
Software-Architecture, Design Patterns and Refactoring – An Overview –

Ingolf H. Krueger
ikrueger@ucsd.edu

Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA



California Institute for Telecommunications
and Information Technologies
La Jolla, CA 92093-0405, USA



Overview

- The Notion of Architecture and Architectural Aspects
- The Role of Software Architecture
- Modeling and Documenting Architectures
- Patterns for Architecture and Design
- Refactoring Techniques

The Notion of Architecture

A **Software-Architecture** describes the decomposition of a system into

units / components / subsystems,

and their

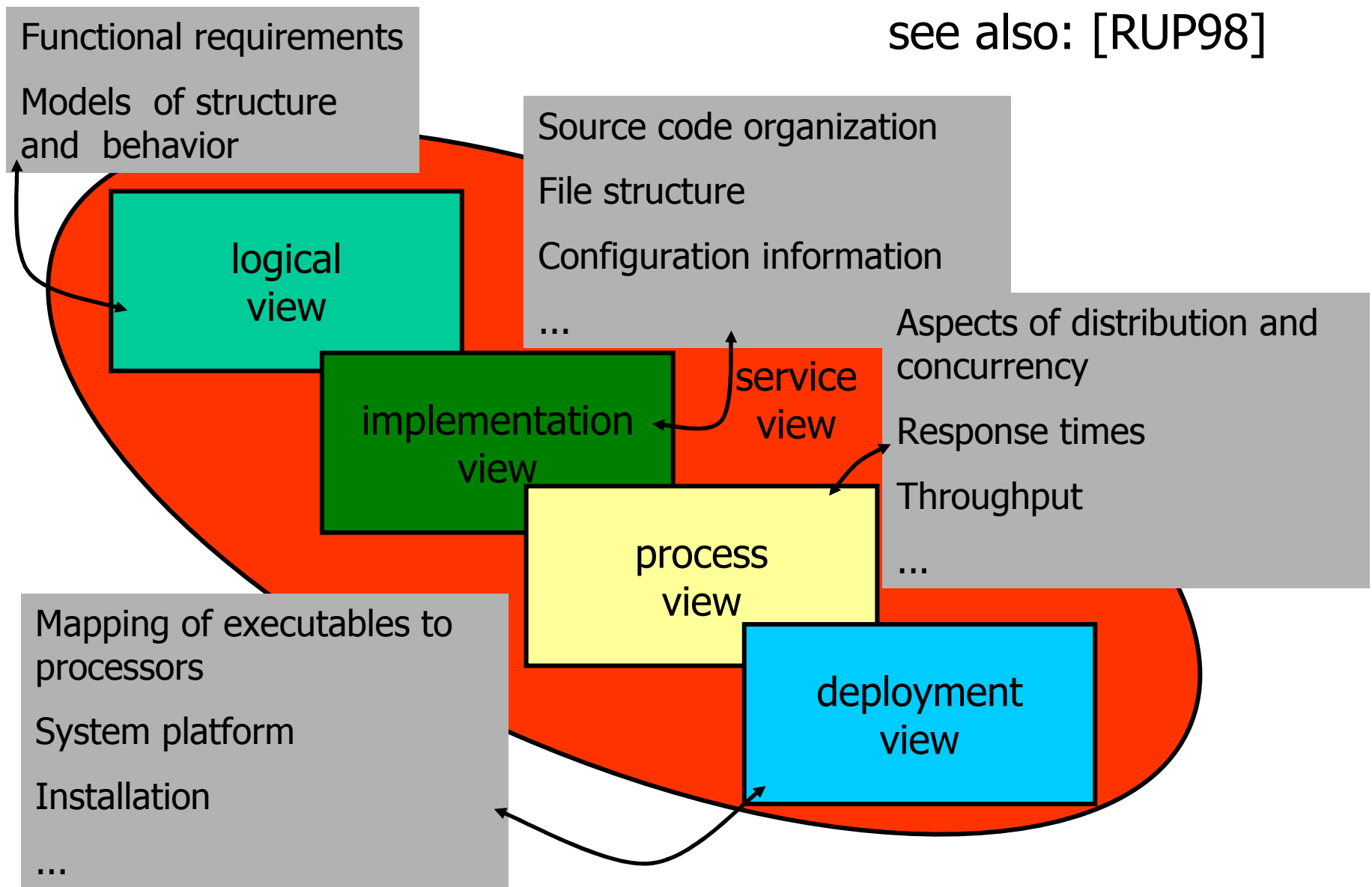
connections / interactions / relationships

observing

quality requirements / design guidelines / constraints

Architectural Aspects

see also: [RUP98]



Overview

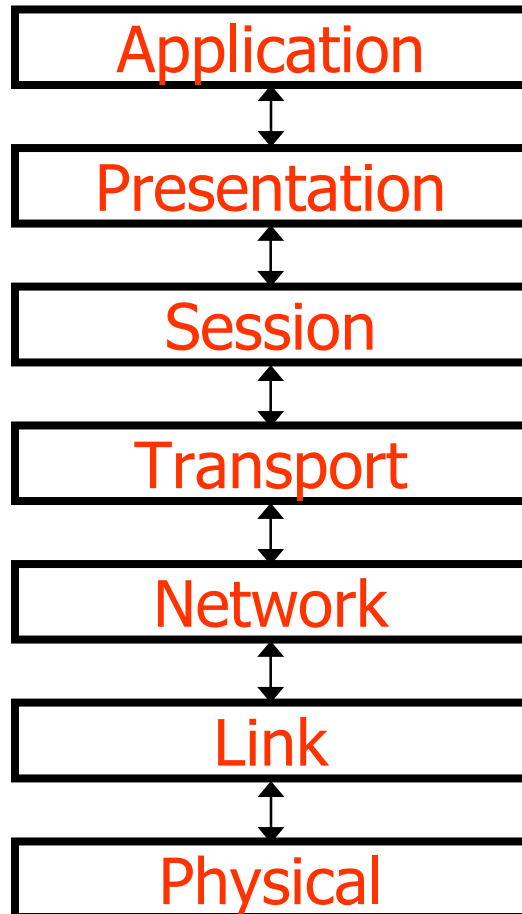
- The Notion of Architecture and Architectural Aspects
- The Role of Software Architecture
- Modeling and Documenting Architectures
- Patterns for Architecture and Design
- Refactoring Techniques

The Role of SW-Architecture

- Bounds leeway for implementation
- Fosters (or impedes!) system quality
- Supports the critical system services
- Defines starting point for
 - Change management
 - Product families
 - Division of work

The Role of SW-Architecture

The Classic: ISO/OSI-Layered Architecture

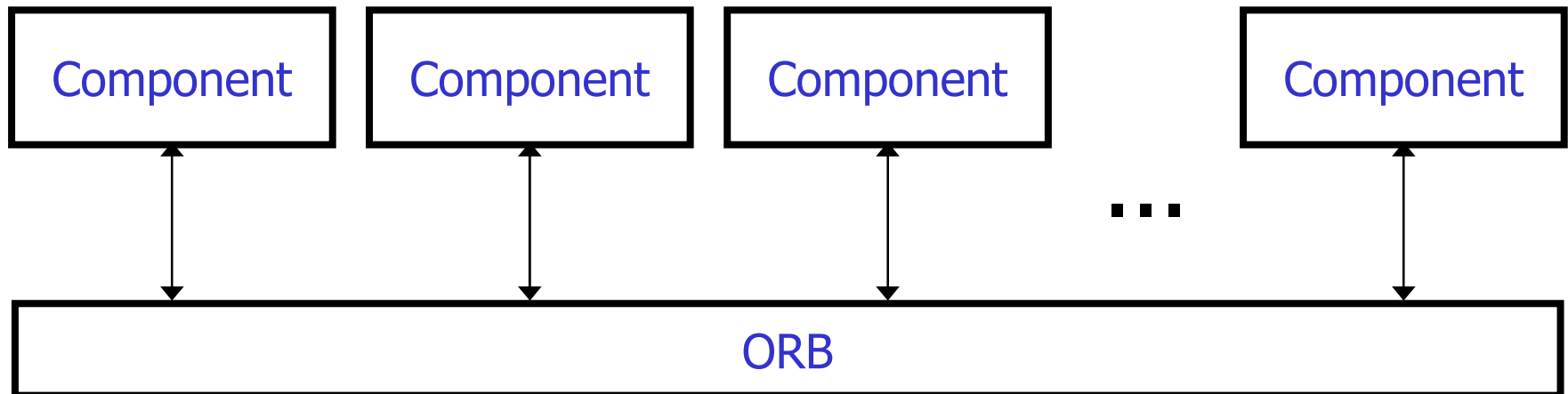


- Clear assignment of responsibilities to layers
 - What functionality does the component perform and where is it located?
 - Who develops what part of the system?
 - Positioning of business models
- Clear interfaces between layers
- Call constraints (no “skipping” of layers)

The Role of SW-Architecture

"CORBA" (simplified)

- Clear interfaces, responsibilities, decoupling
- **Infrastructure** for communication, distribution
- Implementation strategy



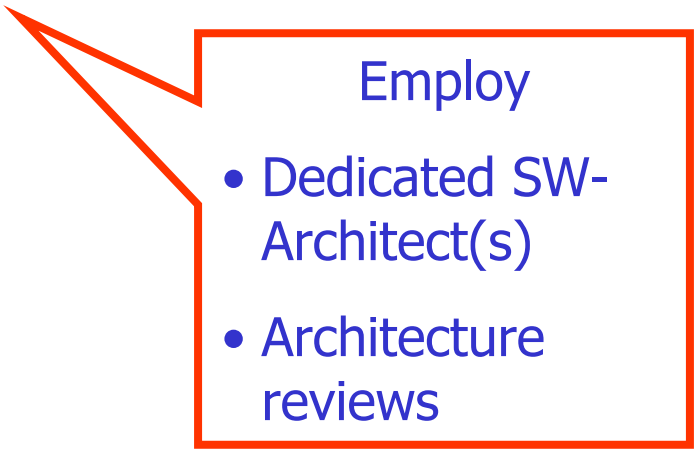
The Role of SW-Architecture

- The selection of an adequate architecture is a key success factor in system design
 - Transparently structured architecture is basis for:
 - Project organization
 - Complexity management
 - Reuse
 - Component- and service-oriented development
 - System comprehension beyond team boundaries
 - Manageable system evolution
- ⇒ Make architecture description part of every software project!

The Role of SW-Architecture

A good SW-Architecture

- Brings stability to the system
 - ⇒ small modifications to the requirements induce small modifications to the system
- Has “conceptual integrity”
 - ⇒ Balance
 - ⇒ Simplicity
 - ⇒ Elegance, Clarity
 - ⇒ Practicality



Employ

- Dedicated SW-Architect(s)
- Architecture reviews

Overview

- The Notion of Architecture and Architectural Aspects
- The Role of Software Architecture
- Modeling and Documenting Architectures
- Patterns for Architecture and Design
- Refactoring Techniques

Modeling and Documenting Architectures

How to model and implement

- *Units / components / subsystems,*
- *Connections / interactions / relationships,*
- *Quality attributes / development guidelines / constraints*

adequately?

⇒ Approaches:

- *Architecture Description Languages (ADLs)*
- *Architectural styles and patterns*
- *Domain specific architectures*
- *Infrastructures*

Modeling



Implementation

Modeling and Documenting Architectures

How to model and implement

- *Units / components / subsystems*
- *Connections / interactions*
- *Quality attributes / development*

adequately?

- Unicon/Wright (Garlan et al., CMU)
- Darwin (Kramer et al., Imperial College)
- Rapide (Luckham et al., Stanford)
- UML-RT (Rational)?!
- Service-ADL (Krueger et al., UCSD)
- ...

⇒ Approaches:

- *Architecture Description Languages (ADLs)*
- *Architectural styles and patterns*
- *Domain specific architectures*
- *Infrastructures*

Modeling



Implementation

Modeling and Documenting Architectures

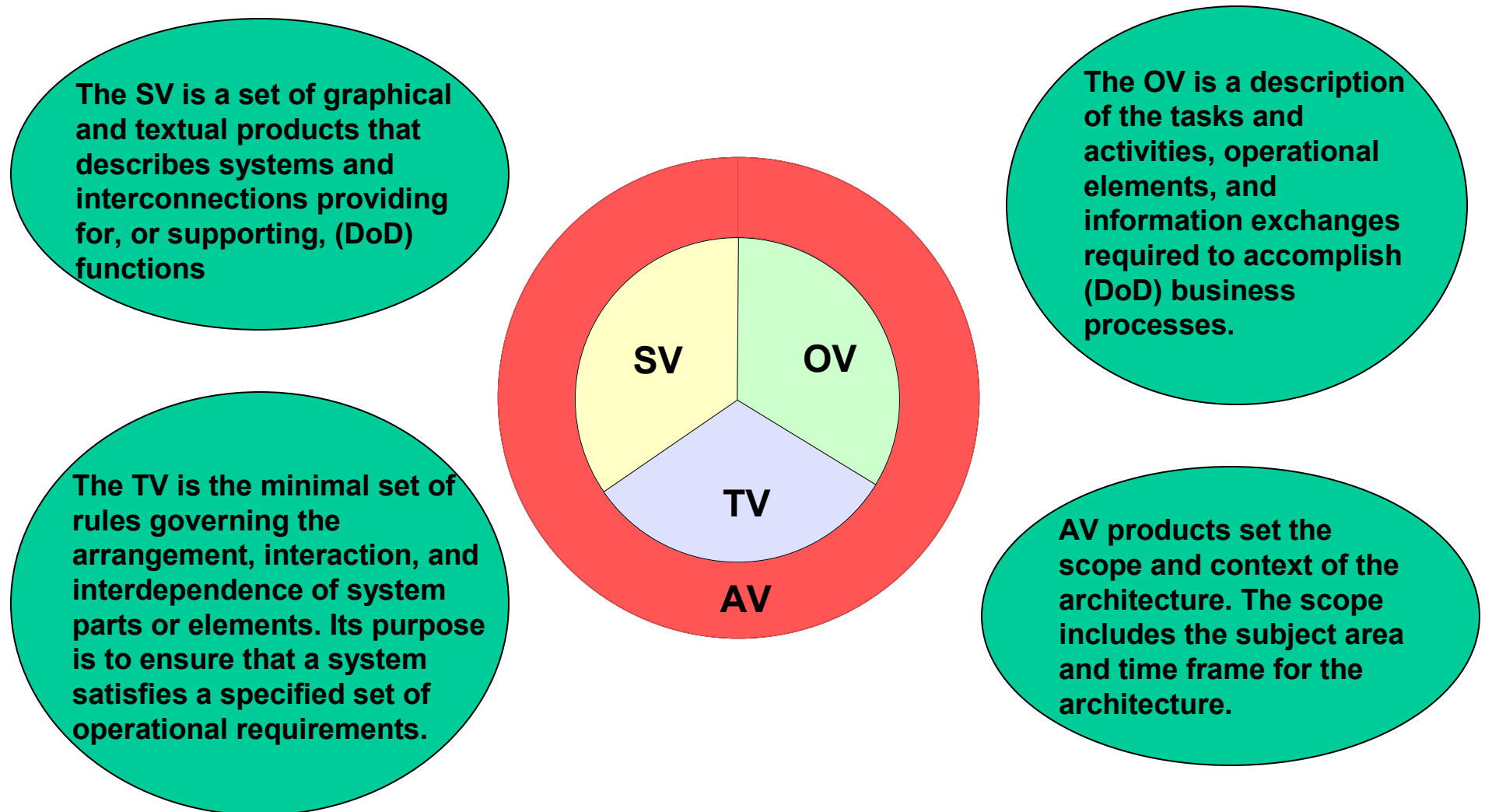
- Architectural styles and patterns:
 - Pipes and Filters
 - Layered Architecture
 - Model-View-Controller
 - Broker, ...
- Domain specific architectures:
 - Telecommunication
 - Standard architectures in the business domain
 - Avionics, automotive
- Manifestations of architectures:
 - Middleware technologies (CORBA, DCOM, .NET, Jini, ...)
 - Operating systems
 - Databases

Modeling and Documenting Architectures

Content of an “ad-hoc” architecture document (example)

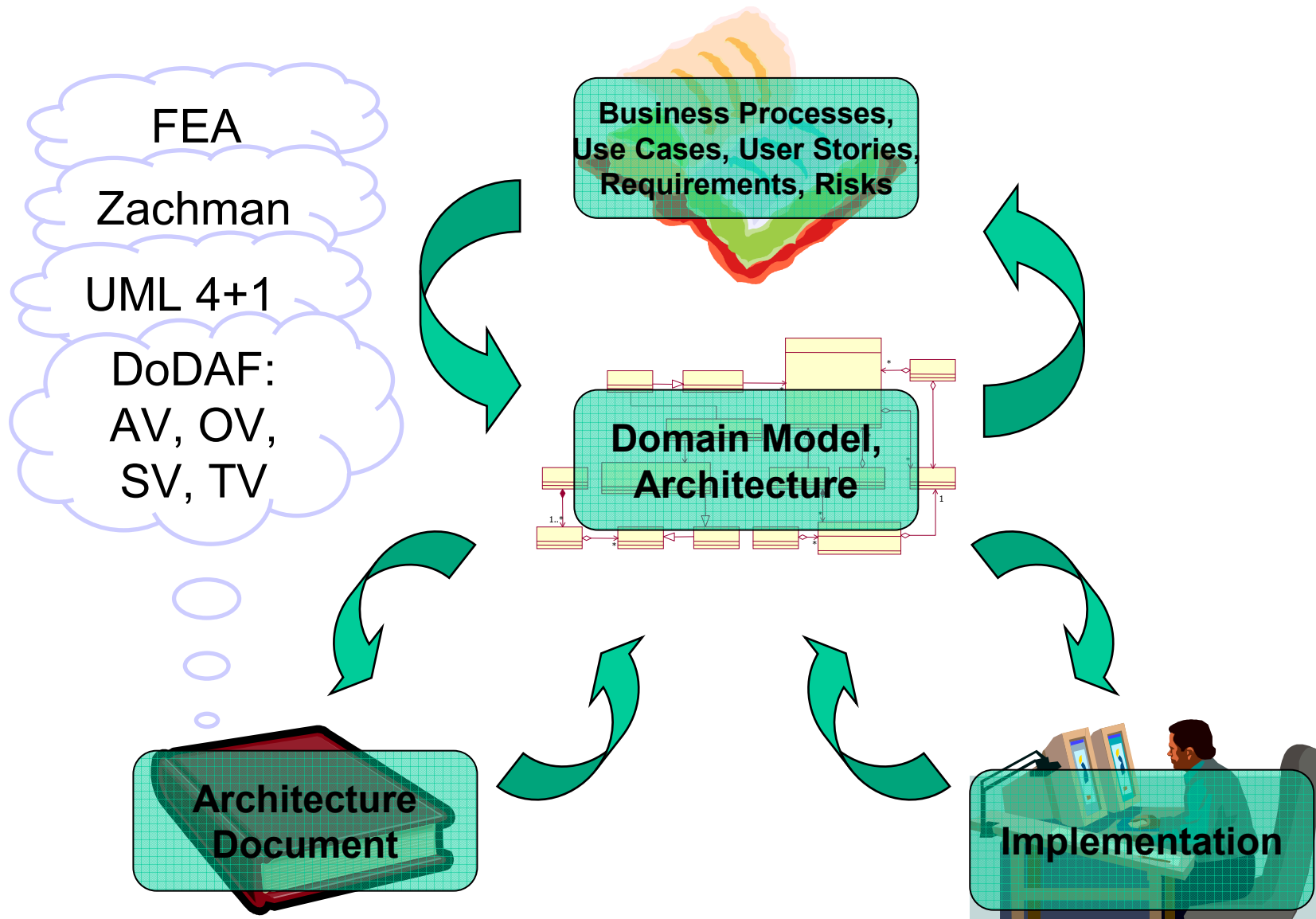
- Short description of the system
- Essential use cases (including exceptions)
- Component structure
(possibly taken from the domain model)
- Interaction patterns
- Rationale for the architecture’s adequacy
(along the analysis model)
- other relevant views
(process view, distribution view, ... – see above)

DOD Architecture Framework V1.0 Views and Products



see, for instance, <http://www.software.org/pub/architecture/dodaf.asp> and <http://www.aitcnet.org/dodfw/>

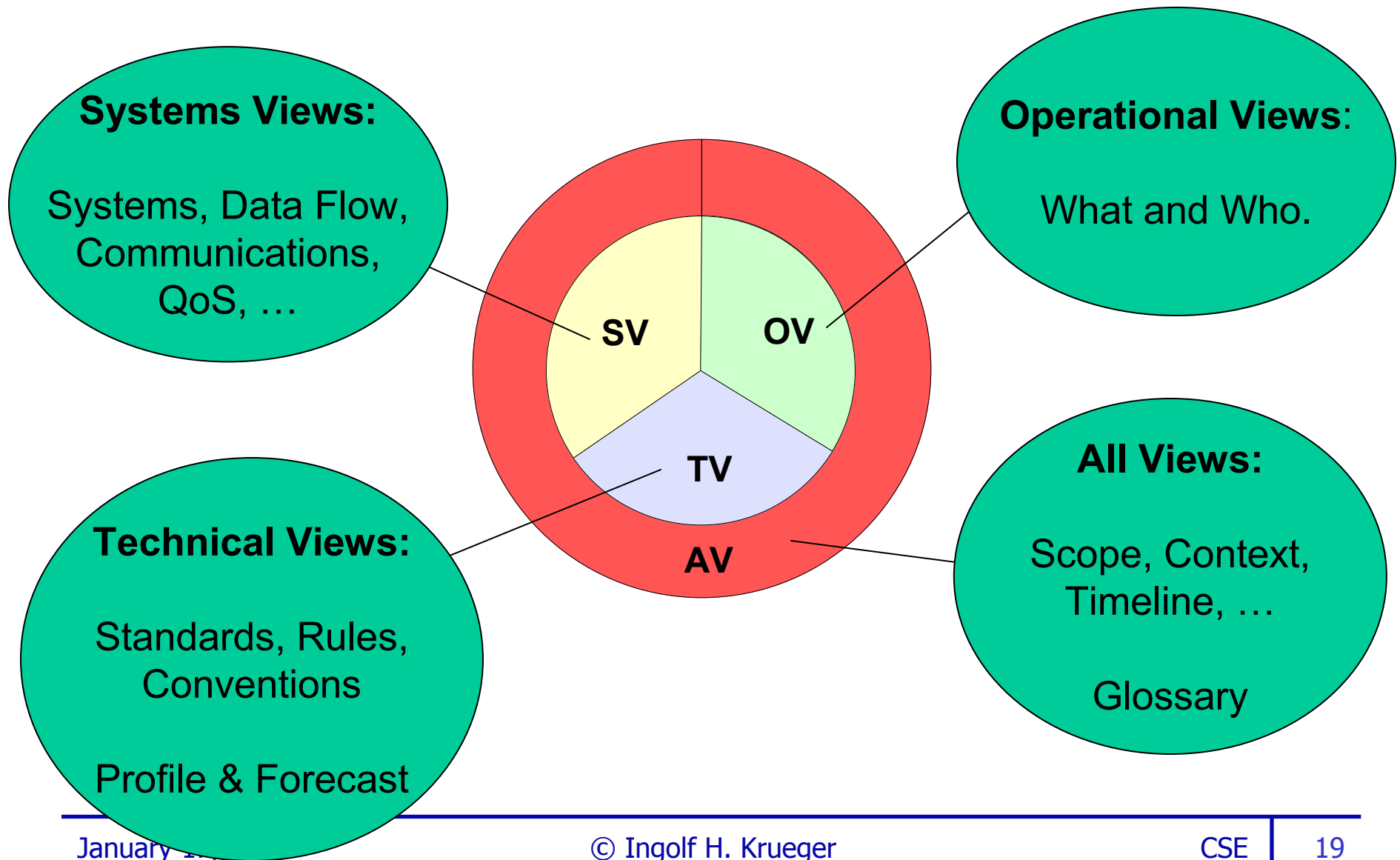
Architecture vs. Process vs. Documentation Standard



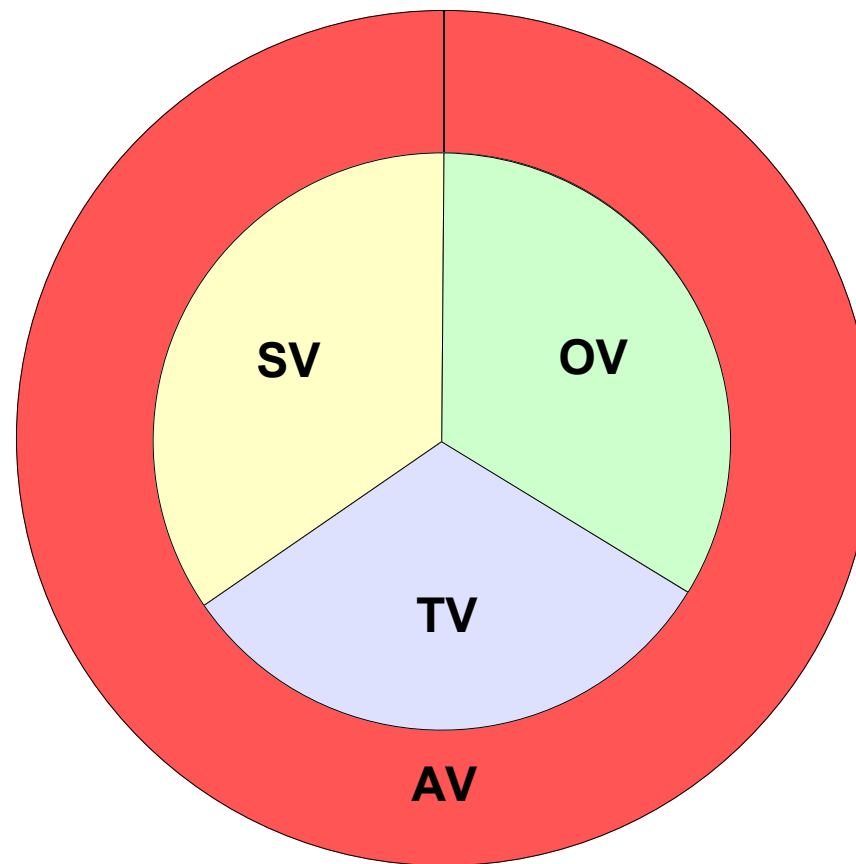
Architecture Standard Followed

- DoD Architecture Framework 1.0
- References:
 - DoD Architecture Framework Volume I, 15 August 2003
 - DoD Architecture Framework Volume II, 15 August 2003
 - DoD Architecture Framework Deskbook, 15 August 2003
- Covers broad spectrum of architectural aspects
- Architecture product selection based on relevance and availability of information
 - Operational Views
 - System Views
 - Technical Views

DoD Architecture Framework Views



DoDAF Products



Product Selection and Rationale

- = Product is highly applicable
- ⊙ = Product is often or partially applicable
- = Product is specifically addressed in policy
- ▒ = Product is required for an integrated architecture
- blank = Product is usually not applicable

APPLICABLE ARCHITECTURE PRODUCTS

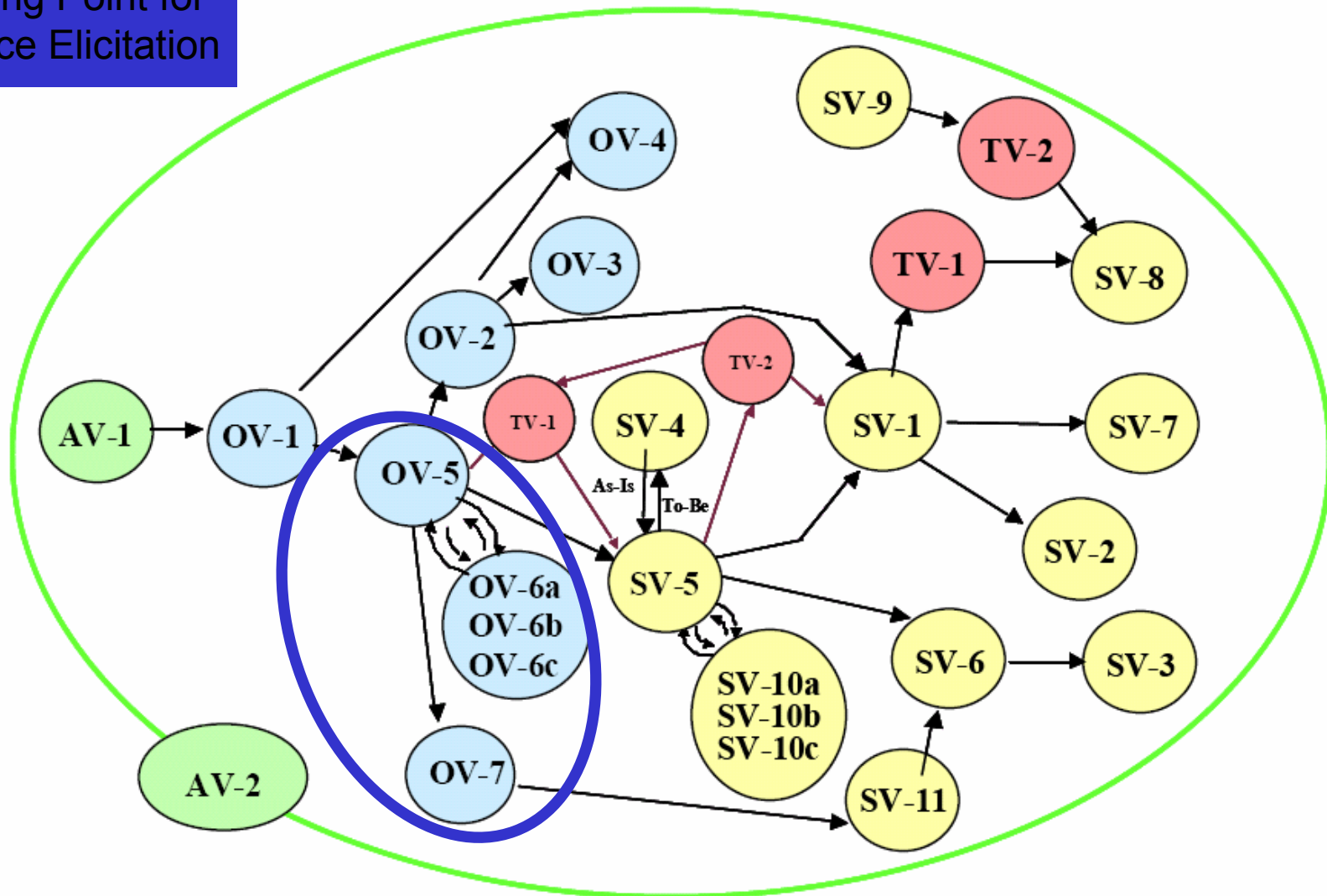
All View		Operational View (OV)							Systems View (SV)											Tech Stds View	
1	2	1	2	3	4	5	6	7	1	2	3	4	5	6	7	8	9	10	11	1	2

RECOMMENDED USES OF ARCHITECTURE:

Planning, Programming, Budgeting Execution Process																					
Capability-Based Analysis for IT Investment Decisions	●	●	●	●	●	●	●	⊙	●	⊙	●	●	●	●	●	●	●	●	●	●	⊙
Modernization Planning and Technology Insertion/Evolution	●	●	⊙	●	⊙	●	⊙		●	⊙	⊙	⊙	●	⊙	●	●	●			●	●
Portfolio Management	●	●		●	▒		●	⊙	●			⊙	●		⊙	●				⊙	
Joint Capabilities Integration and Development System																					
JCIDS Analysis (FAA, FNA, FSA)	●	●	●	●	⊙	⊙	●	●		●	⊙		⊙	●						⊙	
ICD/CDD/CPD/CRD	●	●	●	●	●		●	●		●	⊙	●	●	●	⊙	⊙	⊙	●		●	⊙
Analysis of Alternatives (AoA)	●	●	●	●	⊙		●	●		●	⊙	⊙	●	●	⊙	⊙	⊙			⊙	⊙
Acquisition Process																					
Acquisition Strategy	●	●	●	●	⊙		●	⊙		●	⊙			●						●	
C4ISP	●	●	●	●	●		●	●		●			⊙		●	⊙				●	⊙
System Design and Development	●	●		●	●		●	⊙	●	●	●	●	●	●	●	⊙	⊙	⊙	⊙	●	⊙
Interoperability and Supportability of NSS and IT Systems	●	●	●	●	⊙		●	⊙	⊙	●	●	●		●	●	⊙	⊙	⊙	⊙	●	⊙
Integrated Test & Evaluation	●	●		●	●	⊙	●	●	⊙	●	●	●	●	●	●			●	⊙	●	
Operations (Assessment, Planning, Execution, ...)																					
Operations Planning & Execution	●	●	●	●	●	●	●	⊙	●	●	⊙	⊙	●	⊙	⊙					⊙	
CONOPS & TTP	●	●	●	●	●	●	●		●	⊙	⊙	⊙	⊙							⊙	
Communications Plans	●	●	●	●	▒	⊙	⊙		●	●						⊙	⊙			●	⊙
Exercise Planning & Execution	●	●	●	●	●	●	●		●	●	⊙	⊙	⊙	⊙						⊙	
Organizational Design	●	●	●	●	●	●	⊙	⊙	⊙	⊙			⊙							▒	
BPR/FPI	●	●	⊙	●	●	●	●	⊙	▒											▒	

Iterative, Data-Centric Build Sequence

Starting Point for
Service Elicitation



Domain Model

- Informs most of the architecture products
- Central **data** and **processing entities**, and their structural **relationships**
- Induces **communication links**
- Supports **concepts of operation**
- Basis for determining component structure and distribution, interface definition
- OV-7, OV-5, OV-6

Overview

- The Notion of Architecture and Architectural Aspects
- The Role of Software Architecture
- Modeling and Documenting Architectures
- Patterns for Architecture and Design
- Refactoring Techniques

What are Patterns?

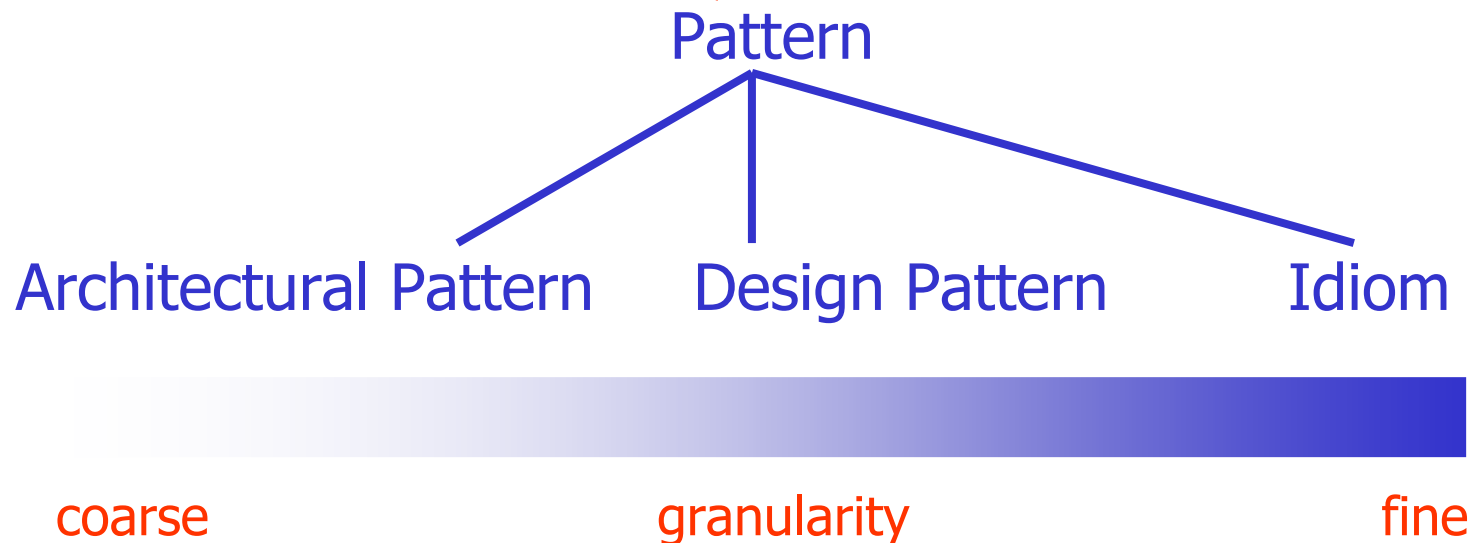
A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate.

[POSA96]

What are Patterns?

adapted from [POSA96]

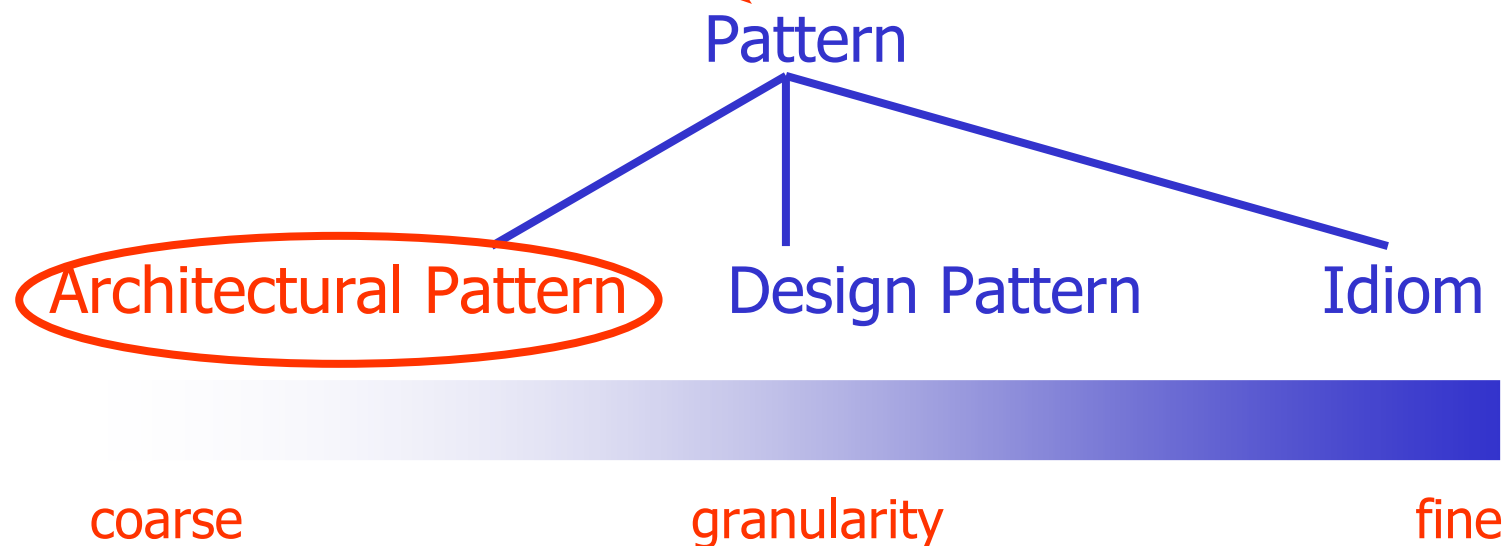
- Describes *one* proven solution for a recurrent design problem
- Defines the context for the solution's applicability



What are Patterns?

adapted from [POSA96]

- Describes *one* proven solution for a recurrent design problem
- Defines the context for the solution's applicability

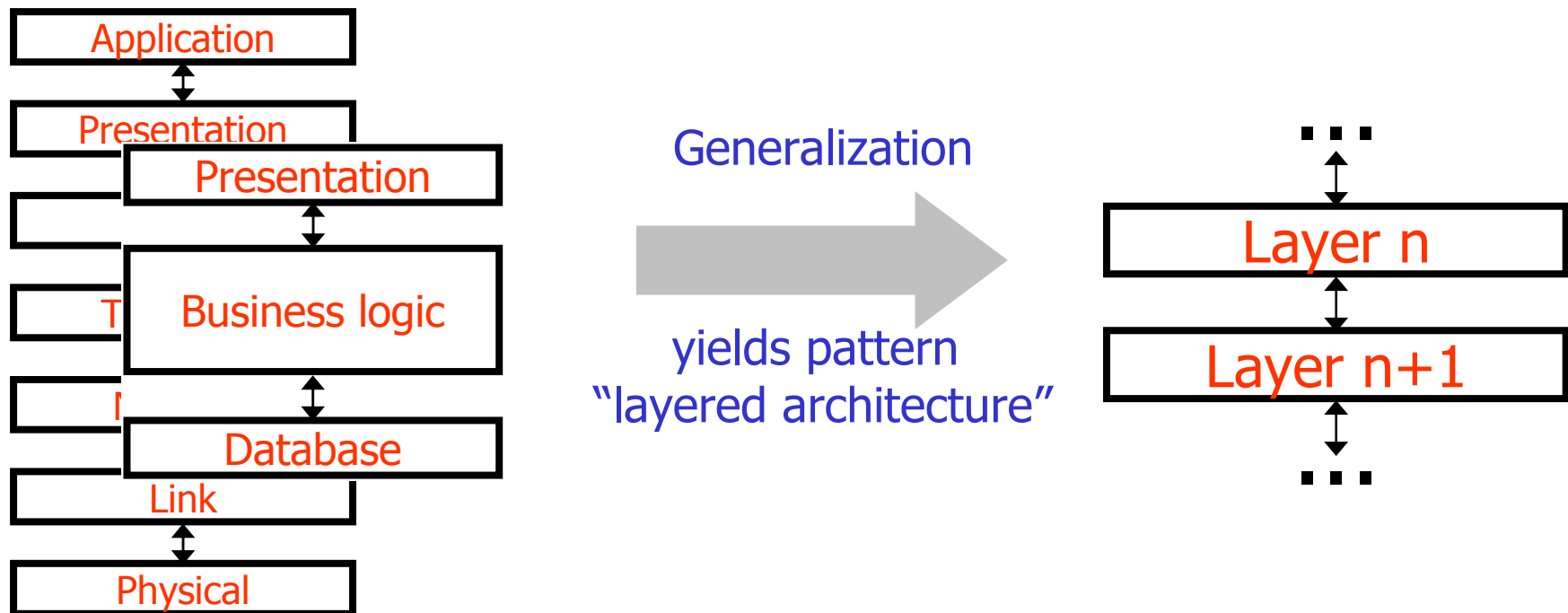


Architectural Patterns

"An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them."

[POSA96]

Architectural Patterns



Architectural Patterns – Schema

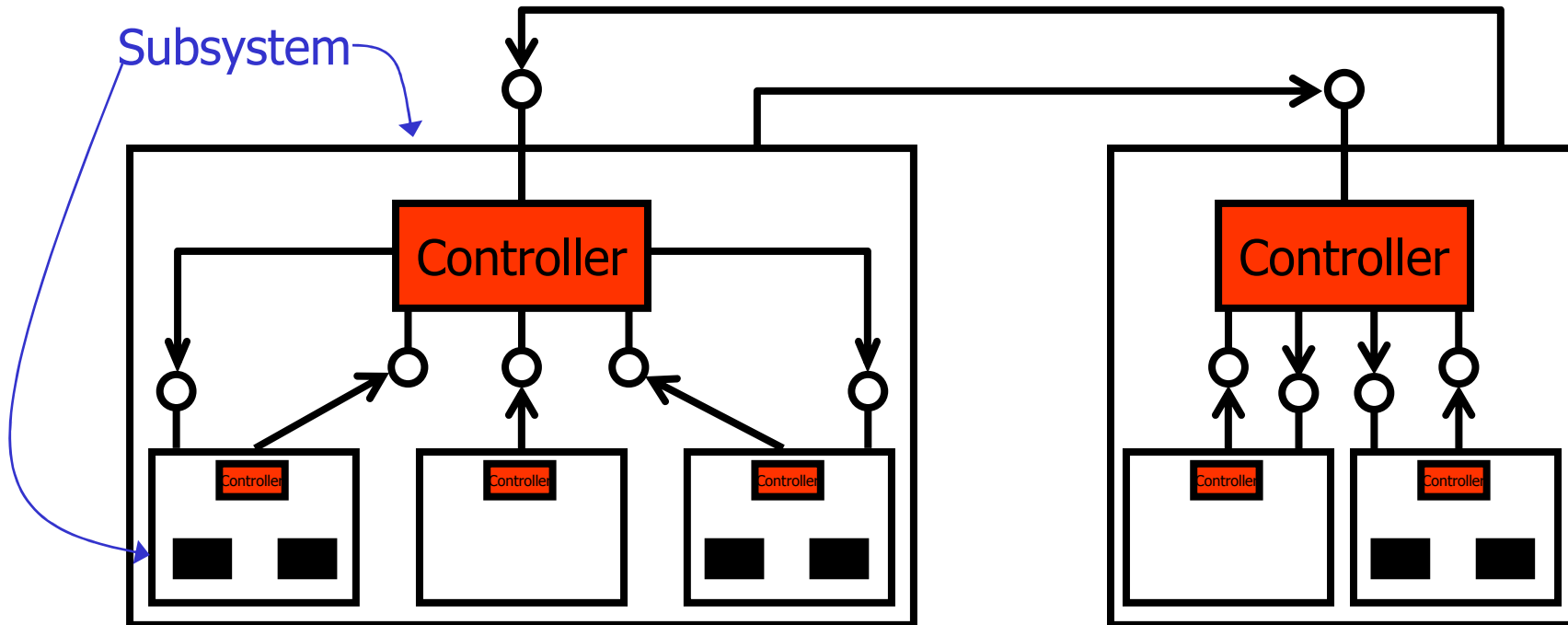
- Name:
 - Concise identifier for the pattern
- Context:
 - Under what circumstances is the pattern applicable?
- Problem:
 - What problem does the pattern solve?
 - What are the **tradeoffs**?
- Solution:
 - Structure
 - Behavior
 - Implementation
- Application examples, variants, consequences

Example: Subsystem Controller

- Context & Problem:
 - Complex system with large number of communication links between individual components
 - Tight coupling of components
 - Coherent subsystems can be identified
- Solution:
 - Decompose hierarchically into subsystems
 - Introduce controller components to decouple subsystems
 - Every subsystem consists of one controller and a set of further subsystems
 - Controllers manage communication
 - of subsystems on the same level of abstraction
 - of subsystems on lower levels of abstraction
 - Subsystems interact exclusively via “their” controller

Example: Subsystem Controller

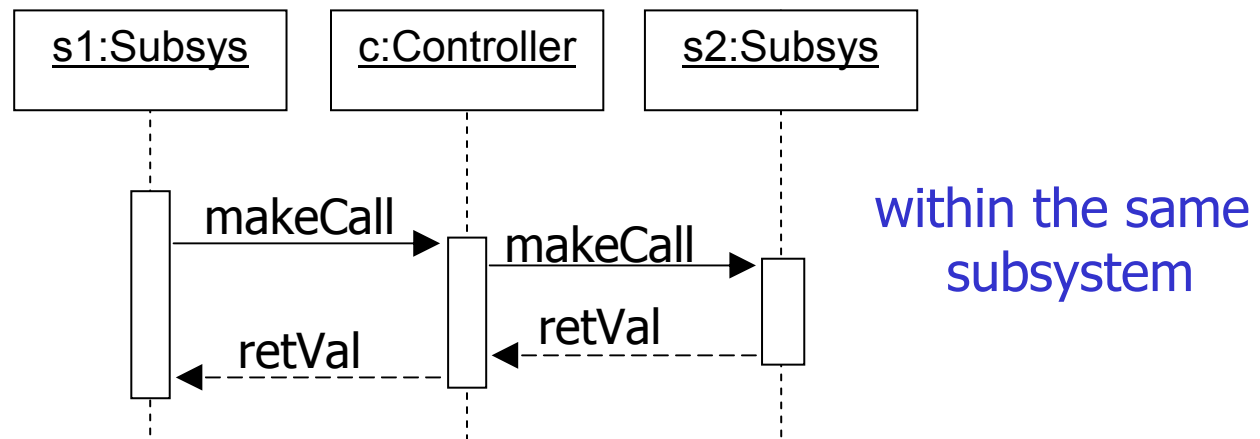
- Structure:



Communication proceeds exclusively via
the controllers' interfaces

Example: Subsystem Controller

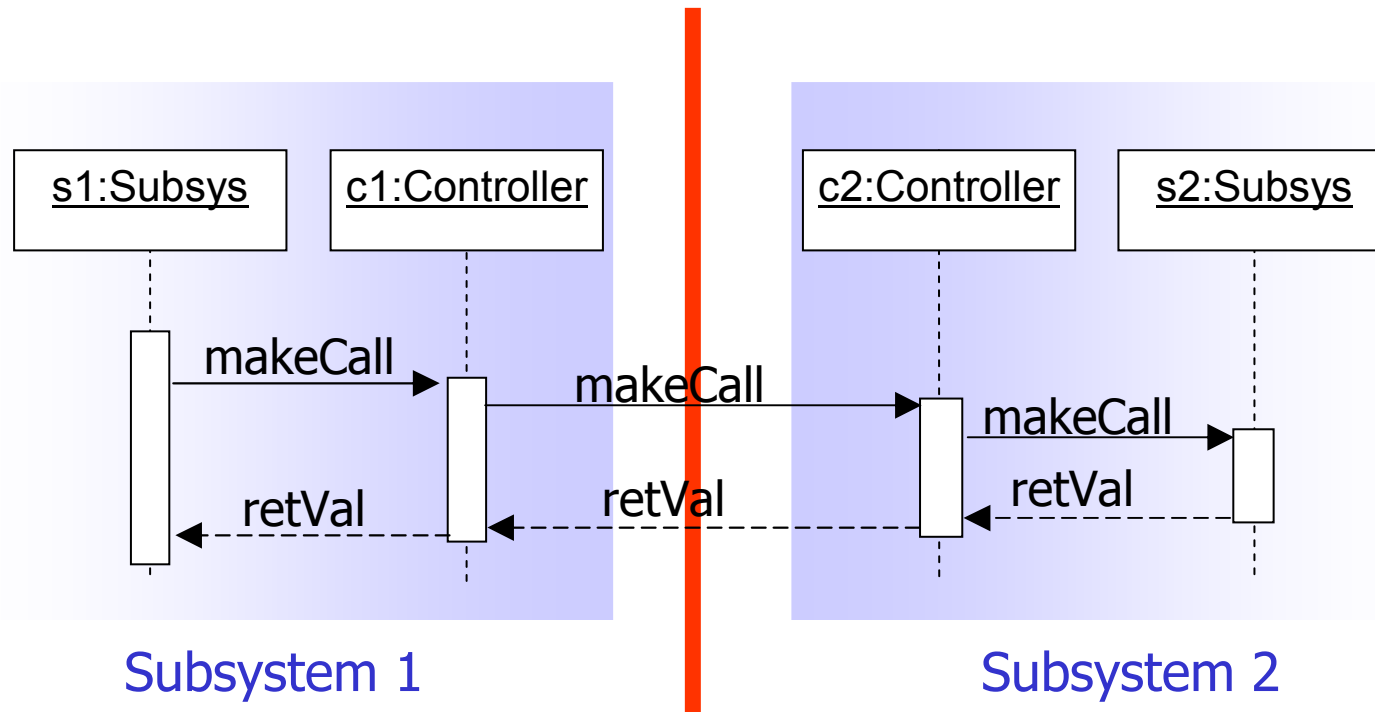
- Behavior:



Use efficient communication mechanisms!

Example: Subsystem Controller

- Behavior:



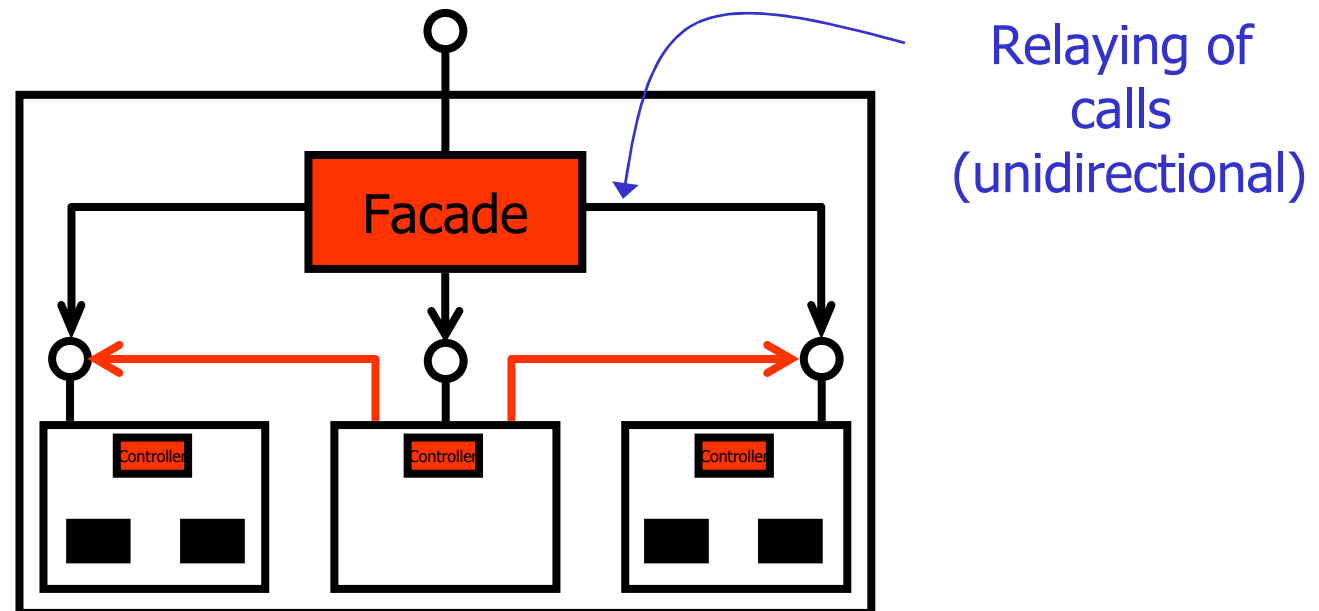
Communication across subsystem boundaries,
may have to use less efficient communication mechanisms

Example: Subsystem Controller

- Example Applications:
 - Telecommunication switches
- Consequences:
 - Clear communication structure
 - High degree of decoupling
 - Possibly less efficient
 - due to communication across multiple layers of hierarchy
 - in case of frequent communication across subsystem boundaries on the same layer of hierarchy
 - Starting point for performance analysis regarding communication paths
 - Starting point for separation between “hard” and “soft” real-time constraints:
 - ⇒ Within subsystems: “fast” communication
 - ⇒ between controllers: “slow” communication

Example: Subsystem Controller

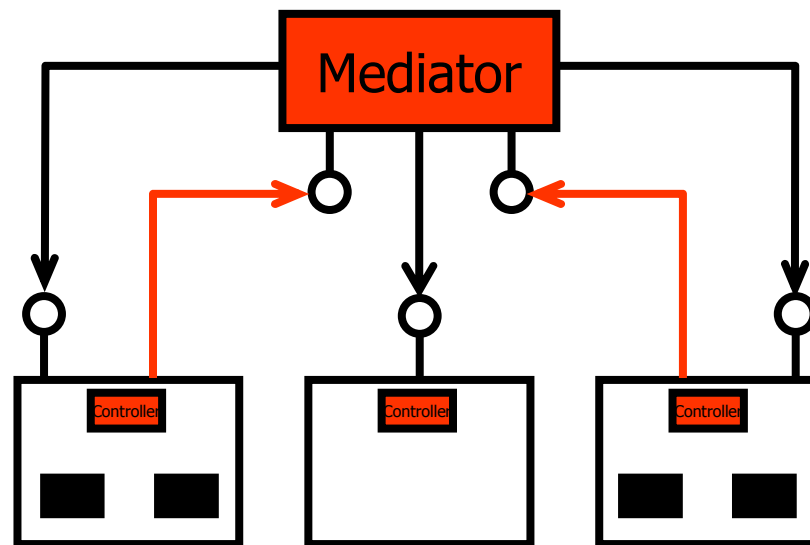
- Related Patterns:
 - Facade (see [GOF95])



The facade shields components within a subsystem from the environment. Components within the subsystem can interact directly (no detour via the facade).

Example: Subsystem Controller

- Related Patterns:
 - Mediator (see [GOF95])



The mediator coordinates the communication among a set of components. The components interact exclusively via the mediator

Overview

- The Notion of Architecture and Architectural Aspects
- The Role of Software Architecture
- Modeling and Documenting Architectures
- Patterns for Architecture and Design
- Refactoring Techniques

Refactoring Techniques

- Given:
 - A design model of the system under consideration, or
 - An implementation of the system
- Problem:
 - The system requirements change (frequently)
 - Mandatory/desirable system qualities are missing:
 - Clear structure
 - Reusability
 - Clarity
 - Changeability
 - ...
 - The adaptation to changing requirements is difficult
 - The investment into the existing design should pay off

Refactoring Techniques

- Can we modify the existing design/implementation, such that
 - The observable system behavior doesn't change, but
 - Afterwards the mandatory/desirable requirements are fulfilled?
- Example:
 - Let an elevator controller for a single cabin be given
 - The dispatching strategy is tightly coupled with the controller
 - This makes it difficult to extend the system such that it supports multiple cabins and dispatching strategies
 - What modifications are necessary to decrease the coupling between the dispatching strategy and the controller?
 - How to ensure that no errors are introduced in the process?

Refactoring Techniques

"Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure. It is a disciplined way to clean up code that minimizes the chances of introducing bugs. In essence when you refactor you are improving the design of the code after it has been written."

[F99]

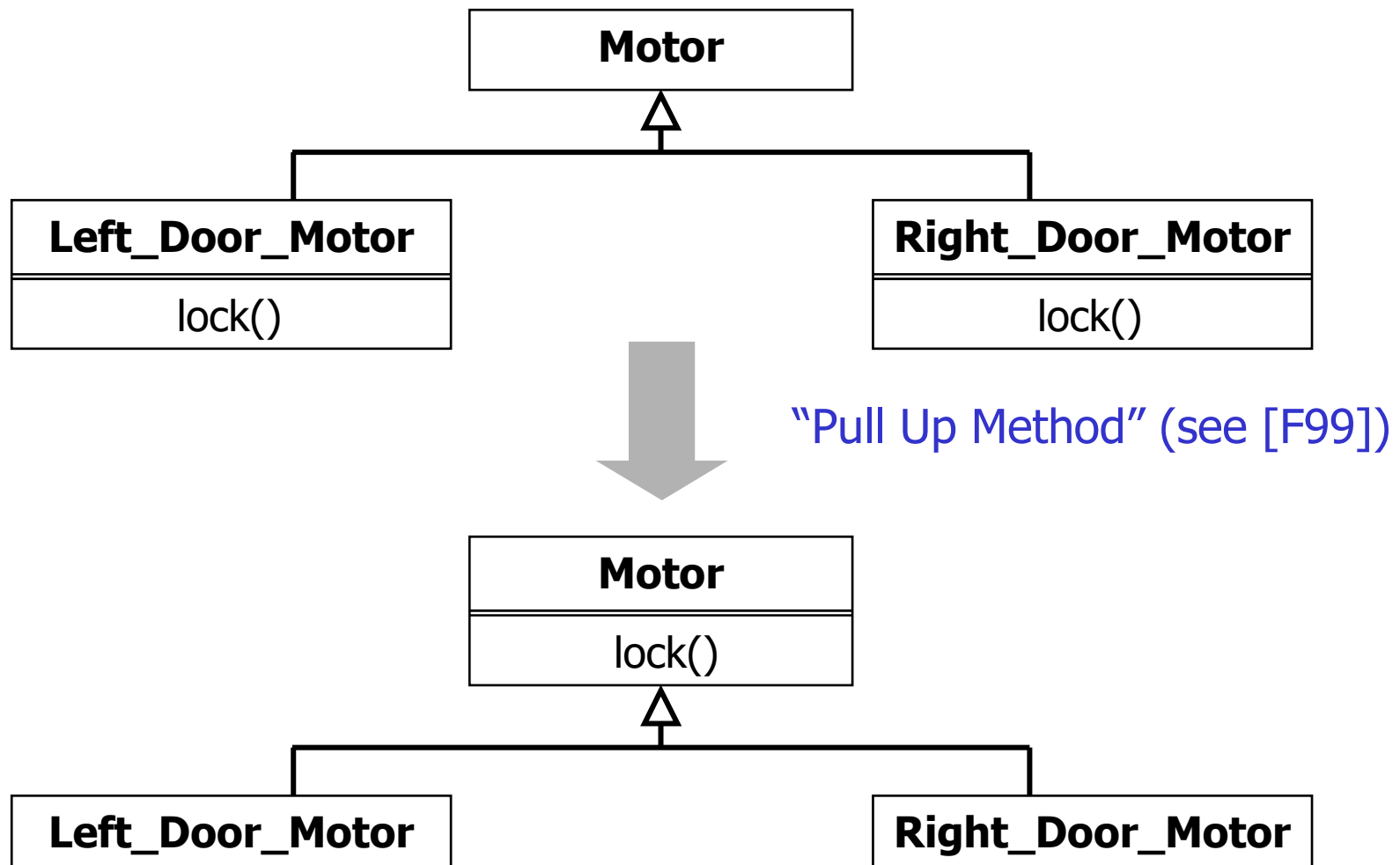
Refactoring Techniques

Refactoring:

- “minimally invasive” modifications to system structure
 - ⇒ Strategy of small steps
- Set up adequate test suite **before** changing the system
- Carry out tests during and after performing the change
 - ⇒ Increases confidence in correctness;
Goal: no change of observable behavior

Refactoring Techniques

Example: Move method to superclass



Refactoring Techniques

- Refactoring proceeds in extremely small steps
- Each individual change is manageable
- Refactoring fits particularly well with design patterns and architectural patterns:
 - 1.) Evaluate current state
 - 2.) Select target pattern
 - 3.) Identify sequence of refactoring steps leading to target pattern
 - 4.) Perform refactoring until target pattern is reached

Refactoring Techniques

- Must know catalog of refactoring techniques and architecture/design patterns for effective usage of refactoring (see [F99,GOF95,POSA96,E03]):
 - Simplification of
 - Method calls
 - Class hierarchies
 - Increase/Decrease coupling
 - Simplification of control structure
 - ...

Refactoring Techniques

- Skillful application of refactoring techniques allows picking the simplest solution to a given problem initially
 - ⇒ Refactoring helps adapt the solution to changing requirements
 - ⇒ Refactoring helps avoid “overengineering”
 - ⇒ Link to “eXtreme Programming”
- Conclusion:
 - Refactoring helps improve a design/an implementation in a series of small steps
 - High degree of experience/knowledge of refactoring techniques required

Conclusion

- The selection of a sustainable software architecture is one of the key success factors in the development of complex systems of high quality
- ADLs and patterns help expose important structural and behavioral aspects of a software architecture
- Refactoring techniques: “policy of small steps”, used for reaching a target architecture displaying the desirable/mandatory quality requirements

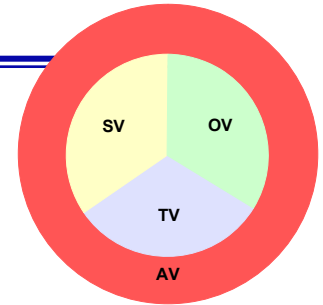
The End

Thank you for your attention!

Questions?

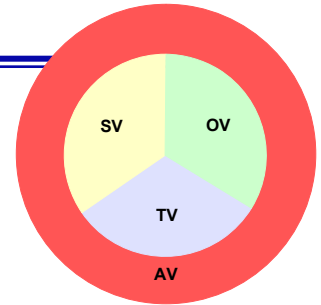
AV - All Views

- [Overview and Summary Information \(AV-1\)](#)
- [Integrated Dictionary \(AV-2\)](#)

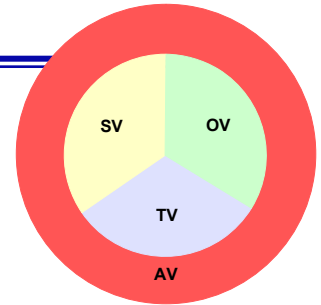


OV – Operational Views

- [High Level Operational Concept Graphic \(OV-1\)](#)
- [Operational Node Connectivity Description \(OV-2\)](#)
- [Operational Information Exchange Matrix \(OV-3\)](#)
- [Organizational Relationships Chart \(OV-4\)](#)
- [Operational Activity Model \(OV-5\)](#)
- [Operational Rule Model \(OV-6a\)](#)
- [Operational State Transition Description \(OV-6b\)](#)
- [Logical Data Model \(OV-7\)](#)



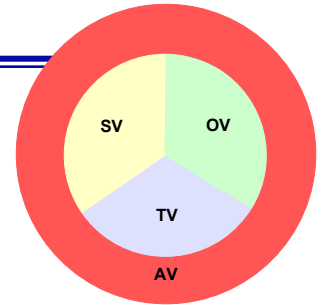
SV – System Views

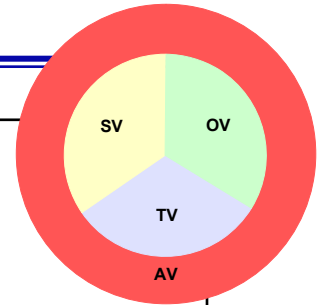


- [Systems Interface Description \(SV-1\)](#)
- [Systems Communication Description \(SV-2\)](#)
- [Systems-Systems Matrix \(SV-3\)](#)
- [Systems Functionalities Description \(SV-4\)](#)
- [Operational Activities to System Functionalities Traceability Matrix \(SV-5\)](#)
- [Systems Data Exchange Matrix \(SV-6\)](#)
- [Systems Performance Parameters Matrix \(SV-7\)](#)
- [Systems Evolution Description \(SV-8\)](#)
- [Systems Technology Forecasts \(SV-9\)](#)
- [Systems Rules Model \(SV-10a\)](#)
- [Systems State Transitions Description \(SV-10b\)](#)
- [Systems Event-Trace Description \(SV-10c\)](#)
- [Physical Schema \(SV-11\)](#)

TV – Technical Views

- [Technical Standards Profile \(TV-1\)](#)
- [Technical Standards Forecast \(TV-2\)](#)





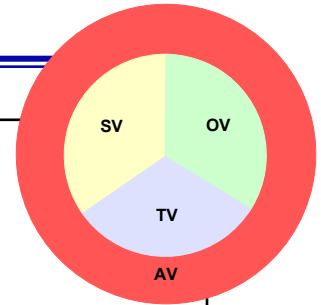
Overview and Summary Information (AV-1)

The Overview and Summary Information provides:

- Executive-level summary information in a consistent form that allows quick reference.

AV-1 includes:

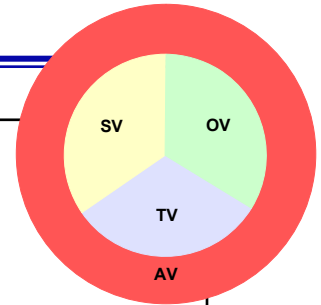
- Assumptions
- Constraints
- Limitations that may affect high-level decision processes involving the architecture.



Integrated Dictionary (AV-2)

The Integrated Dictionary contains:

- Definitions of terms used in the given architecture
- Textual definitions in the form of a glossary
- A repository of architecture data, their taxonomies, and their metadata (i.e., data about architecture data)
- Including metadata for tailored products, associated with the architecture products developed
- Metadata are the architecture data types, possibly expressed in the form of a physical schema. In this document, architecture data types are referred to as architecture data elements

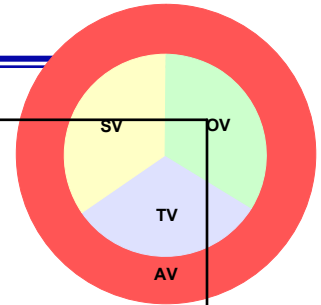


High Level Operational Concept Graphic (OV-1)

The High- Level Operational Concept Graphic describes:

- Mission
- Highlights main operational nodes (see OV-2 definition) and
- Interesting or unique aspects of operations
- A description of the interactions between the subject architecture and its environment, and between the architecture and external systems
- A textual description accompanying the graphic is crucial
- Graphics alone are not sufficient for capturing the necessary architecture data.

Operational Views



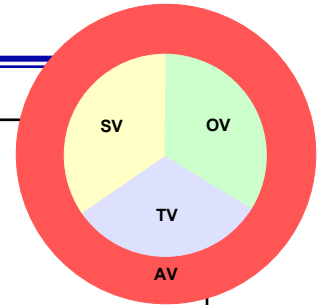
Operational Node Connectivity Description (OV-2)

The Operational Node Connectivity Description graphically depicts:

- The operational nodes (or organizations) with needlines between those nodes that indicate a need to exchange information

The graphic includes:

- Internal operational nodes (internal to the architecture) as well as external nodes

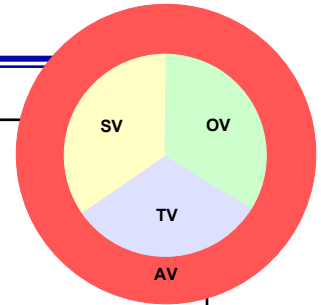


Operational Information Exchange Matrix (OV-3)

The Operational Information Exchange Matrix details:

- Information exchanges and identifies “who exchanges what information, with whom
- Why the information is necessary
- And how the information exchange must occur.” [CJCSI 6212.01B, 2000]

There is not a one-to-one mapping of OV-3 information exchanges to OV-2 needlines; rather, many individual information exchanges may be associated with one needline.

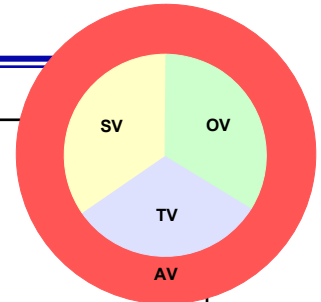


Organizational Relationships Chart (OV-4)

The Organizational Relationships Chart illustrates the command structure or relationships (as opposed to relationships with respect to a business process flow) among:

- Human roles
- Organizations, or organization types that are the key players in an architecture.

Operational Views



Operational Activity Model (OV-5)

Read:
“services”

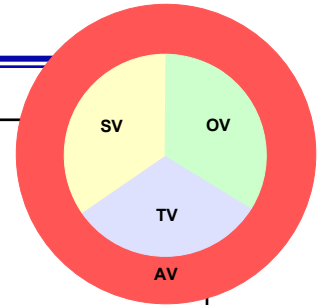
The Operational Activity Model describes:

- The operations that are normally conducted in the course of achieving a mission or a business goal.
- It describes **capabilities**, operational activities (or tasks), input and output (I/O) flows between activities, and I/O flows to/from activities that are outside the scope of the architecture.

High- level operational activities should trace to (are decompositions of):

- A Business Area
- An Internal Line of Business, and/or a Business Sub-Function as published in OMB’s Business Reference Model [OMB, 2003].

Operational Views

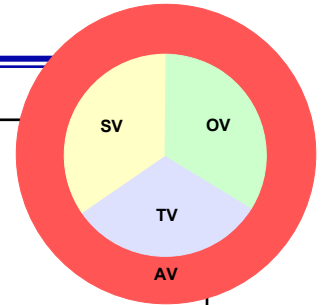


Operational Rule Model (OV-6a)

The Operational Rules Model specifies operational or business rules constraining:

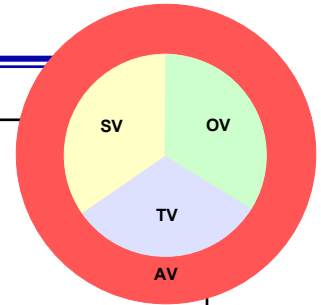
- An enterprise
- A mission
- Operation
- Business
- Or an architecture

Operational Views



Operational State Transition Description (OV-6b)

- The Operational State Transition Description is a graphical method of describing how an operational node or activity responds to various events by changing its state.
- The diagram represents the sets of events to which the architecture will respond (by taking an action to move to a new state) as a function of its current state.
- Each transition specifies an event and an action.



Logical Data Model (OV-7)

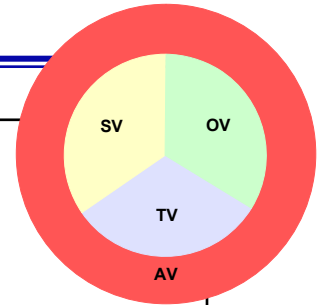
The Logical Data Model describes:

- The structure of an architecture domain's system data types and
- The structural business process rules (defined in the architecture's Operational View) that govern the system data.

It provides a definition of:

- Architecture domain data types
- Their attributes or characteristics
- And their interrelationships.

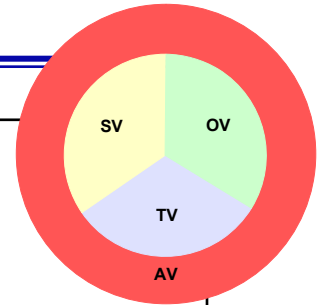
System Views



Systems Interface Description (SV-1)

- The Systems Interface Description depicts systems nodes and the systems resident at these nodes to support organizations/human roles represented by operational nodes of the Operational Node Connectivity Description (OV-2).
- SV-1 also identifies the interfaces between systems and systems nodes.

System Views

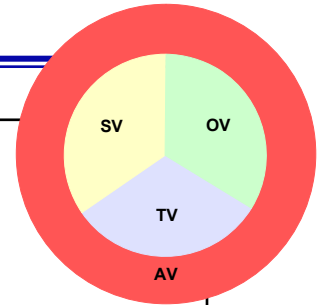


Systems Communication Description (SV-2)

The Systems Communications Description depicts pertinent information about:

- Communications systems
- Communications links
- Communications networks.
- SV-2 documents the kinds of communications media that support the systems and implement their interfaces as described in SV-1.
- Thus, SV-2 shows the communications details of SV-1 interfaces that automate aspects of the needlines represented in OV-2.

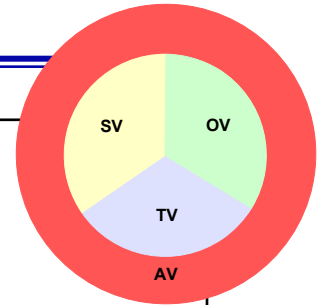
System Views



Systems-Systems Matrix (SV-3)

The Systems-Systems Matrix provides detail on:

- The interface characteristics described in SV-1 for the architecture
- Arranged in matrix form.



Systems Functionalities Description (SV-4)

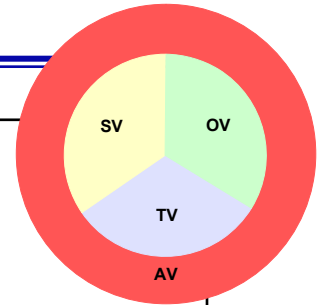
The Systems Functionality Description documents:

- System functional hierarchies
- System functions
- The system data flows between them.

NOTE:

Although there is a correlation between Operational Activity Model (OV-5) or business-process hierarchies and the system functional hierarchy of SV-4, it need not be a one-to-one mapping, hence, the need for the Operational Activity to Systems Function Traceability Matrix (SV-5), which provides that mapping.

System Views

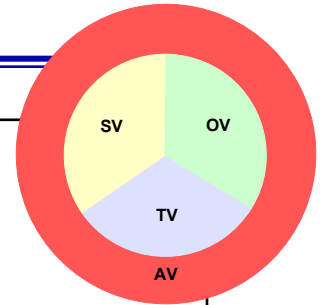


Operational Activities to System Functionalities Traceability Matrix (SV-5)

Operational Activity to Systems Function Traceability Matrix is a specification of:

- The relationships between the set of operational activities applicable to an architecture
- The set of system functions applicable to that architecture.

System Views

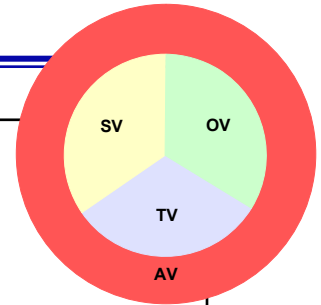


Systems Data Exchange Matrix (SV-6)

The Systems Data Exchange Matrix specifies the characteristics of the system data exchanged between systems.

This product focuses on automated information exchanges (from OV-3) that are implemented in systems.

Non-automated information exchanges, such as verbal orders, are captured in the OV products only.

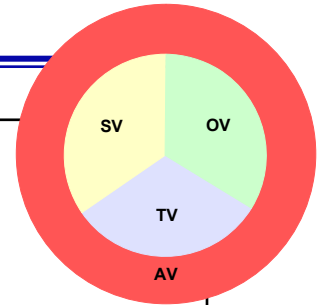


Systems Performance Parameters Matrix (SV-7)

The Systems Performance Parameters Matrix product specifies:

- The quantitative characteristics of systems and system hardware/software items
- Their interfaces (system data carried by the interface as well as communications link details that implement the interface)
- System functions.
- The current performance parameters of each system, interface, or system function,
- And the expected or required performance parameters at specified times in the future.

System Views

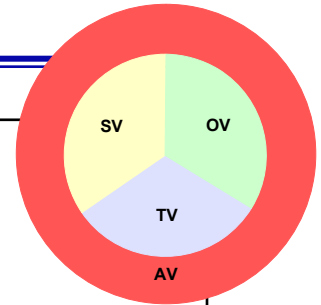


Systems Evolution Description (SV-8)

The Systems Evolution Description captures evolution plans that describe how the system or the architecture in which the system is embedded, will evolve over a lengthy period of time.

Generally, the timeline milestones are critical for a successful understanding of the evolution timeline.

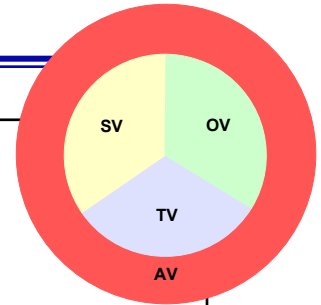
System Views



Systems Technology Forecasts (SV-9)

- The Systems Technology Forecast defines the underlying current and expected supporting technologies.
- It is not expected to include predictions of technologies as with a crystal ball.
- Expected supporting technologies are those that can be reasonably forecast given the current state of technology and expected improvements.
- New technologies should be tied to specific time periods, which can correlate against the time periods used in SV-8 milestones.

System Views



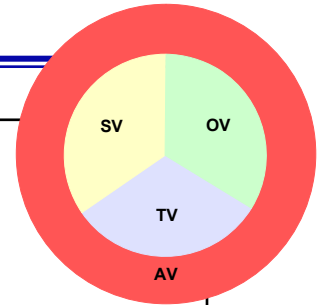
Systems Rules Model (SV-10a)

Systems rules are constraints on:

- An architecture
- On a system(s), or system hardware/software item(s)
- And/or on a system function(s).

While other SV products (e.g., SV-1, SV-2, SV-4, SV-11) describe the static structure of the Systems View (i.e., what the systems can do), they do not describe, for the most part, what the systems *must* do, or what it *cannot* do.

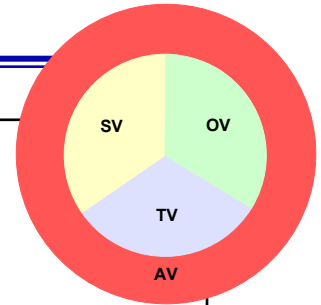
System Views



Systems State Transitions Description (SV-10b)

- The Systems State Transition Description is a graphical method of describing a system (or system function) response to various events by changing its state.
- The diagram basically represents the sets of events to which the systems in the architecture will respond (by taking an action to move to a new state) as a function of its current state.
- Each transition specifies an event and an action.

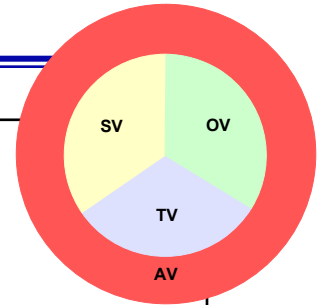
System Views



Systems Event-Trace Description (SV-10c)

- The Systems Event-Trace Description provides a time-ordered examination of the system data elements exchanged between participating systems (external and internal), system functions, or human roles as a result of a particular scenario.
- Each event-trace diagram should have an accompanying description that defines the particular scenario or situation.
- SV-10c in the Systems View may reflect system-specific aspects or refinements of critical sequences of events described in the Operational View.

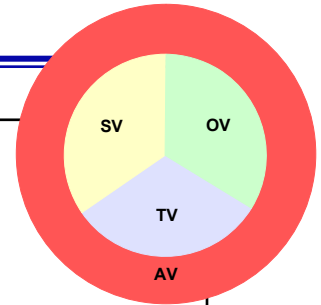
System Views



Physical Schema (SV-11)

- The Physical Schema product is one of the architecture products closest to actual system design in the Framework.
- The product defines the structure of the various kinds of system data that are utilized by the systems in the architecture.

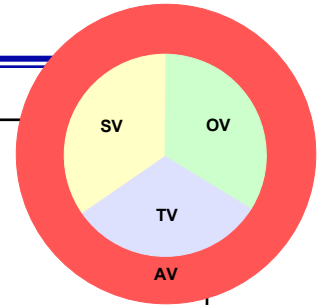
Technical Views



Technical Standards Profile (TV-1)

The Technical Standards Profile collects the various systems standards rules that implement and sometimes constrain the choices that can be made in the design and implementation of an architecture.

Technical Views



Technical Standards Forecast (TV-2)

- The Technical Standards Forecast contains expected changes in technology-related standards and conventions, which are documented in the TV-1 product.
- The forecast for evolutionary changes in the standards should be correlated against the time periods as mentioned in the SV-8 and SV-9 products.