Summary of Lecture 29.11.2017

http://englishskills.se

Horizontal Architecture Layer Principles:

- A1: Architecture Layer Isolation
- A2: Partitioning, Encapsulation and Coupling
- A3: Conceptual Integrity
- A4: Redundancy
- A5: Interoperability
- A6: Common Functions
- A7: Reference Architectures, Frameworks and Patterns
- A8: Reuse and Parametrization
- A9: Industry Standards
- A10: Information Architecture
- A11: Formal Modeling
- A12: Complexity and Simplification

29.11.2017

A5

Architecture Principle A5:

Interoperability

1. Precisely (formally) specify syntax and semantics in all interoperations
2. Whenever possible use formal contracts for the definition of interfaces
3. Whenever possible adopt and enforce accepted interoperability industry standards

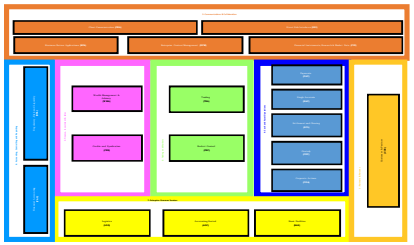
Justification: Successful, unambigous interoperability is a key factor in today‘s distributed systems. Interoperabilty failures have severe consequences and are difficult to pinpoint. Formal contracts isolate the parts of the system.

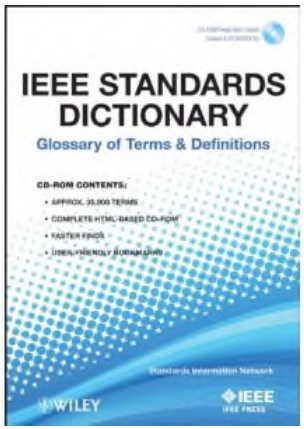
Future-Proof Software-Systems: Summary 29.11.2017

System Part **A**

Interoperability is the capability to exchange and make use of information and control

System Part **B**





Applications Interoperability	Collaborating applications must share a common **conceptual model**	**Instruments**: • Conceptual Models • Domain Models • Ontologies
Semantic Interoperability	Meaning of the data	**Instruments**: • Controlled Vocabularies •Taxonomies
Syntactic Interoperability	Structure of the data	**Instruments**: • Syntax specification languages
Technical Interoperability	Interaction infrastructure	**Instruments**: • Network standards • Internet standards

A6

Architecture Principle A6:

Common Functions

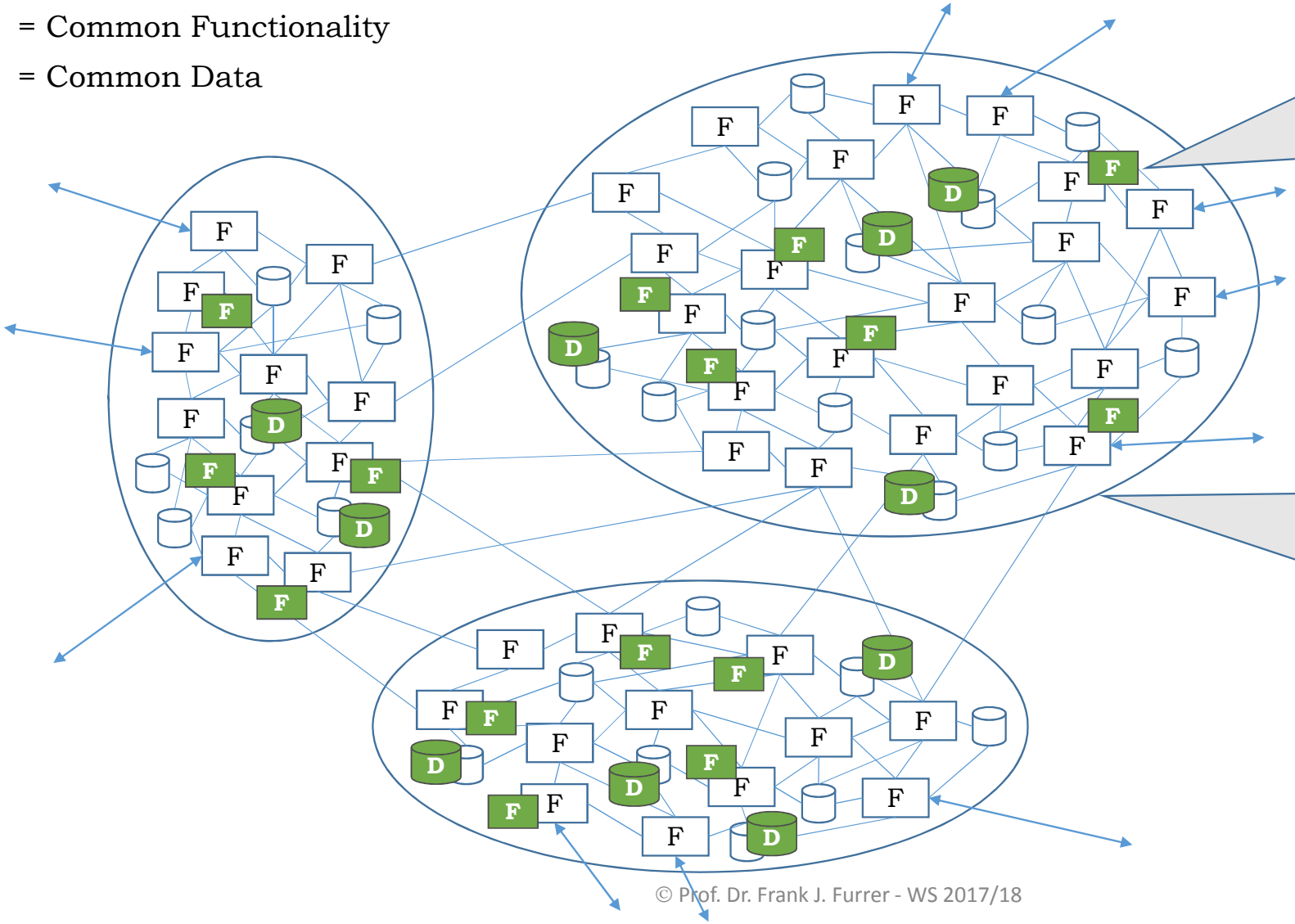
1. Identify all common functions and common data (= cross-cutting concerns in an IT-architecture)
2. Provide managed solutions to all cross-cutting concerns, avoiding unmanaged redundancy
3. Whenever possible provide and enforce a company-wide software-infrastructure

Justification: Cross-cutting concerns (Common functions and data) have a high inherent risk to diverge and thus cause unmanged reduncancy or inconsistent implementations – which can be an unknown and serious danger to an IT-system (especially a large or very large IT-system)

Future-Proof Software-Systems: Summary 29.11.2017

F = Common Functionality

D = Common Data



Some ***functionality*** or ***data*** is required in different encapsulated partitions

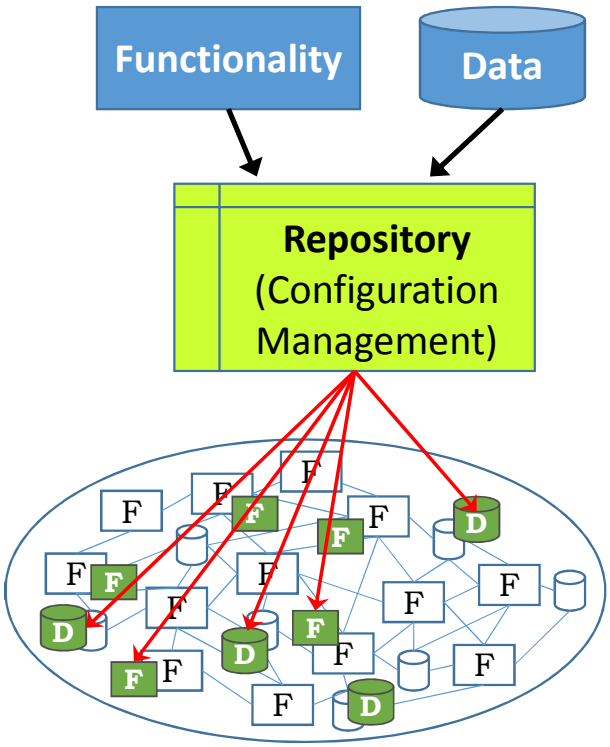
How can we avoid the **violation** of:

- Encapsulation
- Redundancy

?

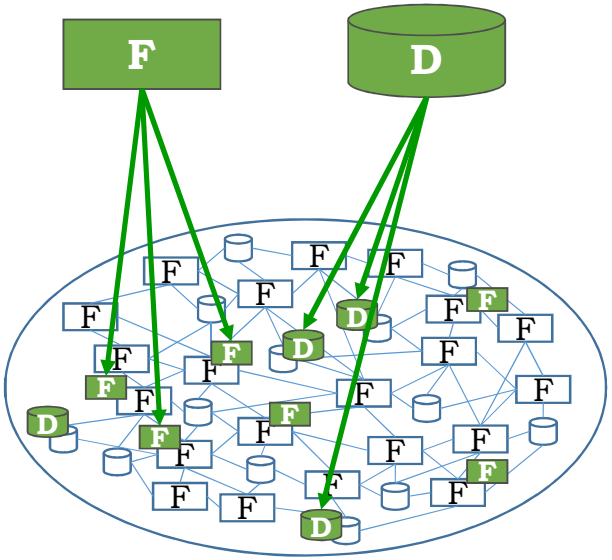
F = Common Functionality
D = Common Data

1 Single source!



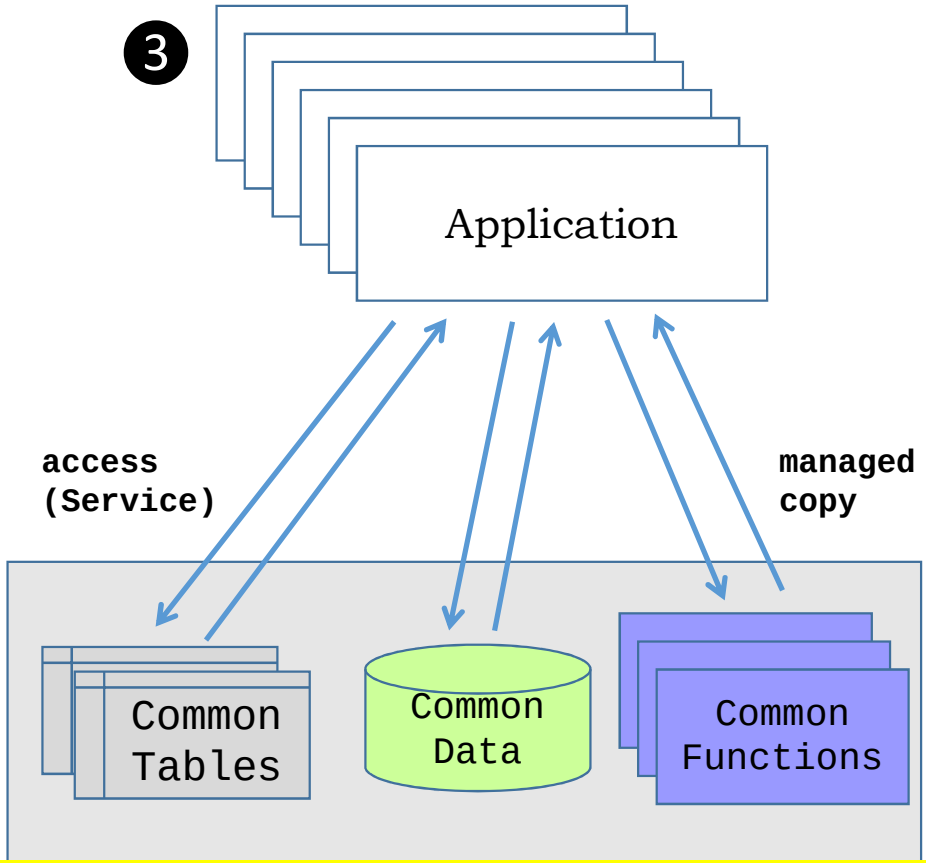
2

Common functionality and data are provided via ***services***



In different, encapsulated partitions

3



A5

Architecture Principle A5:

Reference Architectures, Frameworks and Patterns

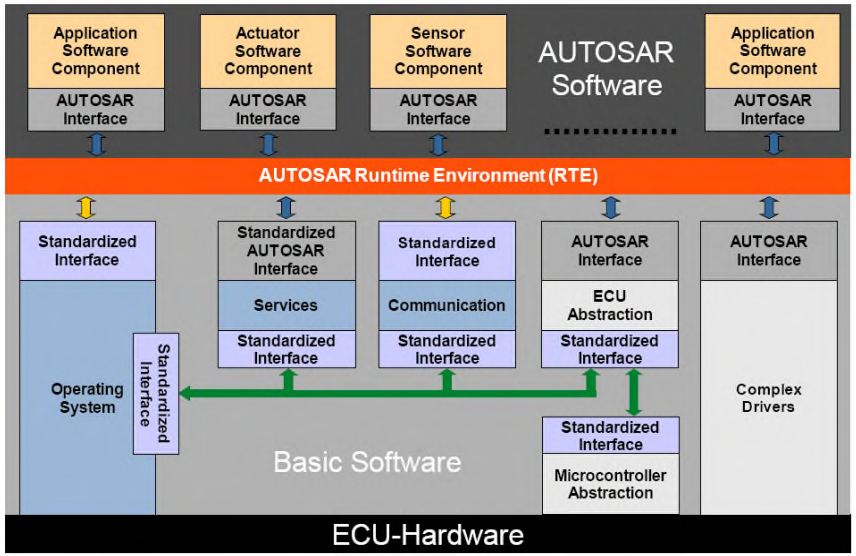
1. Always consider applicable reference architectures, frameworks and patterns. Whenever possible adhere to proven patterns as foundations of architecture and design
2. Manage and cultivate a set of patterns (in a repository) in your company for easy reference. Annotate these with your own experience
3. For 3rd party software insist (as much as possible) on industry- standardized interfaces, services and managed redundancy

Justification: Reference architectures, frameworks and patterns are highly valuable architecture and design knowledge in condensed form. Using suitable, proven reference architectures, frameworks and patterns leads to good, safe and often optimum solutions

Reference Architecture:

A reference architecture provides a template solution for an architecture for a particular application domain

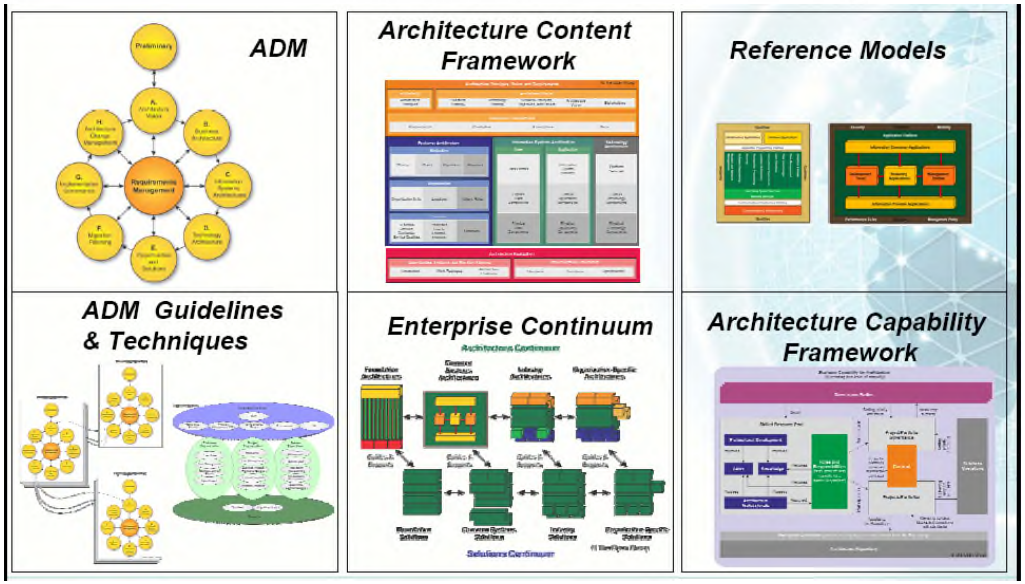
- *such as financial systems, automotive, aerospace etc.*



Architecture Framework:

An architecture framework establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular application domain

[ISO/IEC/IEEE 42010]



http://www.autosar.org/download/papersandpresentations/AUTOSAR_Brochure_EN.pdf

http://www.togaf.org

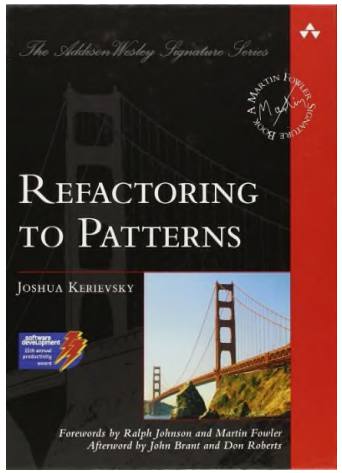
Architecture Pattern:

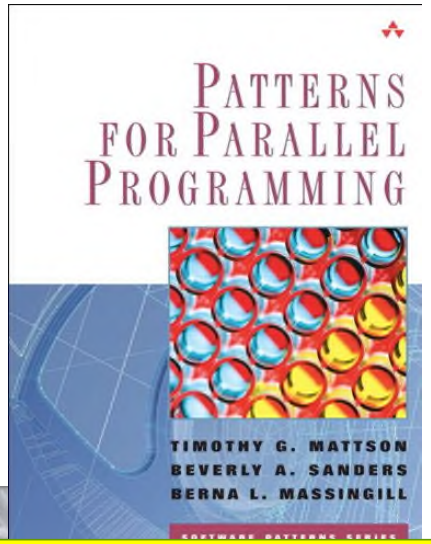
An architectural pattern is a concept that solves and delineates some essential cohesive elements of a software architecture

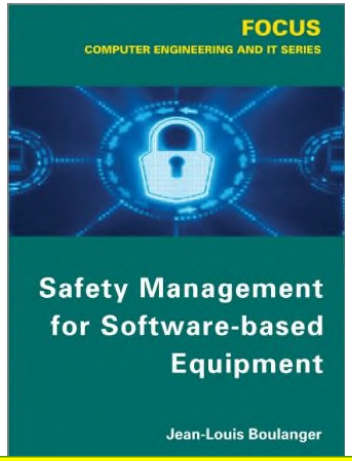
http://en.wikipedia.org/wiki/Architectural_pattern

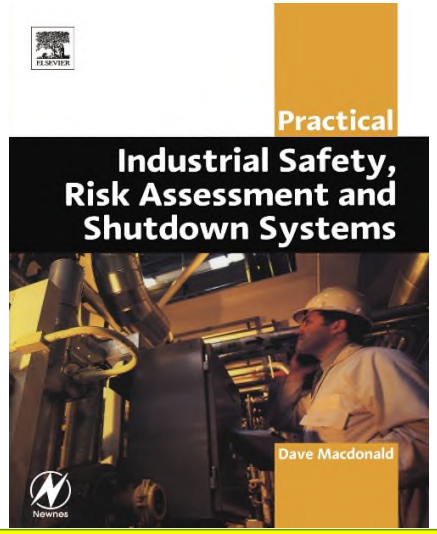
→ highly valuable software/system architecture **knowledge** in proven & easily accessible form

Future-Proof Software-Systems: Summary 29.11.2017

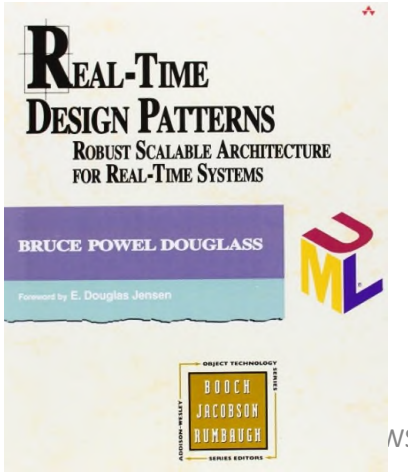


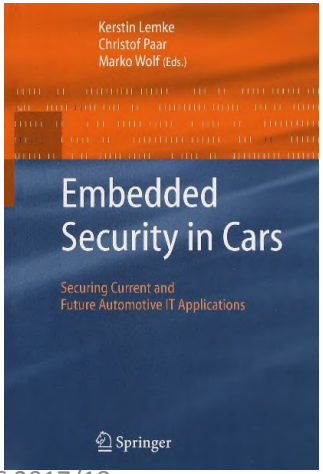


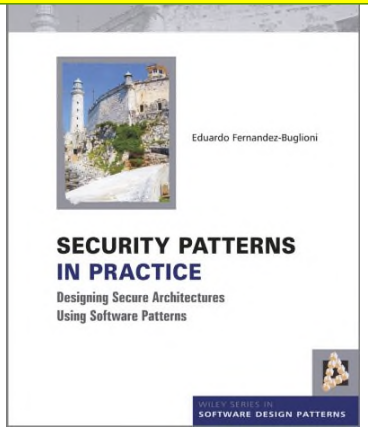


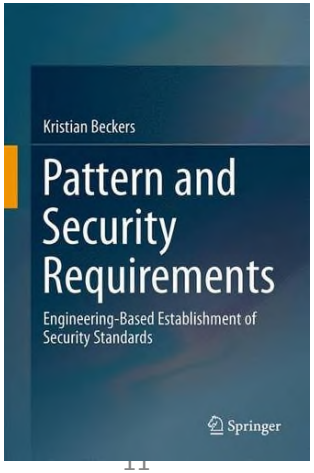


A future-proof software-systems engineer reads one pattern book per month!









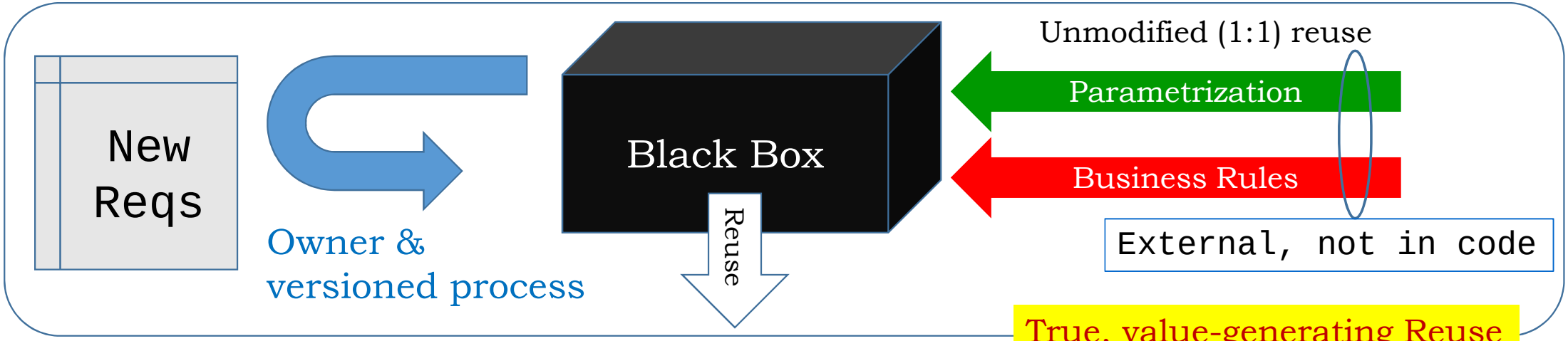
A8

Architecture Principle A8:

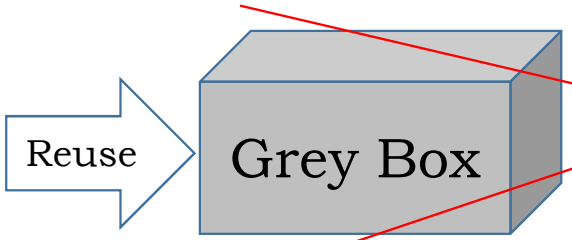
Reuse and Parametrization

1. Use only the black-box concept to build reusable software
2. Whenever possible, configure the reusable modules via parameters or business rules (loaded or initiated at run-time)
3. Install and consequently use a configuration management system to control the distribution of reusable software modules
4. Provide the 4 elements of successful reuse: Committed management, reuse-strategy, reuse-organization and competent software architects
5. Adapt your software development process to produce reusable software

Justification: If done *correctly*, reuseable components have a significant positive effect on the agility of the IT-system.

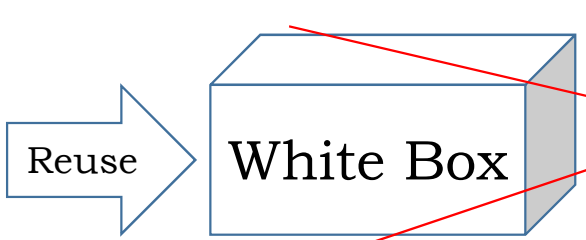


True, value-generating Reuse



Limited modified reuse
(Specific changes ≤ 25 %)

Not reuse ⇒ unmanaged redundancy

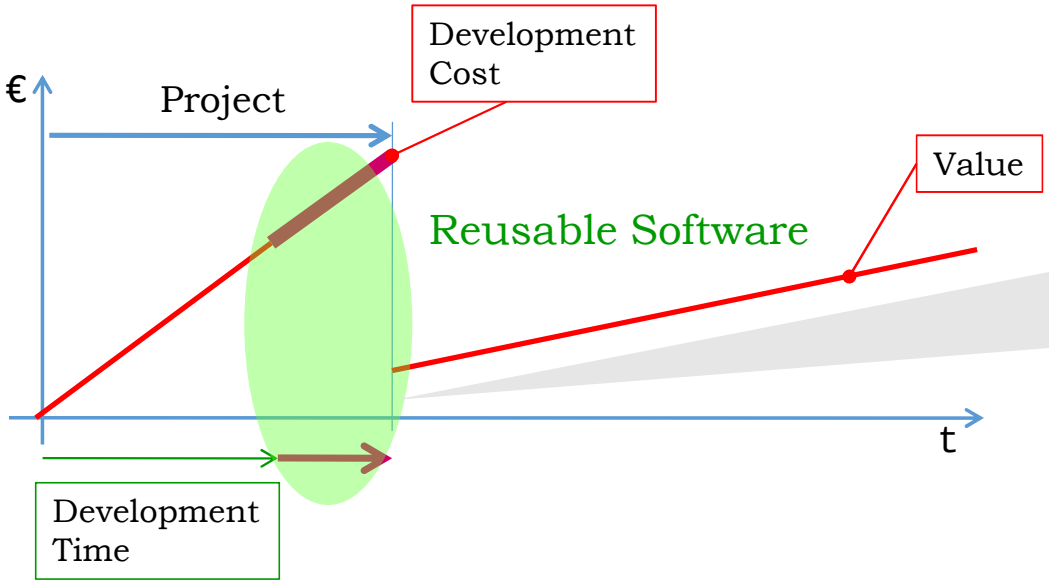


Significantly modified
(Specific changes ≥ 25 %)

Not reuse ⇒ unmanaged redundancy

http://sr.photos2.fotosearch.com

reuse



Reusable software requires **considerable more effort** in planning, design, development and implementation

Management

Architect

http://artofsoftwarereuse.com/tag/schemas/

Strategy

Organization

Reuse Strategy

A9

Architecture Principle A9:

Industry Standards

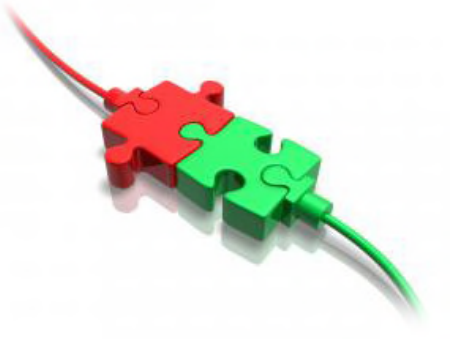
1. Strictly adhere to proven, accepted industry-standards in all 5 architecture layers and for all phases of the system lifecycle
2. Never allow any use of vendor-specific standards «extensions» (even if they look tempting and useful)
3. Keep the number of standards in use to a minimum
4. Introduce new standards only based on very good reasons
5. If for a certain field of your activity there is no industry standard, formulate and instantiate a company standard
6. Enforce strict adherence to (pure) standards via regular reviews

Justification: A heterogenous industry (such as software-production) requires *clearly stated foundations* for technologies, products and processes – otherwise no interoperability, certification, reuse and vendor-independence is possible

ISO International Organization for Standardization

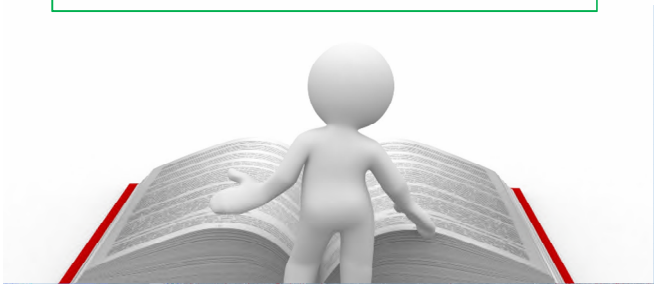
Industry-Standard

Technical Impact:
- Interoperability
- Communications
- Technology
- …



Certification:
- Safety
- Security
- Interfaces
- …

Knowledge:
- Processes
- Domain-Knowledge
- Cooperation
- …



Respected **standards** are powerful *interoperability* and *productivity* tools

A10 – Part 1

Architecture Principle A10 (a):

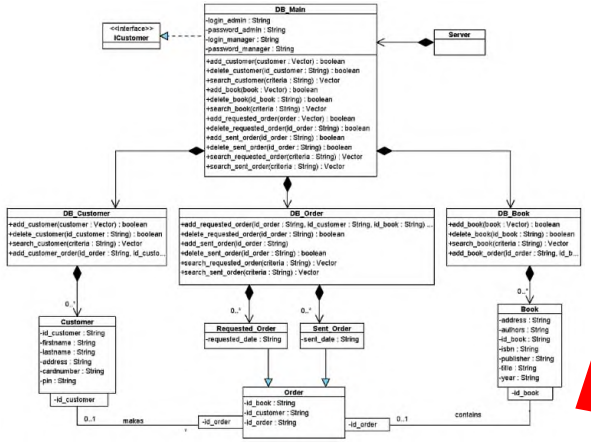
Enterprise Data/Information Architecture

1. Define and adhere to an enterprise wide data/information strategy (approved by CIO and CEO)
2. Model top-down with consistent, redundancy-free, complete models

   [⇒ Metadata & Semantics]
3. Assign roles and responsibilities for all data/information items
4. Define and strictly enforce data quality standards
5. Never allow unmanaged redundancy („single version of truth“)
6. Specify and enforce data naming and abbreviation standards
7. Define and implement suitable mechanisms for data validation (correctness, timeliness – possibly using acquisition redundancy)

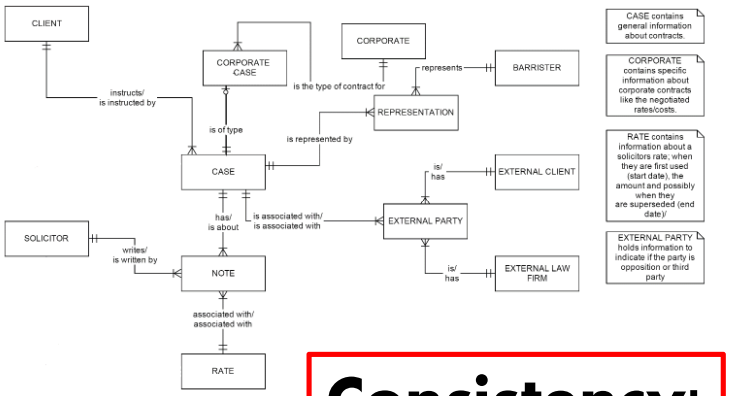
Justification: A good data/information architecture (and implementation!) is a highly valuable backbone for the enterprise. On the contrary, an unsuitable, inconsistent or badly implemented data/information architecture is a constant source of problems

UML Data/Information Model



http://indalog.ual.es/mdd/udbi/DB_ClassDiagram.png

Entity Relationsship Diagram (ERD)



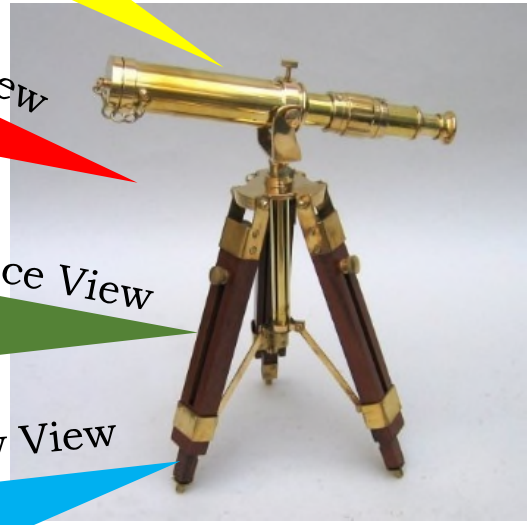
http://i.stack.imgur.com/GiAos.png

Consistency!

Multiple Views:

Show only **relevant** elements

- Structure View
- Security View
- Performance View
- Technology View

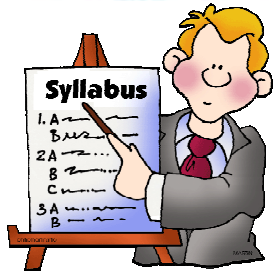


http://www.replicamuseum.com

Information Architecture

Structure

Metadata



Semantics

Strategy

A10 – Part 2

Architecture Principle A10 (b):

Embedded Data/Information Architecture

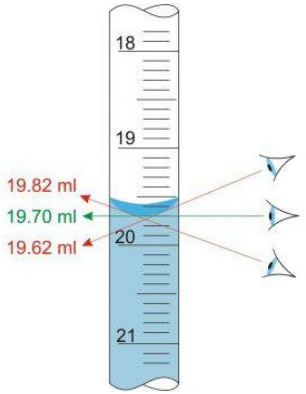
1. Define and adhere to a product data/information strategy
2. Model top-down with consistent, redundancy-free, complete models
   [⇒ Metadata & Semantics]
3. Never allow unmanaged redundancy („single version of truth“)
4. Stronly validate data/information after acquisition and before use (correctness, timeliness – possibly using acquisition redundancy)

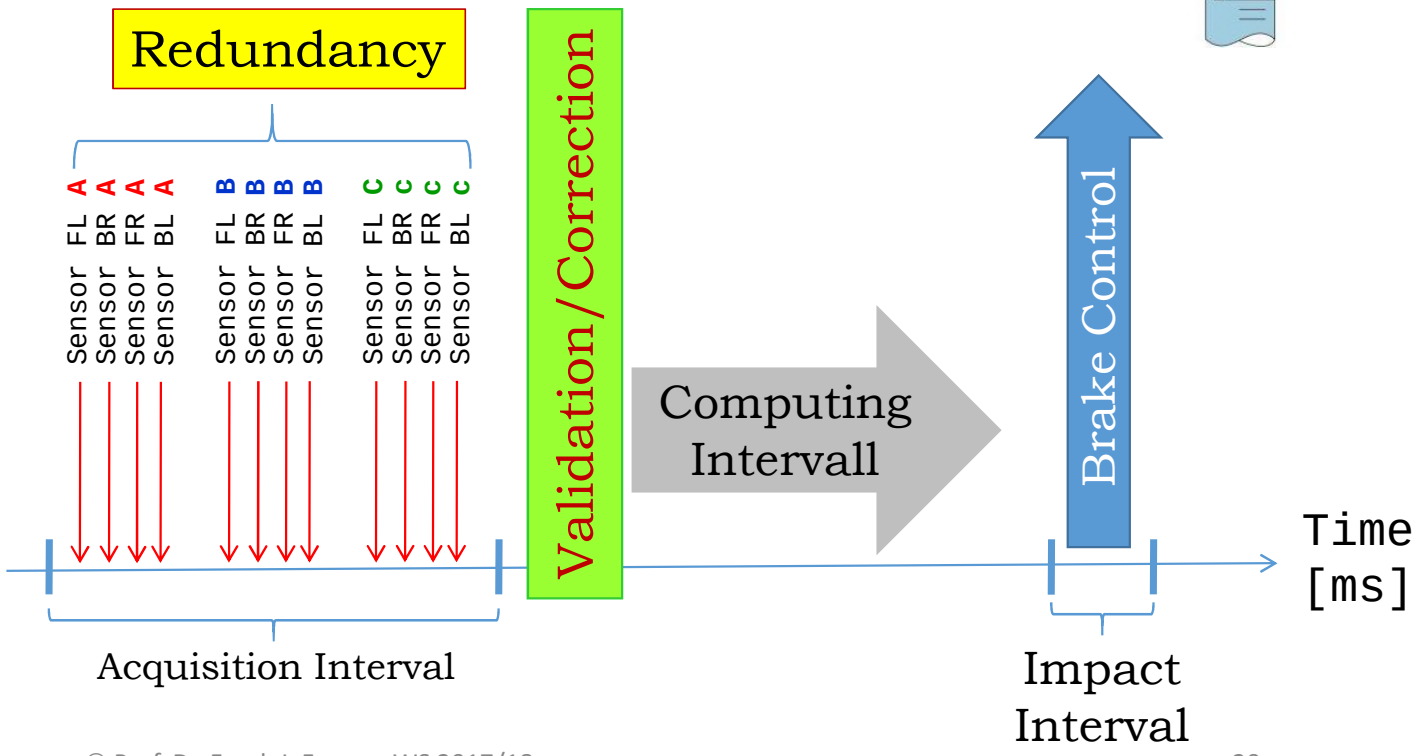
Justification: A good data/information architecture (and implementation!) is necessary for all products based on embedded software.

Enterprise
Information Architecture

- Enterprise Model
- Business Logic Model
- Domain Model
  Business Object Model
- Database/Table Models

Real-time (embedded)
Information Architecture





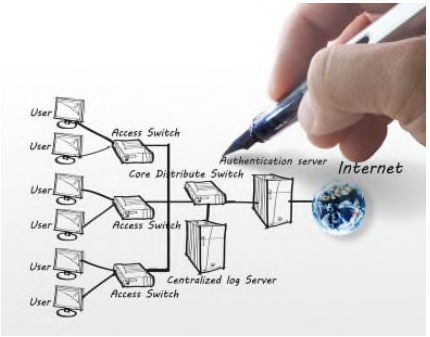
A11

… we continue now with A11: Formal Modeling (Slide 120)

Repeat

Model: ?

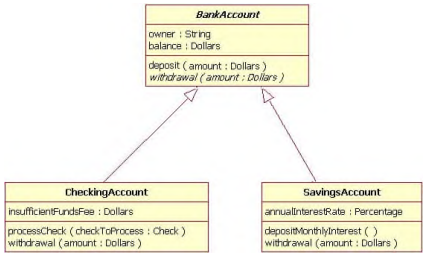
Informal Modeling ⇒



Syntax: Intuitive
Semantics: Intuitive

Informal discussions

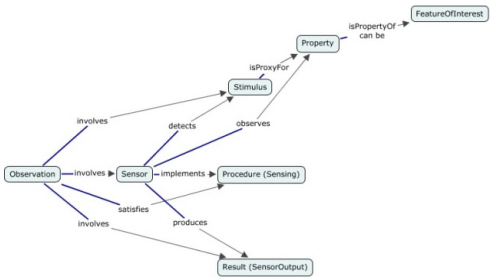
Semi-formal Modeling ⇒



Syntax: Formalized
Semantics: Semi-formal

Semi-formal discussions
Model-exchange, Profiles
Limited Model Checking

Formal Modeling ⇒



Syntax: Formalized
Semantics: Formalized

Formal discussions
Extensive Model Checking
Reasoning

Power of model

low

high

Repeat

C The **4** C's of models

C_1 Clarity

The concepts, relationships, and their attributes are unambigously *defined* and *understood* by all stakeholders

C_2 Committment

All stakeholders have *accepted* the model, its representation and the consequences (agreement)

C_3 Communication

The model truly and sufficiently represents the key properties of the real world to be mapped into the IT-solution

C_4 Control

The model is used for the assessment of specifications, design, implementation, reviews and evolution