

Software Architecture and the UML



Grady Booch

RATIONAL
SOFTWARE

Architecting a dog house



Can be built by one person
Requires
Minimal modeling
Simple process
Simple tools

RATIONAL
SOFTWARE

Architecting a house



Built most efficiently and timely by a team
Requires
Modeling
Well-defined process
Power tools

RATIONAL
SOFTWARE

Architecting a high rise



RATIONAL
SOFTWARE

Early architecture



Progress

- Limited knowledge of theory

RATIONAL
SOFTWARE

Modern architecture



Progress

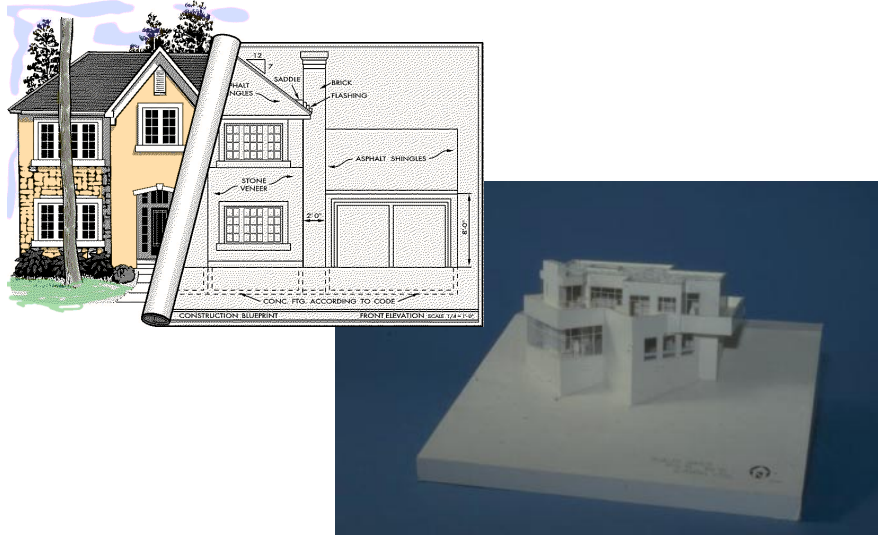
- Advances in materials
- Advances in analysis

Scale

- 5 times the span of the Pantheon
- 3 times the height of Cheops

RATIONAL
SOFTWARE

Modeling a house



RATIONAL
SOFTWARE

Movements in civil architecture

- Bronze age/Egyptian (Imhotep)
 - Grecian/Roman (Vitruvius)
 - Byzantine/Romanesque
 - Gothic
 - Mannerism (Michelangelo, Palladio)
 - Baroque
 - Engineering/Rational/National/Romantic
 - Art nouveau
 - Modern movement (Wright, LeCorbusier)
- Progress**
- Imitation of previous efforts
 - Learning from failure
 - Integration of other forces
 - Experimentation

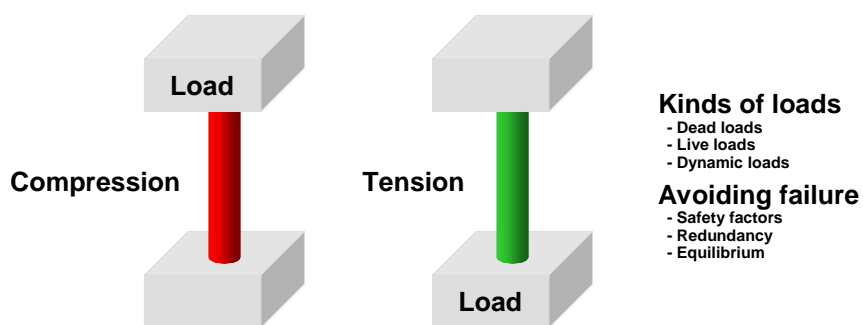
RATIONAL
SOFTWARE

Kinds of civil architecture

- Community
 - houses, flats and apartments, gardens, education, hospitals, religion
- Commerce
 - shops and stores, restaurants, hotels, office buildings, banks, airports
- Industry
 - industrial buildings, laboratories, farm buildings
- Leisure
 - sport, theaters and cinemas, museums

RATIONAL
SOFTWARE

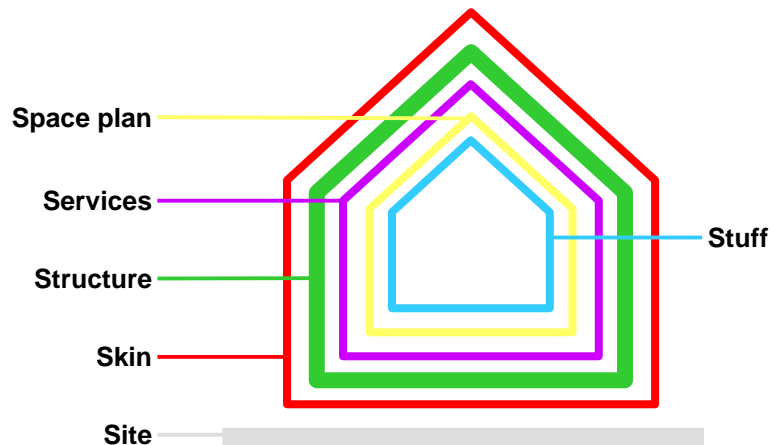
Forces in civil architecture



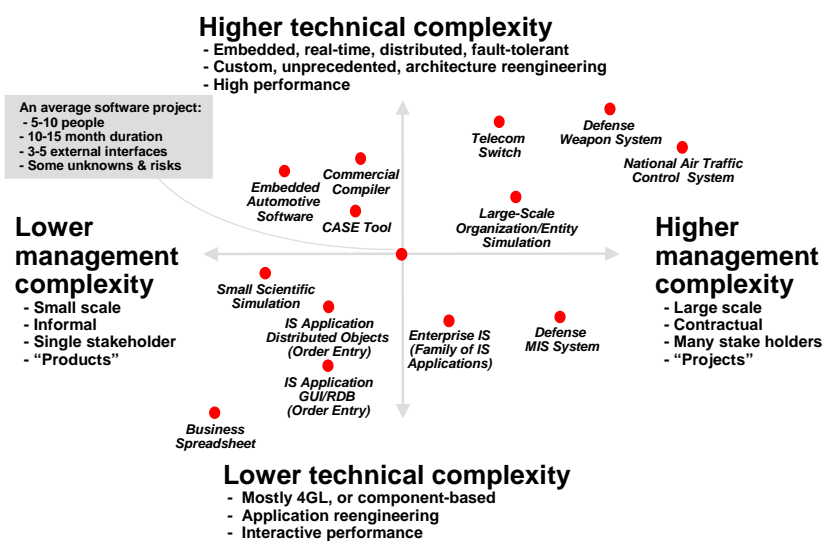
Any time you depart from established practice, make ten times the effort, ten times the investigation. Especially on a very large project.
- LeMessurier

RATIONAL
SOFTWARE

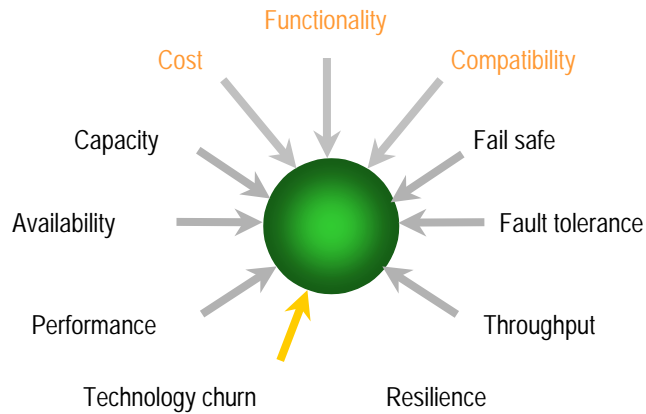
Shearing layers of change

RATIONAL
SOFTWARE

Dimensions of software complexity

RATIONAL
SOFTWARE

Forces in Software



The challenge over the next 20 years will not be speed or cost or performance; it will be a question of complexity.

Bill Raduchel, Chief Strategy Officer, Sun Microsystems

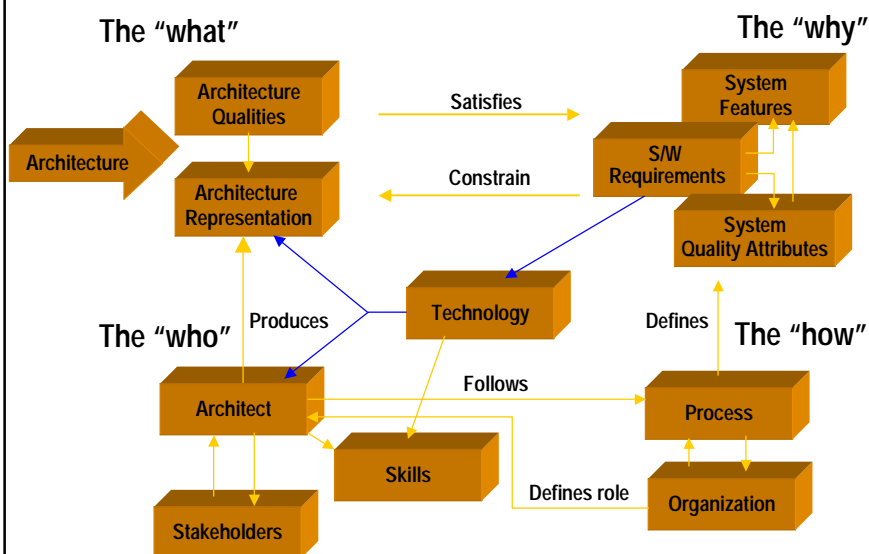
Our enemy is complexity, and it's our goal to kill it.

Jan Baan

RATIONAL
SOFTWARE

The domain of architecting

Wojtek Kozaczynski



RATIONAL
SOFTWARE

We all know that ...

Architecture and design are the same thing
Architecture and infrastructure are the same thing
<my favorite technology> is the architecture
A good architecture is the work of a single architect
Architecture is flat, one blueprint is enough
Architecture is just structure
System architecture precedes software architecture
Architecture cannot be measured and validated
Architecture is a Science
Architecture is an Art

Architecture defined (again)

Architecture n (1555) 1: the art of science of building, specifically, the art or practice of designing and building structures and esp. habitable ones 2 a: formation or construction as or as if as the result of conscious act <the ~ of the garden> b: a unifying or coherent form or structure <the novel lacks ~>

Architecture defined (yet again)

Mary Shaw, CMU
Grady Booch,
Philippe Kruchten,
Rich Reitman
Kurt Bittner, Rational

- Software architecture encompasses the set of significant decisions about the organization of a software system
 - selection of the structural elements and their interfaces by which a system is composed
 - behavior as specified in collaborations among those elements
 - composition of these structural and behavioral elements into larger subsystem
 - architectural style that guides this organization

RATIONAL
SOFTWARE

Architecture defined (continued)

Mary Shaw, CMU
Grady Booch,
Philippe Kruchten,
Rich Reitman
Kurt Bittner, Rational

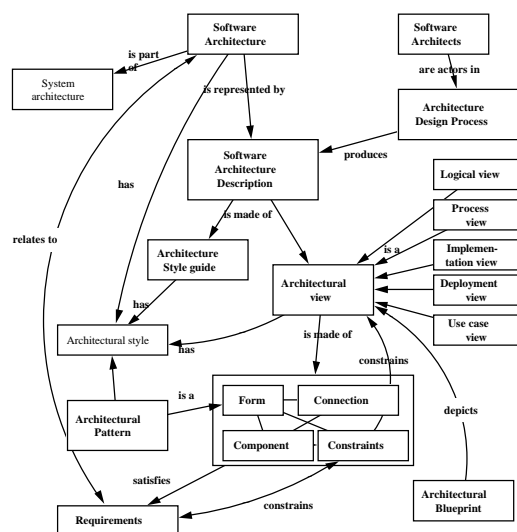
- Software architecture also involves
 - usage
 - functionality
 - performance
 - resilience
 - reuse
 - comprehensibility
 - economic and technology constraints and tradeoffs
 - aesthetic concerns

RATIONAL
SOFTWARE

Architectural style

- An architecture style defines a family of systems in terms of a pattern of structural organization.
- An architectural style defines
 - a vocabulary of components and connector types
 - a set of constraints on how they can be combined
 - one or more semantic models that specify how a system's overall properties can be determined from the properties of its parts

Architecture metamodel



Models

- Models are the language of designer, in many disciplines
- Models are representations of the system to-be-built or as-built
- Models are vehicle for communications with various stakeholders
- Visual models, blueprints
- Scale
- Models allow reasoning about some characteristic of the real system

RATIONAL
SOFTWARE

Many stakeholders, many views

- Architecture is many things to many different interested parties
 - end-user
 - customer
 - project manager
 - system engineer
 - developer
 - architect
 - maintainer
 - other developers
- Multidimensional reality
- Multiple stakeholders
 - ➡ multiple views, multiple blueprints

RATIONAL
SOFTWARE

Architectural view

- An architectural view is a simplified description (an abstraction) of a system from a particular perspective or vantage point, covering particular concerns, and omitting entities that are not relevant to this perspective

RATIONAL
SOFTWARE

Architecturally significant elements

- Not all design is architecture
- Main “business” classes
- Important mechanisms
- Processors and processes
- Layers and subsystems
- Architectural views = slices through models

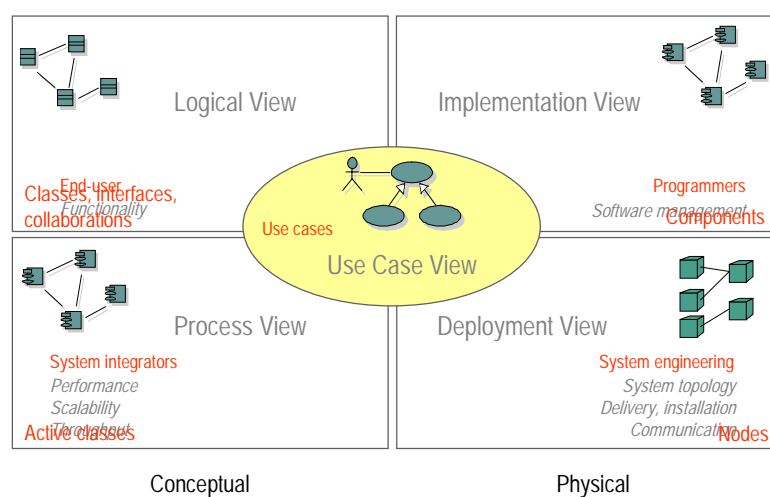
RATIONAL
SOFTWARE

Characteristics of a Good Architecture

- Resilient
- Simple
- Approachable
- Clear separation of concerns
- Balanced distribution of responsibilities
- Balances economic and technology constraints

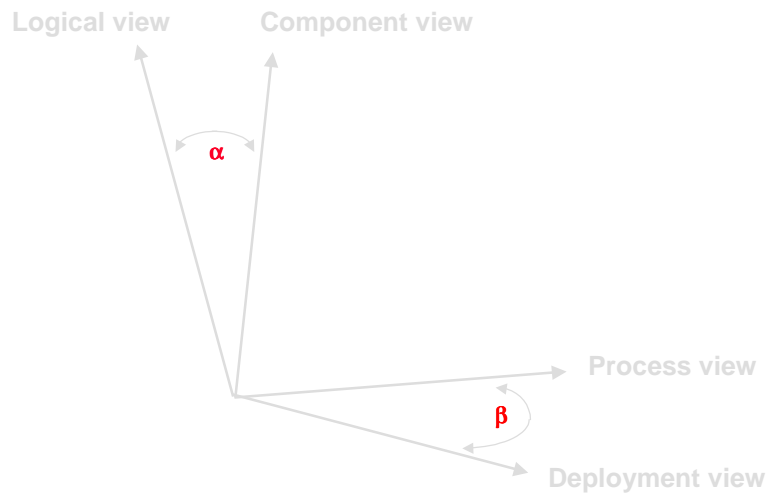
RATIONAL
SOFTWARE

Representing System Architecture



RATIONAL
SOFTWARE

Relation Between Views



RATIONAL
SOFTWARE

How many views?

- Simplified models to fit the context
- Not all systems require all views:
 - Single processor: drop deployment view
 - Single process: drop process view
 - Very Small program: drop implementation view
- Adding views:
 - Data view, security view

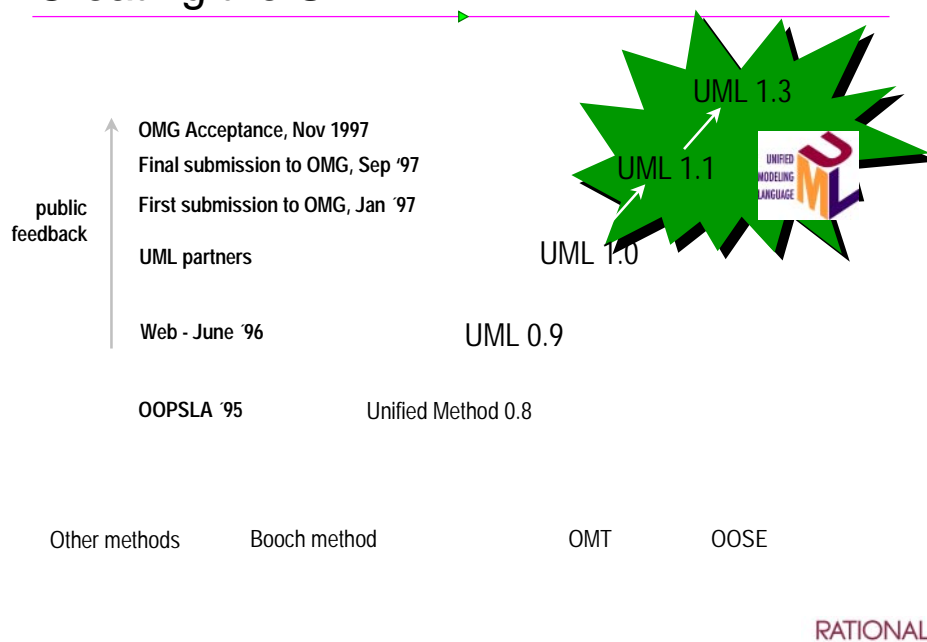
RATIONAL
SOFTWARE

The Value of the UML

- Is an open standard
- Supports the entire software development lifecycle
- Supports diverse applications areas
- Is based on experience and needs of the user community
- Supported by many tools

RATIONAL
SOFTWARE

Creating the UML

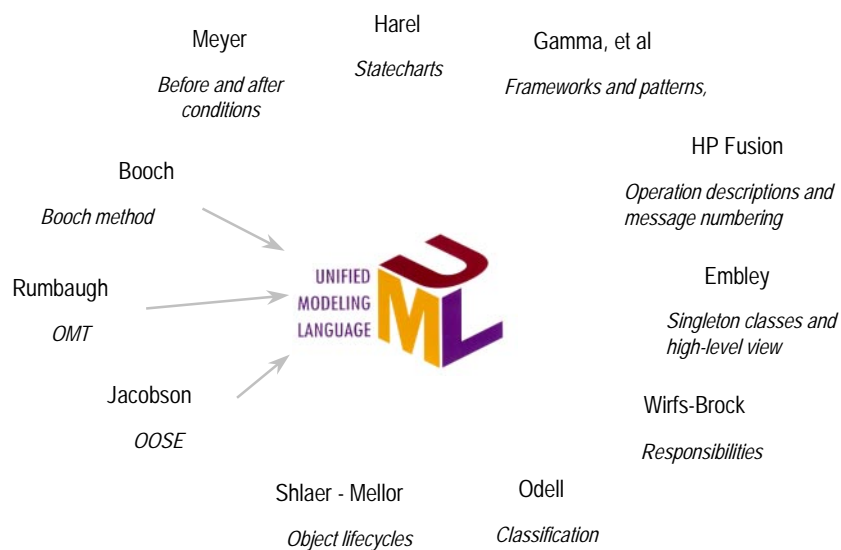


UML Partners

- Rational Software Corporation
- Hewlett-Packard
- I-Logix
- IBM
- ICON Computing
- Intellicorp
- MCI Systemhouse
- Microsoft
- ObjecTime
- Oracle
- Platinum Technology
- Taskon
- Texas Instruments/Sterling Software
- Unisys

RATIONAL
SOFTWARE

Contributions to the UML



RATIONAL
SOFTWARE

Overview of the UML

➤ The UML is a language for

- visualizing
- specifying
- constructing
- documenting



the artifacts of a software-intensive system

RATIONAL
SOFTWARE

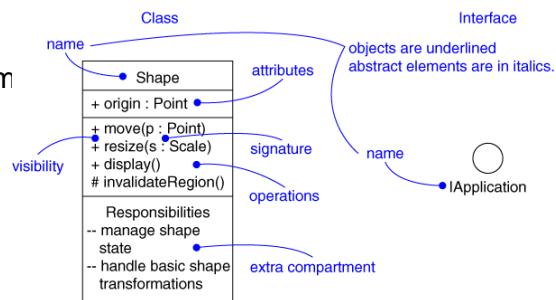
Overview of the UML

- Modeling elements
- Relationships
- Extensibility Mechanisms
- Diagrams

RATIONAL
SOFTWARE

Modeling Elements

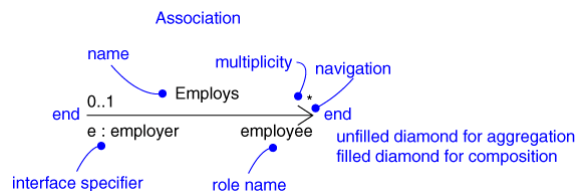
- Structural elements
 - class, interface, collaboration, use case, active class, component, node
- Behavioral elements
 - interaction, state machine
- Grouping elements
 - package, subsystem
- Other elements
 - note



RATIONAL
SOFTWARE

Relationships

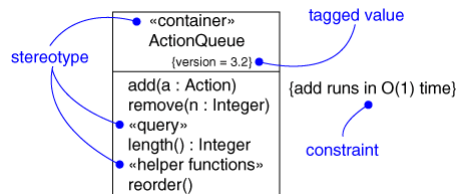
- Dependency
- Association
- Generalization
- Realization



RATIONAL
SOFTWARE

Extensibility Mechanisms

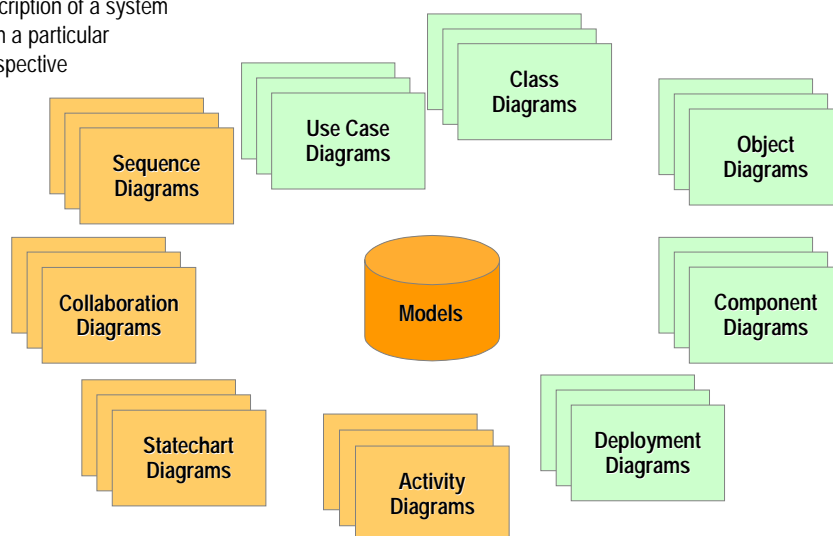
- Stereotype
- Tagged value
- Constraint



RATIONAL
SOFTWARE

Models, Views, and Diagrams

A **model** is a complete description of a system from a particular perspective



RATIONAL
SOFTWARE

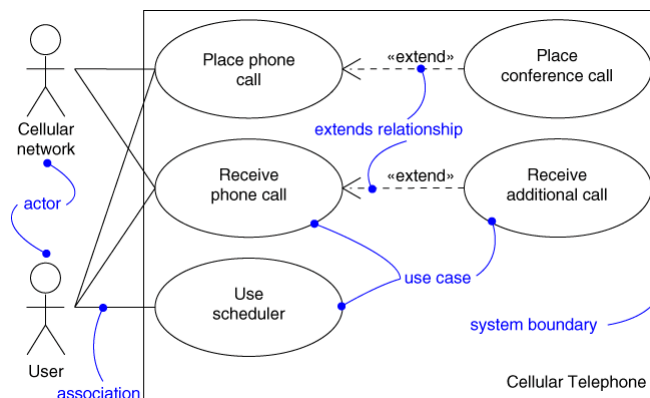
Diagrams

- A diagram is a view into a model
 - Presented from the aspect of a particular stakeholder
 - Provides a partial representation of the system
 - Is semantically consistent with other views
- In the UML, there are nine standard diagrams
 - Static views: use case, class, object, component, deployment
 - Dynamic views: sequence, collaboration, statechart, activity

RATIONAL
SOFTWARE

Use Case Diagram

- Captures system functionality as seen by users



RATIONAL
SOFTWARE

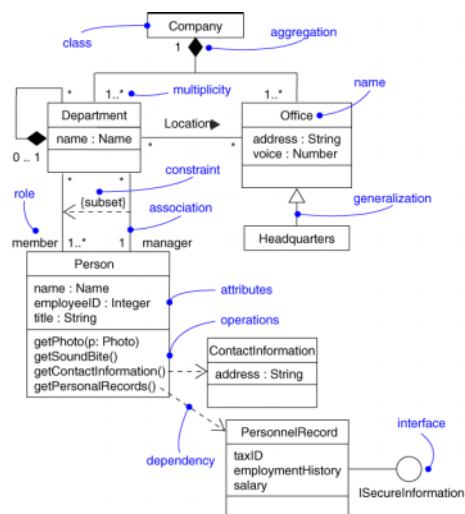
Use Case Diagram

- Captures system functionality as seen by users
- Built in early stages of development
- Purpose
 - Specify the context of a system
 - Capture the requirements of a system
 - Validate a system's architecture
 - Drive implementation and generate test cases
- Developed by analysts and domain experts

RATIONAL
S O F T W A R E

Class Diagram

- Captures the vocabulary of a system



RATIONAL
S O F T W A R E

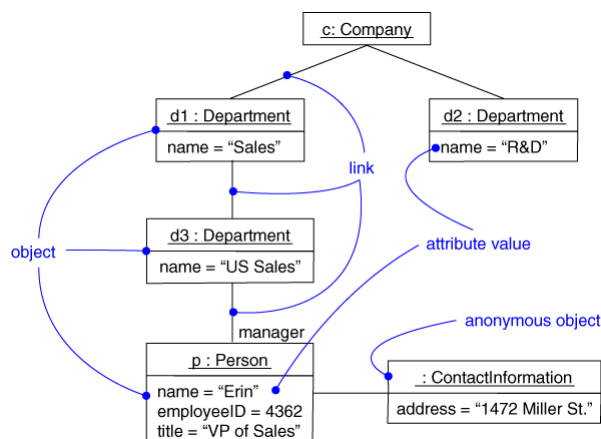
Class Diagram

- Captures the vocabulary of a system
- Built and refined throughout development
- Purpose
 - Name and model concepts in the system
 - Specify collaborations
 - Specify logical database schemas
- Developed by analysts, designers, and implementers

RATIONAL
SOFTWARE

Object Diagram

- Captures instances and links



RATIONAL
SOFTWARE

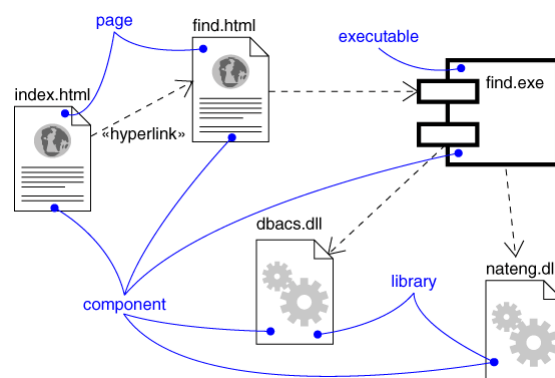
Object Diagram

- Shows instances and links
- Built during analysis and design
- Purpose
 - Illustrate data/object structures
 - Specify snapshots
- Developed by analysts, designers, and implementers

RATIONAL
SOFTWARE

Component Diagram

- Captures the physical structure of the implementation



RATIONAL
SOFTWARE

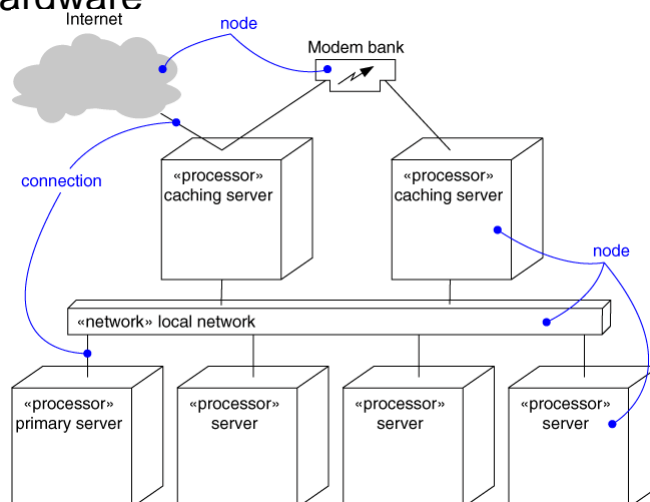
Component Diagram

- Captures the physical structure of the implementation
- Built as part of architectural specification
- Purpose
 - Organize source code
 - Construct an executable release
 - Specify a physical database
- Developed by architects and programmers

RATIONAL
SOFTWARE

Deployment Diagram

- Captures the topology of a system's hardware



RATIONAL
SOFTWARE

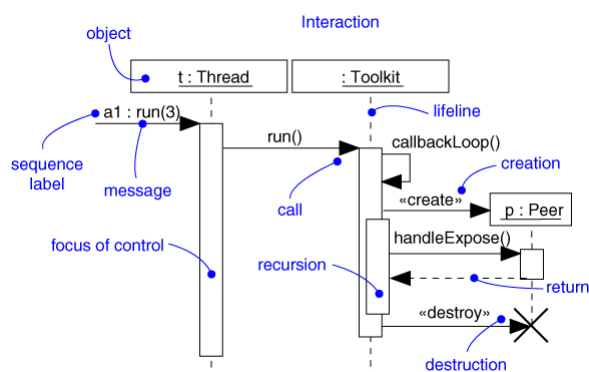
Deployment Diagram

- Captures the topology of a system's hardware
- Built as part of architectural specification
- Purpose
 - Specify the distribution of components
 - Identify performance bottlenecks
- Developed by architects, networking engineers, and system engineers

RATIONAL
S O F T W A R E

Sequence Diagram

- Captures dynamic behavior (time-oriented)



RATIONAL
S O F T W A R E

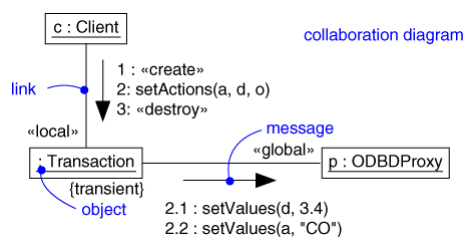
Sequence Diagram

- Captures dynamic behavior (time-oriented)
- Purpose
 - Model flow of control
 - Illustrate typical scenarios

RATIONAL
S O F T W A R E

Collaboration Diagram

- Captures dynamic behavior (message-oriented)



RATIONAL
S O F T W A R E

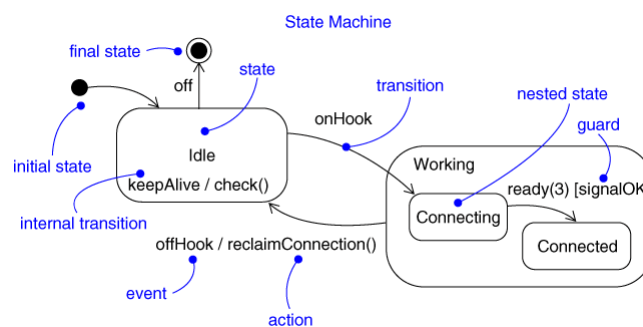
Collaboration Diagram

- Captures dynamic behavior (message-oriented)
- Purpose
 - Model flow of control
 - Illustrate coordination of object structure and control

RATIONAL
SOFTWARE

Statechart Diagram

- Captures dynamic behavior (event-oriented)



RATIONAL
SOFTWARE

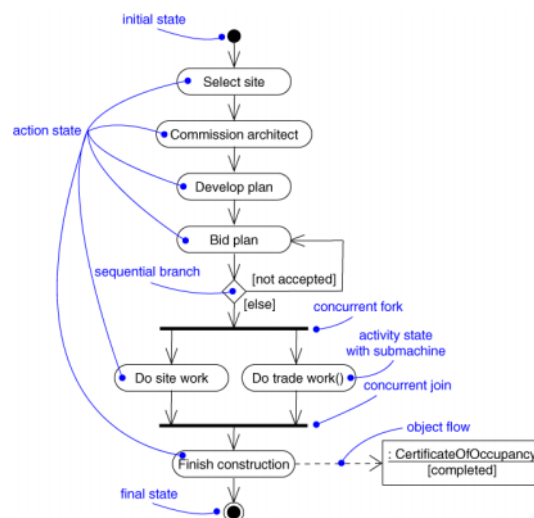
Statechart Diagram

- Captures dynamic behavior (event-oriented)
- Purpose
 - Model object lifecycle
 - Model reactive objects (user interfaces, devices, etc.)

RATIONAL
S O F T W A R E

Activity Diagram

- Captures dynamic behavior (activity-oriented)



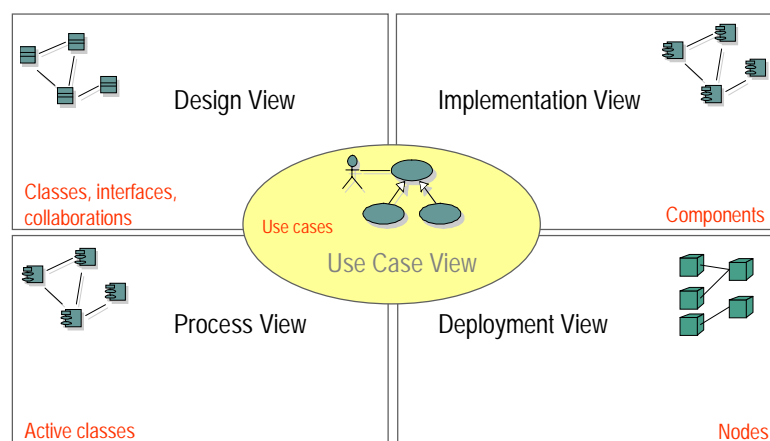
RATIONAL
S O F T W A R E

Activity Diagram

- Captures dynamic behavior (activity-oriented)
- Purpose
 - Model business workflows
 - Model operations

RATIONAL
SOFTWARE

Architecture and the UML



RATIONAL
SOFTWARE

Software engineering process

A set of partially ordered steps intended to reach a goal. In software engineering the goal is to build a software product or to enhance an existing one.

- Architectural process
 - Sequence of activities that lead to the production of architectural artifacts:
 - A software architecture description
 - An architectural prototype

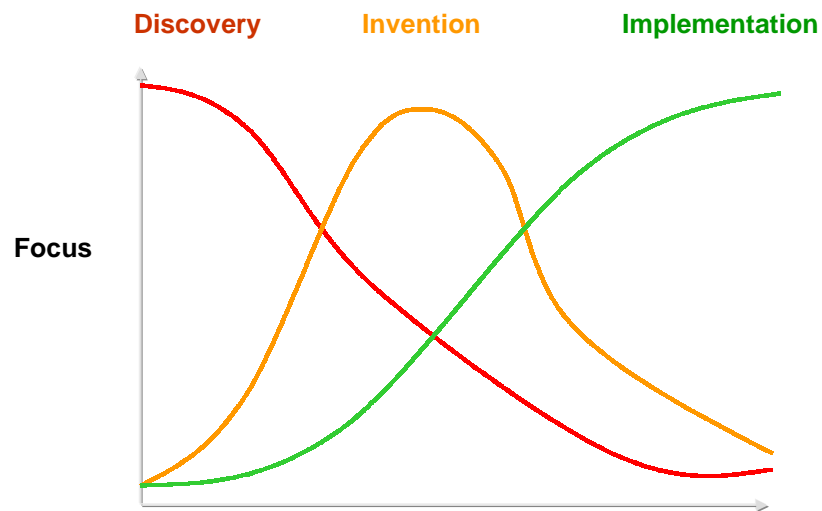
RATIONAL
SOFTWARE

Rational Unified Process

- Iterative
- Architecture-centric
- Use-case driven
- Risk confronting

RATIONAL
SOFTWARE

Focus over time



RATIONAL
SOFTWARE

Key concepts

- | | |
|---|--------------------------------|
| ➤ Phase, Iterations | When does architecture happen? |
| ➤ Process Workflows <ul style="list-style-type: none">- Activity, steps | What does happen? |
| ➤ Artifacts <ul style="list-style-type: none">- models- reports, documents | What is produced? |
| ➤ Worker: Architect | Who does it? |

RATIONAL
SOFTWARE

Lifecycle Phases



time

- Inception Define the scope of the project and develop business case
- Elaboration Plan project, specify features, and baseline the architecture
- Construction Build the product
- Transition Transition the product to its users

RATIONAL
SOFTWARE

Major Milestones

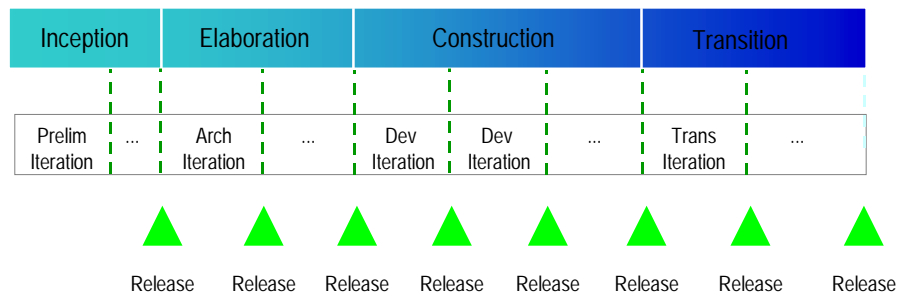


time



RATIONAL
SOFTWARE

Phases and Iterations

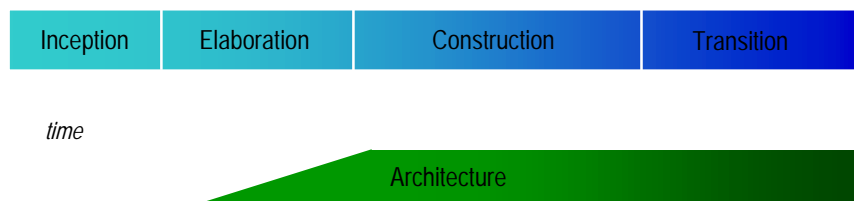


An iteration is a sequence of activities with an established plan and evaluation criteria, resulting in an executable release

RATIONAL
SOFTWARE

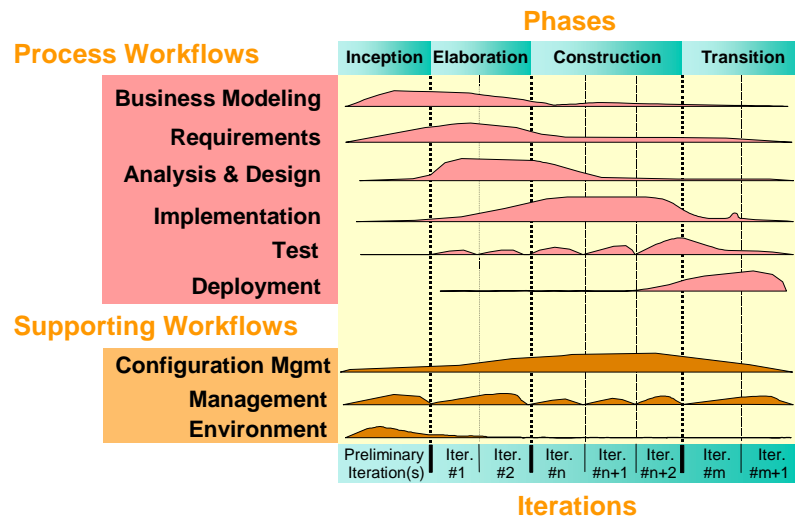
Architecture-Centric

- Models are vehicles for visualizing, specifying, constructing, and documenting architecture
- The Unified Process prescribes the successive refinement of an executable architecture



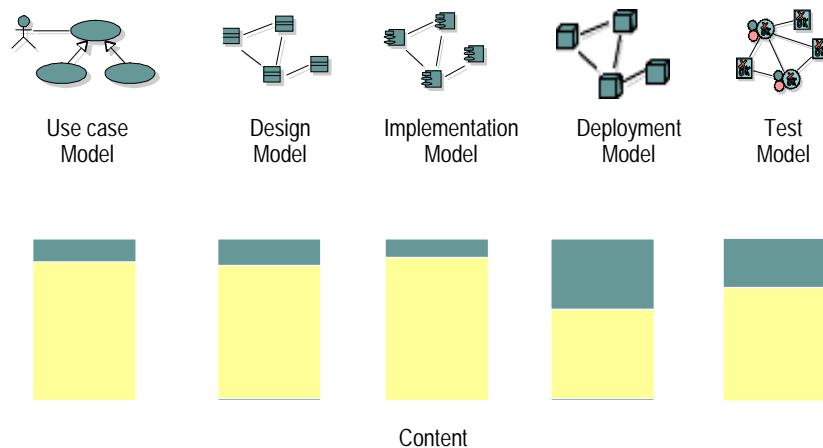
RATIONAL
SOFTWARE

Unified Process structure



RATIONAL
SOFTWARE

Architecture and Iterations



RATIONAL
SOFTWARE

Architectural design

- Identify, select, and validate “architecturally significant” elements
- Not everything is architecture
 - Main “business” classes
 - Important mechanisms
 - Processors and processes
 - Layers and subsystems
 - Interfaces
- Produce a Software Architecture Document

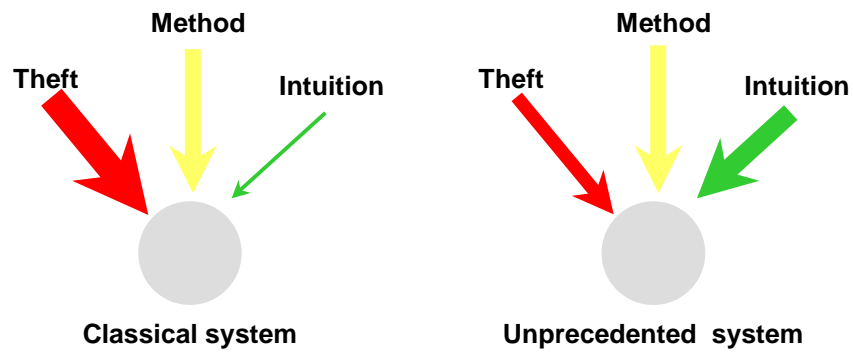
RATIONAL
SOFTWARE

Architectural design workflow

- Select scenarios: **criticality** and **risk** Use case view
 - Identify main **classes** and their responsibility Logical view
 - Distribute behavior on classes
 - Structure in subsystems, layers, define interfaces Implementation view
 - Define distribution and concurrency Deployment view
 - Implement architectural prototype Process view
 - Derive tests from use cases
 - Evaluate architecture
- Iterate***

RATIONAL
SOFTWARE

Sources of architecture



RATIONAL
SOFTWARE

Patterns

- A pattern is a solution to a problem in a context
- A pattern codifies specific knowledge collected from experience in a domain
- All well-structured systems are full of patterns
 - Idioms
 - Design patterns
 - Architectural patterns

RATIONAL
SOFTWARE

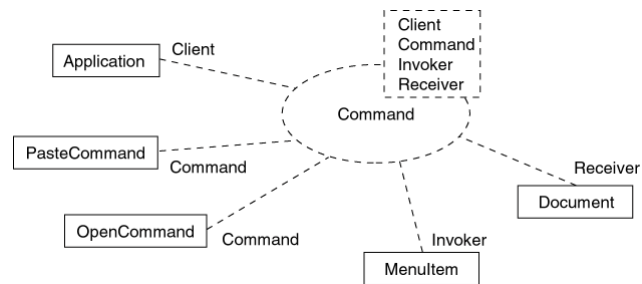
Mechanisms

- Screws
- Keys
- Rivets
- Bearings
- Pins, axles, shafts
- Couplings
- Ropes, belts, and chains
- Friction wheels
- Toothed wheels
- Flywheels
- Levers and connecting rods
- Click wheels and gears
- Ratchets
- Brakes
- Pipes
- Valves
- Springs
- Cranks and rods
- Cams
- Pulleys
- Engaging gears

Design patterns

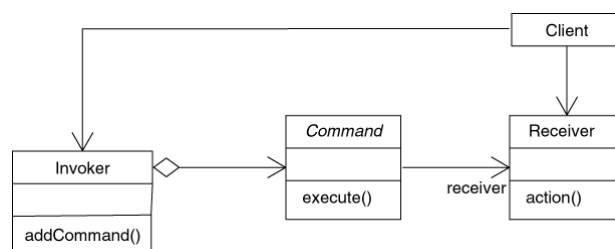
- Creational patterns
 - Abstract factory
 - Prototype
- Structural patterns
 - Adapter
 - Bridge
 - Proxy
- Behavioral patterns
 - Chain of responsibility
 - Mediator
 - Visitor
- Mechanisms are the soul of an architecture

Modeling a design pattern



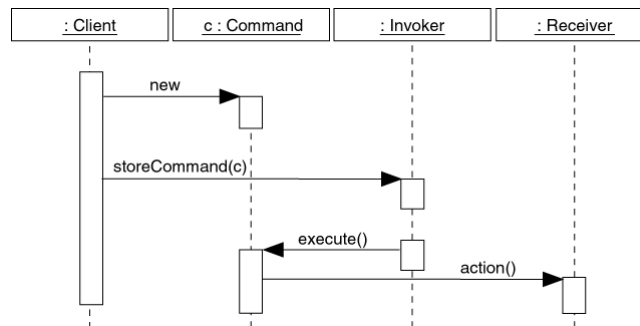
RATIONAL
SOFTWARE

Modeling a design pattern (cont.)



RATIONAL
SOFTWARE

Modeling a design pattern (cont.)



RATIONAL
SOFTWARE

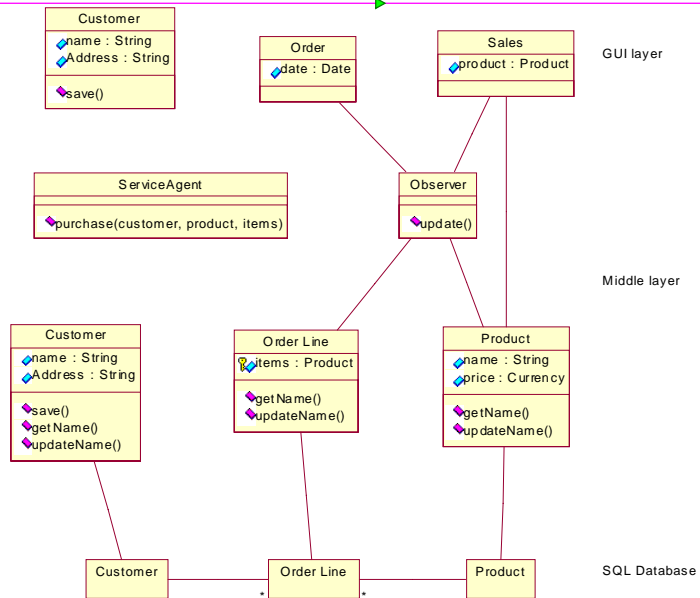
Architectural patterns

- Distributed
- Event-driven
- Frame-based
- Batch
- Pipes and filters
- Repository-centric
- Blackboard
- Interpreter
- Rule-based
- Layered
- MVC
- IR-centric
- Subsumption
- Disposable

Software Architecture
Shaw and Garlan
Buschmann et al
A System of Patterns
Buschman et al
Booch

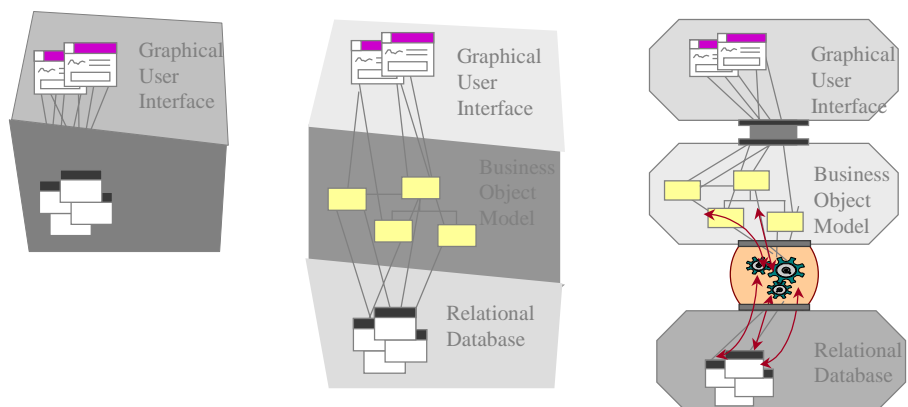
RATIONAL
SOFTWARE

Complex business system



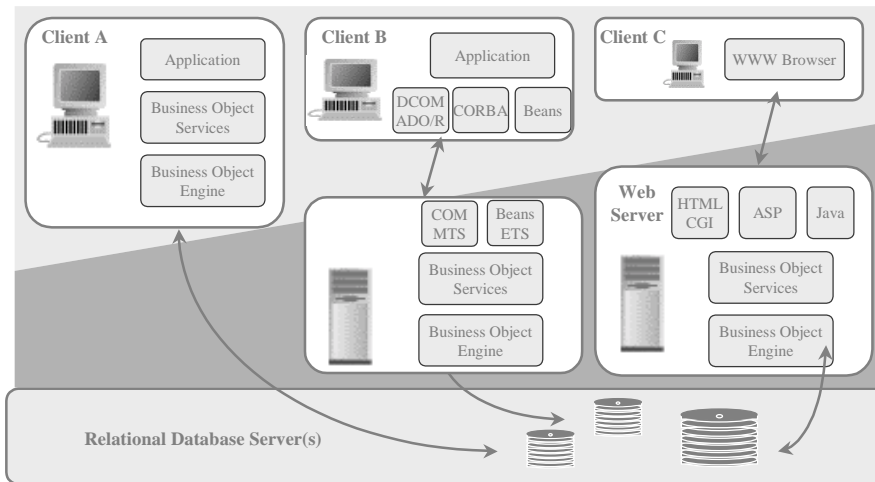
RATIONAL
SOFTWARE

Logical application architecture



RATIONAL
SOFTWARE

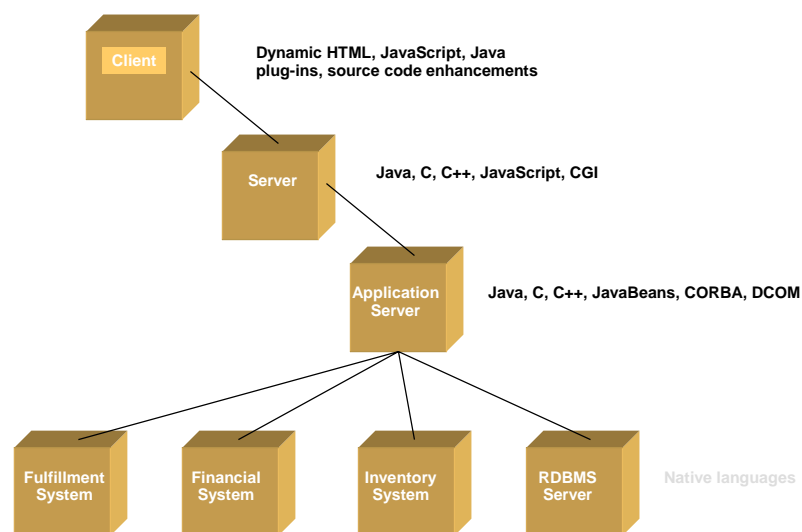
Physical application architecture



RATIONAL
SOFTWARE

Complex Internet system

The Second Wave
Paul Dreyfus, Netscape



RATIONAL
SOFTWARE

Who are the architects?

- Experience
 - software development
 - domain
- Pro-active, goal oriented
- Leadership, authority
- Architecture team
 - balance

RATIONAL
SOFTWARE

Architect

- Not just a top level designer
 - Need to ensure feasibility
- Not the project manager
 - But “joined at the hip”
- Not a technology expert
 - Purpose of the system, “fit”,
- Not a lone scientist
 - Communicator

RATIONAL
SOFTWARE

Software architecture team charter

- Defining the architecture of the software
- Maintaining the architectural integrity of the software
- Assessing technical risks related to the software design
- Proposing the order and contents of the successive iterations
- Consulting services
- Assisting marketing for future product definition
- Facilitating communications between project teams

RATIONAL
SOFTWARE

Architecture is making decisions

The life of a software architect is a long (and sometimes painful) succession of suboptimal decisions made partly in the dark

RATIONAL
SOFTWARE

Futures

- ADL: Architecture Description Languages
 - UML, UniCon, LILEAnna, P++, LEAP, Wright, μ Rapid
- Standardization of concepts
 - IEEE Working Group on Architecture
 - INCOSE Working Group on System Architecture
- Systematic capture of architectural patterns

RATIONAL
SOFTWARE

References (Architecture)

- Len Bass, Paul Clements & Rick Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.
- Frank Buschmann, Régine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stahl, *Pattern-Oriented Software Architecture - A System of Patterns*, Wiley and Sons, 1996.
- Christine Hofmeister, Robert Nord, Dilip Soni, *Applied Software Architecture*, Addison-Wesley 1999.
- Eric Gamma, John Vlissides, Richard Helm, Ralph Johnson, *Design Patterns*, Addison-Wesley 1995.
- Philippe Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*, 12 (6), November 1995, IEEE.
 - <http://www.rational.com/support/techpapers/ieee/>
- Eberhardt Rechtin, *Systems Architecting: Creating and Building Complex Systems*, Englewood Cliffs NJ, Prentice-Hall, 1991.

RATIONAL
SOFTWARE

References (Architecture)

- Eberhardt Rechtin & Mark Maier, *The Art of System Architecting*, CRC Press, 1997.
- *Recommended Practice for Architectural Description*, Draft 2.0 of IEEE P1471, May 1998
 - <http://www.pithecanthropus.com/~awg/>
- Mary Shaw, and David Garlan, *Software Architecture—Perspectives on an Emerging Discipline*, Upper Saddle River, NJ, Prentice-Hall, 1996.
- Bernard I. Witt, F. Terry Baker, and Everett W. Merritt, *Software Architecture and Design—Principles, Models, and Methods*, New York NY, Van Nostrand Reinhold, 1995.
- The World-wide Institute of Software Architects
 - <http://www.wwisa.org>

RATIONAL
SOFTWARE

References (UML)

- Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

RATIONAL
SOFTWARE

References (Process)

- Barry Boehm, "A spiral model of software development and enhancement," *IEEE Computer*, May 1998.
- Barry Boehm, "Anchoring the software process," *IEEE Software*, July 1996.
- Grady Booch, *Object Solutions*, Addison-Wesley, 1995.
- Philippe Kruchten, "A Rational Development Process," *CrossTalk*, July 1996.
 - <http://www.rational.com/support/techpapers/devprcs/>
- Philippe Kruchten, *The Rational Unified Process - An Introduction*, Addison-Wesley, 1999.
- *Rational Unified Process 5.0*, Rational, Cupertino, CA, 1998
- Walker Royce, *Software Project Management: a Unified Framework*, Addison-Wesley, 1998
- The Software Program Manager's Network
 - <http://www.spmn.com>

RATIONAL
SOFTWARE