Architecting Large-Scale Systems

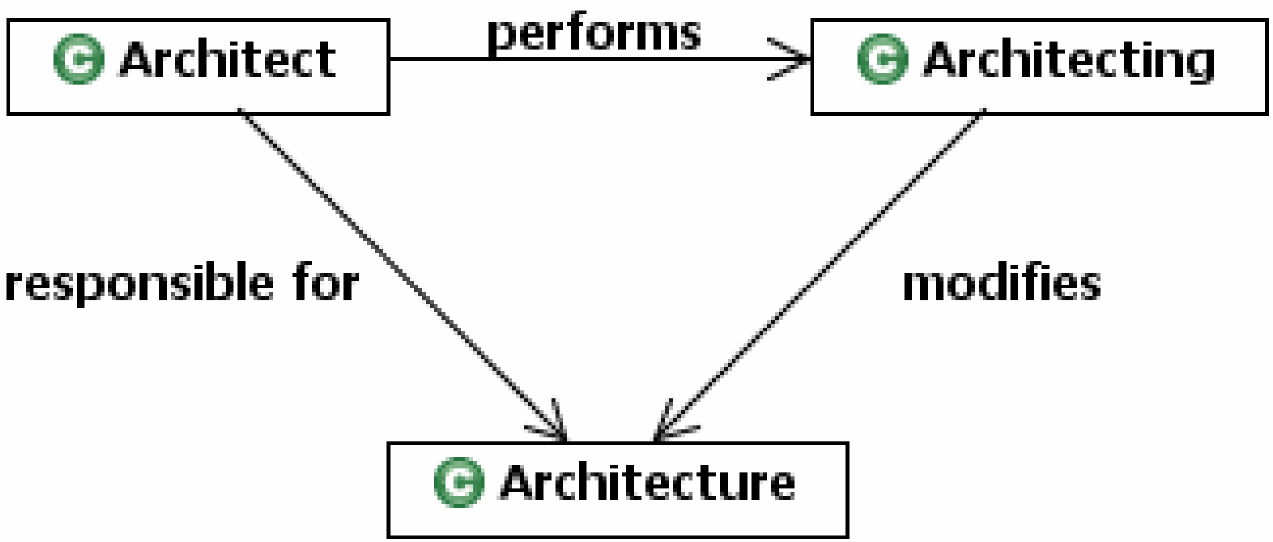
Peter Eeles
IBM Executive IT Architect, IBM
peter.eeles@uk.ibm.com

IBM Rational Software Development Conference UK 2007

What keeps me **Rational**?

TD2

Architecture, Architect, Architecting



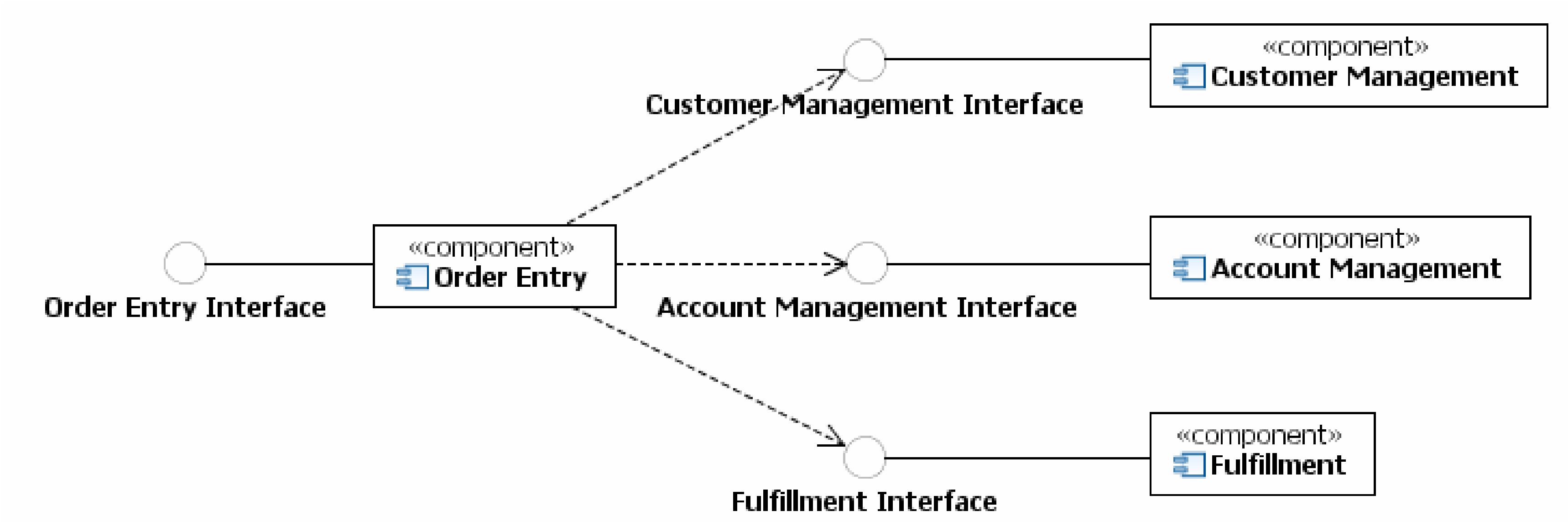
Agenda

- ➡ What are the characteristics of an Architecture?
- What are the characteristics of an Architect?
- What are the characteristics of Architecting?
- What are the benefits of Architecting?
- What is a large-scale system?
- A worked example
- Summary

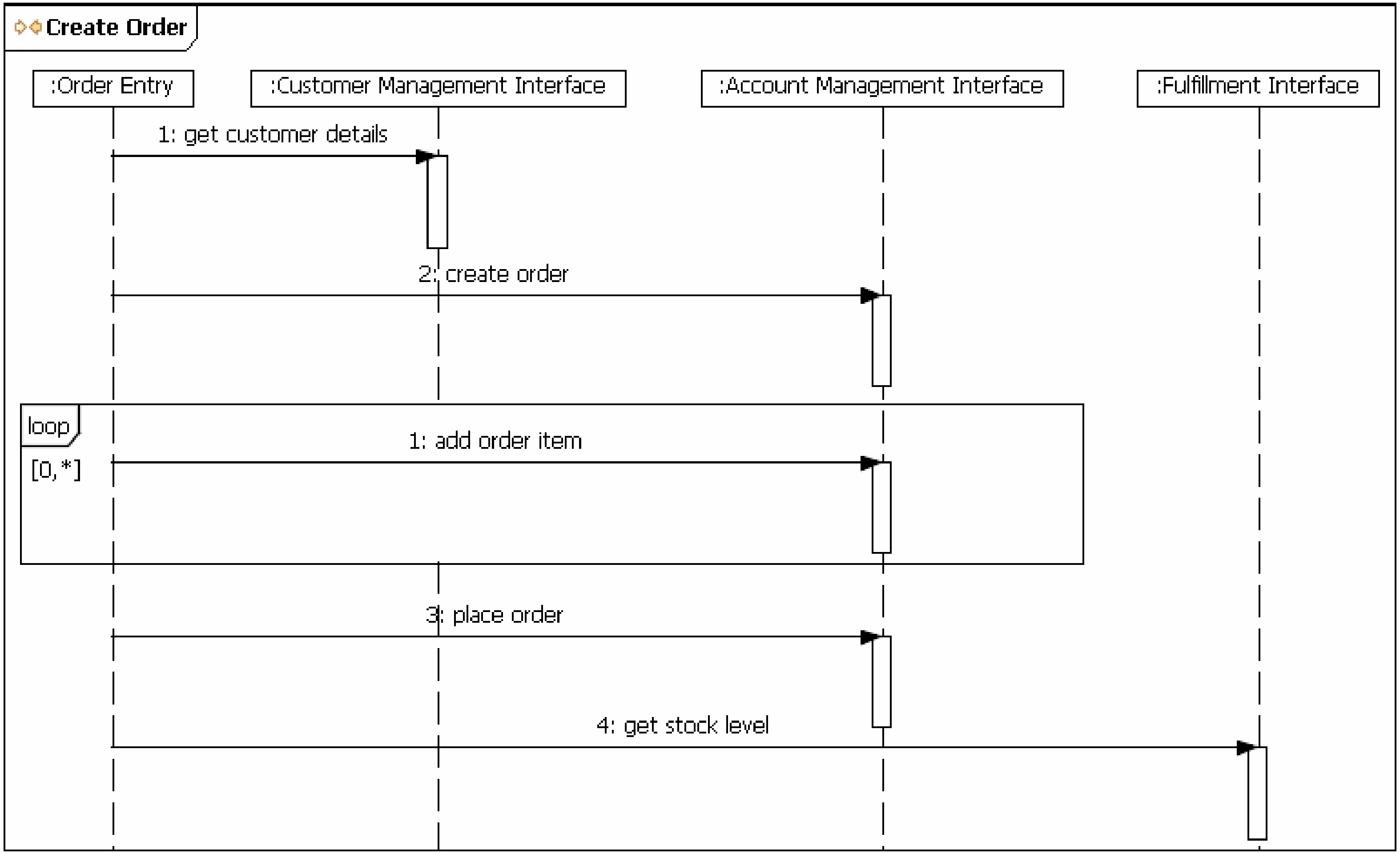
Architecture

- *Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. [IEEE 1471]*
- *The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. [Bass]*
- *[Architecture is] the organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems. [UML 1.5]*

An architecture defines structure



An architecture defines behaviour



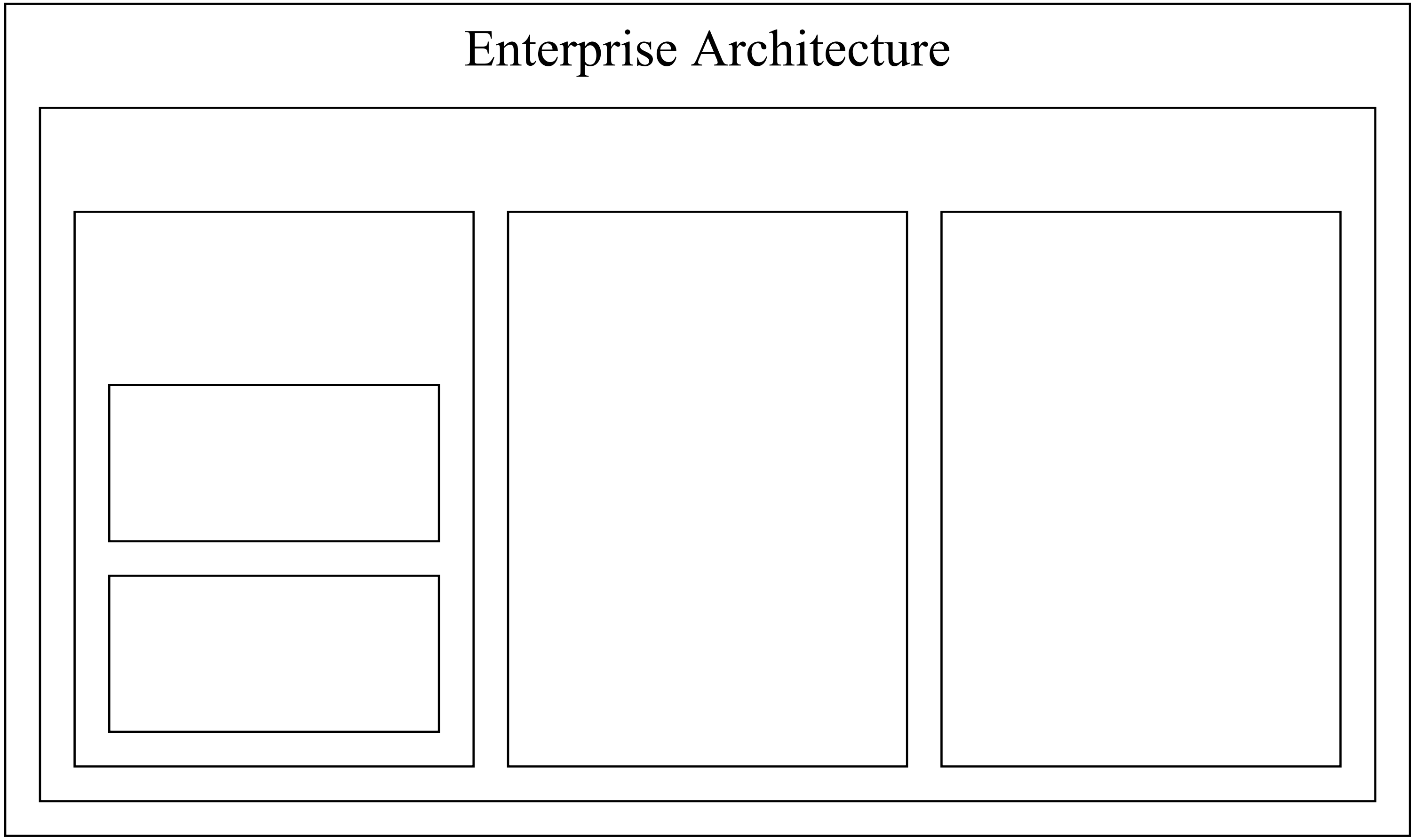
An architecture is concerned with significant elements

- The element relates to some critical functionality of the system
  - E.g. monetary transactions
- The element relates to some critical property of the system
  - E.g. reliability
- The element relates to a particular architectural challenge
  - E.g. external system integration
- The element is associated with a particular technical risk
- The element relates to a capability that is considered to be unstable
- The element relates to some key element of the solution
  - E.g. login mechanism

An architecture meets stakeholder needs

- The end user is concerned with intuitive and correct behavior, performance, reliability, usability, availability and security
- The system administrator is concerned with intuitive behavior, administration and tools to aid monitoring
- The marketer is concerned with competitive features, time to market, positioning with other products, and cost
- The customer is concerned with cost, stability and schedule
- The developer is concerned with clear requirements, and a simple and consistent design approach
- The project manager is concerned with predictability in the tracking of the project, schedule, productive use of resources and cost
- The maintainer is concerned with a comprehensible, consistent and documented design approach, and the ease with which modifications can be made

An architecture comes in many forms



And …

- An architecture embodies decisions based on rationale
- An architecture conforms to an architectural style
- An architecture is influenced by its environment
- An architecture influences organizational structure
- An architecture is present in every system

Agenda

- What are the characteristics of an Architecture?
- What are the characteristics of an Architect?
- What are the characteristics of Architecting?
- What are the benefits of Architecting?
- What is a large-scale system?
- A worked example
- Summary

Architect

- The architect is a technical leader
- The architect understands the software development process
- The architect has knowledge of the business domain
- The architect has technology knowledge
- The architect has design skills
- The architect has programming skills
- The architect is a good communicator
- The architect makes decisions
- The architect is a mentor
- The architect is aware of organizational politics
- The architect is a negotiator
- The architect role may be fulfilled by a team

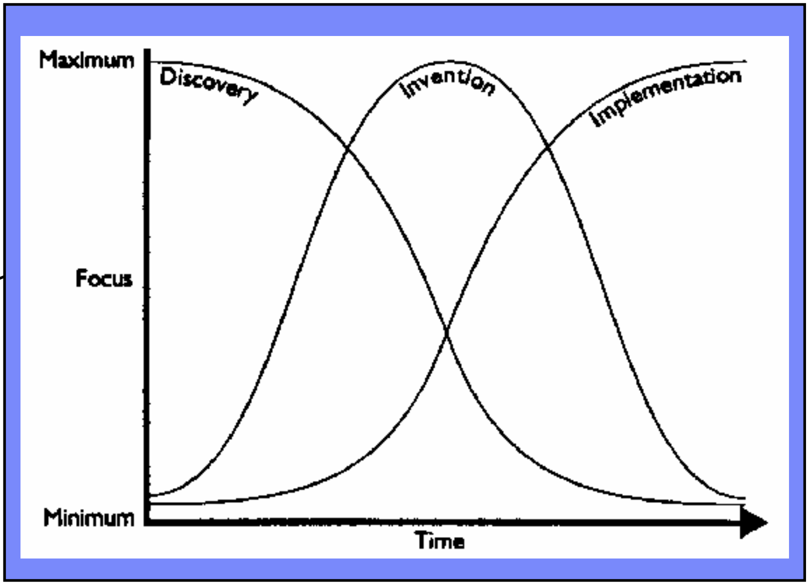
"The life of a software architect is a long and rapid succession of suboptimal design decisions taken partly in the dark." [Kruchten]

Agenda

- What are the characteristics of an Architecture?
- What are the characteristics of an Architect?
- What are the characteristics of Architecting?
- What are the benefits of Architecting?
- What is a large-scale system?
- A worked example
- Summary

Architecting

- Architecting is a science
- Architecting is an art
- Architecting spans many disciplines
- Architecting changes emphasis over time
- Architecting involves many stakeholders
- Architecting is involved in tradeoffs
- Architecting considers reusable assets
- Architecting is both top-down and bottom-up



Agenda

- What are the characteristics of an Architecture?
- What are the characteristics of an Architect?
- What are the characteristics of Architecting?
- ➡ What are the benefits of Architecting?
- What is a large-scale system?
- A worked example
- Summary

The benefits of architecting

- Architecting addresses system qualities
- Architecting drives consensus
- Architecting ensures architectural integrity
- Architecting helps manage complexity
- Architecting provides a basis for reuse
- Architecting reduces maintenance costs
- Architecting supports impact analysis
- Architecting supports the planning process

Agenda

- What are the characteristics of an Architecture?
- What are the characteristics of an Architect?
- What are the characteristics of Architecting?
- What are the benefits of Architecting?
- ➡ What is a large-scale system?
- A worked example
- Summary

Large-scale initiatives

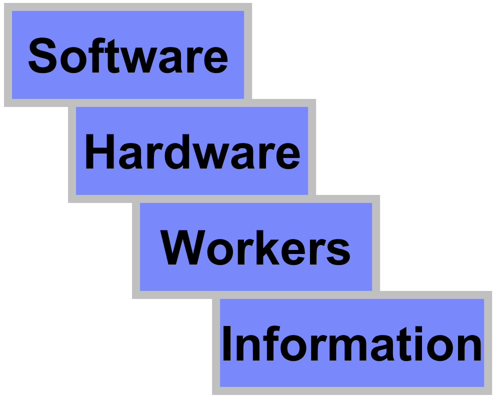
- Enterprise architecting
  - Defining an architecture that underpins a number of systems
- Strategic reuse
  - Developing reusable assets that are used within a number of systems
- Systems engineering
  - Developing a system that contains elements of hardware, software, workers and data
- Enterprise Application Integration
  - Developing a solution that includes the integration of a number of legacy systems
- Packaged application development
  - Developing a solution that includes the configuration of a packaged application, such as an ERP or CRM solution
- Outsourced development
  - Defining an architecture that lends itself to the outsourced development of its constituent parts, whilst ensuring the quality and integrity of these parts
- Service-Oriented Architecture
  - Supporting the creating of composite applications whose parts are reusable services

Enterprise, Business, System

- Enterprise
  - Set of resources that are used to meet a business need or mission
  - Enterprises can cross organization and even business boundaries
  - Enterprises provide value to their stakeholders (e.g. stockholders, community, nation, etc.)
- Business (Organization)
  - A part of an enterprise responsible for one or more business processes (may also be Business Unit, Segment, etc.)
- System
  - An entity consisting of hardware, software, workers and information ... that provides services used by an enterprise in meeting its purpose or mission

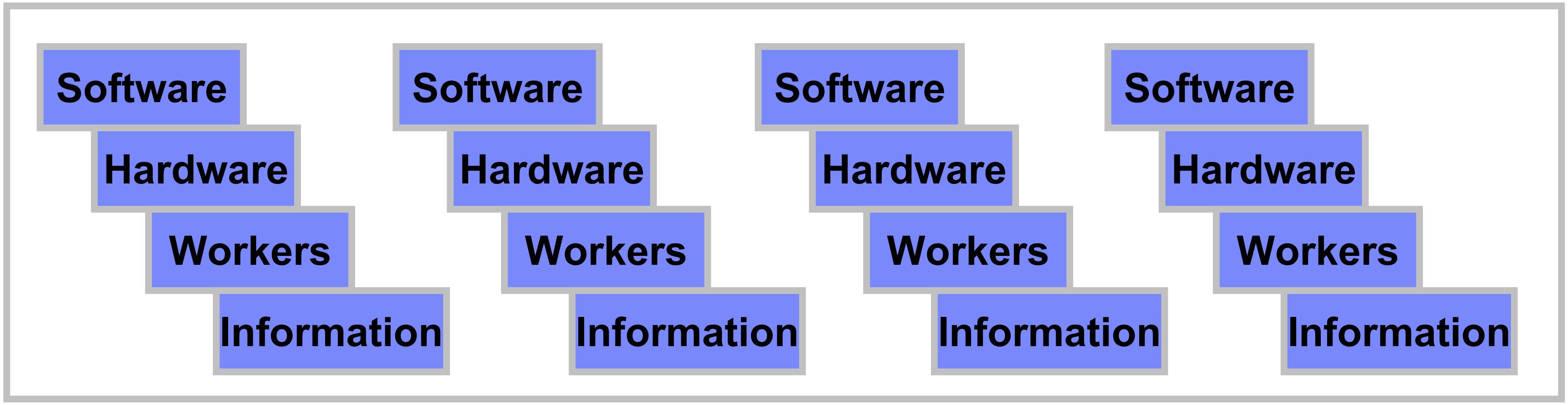
A System

- Made up of
  - Software
  - Hardware
  - Workers (people)
  - Information (data)

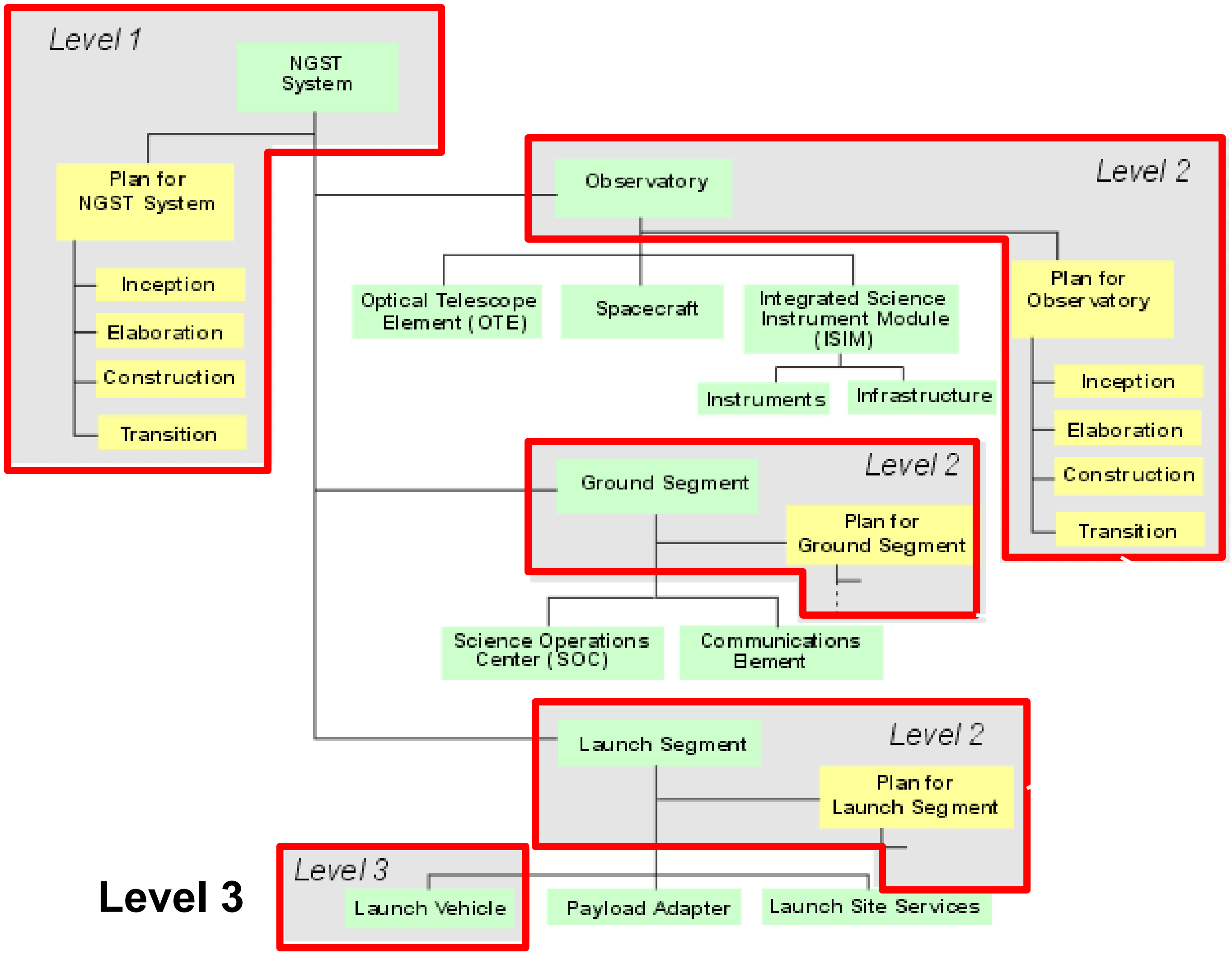


A System of Systems

- Consider a system to be made up of a collection of other systems, each made up of software, hardware, workers and information
  - A “system of systems”



An Example



Level 1

Level 2

Level 3

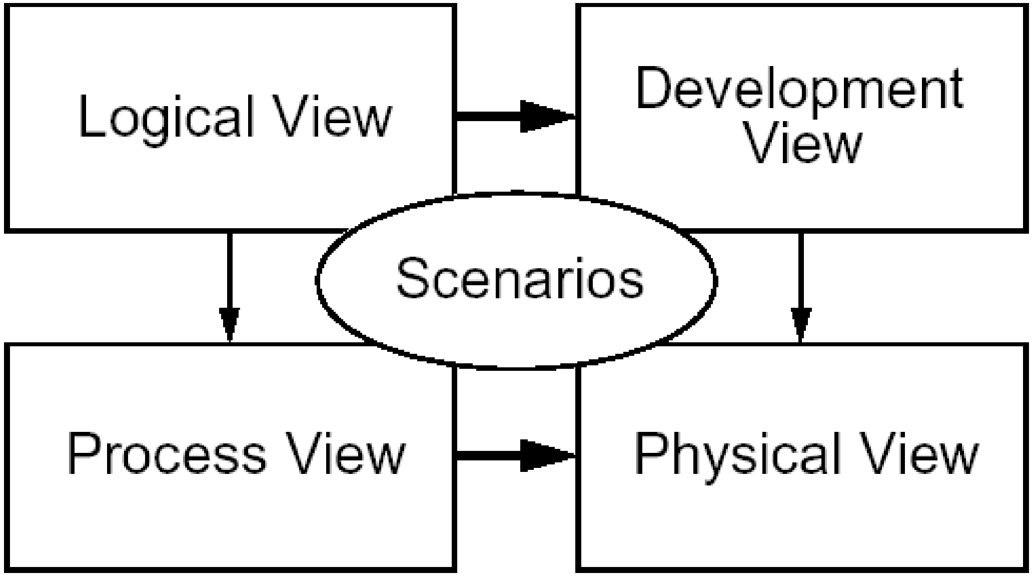
Applying the Pattern

- Enterprise Architecting
  - The decomposition of an enterprise into its respective elements can be expressed in terms of a “system of systems”
- Strategic Reuse
  - Reusable assets and their relationships can be described in terms of a “system of systems”
- Systems Engineering
  - The system as a whole can be expressed in terms of a superordinate system, and each of the elements that comprise the system can be expressed in terms of a subordinate system
- Enterprise Application Integration
  - The context within which a legacy system fits can be described in terms of a superordinate system, with the legacy system itself represented as a subordinate system
- Packaged Application Development
  - The packaged application may represent a subordinate system (if it is a “piece”) or a superordinate system (if it is a “whole”)
- Outsourced Development
  - The overall architecture can be described in terms of a superordinate system, with the constituent parts described in terms of subordinate systems
- Service-Oriented Architecture
  - The SOA itself can be described as a superordinate system, and the composite applications and services described as subordinate systems

Architectural Representation

- IEEE-1471
  - The IEEE Recommended Practice for Architectural Description of Software-Intensive Systems
  - This standard provides a conceptual framework for architectural description and defines what is meant by a 1471-compliant architectural description
- 4 + 1 Views of Software Architecture
- Siemens
- DoDAF
- MoDAF
- ToGAF
- RM-ODP
- The Zachman Framework
- RUP for Systems Engineering (RUP-SE)
- …

Describing an Architecture – Kruchten 4+1 views



Describing an Architecture – Zachman framework

Abstractions Perspectives	Data	Function	Network	People	Time	Motivation
Scope Planner contextual						
Enterprise Model Owner conceptual						
System Model Designer logical						
Technology Constrained Model Builder physical						
Detailed Representations Subcontractor out-of-context						
Functioning Enterprise						

Describing an Architecture – Cantor (RUP-SE)

Viewpoint Level	Worker	Logical	Information	Physical	Process
Context					
Analysis					
Design					
Implementation					

Agenda

- What are the characteristics of an Architecture?
- What are the characteristics of an Architect?
- What are the characteristics of Architecting?
- What are the benefits of Architecting?
- What is a large-scale system?

➡ A worked example

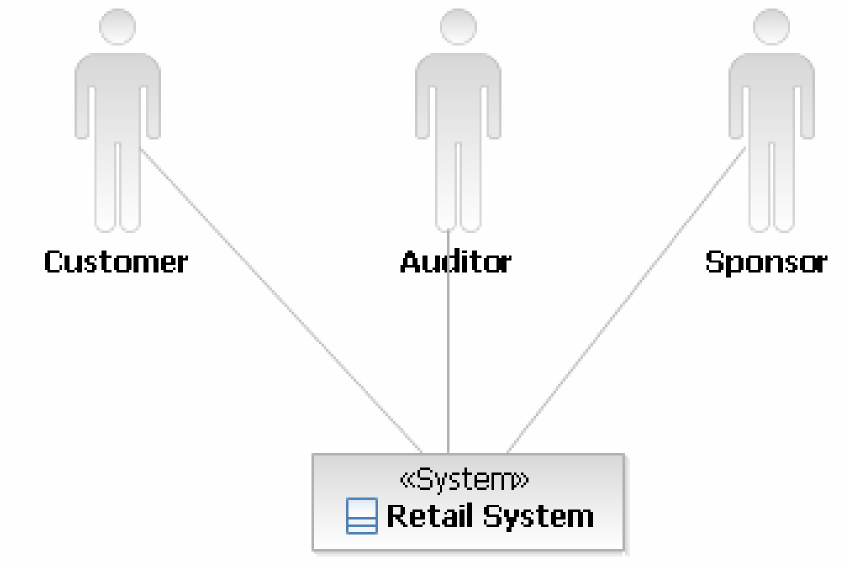
- Summary

An example

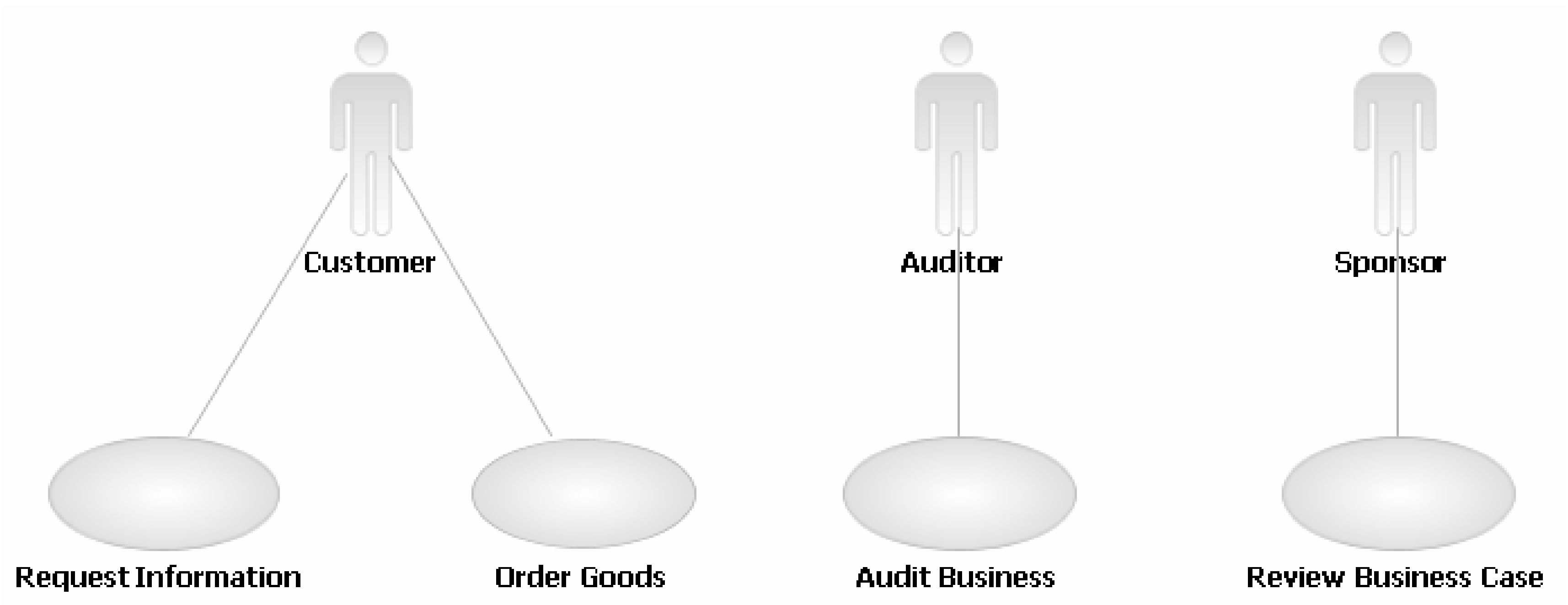
- A retail store
- Selling books, videos, DVDs, music CDs, etc.

Is a sales clerk inside or outside the system?

Level 1: Context Diagram (initial)



Level 1: Use-Case Model



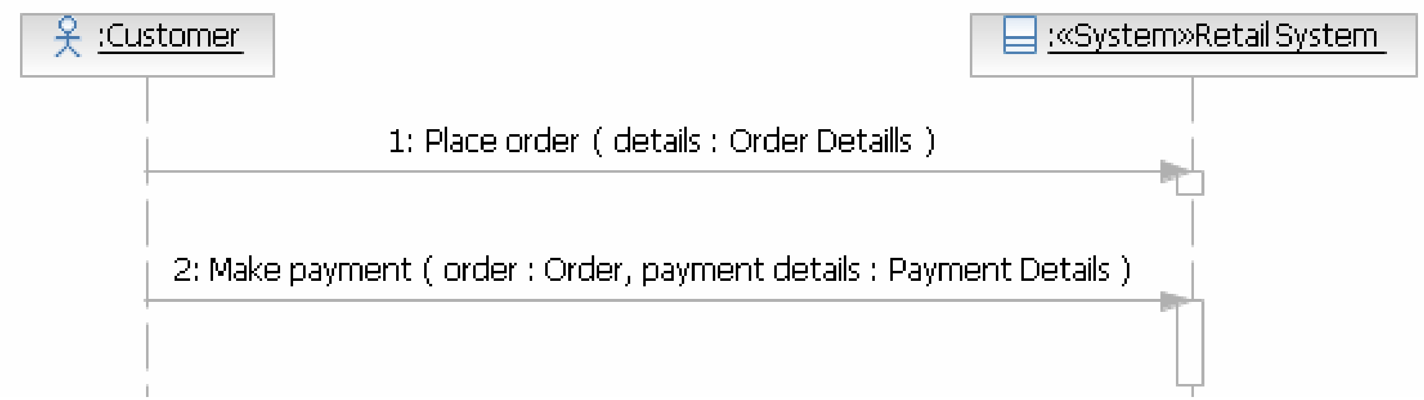
Level 1: Use-Case Model

- Basic Flow of the “Order Goods” Business Use Case

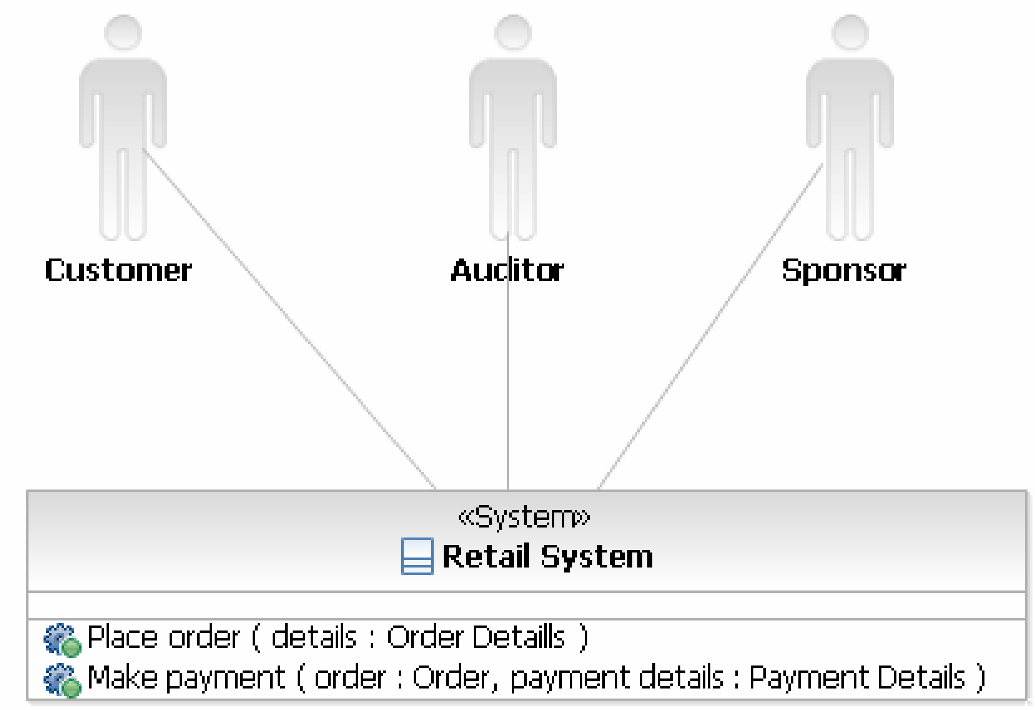
1. The use case starts when the Customer initiates the placing of an Order for Products.
2. An appropriate Order is placed that contains the Products to be purchased, along with the relevant quantity of each Product. The Customer receives the ordered Products and a request for payment.
3. The Customer pays for the Order.
4. The use case ends.

Level 1: Use-Case Model

- Basic Flow of the “Order Goods” Business Use Case
- The system is treated as a “black box”
  - *How* the order is fulfilled and payment requested is *internal to the system*

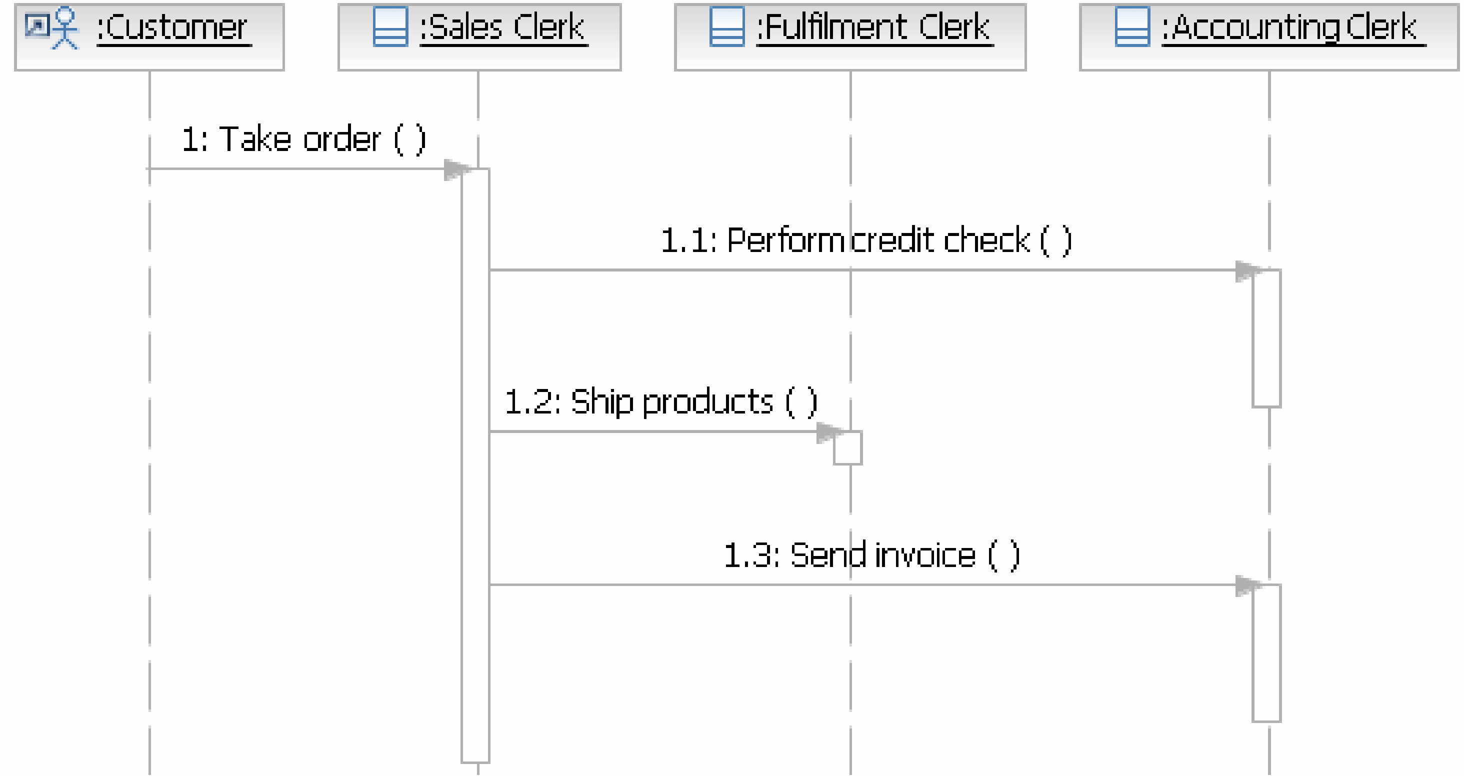

:Customer
:«System»Retail System
1: Place order (details : Order Detaills)
2: Make payment (order : Order, payment details : Payment Details)

Level 1: Context Diagram (partial)


Customer
Auditor
Sponsor
«System»
Retail System
Place order (details : Order Detaills)
Make payment (order : Order, payment details : Payment Details)

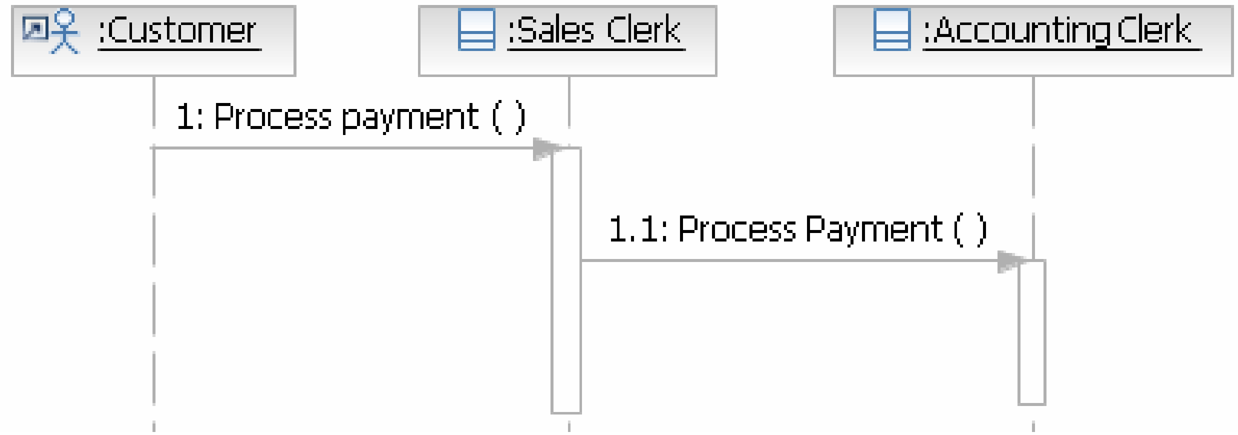
Level 1: Operation Realization

- For “Place order” operation
- The system is treated as a “white box”

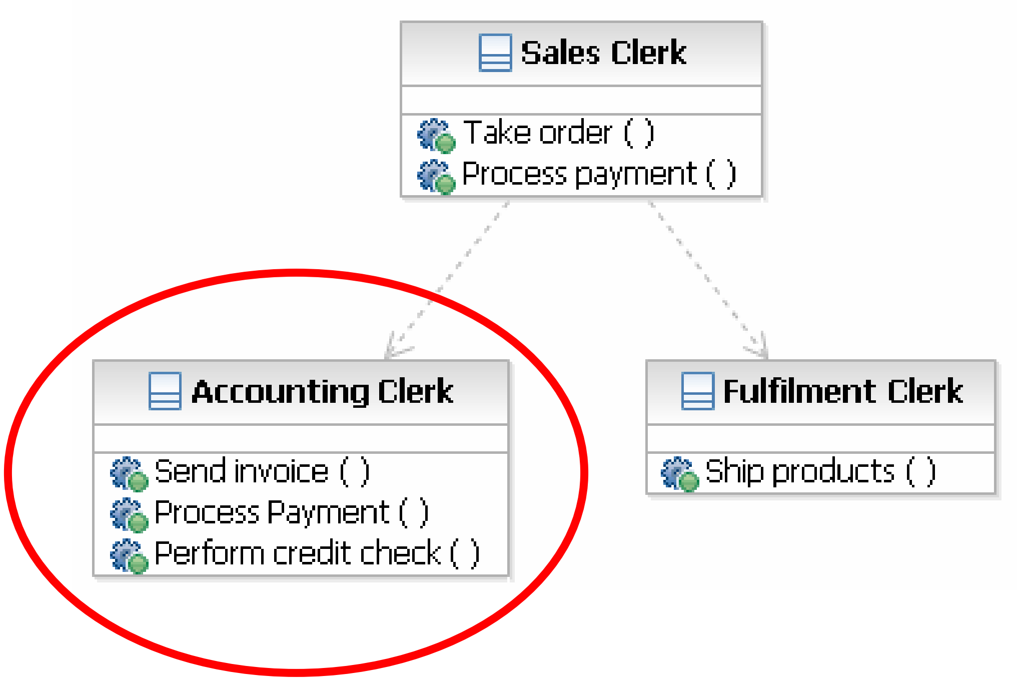


Level 1: Operation Realization

- For “Make Payment” operation



From Level 1 to Level 2



Sales Clerk

Take order ( )
Process payment ( )

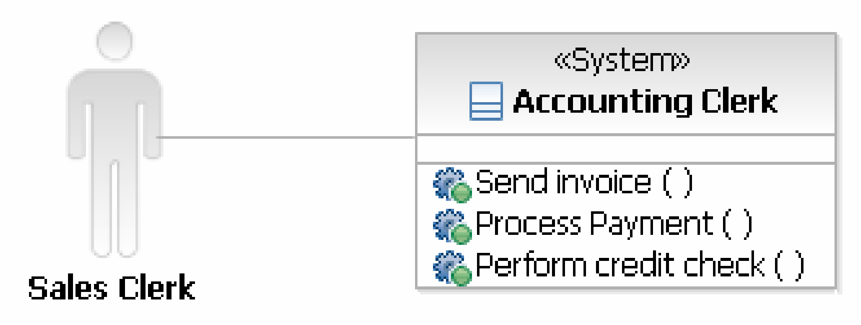
Accounting Clerk

Send invoice ( )
Process Payment ( )
Perform credit check ( )

Fulfilment Clerk

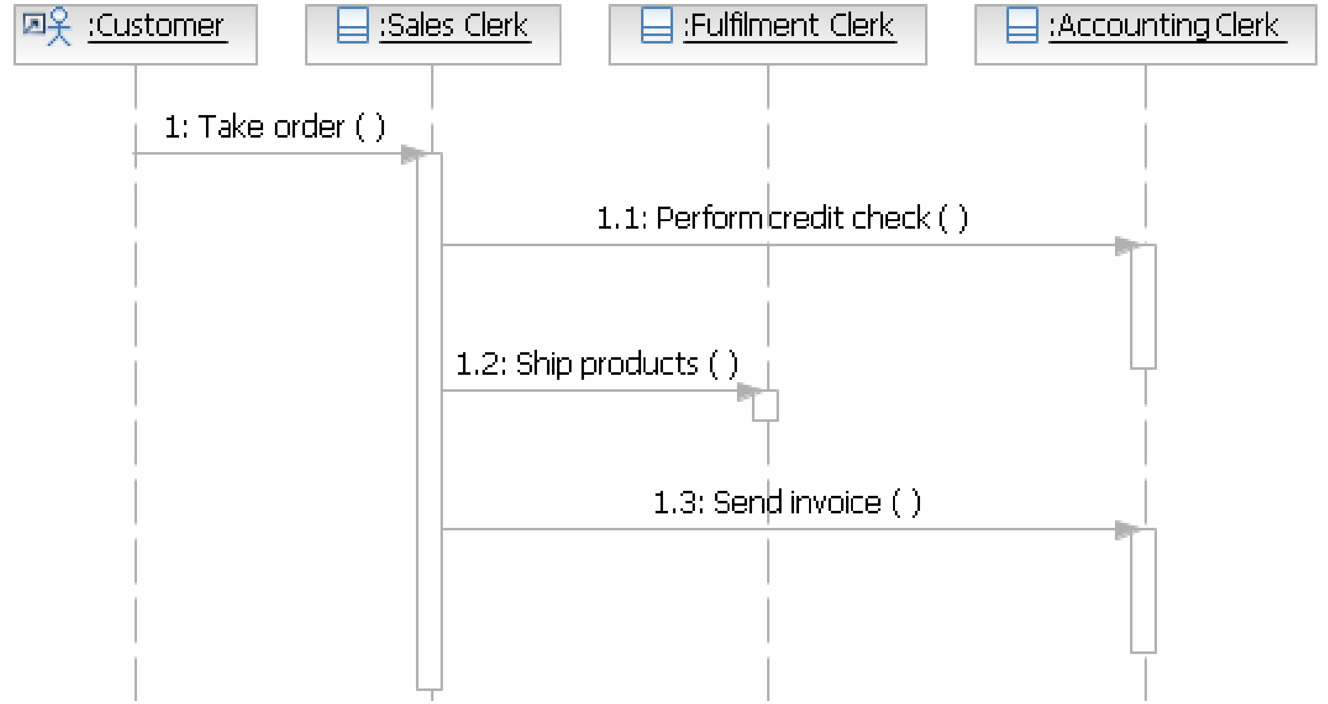
Ship products ( )

Level 2: Context Diagram

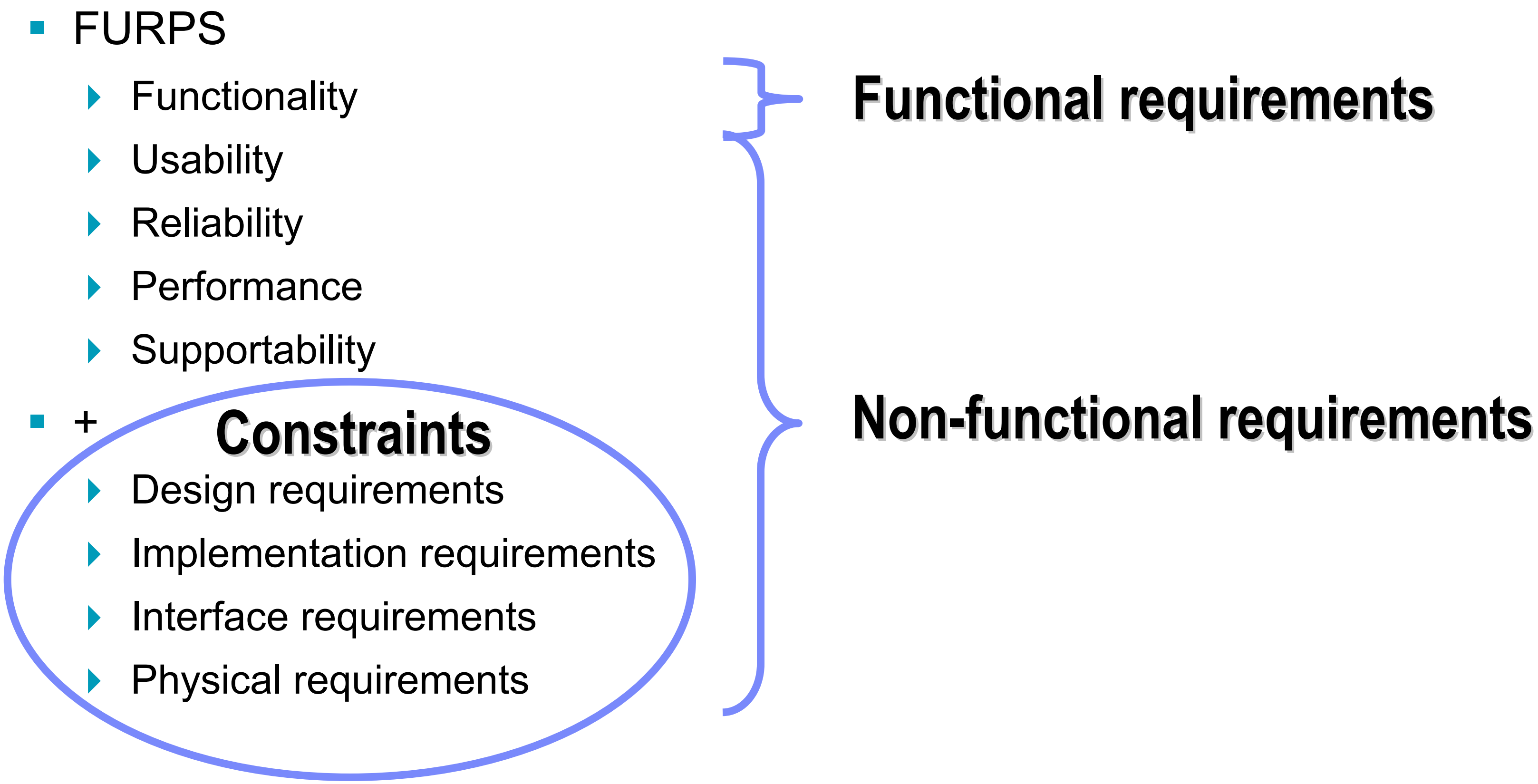


Level 1: Operation Realization

- What about non-functional requirements?
- What about other viewpoints (other than logical or worker)?



Classifying Requirements with “FURPS+”



- FURPS
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability
- + Constraints
  - Design requirements
  - Implementation requirements
  - Interface requirements
  - Physical requirements

Functional requirements

Non-functional requirements

*The FURPS classification was devised by Robert Grady at Hewlett-Packard

"FURPS+" - Functionality

- All functional requirements
- Usually represent main product features
  - E.g. Order Processing requirements
- Can also be architecturally significant
  - Auditing, Licensing, Localization, Mail, Online help, Printing, Reporting, Security, System management, Workflow

Can software (alone) satisfy these requirements?

“FURPS+”

- Usability
  - User interface issues such as accessibility, aesthetics and consistency
- Reliability
  - Availability, accuracy, recoverability
- Performance
  - Throughput, response time, recovery time, start-up time
- Supportability
  - Testability, adaptability, maintainability, compatibility, configurability, installability, scalability and localizability

Can software (alone) satisfy these requirements?

“FURPS+”

- Design requirement
  - Constrains the design
  - E.g. a relational database is required
- Implementation requirement
  - Constrains the coding or construction
  - E.g. required standards, platform or implementation language
- Interface requirement
  - A requirement to interact with an external item
- Physical requirement
  - A physical constraint imposed on the hardware used to house the system; for example, shape, size and weight

Can software (alone) satisfy these requirements?

Describing an Architecture – Cantor (RUP-SE)

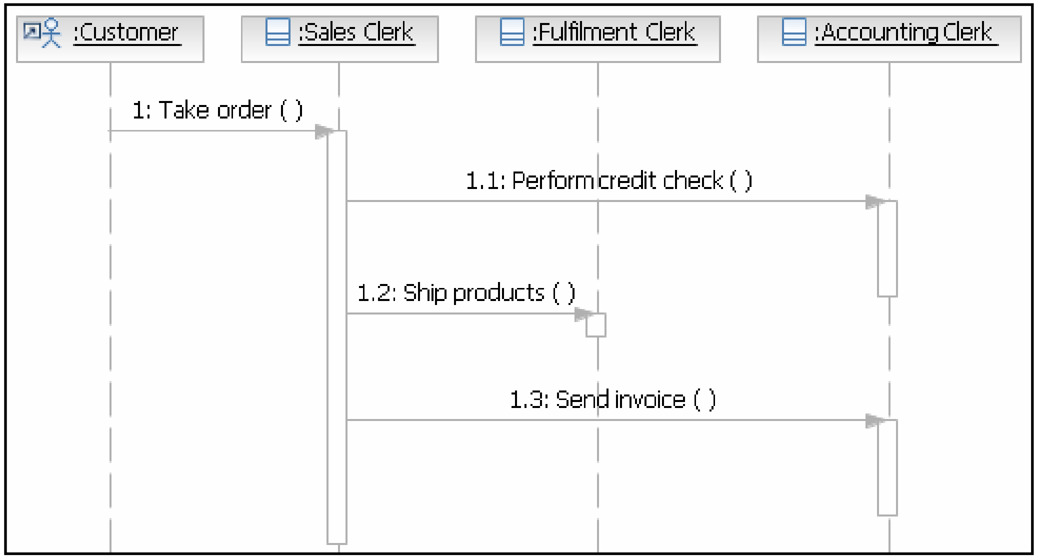
Viewpoint Level	Worker	Logical	Information	Physical	Process
Context					
Analysis	Subsystem	Subsystem		Locality	
Design					
Implementation					

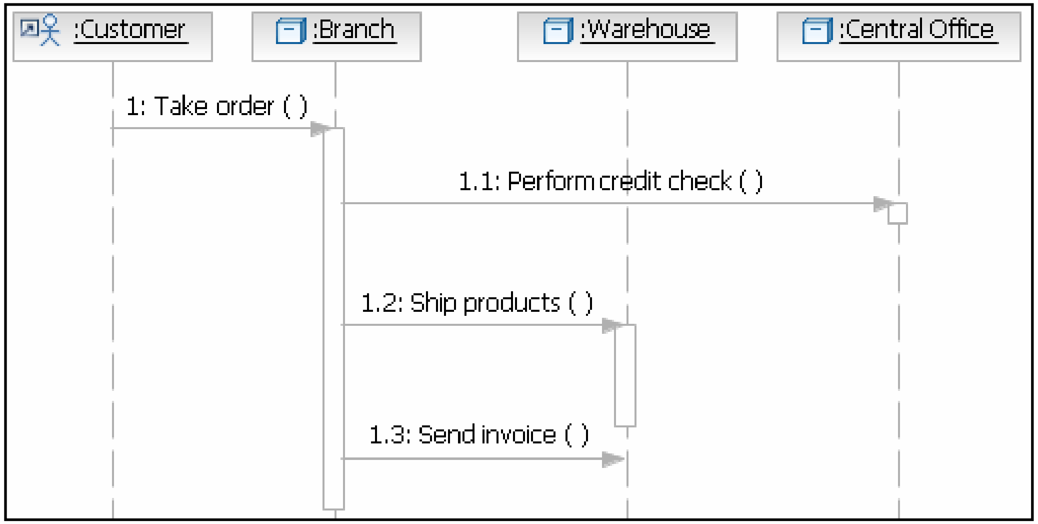
Level 1: Operation Realization

- For “Place order” operation
- This is “Joint realization” across different viewpoints (logical, worker, physical)

Step	Action Performed	Subsystem	Locality	Budgeted Requirements
1	The order details are taken	Sales Clerk	Branch	60 seconds
2	A credit check is performed	Accounting Clerk	Central Office	10 seconds
3	The products are shipped to the customer	Fulfilment Clerk	Warehouse	1 day
4	An invoice is sent to the customer	Accounting Clerk	Warehouse	1 day

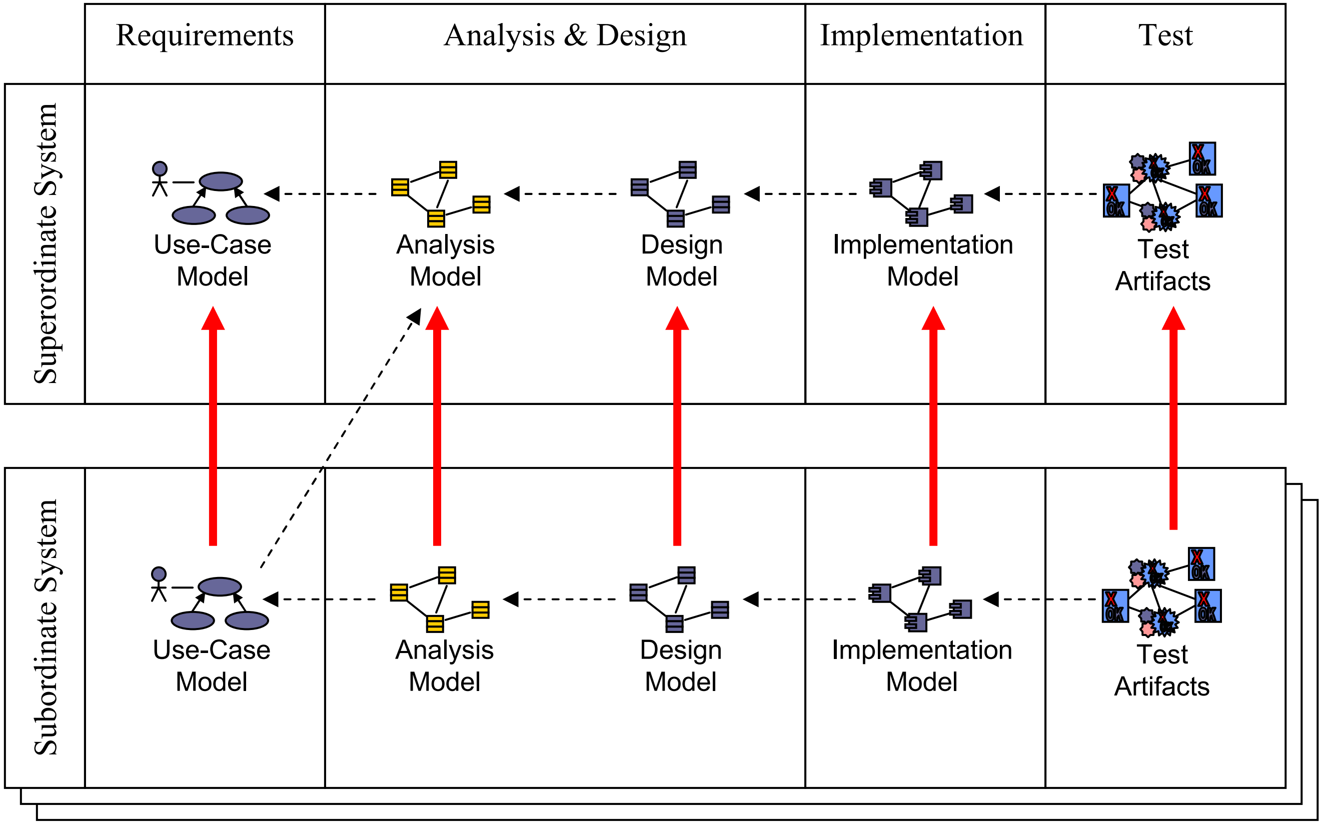
Joint Realization





The “System of Interconnected Systems” Pattern

- An example using the Rational Unified Process



Distributed Development / Outsourcing

- Stay in control ☺
- “Subcontractors” (internal or outsourced) must be managed
- Requires a project management contract
- Requires an architectural contract
- Requires a “joined up” development environment
  - Processes, tools, training, assets, …

Programme / Project Governance

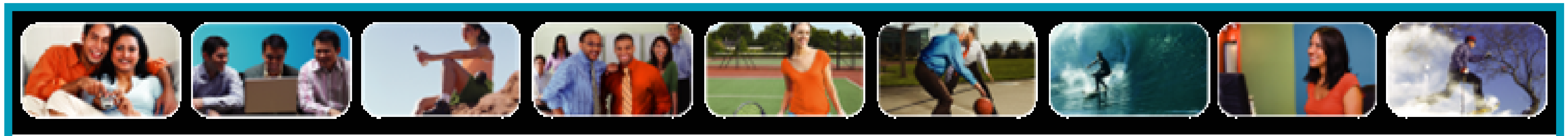
- Programme concerns
  - Alignment of projects within a programme
- Alignment of project management work products
  - Programme / project vision
  - Programme / project plans (schedules, budgets, signoff points, funding, releases)
- Alignment of project management processes
  - Scope (requirements) management
  - Change management
  - Test management
  - Risk and issues management
  - Quality management
  - Measurement / metrics gathering
  - Programme / project management reviews
  - Configuration management
  - etc.

Architectural (Solution) Governance

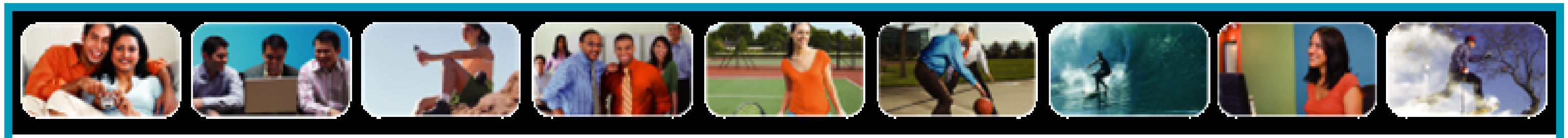
- Architectural concerns
  - Alignment of subordinate systems with the superordinate system
- Alignment of architectural work products
  - Requirements model
  - Design model
  - Implementation model
  - Data model
  - Standards and guidelines
  - Infrastructure definition
- Alignment of architectural processes
  - Identification / refinement of interfaces
  - Identification / refinement of architectural components
  - Identification / refinement of architectural component properties (cost, performance)
  - Architecture reviews
  - etc.

Summary

- "Systems" thinking requires us to think beyond software
  - Systems engineering, enterprise architecture, strategic reuse, …
- Certain qualities cannot be achieved by software alone
  - Performance, reliability, …
- Software/systems engineering principles and practices can scale to support the architecting of large-scale systems
- The "system of interconnected systems" pattern provides a means of managing complexity within such initiatives



Questions



Thank You

Peter Eeles (peter.eeles@uk.ibm.com)