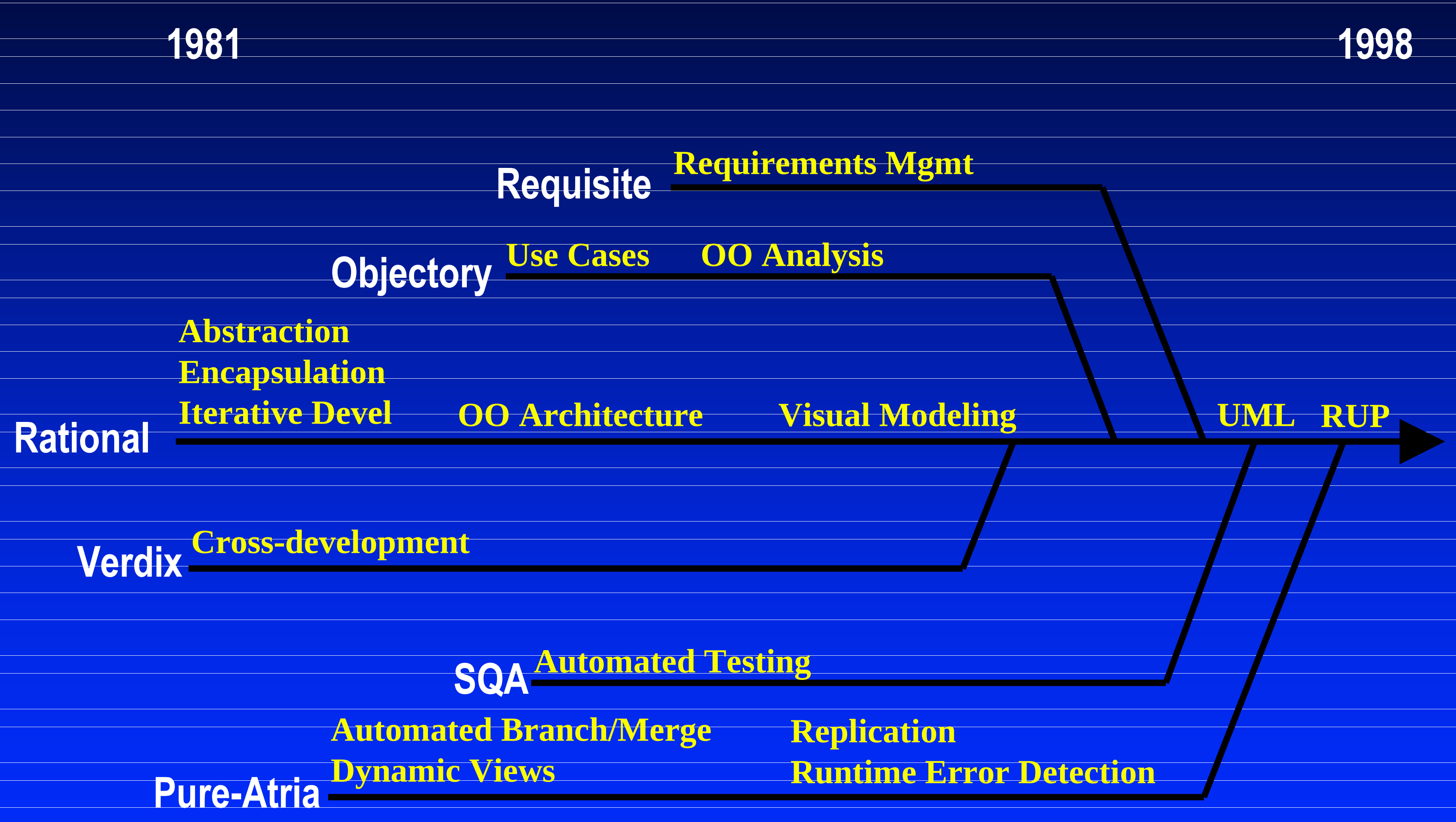
The Unified Modeling Language (UML)



- The Rational Unified Process and the UML — developed hand-in-hand — by Rational
- Contributions by major vendors
  - Microsoft
  - HP
  - IBM
  - Oracle
  - Texas Instruments
  - MCI SystemHouse
- Standard through OMG

R

Rational Unified Process: Geneology



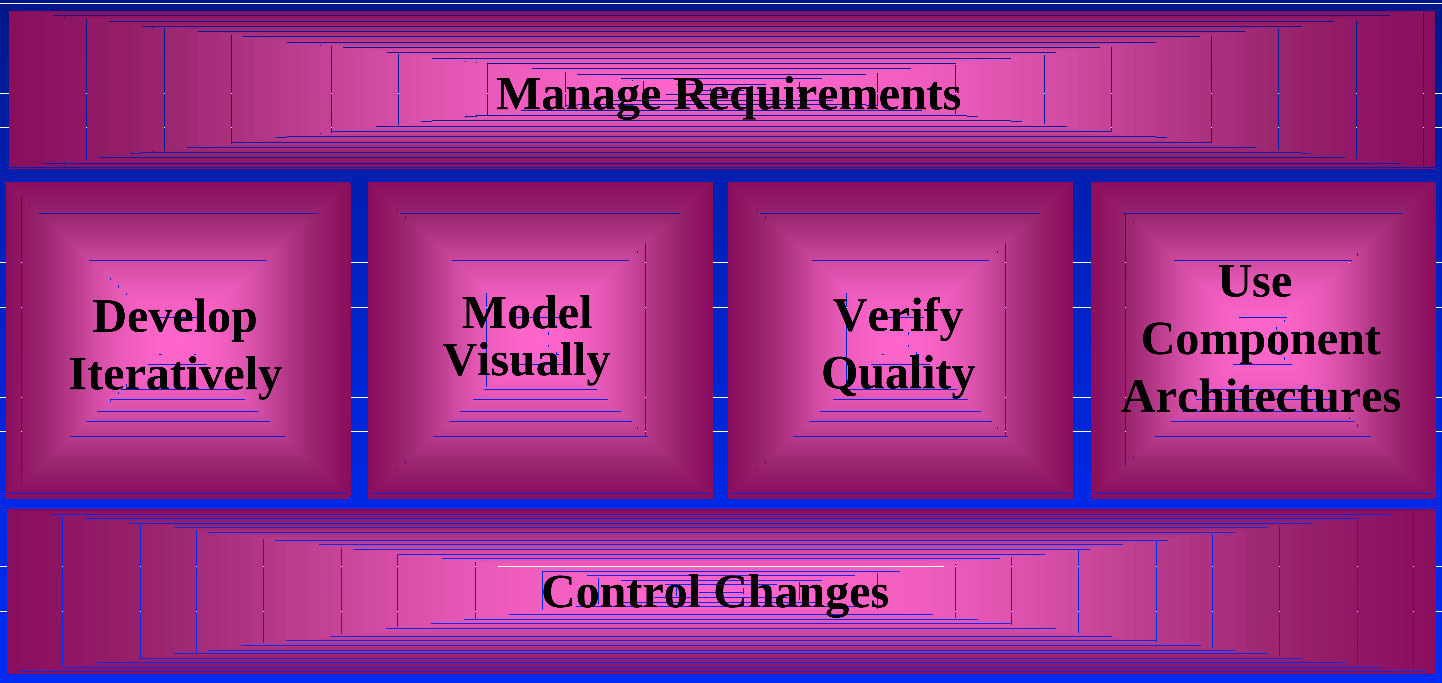
R

Agenda

- What is the Rational Unified Process
- ➡ Implementing Best Practices
- Phases, Iterations, Workflows and Activities
- The Product
- Implementing the Rational Unified Process
- Software Economics
- Case Study: Skandia IT
- Summary

Rational Unified Process

Describes the effective implementation of key “Best Practices”



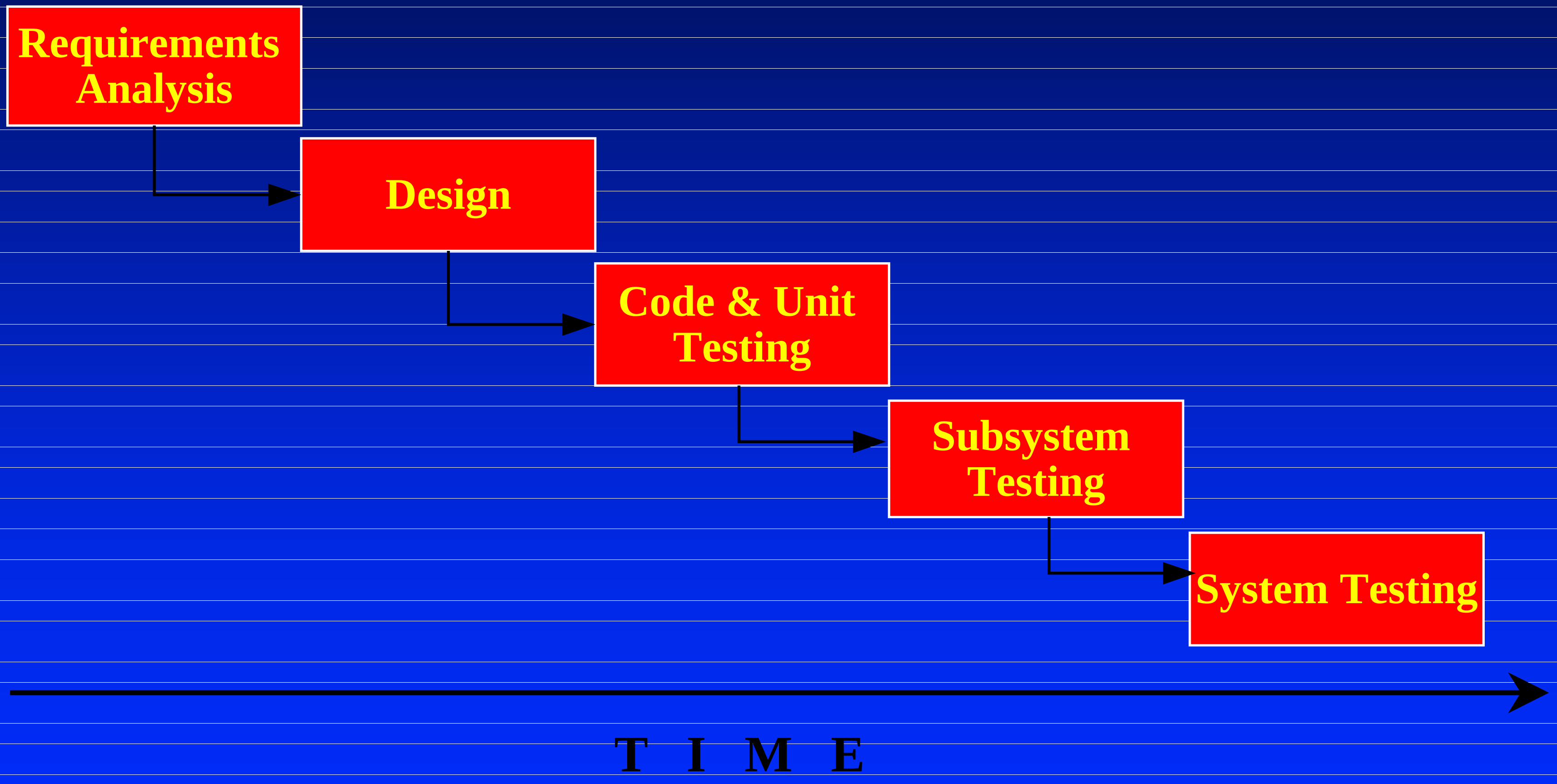
1. Develop Software Iteratively

- 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

The time and money spent implementing a faulty design are not recoverable

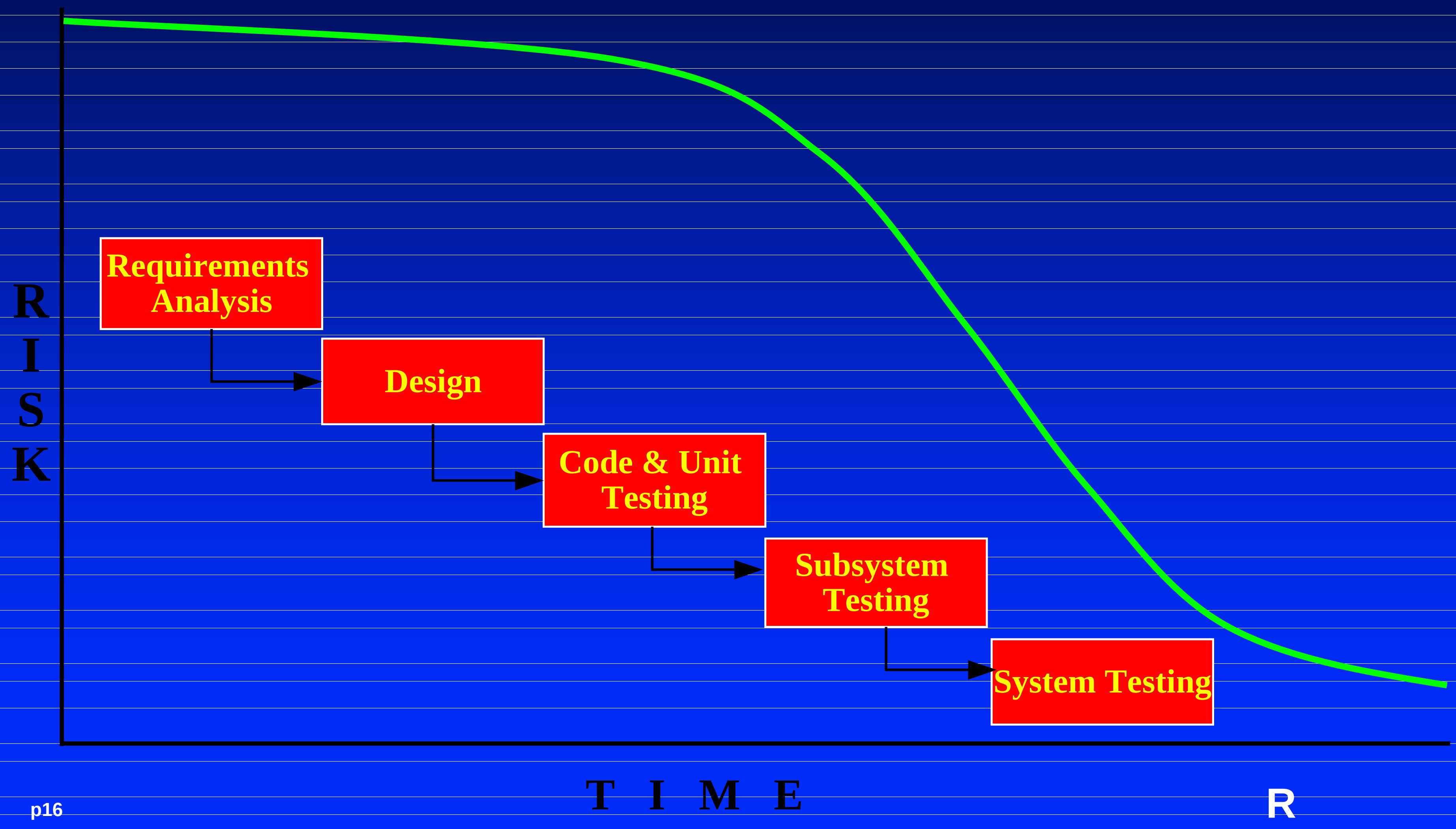
R

Waterfall Development



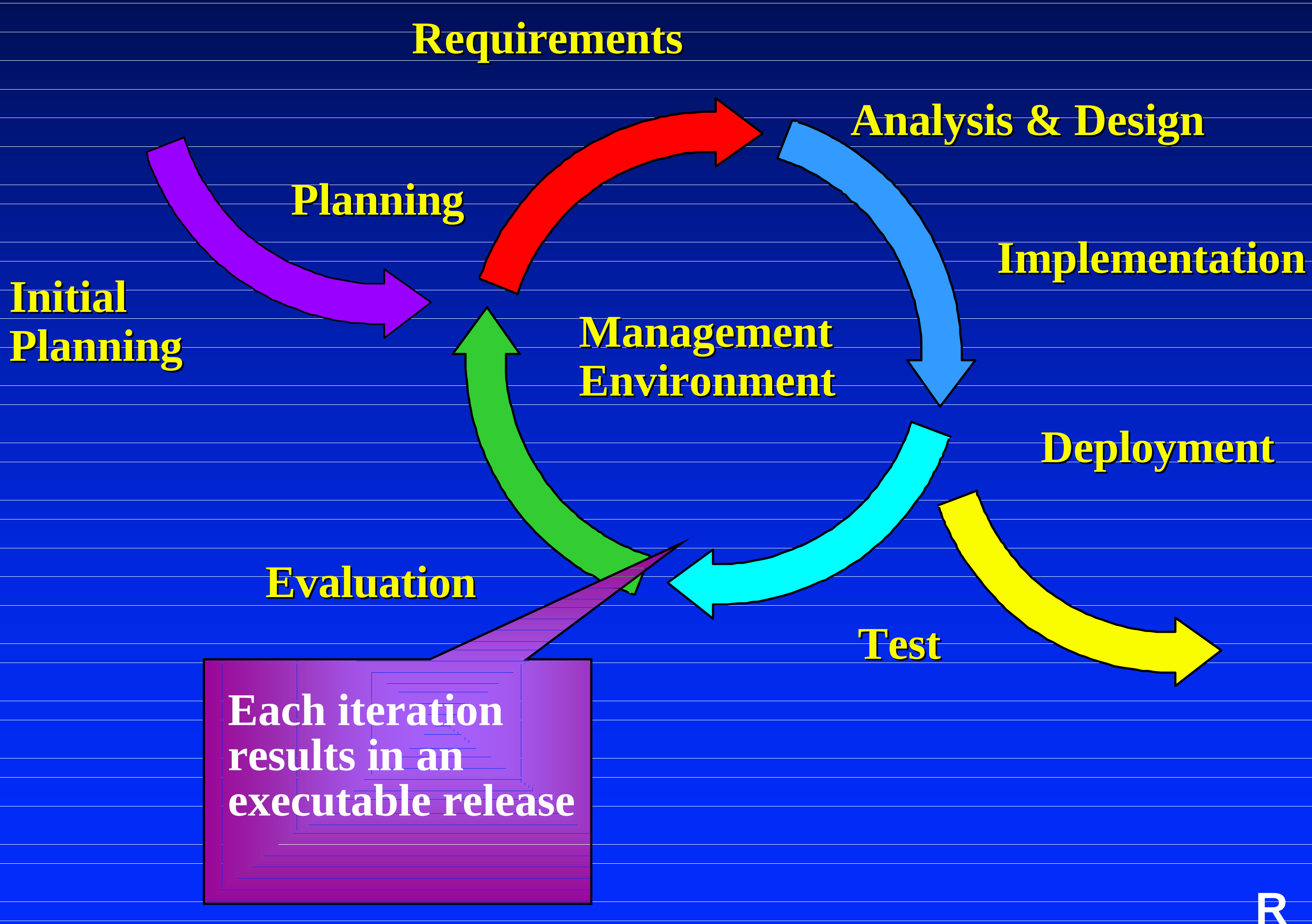
R

Waterfall Development: Risk vs. Time



R

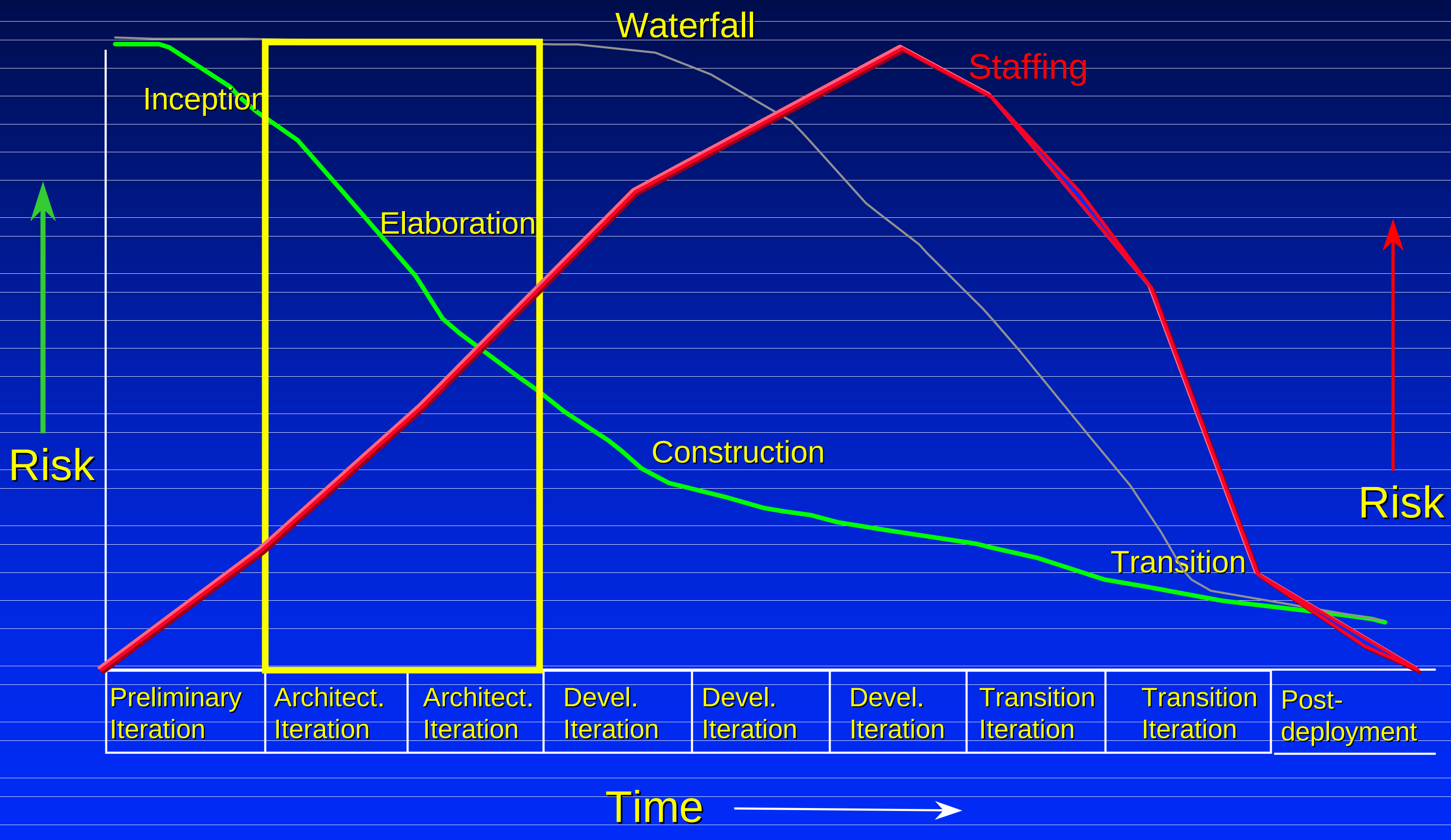
Iterative Development



Each iteration results in an executable release

R

Risk Profile of an Iterative Development



R

Iterative Development Characteristics

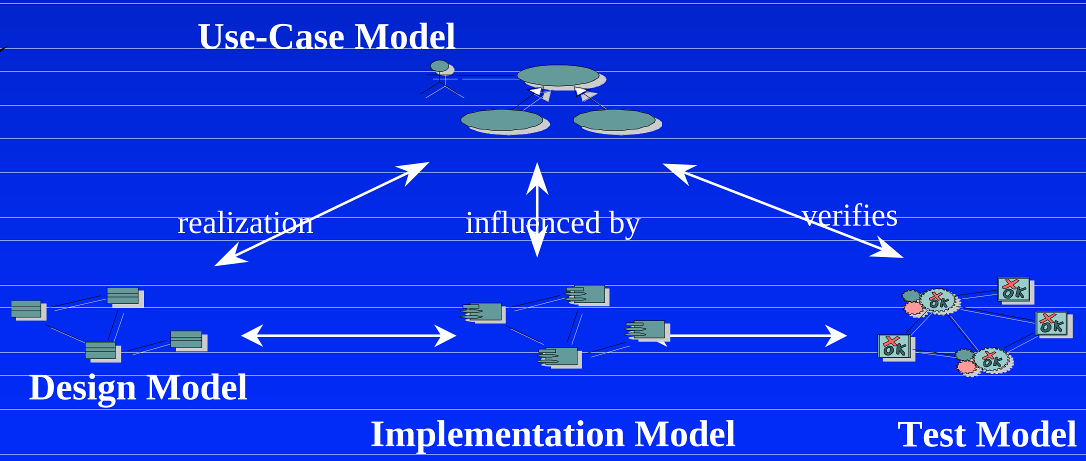
- Critical risks are resolved before making large investments
- Initial iterations enable early user feedback
- Testing and integration are continuous
- Objective milestones provide short-term focus
- Progress is measured by assessing implementations
- Partial implementations can be deployed

R

2. 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

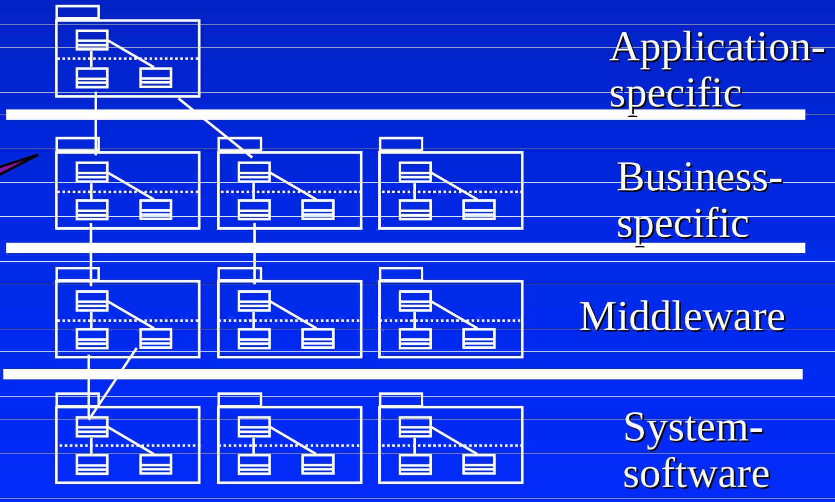
Use Cases drives the work from analysis through test



R

3. Employ Component-based Architecture

- Design, implement and test your architecture up-front!
- A systematic approach to define a “good” architecture
  - resilient to change by using well-defined interfaces
  - by using and reverse engineering components
  - derived from top rank use cases
  - intuitively understandable



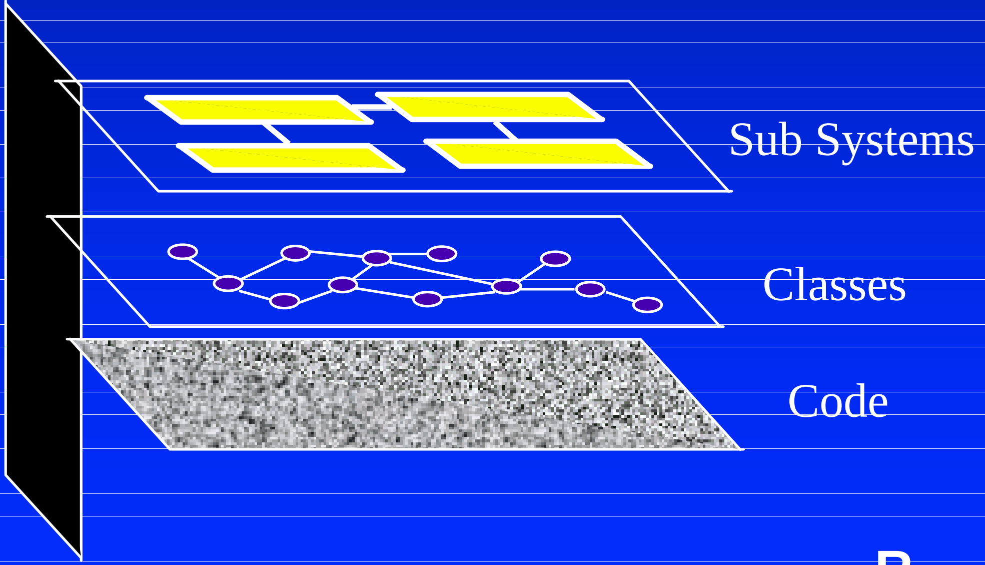
Component-based Architecture with layers

R

4. Model Software Visually

- Capture the structure and behavior of architectures and components
- Show how the elements of the system fit together
- Maintain consistency between a design and its implementation
- Promote unambiguous communication

Visual Modeling raises the level of abstraction

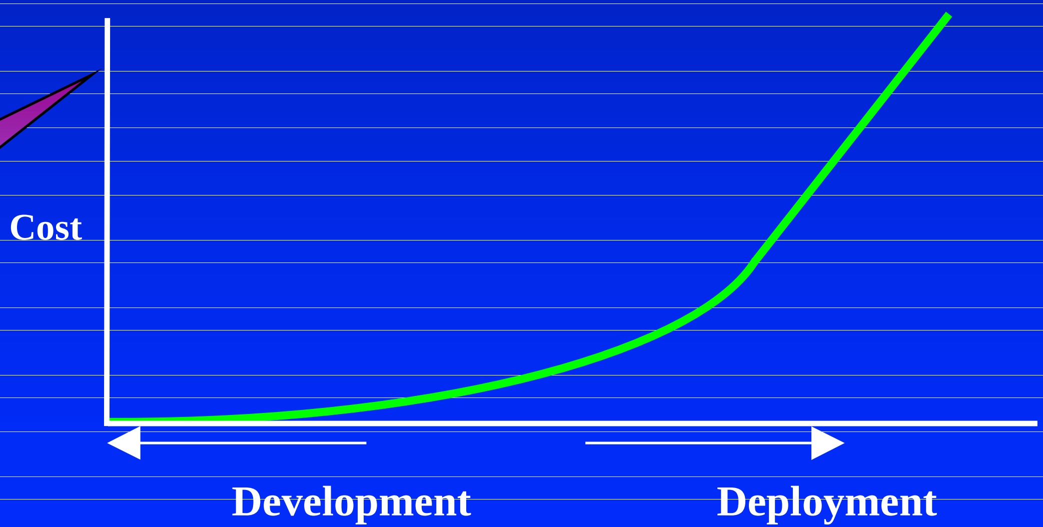


R

5. Verify Software Quality

- Create tests for each key scenario to ensure that all requirements are properly implemented
- Unacceptable application performance hurts as much as unacceptable reliability
- Verify software reliability - memory leaks, bottle necks
- Test every iteration - automate test!

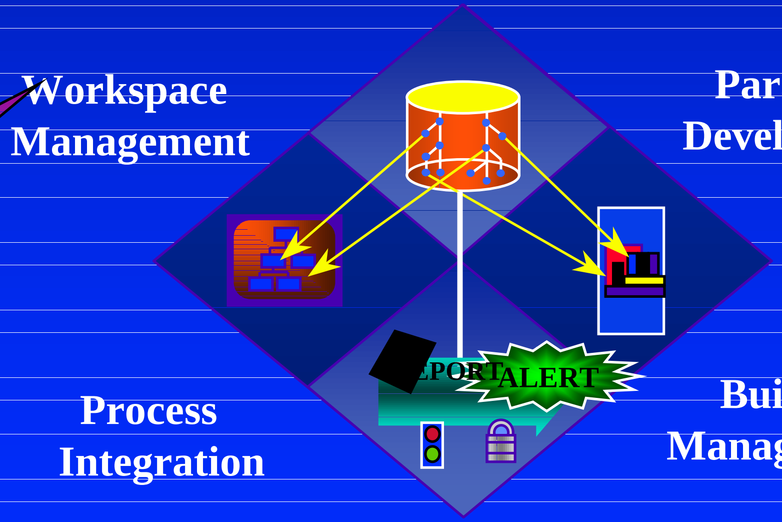
Software problems are 100 to 1000 times more costly to find and repair after deployment



R

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



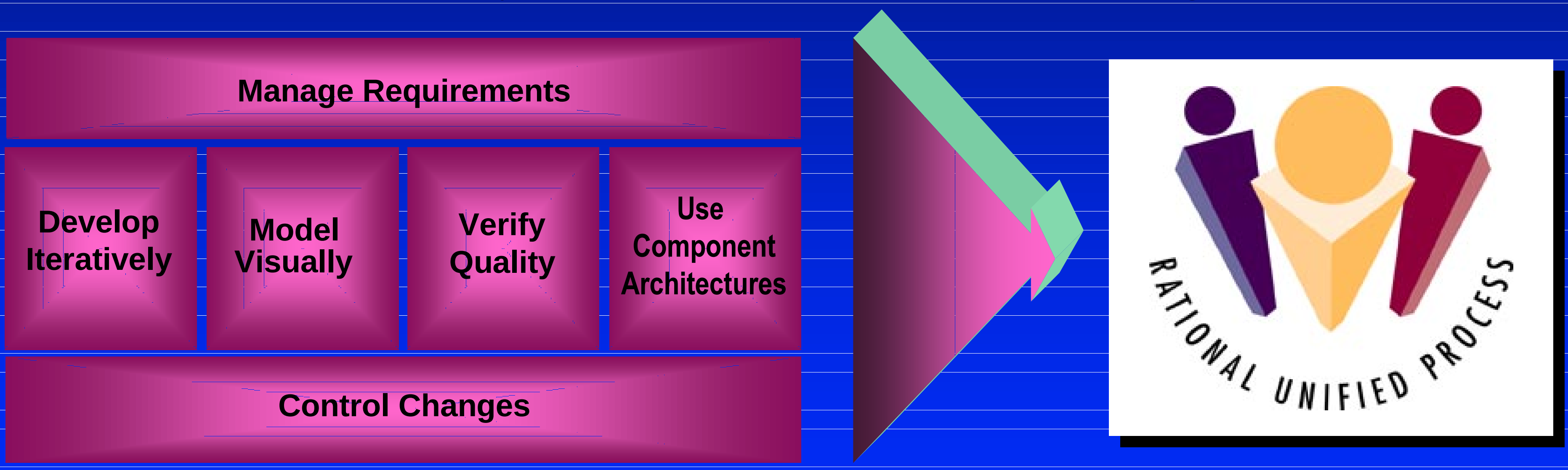
CM is more than just check-in and check-out

R

Rational Unified Process

Provides guidelines, templates and tool mentors for the effective implementation of key best practices

Delivered through a web-enabled searchable knowledge base



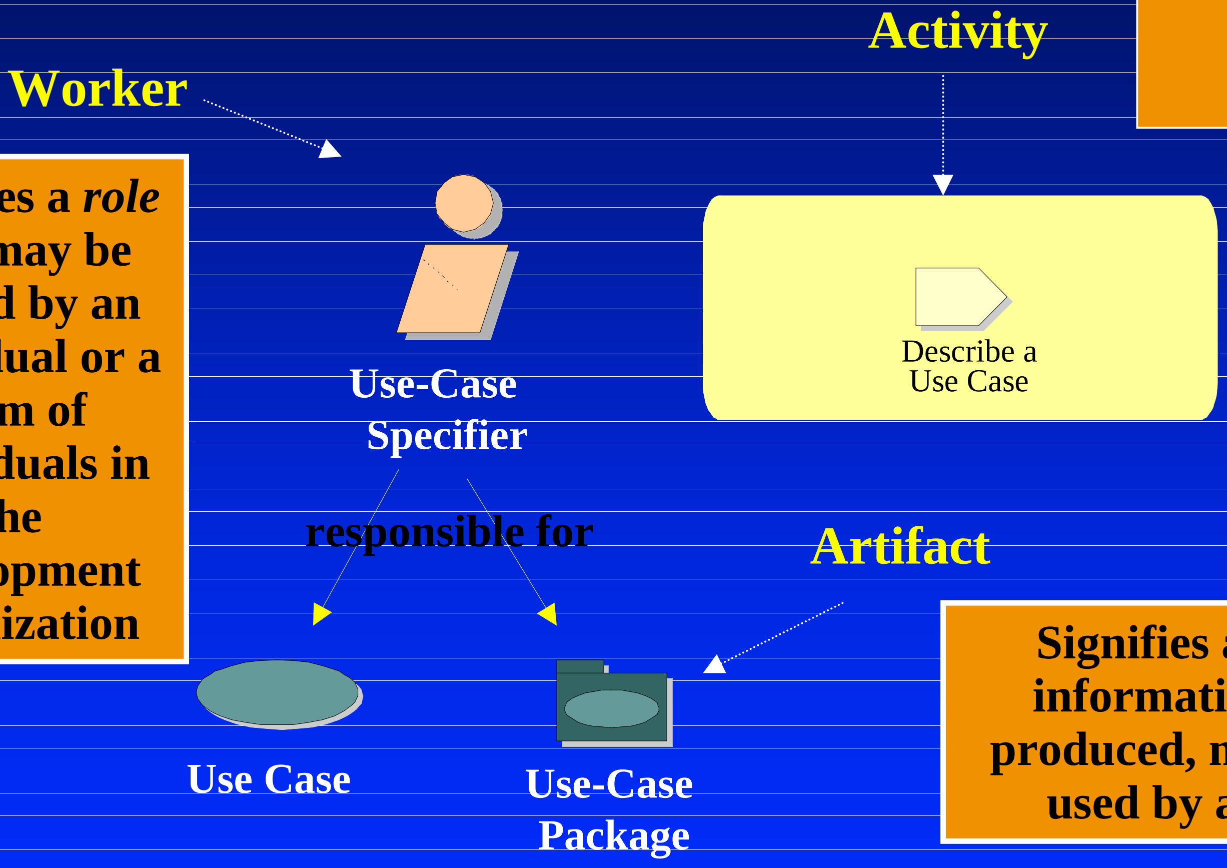
R

Agenda

R

Process Notation

Signifies a *role* that may be played by an individual or a team of individuals in the development organization

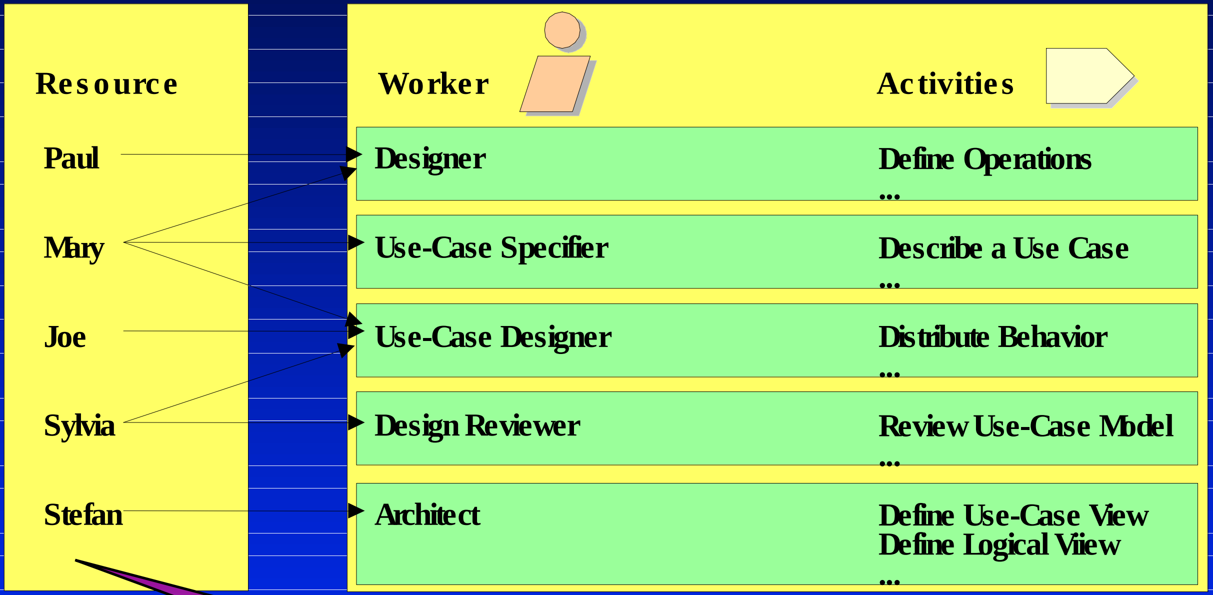


Describes a piece of work a worker may be asked to perform.

Signifies a piece of information that is produced, modified, or used by a process

R

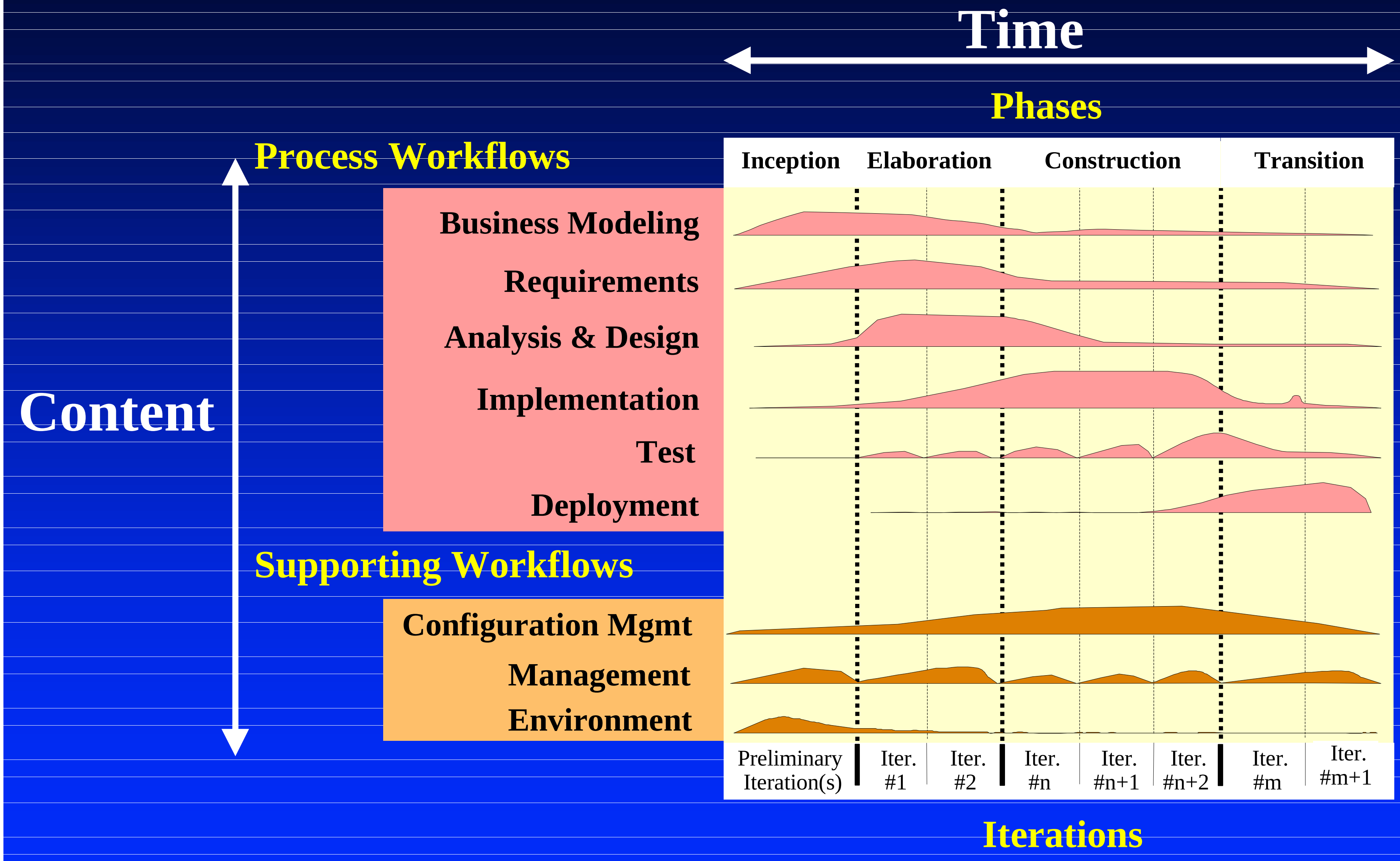
Workers Are Used for Resource Planning



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

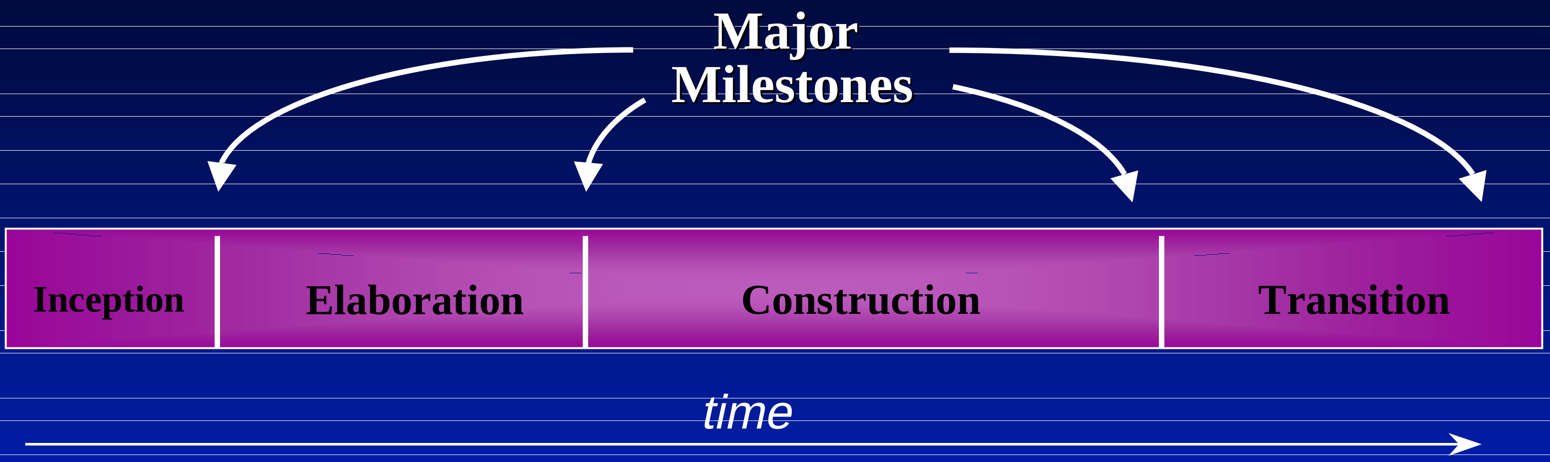
R

Process Architecture



R

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

R

Inception Phase

- Purpose
  - To establish the business case for a new system or for a major update of an existing system
  - To specify the project scope
- Outcome
  - A general vision of the project's requirements, i.e., the core requirements
    - Initial use-case model and domain model (10-20% complete)
  - An initial business case, including:
    - Success criteria (e.g., revenue projection)
    - An initial risk assessment
    - An estimate of resources required

R

Elaboration Phase

- **Purpose**
  - To analyze the problem domain
  - To establish a sound architectural foundation
  - To address the highest risk elements of the project
  - To develop a comprehensive plan showing how the project will be completed
- **Outcome**
  - Use-case and domain model 80% complete
  - An executable architecture and accompanying documentation
  - A revised business case, incl. revised risk assessment
  - A development plan for the overall project

R

Construction Phase

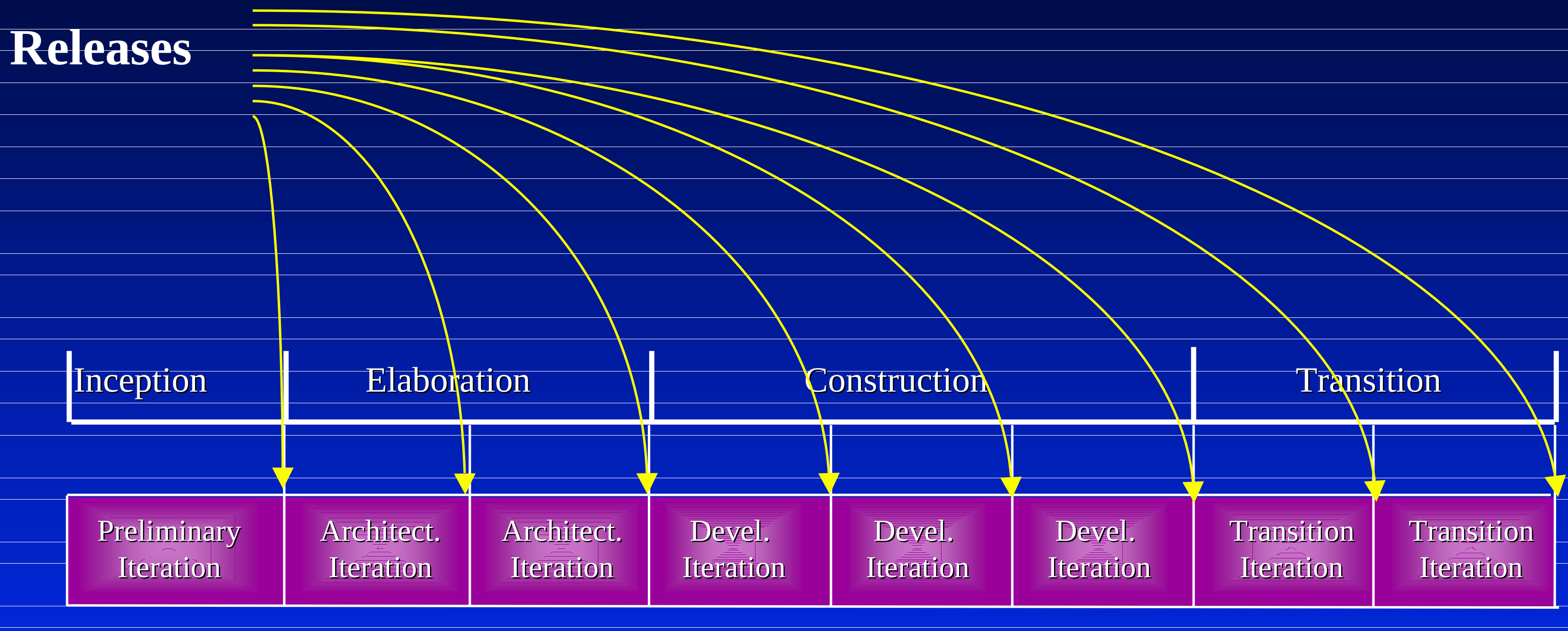
- **Purpose**
  - To incrementally develop a complete software product which is ready to transition into the user community
- **Products**
  - A complete use-case and design model
  - Executable releases of increasing functionality
  - User documentation
  - Deployment documentation
  - Evaluation criteria for each iteration
  - Release descriptions, including quality assurance results
  - Updated development plan

Transition Phase

- Purpose
  - To transition the software product into the user community
- Products
  - Executable releases
  - Updated system models
  - Evaluation criteria for each iteration
  - Release descriptions, including quality assurance results
  - Updated user manuals
  - Updated deployment documentation
  - “Post-mortem” analysis of project performance

R

Iterations and Phases



An *iteration* is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release (internal or external).

R

Iteration Assessment

Iteration N Cost and Schedule Actuals

Quality Assessment for Iteration N

- Test Results
- Defect Density
- Architecture Stability
- Other metrics

Iteration N Assessment

- Compare iteration actual cost, schedule, and content with iteration plan
- Determine rework (if any) to be done
  - Assign to future iteration(s)
- Determine what risks have been eliminated, reduced, or newly identified in this iteration
- Update project plan
- Prepare detailed plan for next iteration
  - Use revised risk list and select appropriate scenarios

Revised Risk List

Revised Project Plan

- Total Cost
- Overall Schedule
- Scope/Content

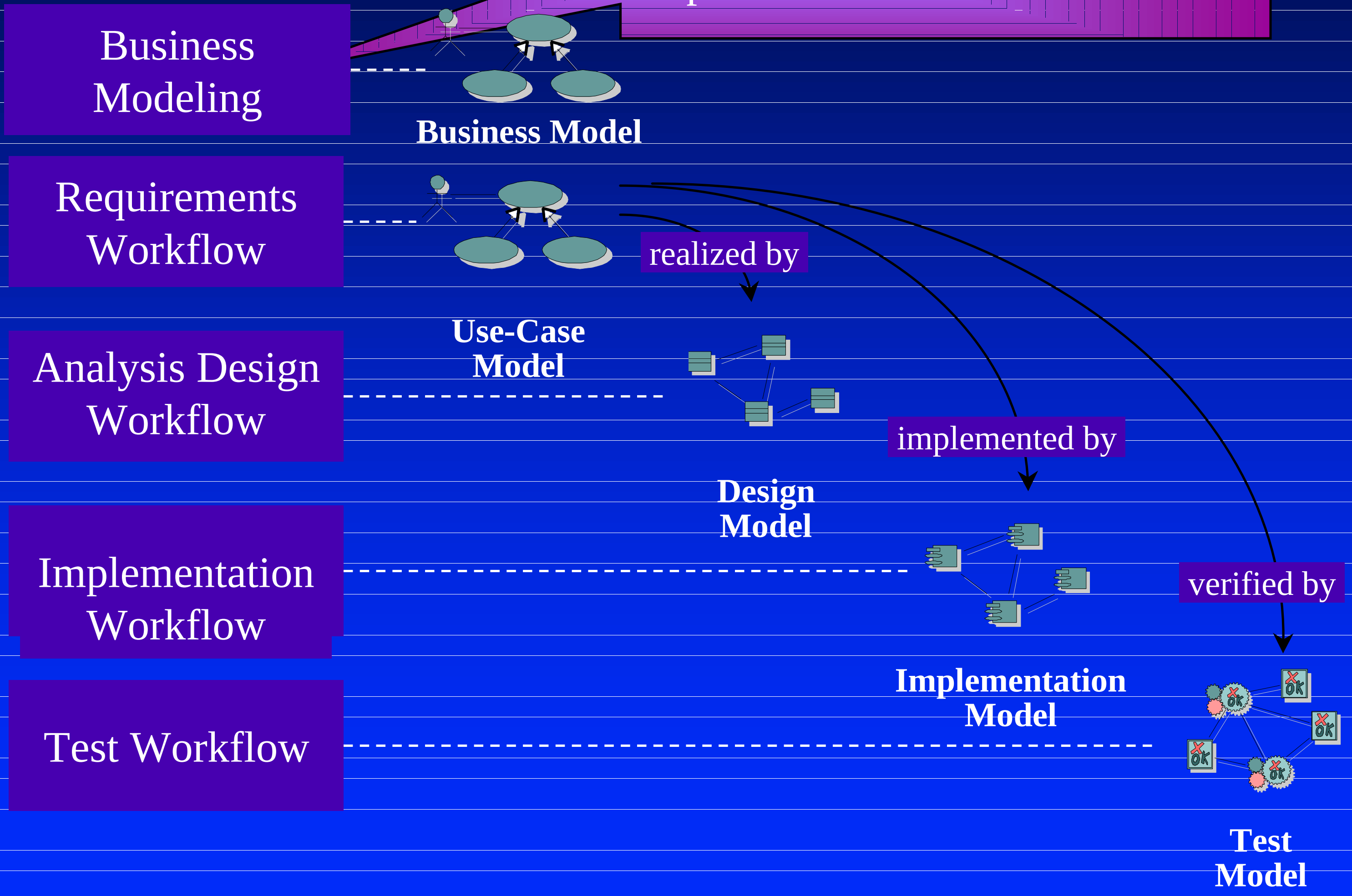
Iteration N+1 Plan

- Cost
- Schedule
- Content

R

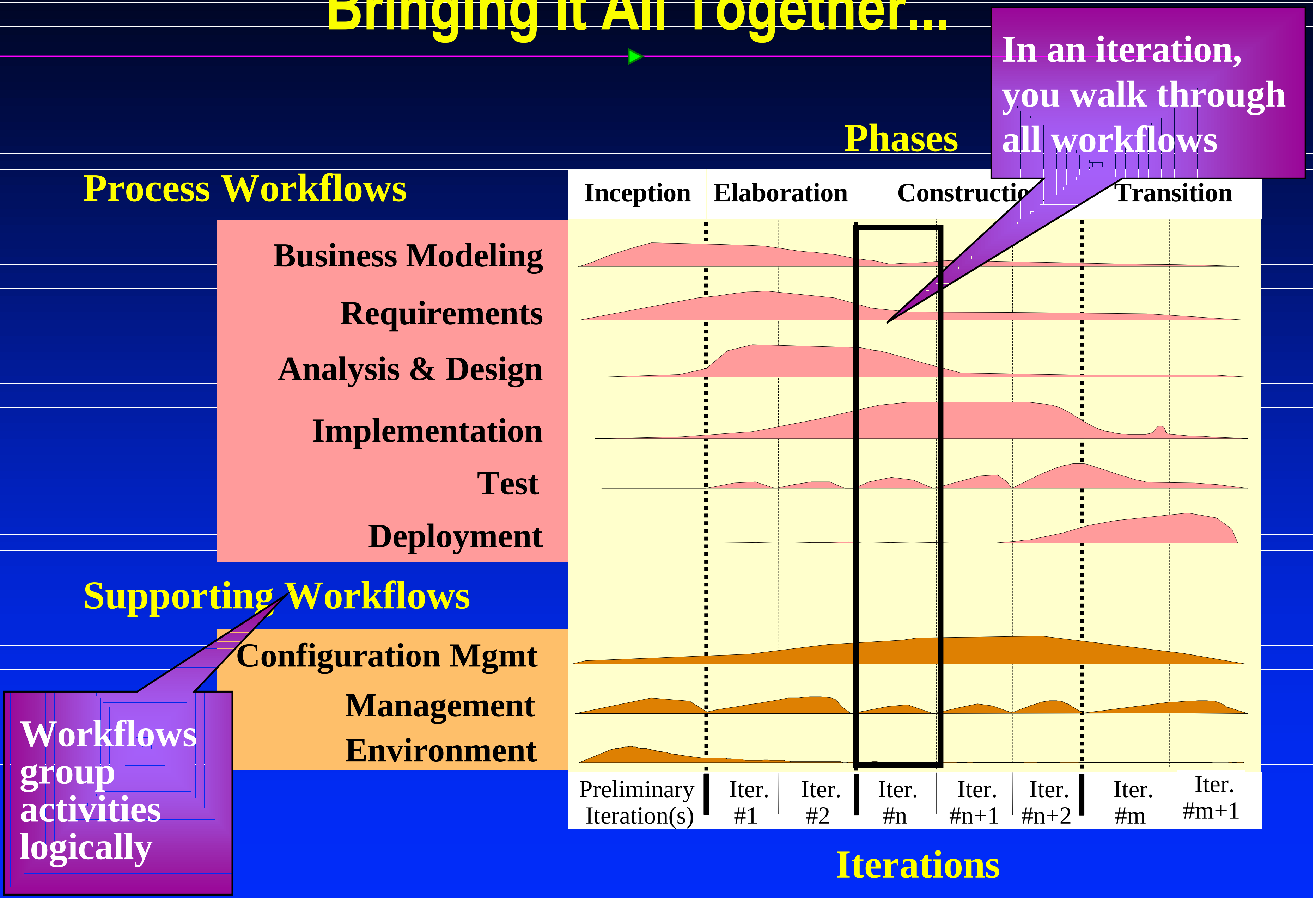
Models and Workflows

Each major workflow describes how to create and maintain a particular model



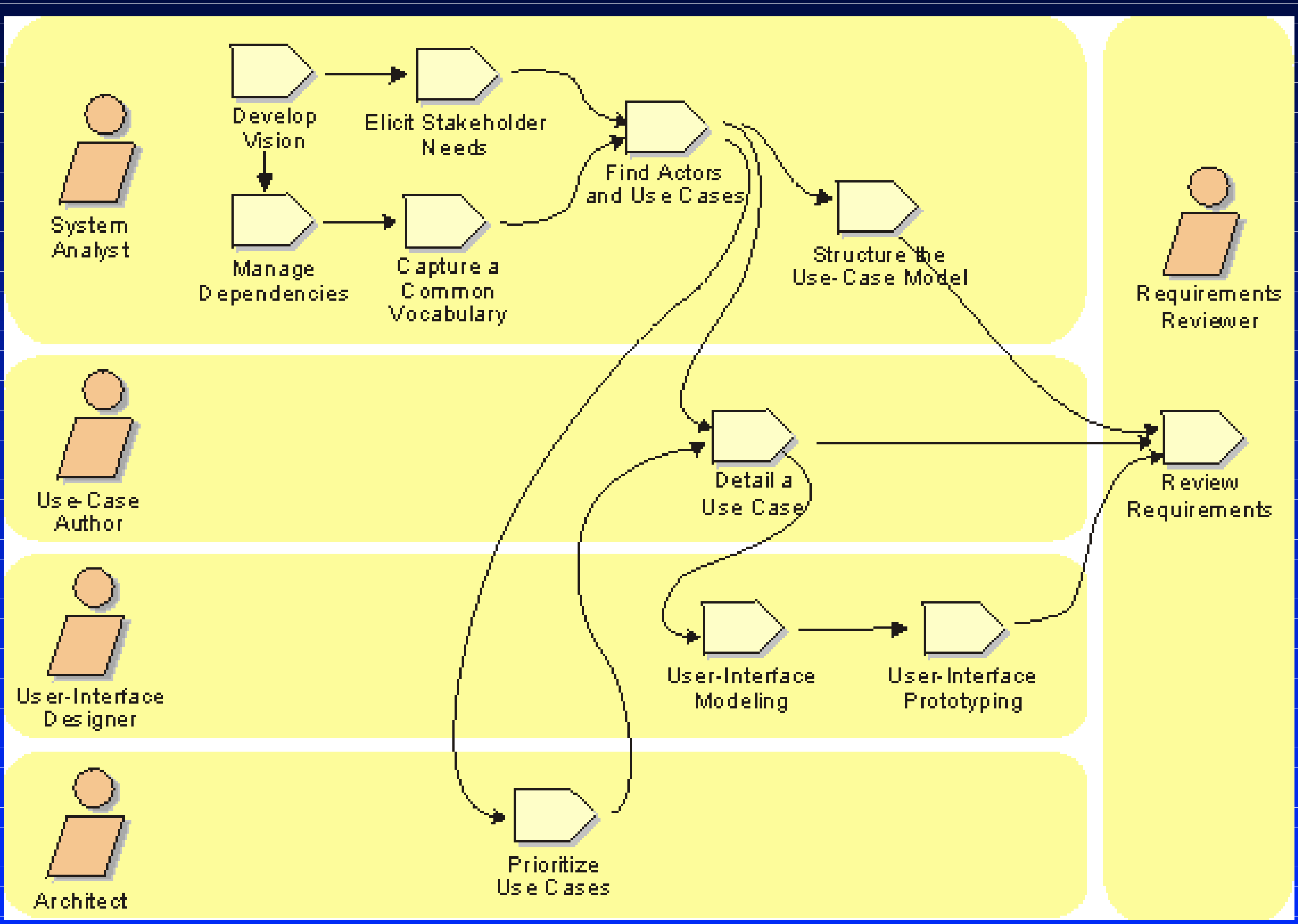
R

Bringing It All Together...



R

Example of a Workflow



R

Agenda

R

Process Delivery In the Past….

- Thick binder on every developers shelf
- ….collecting dust…
- hard to understand, hard to use, seen as driving overhead

not used

R

Rational Unified Process: Web-enabled

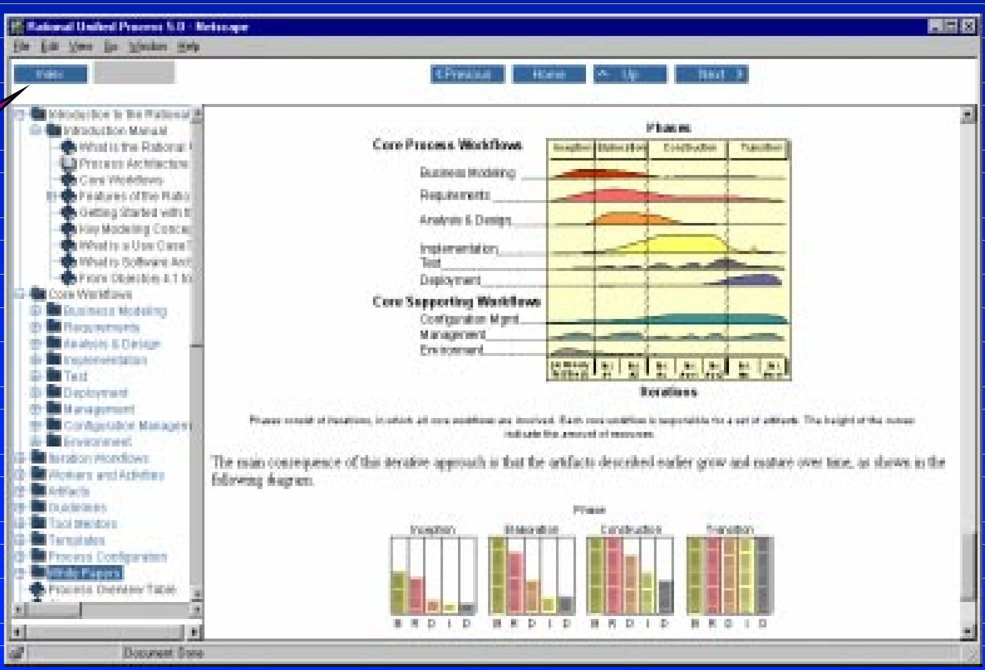
- Interactive knowledge base accessible from tools
- Powerful graphical navigation, search engine, index...
- Guidelines, templates, tool mentors at your finger tips

Searchable

Accessible

Navigable

Easy to use



The main consequence of this iterative approach is that the artifacts described earlier grow and mature over time, as shown in the following diagram.

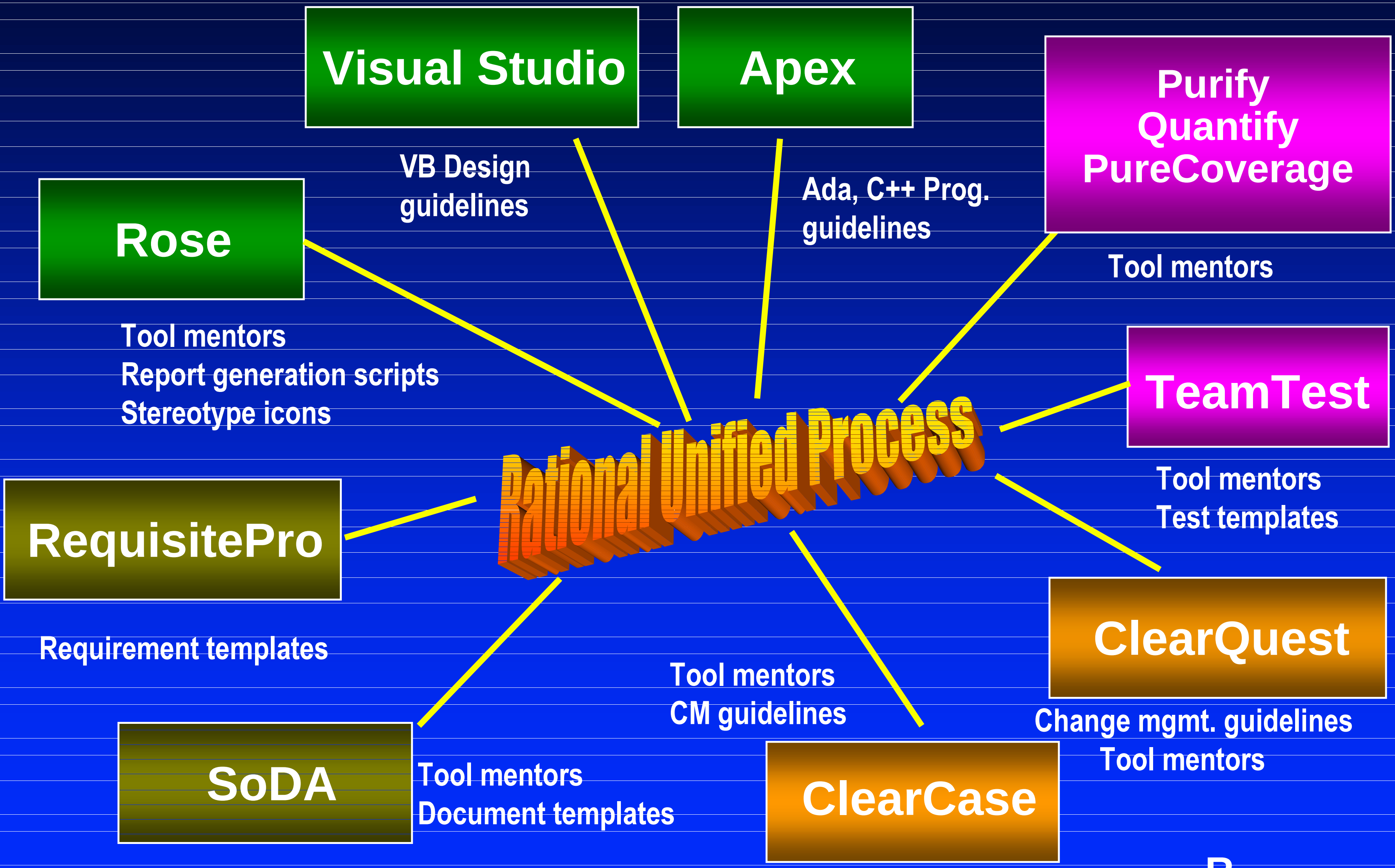
R

Knowledge Base Content

- Extensive guidelines for all team members
- Tool mentors (most Rational tools)
- Templates
  - **Rational Rose (examples and template for how to structure your Rose models)**
  - **Word (30+)**
  - **SoDA (10+)**
  - **MS Project**
- Development kit - guidelines, tools, templates for customizing the process
- Access to Resource Center (white papers, updates, hints, and add-on products)

R

Tighter Integration with Tools



R

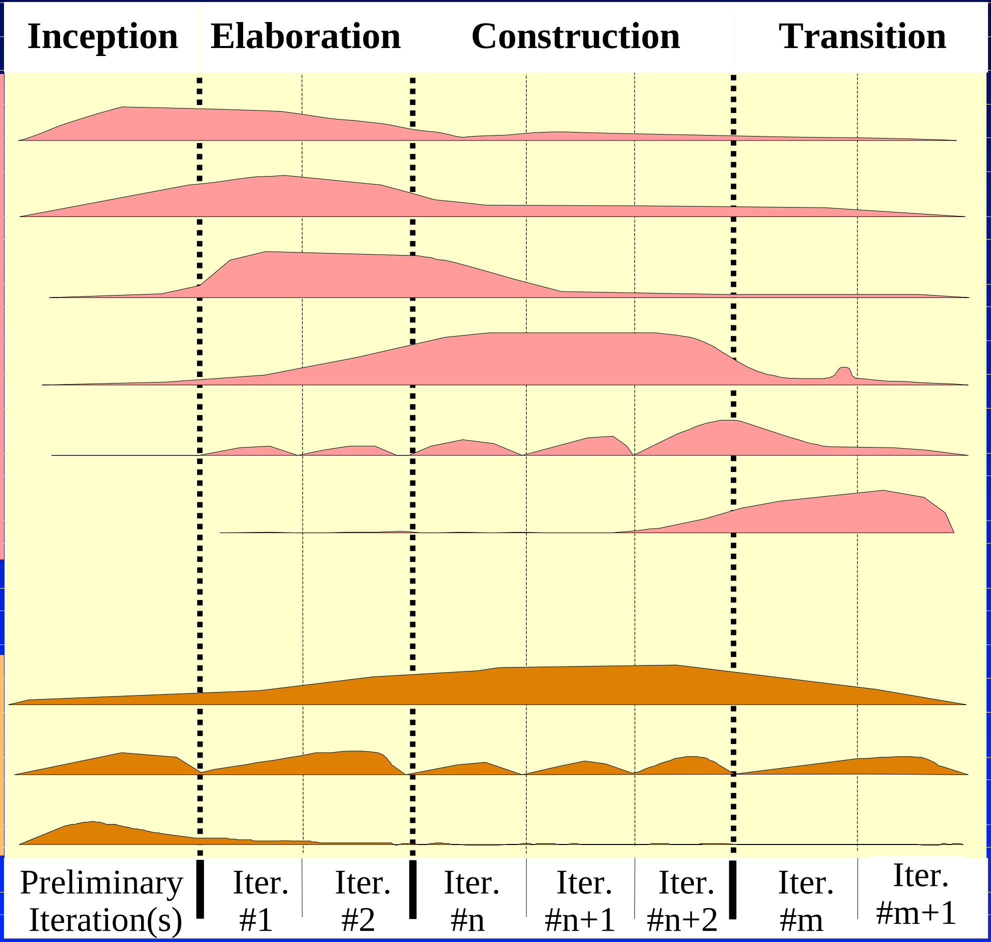
Rational Unified Process - Books

- Included in the product
  - *Rational Unified Process - An Introduction*, Philippe Kruchten, Addison-Wesley
- Other recommended books
  - *Software Project Management - A Unified Framework*, Walker Royce, Addison-Wesley
  - *Unified Modeling Language - An Application Guide*, Booch, Rumbaugh, Jacobson, Addison-Wesley
  - *Unified Software Development Process*, Jacobson, Booch, Rumbaugh, Jacobson, Addison-Wesley - coming Q1, 1999

R

Process Training

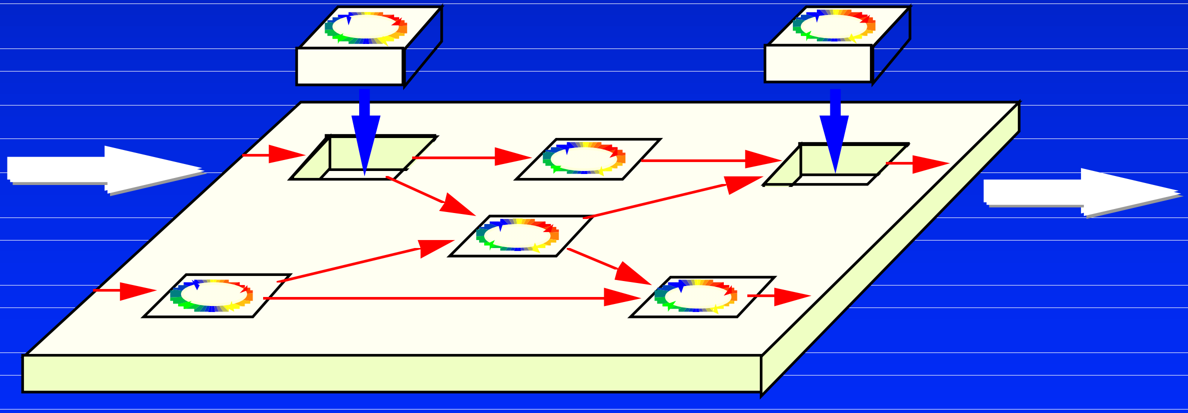
Courses	RUPO
	Business Modeling
RMUC	Requirements
OOAD	Analysis & Design
	Implementation
ASQ	Test
	Deployment
CCM	Configuration Management
OOPM	Project Management
	Environment



R

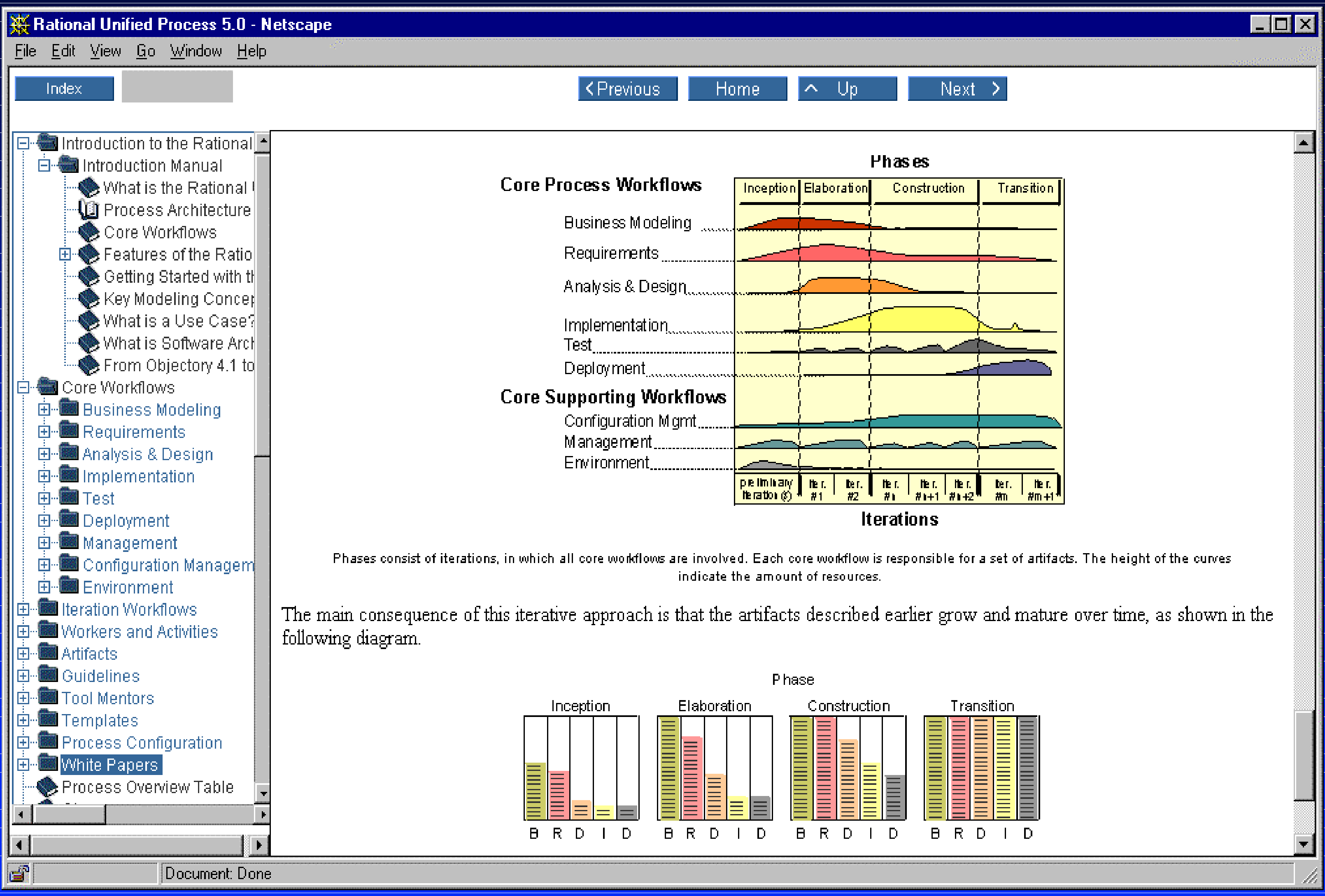
Rational Unified Process: Tailorable

- Use in whole or in part
- Tailor by creating a project-specific or organization-specific “Development Case”
- Development kit - guidelines, tools and templates for customizing the process



R

Demo



Phases consist of iterations, in which all core workflows are involved. Each core workflow is responsible for a set of artifacts. The height of the curves indicate the amount of resources.

The main consequence of this iterative approach is that the artifacts described earlier grow and mature over time, as shown in the following diagram.

R

Agenda

- What is the Rational Unified Process
- Implementing Best Practices
- Phases, Iterations, Workflows and Activities
- The Product
- ➡ Implementing the Rational Unified Process
- Software Economics
- Case Study: Skandia IT
- Summary

R

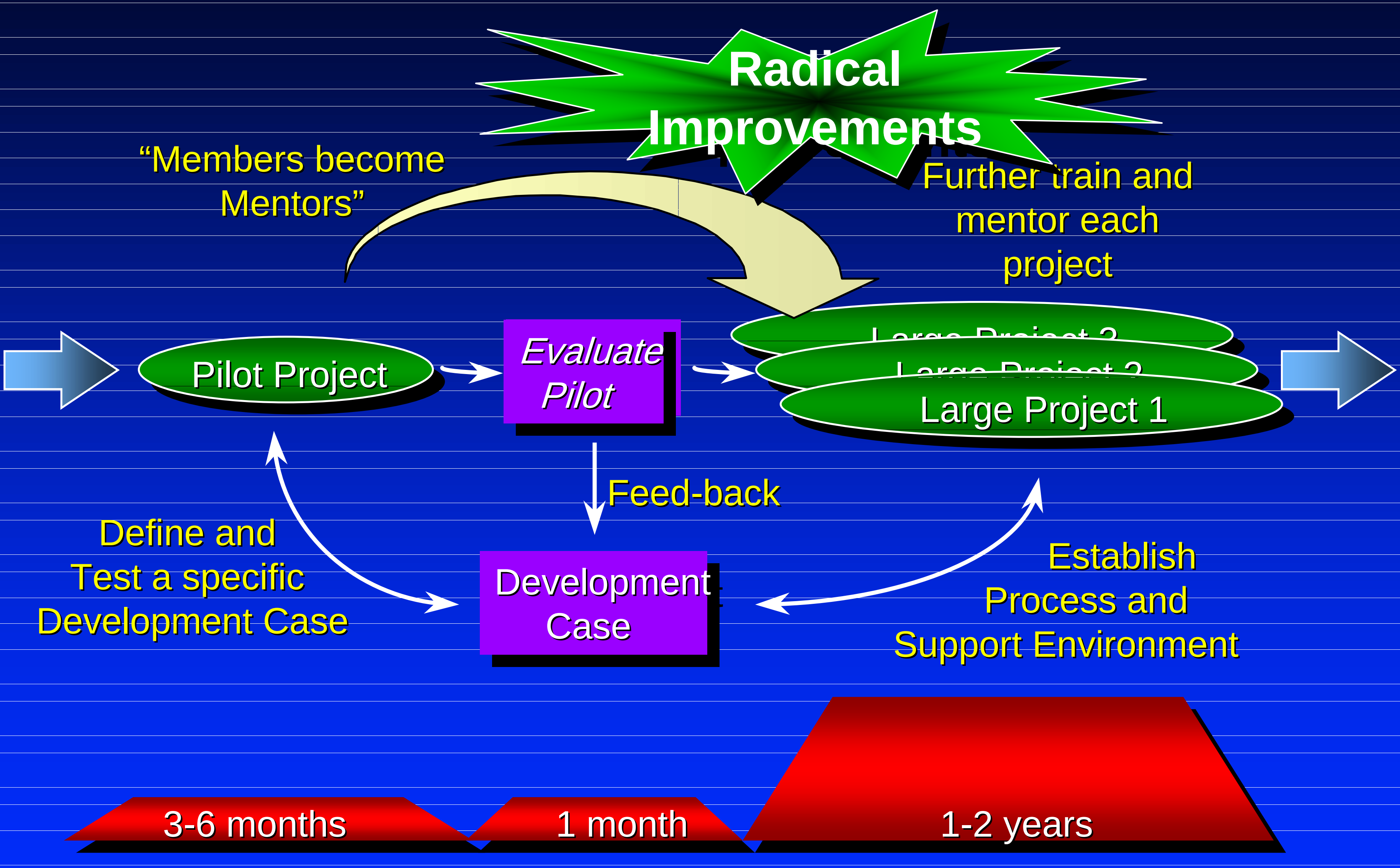
Approach 1 - The Interactive Knowledgebase...

- Limited training
- Decide what artifacts to produce
- Look at the process when you need help producing the artifacts
- Benefits increases over time as people start following the process…

Easy to adopt

R

Approach 2: The Full Scale Adoption….



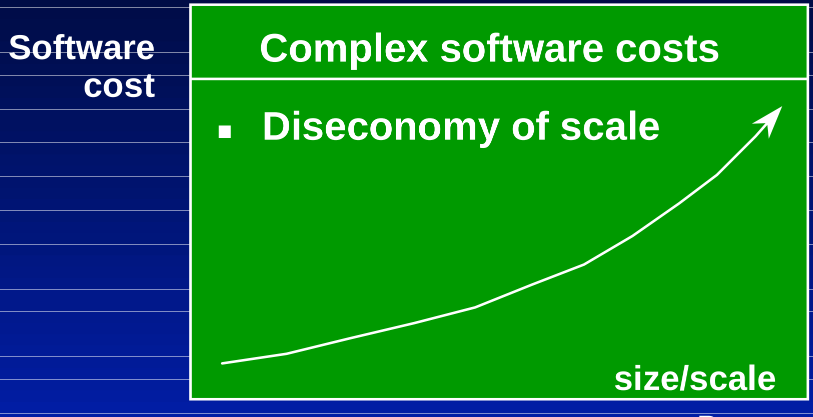
R

Agenda

R

What Makes Systems Complex?

- Performance constraints
- Time-to-market pressures
- Certification requirements
- Distributed, real-time requirements
- Size and geographic distribution of the engineering team
- Reliability and fault-tolerance requirements
- Rate of requirements and technology change
- The interplay of these factors



Software costs = E $*$ $(\text{Size})^P$

R

Current Software Technology Thrusts

$$\text{Software Costs} = E * (\text{Size})^P$$

E Environment technologies and tools	• Integrated tools (compiler-editor-debugger-CM) • Open systems • Hardware platform performance • Automation of documentation, testing, quality analyses
Size (Human generated code) Component based development technologies	• Reuse, commercial-off-the-shelf (COTS) products • Object-oriented (analysis, design, programming) • Higher level languages (Ada95, C++, VB, Java, etc.) • CASE tools (automatic code generation) • Distributed middleware
P Process technologies and teamwork	• Iterative development • Process maturity models • Architecture-first development • Acquisition reform • Training and personnel skill development

R

Next-Generation Cost Models

	R&D Component		Production Component
Software cost =	$E_{Arch} \ (Size_{Arch})^{P_{Arch}}$	+	$E_{App} \ (Size_{App})^{P_{App}}$
Team size:	Arch: 5 to 10 S/W engineers Apps: 5 to 10 mission engineers Small and expert as possible		Arch: 5 to 10 S/W engineers Apps: As many as needed Large and diverse as needed
Product:	Executable architecture Production plans Mission scenarios/models		Deliverable, useful function Tested baselines Warranted quality
Focus:	Design and integration Host development environment		Implement, test, and maintain Target technology
Phases:	Inception and elaboration		Construction and transition

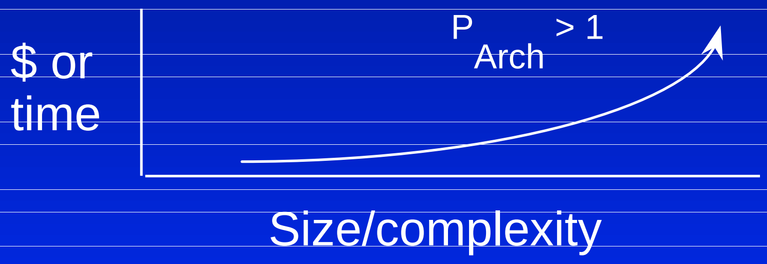
Next-Generation Cost Models

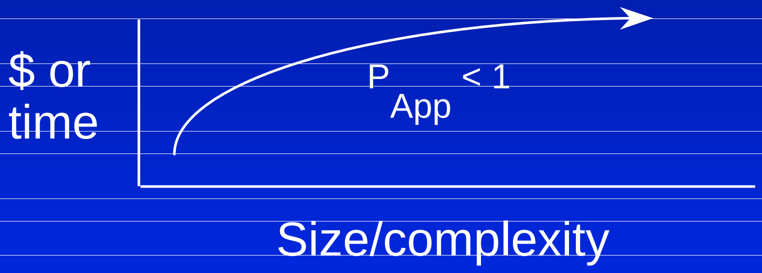
- Risk resolution, low-fidelity plan
- Schedule/technology driven
- Risk sharing contracts/funding

- Low-risk, high-fidelity plan
- Cost driven
- Fixed-price contracts/funding

N month design phase

N/2 month production increments





Architecture-centric Development

R

Next-Generation Productivities

- Today's focus:

Maximize SLOC per staff-month

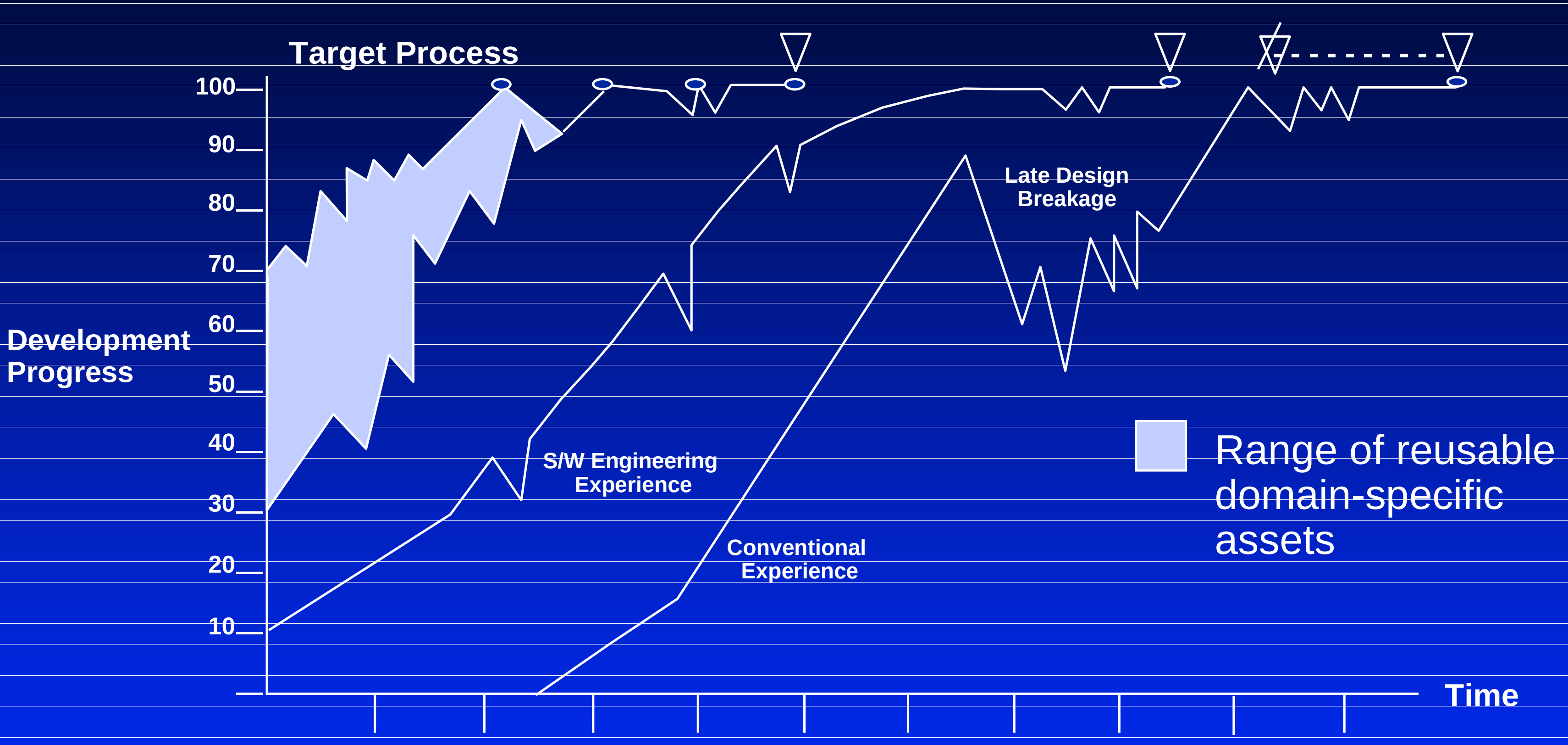
- Tomorrow's focus:

Minimize the amount of custom development

Component-Based Development

R

Automated Component-Based Development



- Integrated environments
- Reusable components (architectures, instrumentation, etc.)
- Platform independence
- Integrated components (reuse, custom, COTS, adapted)
- Off-the-shelf solutions to most of the difficult computer science issues

R

Agenda

- What is the Rational Unified Process
- Implementing Best Practices
- Phases, Iterations, Workflows and Activities
- The Product
- Implementing the Rational Unified Process
- Software Economics
- ➡ Case Study: Skandia IT
- Summary

R

Case Study - Skandia IT

Largest insurance company in the Nordic countries.

How Did Skandia IT Increase Productivity with 80% in a year?

Skandia IT 1996

- Legacy systems could not be changed as fast as business required
- EEC opened up border for international competition
- Project performance unpredictable
- Systems delivered often not meeting end user expectations
- Difficult to attract experienced developers

Action Plan

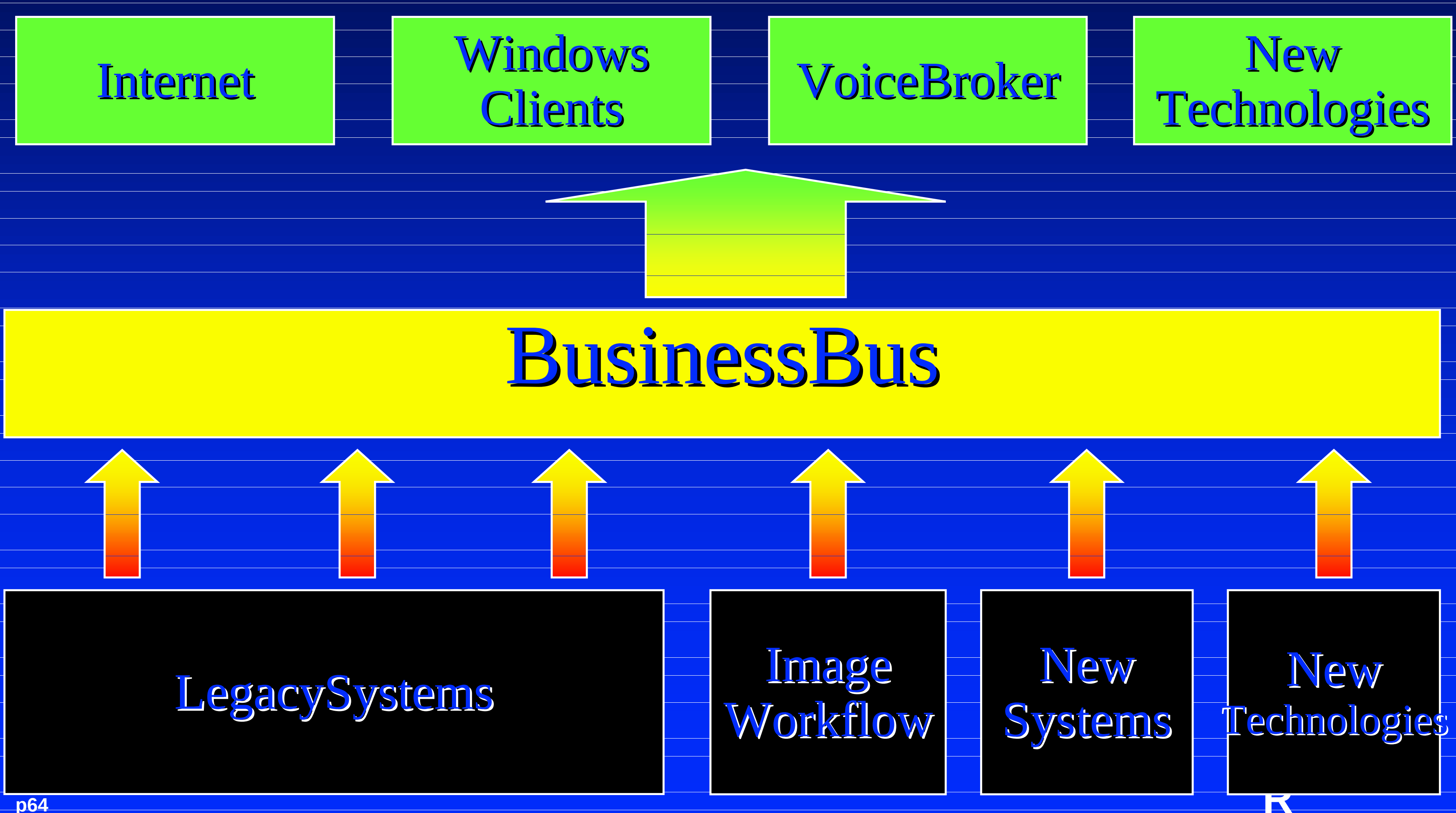
- Invest in Microsoft and Rational technologies
  - Visual Studio, COM/DCOM, MTS, SQL Server
  - Rational Rose, Rational Unified Process, Rational SoDA
- Recruit top consultants and employees
- Put employees through extensive on-the-job training program
- Deploy a component-based architecture

R

Reuse and Architecture Team

- Highly qualified staff of 20 developers / architects
- Evaluated and customized tool environment
- Evaluated and customized process
- Developed reusable components
- Put the architecture in place

System Architecture



Skandia IT 1997

- 10 projects - all on time
- Independent Gartner Group Evaluation
  - **80% in-house productivity increase in 1 year**
  - **40% more cost effective than Nordic average**
- Ahead of competition
- No problems recruiting top personnel

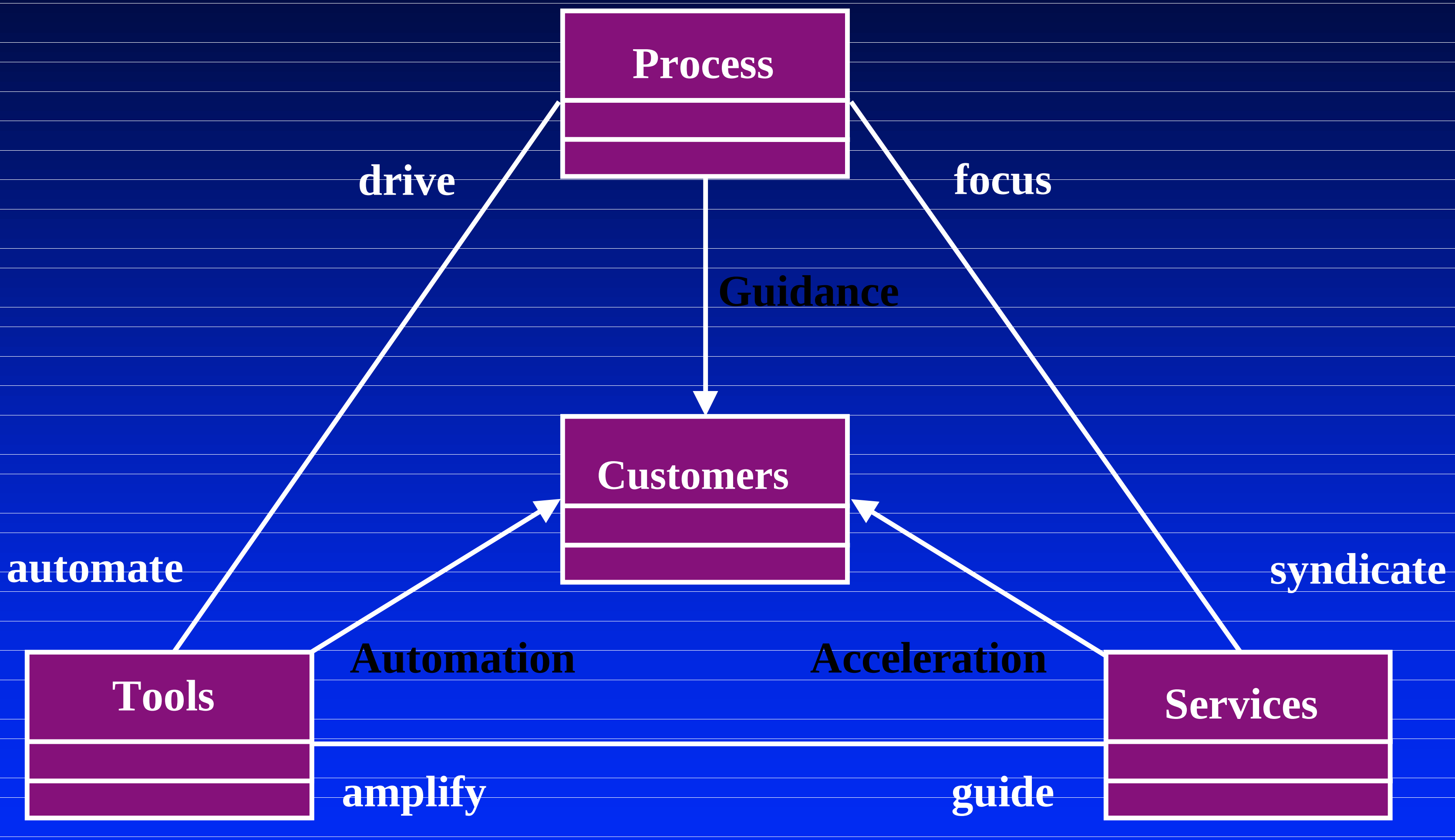
R

Agenda

- What is the Rational Unified Process
- Implementing Best Practices
- Phases, Iterations, Workflows and Activities
- The Product
- Implementing the Rational Unified Process
- Software Economics
- Case Study: Skandia IT
- ➡ Summary

R

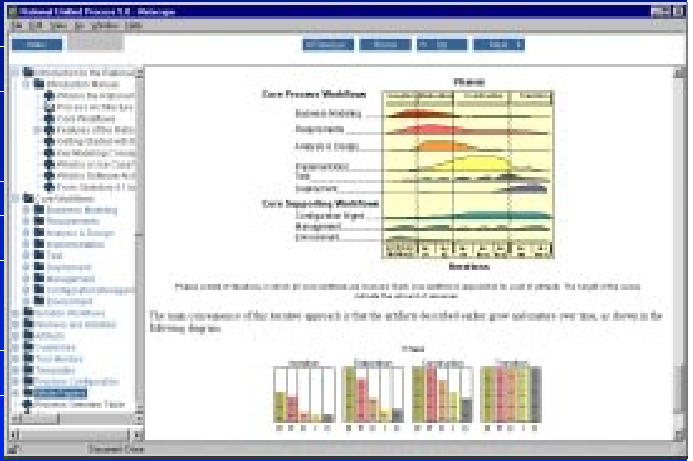
Rational’s Strategy



R

Why Rational Unified Process is the Right Choice

- Developed by the company that created the UML
- Unifies best practices from many disciplines into a full lifecycle process
- Integrated with Rational’s tools
- Online mentor on your desktop
- Supported by comprehensive professional education



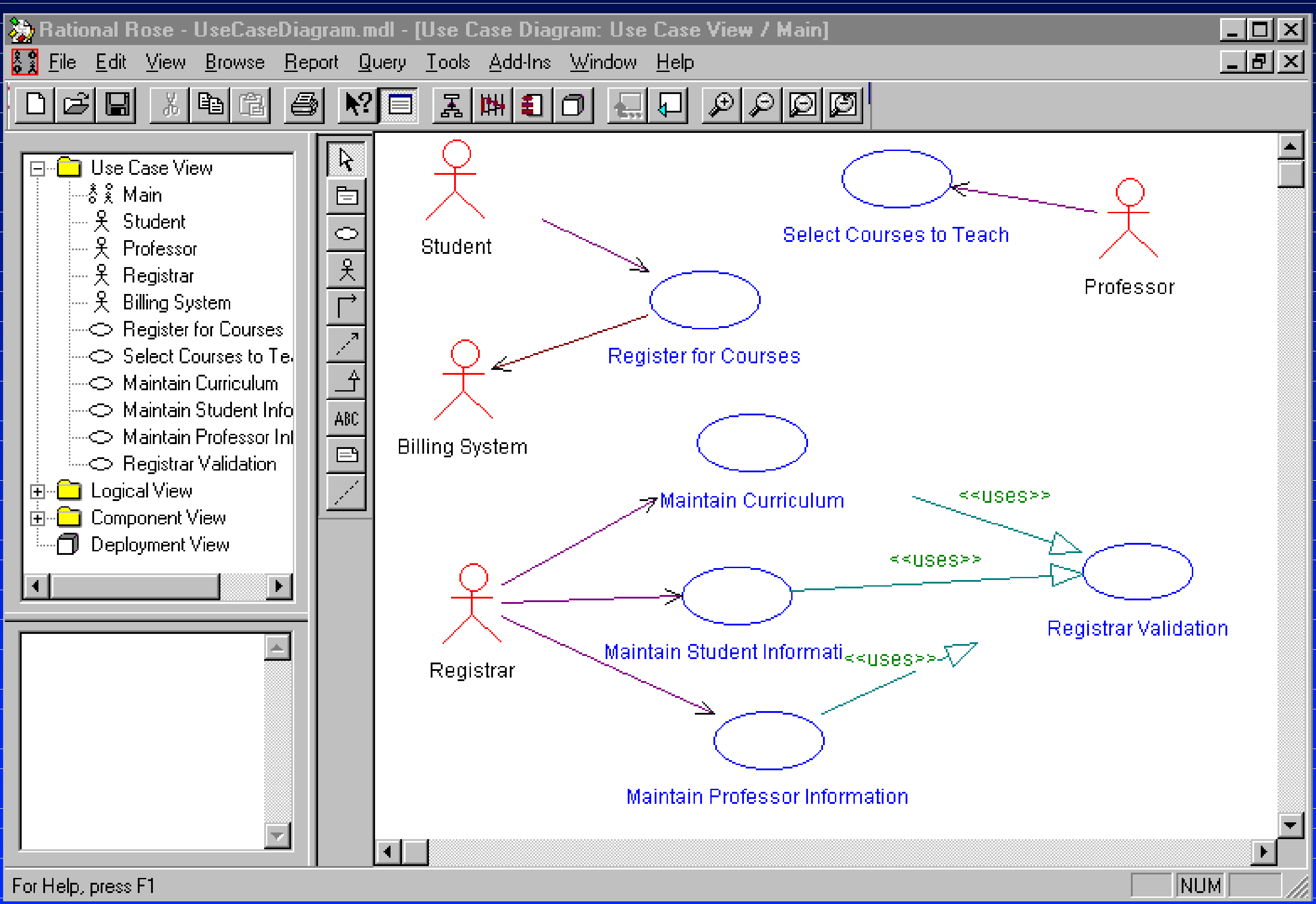
Delivers unprecedented content to a low price

R

Goodies

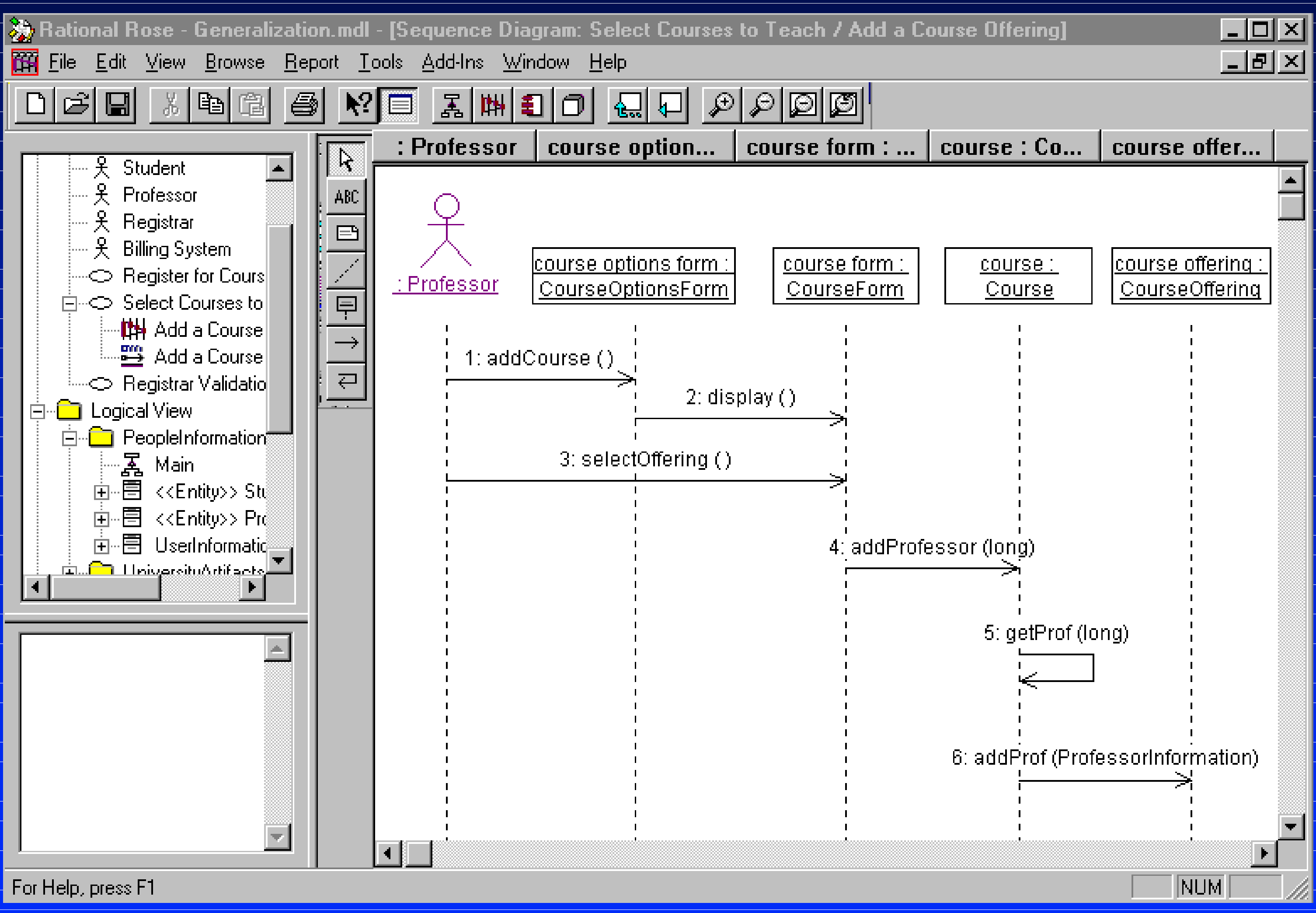
R

Use case diagram



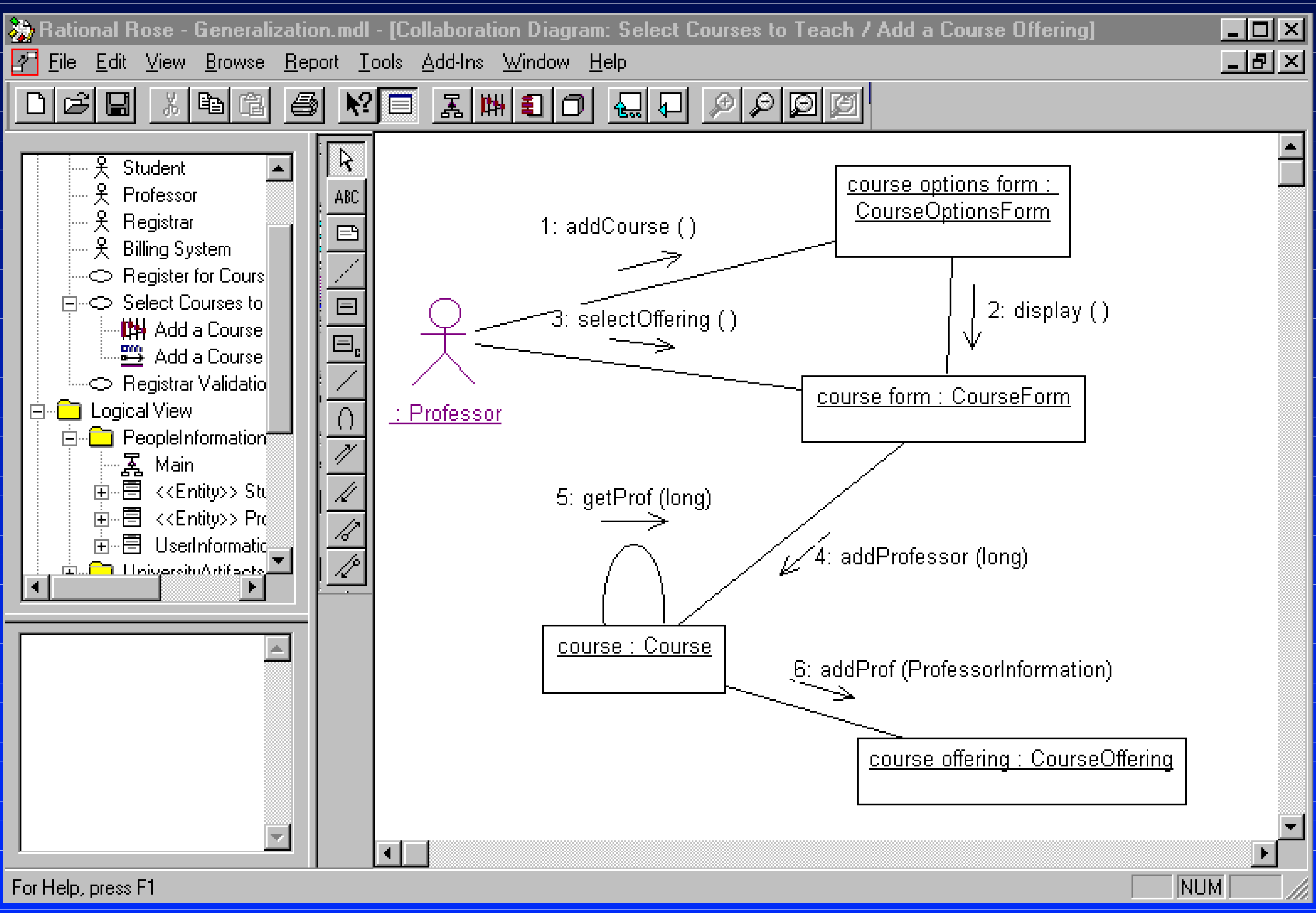
R

Scenario diagram



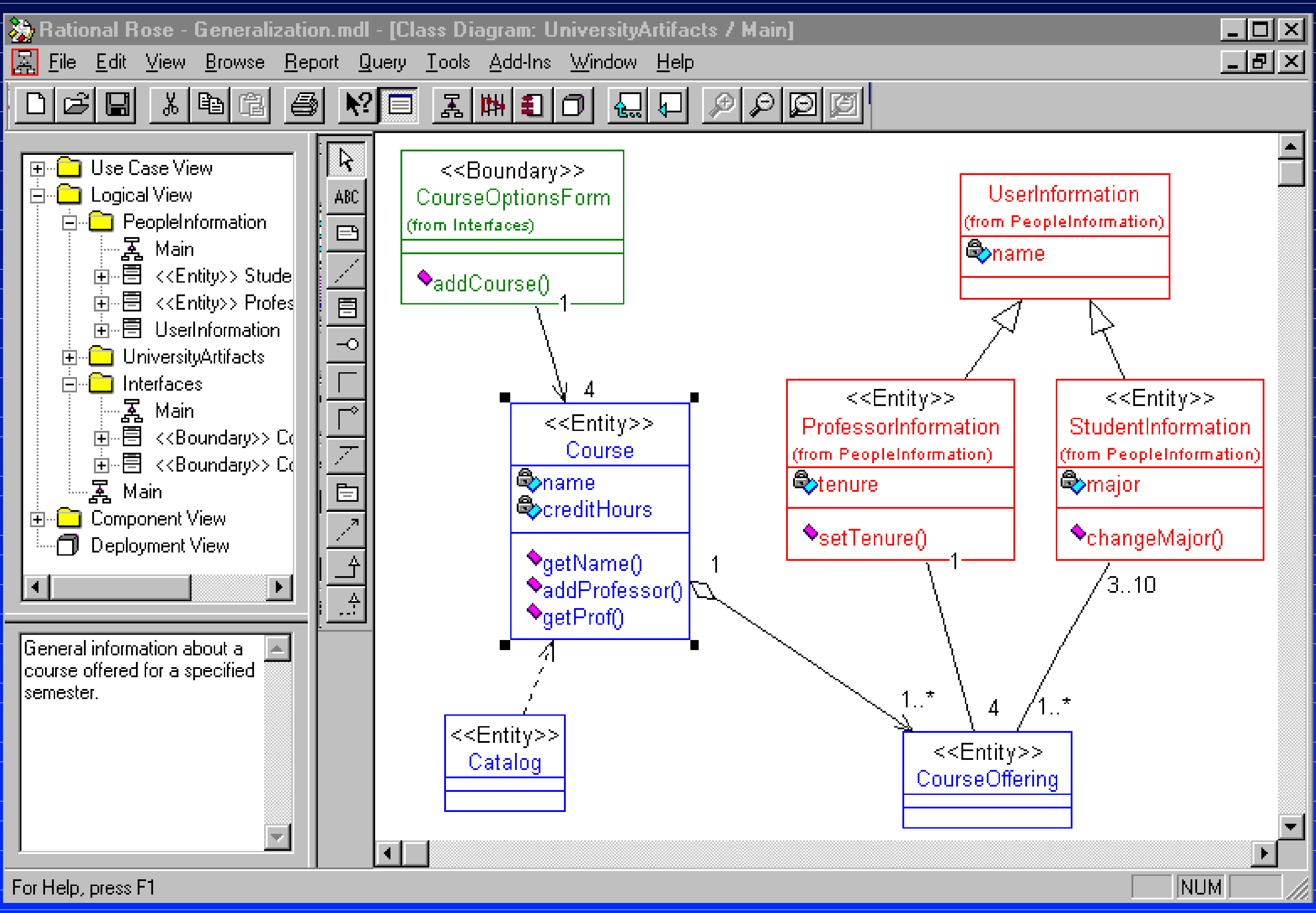
R

Collaboration diagram



R

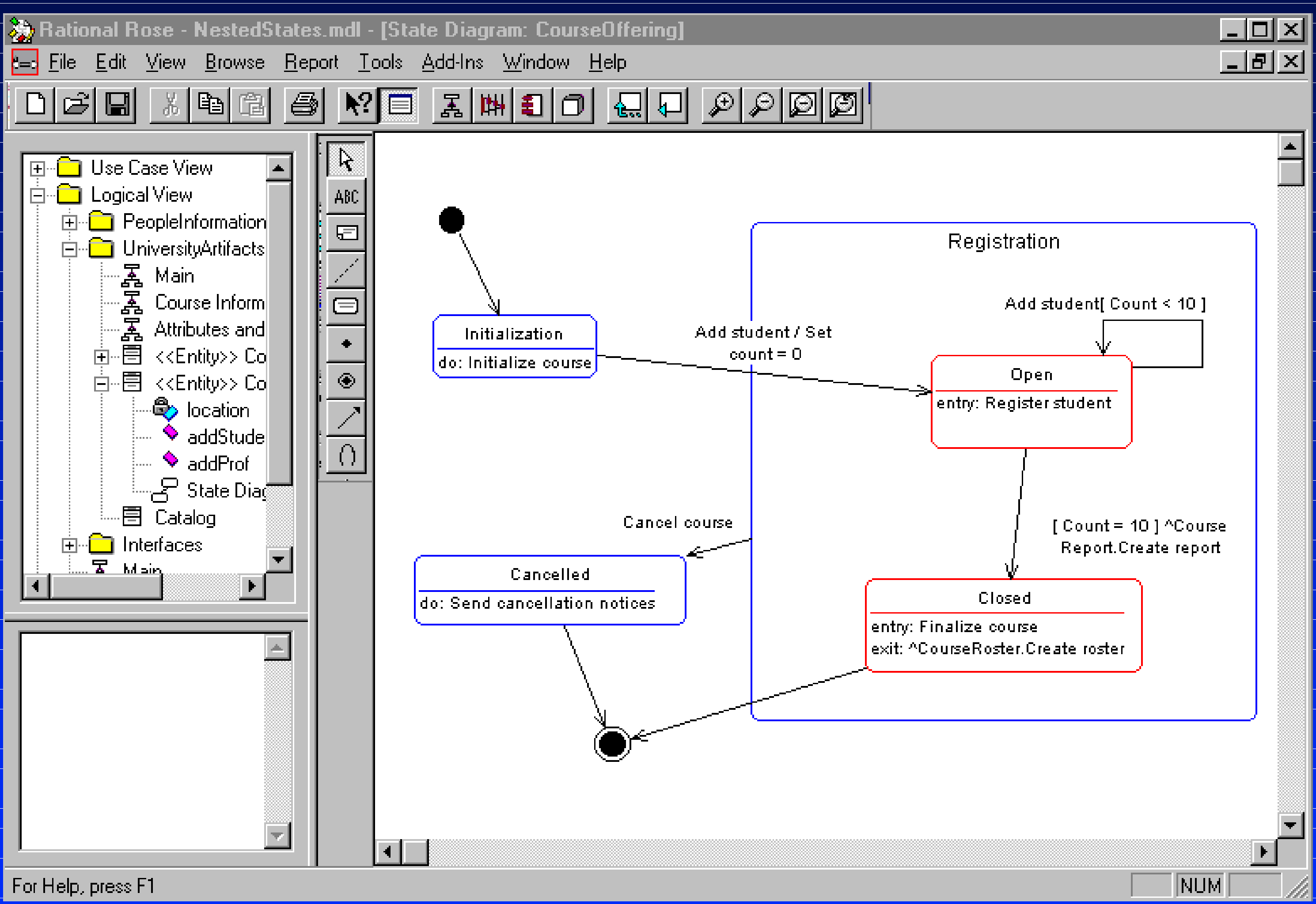
Class diagram


Rational Rose - Generalization.mdl - [Class Diagram: UniversityArtifacts / Main]
File Edit View Browse Report Query Tools Add-Ins Window Help
Use Case View
Logical View
PeopleInformation
Main
<<Entity>> Stude
<<Entity>> Profes
UserInformation
UniversityArtifacts
Interfaces
Main
<<Boundary>> Co
<<Boundary>> Co
Main
Component View
Deployment View
<<Boundary>> CourseOptionsForm (from Interfaces)
addCourse()
<<Entity>> Course
name
creditHours
getName()
addProfessor()
getProf()
<<Entity>> Catalog
UserInformation (from PeopleInformation)
name
<<Entity>> ProfessorInformation (from PeopleInformation)
tenure
setTenure()
<<Entity>> StudentInformation (from PeopleInformation)
major
changeMajor()
<<Entity>> CourseOffering
1
4
1
1..*
1
4
3..10
1..*
NUM

General information about a course offered for a specified semester.

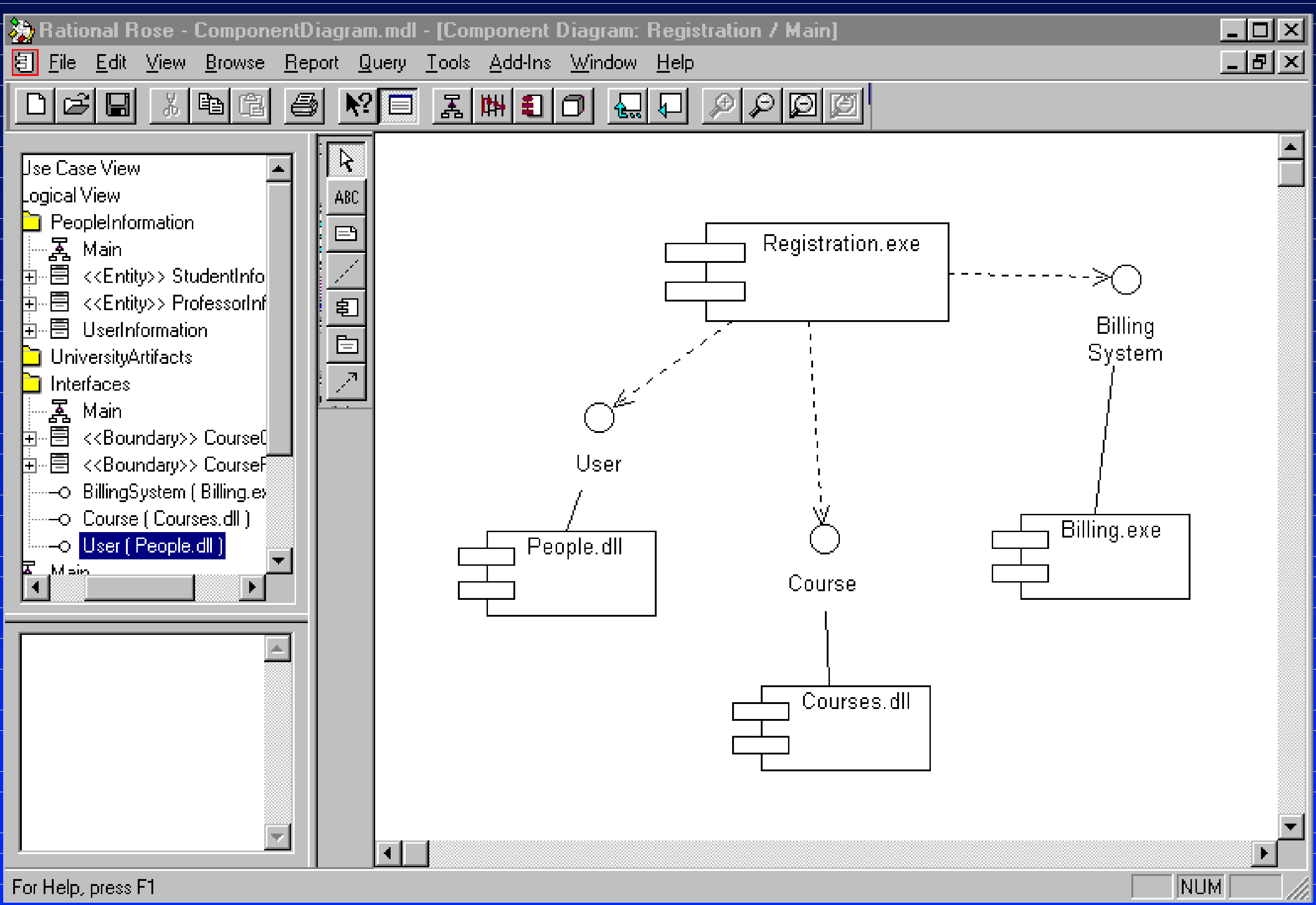
For Help, press F1

State transition diagram



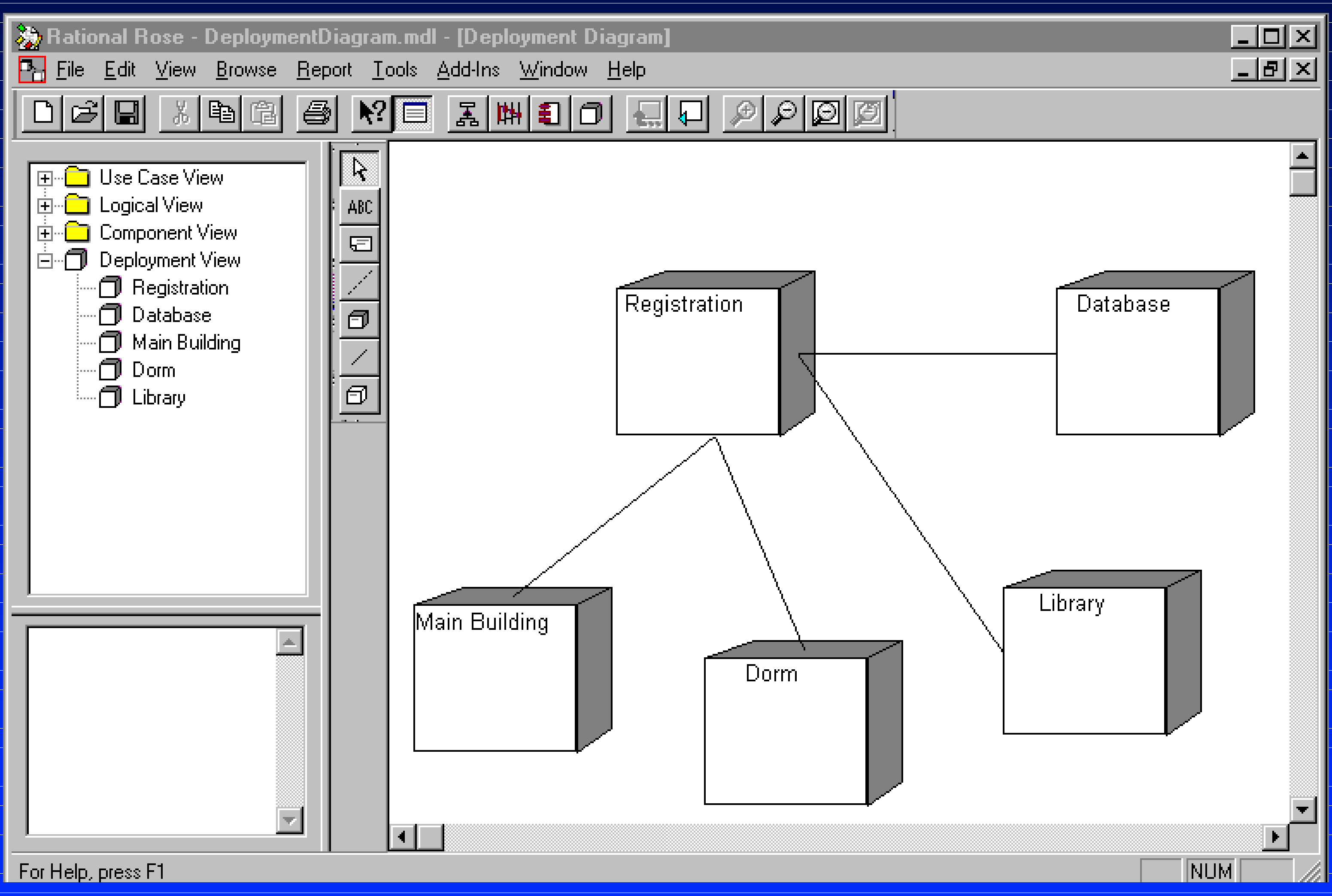
R

Component diagram



R

Deployment diagram



R