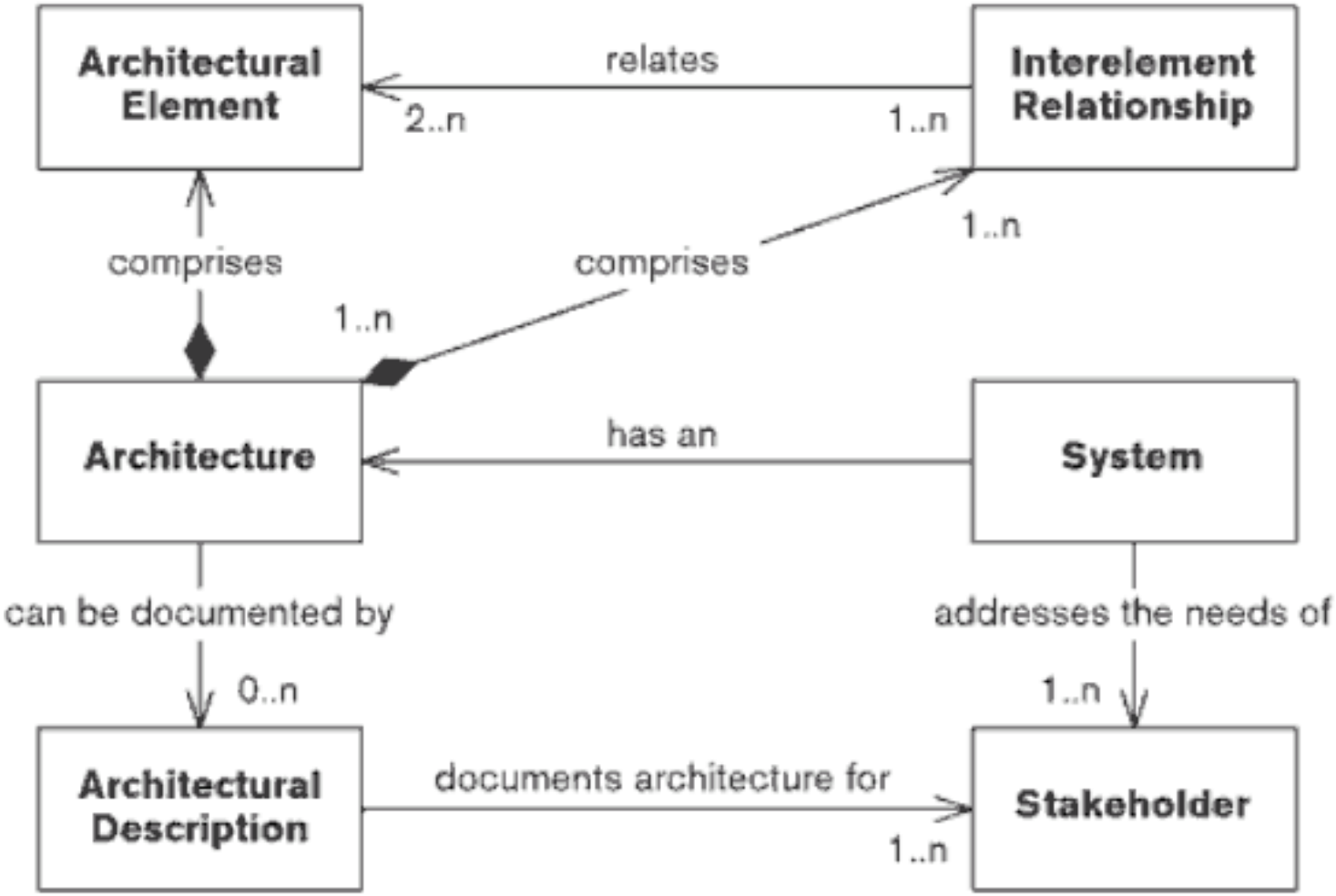
Views and Perspectives

Tomado de
Rozanski Nick and Woods Eoin
Software System Architecture
Preparado por Gilberto Pedraza García
2012

Core Concepts



Concepts (1)

- When we refer to a computer **system**, we mean the software elements that you need to specify and/or design in order to meet a particular set of requirements and the hardware that you need to run those software elements on.

Concepts (2)

- The **architecture** of a system is the set of fundamental concepts or properties of the system in its environment, embodied in its elements, relationships, and the principles of its design and evolution.
- The **static structures** of a system define its internal design-time elements and their arrangement.
- The **dynamic structures** of a system define its runtime elements and their interactions.

Concepts (3)

- The **externally visible behavior** of a system defines the functional interactions between the system and its environment.
- A **quality property** is an externally visible, nonfunctional property of a system such as performance, security, or scalability.
- A **candidate architecture** for a system is a particular arrangement of static and dynamic structures that has the potential to exhibit the system's required externally visible behaviors and quality properties.

Concepts (4)

- Every system has an architecture, whether or not it is documented and understood.
- An **architectural element** (or just element) is a fundamental piece from which a system can be considered to be constructed.
- A **stakeholder** in the architecture of a system is an individual, team, organization, or classes thereof, having an interest in the realization of the system.
- A **concern** about an architecture is a requirement, an objective, a constraint, an intention, or an aspiration a stakeholder has for that architecture.

Concepts and Principles

- Architectures are created solely to meet stakeholder needs.
- A good architecture is one that successfully addresses the concerns of its stakeholders and, when those concerns are in conflict, balances them in a way that is acceptable to the stakeholders.

Architectural Description

- An **architectural description (AD)** is a set of products that documents an architecture in a way its stakeholders can understand and demonstrates that the architecture has met their concerns.
- Although every system has an architecture, not every system has an architecture that is effectively communicated via an architectural description.
- A good architectural description is one that effectively and consistently communicates the key aspects of the architecture to the appropriate stakeholders.

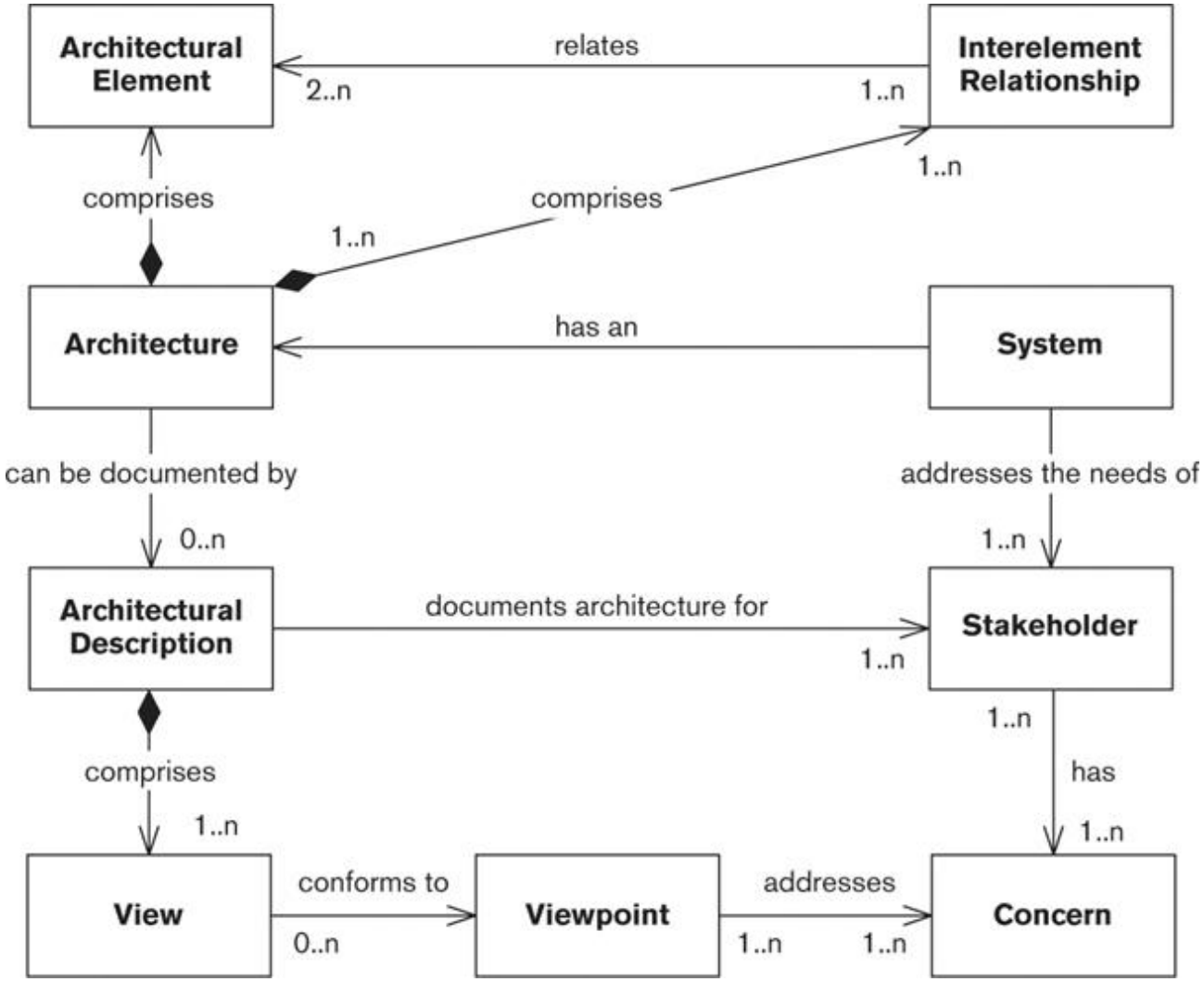
Concepts and Principles

- It is not possible to capture the functional features and quality properties of a complex system in a single comprehensible model that is understandable by, and of value to, its stakeholders.
- A complex system is much more effectively described by a set of interrelated views, which collectively illustrate its functional features and quality properties and demonstrate that it meets its goals, than by a single overloaded model.
- A view is a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders
- Only include in a view information that furthers the objectives of your AD—that is, information that helps explain the architecture to stakeholders or demonstrates that the goals of the system (i.e., the concerns of its stakeholders) are being met.

Viewpoint

- A **viewpoint** is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views.
- When developing a view, whether or not you use a formally defined viewpoint, be clear in your own mind what sorts of concerns the view is addressing, what types of architectural elements it presents, and who the viewpoint is aimed at. Make sure that your stakeholders understand these as well.

Views and Viewpoints



Benefits of Using Viewpoints and Views

- Separation of concerns
- Communication with stakeholder groups
- Management of complexity
- Improved developer focus

Viewpoint Pitfalls

- Inconsistency
- Selection of the wrong set of views
- Fragmentation

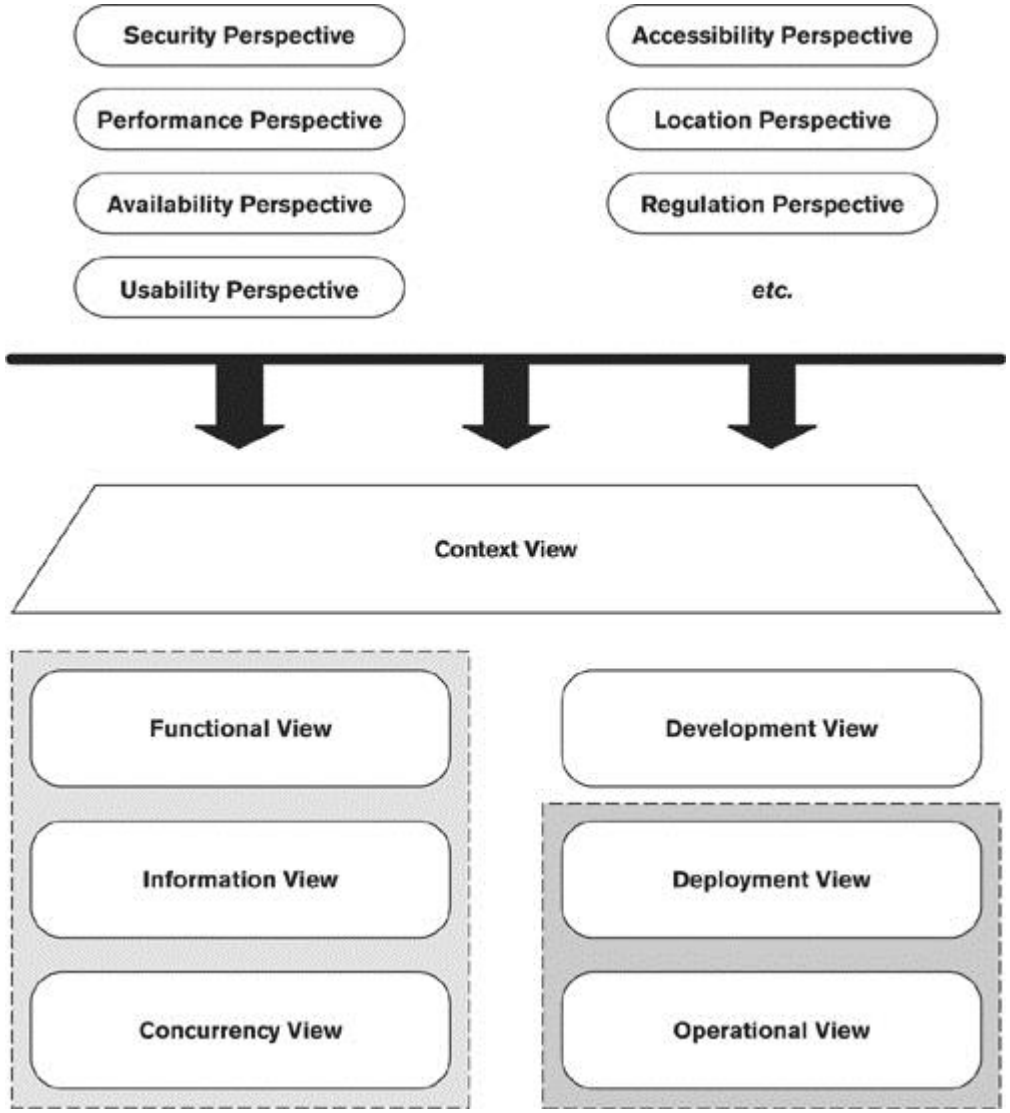
Viewpoint Grouping

- The Context viewpoint describes the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts).
- The Functional, Information, and Concurrency viewpoints characterize the fundamental organization of the system.
- The Development viewpoint exists to support the system's construction.
- The Deployment and Operational viewpoints characterize the system once in its live environment.

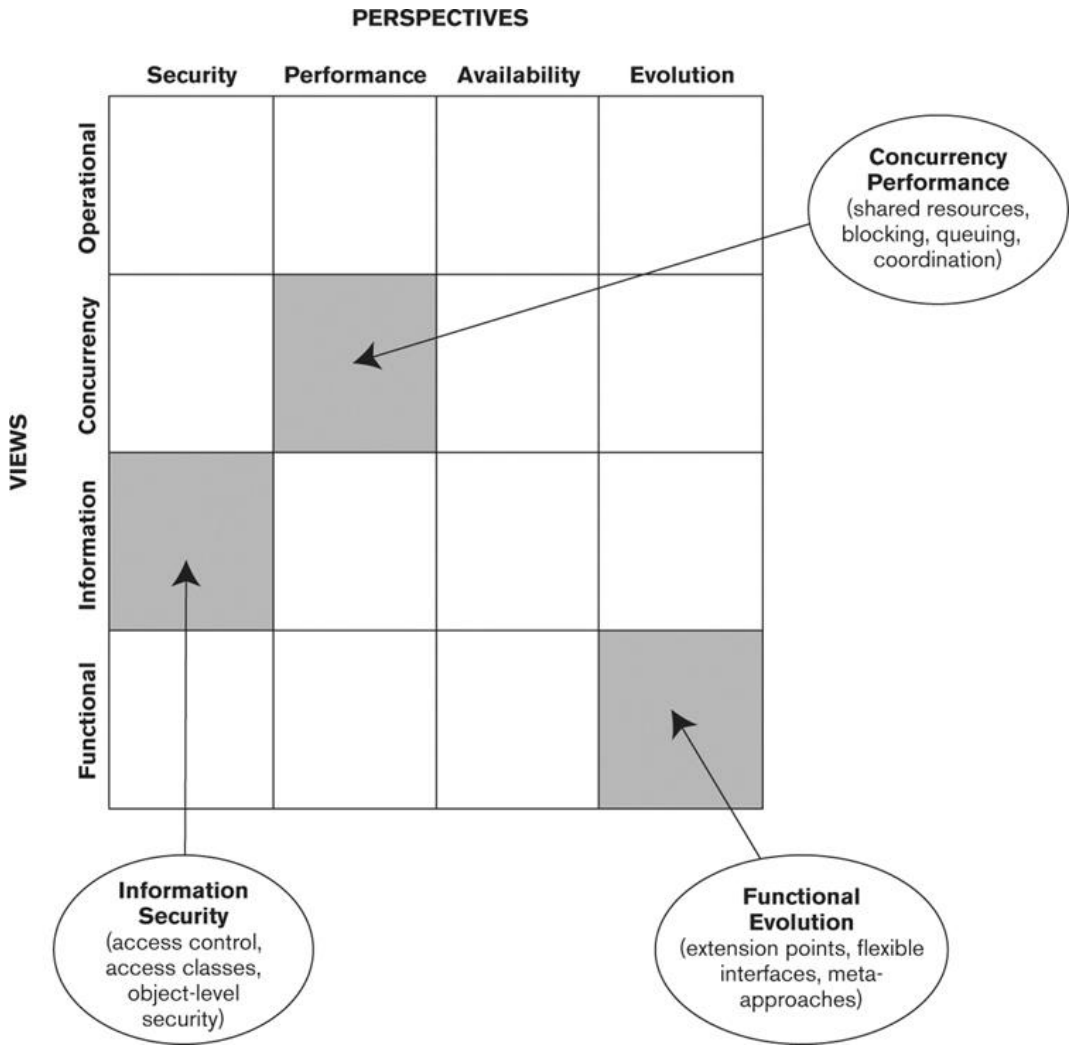
Architectural Perspective

- An **architectural perspective** is a collection of architectural activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views.
- An **architectural tactic** is an established and proven approach you can use to help achieve a particular quality property.

Applying Perspectives to Views



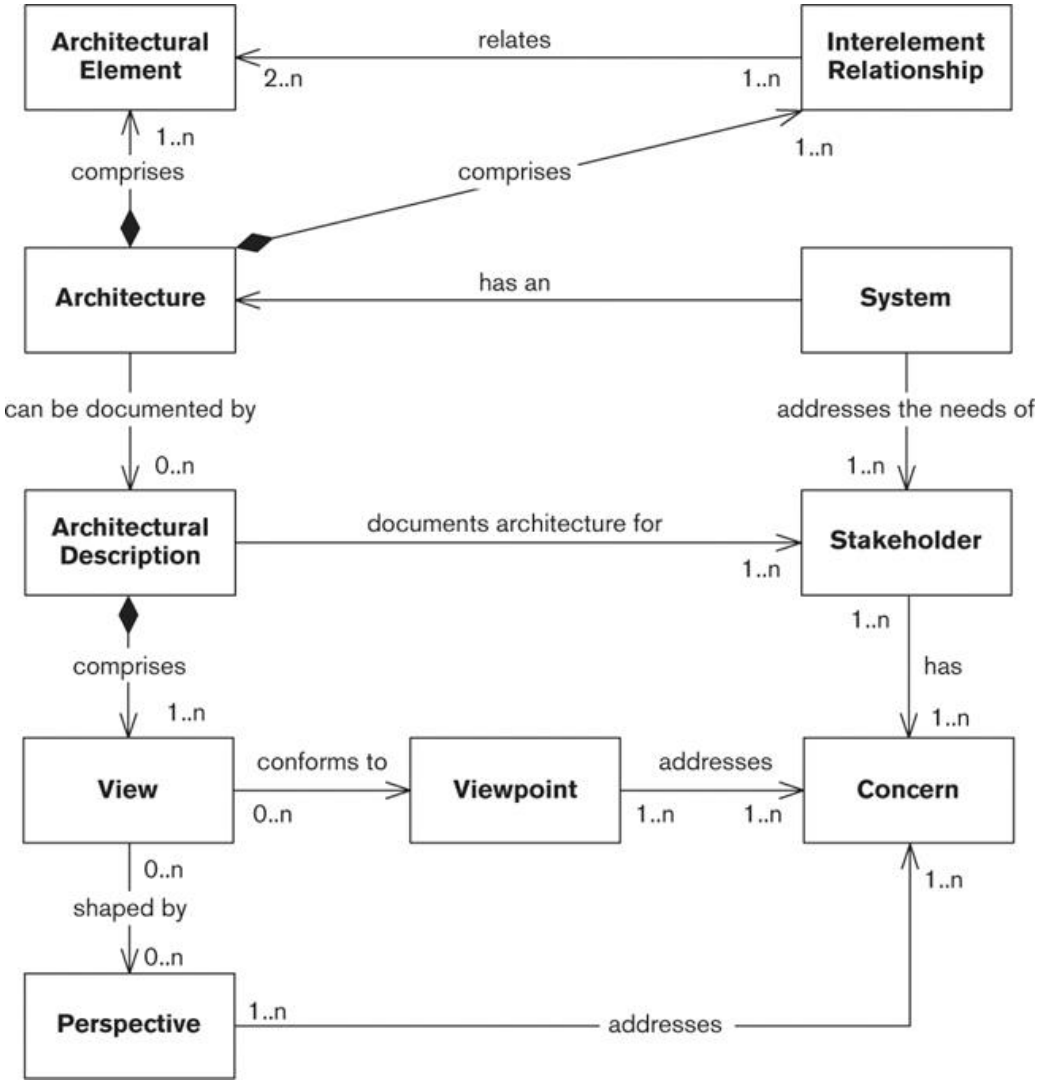
Examples of Applying Perspectives to Views



Consequences of Applying a Perspective

- **Insights**
  - Applying a perspective almost always leads to the creation of something—usually some sort of model—that provides an insight into the system's ability to meet a required quality property.
- **Improvements**
  - If applying the perspective tells you that the architecture will not meet one of its quality properties, the architecture needs to be improved.
- **Artifacts**
  - Some of the models and other deliverables created as a result of applying a perspective will be of only passing interest and will probably be discarded once the insight or improvement they reveal is understood.

Perspectives in Context



View

- A *view* is a representation of all (or part of) an architecture—that is, a way to document its architecturally significant features according to a related set of concerns.
- A view captures a description of one or more of the architectural structures of the system.
- Architects use views to explain the architectural structure of the system to stakeholders and to demonstrate that the architecture will meet their concerns.
- A view comprises a set of tangible architectural products, such as principles and models; the complete set of views of an architecture forms the AD.

Views, Viewpoints and Perspectives

- A *viewpoint* guides the process of creating a particular type of view. A viewpoint defines the concerns addressed by the view and the approach for creating and describing that aspect of the architecture.
- A *perspective* guides the process of design so that the system will exhibit one or more important *qualities*.
- As such, a perspective can be considered analogous to a viewpoint, but for a related set of quality properties rather than a type of architectural structure.
- We also use perspectives as a means of capturing common problems and pitfalls and identifying solutions to them.

Perspective Catalog

Perspective	Desired Quality
Accessibility	The ability of the system to be used by people with disabilities
Availability and Resilience	The ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability
Development Resource	The ability of the system to be designed, built, deployed, and operated within known constraints related to people, budget, time, and materials
Evolution	The ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment, balanced against the costs of providing such flexibility
Internationalization	The ability of the system to be independent from any particular language, country, or cultural group
Location	The ability of the system to overcome problems brought about by the absolute location of its elements and the distances between them
Performance and Scalability	The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes in the future if required
Regulation	The ability of the system to conform to local and international laws, quasi-legal regulations, company policies, and other rules and standards
Security	The ability of the system to reliably control, monitor, and audit who can perform what actions on which resources and the ability to detect and recover from security breaches
Usability	The ease with which people who interact with the system can work effectively

THE ARCHITECTURE DEFINITION PROCESS

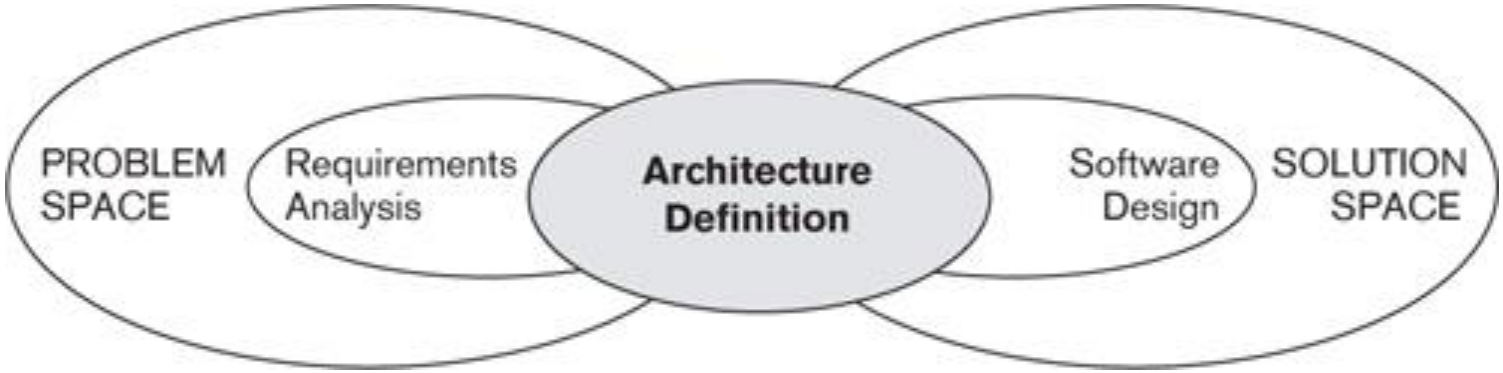
Architecture Definition

- **Architecture definition** is a process by which stakeholder needs and concerns are captured, an architecture to meet these needs is designed, and the architecture is clearly and unambiguously described via an architectural description.

A Good Architecture Definition

- A good architecture definition process is one that leads to a good architecture, documented by an effective architectural description, which can be realized in a way that is time-efficient and cost-effective for the organization.

Architecture Definition, Requirements Analysis, and Software Design



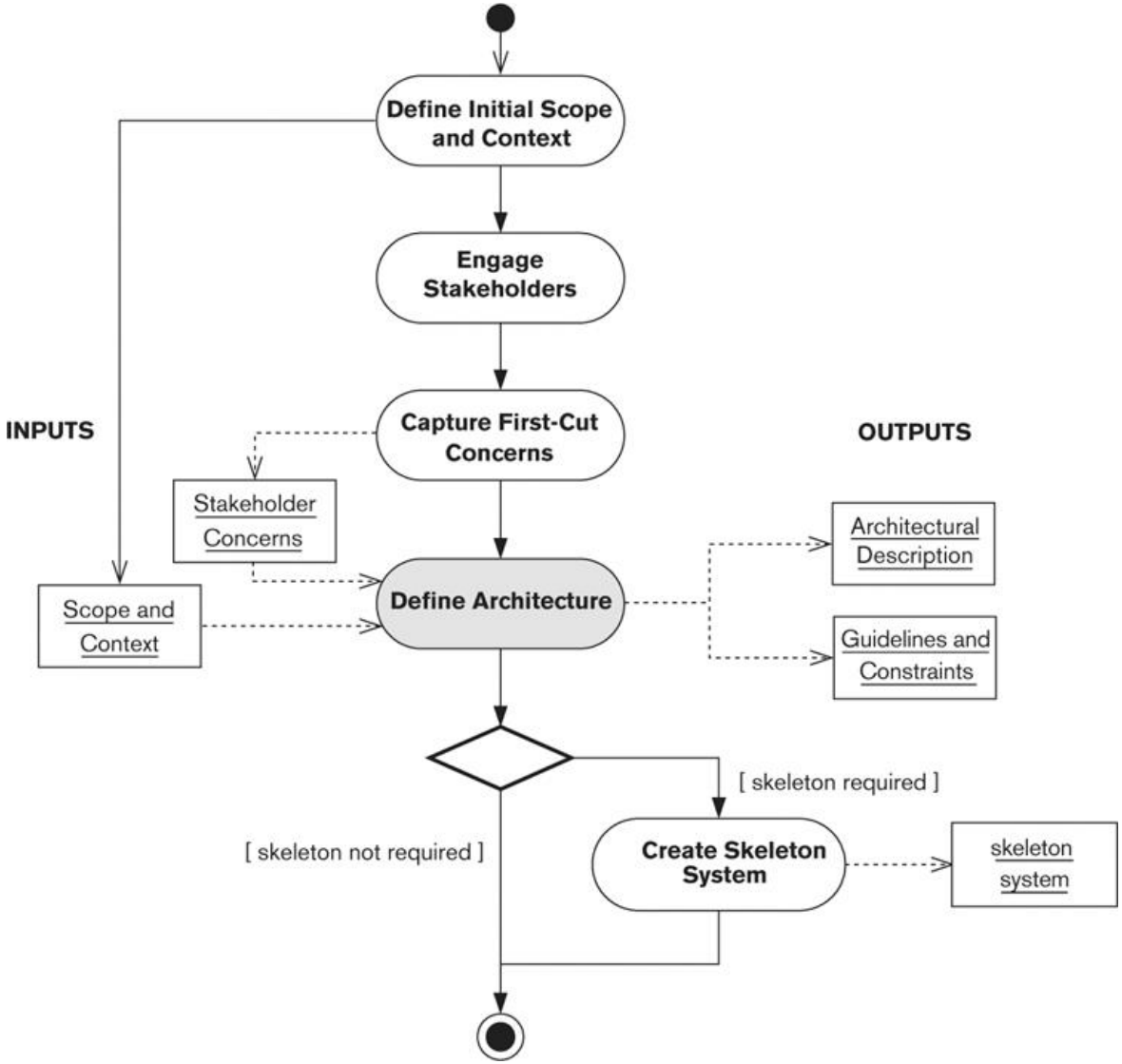
Definition

- A concern, problem, or system element is **architecturally significant** if it has a wide impact on the structure of the system or on its important quality properties such as performance, scalability, security, reliability, or evolvability.
- As you are designing the architecture, review the areas you have determined as being architecturally significant or not, and revise these as necessary in the light of your deeper understanding of your stakeholders' concerns and of the architecture itself.

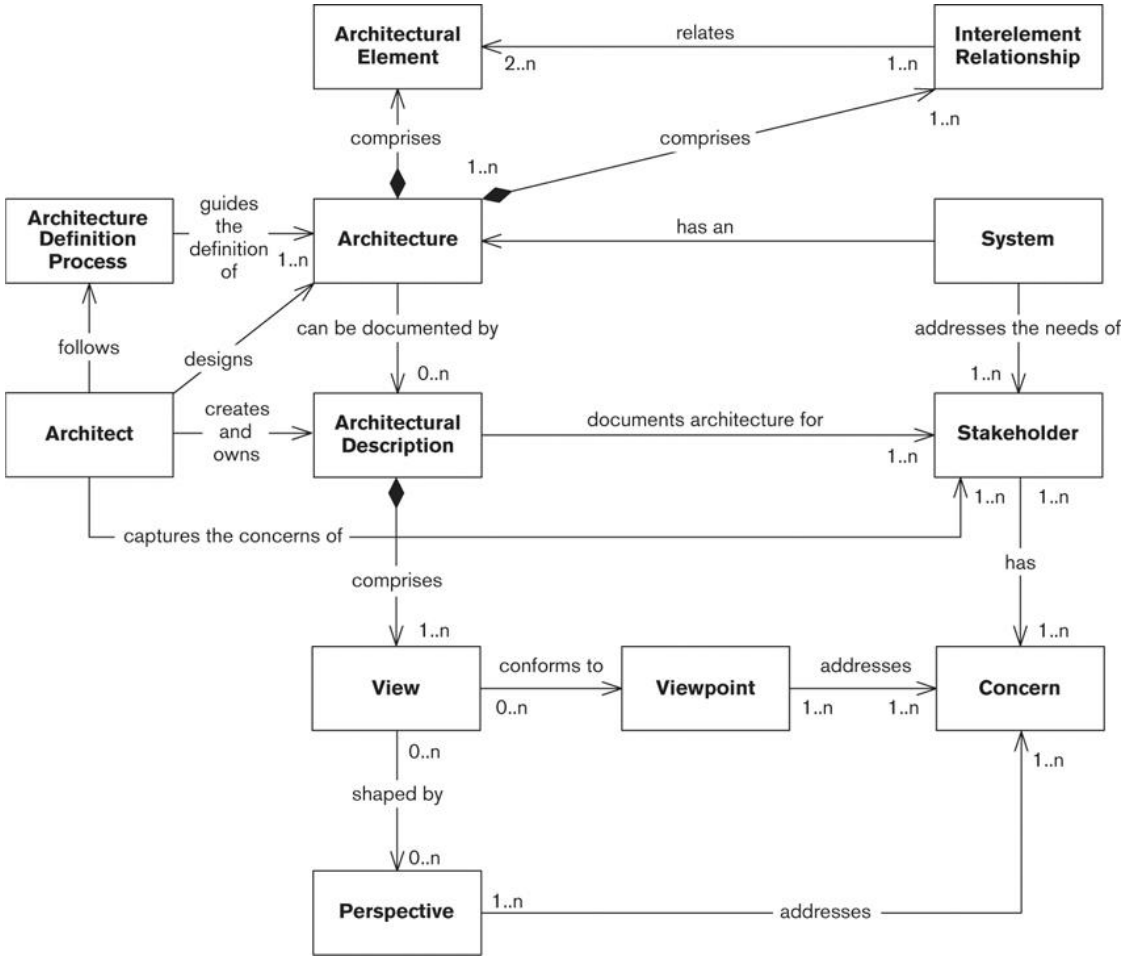
Architect Role

- **1.** To identify and engage the stakeholders
- **2.** To understand and capture the stakeholders' concerns
- **3.** To create and take ownership of the definition of an architecture that addresses these concerns
- **4.** To take a leading role in the realization of the architecture into a physical product or system

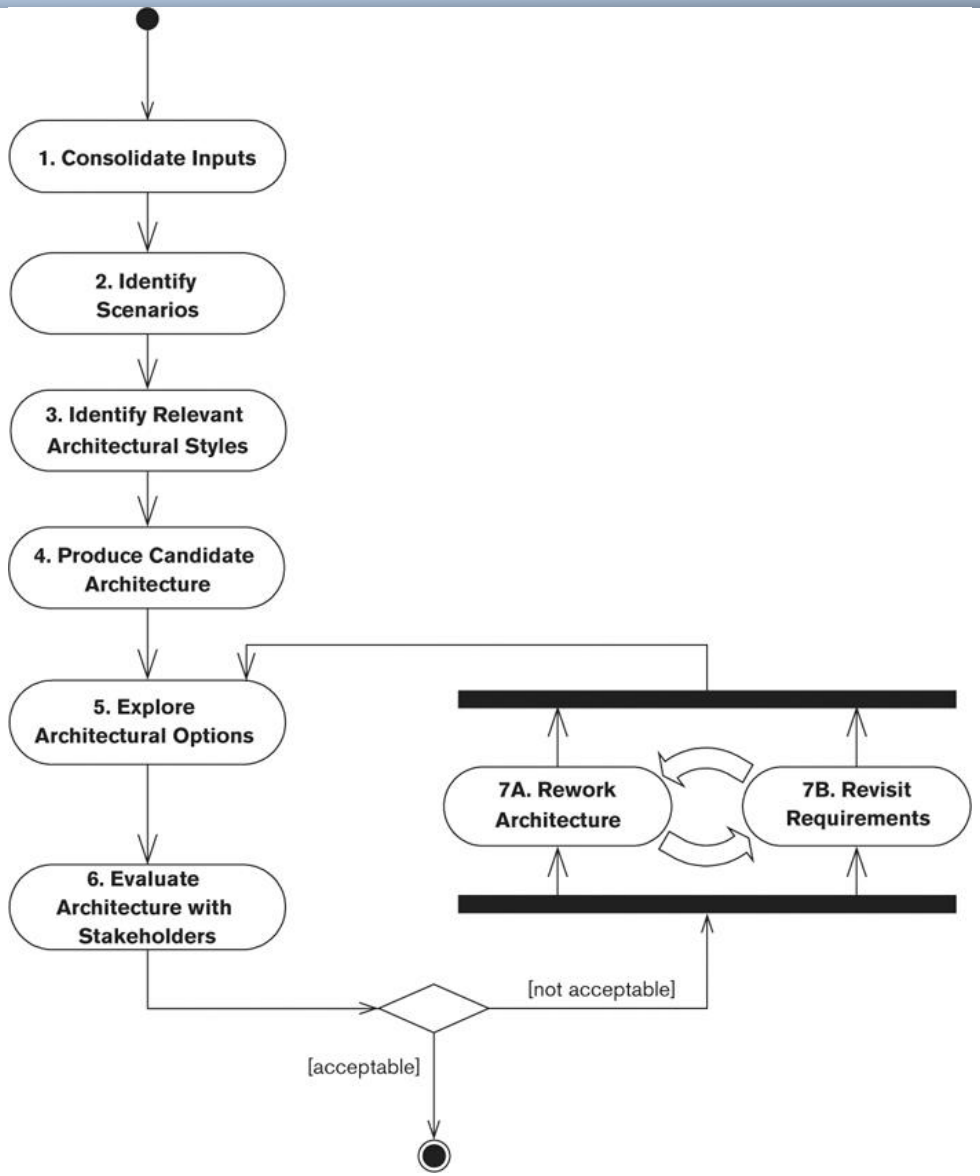
Activities Supporting Architecture Definition



Architecture Definition and the Architect in Context



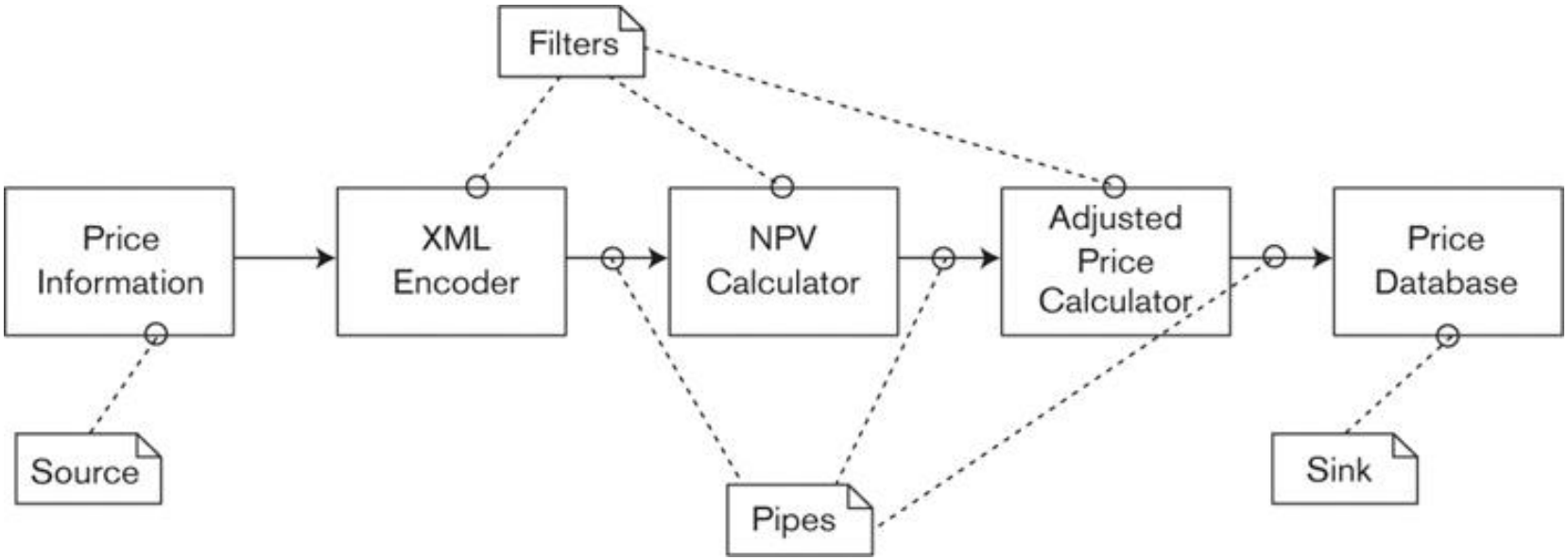
Details of Architecture Definition



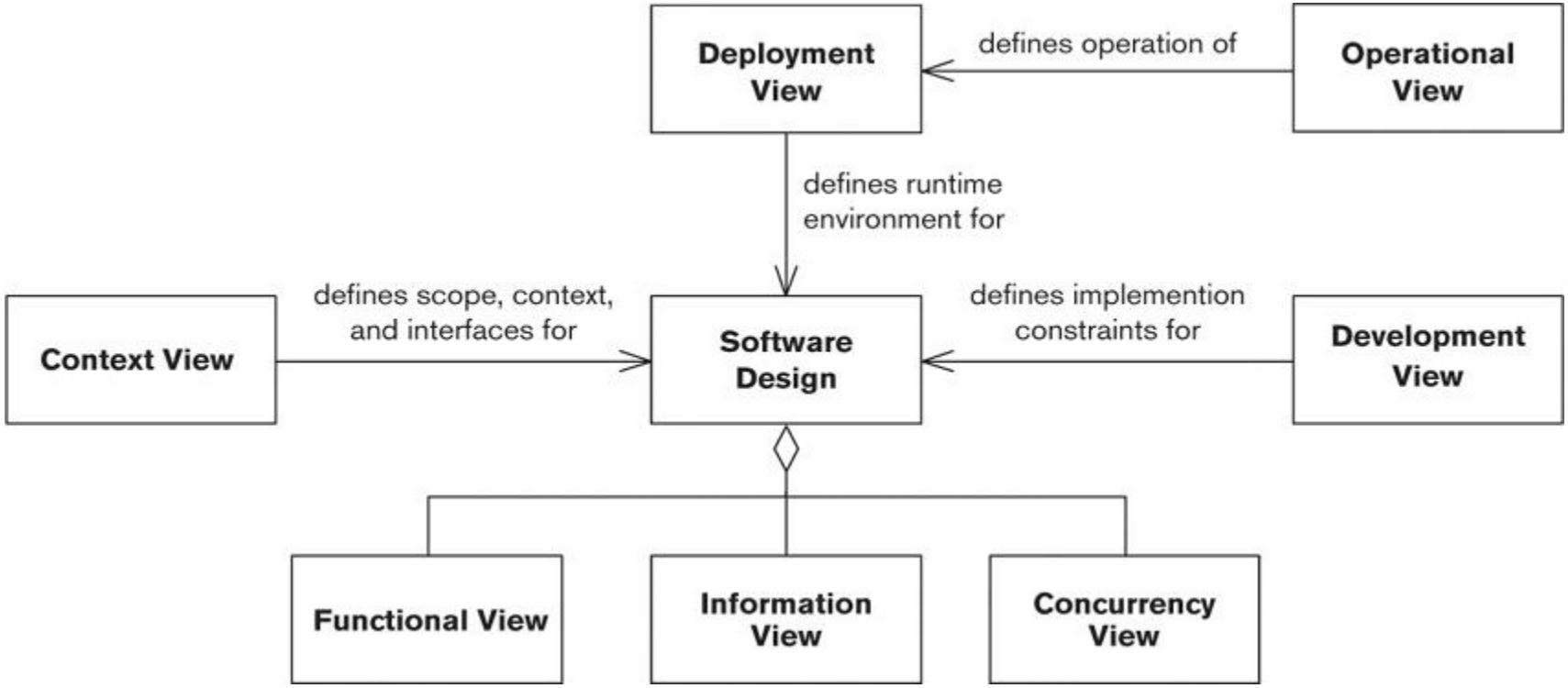
Concepts

- An **architectural style** expresses a fundamental structural organization schema for software systems. It provides a set of predefined element types, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.
- A **design pattern** documents a commonly recurring and proven structure of interconnected design elements that solves a general design problem within a particular context.
- A **language idiom** is a pattern specific to a programming language. An idiom describes how to implement particular aspects of elements or the relationships between them by using the features of a given language.

Example


Filters
Price Information
XML Encoder
NPV Calculator
Adjusted Price Calculator
Price Database
Source
Pipes
Sink

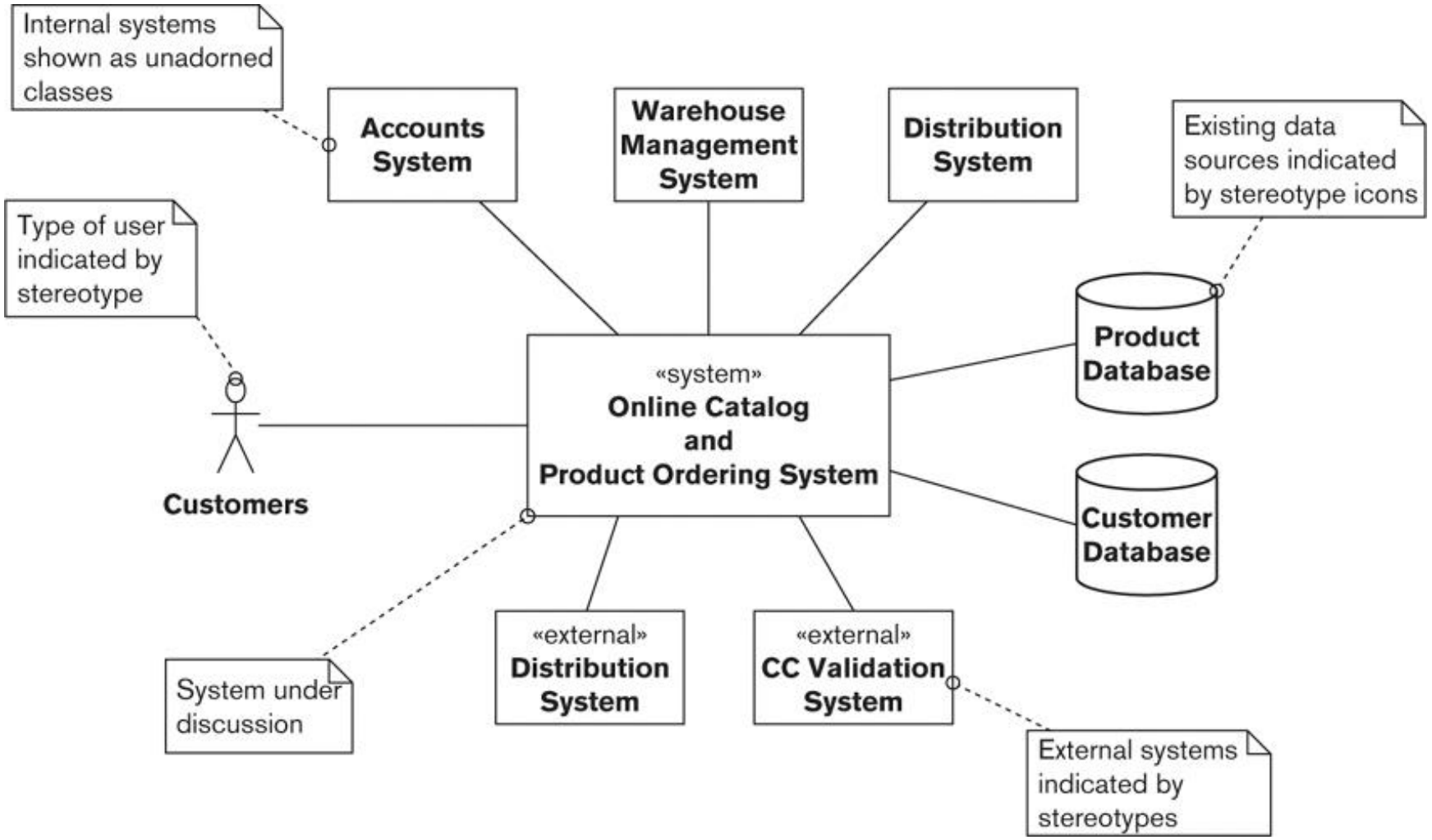
View Relationships



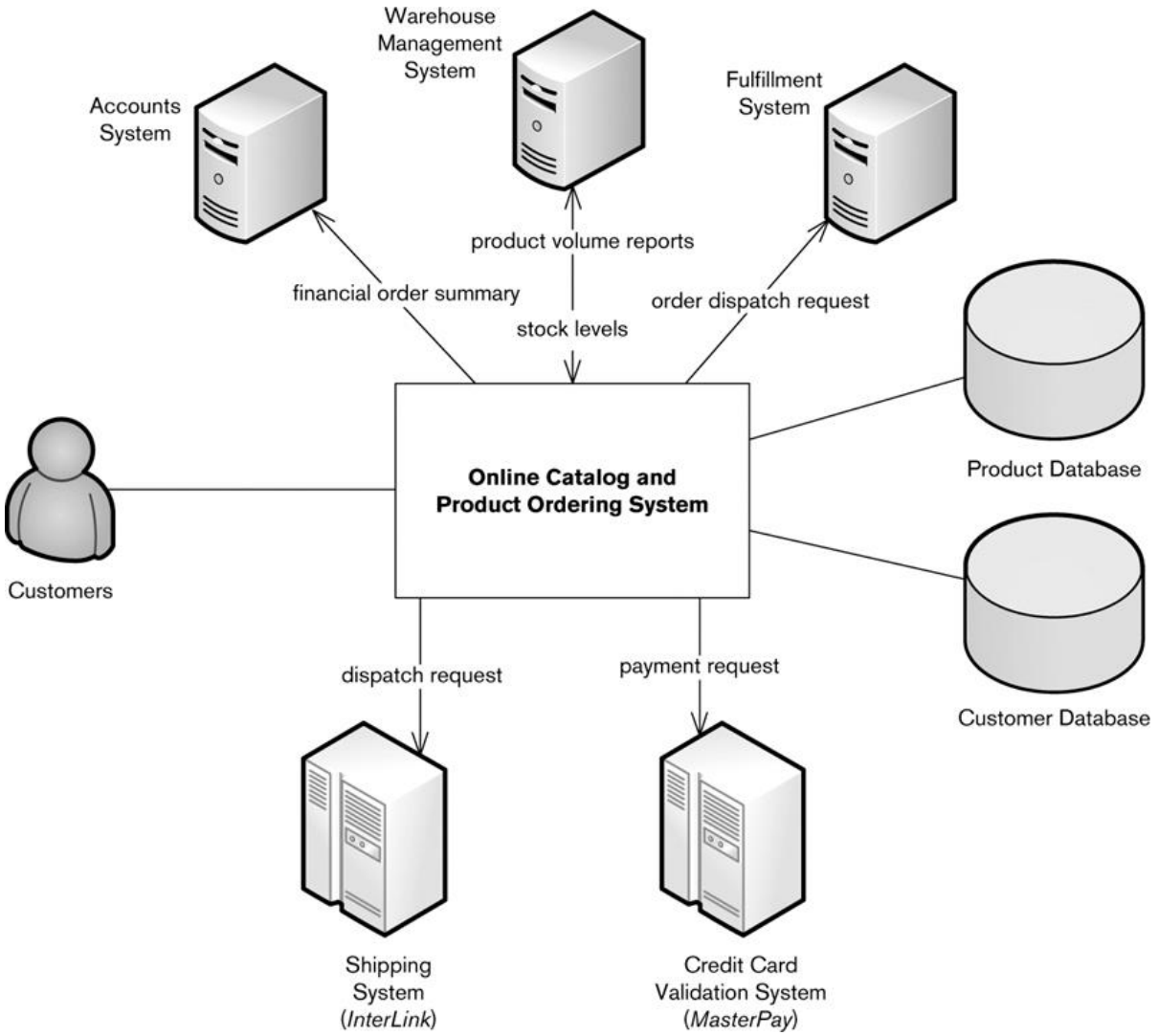
Context Viewpoint

Definition	Describes the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts)
Concerns	System scope and responsibilities, identity of external entities and services and data used, nature and characteristics of external entities, identity and responsibilities of external interfaces, nature and characteristics of external interfaces, other external interdependencies, impact of the system on its environment, and overall completeness, consistency, and coherence
Models	Context model, interaction scenarios
Problems and Pitfalls	Missing or incorrect external entities, missing implicit dependencies, loose or inaccurate interface descriptions, inappropriate level of detail, scope creep, implicit or assumed context or scope, overcomplicated interactions, overuse of jargon
Stakeholders	All stakeholders, but especially acquirers, users, and developers
Applicability	All systems

UML Context Diagram



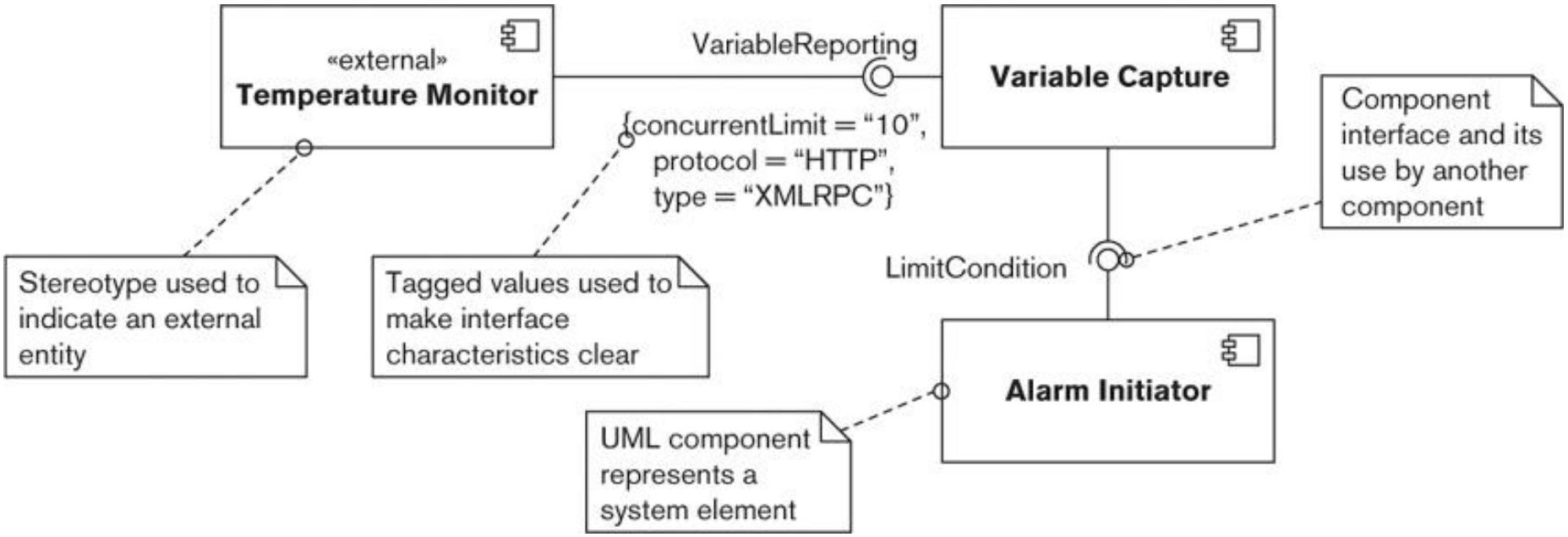
Informal Context Diagram



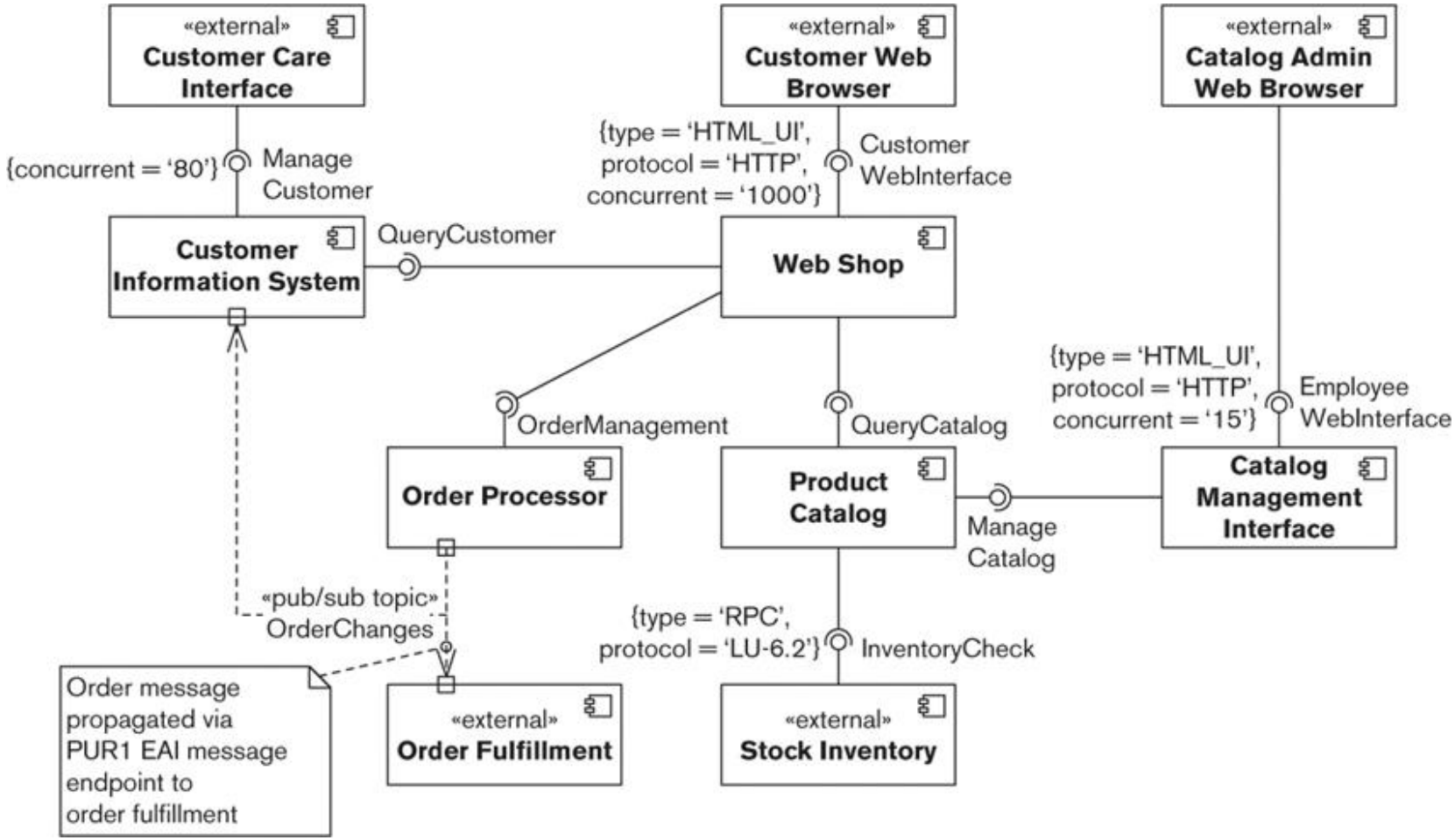
Functional Viewpoint

Definition	Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions
Concerns	Functional capabilities, external interfaces, internal structure, and functional design philosophy
Models	Functional structure model
Problems and Pitfalls	Poorly defined interfaces, poorly understood responsibilities, infrastructure modeled as functional elements, overloaded view, diagrams without element definitions, difficulty in reconciling the needs of multiple stakeholders, wrong level of detail, "God elements," and too many dependencies
Stakeholders	All stakeholders
Applicability	All systems

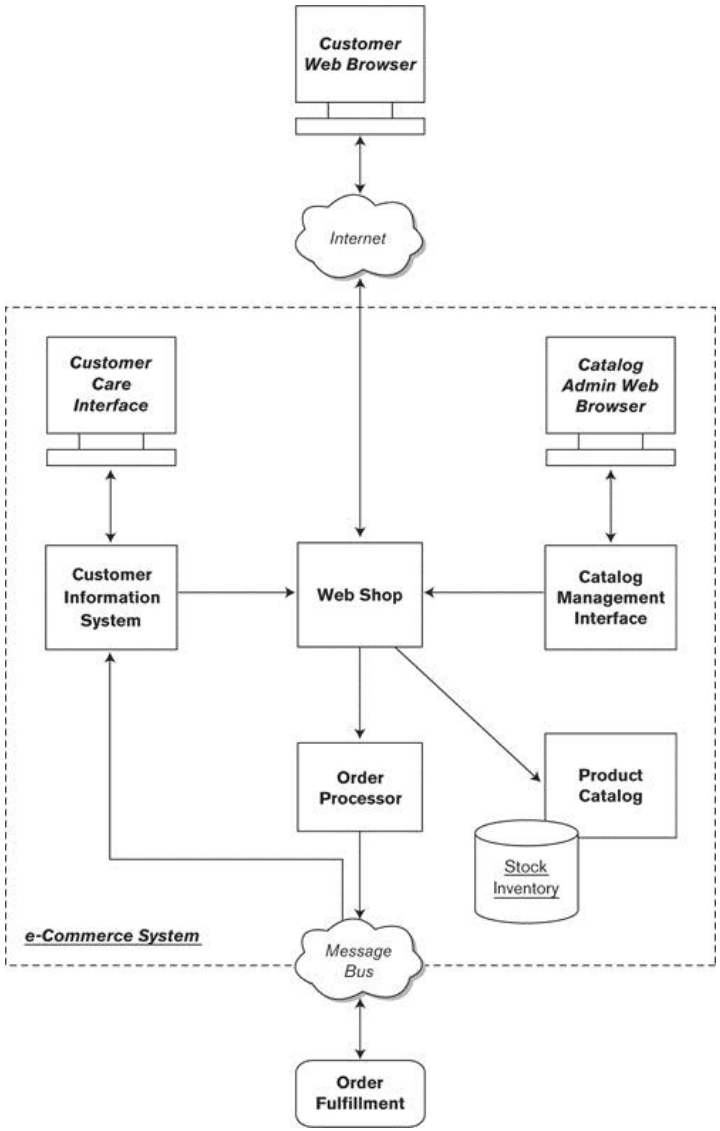
Example of a Functional Structure in UML



Example of a UML Component Diagram



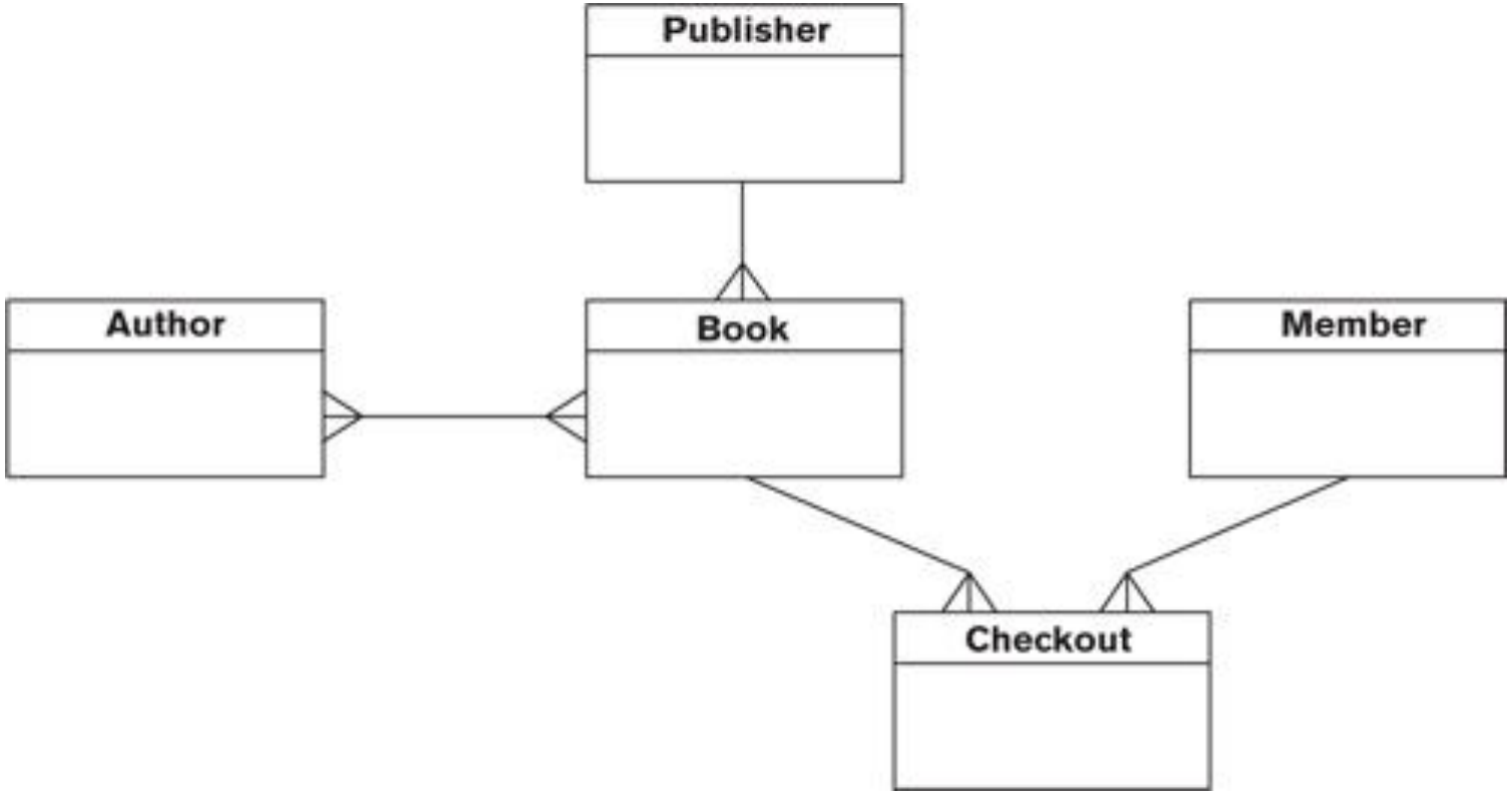
Example of a Boxes-And-Lines Diagram



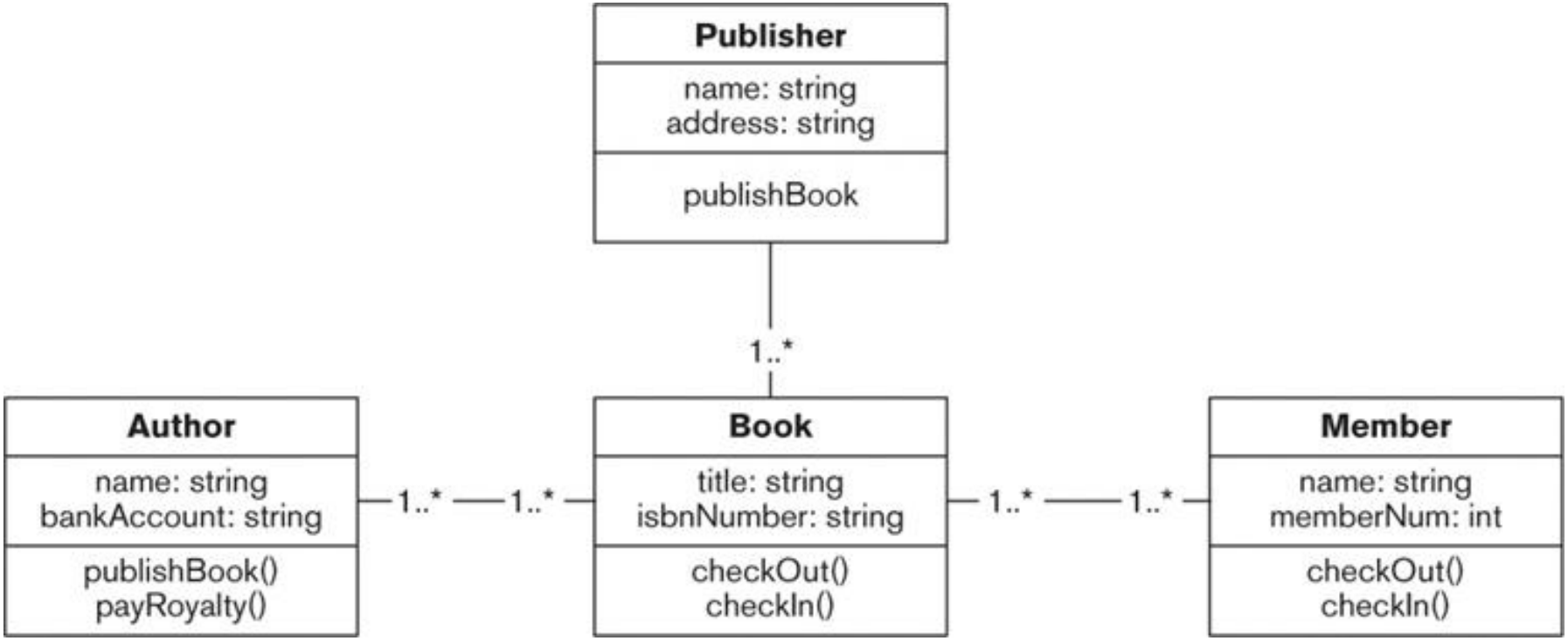
The information ViewPoint

Definition	Describes the way that the system stores, manipulates, manages, and distributes information
Concerns	Information structure and content; information purpose and usage; information ownership; enterprise-owned information; identifiers and mappings; volatility of information semantics; information storage models; information flow; information consistency; information quality; timeliness, latency, and age; and archiving and information retention
Models	Static information structure models, information flow models, information lifecycle models, information ownership models, information quality analysis, metadata models, and volumetric models
Problems and Pitfalls	Representation incompatibilities, unavoidable multiple updaters, key-matching deficiencies, interface complexity, overloaded central database, inconsistent distributed databases, poor information quality, excessive information latency, and inadequate volumetrics
Stakeholders	Primarily users, acquirers, developers, testers, and maintainers, but most stakeholders have some level of interest
Applicability	Any system that has more than trivial information management needs

Entity-Relationship Diagram for the Library Example



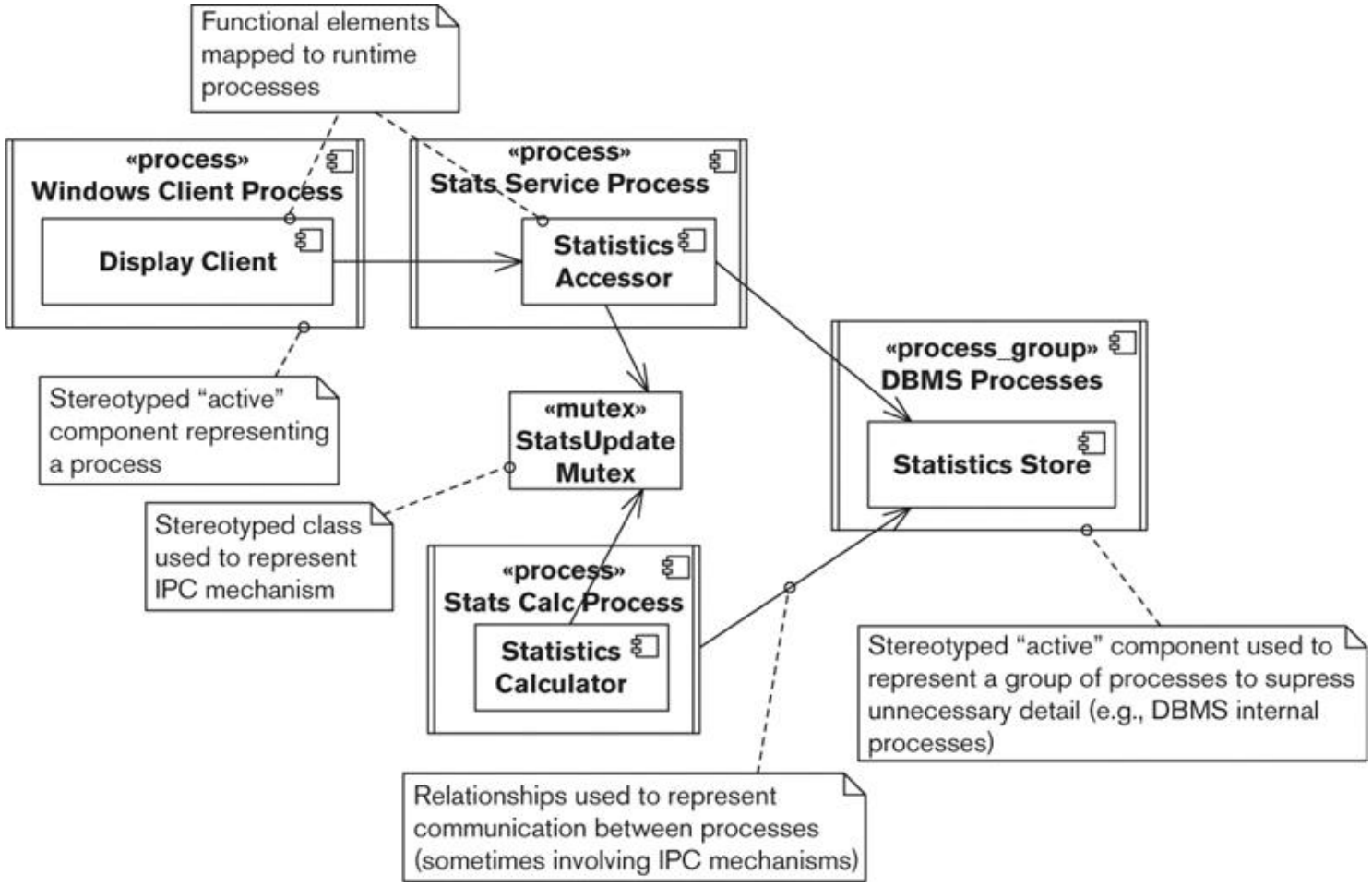
UML Class Model for the Library Example



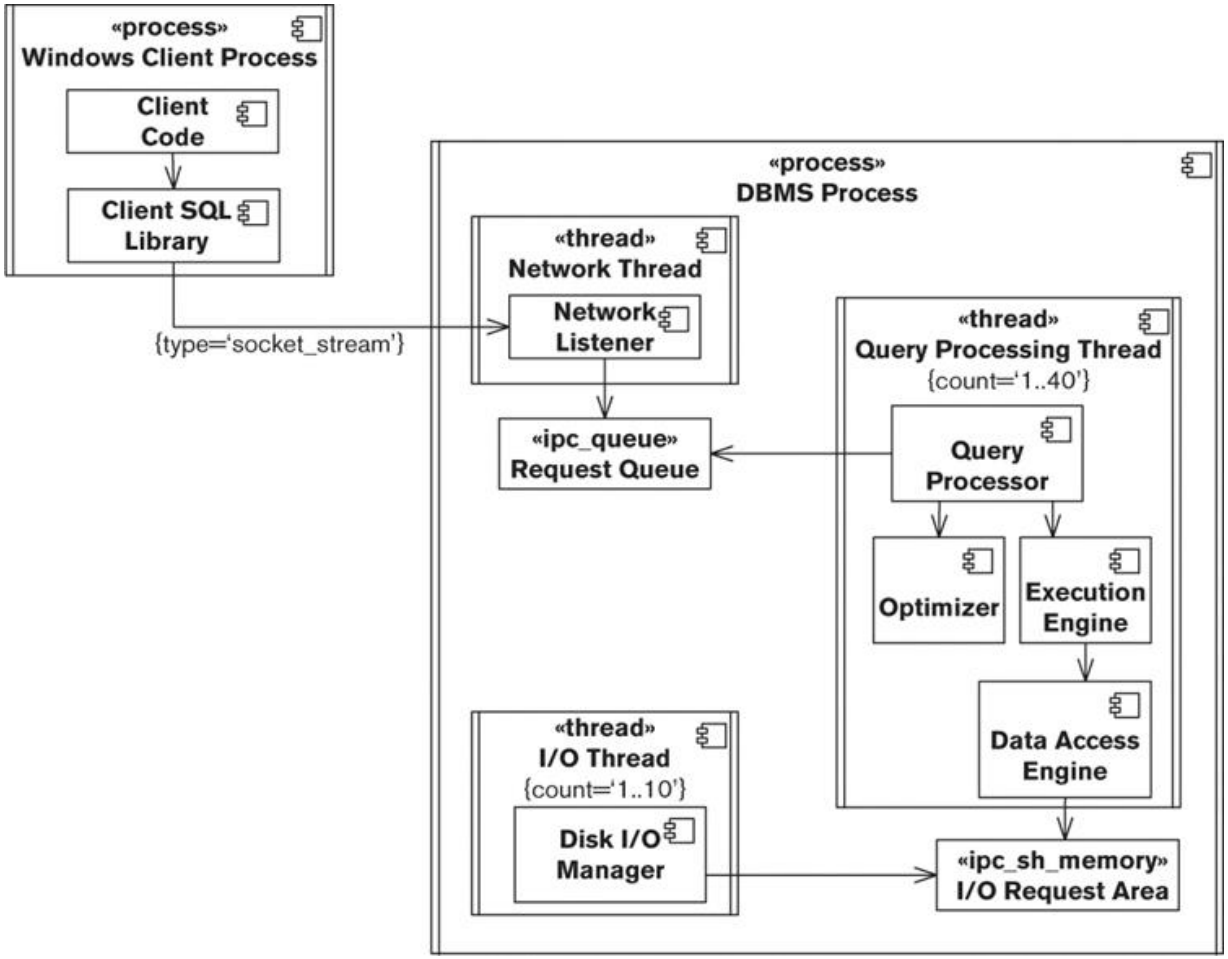
The Concurrency Viewpoint

Definition	Describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled
Concerns	Task structure, mapping of functional elements to tasks, interprocess communication, state management, synchronization and integrity, supporting scalability, startup and shutdown, task failure, and reentrancy
Models	System-level concurrency models and state models
Problems and Pitfalls	Modeling the wrong concurrency, modeling the concurrency wrongly, excessive complexity, resource contention, deadlock, and race conditions
Stakeholders	Communicators, developers, testers, and some administrators
Applicability	All information systems with a number of concurrent threads of execution

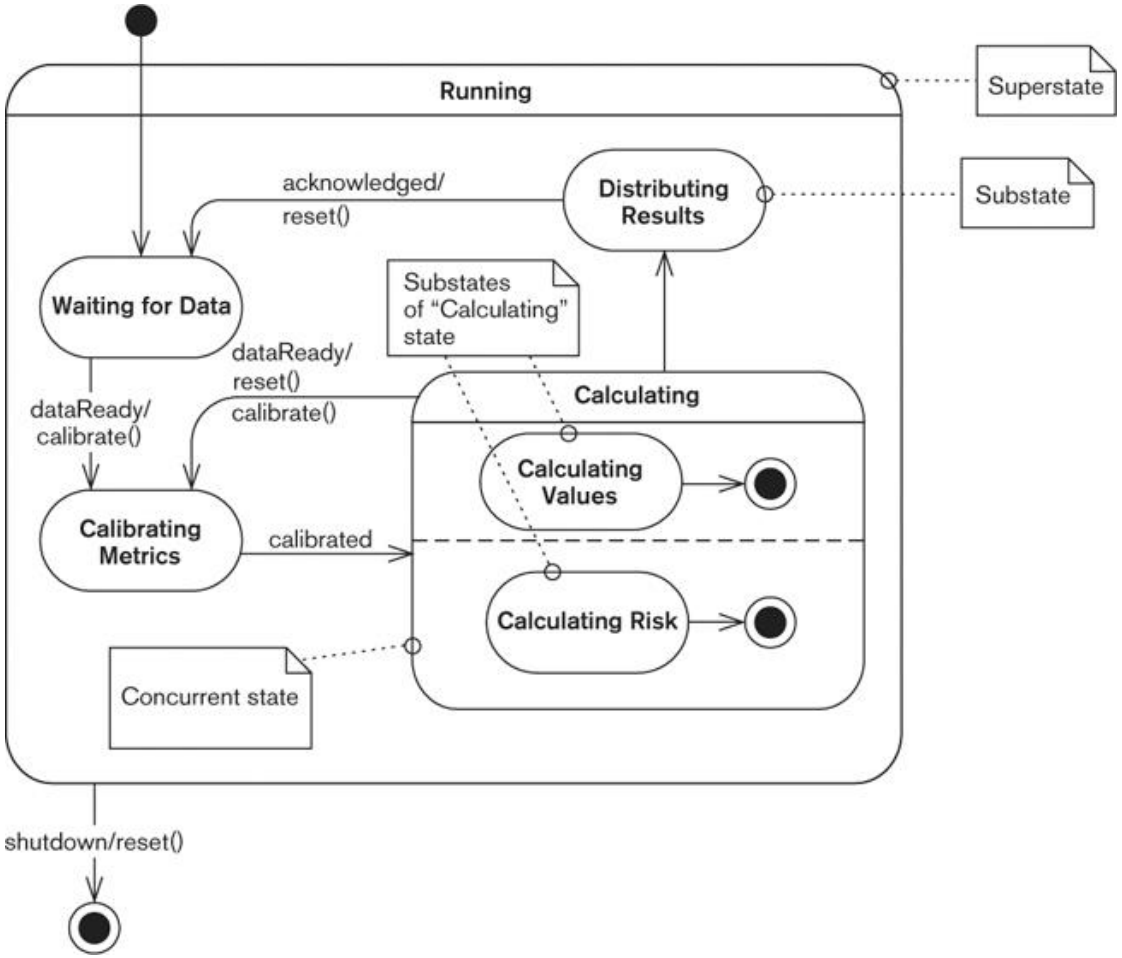
Concurrency Model Documented by Using UML



Thread-Based Concurrency Model



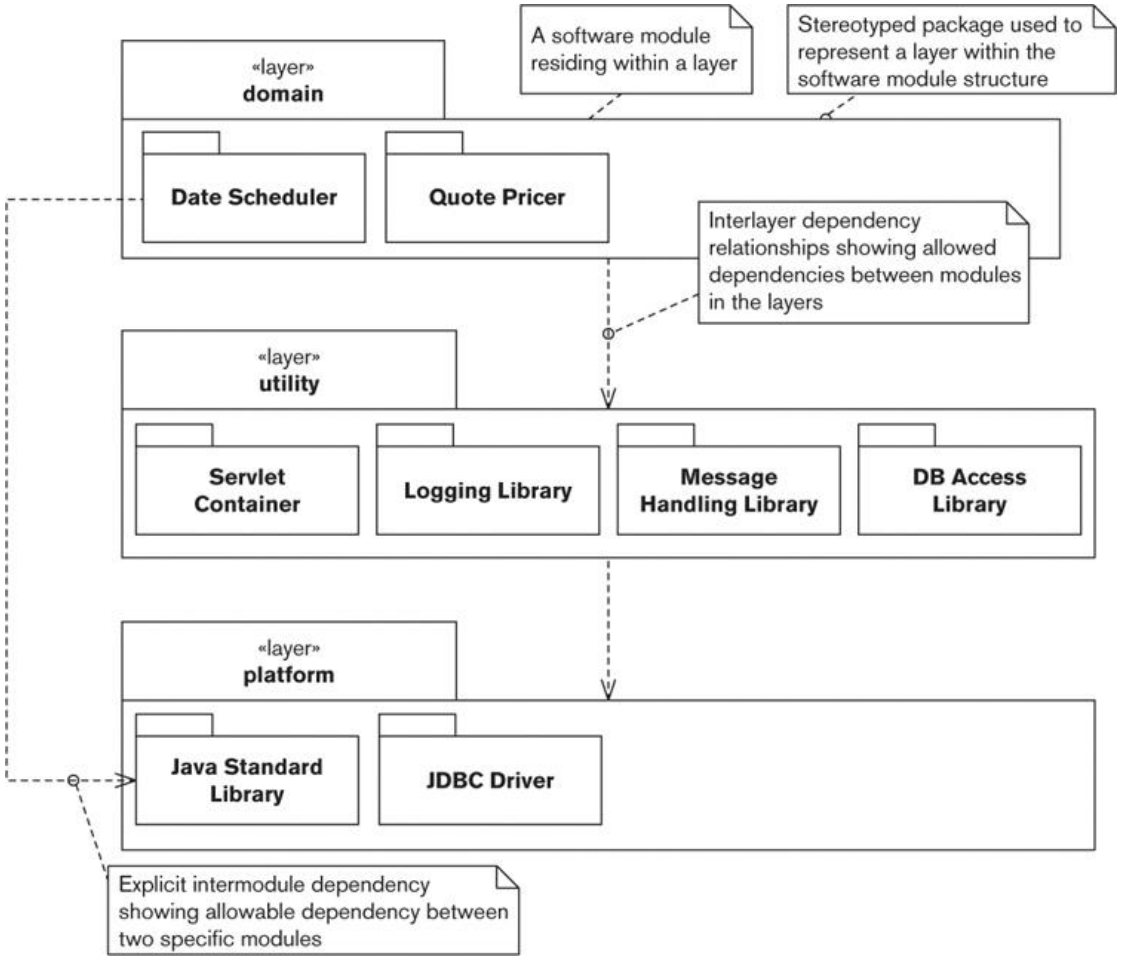
Example Statechart for a Calculation Engine



The Development Viewpoint

Definition	Describes the architecture that supports the software development process
Concerns	Module organization, common processing, standardization of design, standardization of testing, instrumentation, and codeline organization
Models	Module structure models, common design models, and codeline models
Problems and Pitfalls	Too much detail, overburdened architectural description, uneven focus, lack of developer focus, lack of precision, and problems with the specified environment
Stakeholders	Production engineers, software developers and testers
Applicability	All systems with significant software development involved in their creation

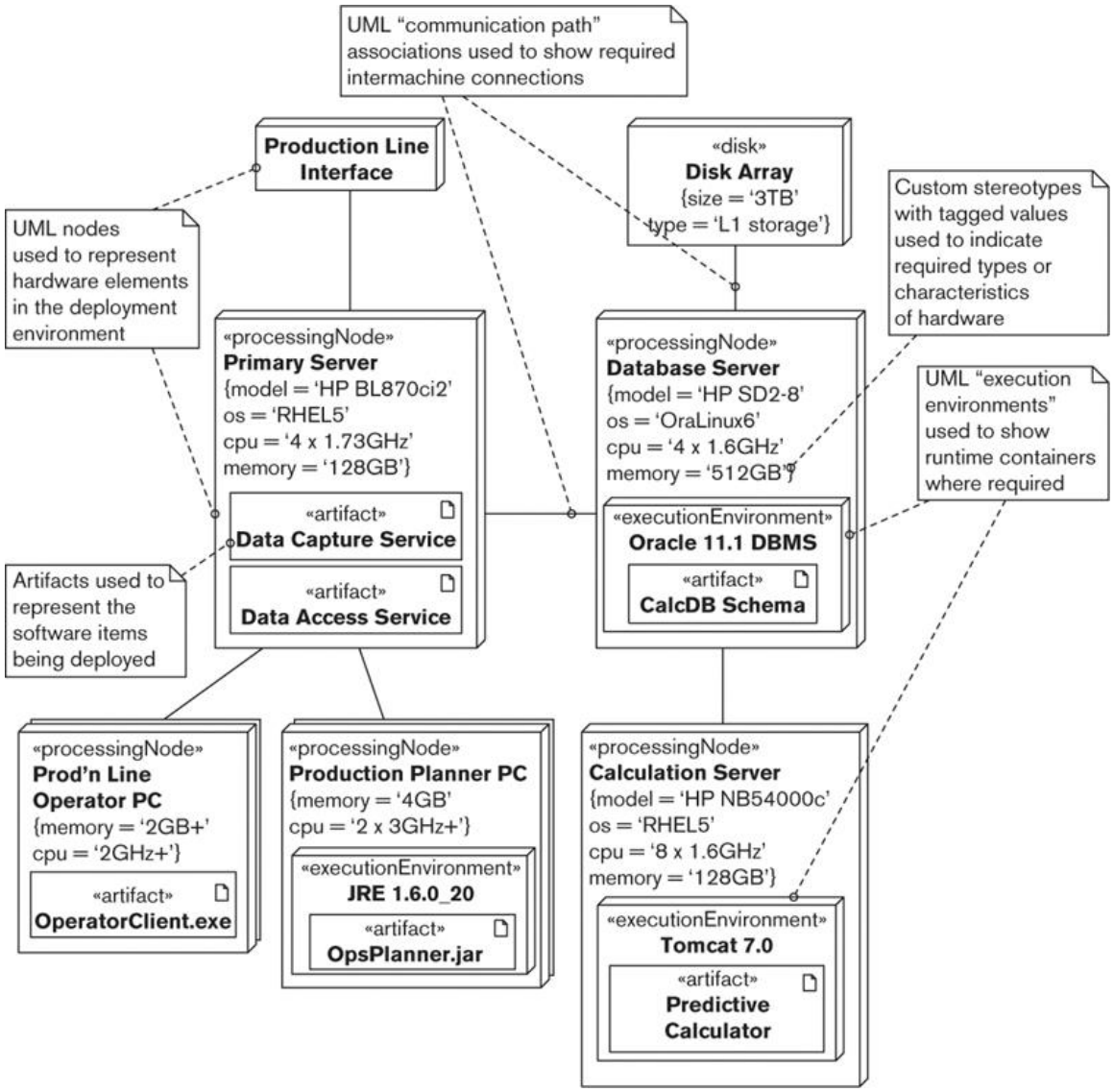
Example of a UML Module Structure Model



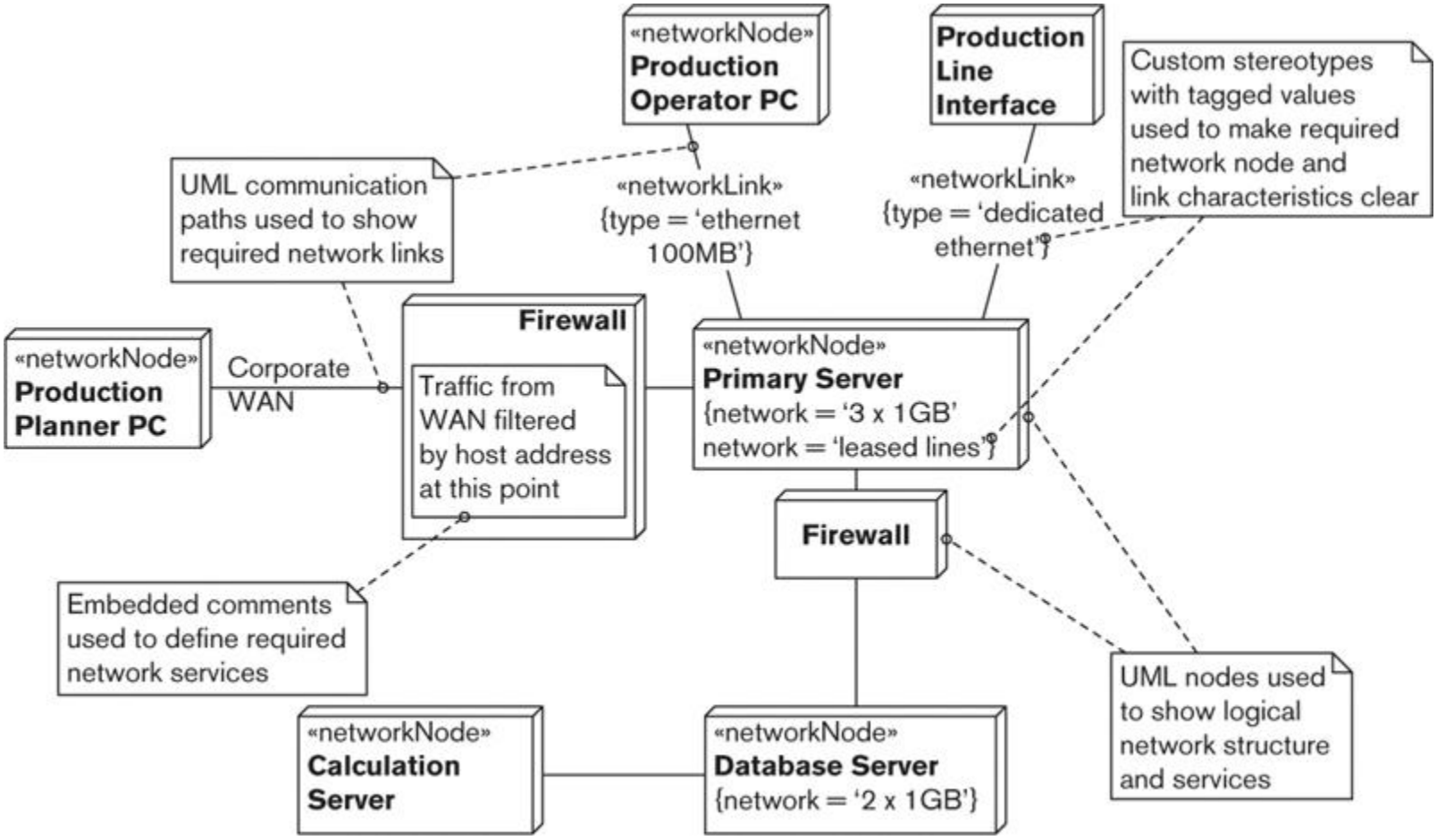
The Deployment Viewpoint

Definition	Describes the environment into which the system will be deployed and the dependencies that the system has on elements of it
Concerns	Runtime platform required, specification and quantity of hardware or hosting required, third-party software requirements, technology compatibility, network requirements, network capacity required, and physical constraints
Models	Runtime platform models, network models, technology dependency models, and intermodel relationships.
Problems and Pitfalls	Unclear or inaccurate dependencies, unproven technology, unsuitable or missing service-level agreements, lack of specialist technical knowledge, late consideration of the deployment environment, ignoring intersite complexities, inappropriate headroom provision, and not specifying a disaster recovery environment
Stakeholders	System administrators, developers, testers, communicators, and assessors
Applicability	Systems with complex or unfamiliar deployment environments

Example of a Runtime Platform Model



Example of a Network Model



The Operational Viewpoint

Definition	Describes how the system will be operated, administered, and supported when it is running in its production environment
Concerns	Installation and upgrade, functional migration, data migration, operational monitoring and control, alerting, configuration management, performance monitoring, support, backup and restore, and operation in third-party environments
Models	Installation models, migration models, configuration management models, administration models, and support models
Problems and Pitfalls	Lack of engagement with the operational staff, lack of backout planning, lack of migration planning, insufficient migration window, missing management tools, production environment constraints, lack of integration into the production environment, inadequate backup models, and unsuitable alerting
Stakeholders	System administrators, production engineers, developers, testers, communicators, and assessors
Applicability	Any system being deployed into a complex or critical operational environment

Data Migration From a Live System