zühlke
DIE DENKFABRIK.

BCS® SPA

Software Architecture with Viewpoints and Perspectives

BCS SPA Specialist Group

6th July 2005

Nick Rozanski
nick@rozanski.com
www.nick.rozanski.com

Eoin Woods
ewo@zuhlke.com
www.eoinwoods.info

Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Uses for Viewpoints and Perspectives

Defining Software Architecture

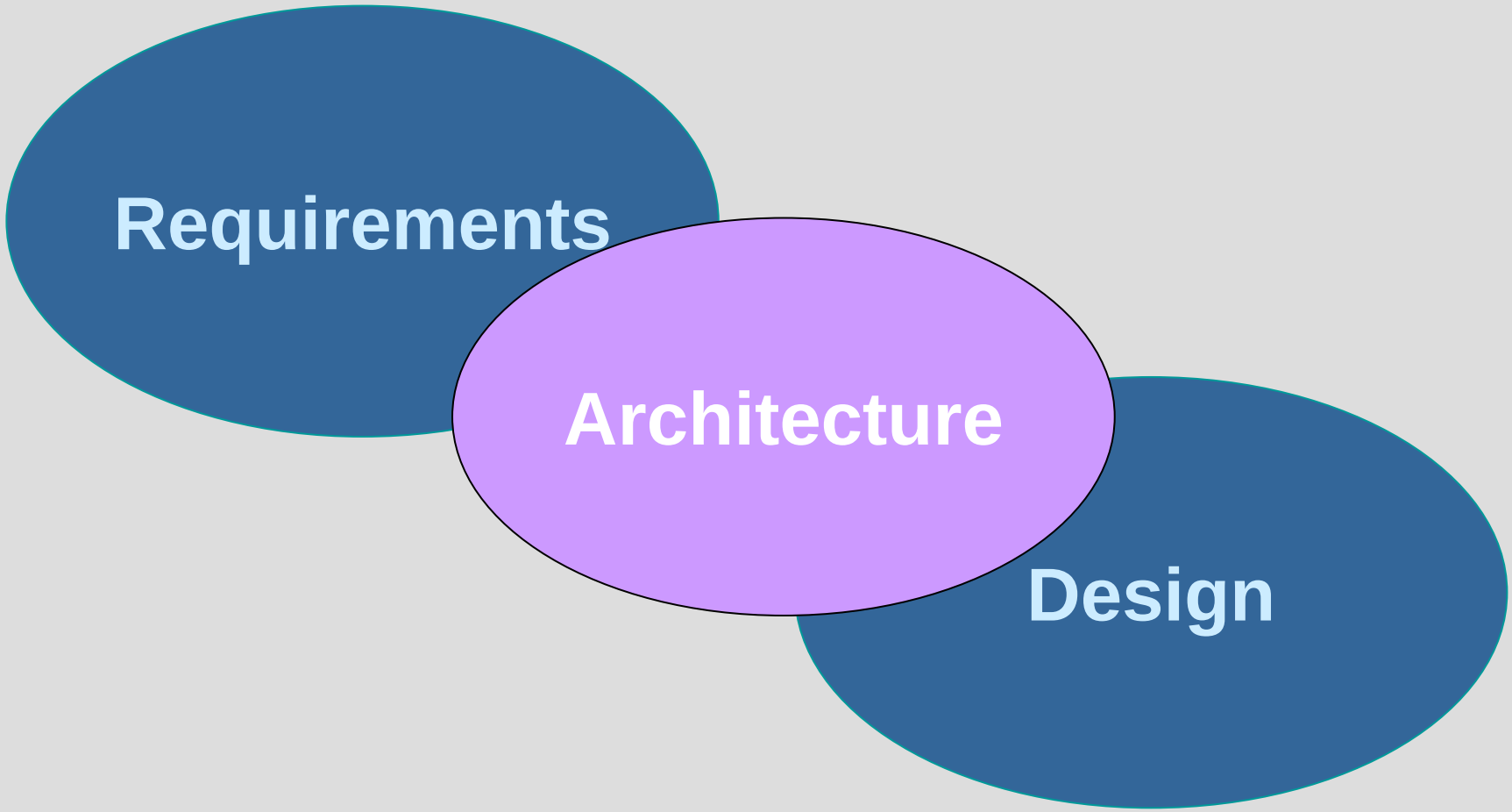
A common definition …

*The software architecture of a program or computing system is the **structure** or structures of the system, which comprise software **elements** the externally visible **qualities** of those elements, and the **relationships** among them*

Len Bass, Paul Clements and Rick Kazman (SEI)
Software Architecture in Practice, 2nd Edition

Role of Software Architecture

A crucial bridge between requirements and design



Software Architecture and Requirements

Requirements frame the architectural problem

- Stakeholder needs and desires

Yet, architecture must influence requirements

- “The art of the possible”
- Helps stakeholder understanding of risk/cost
- Helps stakeholder understanding of possibilities

Software Architecture and Quality Properties

The non-functional system characteristics ("-illities")

- Performance, Efficiency, Security, Availability, …

Quality properties are crucial to stakeholders

- Slow functions don't get used
- Unavailable systems cause business interruption
- Security problems cause headlines

Yet quality properties are often an after-thought

Addressing quality properties is a *key* architectural task

- Understanding "real" stakeholder needs & required tradeoffs
- Often expensive to "retro-fit"

Stakeholders

Who are Stakeholders?

- People
- Groups
- Entities

Those who have an interest in or concerns about the realisation of the architecture

The Importance of Stakeholders

- Architectures are built for stakeholders
- Architectural decisions must reflect stakeholder needs
- A wide stakeholder community increases your chance of success

Stakeholders

Common stakeholder groups

- ***Acquirers***, who pay for it
- ***Assessors***, who check it for compliance
- ***Communicators***, who tell people about it via documents and training
- ***Developers***, who create it
- ***Maintainers***, who evolve it and fix it
- ***Suppliers***, who provide parts of it
- ***Support Staff***, who help people to use it
- ***System Administrators***, who keep it running
- ***Testers***, who verify that it works
- ***Users***, who have to use it as part of their work

Attributes of a good stakeholder

- ***Informed***, to allow them to make good decisions
- ***Committed***, to the process and willing to make themselves available in a constructive manner, even if decisions are hard
- ***Authorised***, to make decisions
- ***Representative***, of their stakeholder group so that they present its views validly

The Software Architecture Problem

Why software architecture is difficult

- Multi-dimensional problem
- Diverse stakeholder community to serve
- Making trade-offs inherent in the process
- Often no one “right” answer

Architecture practice today is largely ad-hoc

- Little standardisation in description
- Difficult to compare and discuss alternatives
- Unclear how to structure architectural activities
- No framework for handling quality properties

The Software Architecture Problem

We need a conceptual framework

- To organise the architectural design process
  - How to relate and organise the different activities
  - Where stakeholders fit

- To allow classification and sharing of ideas and best practice
  - Proven solutions
  - Known problems and pitfalls

- To capture knowledge for discussion and reuse
  - Standard form for easy reference
  - Summary of major ideas for easy use

Architectural Viewpoints

Dealing with architectural structure

- Decompose the architectural description into ***views***
  - Each view addresses one aspect of the architectural structure
- Guide the development of each view via a ***viewpoint***
  - The viewpoint contains proven practice, pitfalls, etc.
- Well understood approach
  - RUP/Kruchten “4+1”
  - Siemens set
  - RM-ODP set
  - Rozanski & Woods set
- Approach standardised by IEEE standard 1471 (2000)

Viewpoints and Views

IEEE 1471 provides standard definitions

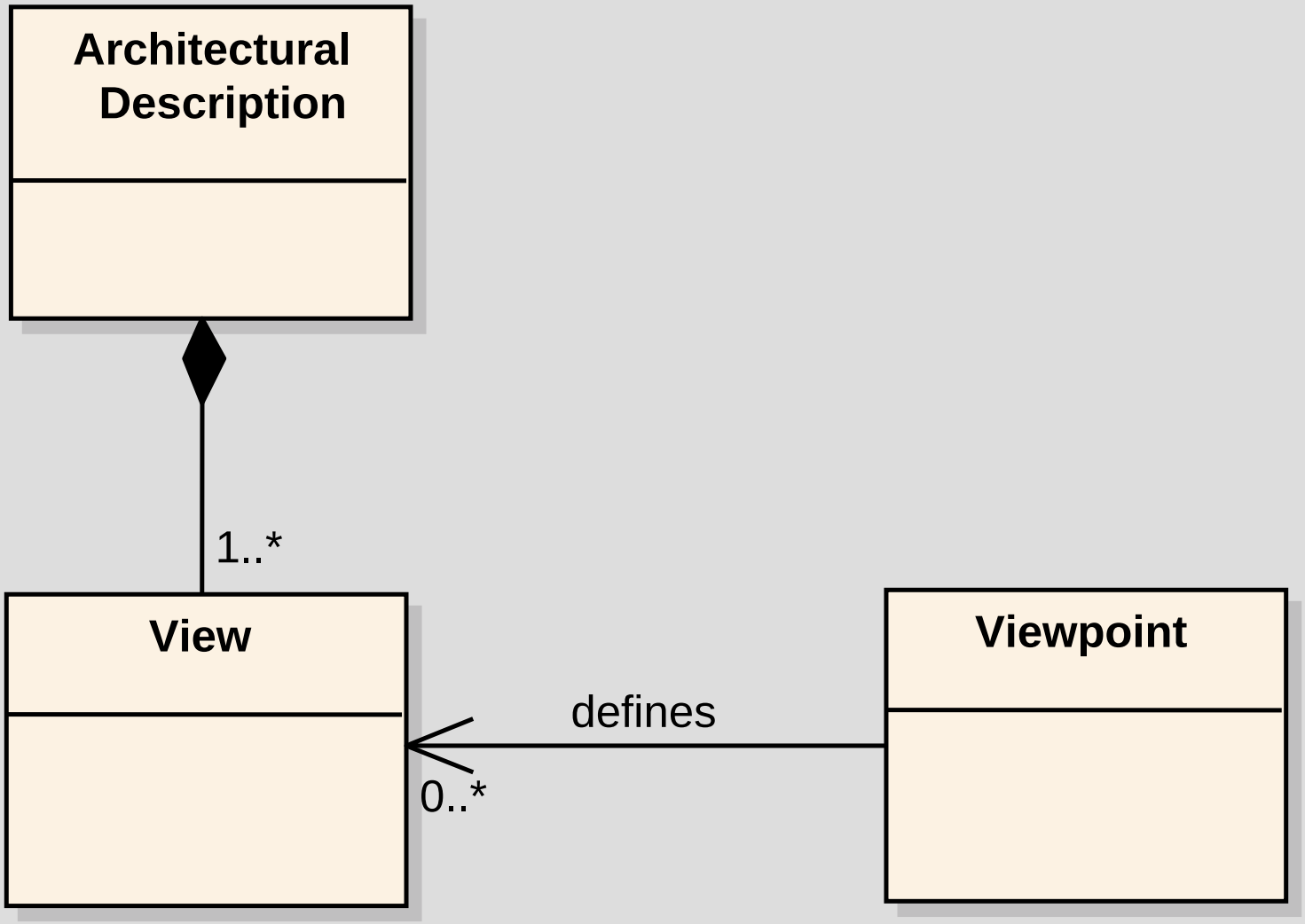
A **viewpoint** *is a collection of* ***patterns***, ***templates*** *and* ***conventions*** *for constructing* ***one type of view****. It defines the* ***stakeholders*** *whose* ***concerns*** *are reflected in the viewpoint, and* ***guidelines*** *and* ***principles*** *and template* ***models*** *for constructing its views.*

A **view** *is a* ***representation*** *of all or* ***part of an architecture****, from the perspective of one or more concerns which are held by one or more of its stakeholders.*

IEEE Standard 1471 – Recommended Practice for Architectural Description (2000)

Viewpoints and Views

Inter-relationships



Viewpoints and Views

Example viewpoint set

- **Functional**: elements, connectors, interfaces
- **Information**: entities, constraints, relationship, ownership, usage
- **Concurrency**: processes, threads, coordination, element mapping
- **Development**: layers, module structure, standard design, codeline
- **Deployment**: hardware, network, dependencies, process mapping
- **Operational**: installation, migration, administration, support

[Rozanski and Woods; "*Software Systems Architecture*" – Addison Wesley, 2005]

Viewpoints and Views

Woods/Rozanski Viewpoint Set

- Aimed at modern, large scale, distributed information systems
- Extension and refinement of Philippe Kruchten's "4+1" set
  - Renamed "Logical", "Process" and "Physical"
  - Added "Information" and "Operational"
- Defines the contents of the viewpoints
  - Not just noted their existence

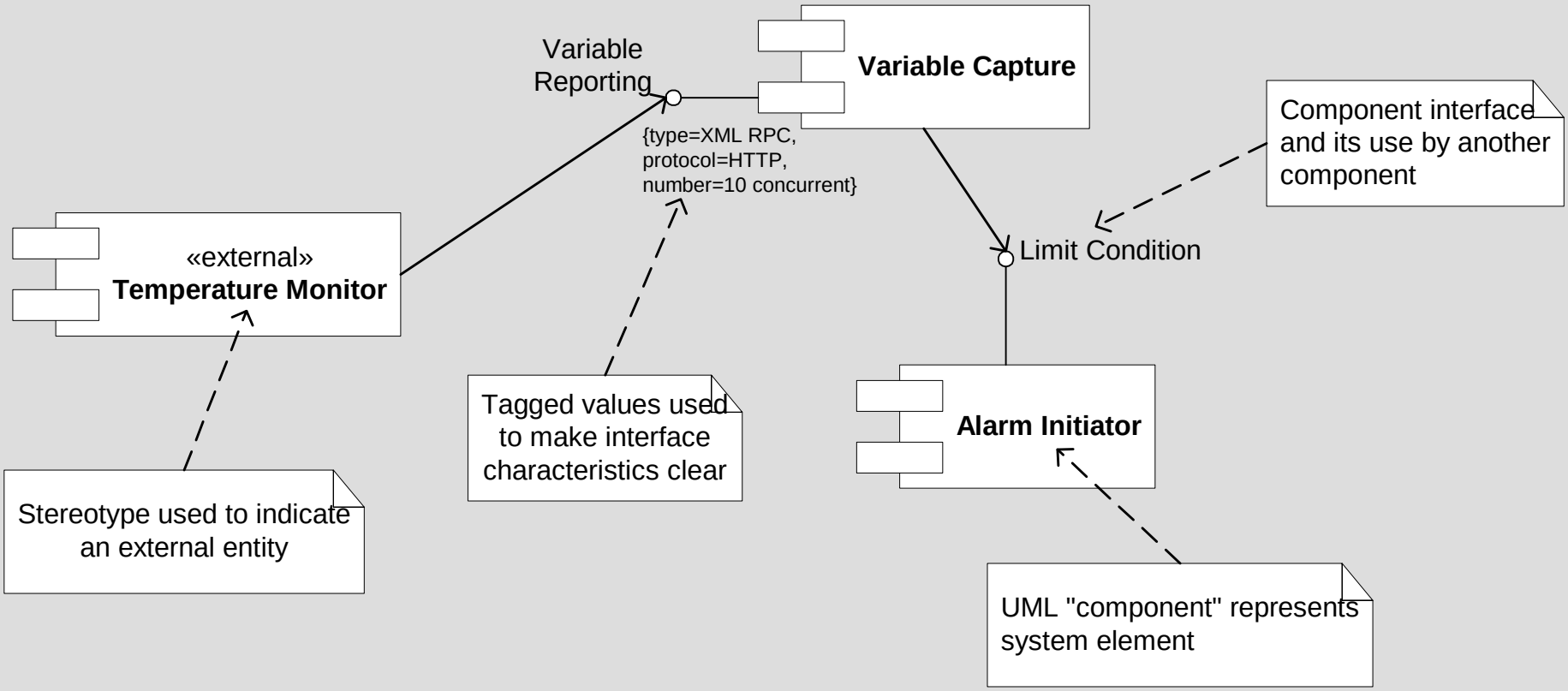
Viewpoints and Views

Functional

The functional structure of the system

- ***Content***: the system's runtime functional elements and their responsibilities, interfaces, and primary interactions
- ***Concerns***: functional capabilities, external interfaces, internal structure, and design philosophy
- ***Models***: functional structure model
- ***Pitfalls***: poorly defined interfaces, poorly understood responsibilities, infrastructure modelled as functional elements, overloaded view, diagrams without element definitions, difficulty in reconciling the needs of multiple stakeholders, inappropriate level of detail, "God elements," and too many dependencies

Functional - Example Model Fragment



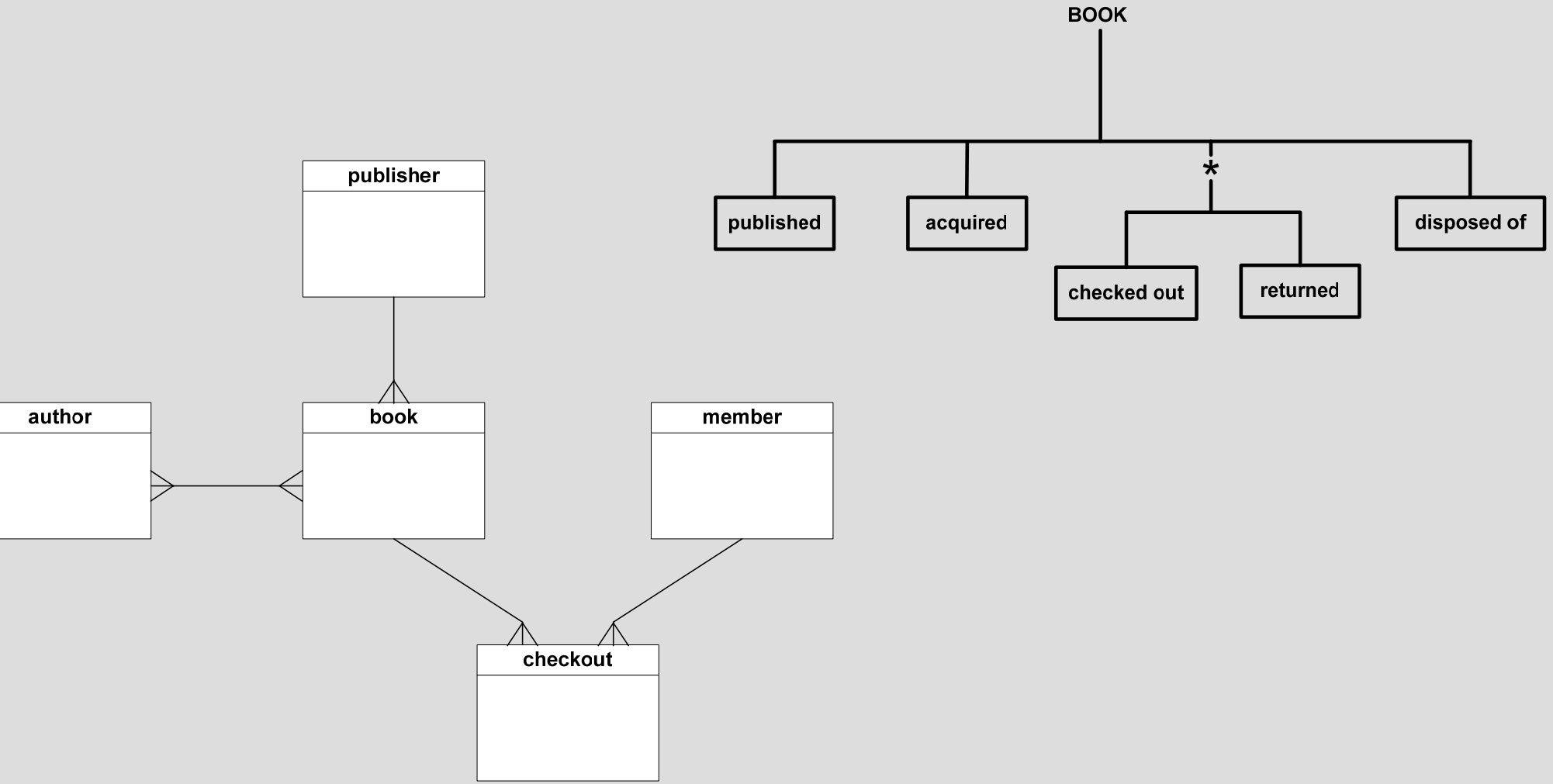
Viewpoints and Views

Information

The information structure, ownership and processing in the system

- ***Content***: how the system stores, manipulates, manages, and distributes information
- ***Concerns***: information structure and content; information flow; data ownership; timeliness, latency, and age; references and mappings; transaction management and recovery; data quality; data volumes; archives and data retention; and regulation
- ***Models***: static data structure models, information flow models, information lifecycle models, data ownership models, data quality analysis, metadata models, and volumetric models
- ***Pitfalls***: data incompatibilities, poor data quality, unavoidable multiple updaters, key matching deficiencies, poor information latency, interface complexity, and inadequate volumetrics

Information - Example Model Fragments

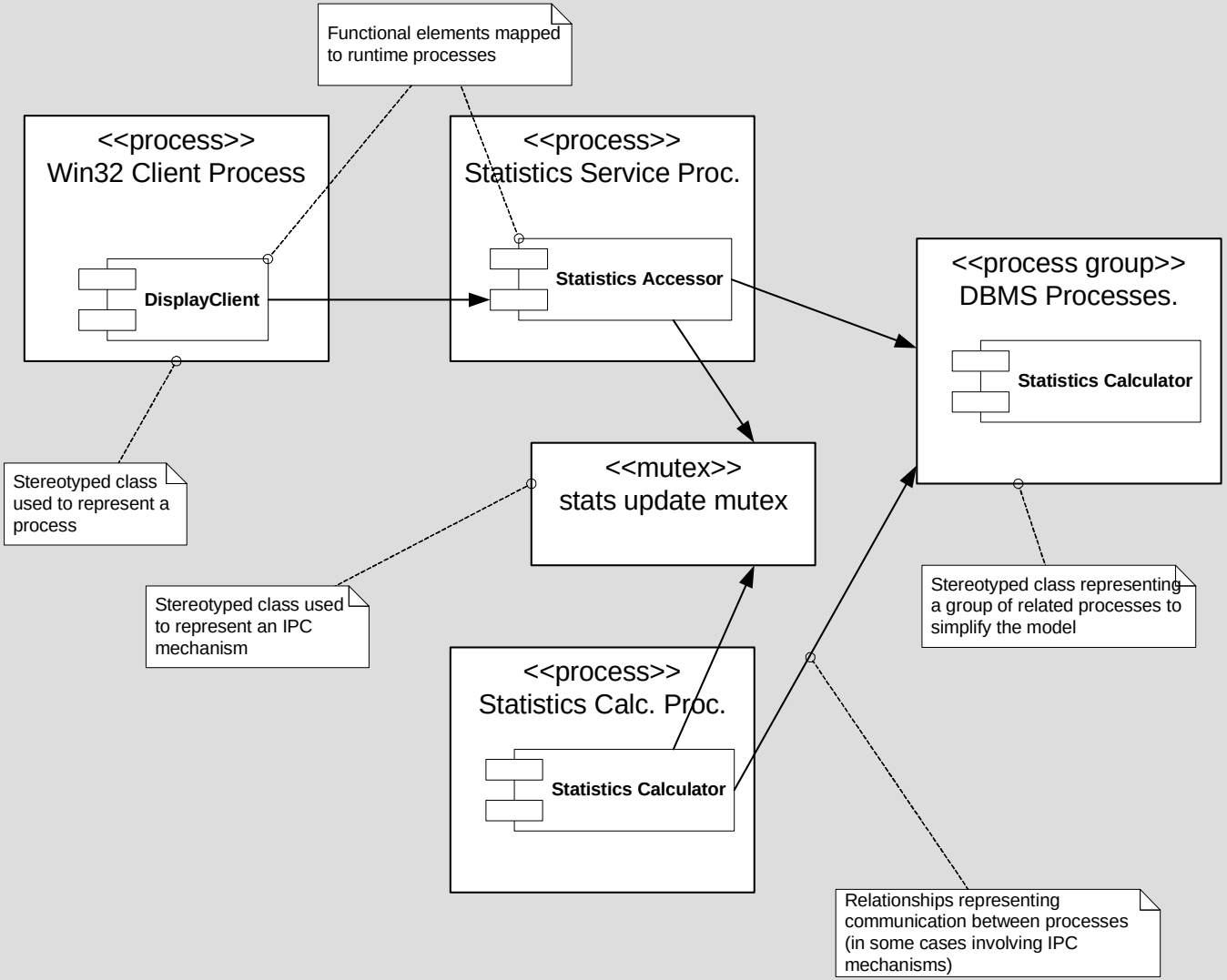


Concurrency

The packaging of the system into processes and threads

- ***Content***: the concurrency structure of the system, mapping 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, inter-process communication, state management, synchronization and integrity, startup and shutdown, task failure, and re-entrancy
- ***Models***: system-level concurrency models and state models
- ***Pitfalls***: modelling of the wrong concurrency, excessive complexity, resource contention, deadlock, and race conditions

Concurrency - Example Model Fragment

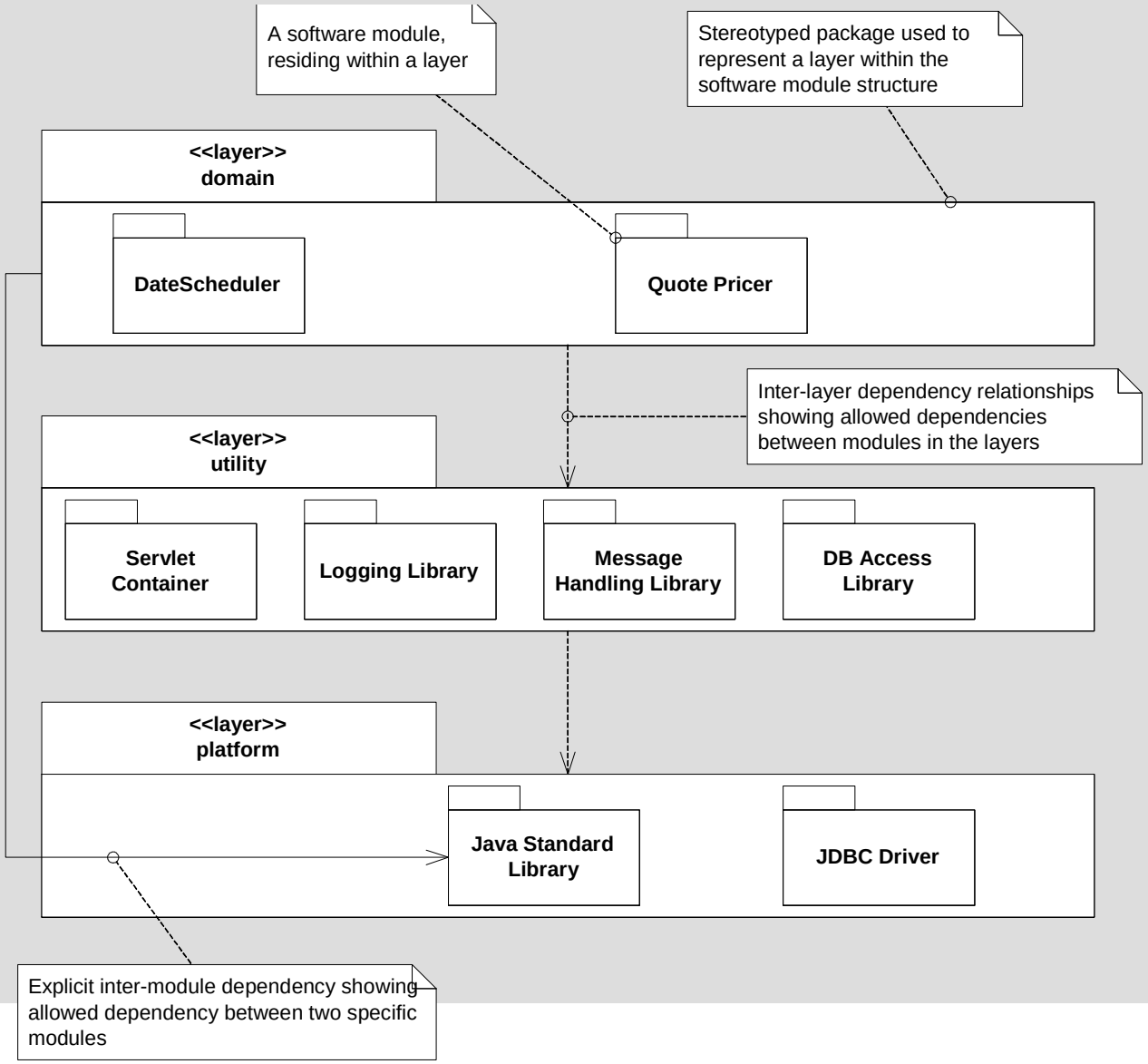


Development

The architectural constraints on the development process

- ***Content***: how the architecture supports and constraints 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
- ***Pitfalls***: too much detail, overburdening the AD, uneven focus, lack of developer focus, lack of precision, and problems with the specified environment

Development - Example Model Fragment

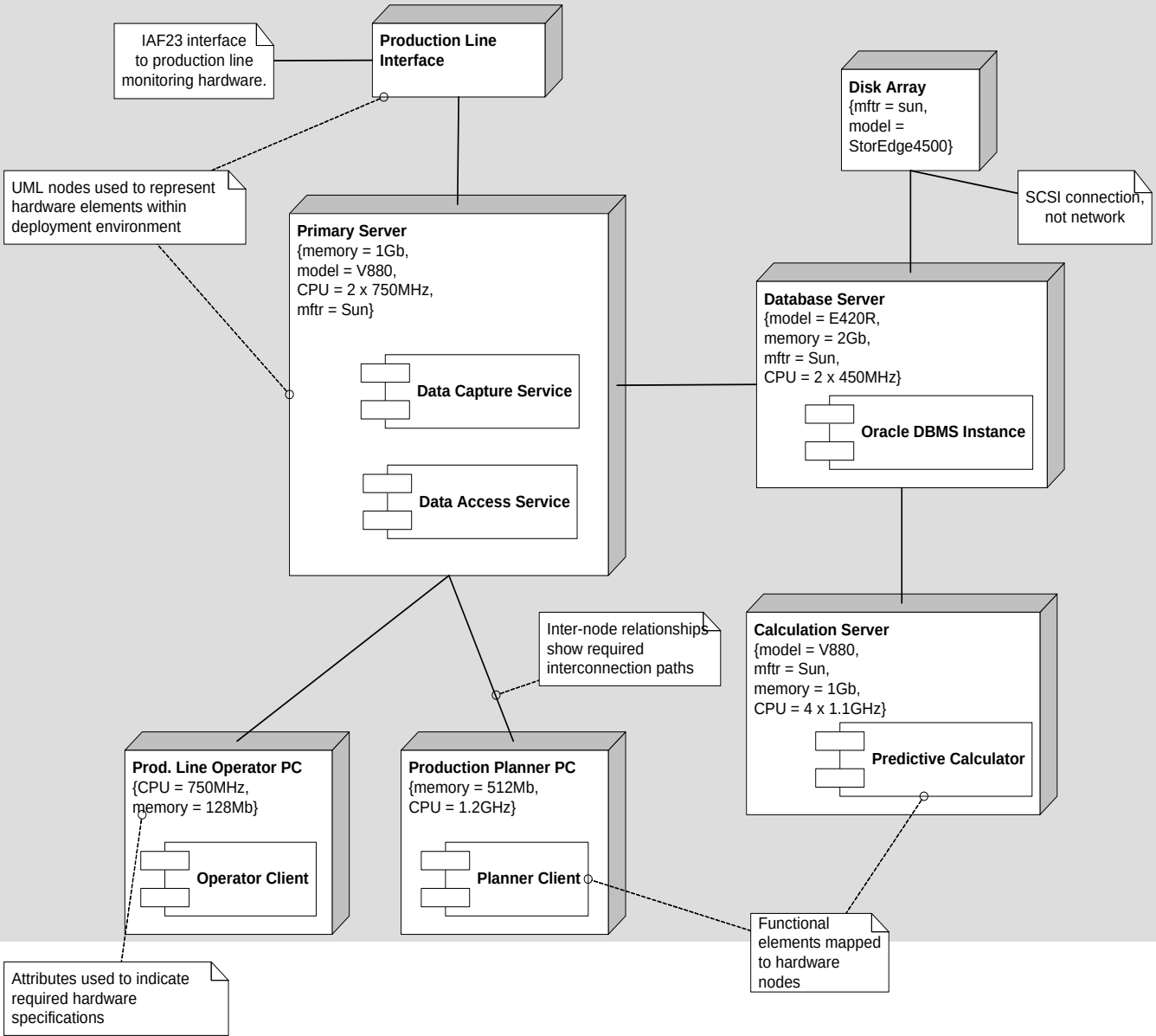


Deployment

The runtime environment and the distribution of software across it

- ***Content***: the environment into which the system will be deployed, including the dependencies the system has on its runtime environment
- ***Concerns***: types of hardware required, specification and quantity of hardware required, third-party software requirements, technology compatibility, network requirements, network capacity required, and physical constraints
- ***Models***: runtime platform models, network models, and technology dependency models
- ***Pitfalls***: unclear or inaccurate dependencies, unproven technology, lack of specialist technical knowledge, and late consideration of the deployment environment

Deployment - Example Model Fragment



Viewpoints and Views

Operational

How the system is installed, migrated to, run and supported

- ***Content***: 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, configuration management, performance monitoring, support, and backup and restore
- ***Models***: installation models, migration models, configuration management models, administration models, and support models
- ***Pitfalls***: lack of engagement with the operational staff, lack of backout planning, lack of migration planning, insufficient migration window, missing management tools, lack of integration into the production environment, and inadequate backup models

Operational Viewpoint Possible Content

- Installation Model
  - Installation groups
  - Dependencies and constraints
  - Backout strategy
- Operational CM Model
  - Configuration groups and dependencies
  - Configuration parameter sets
  - Operational control (switching between sets)
- Administration Model
  - Monitoring and control facilities required and provided
  - Required operational procedures and error conditions etc.

Viewpoints and Views

Viewpoints provide

- A store of knowledge and experience
- A guide to the architect
- Templates to guide the process

Views provide

- A structure for description
- A separation of concerns
- Improved stakeholder communication

Limitations of viewpoints

- Quality properties are critical
- Viewpoints don’t explicitly consider quality properties
- Quality properties usually need cross-viewpoint consideration
- Viewpoints may lead to late consideration of quality properties

Architectural Perspectives

Dealing with quality properties

- Use ***perspectives*** to guide the architect in achieving the required quality properties
  - Each perspective addresses one major quality property
- The perspectives guide ***changes*** to the ***views***

- A new approach, compatible with viewpoints
  - Related to SEI's "tactics" work
- Recent book from Addison Wesley:

  Software Systems Architecture: Working With Stakeholders Using Viewpoints & Perspectives

  Nick Rozanski & Eoin Woods

  April 2005

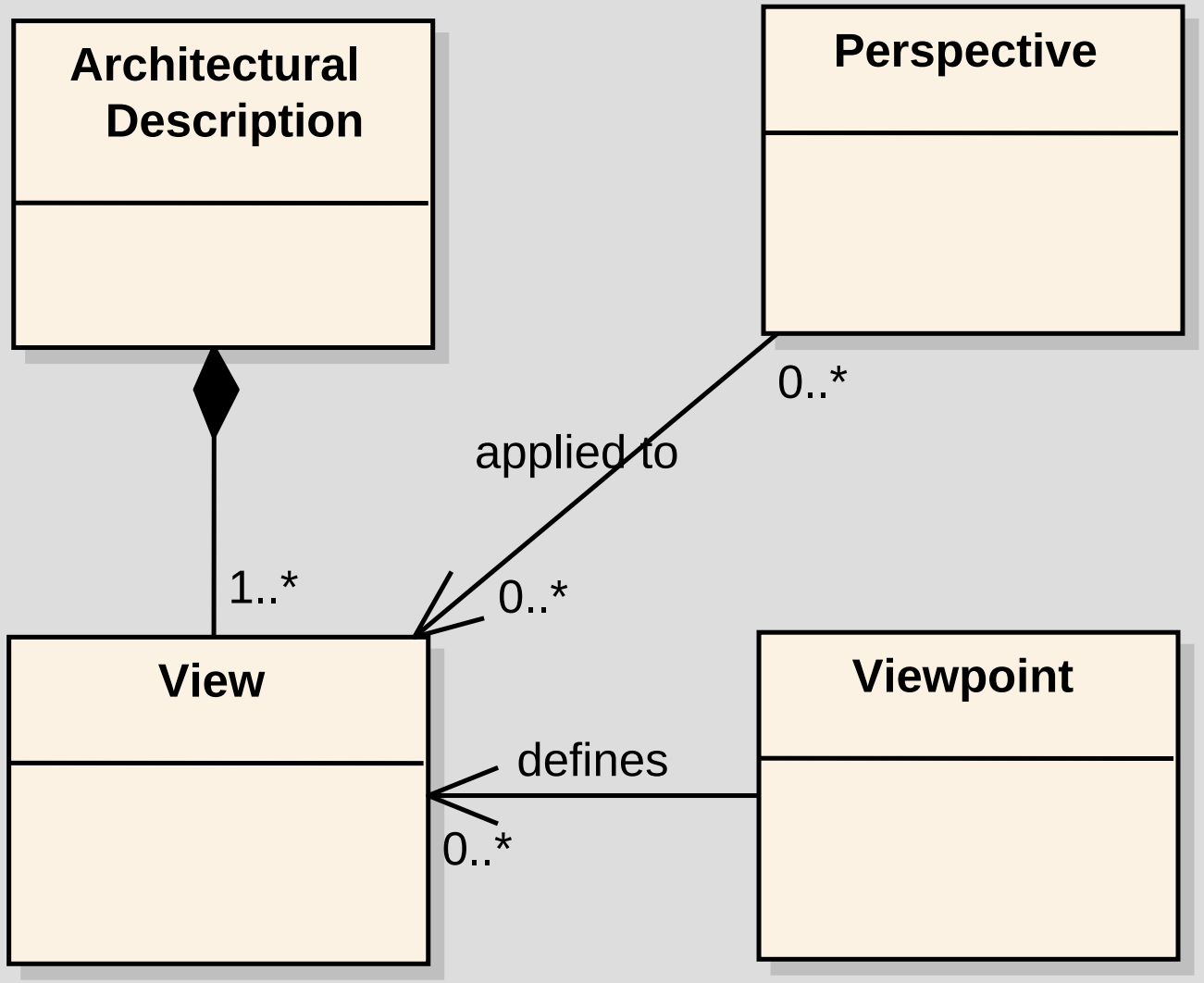
Architectural Perspectives

Defining perspectives

*Architectural **perspective** is a collection of **activities, checklists, tactics and guidelines** to **guide the process** of ensuring that a system **exhibits** a particular set of closely related **quality properties** that require consideration across a number of the system's architectural views.*

Rozanski and Woods, 2005

Adding perspectives to the inter-relationships



Architectural Perspectives

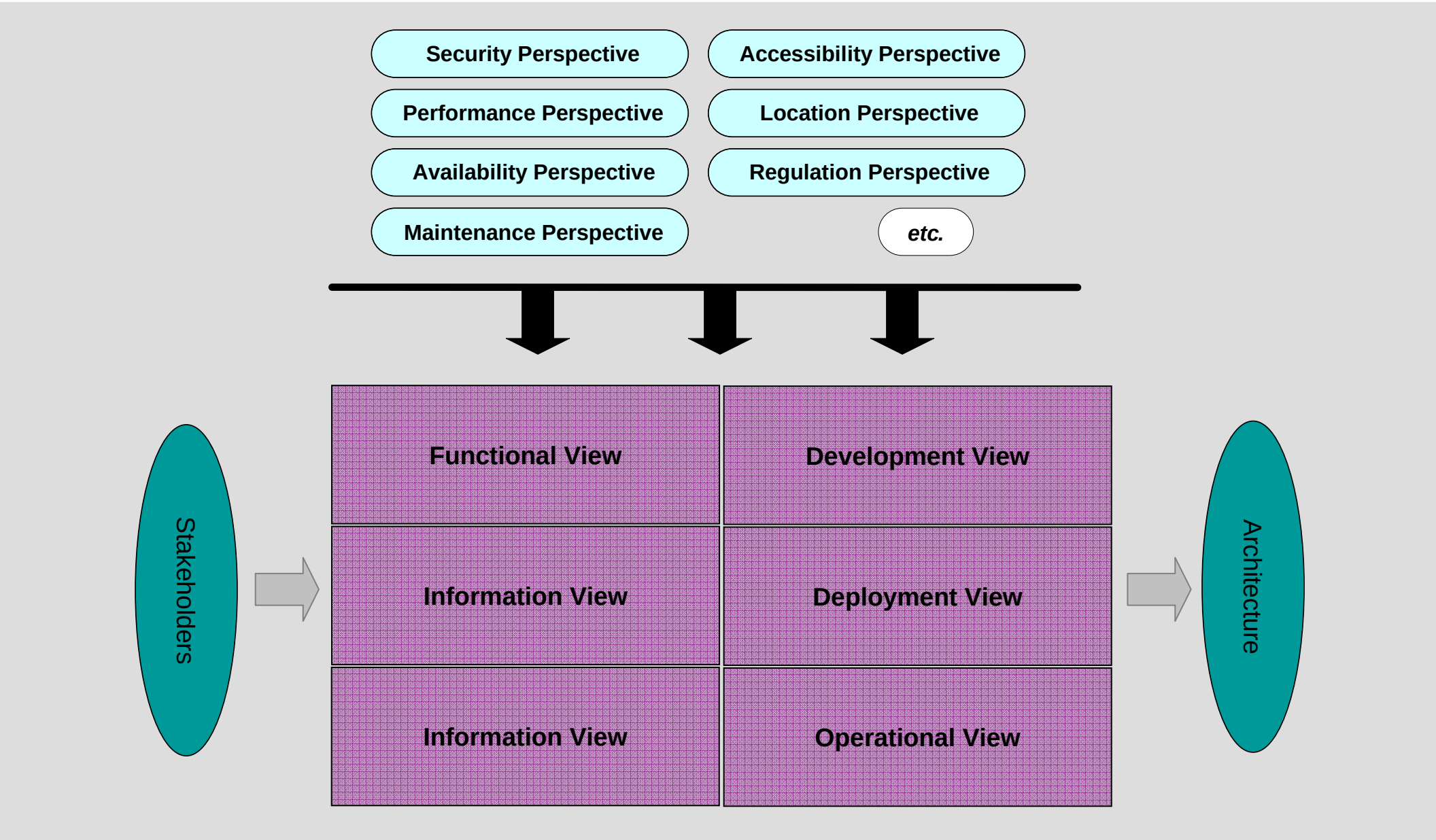
A simple but effective idea

- A store of knowledge and experience
- A guide to the architect
- Templates to guide the process

Analogous to viewpoints but for quality properties

Perspectives “applied” to the views to ensure acceptable qualities and guide changes where required

Architectural Perspectives with Viewpoints



Architectural Perspectives

Our initial core set

- Performance and Scalability
- Security
- Availability and Resilience
- Evolution
- Also: Location, I18N, Usability, Regulation, …

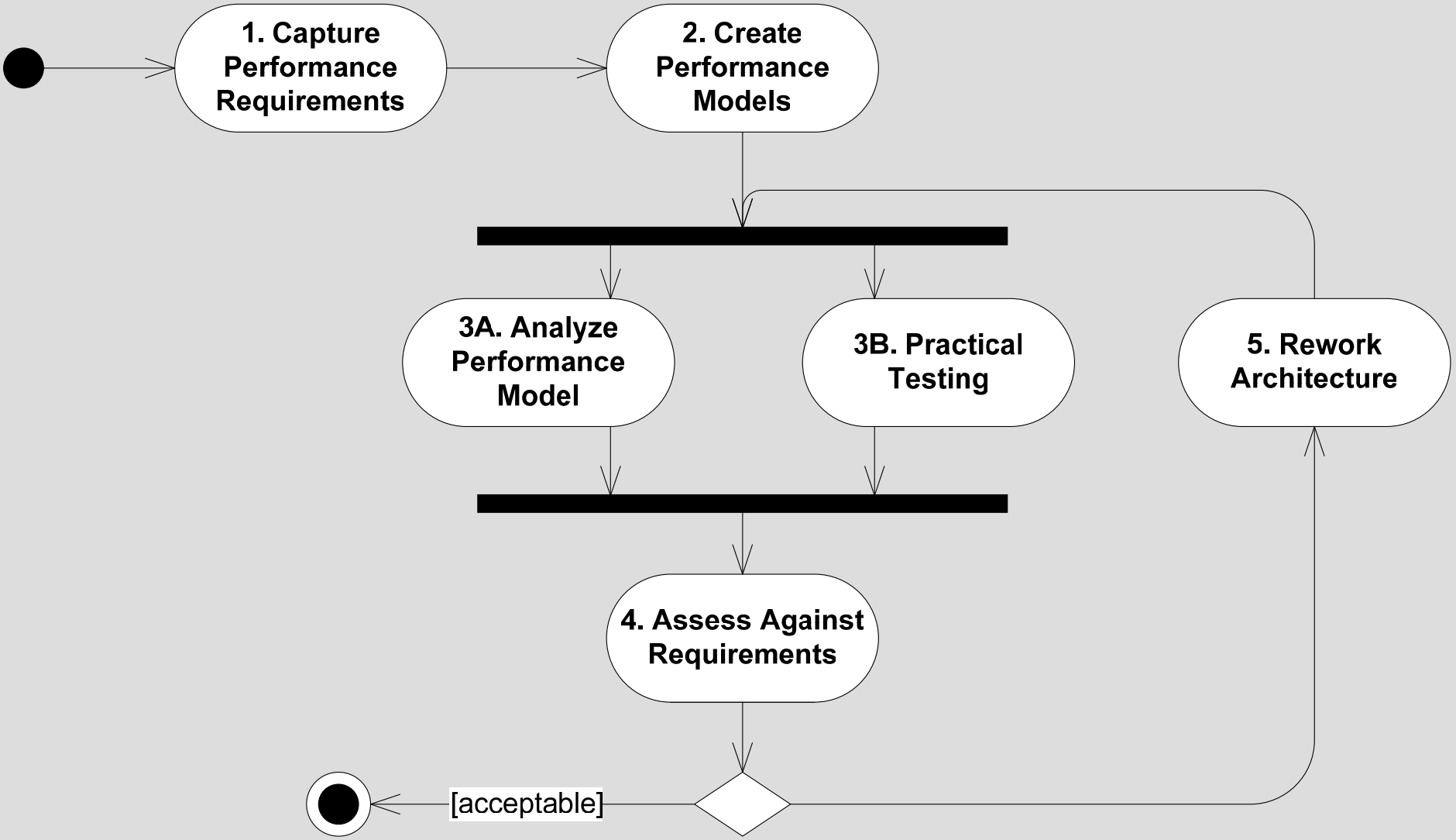
Different sets in different domains

Architectural Perspectives

Performance and Scalability

- ***Required Quality***: the ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes
- ***Concerns***: processing volume, response time, responsiveness, throughput, predictability
- ***Tactics***: Optimize repeated processing, reduce contention via replication, prioritize processing, consolidate related workloads, distribute processing over time, minimize the use of shared resources, partition and parallelize, use asynchronous processing, and make design compromises
- ***Pitfalls***: Imprecise goals, unrealistic models, use of simple measures for complex cases, inappropriate partitioning, invalid environment and platform assumptions, too much indirection, concurrency-related contention, careless allocation of resources, …

Performance and Scalability Perspective Activities

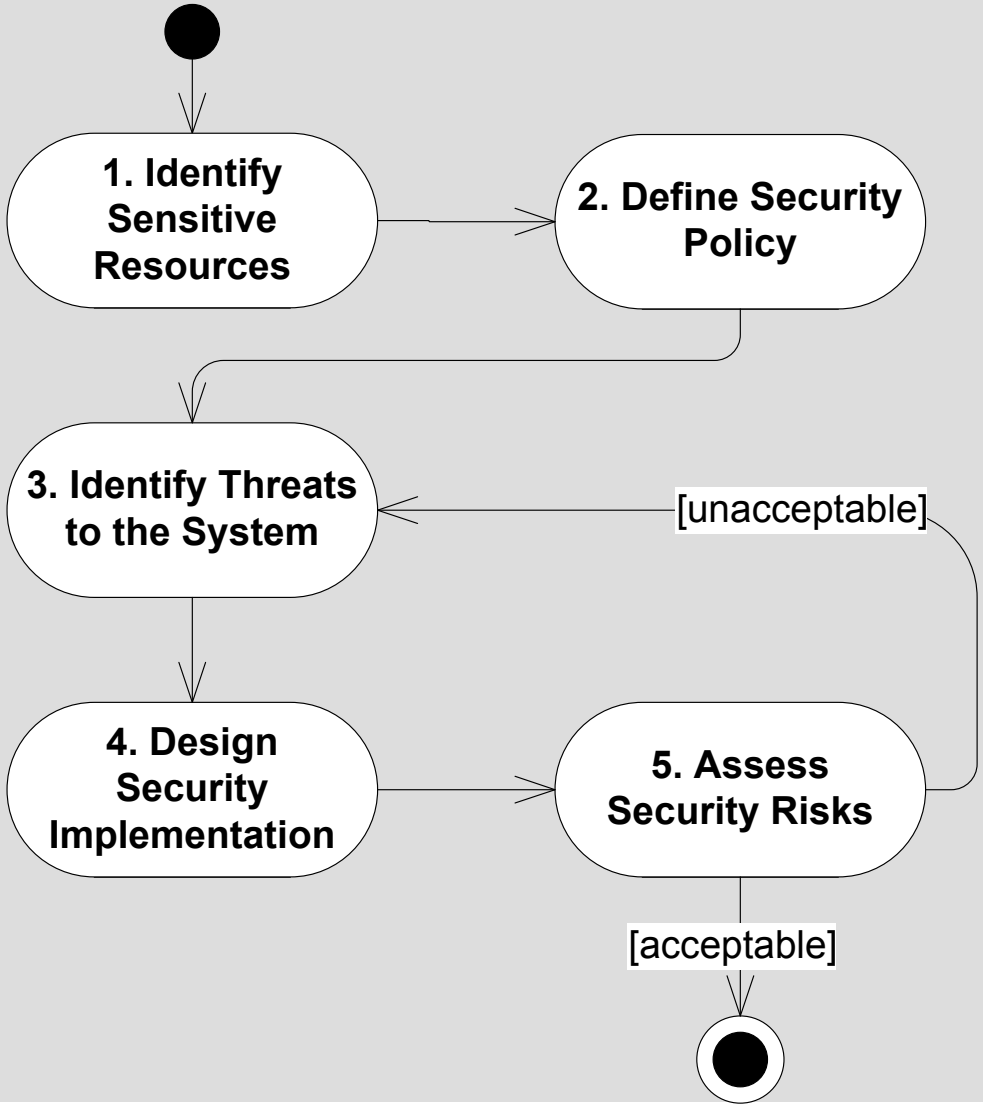


Architectural Perspectives

Security

- ***Required Quality***: the ability of the system to reliably control, monitor, and audit who can perform what actions on these resources and the ability to detect and recover from failures in security mechanisms
- ***Concerns***: Policies, threats, mechanisms, accountability, availability, and detection and recovery
- ***Tactics***: threat identification, threat assessment, vulnerability analysis, application of security technology
- ***Pitfalls***: Complex security policies, unproven security technologies, system not designed for failure, lack of administration facilities, technology-driven approach, failure to consider time sources, over reliance on technology, no clear requirements or models, security as an afterthought, security embedded in the application code, piecemeal security, and ad hoc security technology

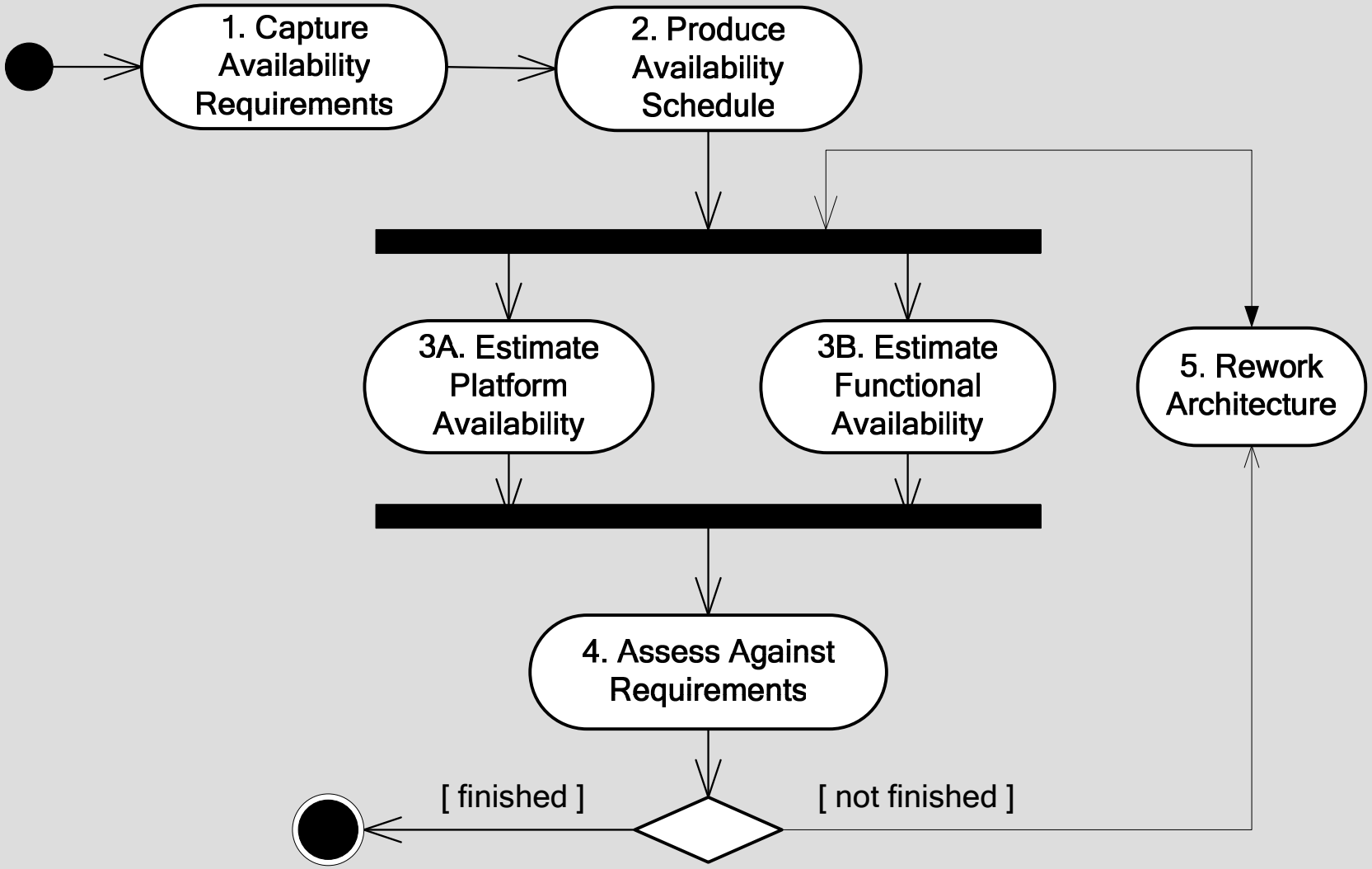
Security Perspective Activities



Availability and Resilience

- ***Required Quality***: 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
- ***Concerns***: classes of service, planned / unplanned downtime, mean time between failures, mean time to repair, disaster recovery, redundancy, clustering, failover
- ***Tactics***: MTBF and MTTR prediction, availability schedules, availability models, availability technology application
- ***Pitfalls***: Single point of failure, overambitious availability requirements, ineffective error detection, overlooked global availability requirements, and incompatible technologies

Availability and Resilience Perspective Activities

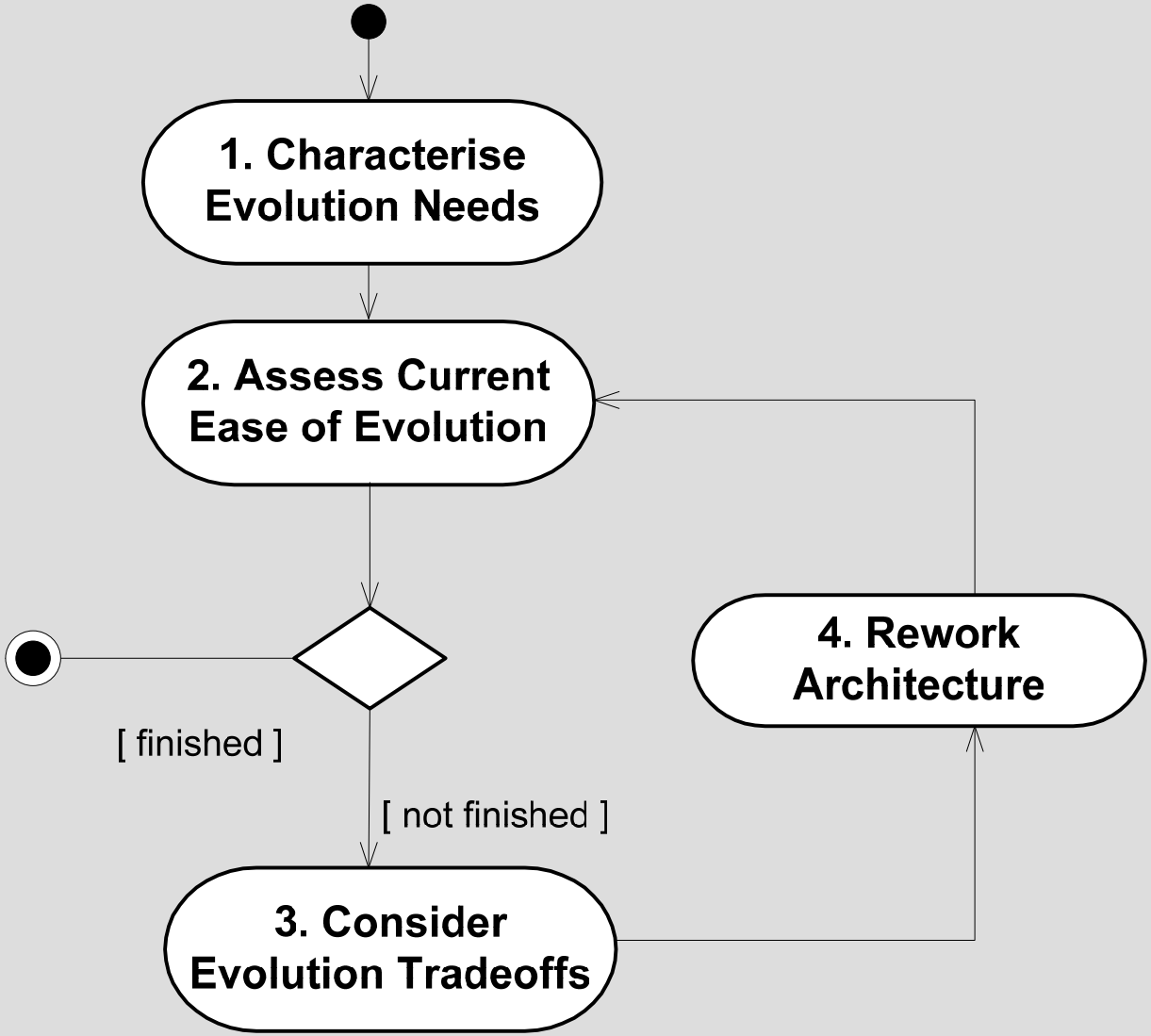


Architectural Perspectives

Evolution

- ***Required Quality***: 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
- ***Concerns***: flexibility, extensibility, functional evolution, deployment evolution, integration evolution
- ***Techniques***: design for change, architectural assessment, configuration management, automated testing, build and release management
- ***Pitfalls***: prioritization of the wrong dimensions, changes that never happen, impacts of evolution on critical quality properties, lost development environments, and ad hoc release management

Evolution Perspective Activities



Architectural Perspectives

Accessibility

- Can the system be used by people with disabilities?

Development Resource

- Can the system be built within people, time, budget constraints?

Internationalisation

- Is the system independent of language, country and culture?

Location

- Will the system work, given its required geographical constraints?

Regulation

- Does the system meet required regulatory constraints?

Usability

- Can people use the system effectively?

Uses of Viewpoints and Perspectives

Viewpoints and Perspectives can

- Mentor *novice* architects

- Guide *working* architects

- Support *expert* architects

Uses of Viewpoints and Perspectives

For Novice Architects

- An introduction to each area of knowledge
- A guide to what is important
- A structure for the process
- Definitions of standards and norms
- Repository of proven practice and tactics
- Checklist to ensure nothing is forgotten

Uses of Viewpoints and Perspectives

For Working Architects

- A reminder of what is important

- A guide to new or rarely used areas of practice

- Repository of proven practice and tactics

- Checklist to ensure nothing is forgotten

Uses of Viewpoints and Perspectives

For Expert Architects

- A framework to allow knowledge sharing

- An aid to tutoring and mentoring

- Checklist to ensure nothing is forgotten

Uses of Viewpoints and Perspectives

Viewpoints and Perspectives can

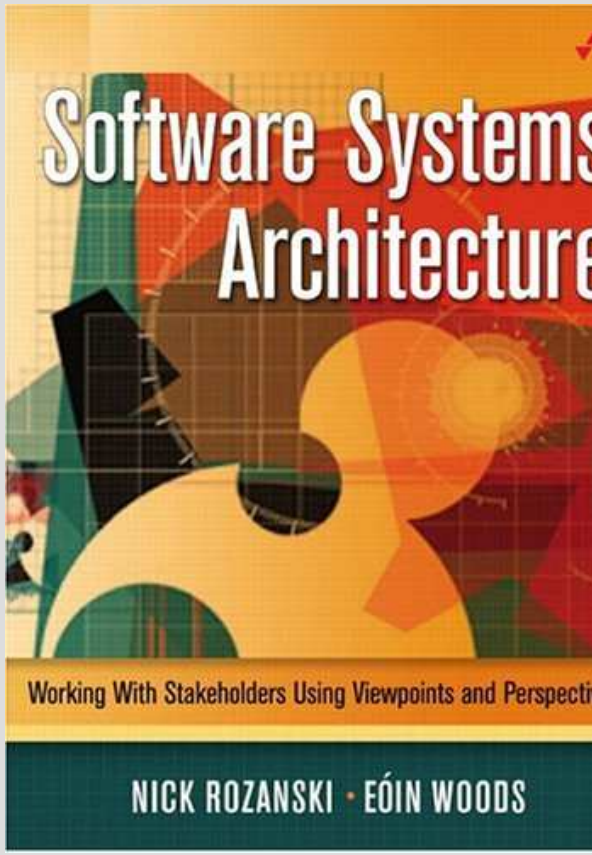
- Provide a framework for sharing knowledge
- Act as a store of architectural knowledge
  - Document proven practice
  - Help standardise language and approach
  - Help to standardise languages and approaches
- Act as a tutorial for new architects
- Act as a guide for working architects
- Act as aide-memoir for experienced architects

In Case We Didn‘t Mention It!

Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives

Nick Rozanski and Eoin Woods
Addison Wesley 2005

http://www.viewpoints-and-perspectives.info



Nick Rozanski
nick@rozanski.com
www.nick.rozanski.com

Eoin Woods
eoin@copse.org.uk
www.eoinwoods.info

Comments and Questions?