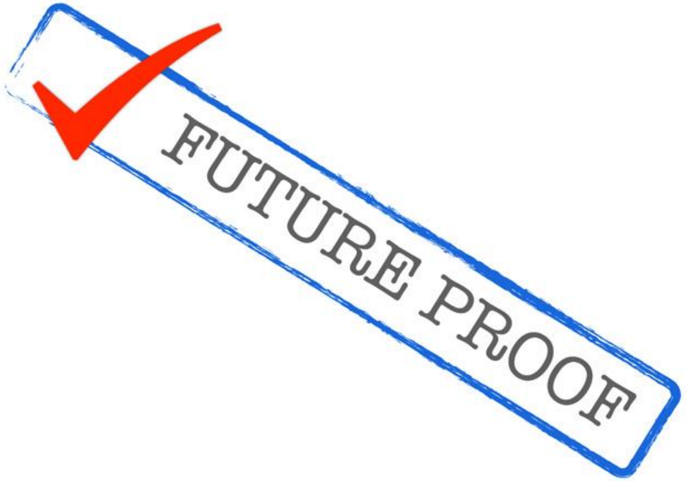


https://static1.squarespace.com

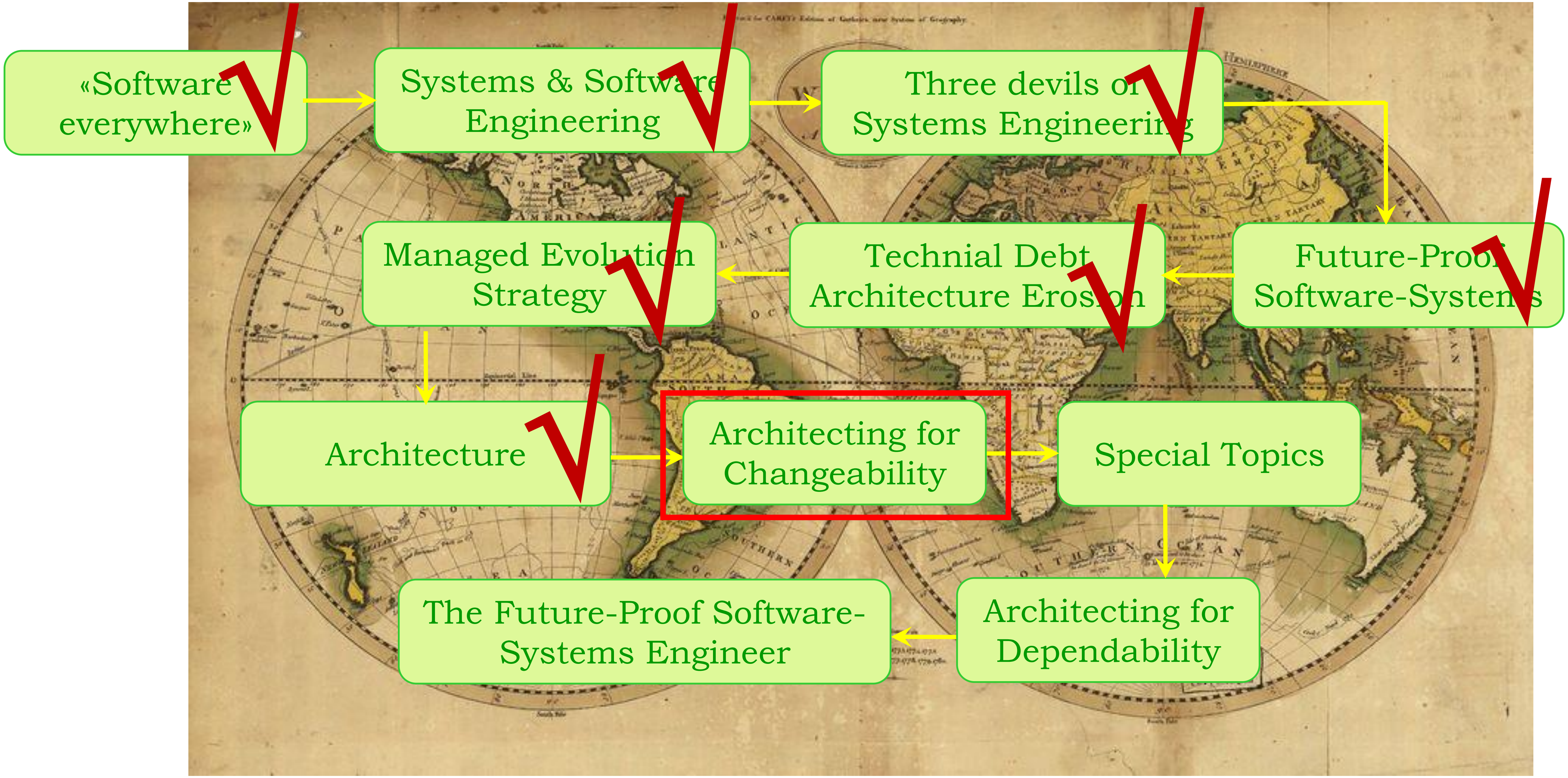
Future-Proof Software-Systems (FPSS)

Part 3B: Architecting for Changeability

Lecture WS 2019/20: Prof. Dr. Frank J. Furrer

Version 1.0

Our journey:



«Software everywhere» √

Systems & Software Engineering √

Three devils of Systems Engineering √

Future-Proof Software-Systems √

Technial Debt Architecture Erosion √

Managed Evolution Strategy √

Architecture √

Architecting for Changeability

Special Topics

Architecting for Dependability

The Future-Proof Software-Systems Engineer

© www.123rf.com – used with permission

Horizontal Architecture Layer Principles:

- A1: Architecture Layer Isolation
- A2: Partitioning, Encapsulation and Coupling
- A3: Conceptual Integrity
- A4: Redundancy
- A5: Interoperability
- A6: Common Functions

Part 3A ✓

- A7: Reference Architectures, Frameworks and Patterns
- A8: Reuse and Parametrization
- A9: Industry Standards
- A10: Information Architecture
- A11: Formal Modeling
- A12: Complexity and Simplification

Part 3B

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

A7

Architecture Principle A7:

Reference Architectures, Frameworks and Patterns

Formalized Architecture Knowledge: **Architecture Principles**

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

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]

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

Structure!

Patterns

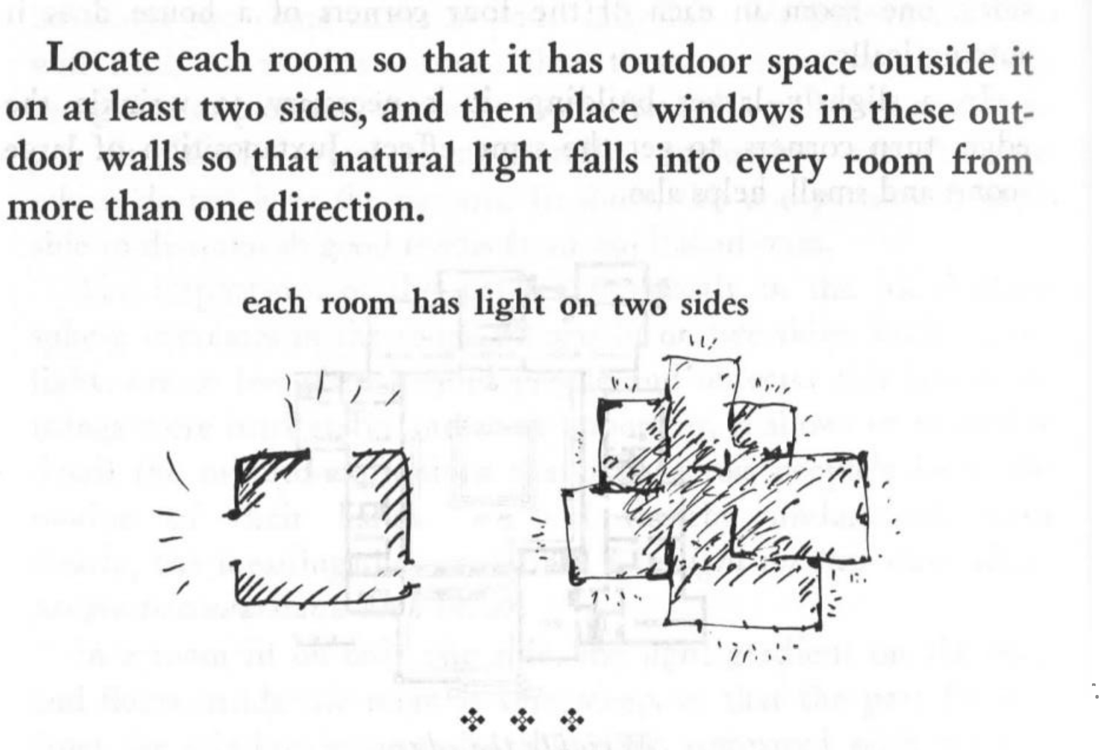
DEFINITIONS

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

Origin of Patterns:
*Christopher Alexander, **1977***



Locate each room so that it has outdoor space outside it on at least two sides, and then place windows in these outdoor walls so that natural light falls into every room from more than one direction.

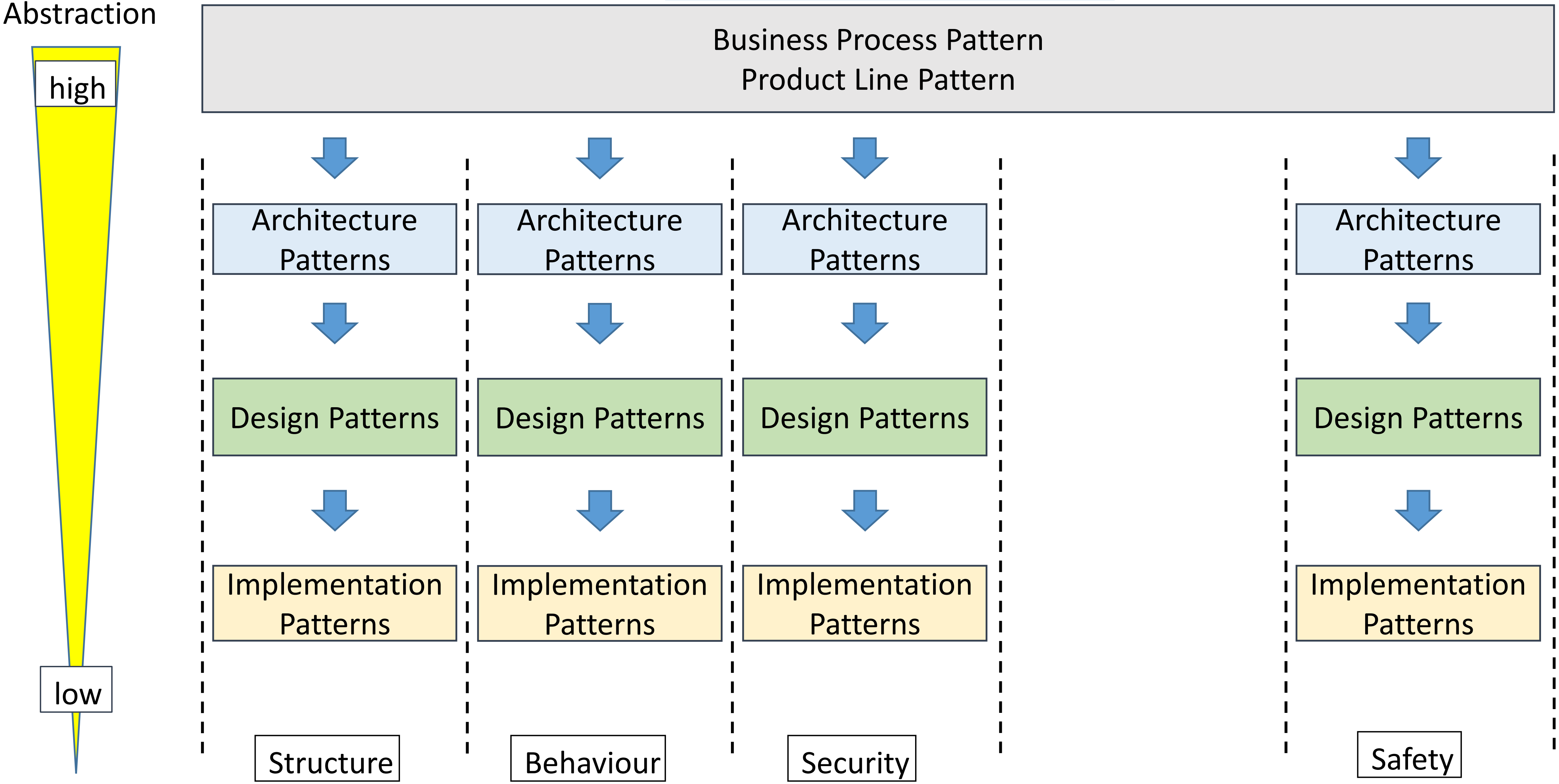
each room has light on two sides

❖ ❖ ❖

Application to Software Architecture:
*Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, **1995** („Gang of Four“)*

http://mihpatte.com/design-patterns-by-gamma

Pattern Hierarchy



Abstraction

high

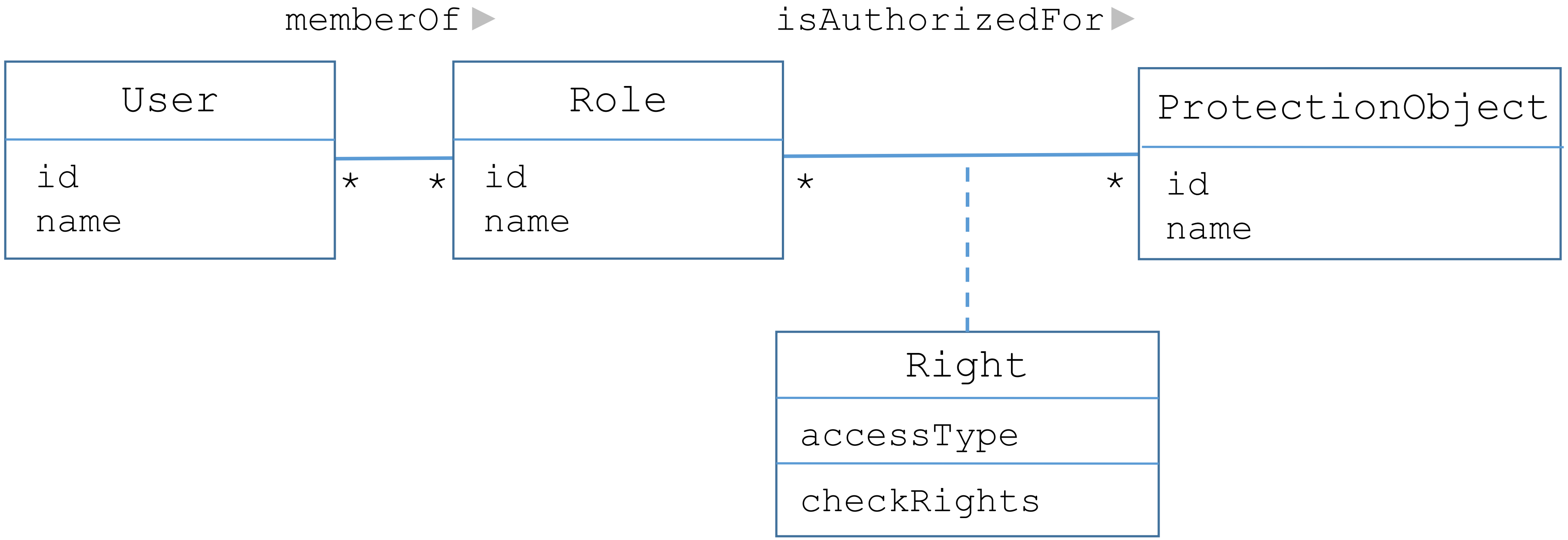
low

Business Process Pattern
Product Line Pattern

Architecture Patterns	Architecture Patterns	Architecture Patterns	Architecture Patterns
Design Patterns	Design Patterns	Design Patterns	Design Patterns
Implementation Patterns	Implementation Patterns	Implementation Patterns	Implementation Patterns
Structure	Behaviour	Security	Safety

Example: Security Pattern „RBAC“ [Role-Based Access Control]

(Fernandez: Security Patterns in Practice, 2013, ISBN 978-1-119-99894-5)

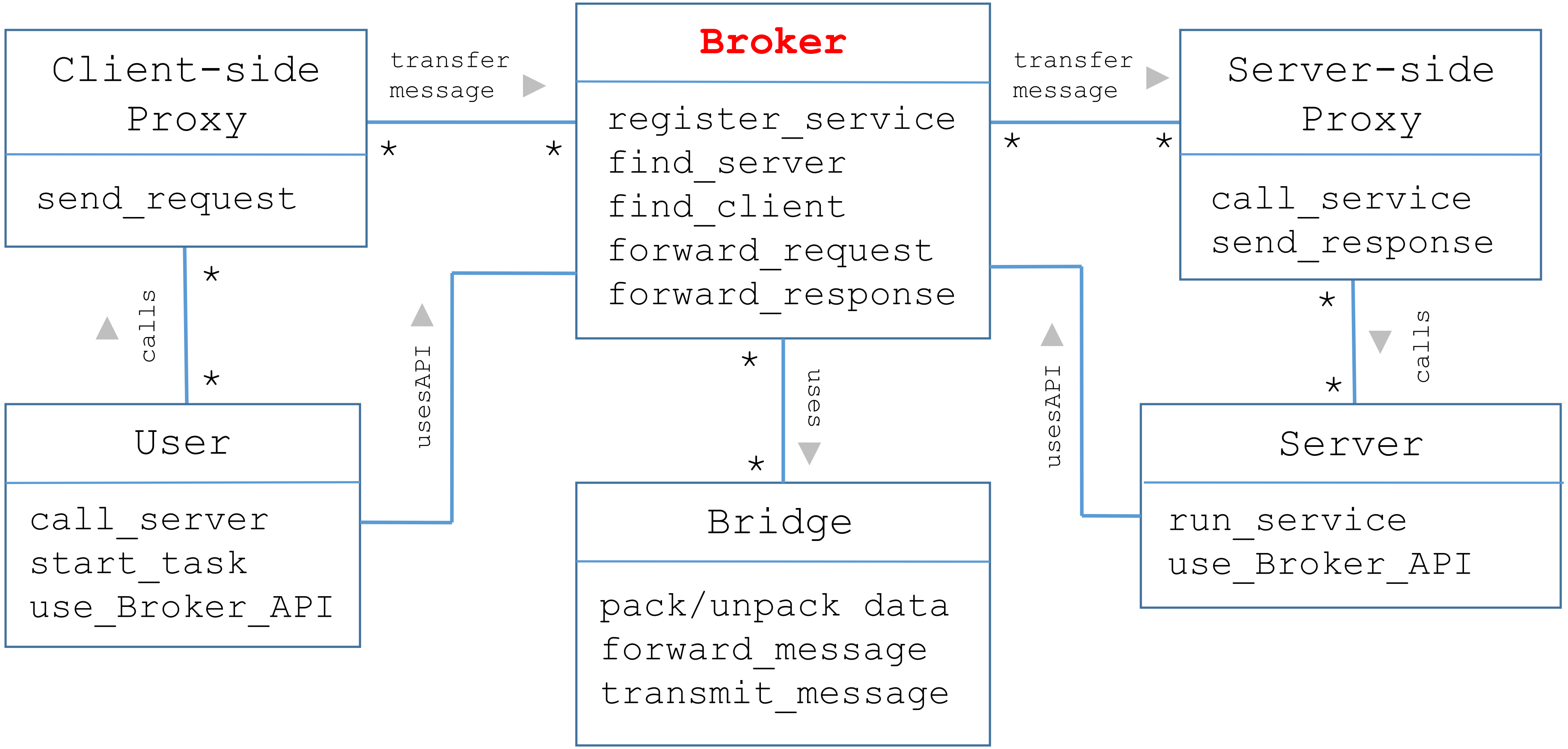


ROLE-BASED ACCESS CONTROL PATTERN:

The `User` and `Role` classes describe registered users and their predefined roles. Users are assigned to roles, roles are given rights according to their functions. The association class `Right` defines the access types that a user within a role is authorized to apply to the `ProtectionObject`.

Example: Broker Pattern

(Buschmann et. al.: A System of Patterns, 1996, ISBN 0-471-95869-7)



BROKER PATTERN:

This pattern is used to structure distributed systems with decoupled components that interact by remote service invocations.

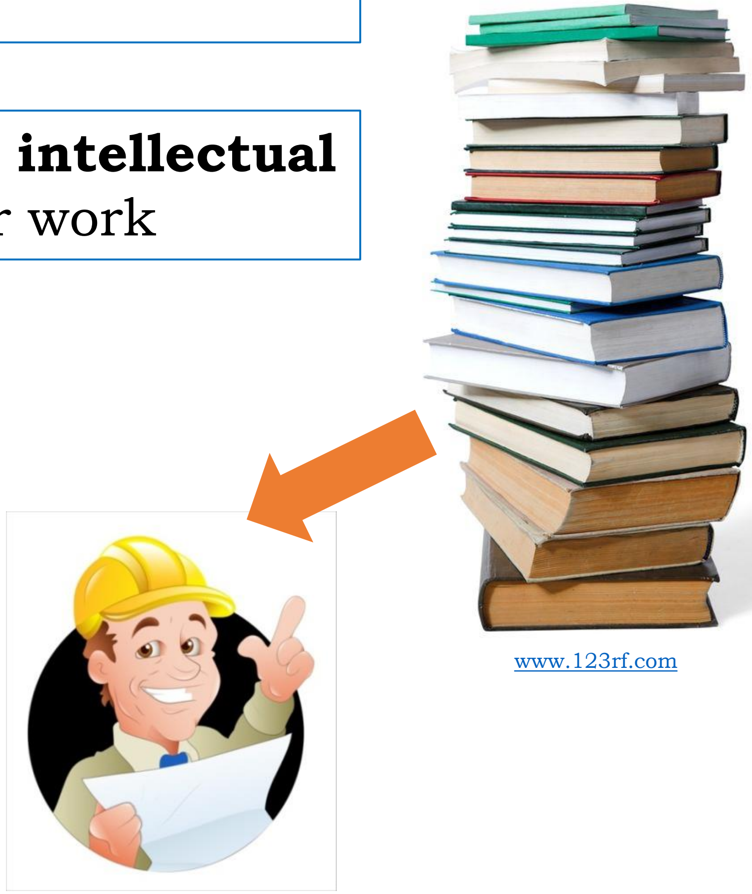
Patterns

Patterns are recorded **architecture and design wisdom** in „canonical“ form. Patterns help you build on the collective experience of skilled architects and software engineers (Buschmann et. al. ISBN 0-471-95869-7)

Patterns are not final, directly applicable solutions! Patterns are **intellectual building blocks** which must be intelligently integrated into your work

Patterns are excellent documentation and communications instruments. They are formal, clear and focussed

There is a *rich literature* about patterns. The future-proof software-system engineer needs to *continuously familiarize* himself with this trove of architecture knowledge!



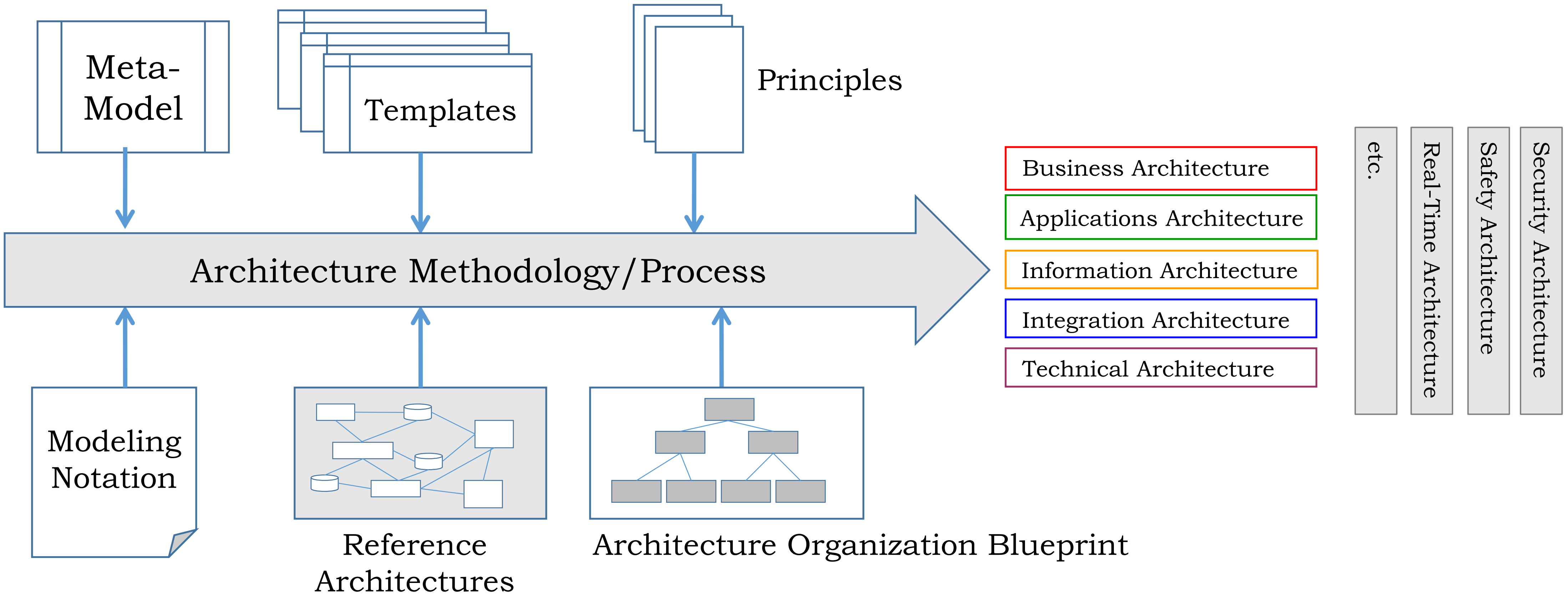
www.123rf.com

Architecture Frameworks

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]



Meta-Model

Templates

Principles

Architecture Methodology/Process

Modeling Notation

Reference Architectures

Architecture Organization Blueprint

Business Architecture

Applications Architecture

Information Architecture

Integration Architecture

Technical Architecture

etc.

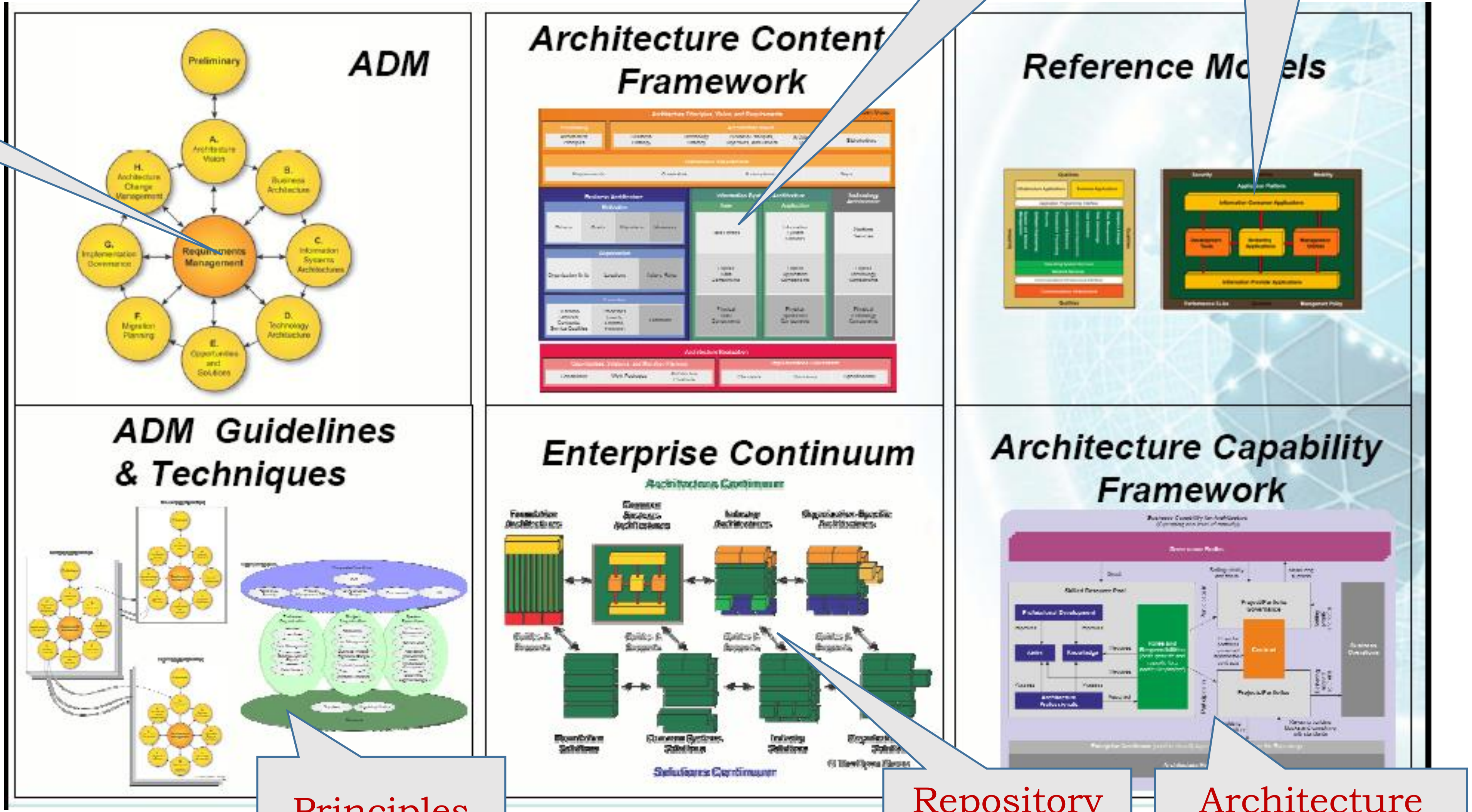
Real-Time Architecture

Safety Architecture

Security Architecture

Example: TOGAF (1/2)

[**T**he **O**pen **G**roup **A**rchitecture **F**ramework] http://www.togaf.org/



Process, Methodology

Deliverables, Artefacts

Reference Models

Principles

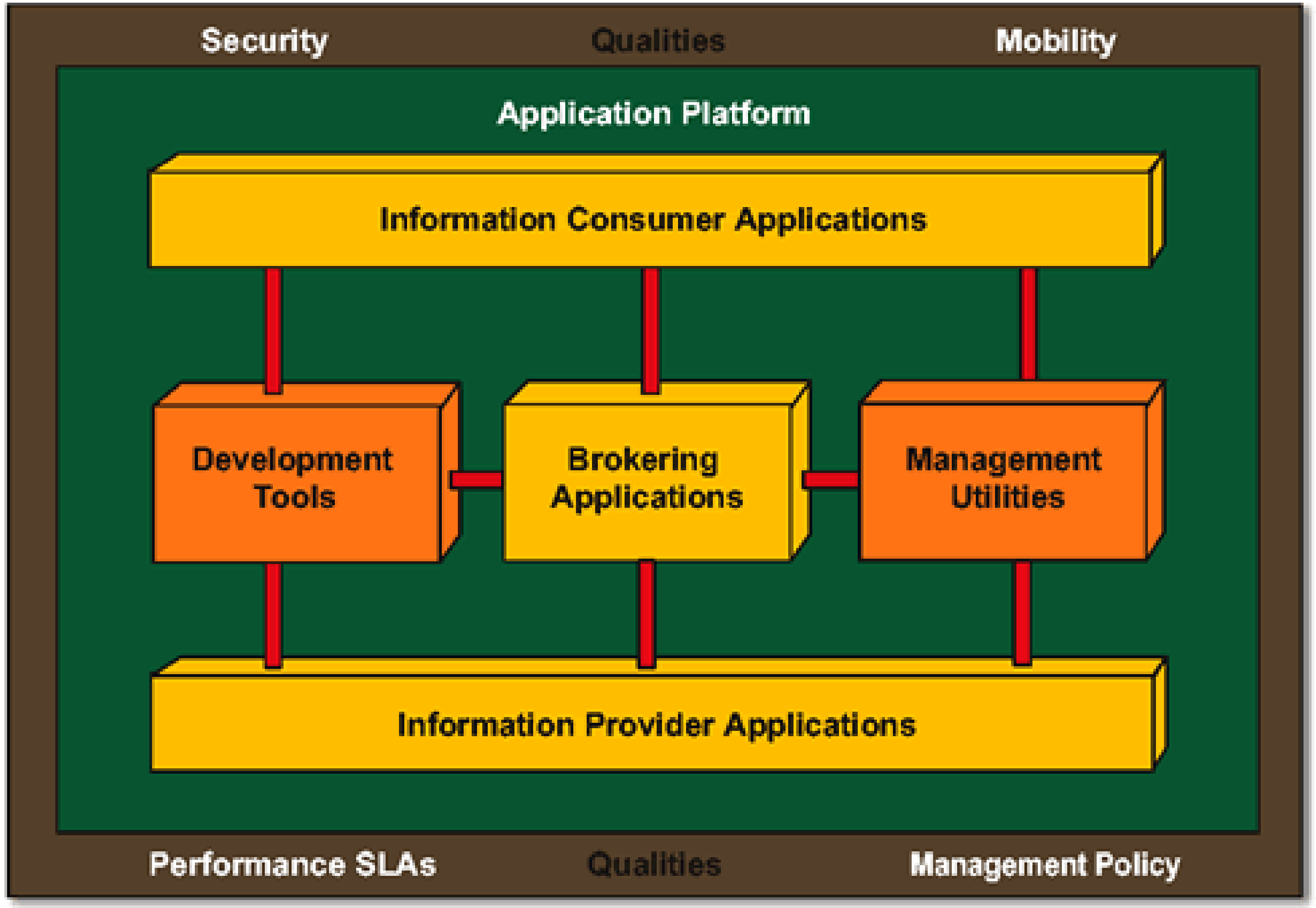
Repository etc.

Architecture Organization

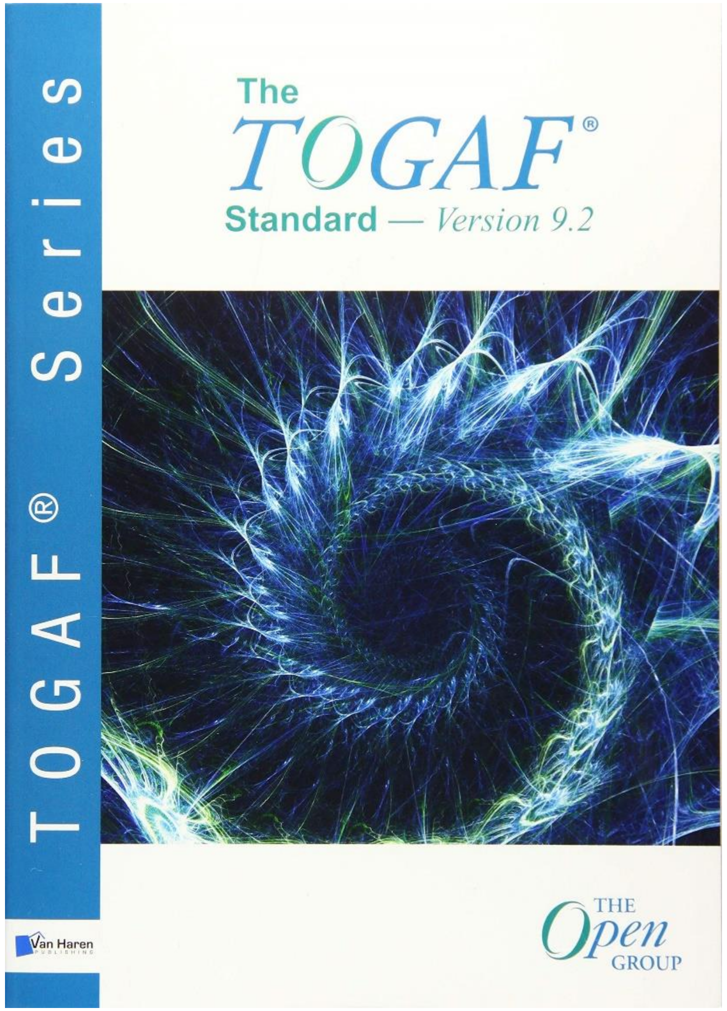
Example: TOGAF (2/2)

[**T**he **O**pen **G**roup **A**rchitecture **F**ramework] http://www.togaf.org/

TOGAF
III-RM
Reference Architecture
(High level)



http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap22.html



Reference Architecture

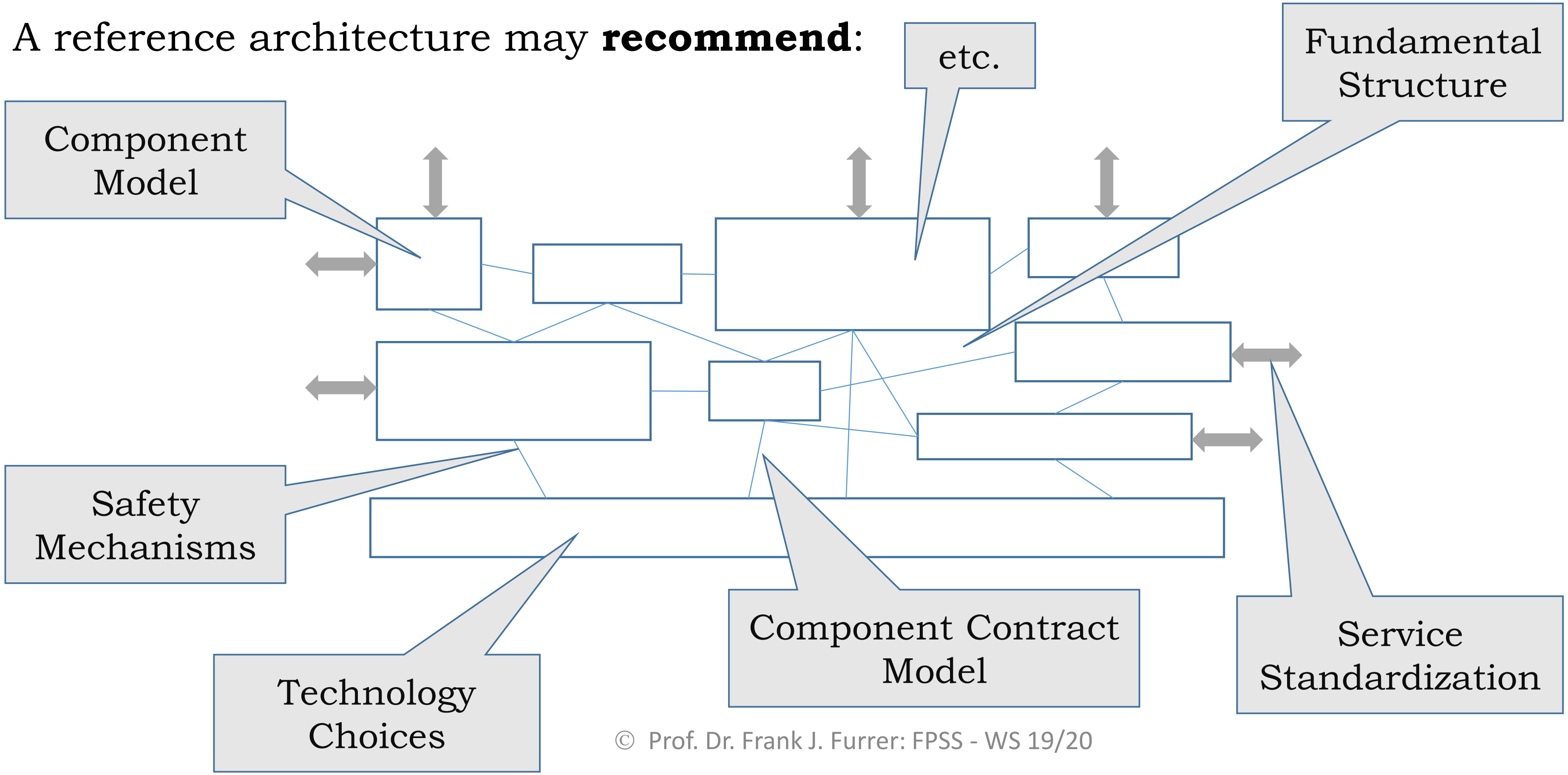
DEFINITIONS

Reference Architecture:

A reference architecture provides a template solution for a *generic architecture* for a particular application domain

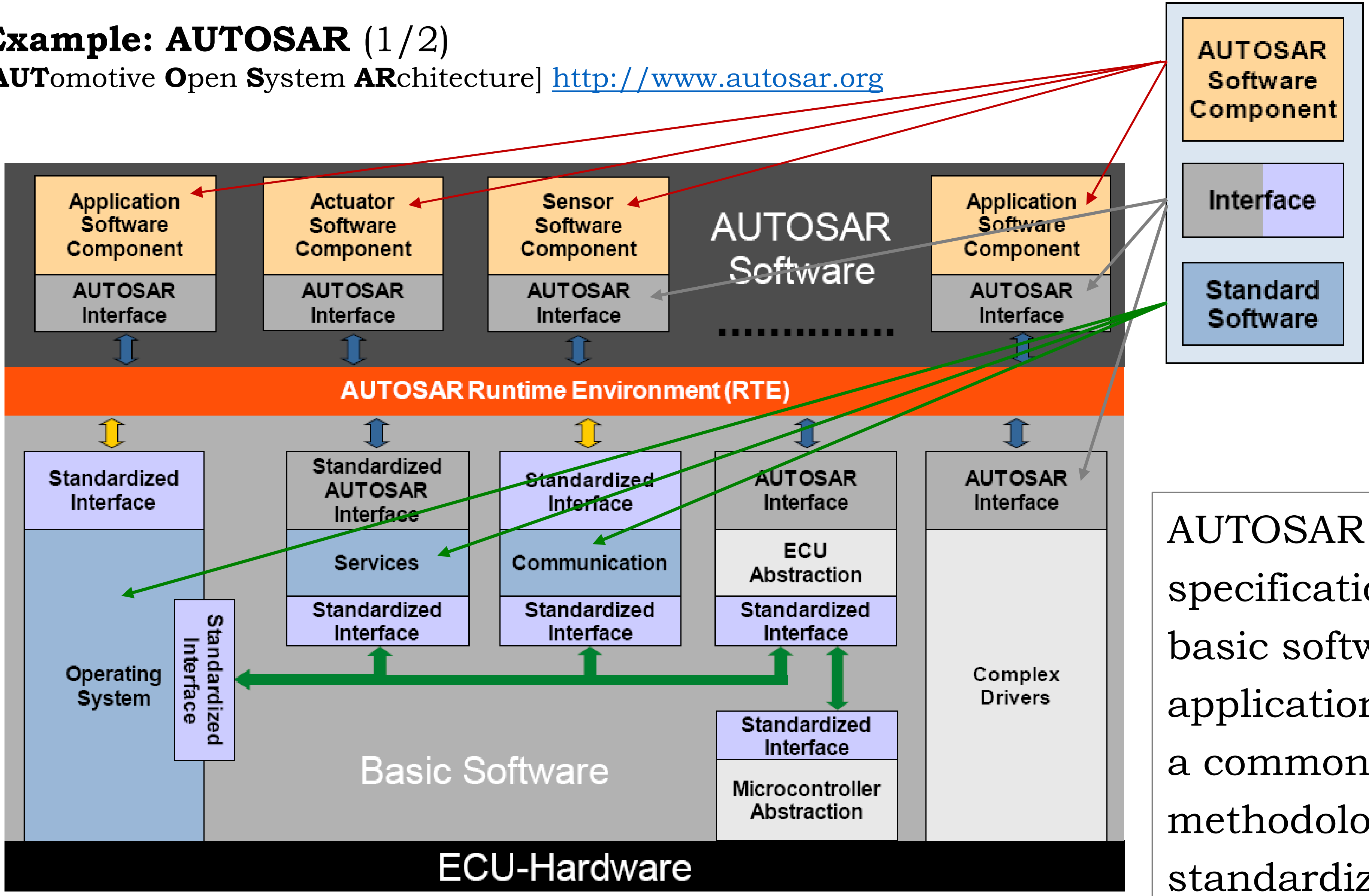
- such as financial systems, automotive, aerospace etc.

A reference architecture may **recommend**:



Example: AUTOSAR (1/2)

[**AUT**omotive **O**pen **S**ystem **AR**chitecture] http://www.autosar.org



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

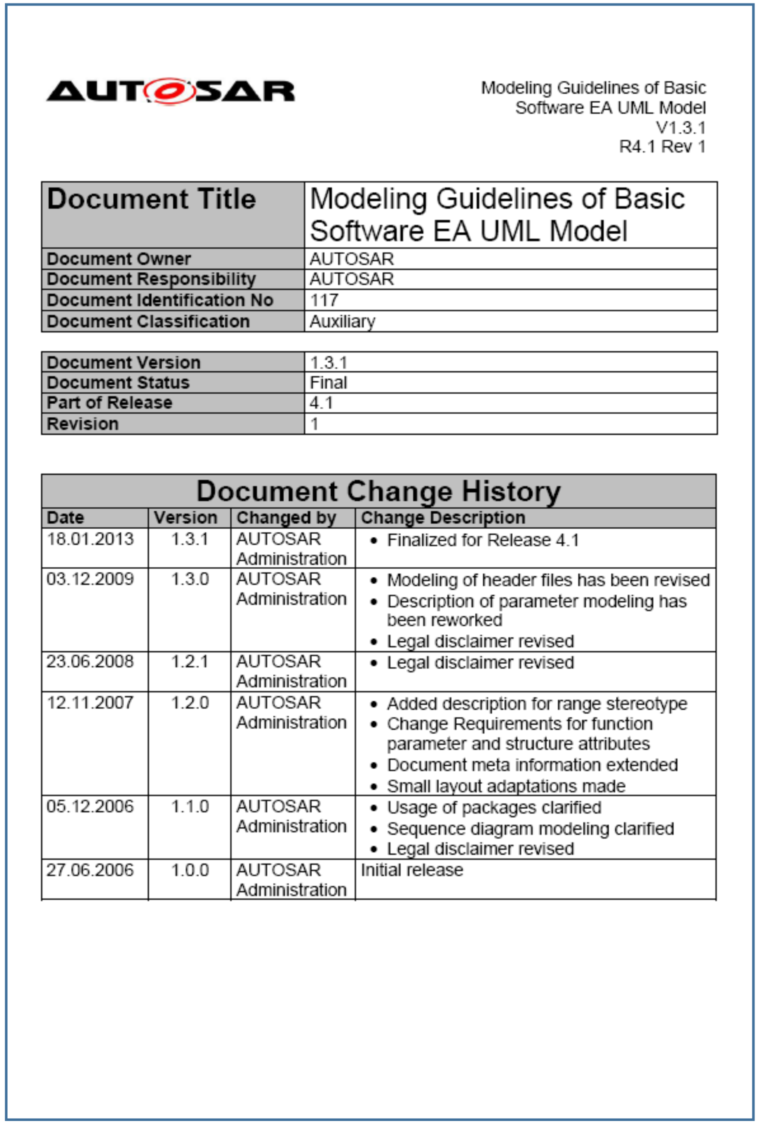
AUTOSAR provides a set of specifications that describes basic software modules, defines application interfaces and builds a common development methodology based on standardized exchange format

[Currently: **13'620 pages**]

Example: AUTOSAR (2/2)

http://www.autosar.org

AUTOSAR is well documented in a number of interesting documents (some only for members)



AUTOSAR

Modeling Guidelines of Basic
Software EA UML Model
V1.3.1
R4.1 Rev 1

Document Title	Modeling Guidelines of Basic Software EA UML Model
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	117
Document Classification	Auxiliary

Document Version	1.3.1
Document Status	Final
Part of Release	4.1
Revision	1

Document Change History

Date	Version	Changed by	Change Description
18.01.2013	1.3.1	AUTOSAR Administration	• Finalized for Release 4.1
03.12.2009	1.3.0	AUTOSAR Administration	• Modeling of header files has been revised • Description of parameter modeling has been reworked • Legal disclaimer revised
23.06.2008	1.2.1	AUTOSAR Administration	• Legal disclaimer revised
12.11.2007	1.2.0	AUTOSAR Administration	• Added description for range stereotype • Change Requirements for function parameter and structure attributes • Document meta information extended • Small layout adaptations made
05.12.2006	1.1.0	AUTOSAR Administration	• Usage of packages clarified • Sequence diagram modeling clarified • Legal disclaimer revised
27.06.2006	1.0.0	AUTOSAR Administration	Initial release

AUTOSAR:

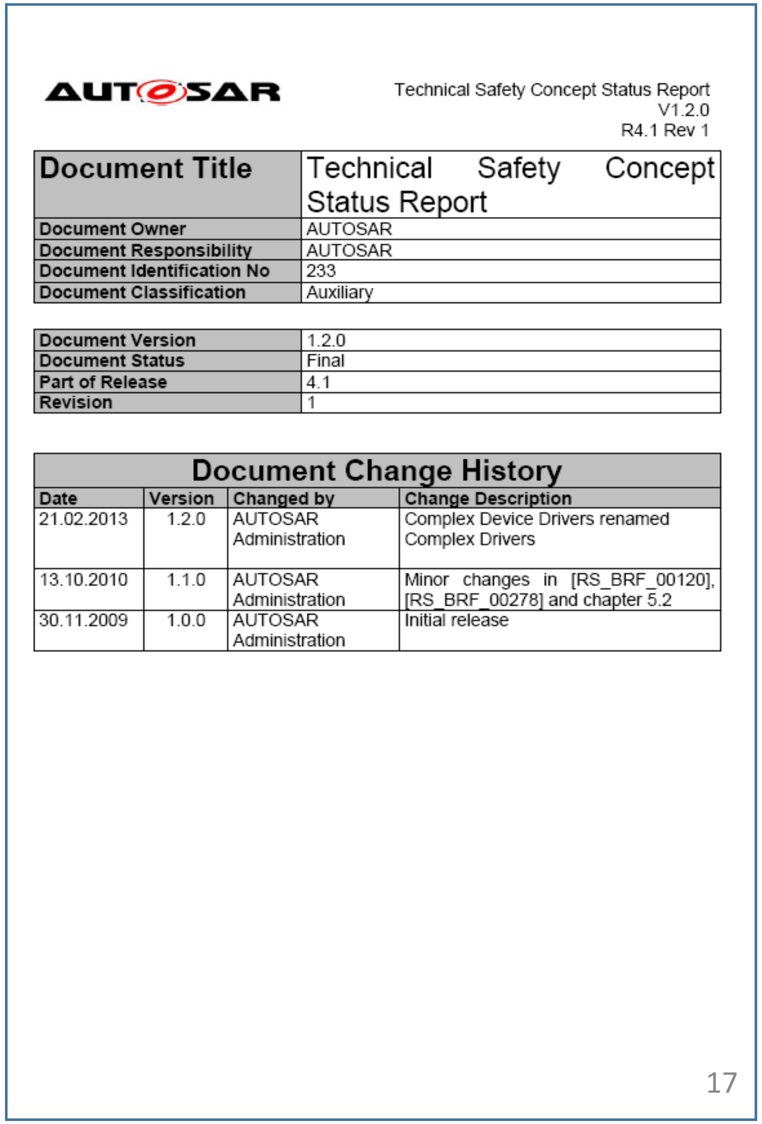
„*Cooperate* on

Standards

–

***Compete* on**

Implementations“



AUTOSAR

Technical Safety Concept Status Report
V1.2.0
R4.1 Rev 1

Document Title	Technical Safety Concept Status Report
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	233
Document Classification	Auxiliary

Document Version	1.2.0
Document Status	Final
Part of Release	4.1
Revision	1

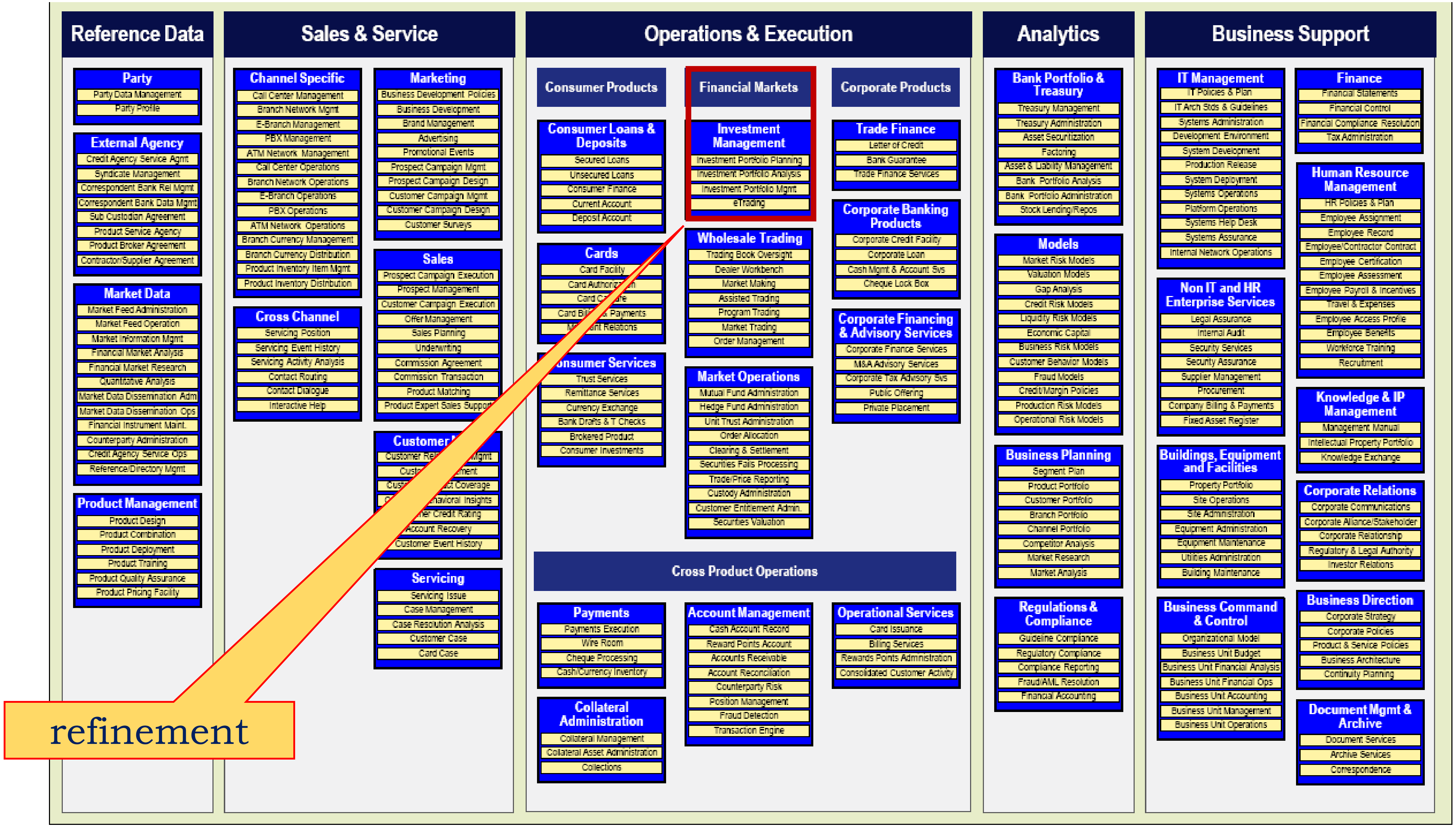
Document Change History

Date	Version	Changed by	Change Description
21.02.2013	1.2.0	AUTOSAR Administration	Complex Device Drivers renamed Complex Drivers
13.10.2010	1.1.0	AUTOSAR Administration	Minor changes in [RS_BRF_00120], [RS_BRF_00278] and chapter 5.2
30.11.2009	1.0.0	AUTOSAR Administration	Initial release

Example: BIAN

Banking **I**ndustry **A**rchitecture **N**etwork: http://www.bian.org

BIAN standardizes the full functional landscape of a *financial institution*



Reference Data

- **Party**
  - Party Data Management
  - Party Profile
- **External Agency**
  - Credit Agency Service Agmt
  - Syndicate Management
  - Correspondent Bank Rel Mgmt
  - Correspondent Bank Data Mgmt
  - Sub Custodian Agreement
  - Product Service Agency
  - Product Broker Agreement
  - Contractor/Supplier Agreement
- **Market Data**
  - Market Feed Administration
  - Market Feed Operation
  - Market Information Mgmt
  - Financial Market Analysis
  - Financial Market Research
  - Quantitative Analysis
  - Market Data Dissemination Adm
  - Market Data Dissemination Ops
  - Financial Instrument Maint.
  - Counterparty Administration
  - Credit Agency Service Ops
  - Reference/Directory Mgmt
- **Product Management**
  - Product Design
  - Product Combination
  - Product Deployment
  - Product Training
  - Product Quality Assurance
  - Product Pricing Facility

Sales & Service

- **Channel Specific**
  - Call Center Management
  - Branch Network Mgmt
  - E-Branch Management
  - PBX Management
  - ATM Network Management
  - Call Center Operations
  - Branch Network Operations
  - E-Branch Operations
  - PBX Operations
  - ATM Network Operations
  - Branch Currency Management
  - Branch Currency Distribution
  - Product Inventory Item Mgmt
  - Product Inventory Distribution
- **Cross Channel**
  - Servicing Position
  - Servicing Event History
  - Servicing Activity Analysis
  - Contact Routing
  - Contact Dialogue
  - Interactive Help
- **Marketing**
  - Business Development Policies
  - Business Development
  - Brand Management
  - Advertising
  - Promotional Events
  - Prospect Campaign Mgmt
  - Prospect Campaign Design
  - Customer Campaign Mgmt
  - Customer Campaign Design
  - Customer Surveys
- **Sales**
  - Prospect Campaign Execution
  - Prospect Management
  - Customer Campaign Execution
  - Offer Management
  - Sales Planning
  - Underwriting
  - Commission Agreement
  - Commission Transaction
  - Product Matching
  - Product Expert Sales Support
- **Customer**
  - Customer Re Mgmt
  - Custo ment
  - Cust ct Coverage
  - C havioral Insights
  - mer Credit Rating
  - Account Recovery
  - Customer Event History
- **Servicing**
  - Servicing Issue
  - Case Management
  - Case Resolution Analysis
  - Customer Case
  - Card Case

Operations & Execution

Consumer Products

- **Consumer Loans & Deposits**
  - Secured Loans
  - Unsecured Loans
  - Consumer Finance
  - Current Account
  - Deposit Account
- **Cards**
  - Card Facility
  - Card Authoriz n
  - Card C re
  - Card Bil & Payments
  - M nt Relations
- **onsumer Services**
  - Trust Services
  - Remittance Services
  - Currency Exchange
  - Bank Drafts & T Checks
  - Brokered Product
  - Consumer Investments

Financial Markets

- **Investment Management**
  - Investment Portfolio Planning
  - Investment Portfolio Analysis
  - Investment Portfolio Mgmt
  - eTrading
- **Wholesale Trading**
  - Trading Book Oversight
  - Dealer Workbench
  - Market Making
  - Assisted Trading
  - Program Trading
  - Market Trading
  - Order Management
- **Market Operations**
  - Mutual Fund Administration
  - Hedge Fund Administration
  - Unit Trust Administration
  - Order Allocation
  - Clearing & Settlement
  - Securities Fails Processing
  - Trade/Price Reporting
  - Custody Administration
  - Customer Entitlement Admin.
  - Securities Valuation

Corporate Products

- **Trade Finance**
  - Letter of Credit
  - Bank Guarantee
  - Trade Finance Services
- **Corporate Banking Products**
  - Corporate Credit Facility
  - Corporate Loan
  - Cash Mgmt & Account Svs
  - Cheque Lock Box
- **Corporate Financing & Advisory Services**
  - Corporate Finance Services
  - M&A Advisory Services
  - Corporate Tax Advisory Svs
  - Public Offering
  - Private Placement

Cross Product Operations

- **Payments**
  - Payments Execution
  - Wire Room
  - Cheque Processing
  - Cash/Currency Inventory
- **Collateral Administration**
  - Collateral Management
  - Collateral Asset Administration
  - Collections
- **Account Management**
  - Cash Account Record
  - Reward Points Account
  - Accounts Receivable
  - Account Reconciliation
  - Counterparty Risk
  - Position Management
  - Fraud Detection
  - Transaction Engine
- **Operational Services**
  - Card Issuance
  - Billing Services
  - Rewards Points Administration
  - Consolidated Customer Activity

Analytics

- **Bank Portfolio & Treasury**
  - Treasury Management
  - Treasury Administration
  - Asset Securitization
  - Factoring
  - Asset & Liability Management
  - Bank Portfolio Analysis
  - Bank Portfolio Administration
  - Stock Lending/Repos
- **Models**
  - Market Risk Models
  - Valuation Models
  - Gap Analysis
  - Credit Risk Models
  - Liquidity Risk Models
  - Economic Capital
  - Business Risk Models
  - Customer Behavior Models
  - Fraud Models
  - Credit/Margin Policies
  - Production Risk Models
  - Operational Risk Models
- **Business Planning**
  - Segment Plan
  - Product Portfolio
  - Customer Portfolio
  - Branch Portfolio
  - Channel Portfolio
  - Competitor Analysis
  - Market Research
  - Market Analysis
- **Regulations & Compliance**
  - Guideline Compliance
  - Regulatory Compliance
  - Compliance Reporting
  - Fraud/AML Resolution
  - Financial Accounting

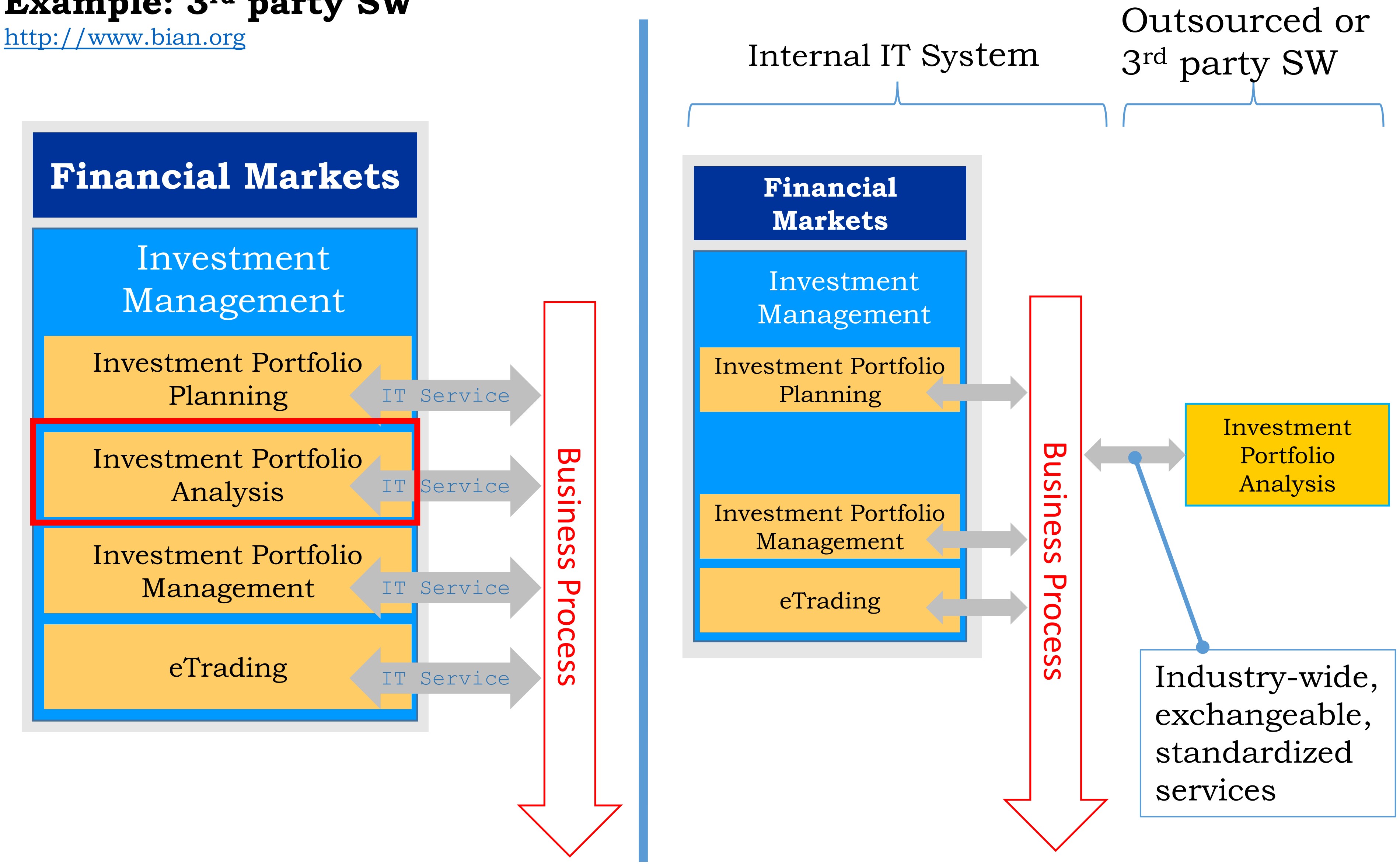
Business Support

- **IT Management**
  - IT Policies & Plan
  - IT Arch Stds & Guidelines
  - Systems Administration
  - Development Environment
  - System Development
  - Production Release
  - System Deployment
  - Systems Operations
  - Platform Operations
  - Systems Help Desk
  - Systems Assurance
  - Internal Network Operations
- **Non IT and HR Enterprise Services**
  - Legal Assurance
  - Internal Audit
  - Security Services
  - Security Assurance
  - Supplier Management
  - Procurement
  - Company Billing & Payments
  - Fixed Asset Register
- **Buildings, Equipment and Facilities**
  - Property Portfolio
  - Site Operations
  - Site Administration
  - Equipment Administration
  - Equipment Maintenance
  - Utilities Administration
  - Building Maintenance
- **Business Command & Control**
  - Organizational Model
  - Business Unit Budget
  - Business Unit Financial Analysis
  - Business Unit Financial Ops
  - Business Unit Accounting
  - Business Unit Management
  - Business Unit Operations
- **Finance**
  - Financial Statements
  - Financial Control
  - Financial Compliance Resolution
  - Tax Administration
- **Human Resource Management**
  - HR Policies & Plan
  - Employee Assignment
  - Employee Record
  - Employee/Contractor Contract
  - Employee Certification
  - Employee Assessment
  - Employee Payroll & Incentives
  - Travel & Expenses
  - Employee Access Profile
  - Employee Benefits
  - Workforce Training
  - Recruitment
- **Knowledge & IP Management**
  - Management Manual
  - Intellectual Property Portfolio
  - Knowledge Exchange
- **Corporate Relations**
  - Corporate Communications
  - Corporate Alliance/Stakeholder
  - Corporate Relationship
  - Regulatory & Legal Authority
  - Investor Relations
- **Business Direction**
  - Corporate Strategy
  - Corporate Policies
  - Product & Service Policies
  - Business Architecture
  - Continuity Planning
- **Document Mgmt & Archive**
  - Document Services
  - Archive Services
  - Correspondence

refinement

Example: 3rd party SW

http://www.bian.org



Financial Markets

Investment Management

Investment Portfolio Planning

Investment Portfolio Analysis

Investment Portfolio Management

eTrading

IT Service

IT Service

IT Service

IT Service

Business Process

Internal IT System

Outsourced or 3rd party SW

Financial Markets

Investment Management

Investment Portfolio Planning

Investment Portfolio Management

eTrading

Business Process

Investment Portfolio Analysis

Industry-wide, exchangeable, standardized services

The BIAN reference architecture allows “plug & play” with own and with 3rd party components

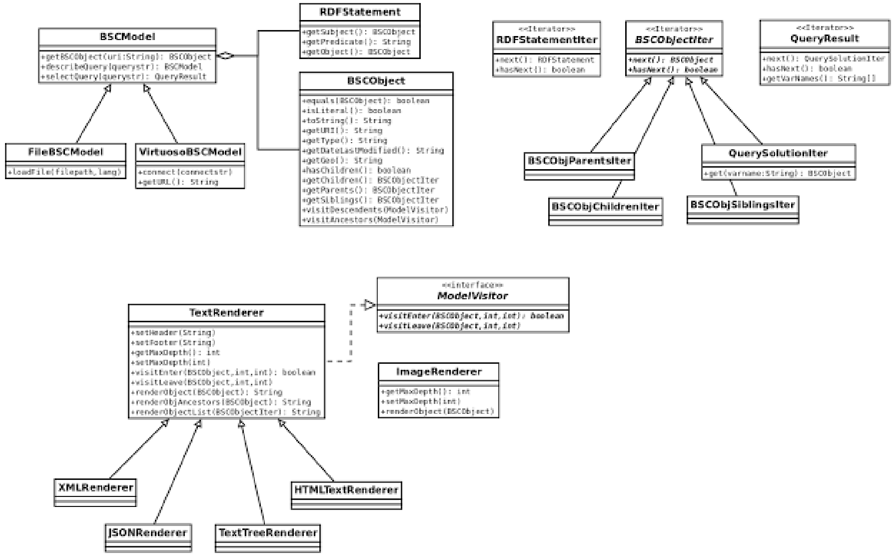
Operationalization

Reference Architectures
Architecture Frameworks
Architecture Patterns

Learn & understand

Future-Proof Software-Systems Engineer

Apply & enforce



http://biscicol.blogspot.ch/2011/06/biscicol-core-software-architecture.html

A7

Architecture Principle A7:

Reference Architectures, Frameworks and Patterns

(1) If a reference architecture exists in your field of application, extract the most of it to improve your architecture and to achieve industry-cooperation capability;

(2) While developing and maintaining your enterprise architecture, chose an adequate architecture framework and try to use as much knowledge from this framework as possible;

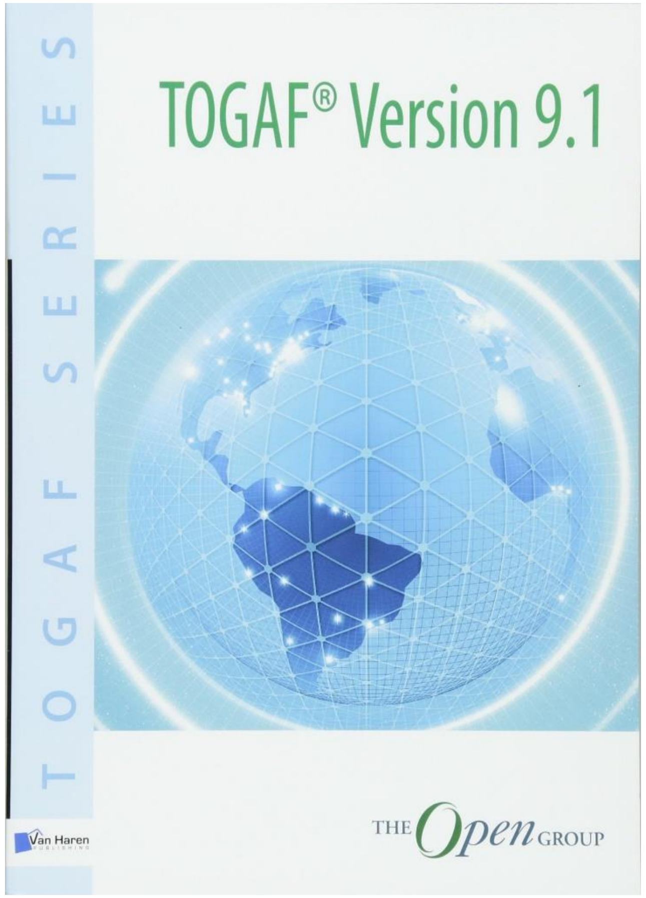
(3) In your organization, maintain a well-organized, comprehensive repository of patterns for all levels and areas of software development. Actively encourage and review the usage of patterns;

(4) Make the use of reference architectures, architecture frameworks, and patterns an integral, valuable part of your architecture development process.

Justification: The usage of proven, well-documented architecture knowledge (greatly) improves your own architecture and designs.

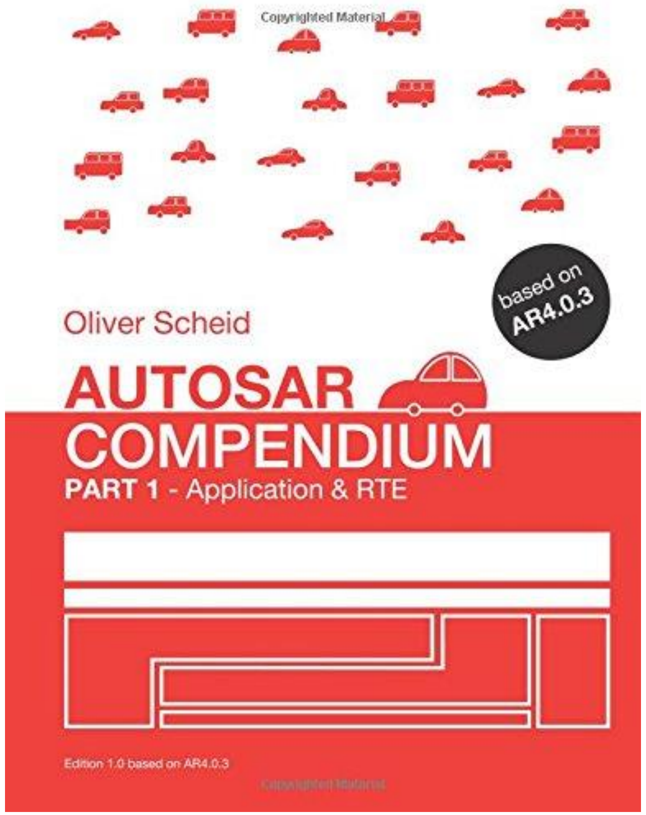
Future-Proof Software-Systems [Part 3A]

Textbook



The Open Group:
TOGAF® Version 9.1
Van Haren Publishing, 2011. ISBN 978-9-0875-3679-4

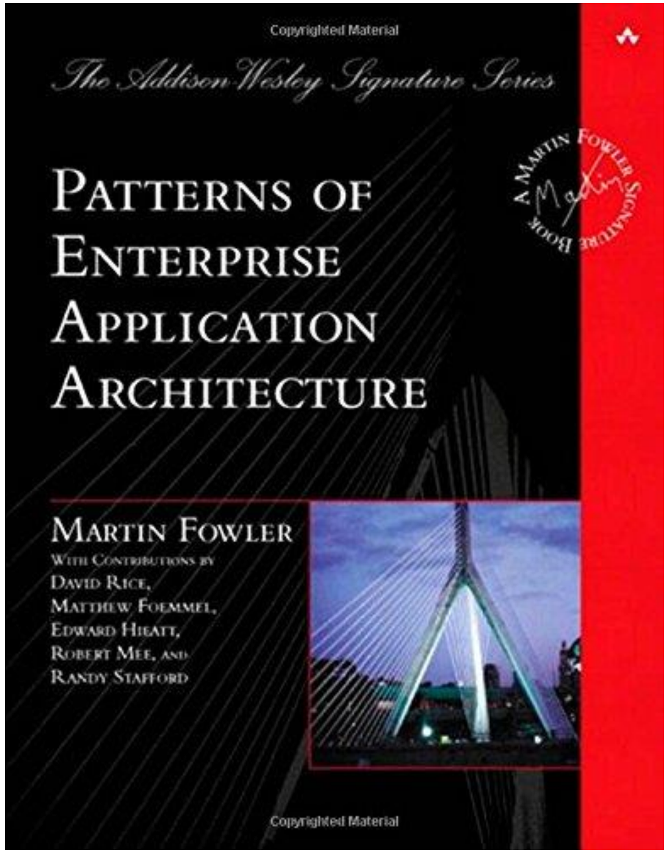
Textbook



Oliver Scheid:
AUTOSAR Compendium, Part 1 – *Application & RTE*
CreateSpace Independent Publishing Platform, 2015. ISBN 978-1-5027-5152-2

Future-Proof Software-Systems [Part 3A]

Textbook



Copyrighted Material

The Addison Wesley Signature Series

PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE

A Martin Fowler Signature Book

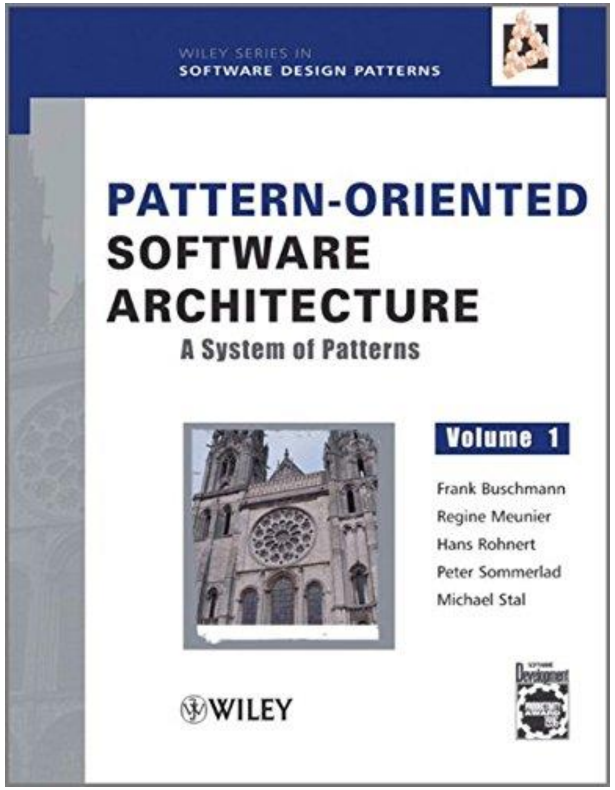
MARTIN FOWLER

With Contributions by
David Rice,
Matthew Foemmel,
Edward Hieatt,
Robert Mee, and
Randy Stafford

Copyrighted Material

Martin Fowler:
Patterns of Enterprise Application Architecture
Addison Wesley, Inc., USA, 2002. ISBN 978-0-321-12742-6

Textbook



WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A System of Patterns

Volume 1

Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal

WILEY

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal:
Pattern-Oriented Software Architecture, Vol. 1 – *A System of Patterns*
John Wiley & Sons., Inc., USA, 1996. ISBN 978-0 471 95869 7

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

A8

Architecture Principle A8:

Reuse and Parametrization

Reuse in Software-Systems Engineering

http://www.lybecker.com

DEFINITIONS

Reuse:

Utilization of Software-Artefacts in another Context or Application

https://static.pexels.com

«Good» reuse can have a strong ***reward*** (in quality, time and money)

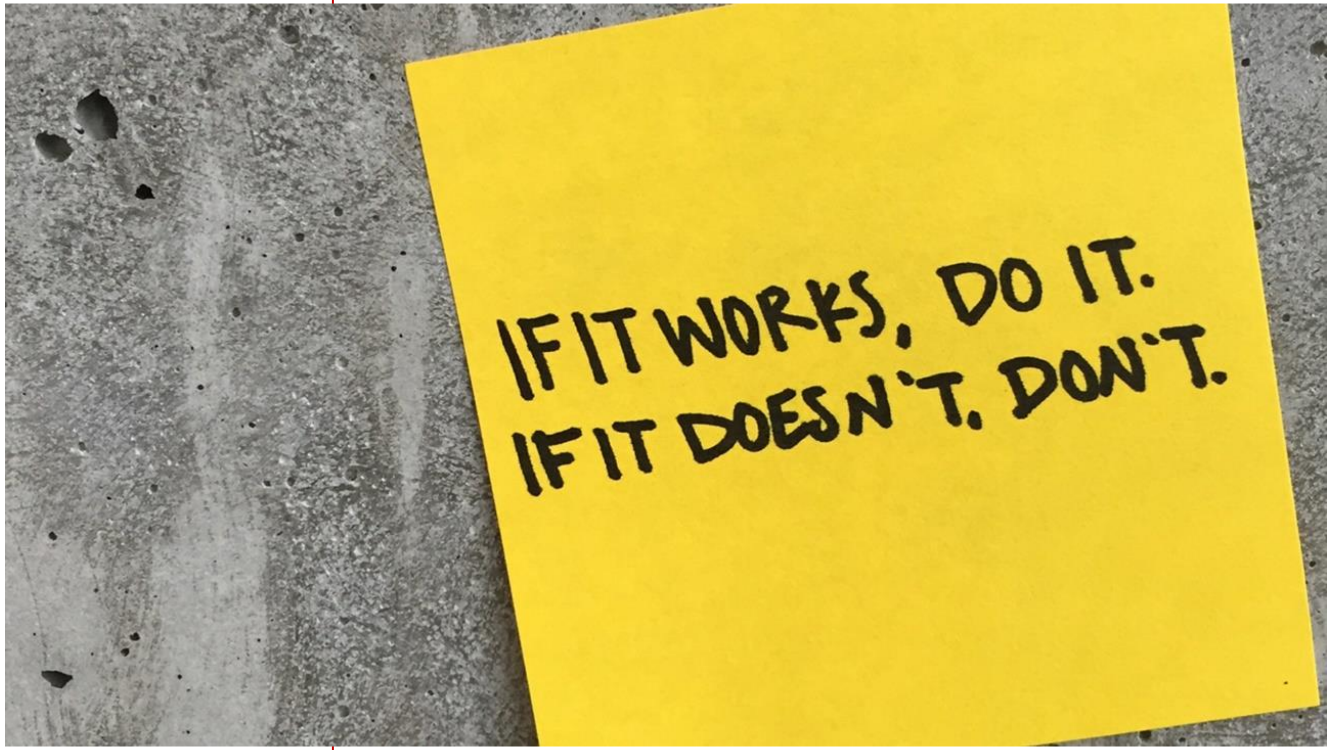
CAUTION:

Reuse can be a ***danger*** for the consistency and integrity of an architecture

http://www.safetysign.com

Successful reuse can be done with:

- Requirements
- Specifications
- Reference architectures
- Patterns
- Code (Functionality)
- Data (Information)
- Algorithms
- Configurations
- Documentation
- Models
- ….



https://www.artefactgroup.com

⇒ Rules for reuse

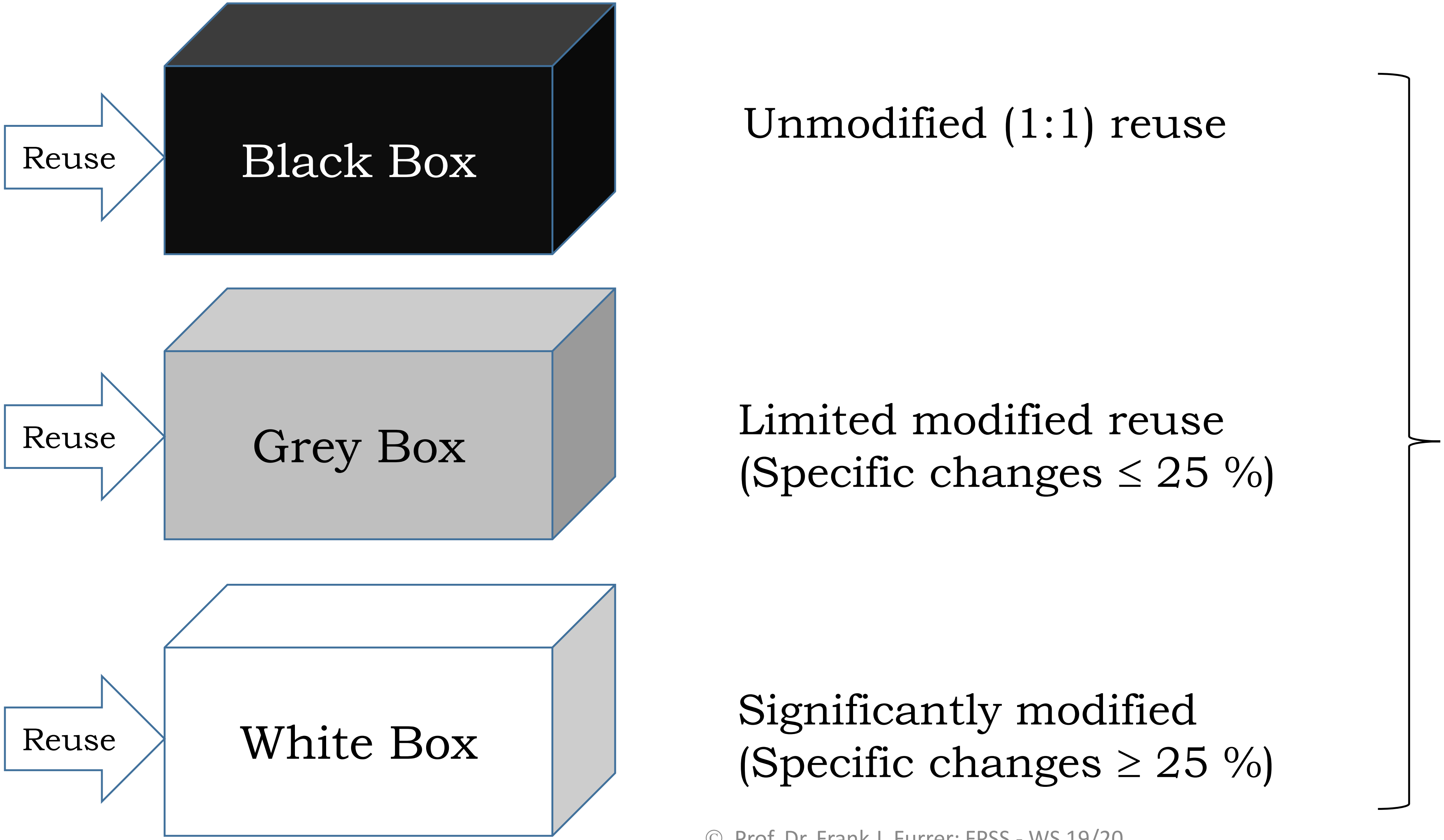
⇒ Rules for reuse

Successful Reuse requires:

- a company-wide *reuse strategy*
- a strong *reuse organization*
- a *dedicated, committed* management
- Adequate development & evolution *processes*

http://www.clipartbest.com

***Types** of Reuse*



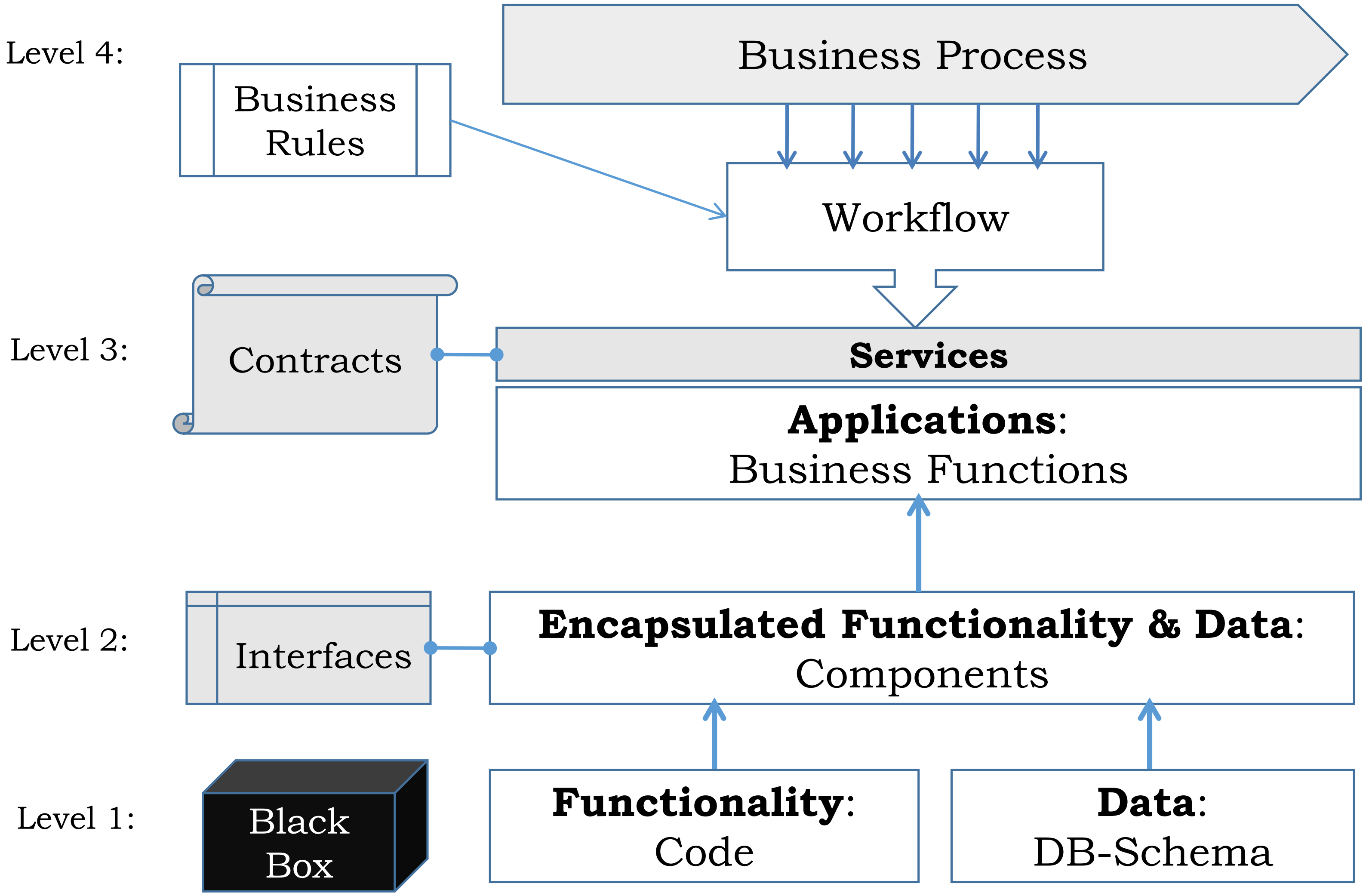
Reuse → Black Box: Unmodified (1:1) reuse

Reuse → Grey Box: Limited modified reuse (Specific changes ≤ 25 %)

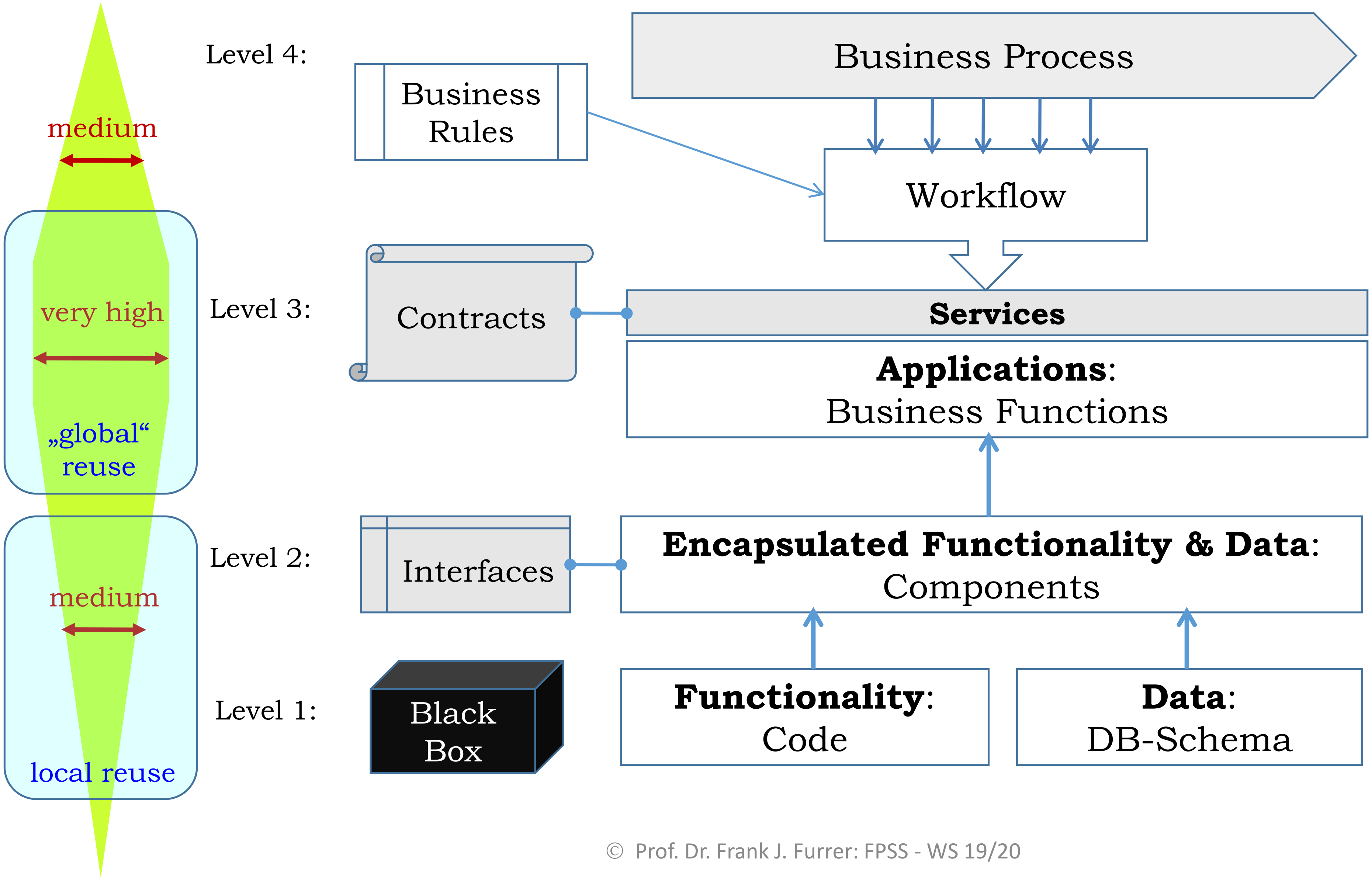
Reuse → White Box: Significantly modified (Specific changes ≥ 25 %)

Important for functionality (**code**) and information (**data**)

***Levels* of Reuse**



Value of Reuse

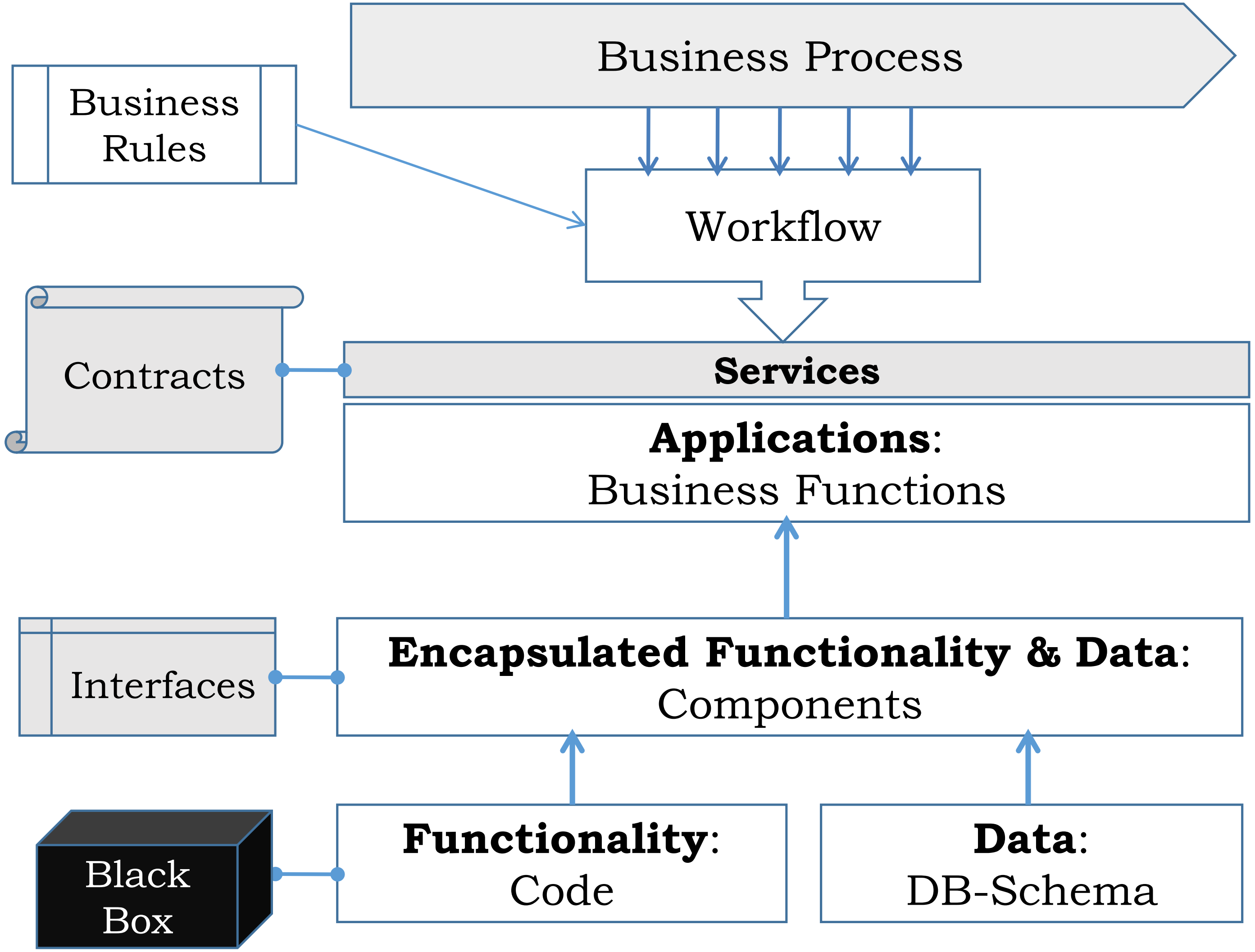


Business rules are **reused** to implement business process steps

Services are ***called*** to build applications or systems

Components are ***composed*** to applications

Fragments of code and data are ***reused*** in programs

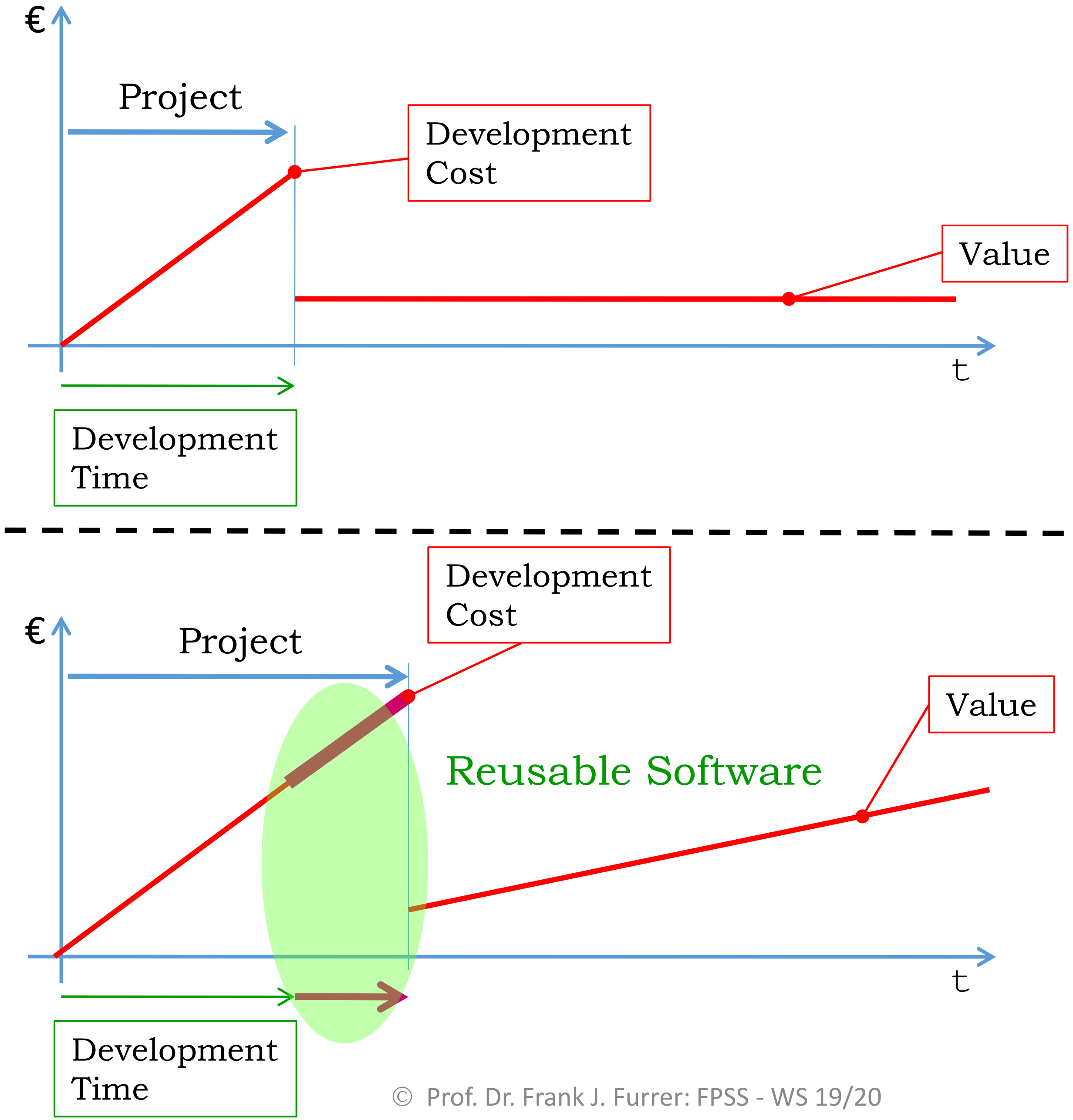


Future-Proof Software-Systems [Part 3B]

Business Case of Reuse

one-time use

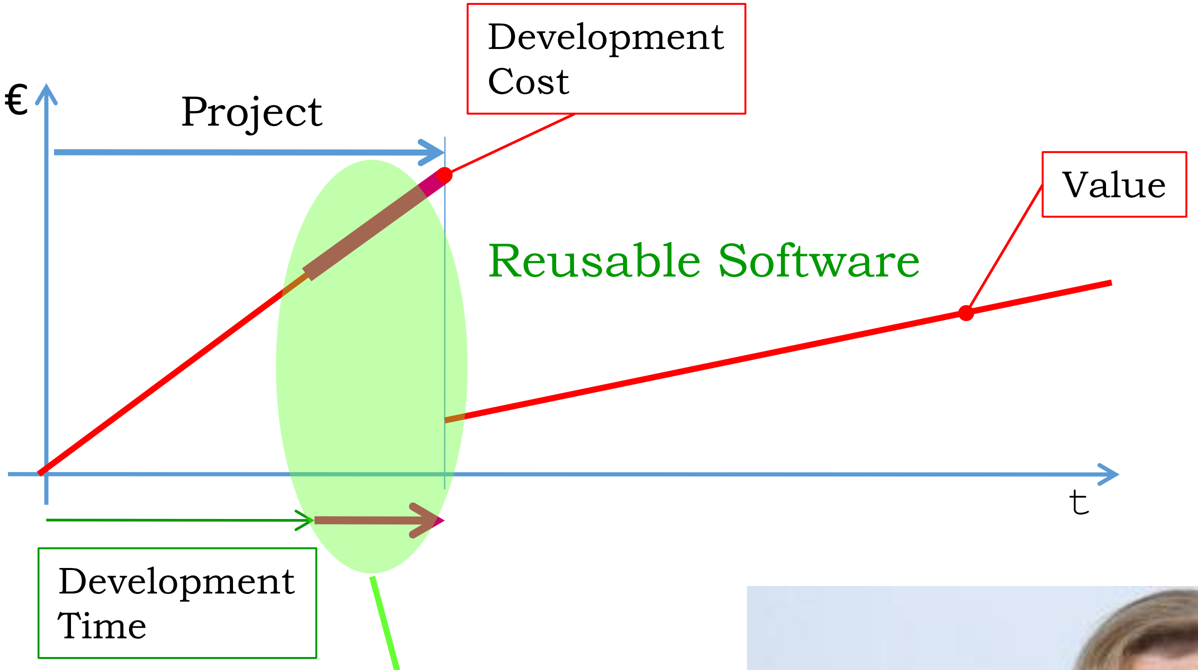
Reuse (n-time use)



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

Reuse-strategy

reuse



Who is paying?


http://blogs.lawyers.com

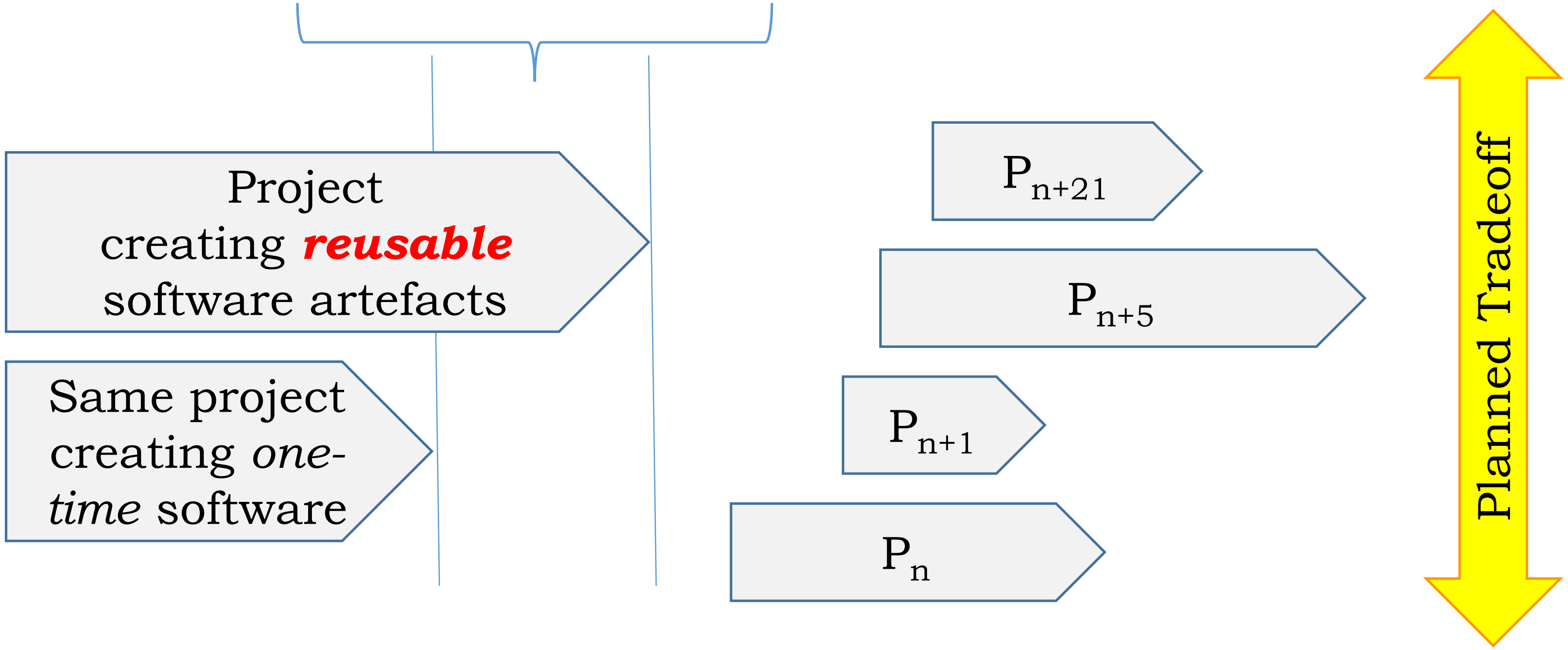
Reusable software requires **considerable more effort** in planning, design, development and implementation

⇒ Enforced reuse-strategy required

Reuse-strategy

The project has additional cost and longer time-to-market

→ *Reuse penalty*



Project creating ***reusable*** software artefacts

Same project creating *one-time* software

P_{n+21}

P_{n+5}

P_{n+1}

P_n

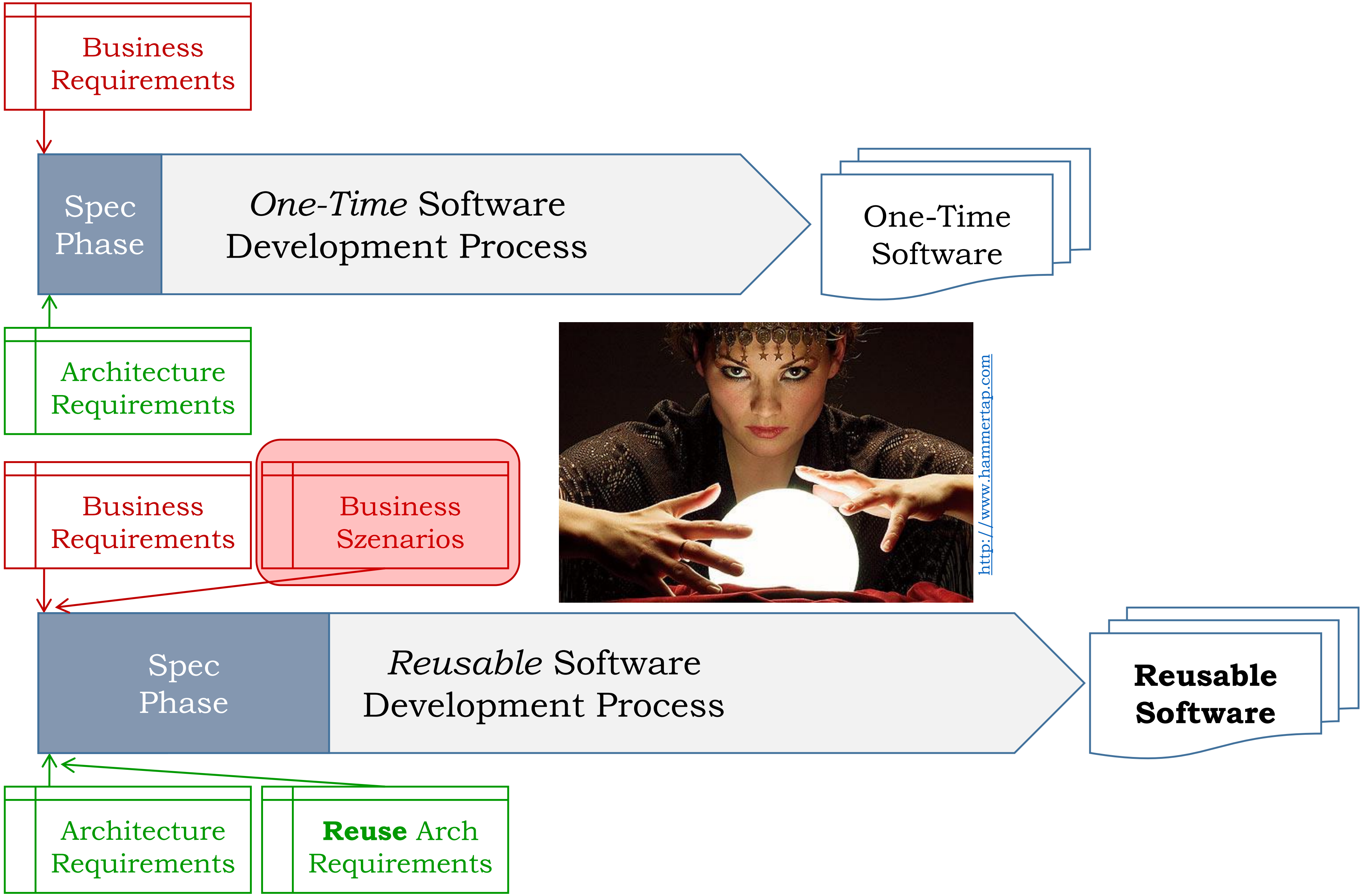
Planned Tradeoff

https://upload.wikimedia.org

All projects reusing the software have lower cost and shorter time-to-market

→ *Reuse benefit*

Reuse-strategy



Business Requirements

Spec Phase

One-Time Software Development Process

One-Time Software

Architecture Requirements

Business Requirements

Business Szenarios

http://www.hammertap.com

Spec Phase

Reusable Software Development Process

Reusable Software

Architecture Requirements

Reuse Arch Requirements

***Elements* of successful Reuse**

http://www.amisinsurance.com

❶ A *dedicated* and *committed* management

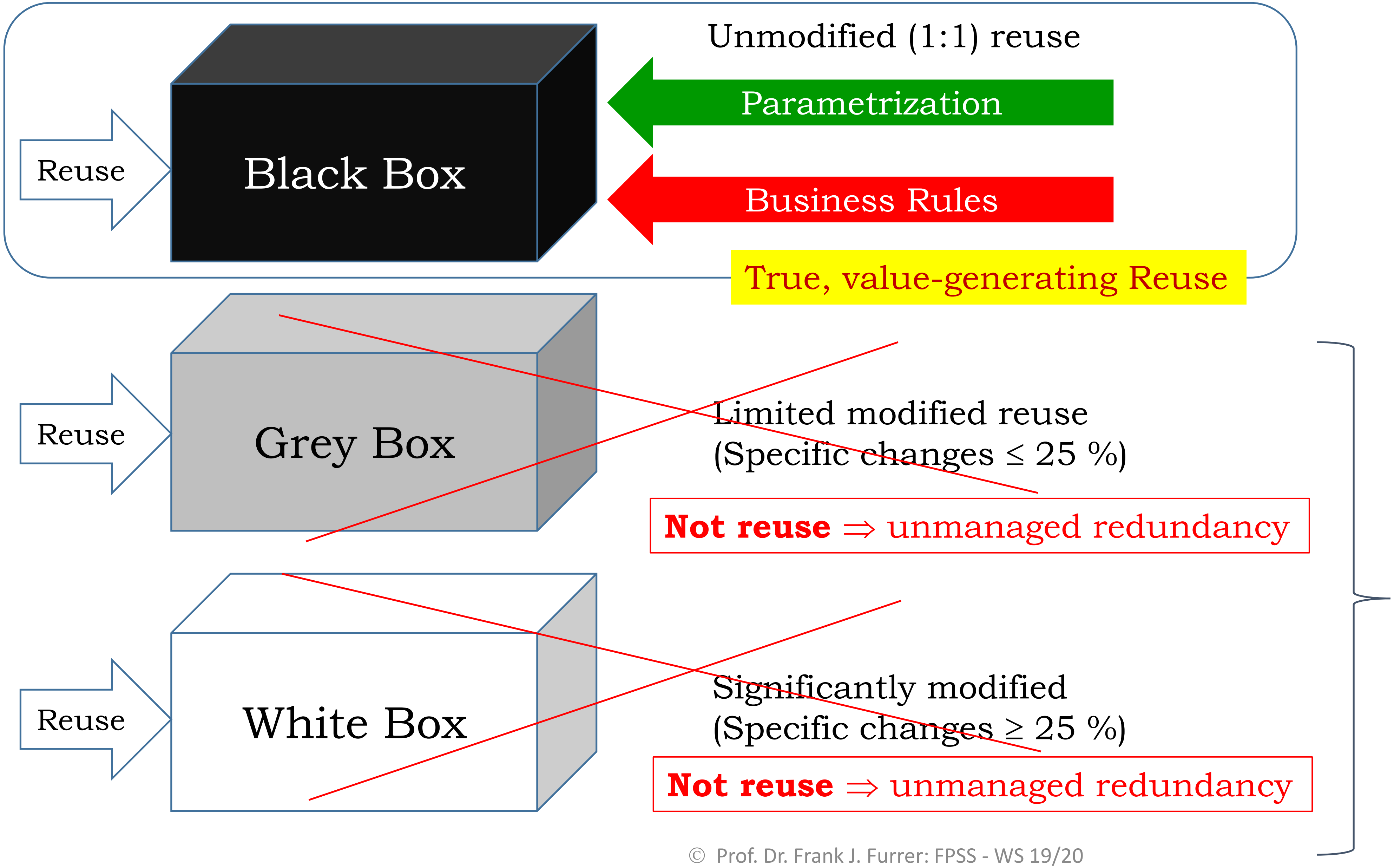
❷ A company-wide *reuse strategy*

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

http://www.globalnpsolutions.com

❸ A strong *reuse organization*

Good software ❹ architects



Reuse

Black Box

Unmodified (1:1) reuse

Parametrization

Business Rules

True, value-generating Reuse

Reuse

~~Grey Box~~

~~Limited modified reuse~~
~~(Specific changes ≤ 25 %)~~

Not reuse ⇒ unmanaged redundancy

Reuse

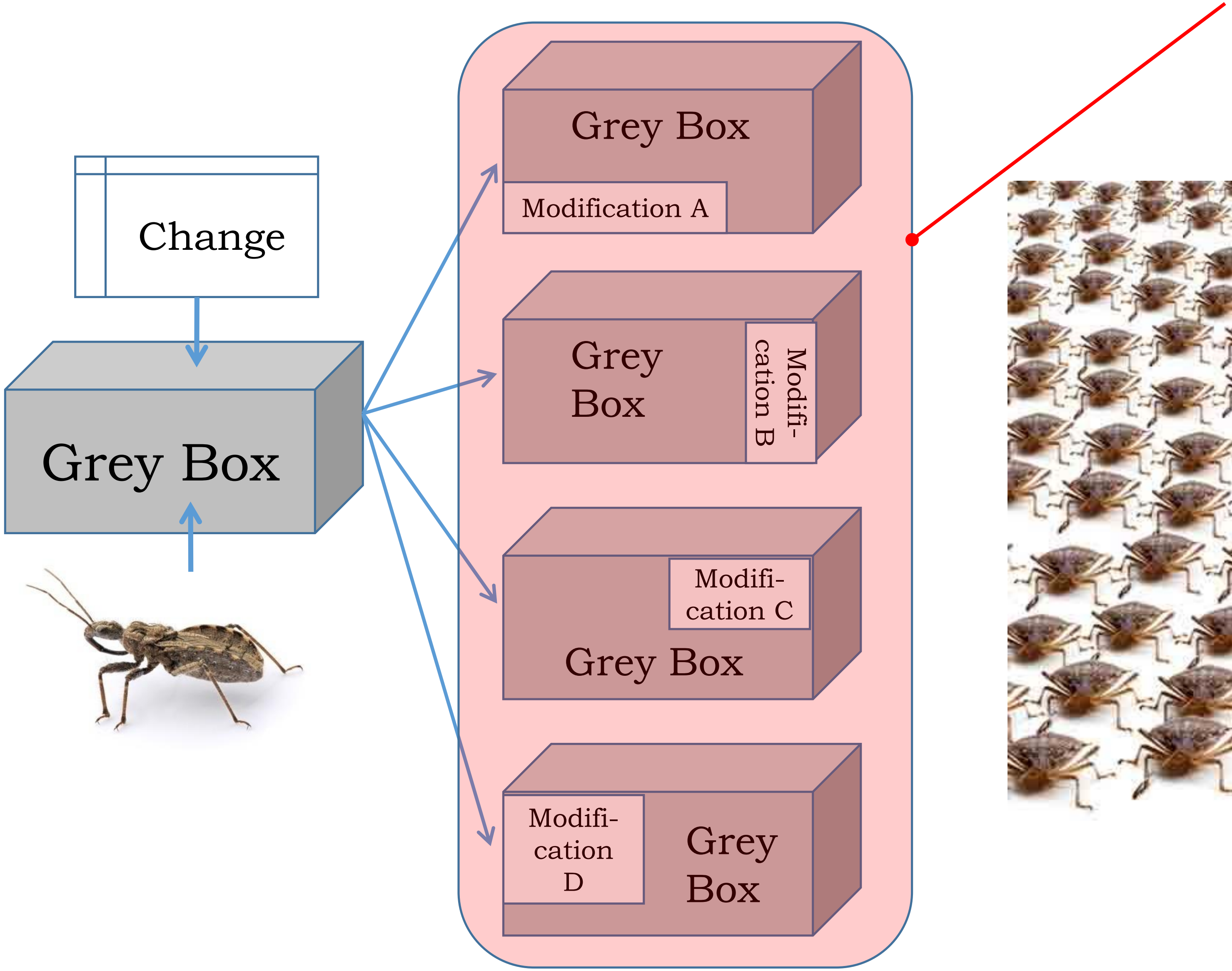
~~White Box~~

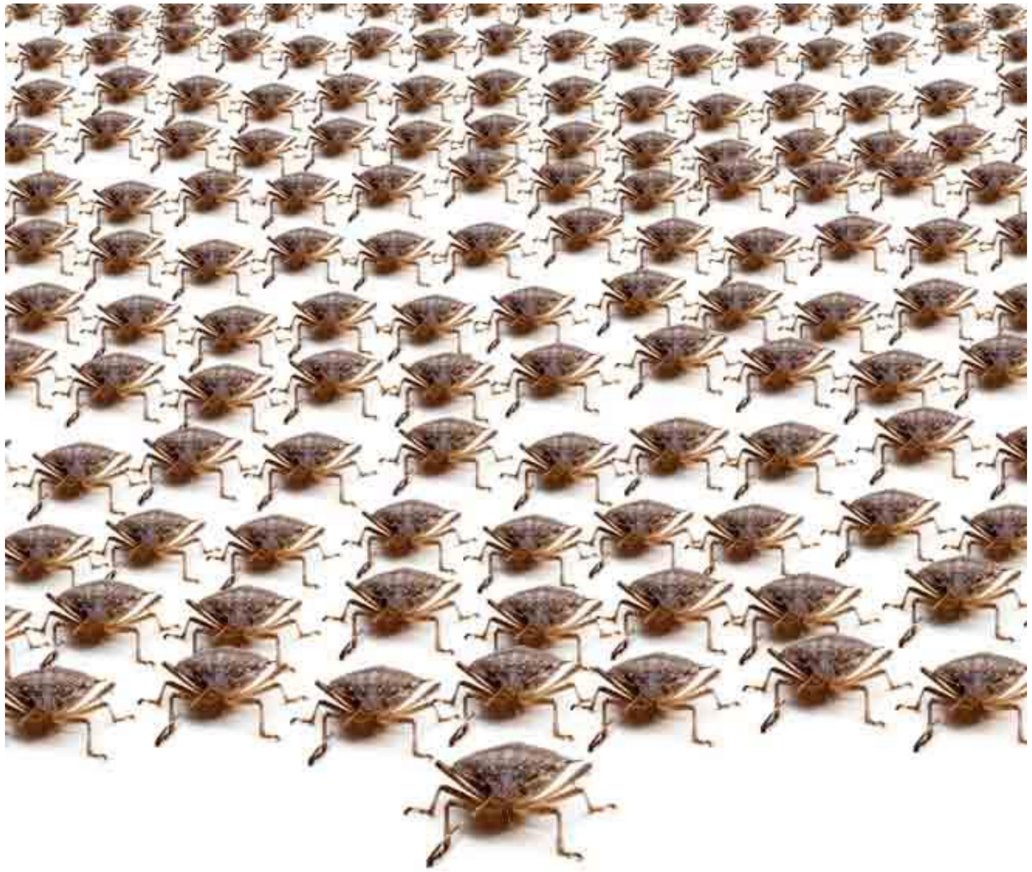
~~Significantly modified~~
~~(Specific changes ≥ 25 %)~~

Not reuse ⇒ unmanaged redundancy

http://sr.photos2.fotosearch.com

Grey Box Modification ⇒ Divergence (Unmanaged Redundancy)





http://www.veggiegardeningtips.com

Component/Service Reuse-Cycle

Black Box

http://www.dorgersoft.com

Owner

New Reqs

Black Box

Black Box

Vx.y + 0.1

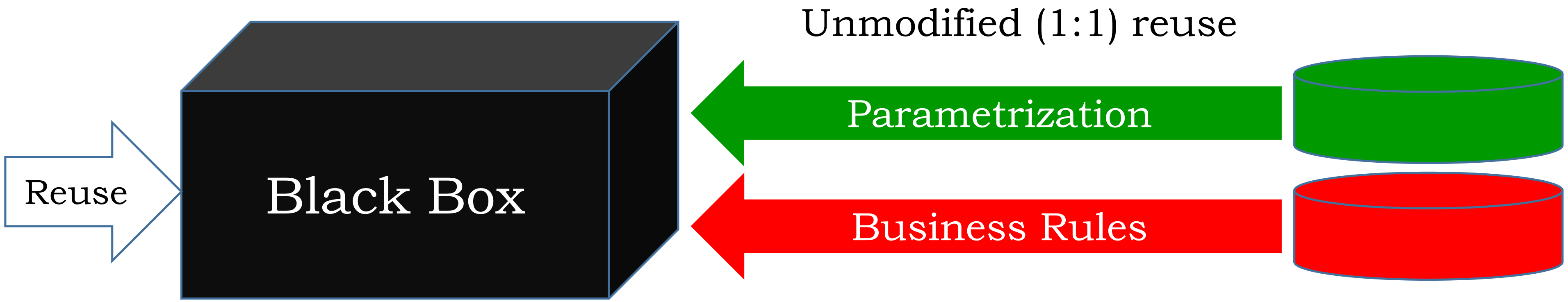
Repository

Grey Box

White Box

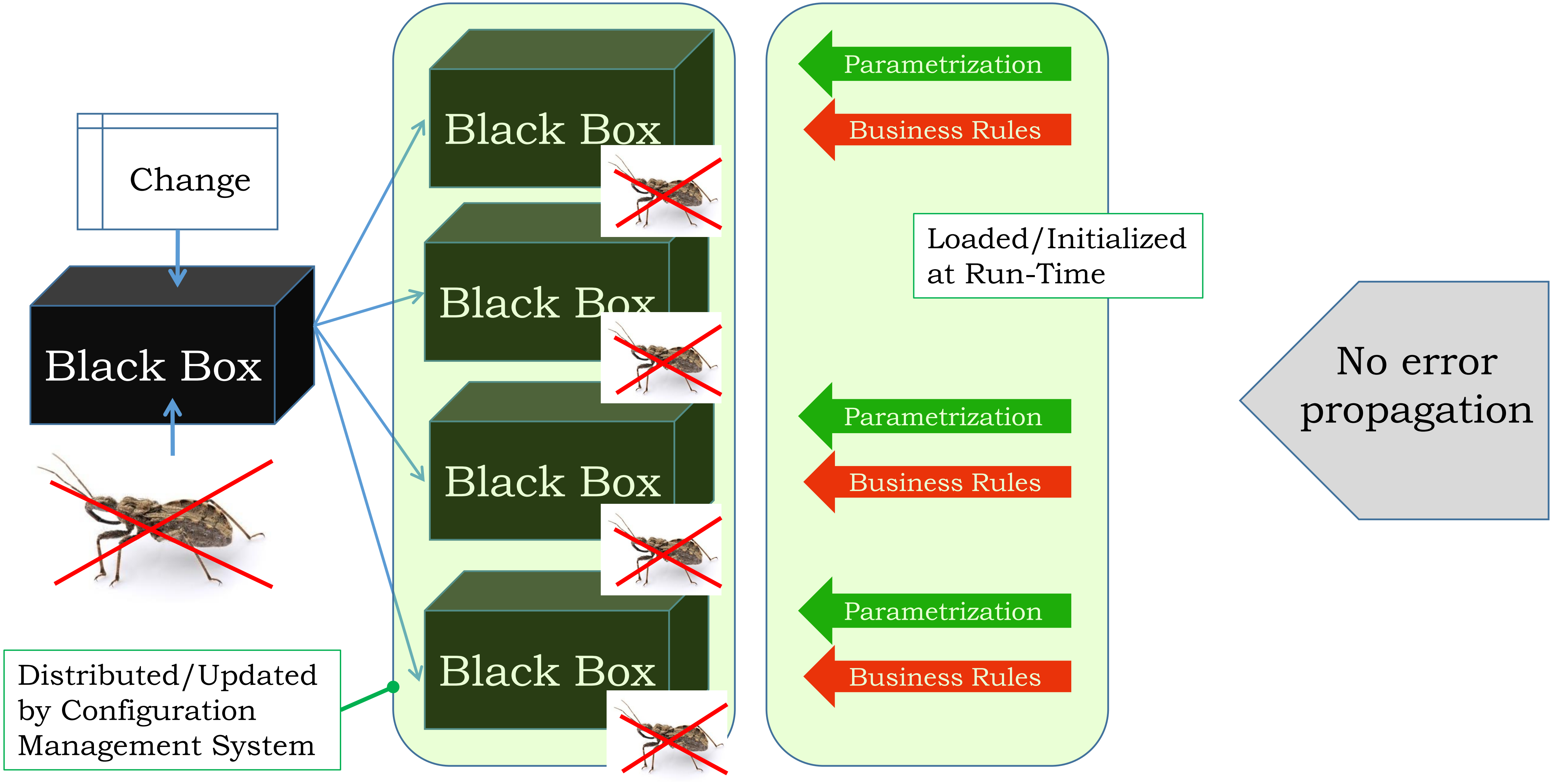
Re-utilization
(intelligent way ...)

Parametrization and Business Rules



Parametrization: Selection of a predefined behaviour of the black box by parameters stored *outside* of the black box (Not part of the black box functionality or data). The parameters are loaded at run-time. New versions of the black box interpret the parameters correctly.

Business Rules: Business rules are specified in BR-languages and define processing logic – instead of having the processing logic implemented in code within the black box (Not part of the black box functionality or data). The business rules are loaded at run-time. New versions of the black box interpret the business rules correctly



Change

Black Box

Black Box

Black Box

Black Box

Black Box

Distributed/Updated by Configuration Management System

Parametrization

Business Rules

Loaded/Initialized at Run-Time

Parametrization

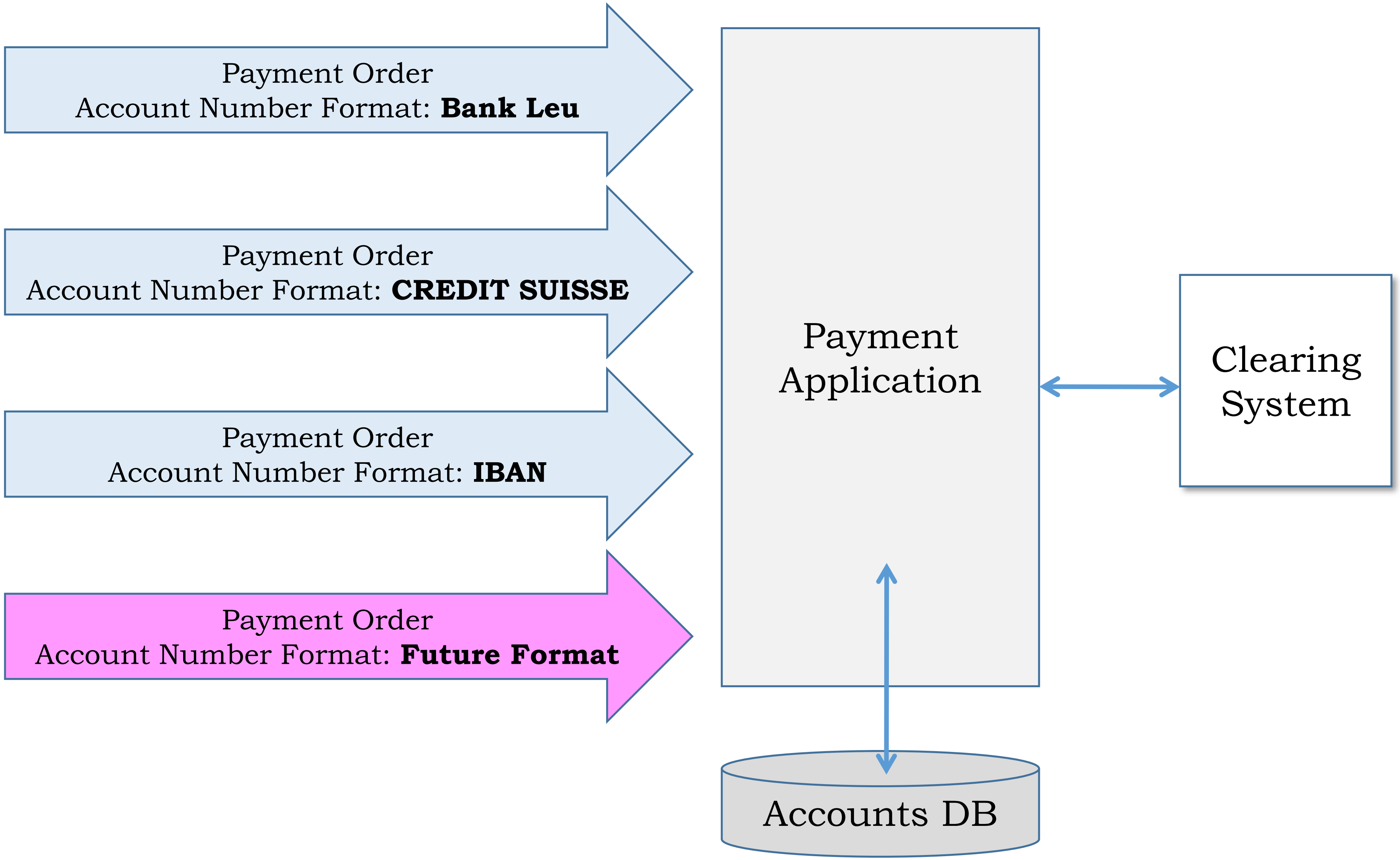
Business Rules

Parametrization

Business Rules

No error propagation

Parametrization Example: Different Account Number Formats



Payment Order
Account Number Format: **Bank Leu**

Payment Order
Account Number Format: **CREDIT SUISSE**

Payment Order
Account Number Format: **IBAN**

Payment Order
Account Number Format: **Future Format**

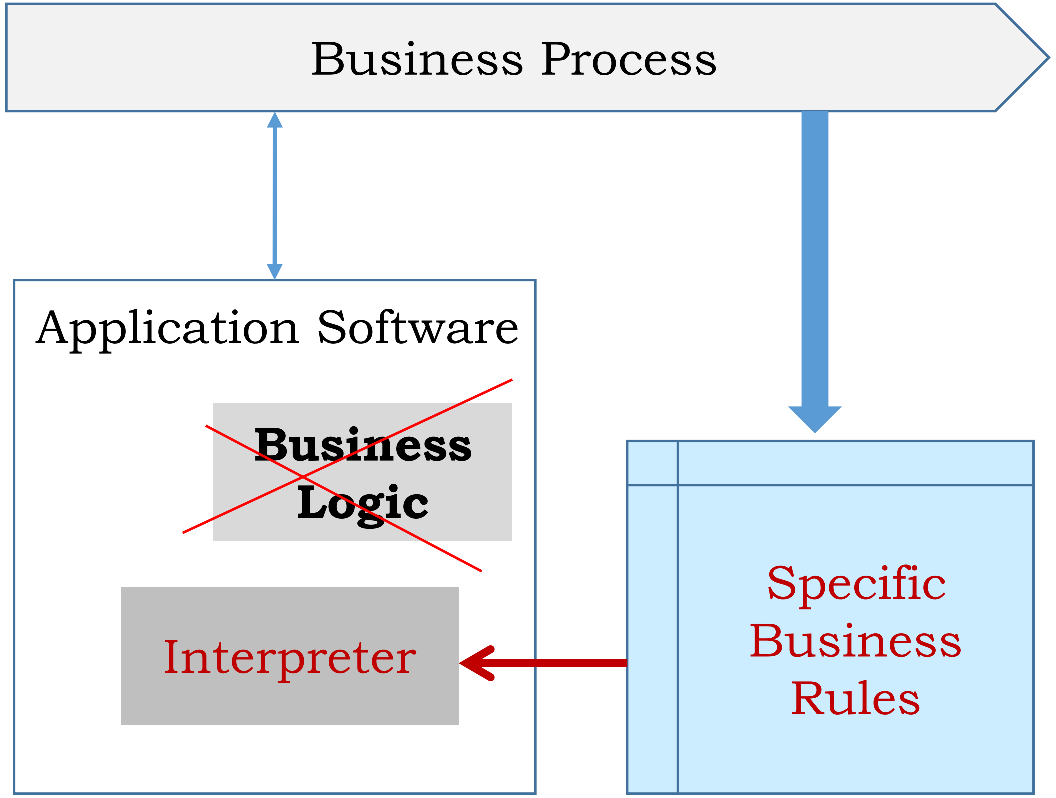
Payment Application

Clearing System

Accounts DB

Business Rules Example: Rental car servicing

http://troopscout.com



Verbal Expression:

"A car with accumulated mileage greater than 5'000 since its last service must be scheduled for service"

Formal Expression:

```
If Car.miles-current-
period > 5000 then
invoke Schedule-
service (Car.id)
End if
```

Measuring the Reuse-Factor:

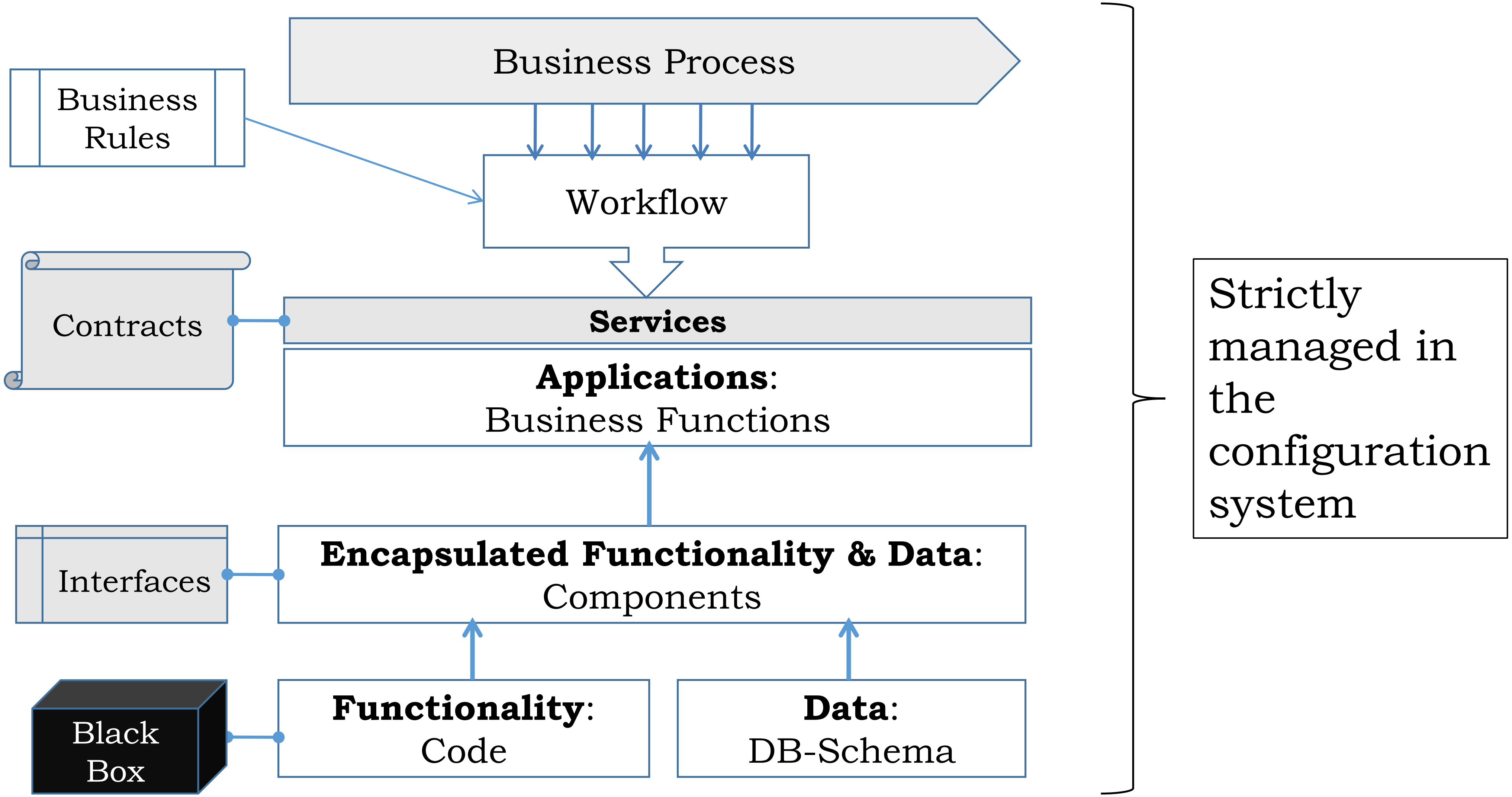
% of reused business rules in a different context

# of applications using the service

# of calls/hour

# of applications using the component

# of components implementing the code/DB fragment



Strictly managed in the configuration system

Why should we work with Reuse?

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

Because of:

- The **benefits** (in development cost and time-to-market) are considerable
- The **quality** of the software is higher (mature components, managed evolution and maintenance)
- Use of proven **3rd party** components and services
- Optimization: reusable components ⇔ one-time components

Which are the risks of reuse?

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

Risks:

- Quality of reusable software not sufficient
- Reuse-factor too low
- Reuse-strategy not complete or adequate
- Creation of unmanged redundancy (both functional and data)
- Development and maintenance process more complicated
- Management not sufficiently supportive of reuse-strategy

A8

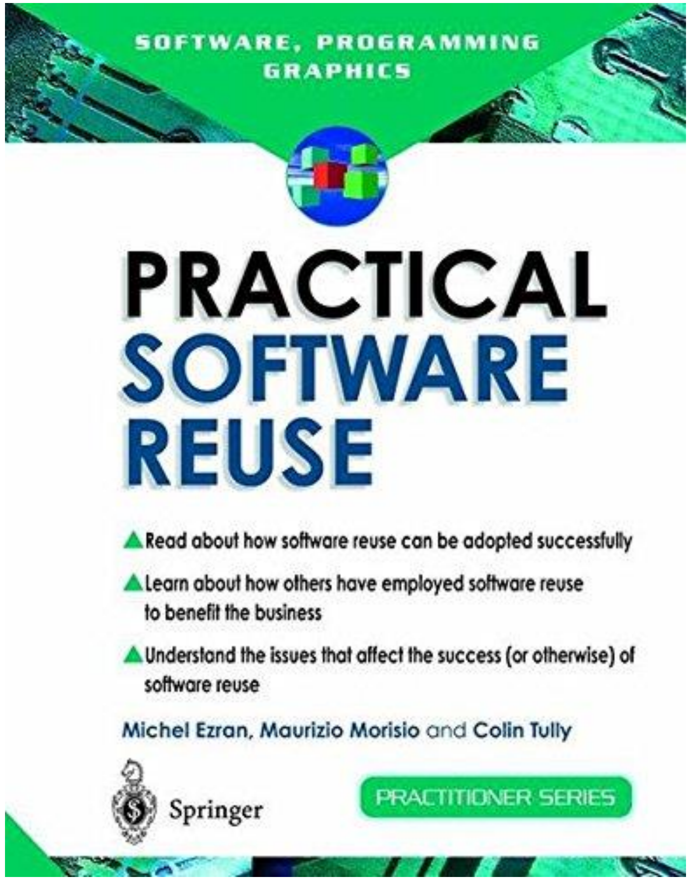
Architecture Principle A8:

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.

Textbook



SOFTWARE, PROGRAMMING GRAPHICS

PRACTICAL SOFTWARE REUSE

- Read about how software reuse can be adopted successfully
- Learn about how others have employed software reuse to benefit the business
- Understand the issues that affect the success (or otherwise) of software reuse

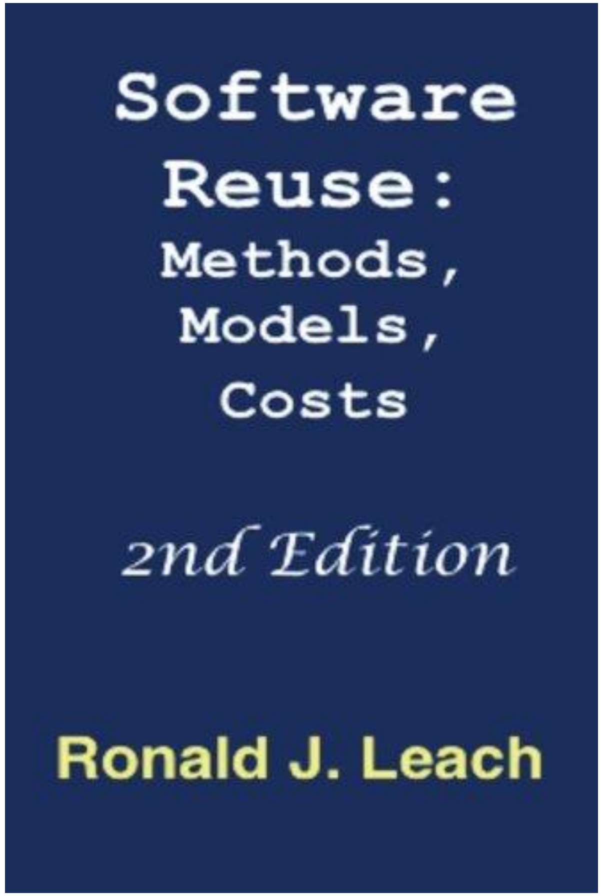
Michel Ezran, Maurizio Morisio and Colin Tully

Springer

PRACTITIONER SERIES

Michel Ezran:
Practical Software Reuse
Springer-Verlag, 2013 (reprint of 2002 edition).
ISBN 978-1-852-33502-1

Textbook



Software Reuse: Methods, Models, Costs

2nd Edition

Ronald J. Leach

Ronald J. Leach:
Software Reuse – *Methods, Models, Costs*
Ronald J Leach Publishing, 2nd edition, 2013.)
ISBN 978-1-9391-4235-1

Horizontal Architecture Layer Principles:

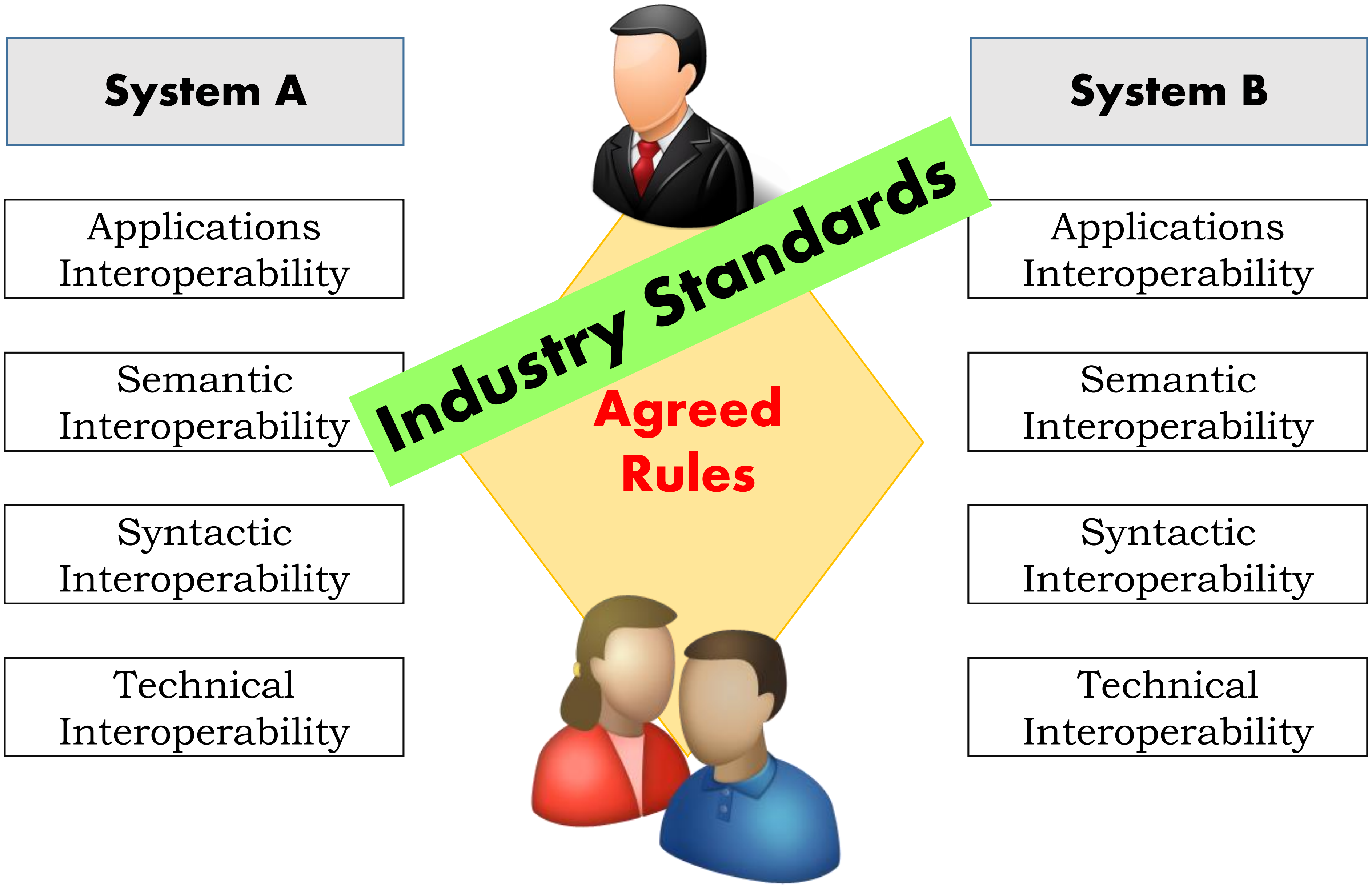
- A1: Architecture Layer Isolation
- A2: Partitioning, Encapsulation and Coupling
- A3: Conceptual Integrity
- A4: Redundancy
- A5: Interoperability
- A6: Common Functions
- A7: Reference Architectures, Frameworks and Patterns
- A8: Reuse and Parametrization
- A9: Industry Standards
- A10: Information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

A9

Architecture Principle A9:

Industry Standards

Interoperability Requirements



System A

Applications Interoperability

Semantic Interoperability

Syntactic Interoperability

Technical Interoperability

System B

Applications Interoperability

Semantic Interoperability

Syntactic Interoperability

Technical Interoperability

DEFINITIONS

A ***standard*** is:

- a formal, established norm for (technical) systems
- a document which establishes uniform (engineering or technical) criteria, principles, methods, processes and practices

ISO International Organization for Standardization
www.iso.org

IETF
www.ietf.org

OMG OBJECT MANAGEMENT GROUP
www.omg.org

International Standards Organizations

CREDIT SUISSE
CS
Business
Object
Model
V1.1/2009

Company
Standards

Why being constrained and restricted by industry standards?

- Slow
- Overkilled
- Behind technology
- …

https://grid.gograph.com

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

Example: Napoleonic Guns

(1/3)

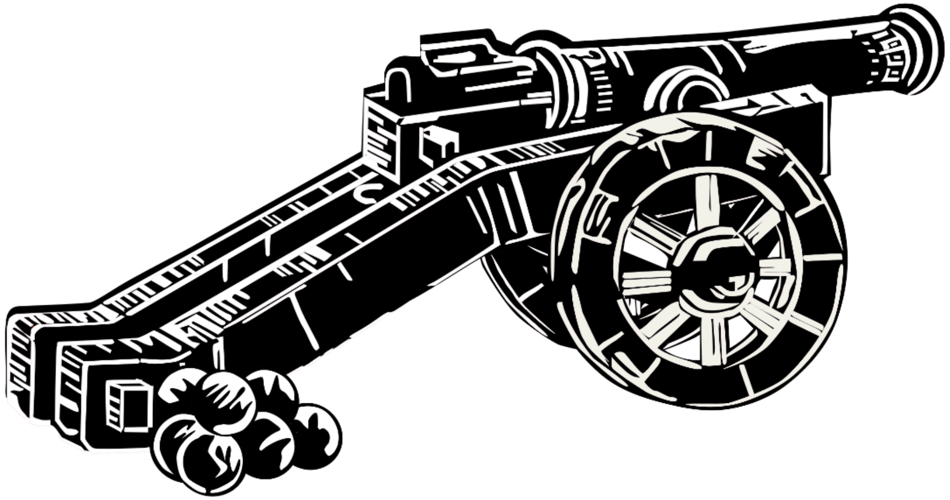
http://civilwartalk.com/threads/largest-reenactor-canon.79423

In early pre-Napoleonic times the artillery cannons were *individually different* and required matched cannon balls → *difficult logistics*

Manufacturing *tolerances* greatly reduced the accuracy and firing power of the artillery cannons → *reduced military impact*

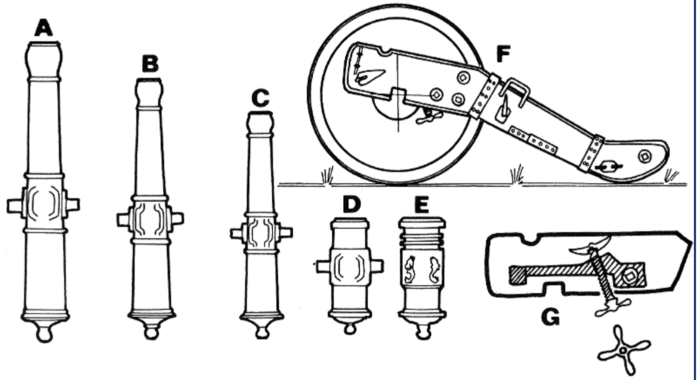
Example: Napoleonic Guns
(2/3)

1715-1789



https://openclipart.org

1776: The **de Gribeauval standard** revolutionized artillery.



de Gribeauval Standard:

- reduced and standardized the calibers
  → *complexity reduction*
- introduced normalized parts for the cannons
  → *component technology*
- set manufacturing processes & tolerances
  → *reuse*

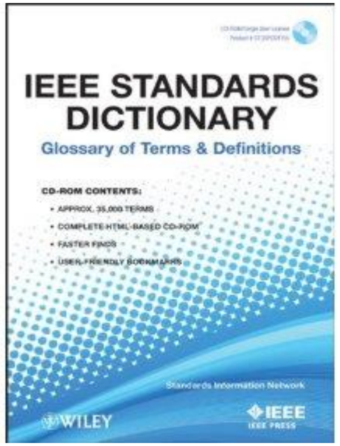
Example: Napoleonic Guns (3/3)

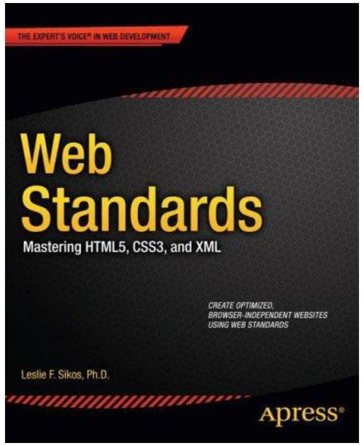
The power of standards

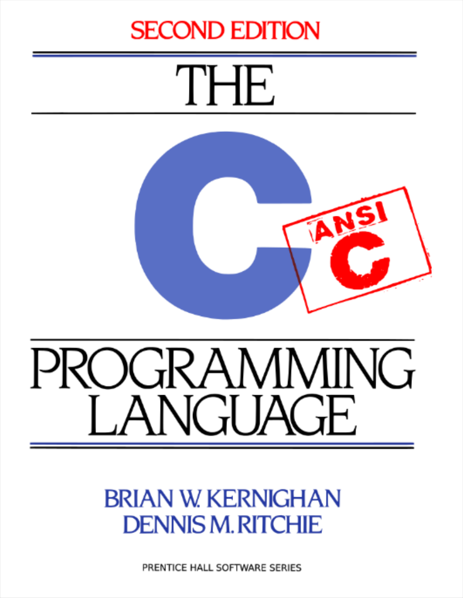
Napoleonic Empire (ca. 1810)

http://www.sandrocastelli.com/works_paginas/napoleonsempire.htm

… standards for interoperability

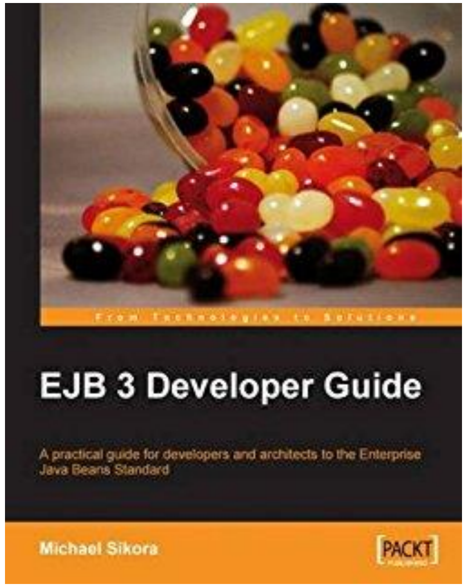






… standards for programming languages

… standards for processes



➢ What is the impact of standards ?

➢ Why are standards important ?

Impact:

- Forcing uniform, interoperable solutions in the industry
- Providing proven, widely accepted and mature solutions
- Enabling exchangeable products (mostly)
- Facilitates reuse
- Foundation for validation & certification

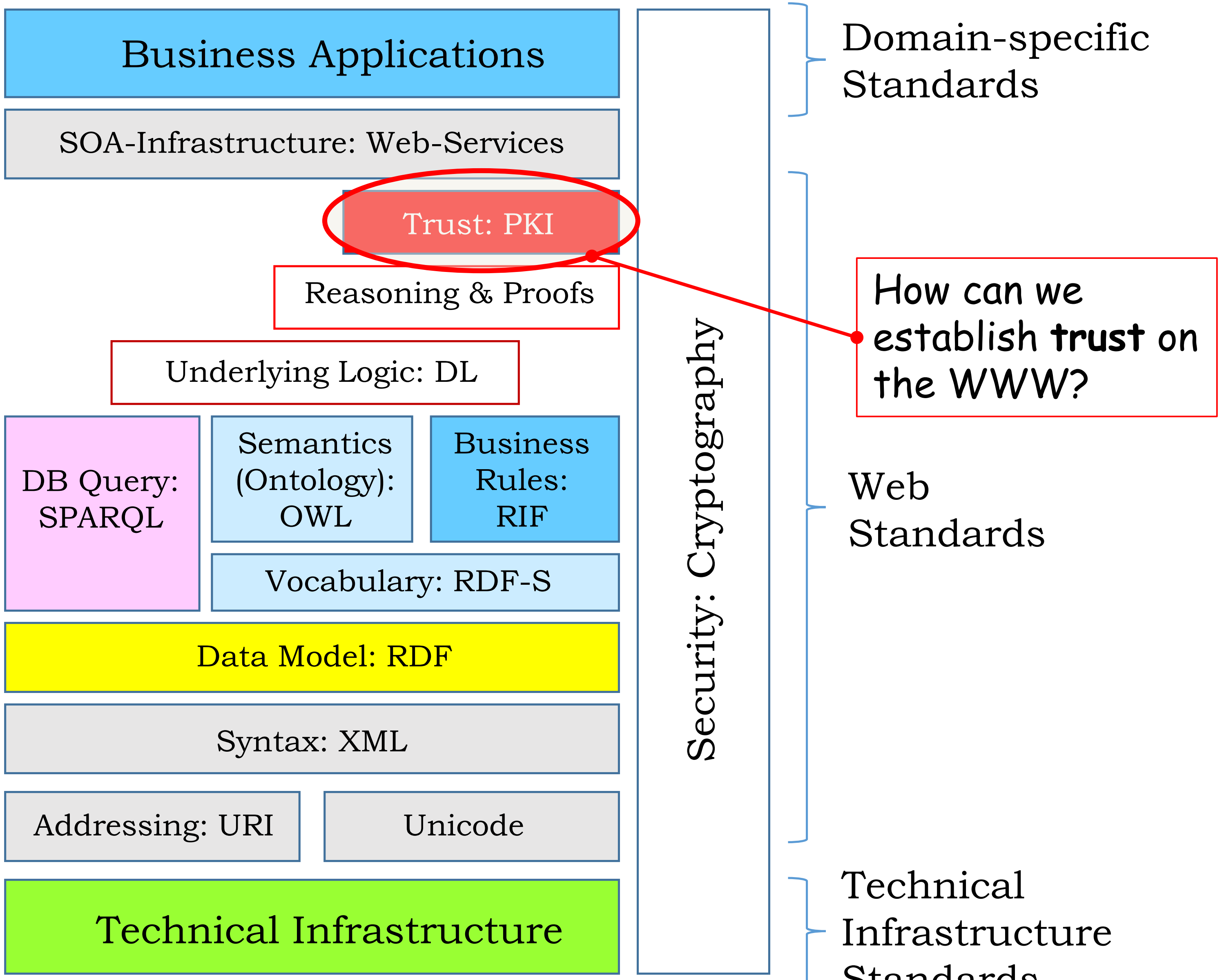
Importance:

- Provides long term stability with managed change
- Forces vendors to comply to interoperable solutions
- Advances industries as a whole
- Provides confidence in technical solutions (e.g. safety or security)

Negative: Standards-setting process is quite slow (Wide consensus required)

Future-Proof Software-Systems [Part 3B]

Example: Web Standards (1/3)



Business Applications

SOA-Infrastructure: Web-Services

Trust: PKI

Reasoning & Proofs

Underlying Logic: DL

DB Query: SPARQL

Semantics (Ontology): OWL

Business Rules: RIF

Vocabulary: RDF-S

Data Model: RDF

Syntax: XML

Addressing: URI

Unicode

Technical Infrastructure

Security: Cryptography

Domain-specific Standards

Web Standards

Technical Infrastructure Standards

How can we establish **trust** on the WWW?

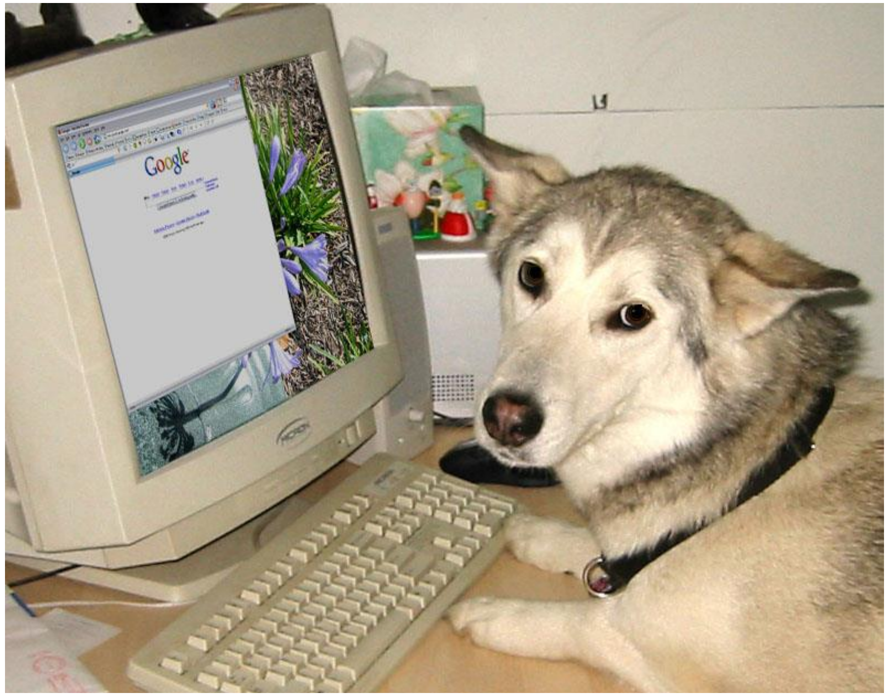
Future-Proof Software-Systems [Part 3B]

Example: Web Standards (2/3)

Trust: PKI

Example: Authentication:

How can we establish trust in the **identity** of an electronic partner ?



On the Internet, nobody knows you're a dog

https://encyclopediadramatica.se/On_the_Internet,_nobody_knows_you're_a_dog

Answer:

Use a Public Key Infrastructure (**PKI**)

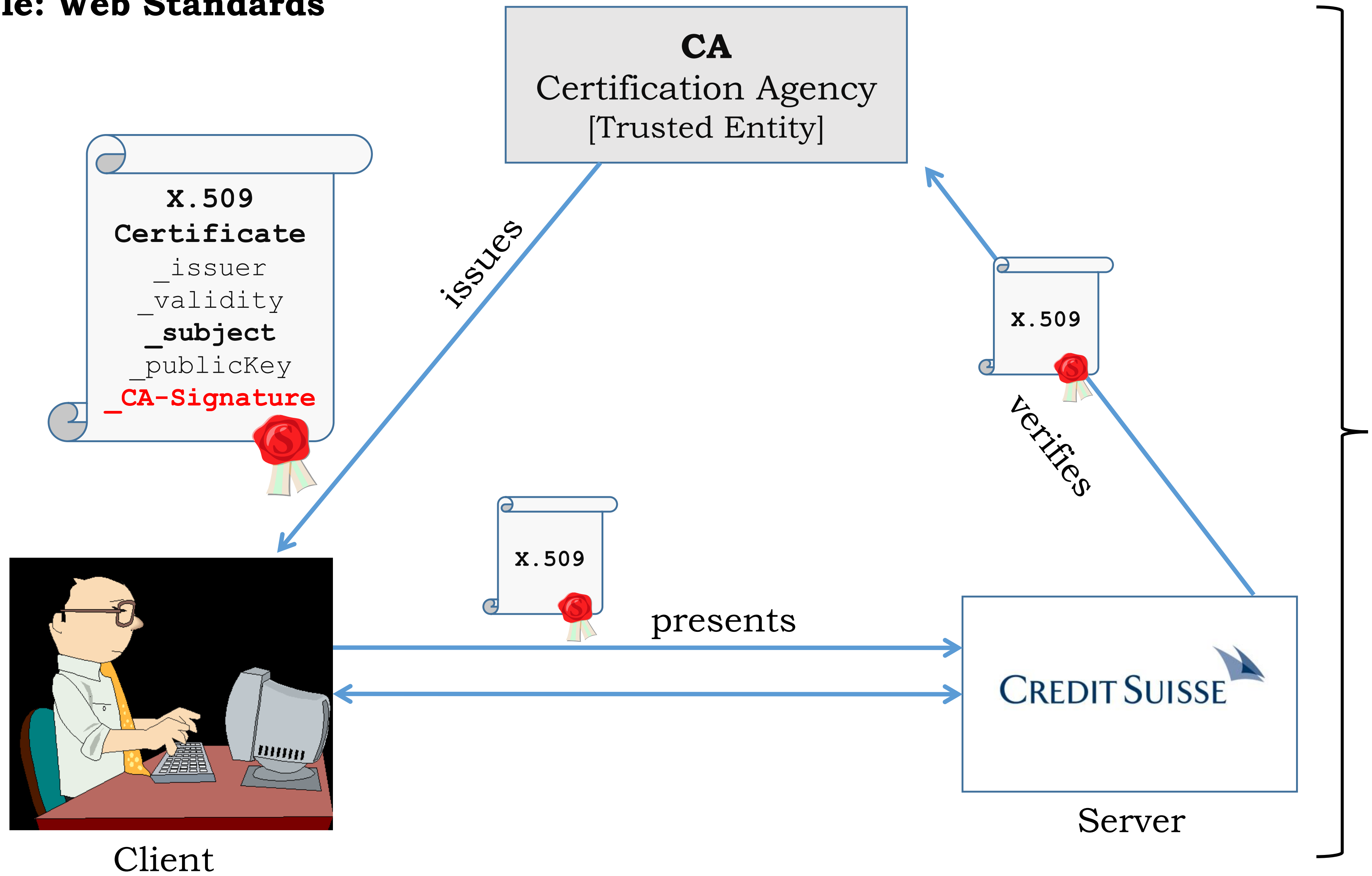
PKI assigns **Digital Certificates** to entities (Persons, organizations)

A digital certificate is an unforgeable electronic **proof of identity**

Digital certificates are **standardized in X.509** and are globally accepted and used

⇒ Global interoperability

Example: Web Standards
(3/3)



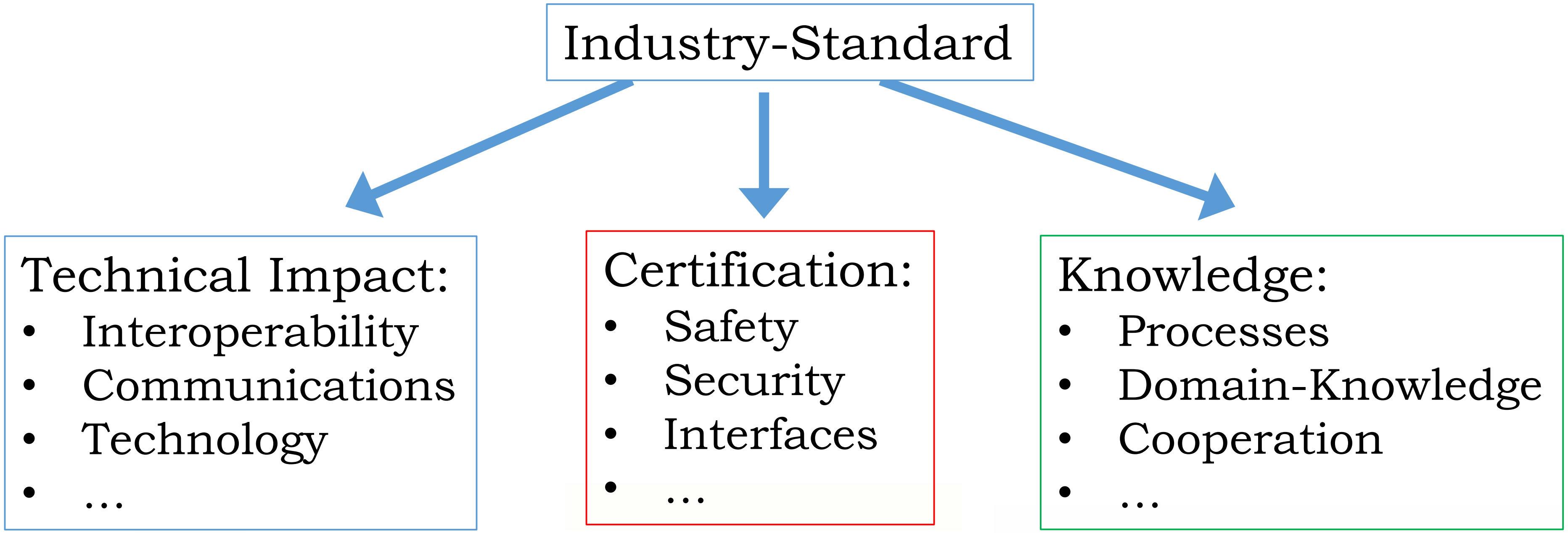
CA
Certification Agency
[Trusted Entity]

X.509
Certificate
`_issuer`
`_validity`
`_subject`
`_publicKey`
`_CA-Signature`

Trustworthy electronic authentication procedure

Future-Proof Software-Systems [Part 3B]

International Organization for Standardization

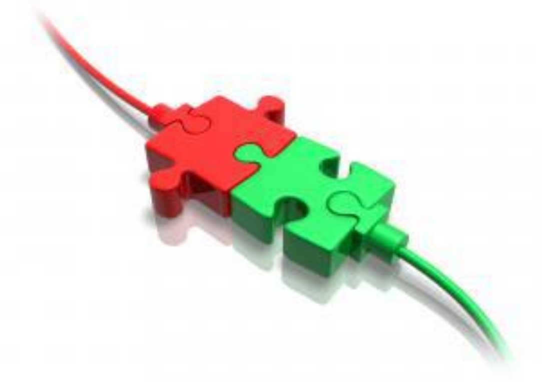


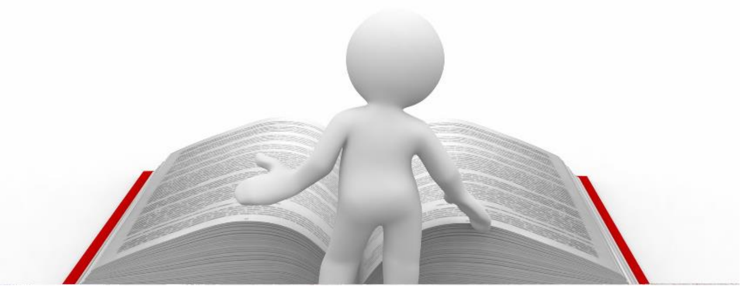
Industry-Standard

Technical Impact:
- Interoperability
- Communications
- Technology
- …

Certification:
- Safety
- Security
- Interfaces
- …

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





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

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

A10

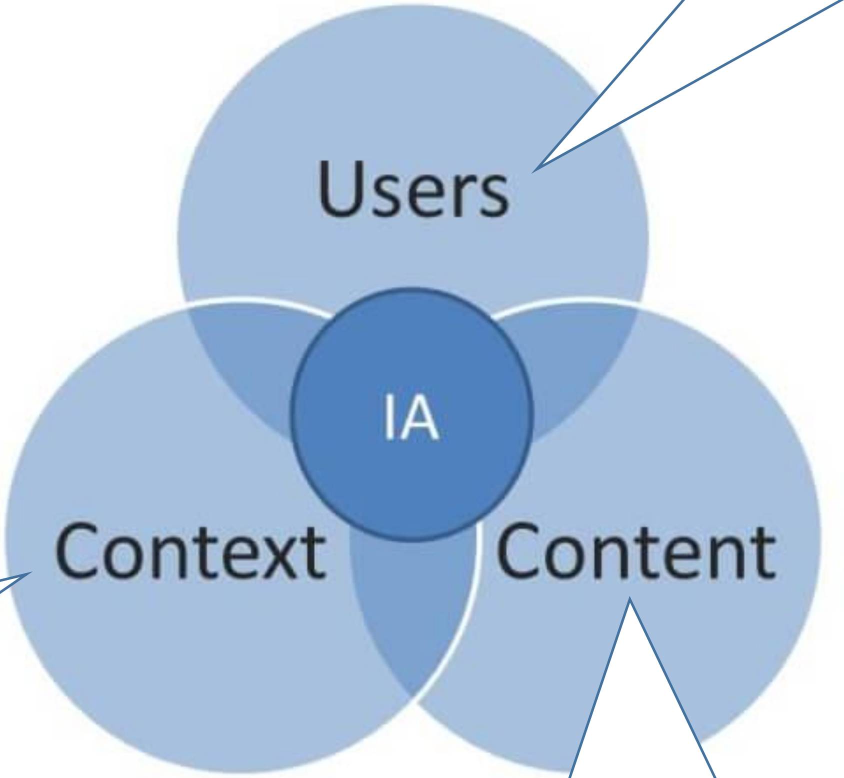
Architecture Principle A10:

Information Architecture

Information Architecture

„Data models are perhaps the most important part of developing software, because they have such a profound effect: Not only on how the software is written, but also on *how we think about the problem* that we are solving"

Martin Kleppmann, 2017



Humans, machines, robots, artificial intelligence, …

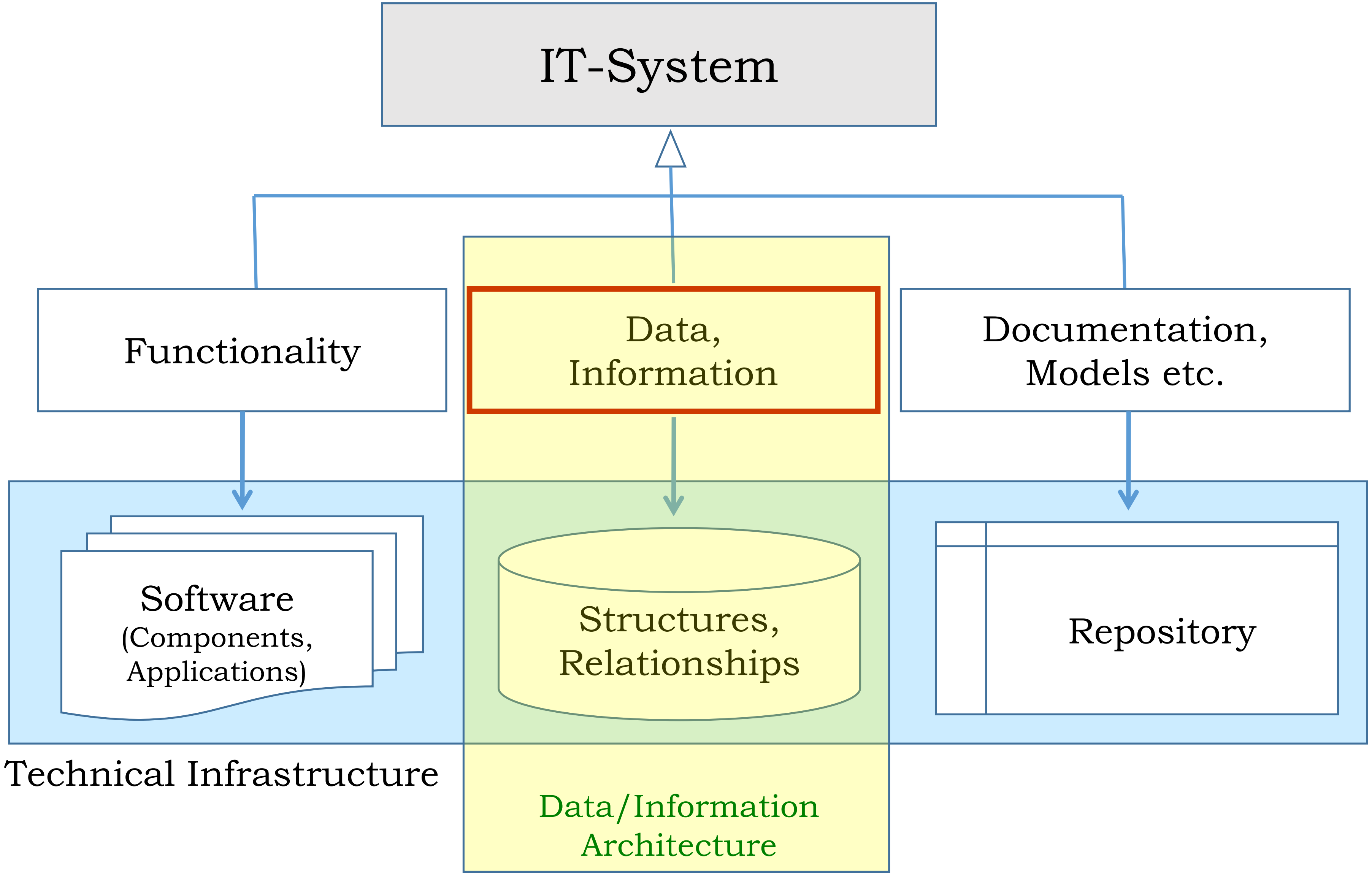
Static, dynamic, stable, unstable, uncertain, …

Business, people, autonomic, safety-critical, …

Big data, real-time, confidential, fuzzy, experimental, long-lived, …

http://www.antlr-interactive.com

Future-Proof Software-Systems [Part 3B]



Future-Proof Software-Systems [Part 3B]

http://www.dmu.ac.uk

DEFINITIONS

Information = Data that is

1. accurate and timely,
2. specific and organized for a purpose,
3. presented within a context that gives it meaning and relevance,
4. leads to an increase in understanding and decrease in uncertainty

http://www.businessdictionary.com

Information Architecture

Structural Architecture Layers
Horizontal Architectures

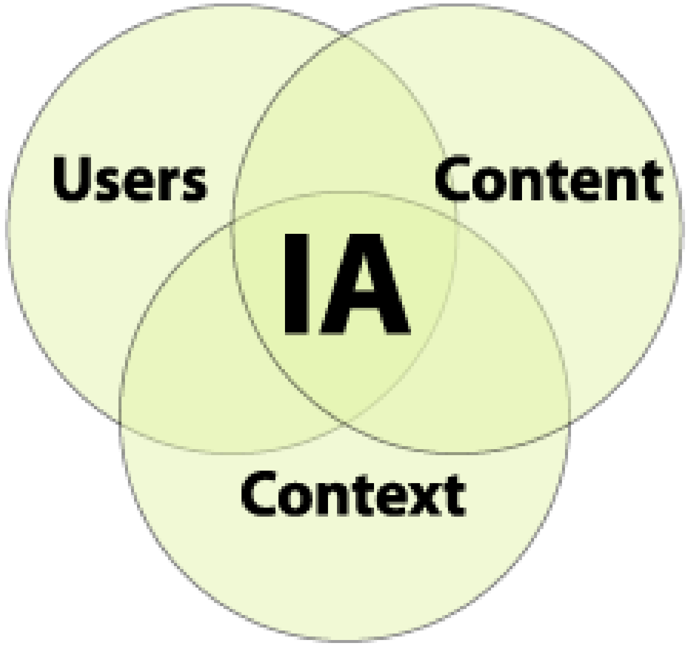
Business Architecture (Business Processes)

Applications Architecture (Functionality)

Information (Data) Architecture (Information & Data)

Integration Architecture (Cooperation Mechanisms)

Technical Architecture (Technical Infrastructure)



http://www.rockley.com

DEFINITIONS

Data/Information Architecture

Definition (1/2):

Information Architecture is a ***engineering discipline*** and a (resulting) ***structure*** that is focused on making information:

- dependable
  - correct (content- & time-wise)
  - complete
  - consistent & integer
  - protected
  - accountable
- understandable
  - semantics
  - structured
- findable
  - organized
  - available
  - unique (no unmanaged redundancy)

DEFINITIONS

Data/Information Architecture

Definition (2/2):

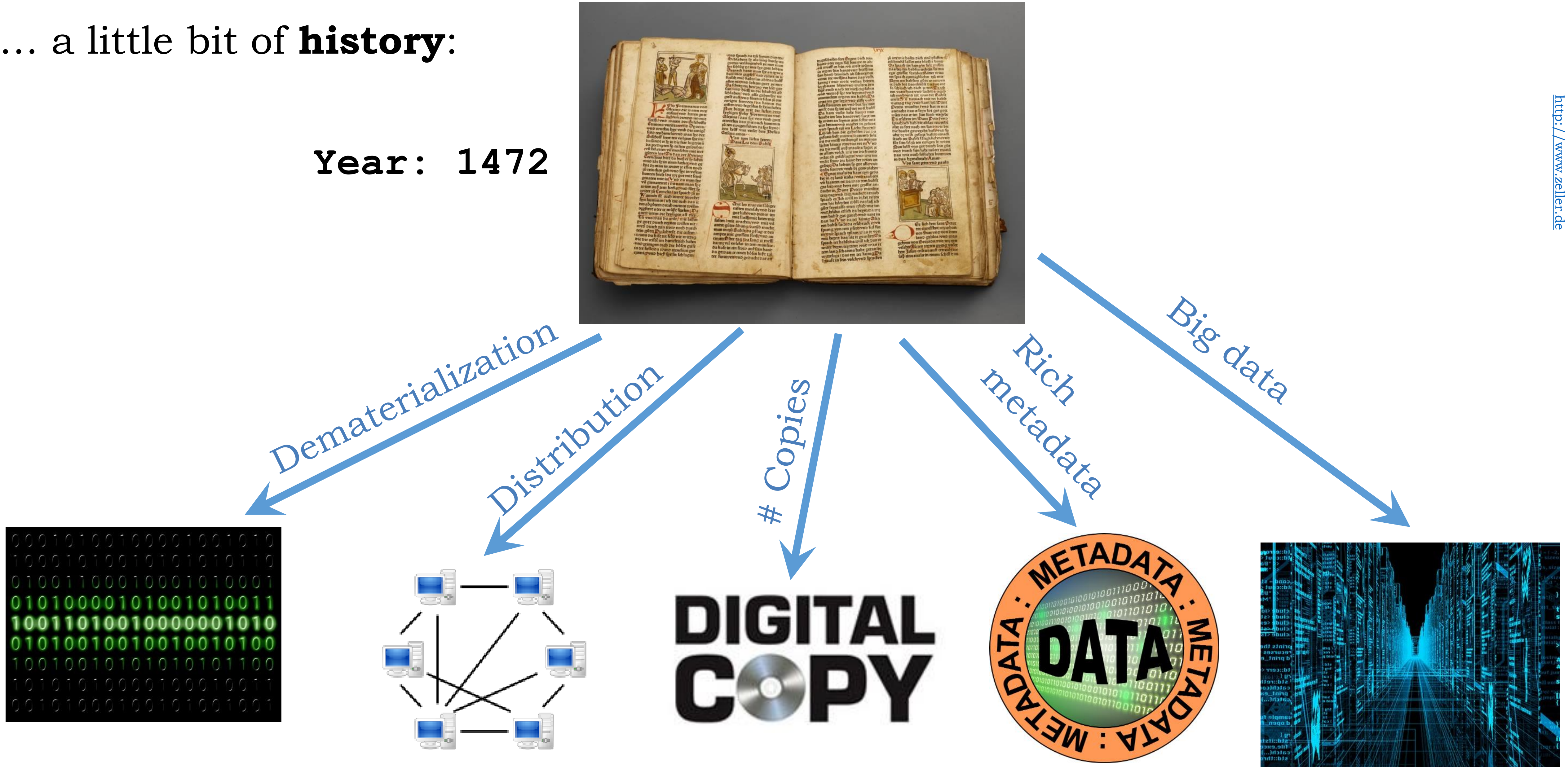
The Data/Information Architecture defines **principles** for:

- The *classification* of data/information
- The *structure* of data/information
- The *modeling* of data/information
- The *quality assurance* of data/information
- The *protection* of data/information
- The *deployment* of data/information
- The *disaster recovery* of data/information
- [The process for building and maintaining the architecture]

http://www.ub.tum.de

… a little bit of **history**:

Year: 1472



Dematerialization

Distribution

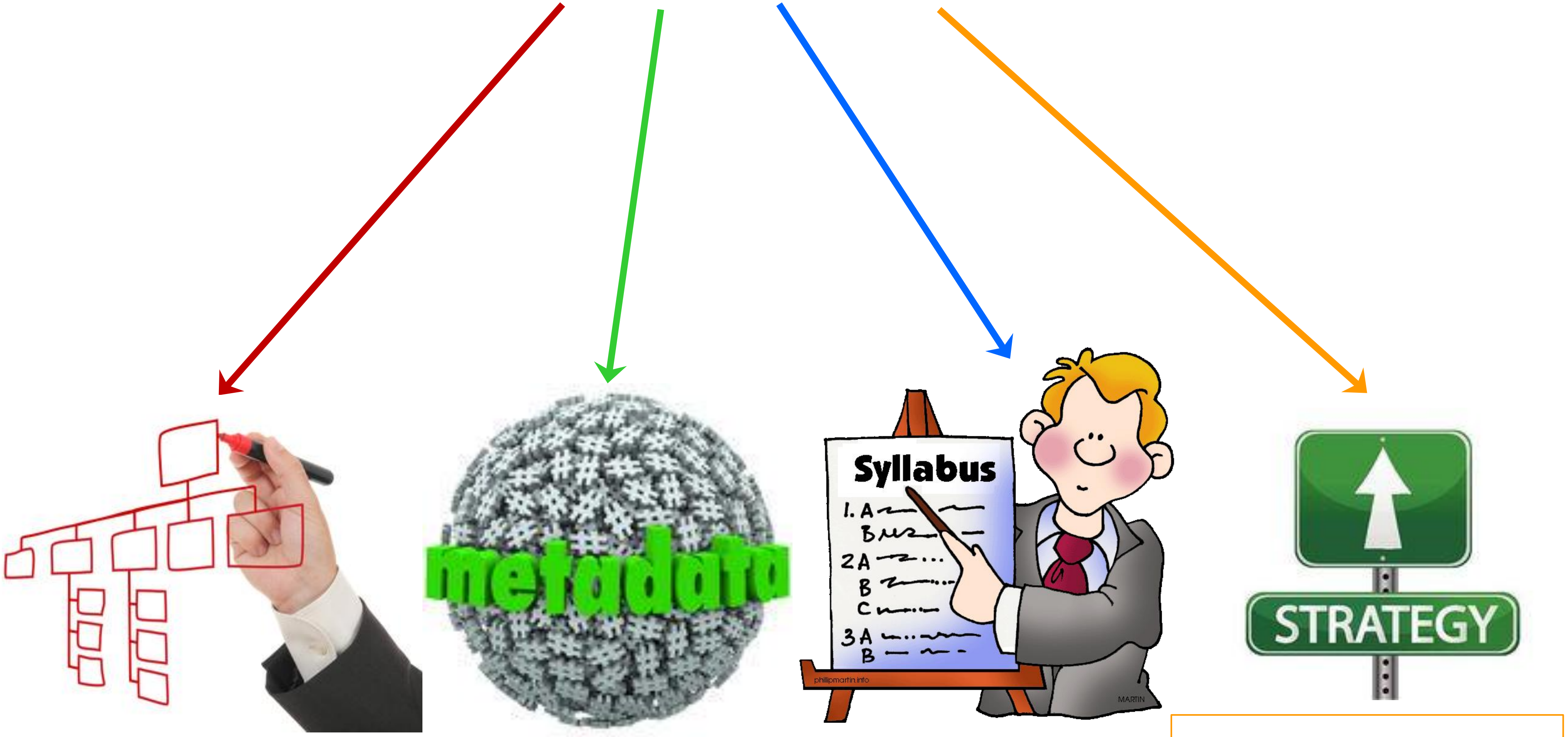
Copies

Rich metadata

Big data

http://www.zeller.de

Information Architecture Artefacts



Structure

Metadata

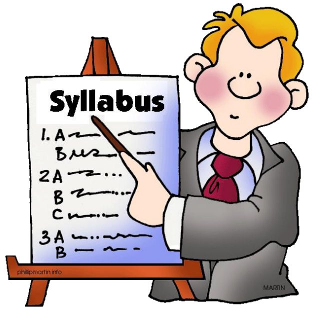
Semantics

Information Strategy

Future-Proof Software-Systems [Part 3B]

Structure	The logical organization of the information universe of a company

Metadata	Metadata is data providing information about aspects of the data (source, purpose, content, …)



Semantics	Definition and representation of meaning of the information

Information Strategy	Objectives, principles and processes for the information architecture

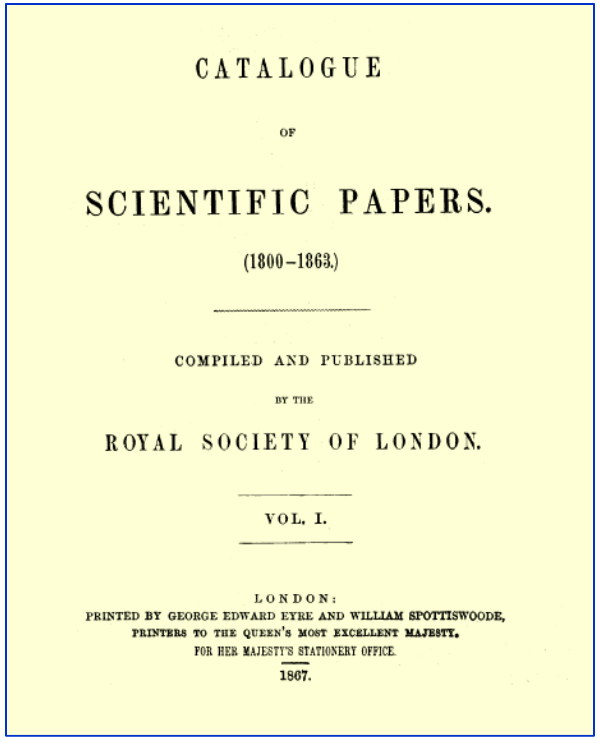
Example: Metadata for Publishing

Standardized [ACM]

Machine-readable (XML)

Complete

Semantics: Keywords [ACM Dictionary]



CATALOGUE
OF
SCIENTIFIC PAPERS.
(1800–1863.)

COMPILED AND PUBLISHED
BY THE
ROYAL SOCIETY OF LONDON.

VOL. I.

LONDON:
PRINTED BY GEORGE EDWARD EYRE AND WILLIAM SPOTTISWOODE,
PRINTERS TO THE QUEEN'S MOST EXCELLENT MAJESTY.
FOR HER MAJESTY'S STATIONERY OFFICE.
1867.

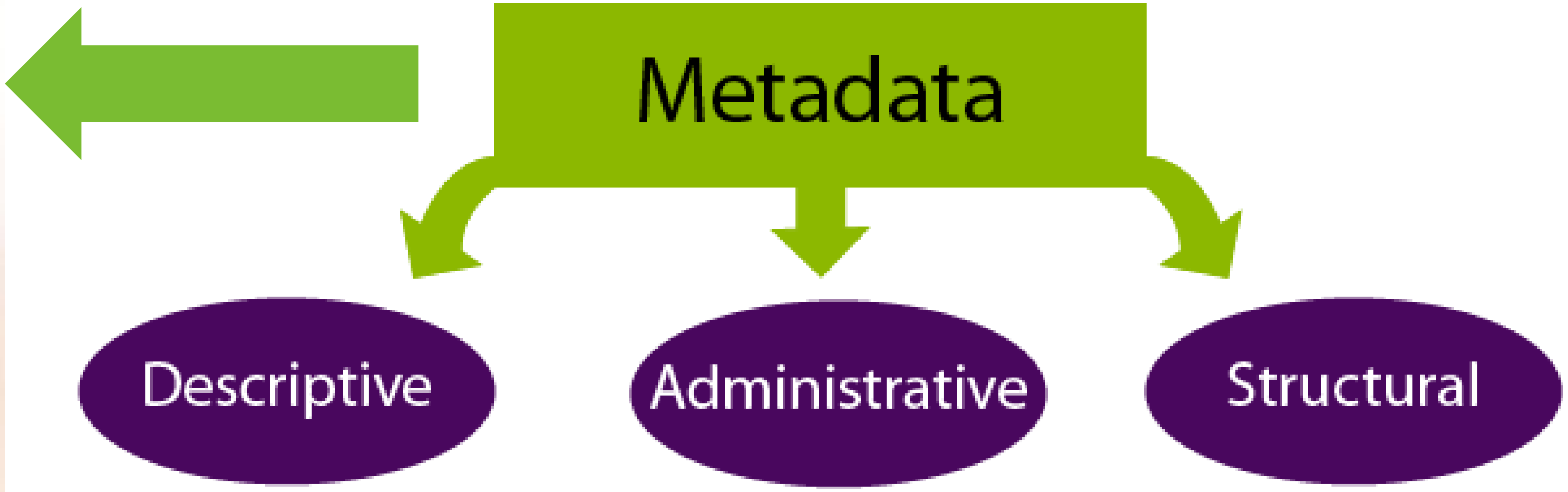
http://www.ghtc.usp.br/sources/catalogue.htm

Metadata =
Data about Data

```
<author>W. H. Jaco</author>
<title>PL minimal surfaces in $3$-manifolds</title>
<ISSN>0022-040X</ISSN>
<URL>J. Differential Goem.</URL>
<article text>The body of the article included here</article text>
... more
```

Full template:
http://www.ams.org/publications/journals/sample-data-file

http://blogs.teradata.com



- Characterization
- Tags
- Keywords
- Description
- Relations

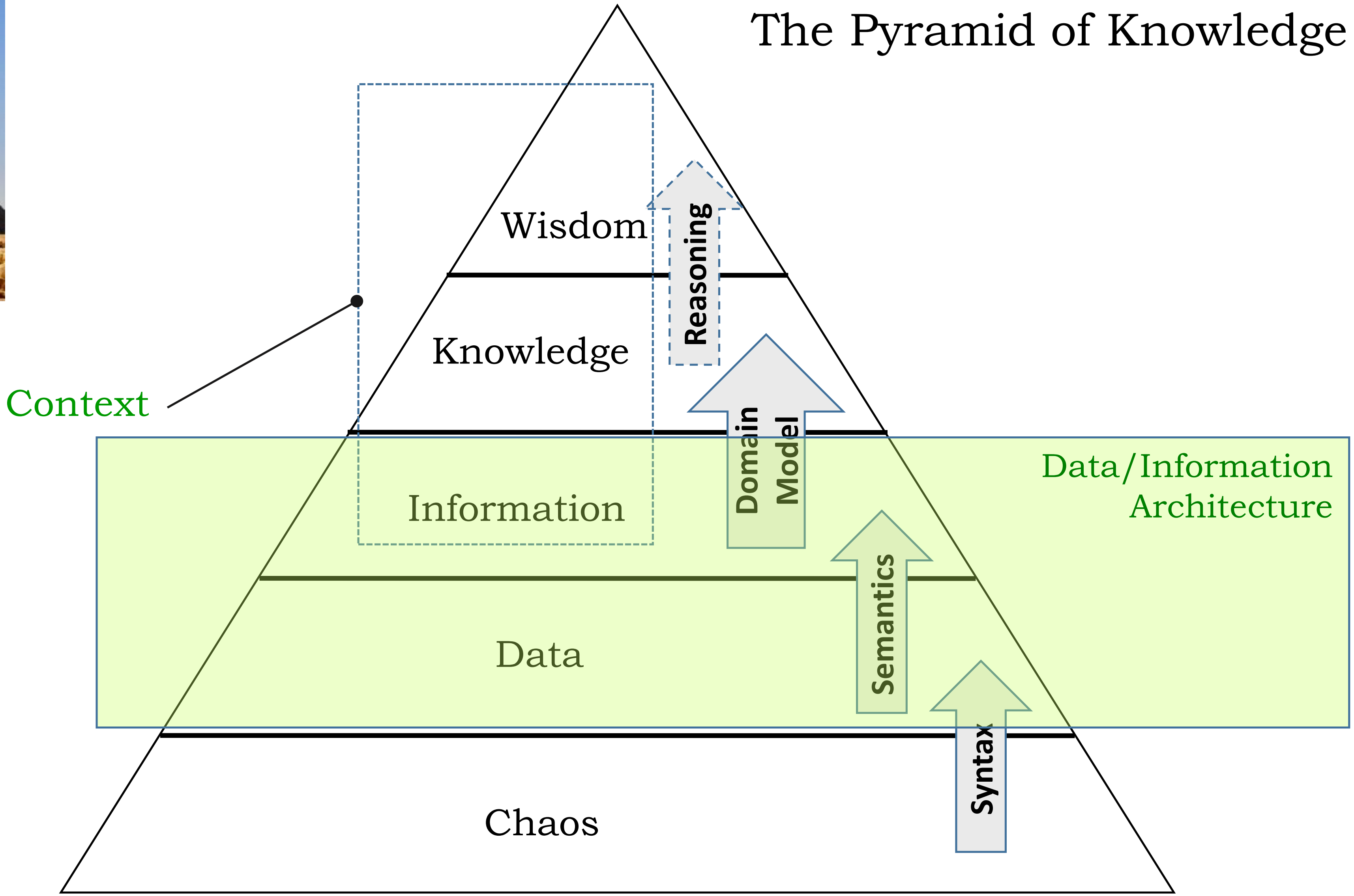
- Storage
- Access
- Rights (IPR)
- Delivery
- Rendering

- Data structure
- Hierarchies
- Composition
- Ordering
- Coherence

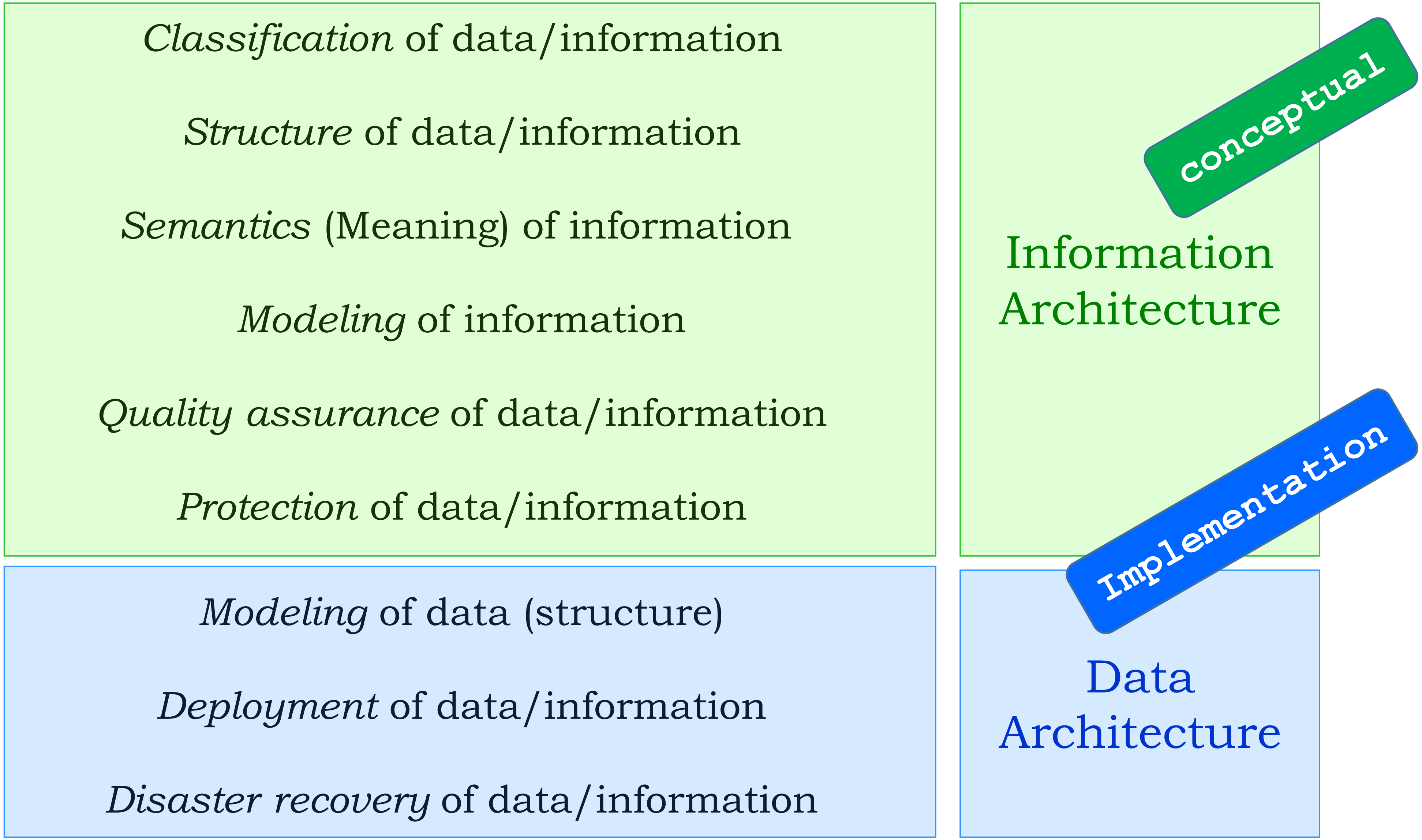
Future-Proof Software-Systems [Part 3B]

http://ancienthistory.about.com/od/pyramids/tp/91012-The-Main-Pyramids-Of-Egypt.htm

The Pyramid of Knowledge



Data & Information Architecture



Classification of data/information

Structure of data/information

Semantics (Meaning) of information

Modeling of information

Quality assurance of data/information

Protection of data/information

Information Architecture

conceptual

Modeling of data (structure)

Deployment of data/information

Disaster recovery of data/information

Data Architecture

Implementation

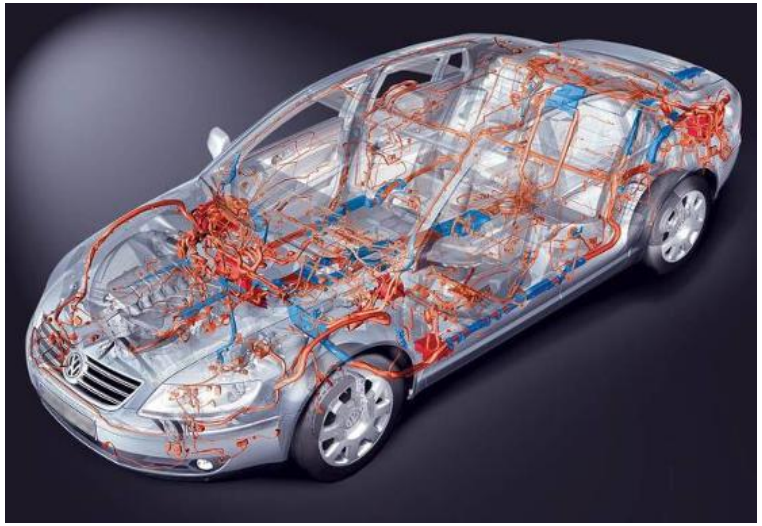
Data/Information Architecture

The principles for building applications are the **same** in all application domains ✓
[sometimes with some tradeoffs]

Q: Is this also true for information/data architecture ?

http://www.ormutual.com

Enterprise data/information architecture

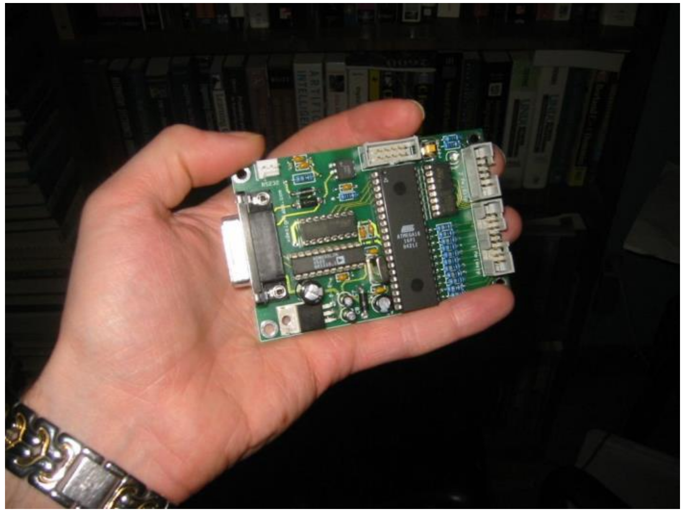


http://www.ove.uk.com

Vehicle data/information Architecture
[Embedded Systems]

… unfortunately NO!

What is *different* in embedded systems data & information?



http://thorntoncenter.net

Time !

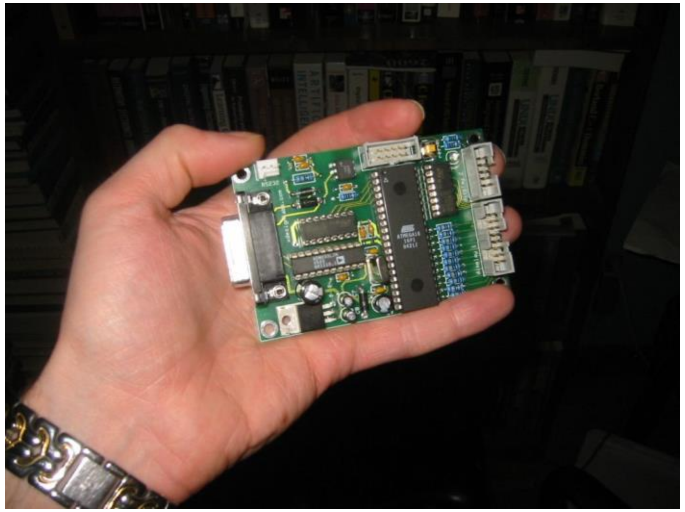
Data items have

timing relationships

between them

… sometimes very demanding and stringent!

What is *different* in embedded systems data & information?



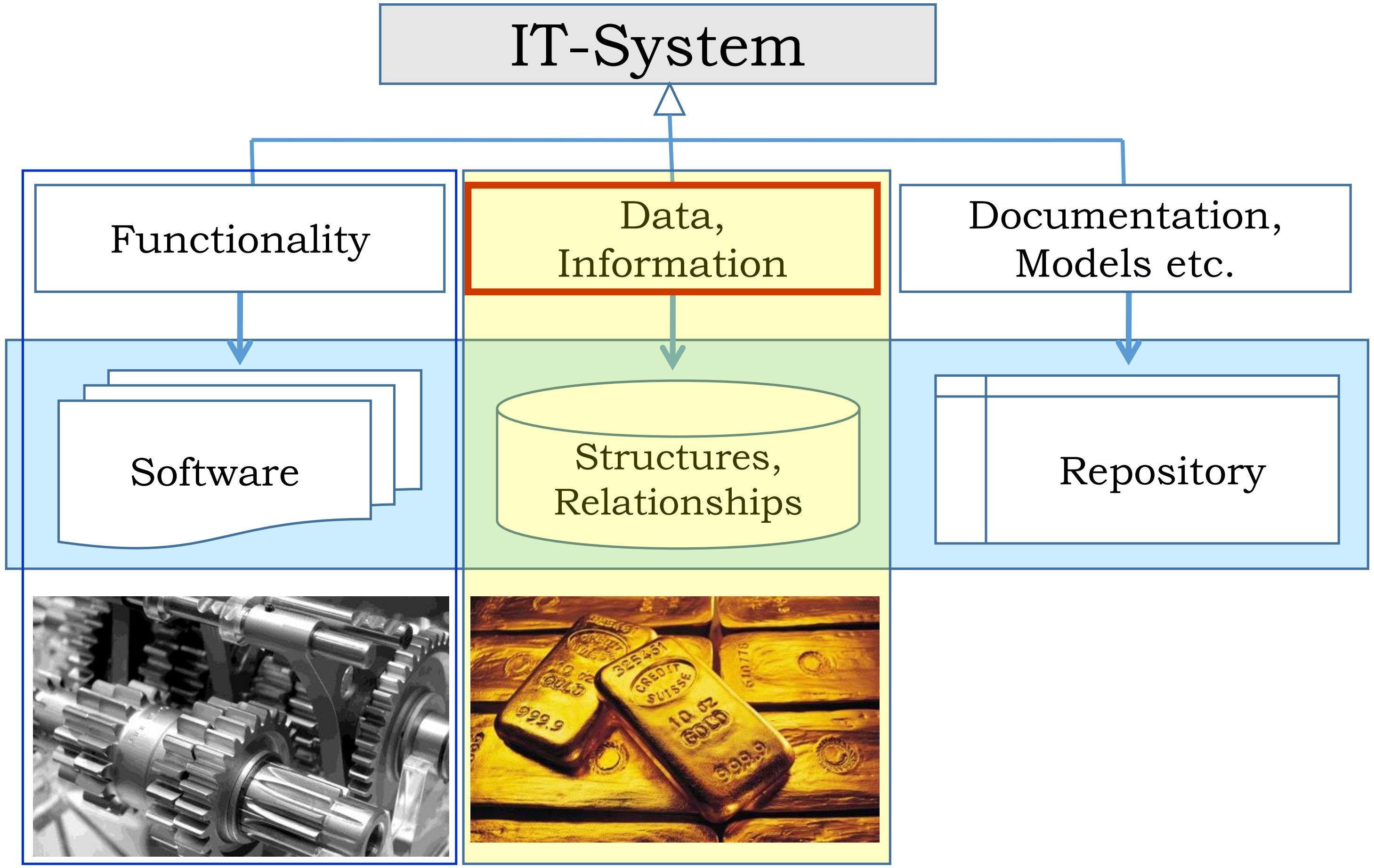
Inconsistency !

Data items may have *inconsistencies* between them

… due to mechanical, communications or electronic glitches

a) Enterprise Data/Information Architecture

http://www.ormutual.com



http://xn--80aqafcrtq.cc/de

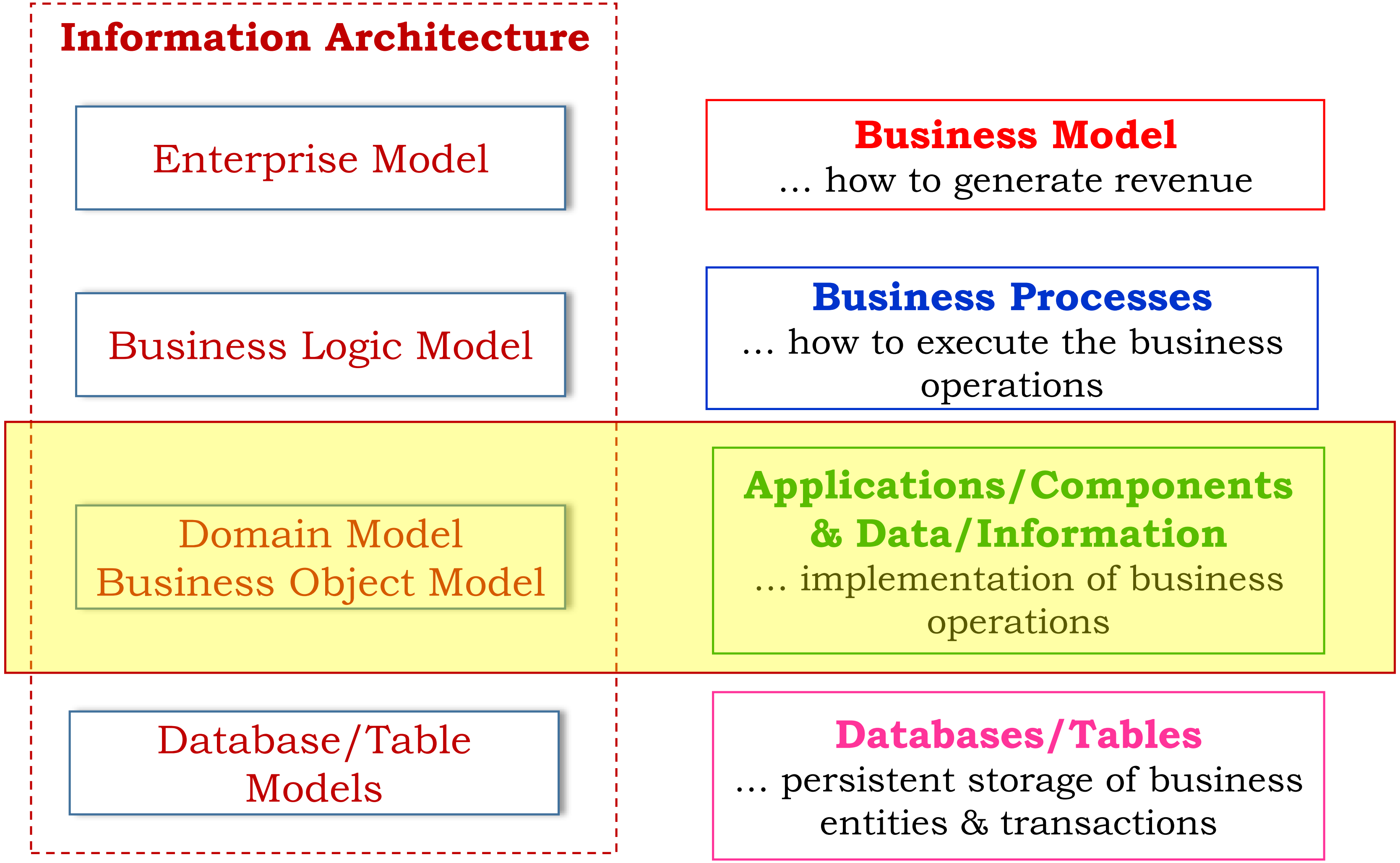
http://www.theguardian.com

Functionality:
- mostly good
- well organized

Data/Information:
- often disorganized
- inconsistent(redundant)

It is easy to change functionality – but very hard to change data/information

Data/Information Architecture Stack



Information Architecture

Enterprise Model

Business Logic Model

Domain Model
Business Object Model

Database/Table Models

Business Model
… how to generate revenue

Business Processes
… how to execute the business operations

Applications/Components & Data/Information
… implementation of business operations

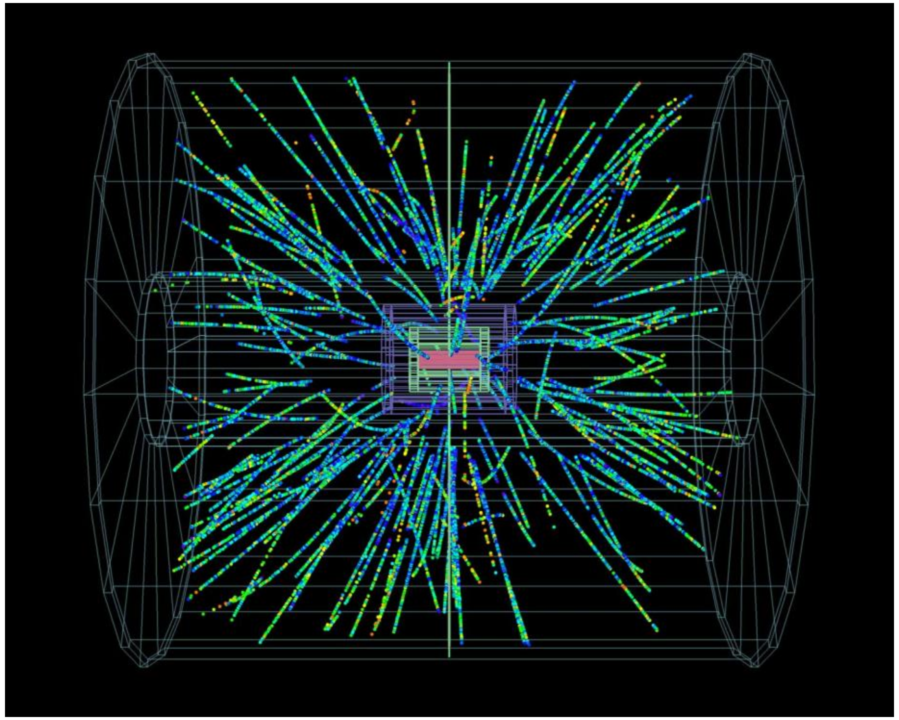
Databases/Tables
… persistent storage of business entities & transactions

Data/information architecture = A set of consistent, complete models

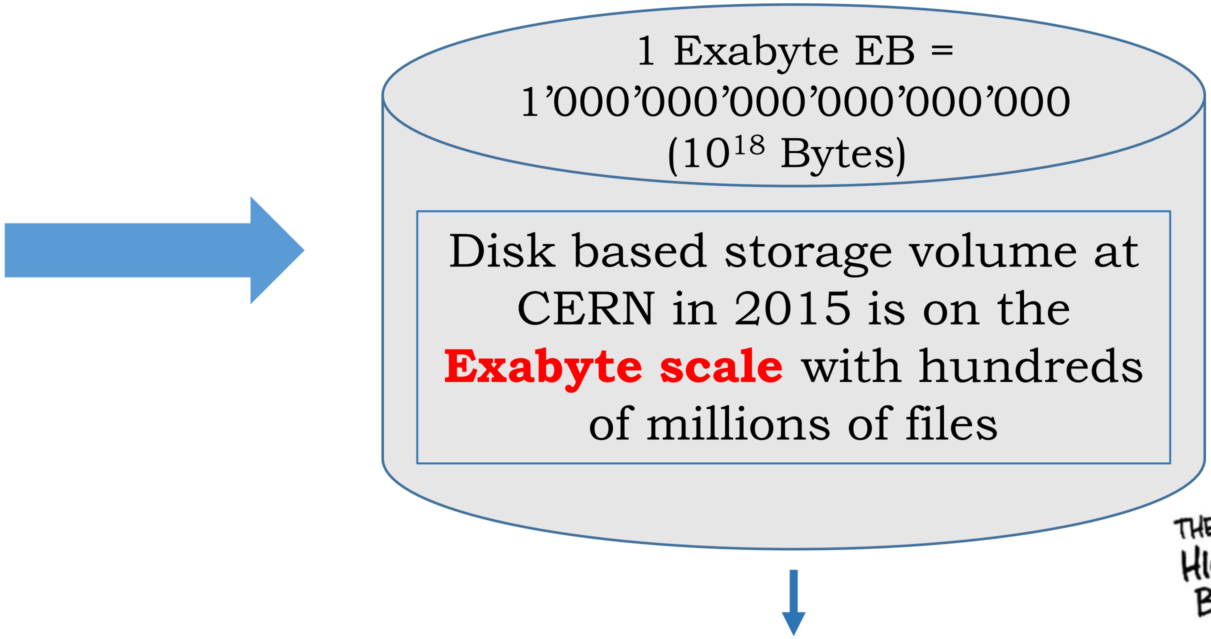
Example: Typical enterprise volumes (large bank)

Data	Criticality	Update/Access Rate	Mirroring-Interval	Save-Intervall	Remarks
Transaction Data	High	40 .. 400 Million Transactions/day	Transaction Level	24 h	Mainframe
Control Table Data & Reference Data	Very high	14’000 accesses/sec	After each update	24 h	Mainframe
Application control data	Very high	2 … 5’000 accesses/sec	After each update	24 h	Mainframe
Accounting data	Very high	50 … 100 Million Transactions/day	24 h	24 h	After EOD (= End of Day) processing
Archive	Very high	High write, very low read rate	8 hrs	daily	
Application Data	High	0 … 10 Million updates/day	After each change	daily	After EOD (= End of Day) processing

Example: CERN storage volume 2015



http://public-archive.web.cern.ch



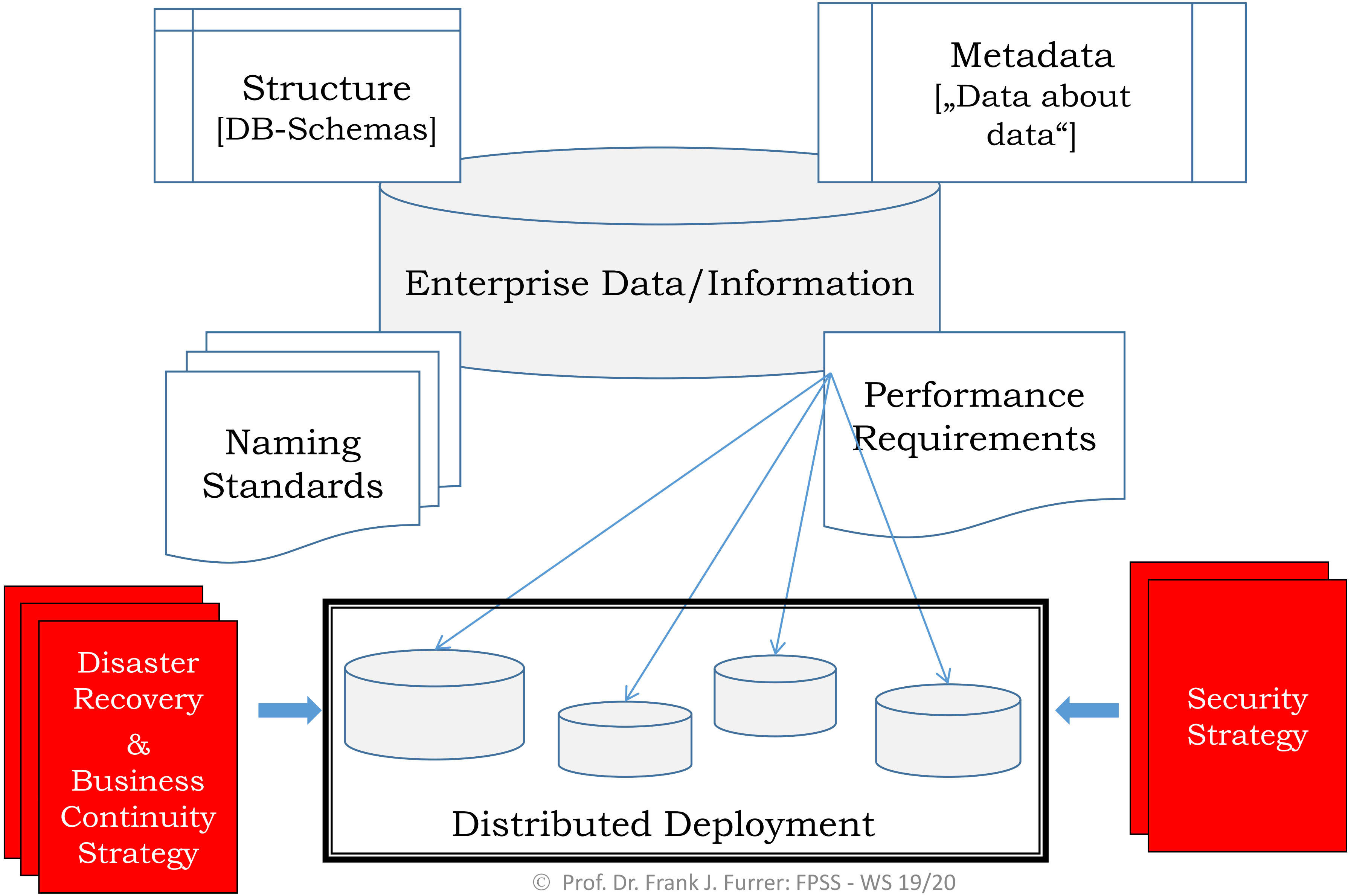
1 Exabyte EB =
1’000’000’000’000’000’000
(10^{18} Bytes)

Disk based storage volume at CERN in 2015 is on the **Exabyte scale** with hundreds of millions of files

https://i1.wp.com/www.weeklypique.com

CERN: Future ICT Challenges in Scientific Research:
Available from:
http://openlab.cern/sites/openlab.web.cern.ch/files/technical_documents/Whitepaper_brochure_ONLINE_0.pdf
[last accessed: 23.11.2017]

Data/Information Architecture **Implementation**



Structure
[DB-Schemas]

Metadata
[„Data about data“]

Enterprise Data/Information

Naming Standards

Performance Requirements

Disaster Recovery & Business Continuity Strategy

Distributed Deployment

Security Strategy

Enterprise Data/Information **Strategy**

APPROVED
by CIO & CEO

http://www.trupphr.com

No enterprise data strategy
= Chaos

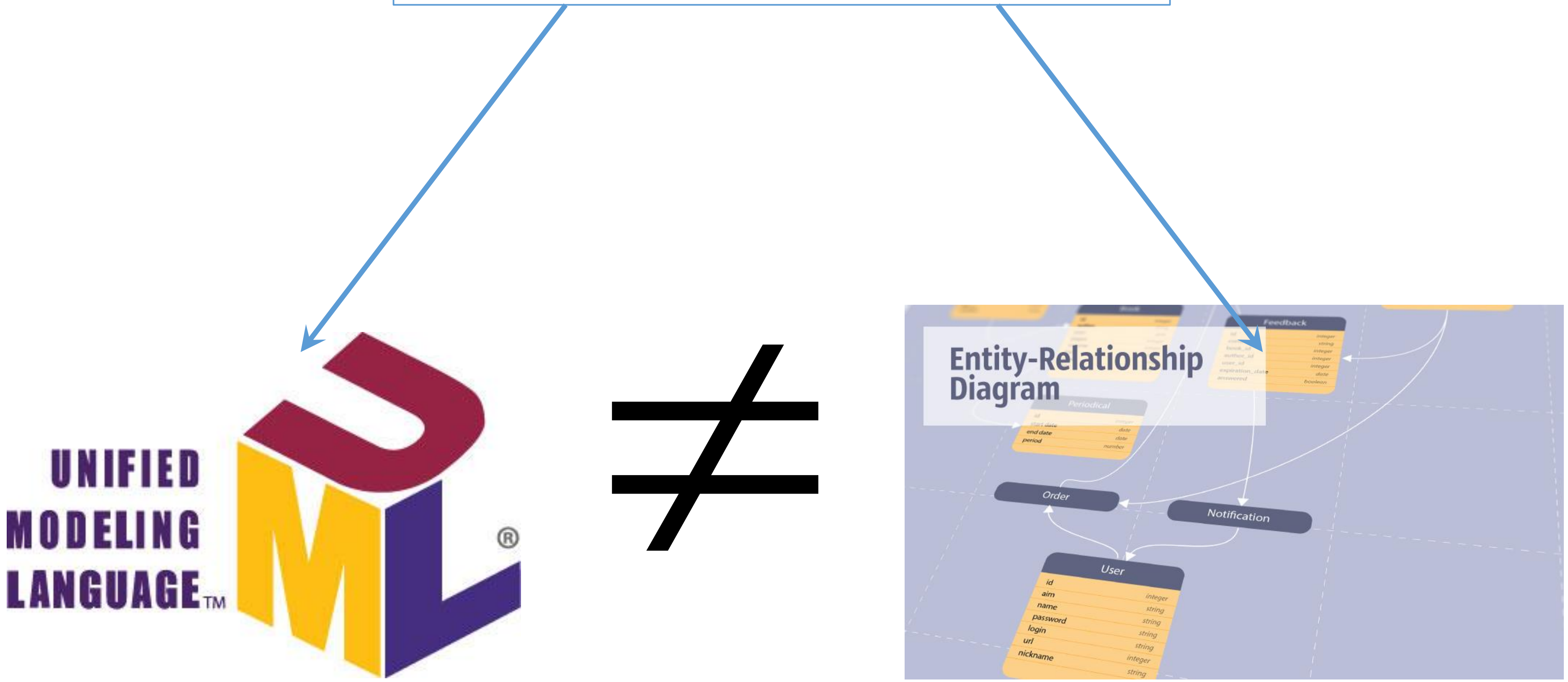
- bad data quality
- redundant data (inconsistent)
- inability to integrate
- low agility for changes
- bad performance
- …

Enterprise Data & Information Strategy

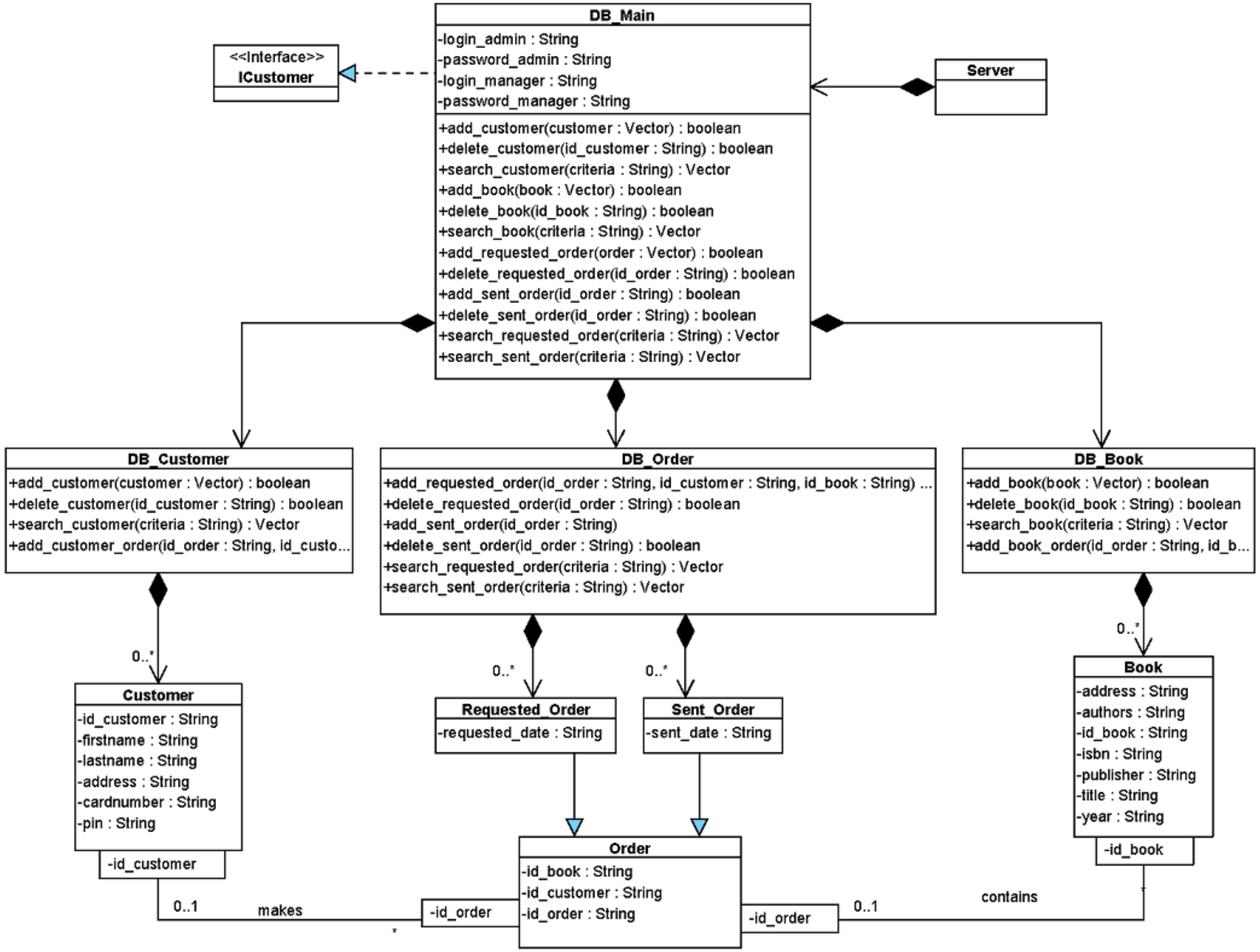
- Enterprise Context
- Data/Information Modelling
- Metadata
- Data Integration
- Data Quality Standards
- Organizational roles & responsibilities
- Performance & Measurement
- Security & Privacy
- Business Continuity & Disaster Recovery
- Legal & Compliance Requirements
- Unstructured Data

Data/information architecture = A set of consistent, complete models

Modeling Data/Information:
⇒ 2 «competing» notations

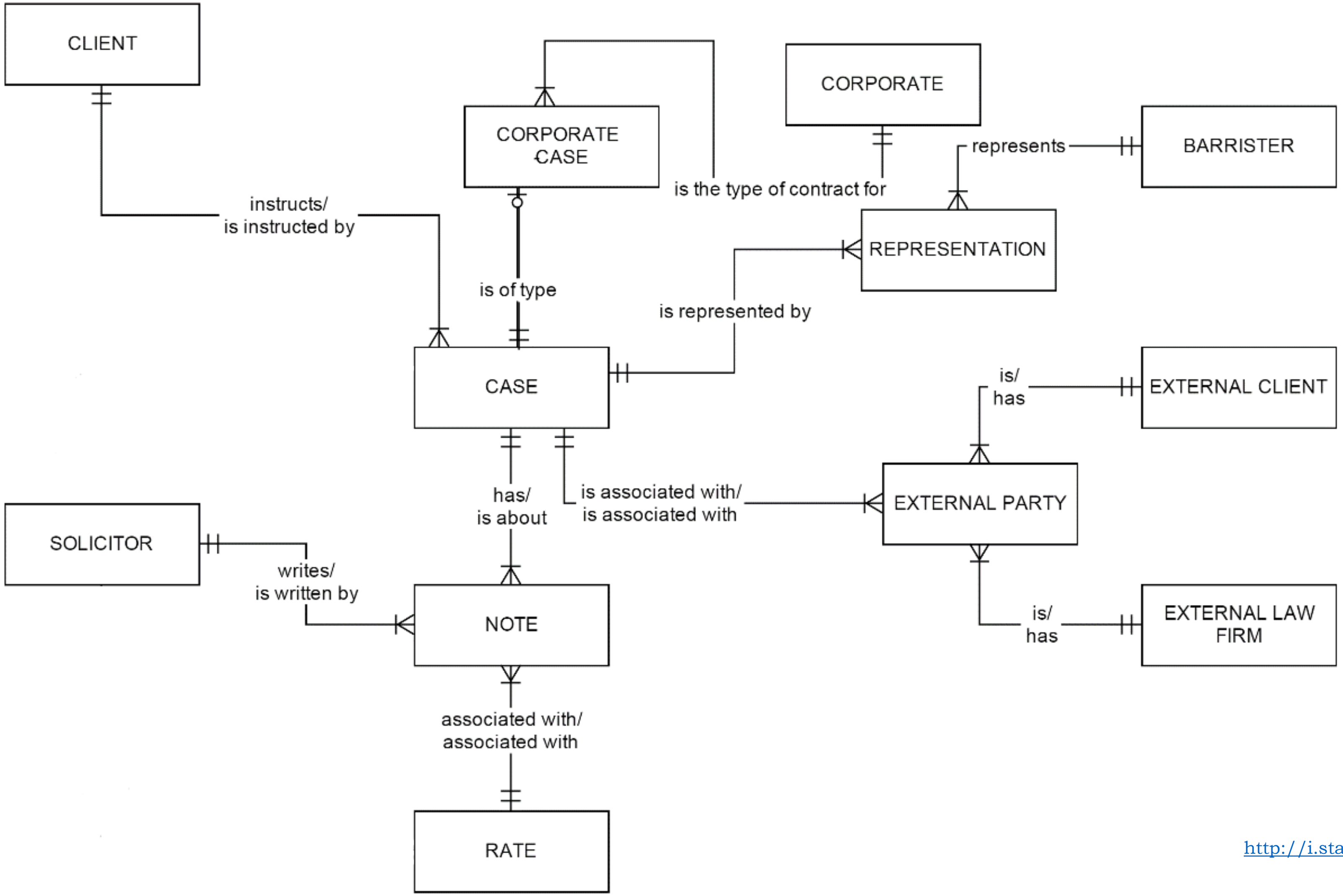


Example: UML Data/Information Model



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

Example: Entity Relationsship Diagram (ERD)



CASE contains general information about contracts.

CORPORATE contains specific information about corporate contracts like the negotiated rates/costs.

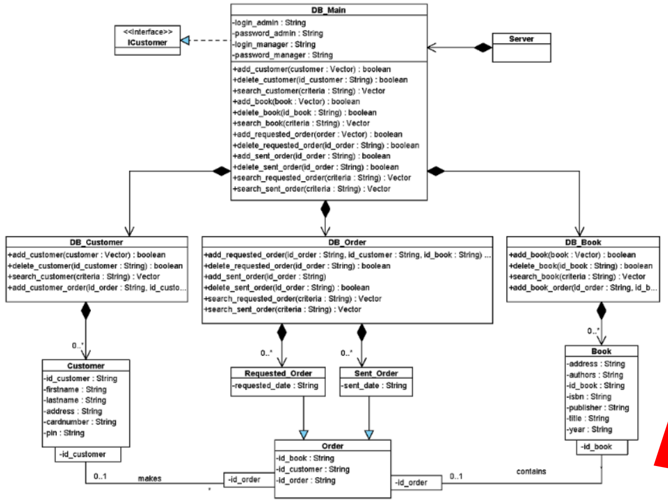
RATE contains information about a solicitors rate; when they are first used (start date), the amount and possibly when they are superseded (end date)/

EXTERNAL PARTY holds information to indicate if the party is opposition or third party

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

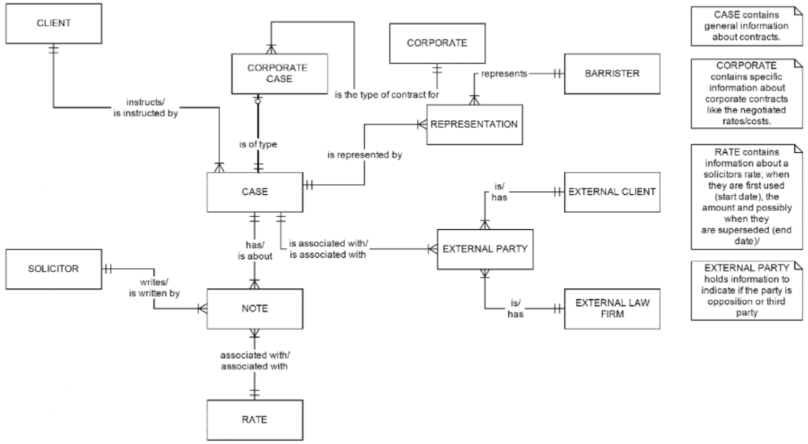
Future-Proof Software-Systems [Part 3B]

UML Data/Information Model



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

Entity Relationsship Diagram (ERD)



CASE contains general information about contracts.

CORPORATE contains specific information about corporate contracts like the negotiated rates/costs.

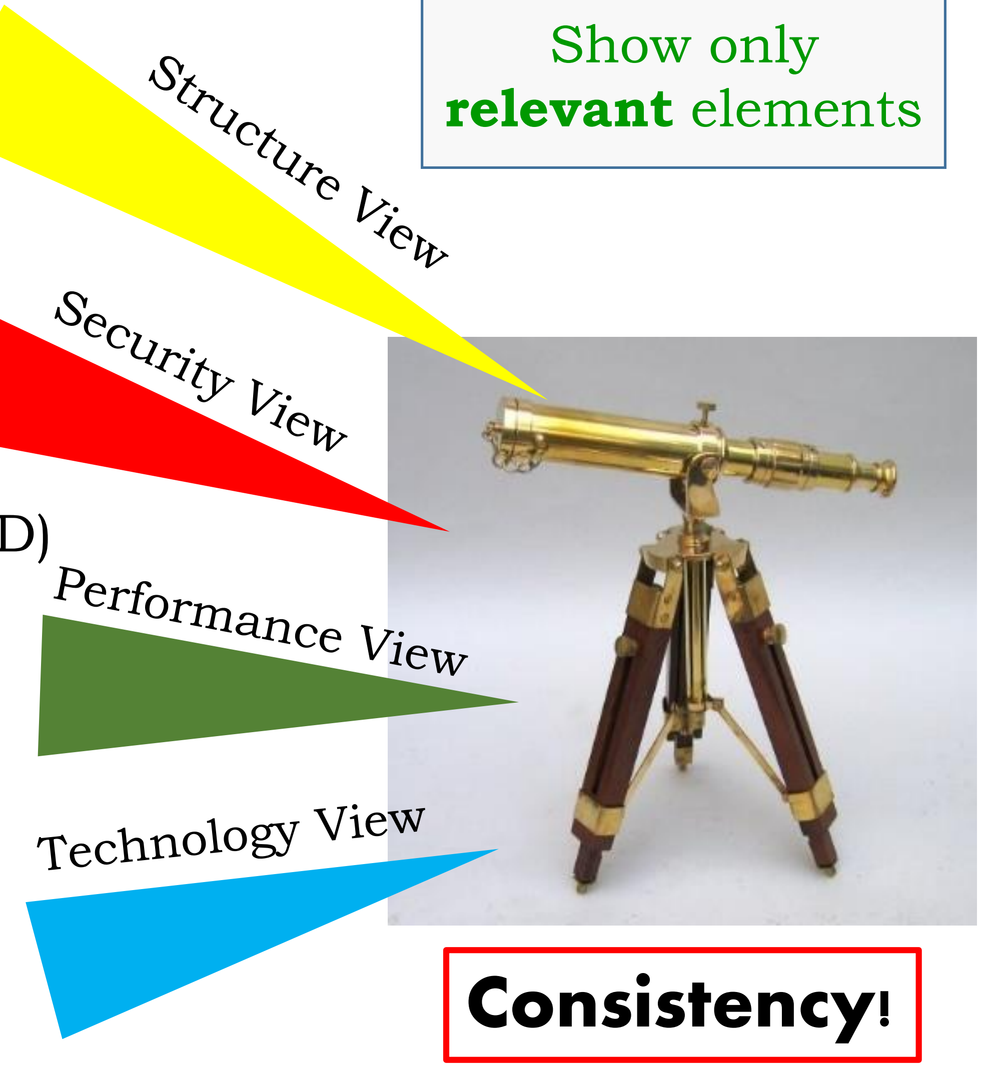
RATE contains information about a solicitors rate; when they are first used (start date), the amount and possibly when they are superseded (end date)/

EXTERNAL PARTY holds information to indicate if the party is opposition or third party

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

Multiple Views:

Show only **relevant** elements



http://www.replicamuseum.com

Consistency!

Future-Proof Software-Systems [Part 3B]

A10 – Part 1

Architecture Principle A10 (a):

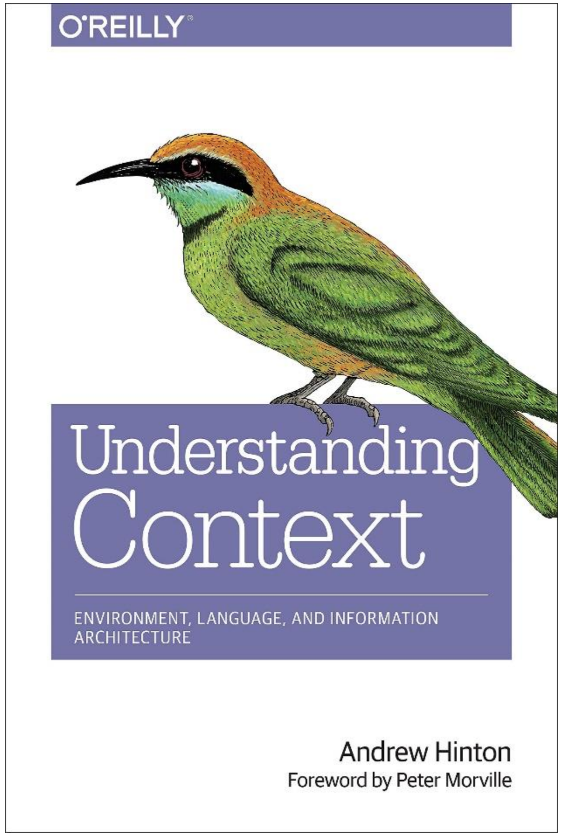
Enterprise Data/Information Architecture

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

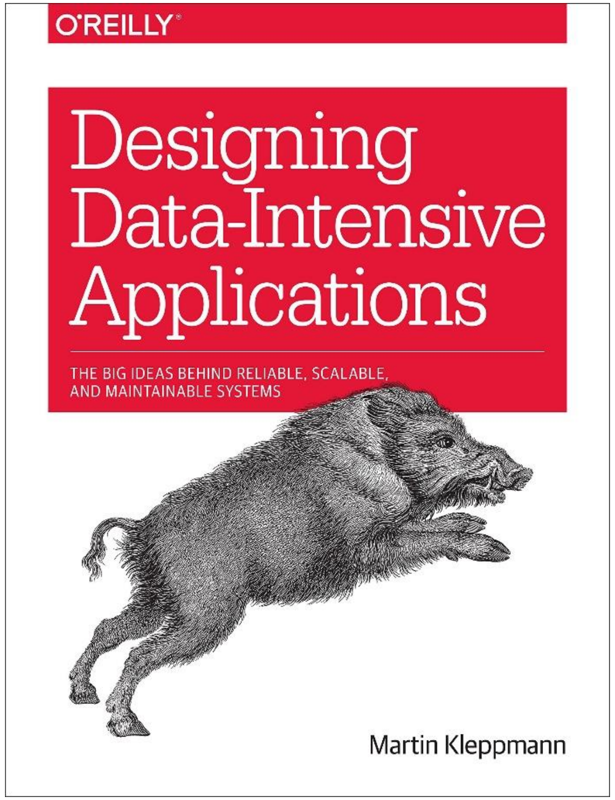
Future-Proof Software-Systems [Part 3A]

Textbook



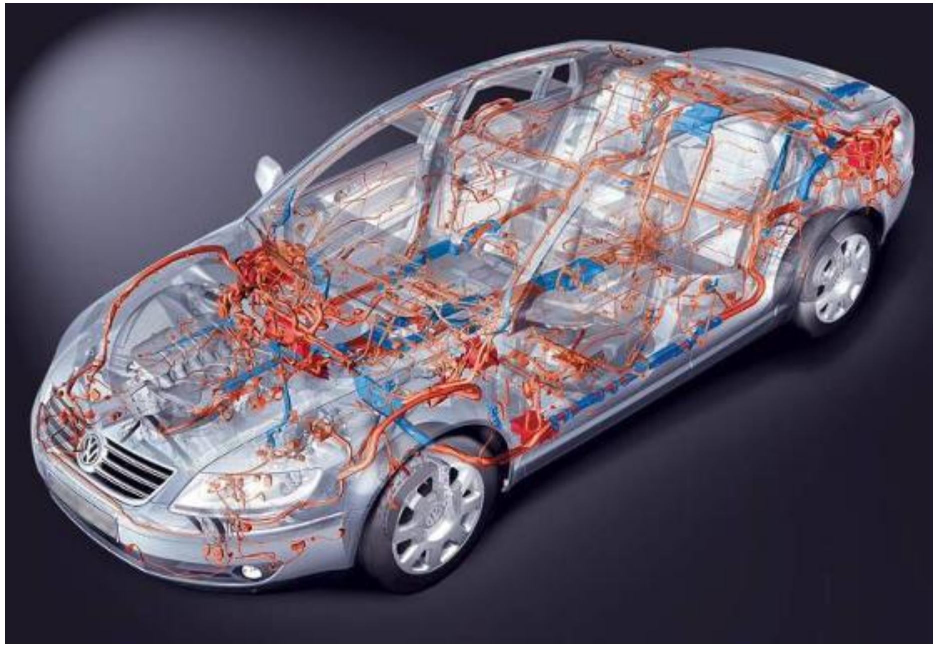
Andrew Hinton:
Understanding Context – *Environment, Language, and Information Architecture*
O'Reilly and Associates, USA, 2015. ISBN 978-1-449-32317-2

Textbook



Martin Kleppmann:
Designing Data-Intensive Applications – *The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*
O'Reilly UK Ltd., revised edition, 2017. ISBN 978-1-449-37332-0

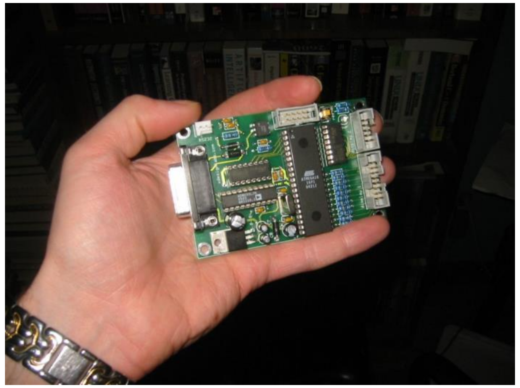
b) Embedded Systems Data/Information Architecture



http://www.ove.uk.com

Example: Vehicle data/information Architecture
[Embedded Systems]

Future-Proof Software-Systems [Part 3B]



http://thorntoncenter.net

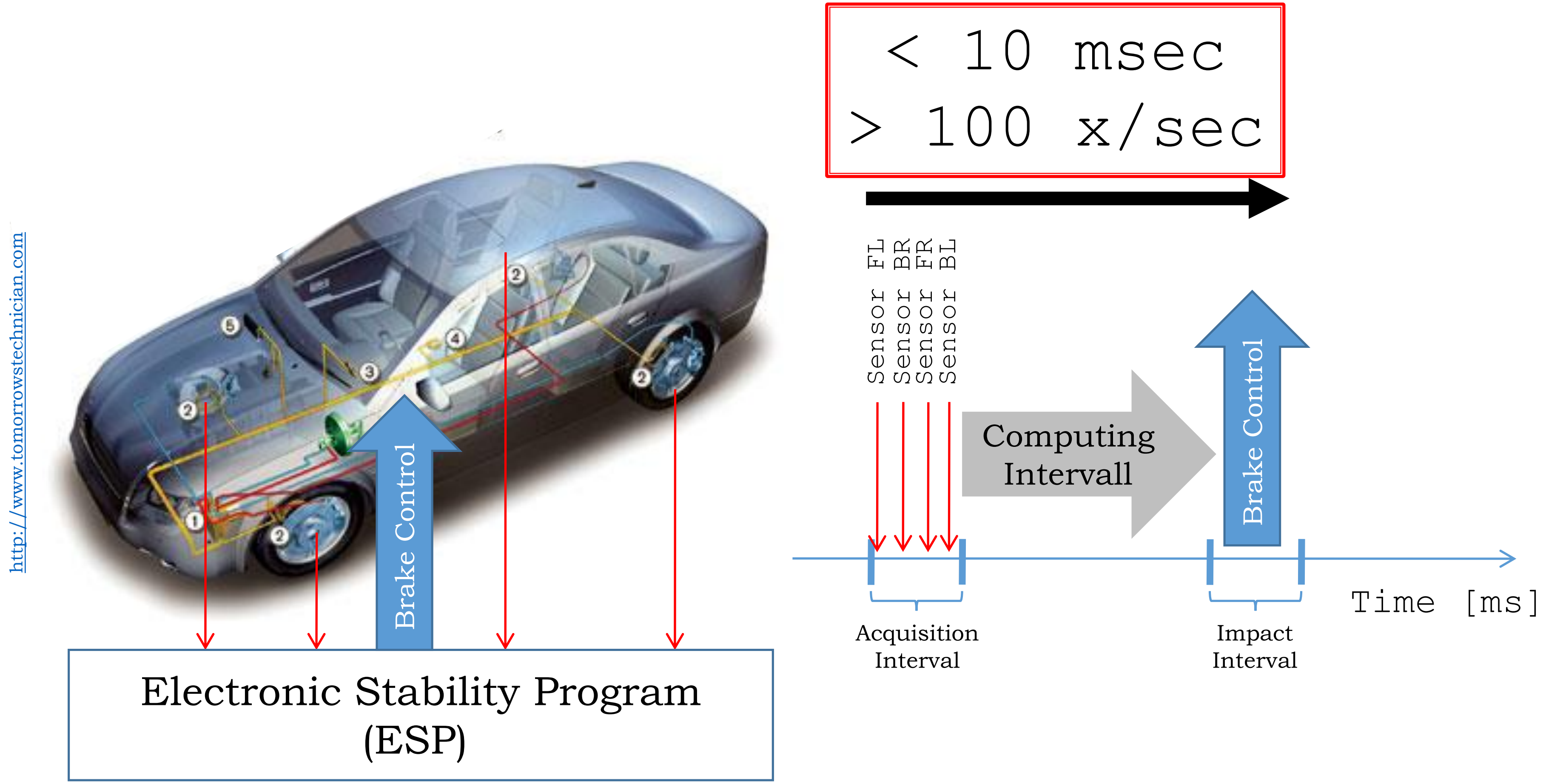
Time !

Data items have *timing relationships* between them

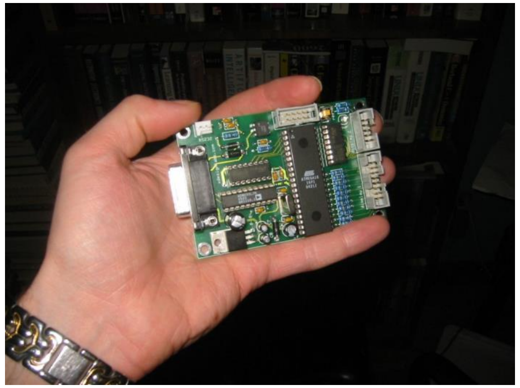
… sometimes very demanding and stringent!

Time & timing relationships are an integral part of an embedded data/information architecture

Example: Wheel rotation information in a brake-by-wire car



Future-Proof Software-Systems [Part 3B]



Data items may have *inconsistencies* between them

… due to mechanical, communications or electronic glitches

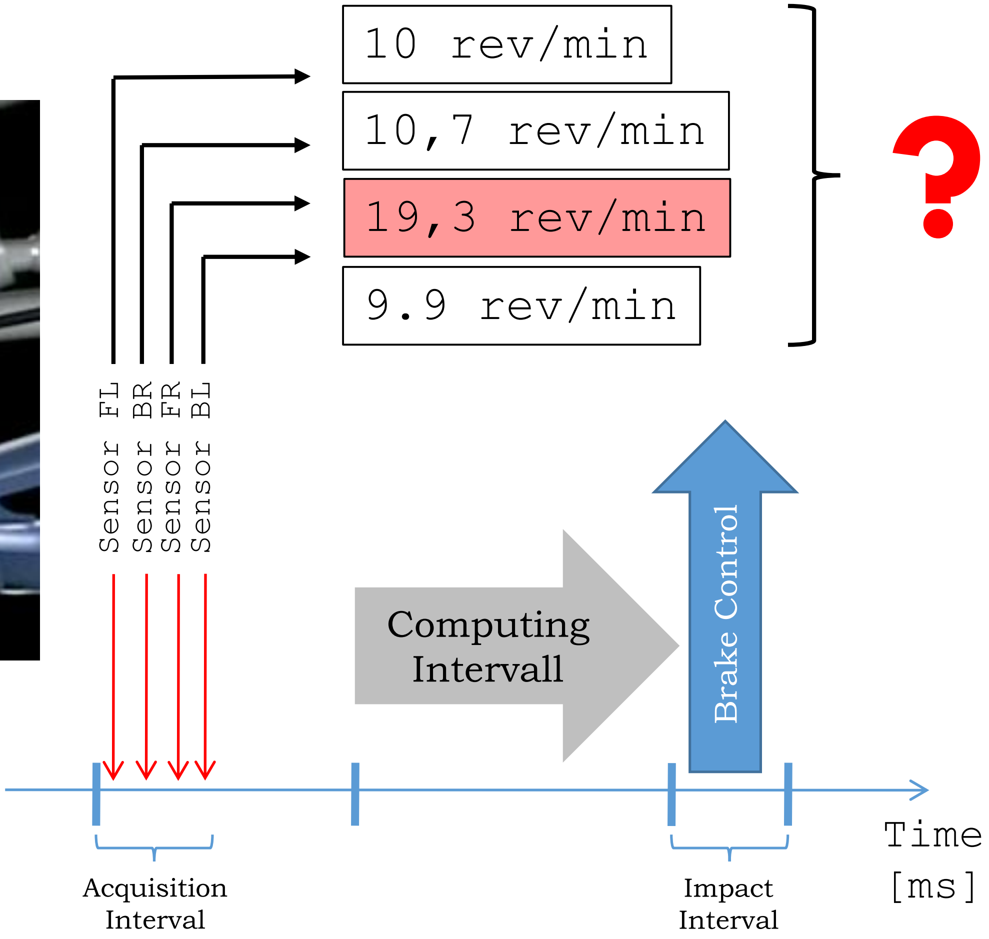
Inconsistency !

Inconsistencies are an important part of an embedded data/information architecture

Example: *Inconsistent* wheel rotation rate information

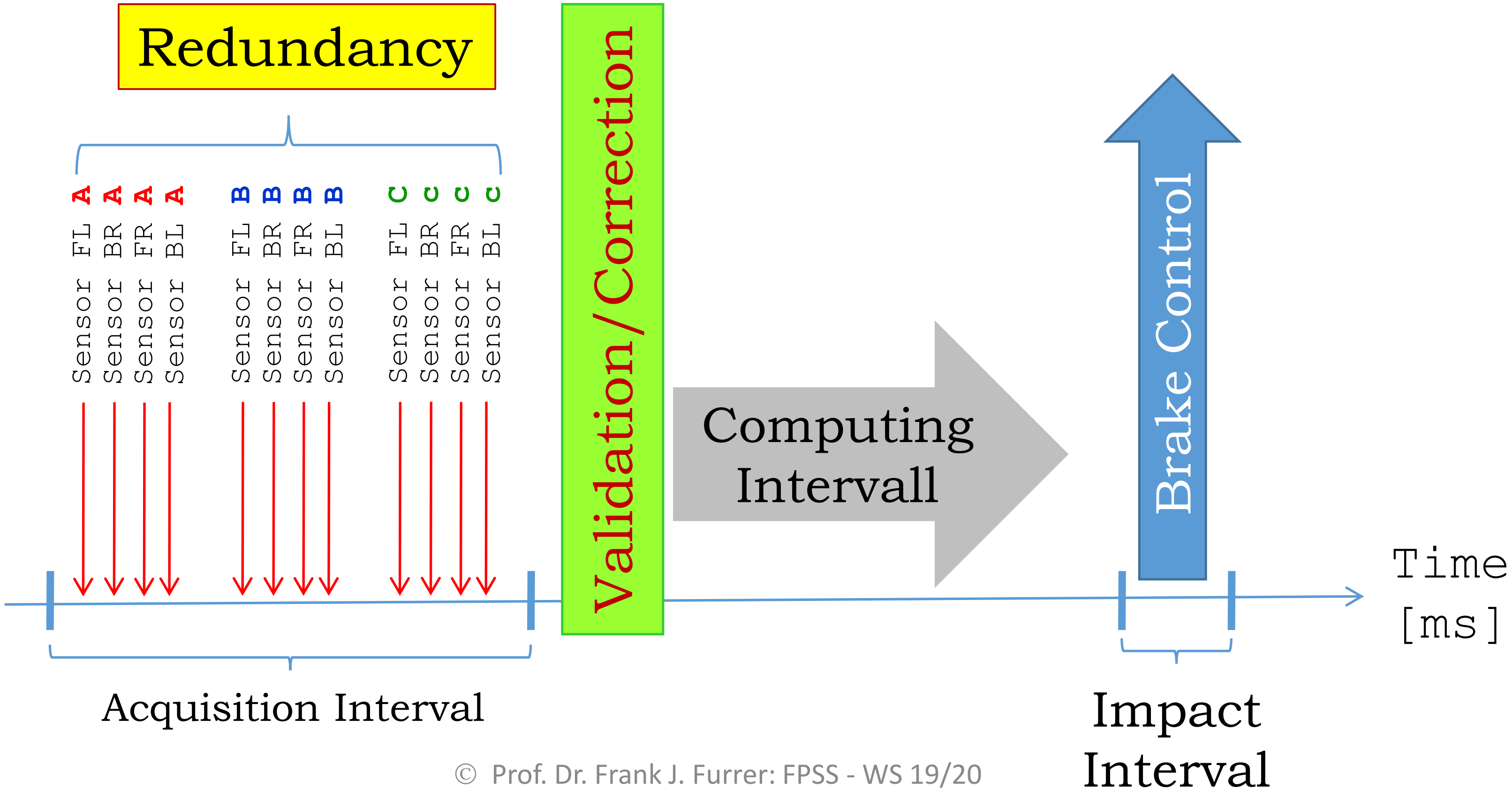
http://www.cdxetextbook.com/brakes

Wheel rotation speed sensor



How do we deal with data inconsistency?

1. Planned *redundancy* in acquisition (multiple sensors)
2. Algorithmic „cleaning“ of data (*Validation*)



Redundancy

Sensor FL A
Sensor BR A
Sensor FR A
Sensor BL A
Sensor FL B
Sensor BR B
Sensor FR B
Sensor BL B
Sensor FL C
Sensor BR C
Sensor FR C
Sensor BL C

Validation/Correction

Computing Intervall

Brake Control

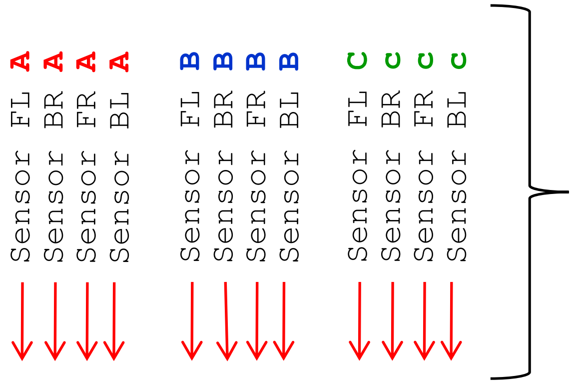
Time [ms]

Acquisition Interval

Impact Interval

Redundancy & Fault Tolerance

Data is acquired multiple times ⇐ managed redundancy



Sensor redundancy

- Time redundancy
- Spatial redundancy
- …

Example: Triple wheel rotation sensor

http://www.designworldonline.com

Sensor data is captured by 3 *independent* sensors and transmitted to the computing unit

Validation

http://www.elevation180.com

Multiple acquisition of data

Validation / Correction

http://www.bfe-inf.org

Consistent, correct data

Deterministic or statistical algorithms

Future-Proof Software-Systems [Part 3B]

A10 – Part 2

Architecture Principle A10 (b):

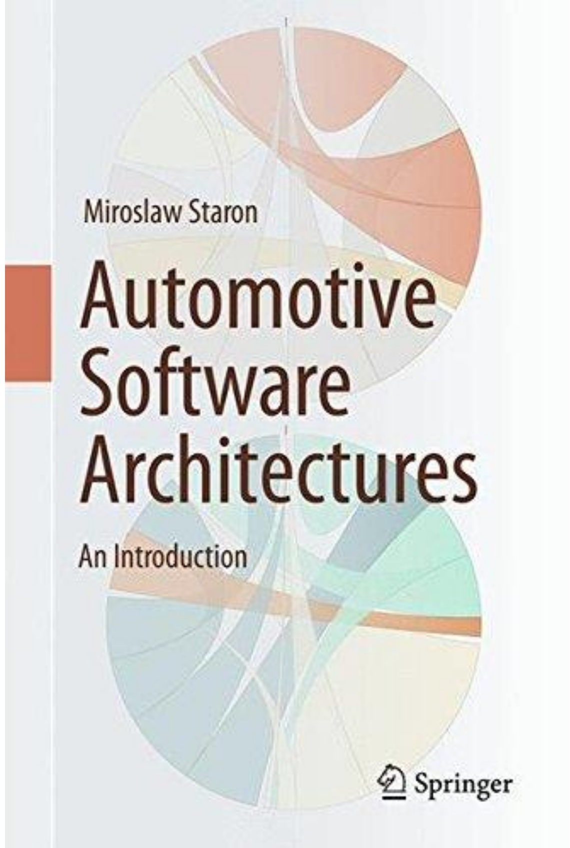
Embedded Data/Information Architecture

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.

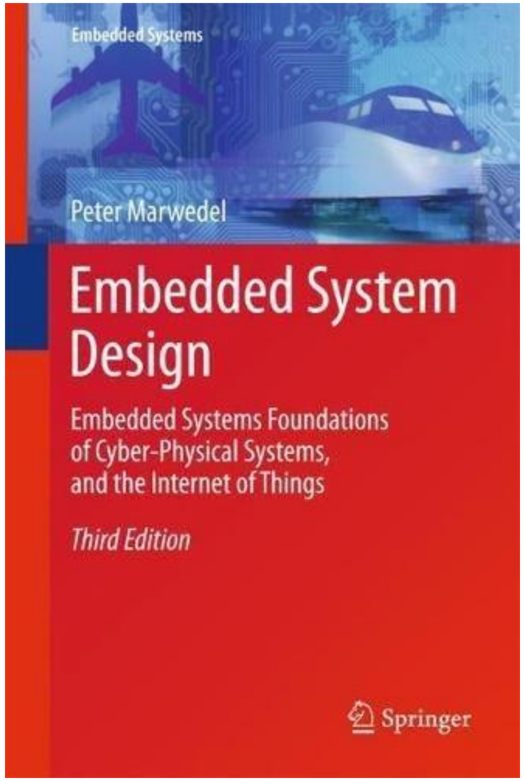
Future-Proof Software-Systems [Part 3A]

Textbook



Miroslaw Staron:
Automotive Software Architectures – *An Introduction*
Springer-Verlag, Germany, 2017. ISBN 978-3-319-58609-0

Textbook



Peter Marwedel:
Embedded System Design – *Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*
Springer-Verlag, Germany, 3rd edition, 2018. ISBN 978-3-319-56043-4

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

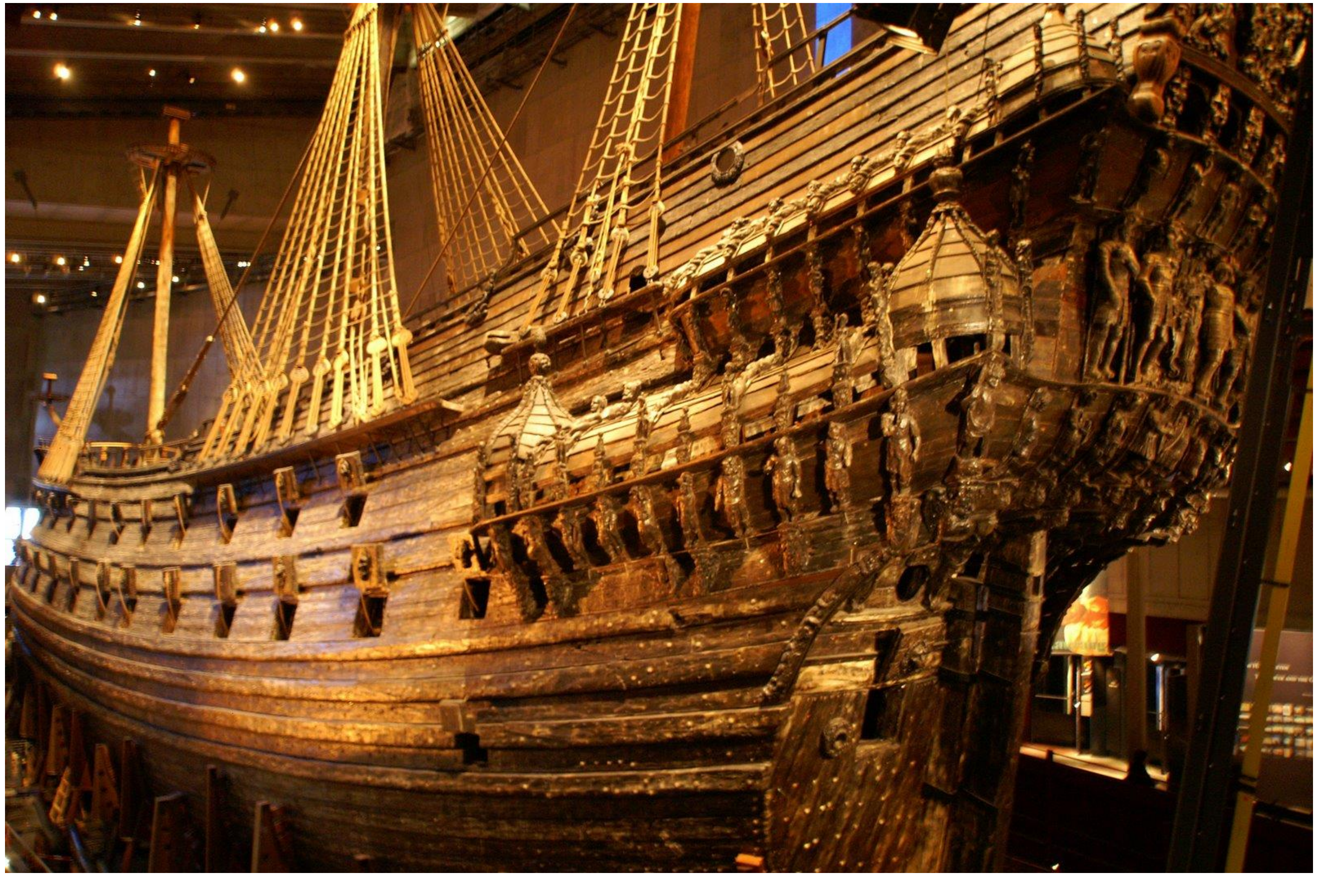
A11

Architecture Principle A11:

Formal Modeling

Example: Vasa (1/3)

1628:
Swedish Warship Vasa
- 2 gun decks
- 32 x 24-pound guns



https://iftachkozik.wordpress.com

On August 10th, 1628 the warship Vasa set sail in Stockholm harbor on its maiden voyage as the newest ship in the Royal Swedish Navy.

The country was at war with Poland and the ship Vasa was urgently needed for the war effort

Example: Vasa (2/3)

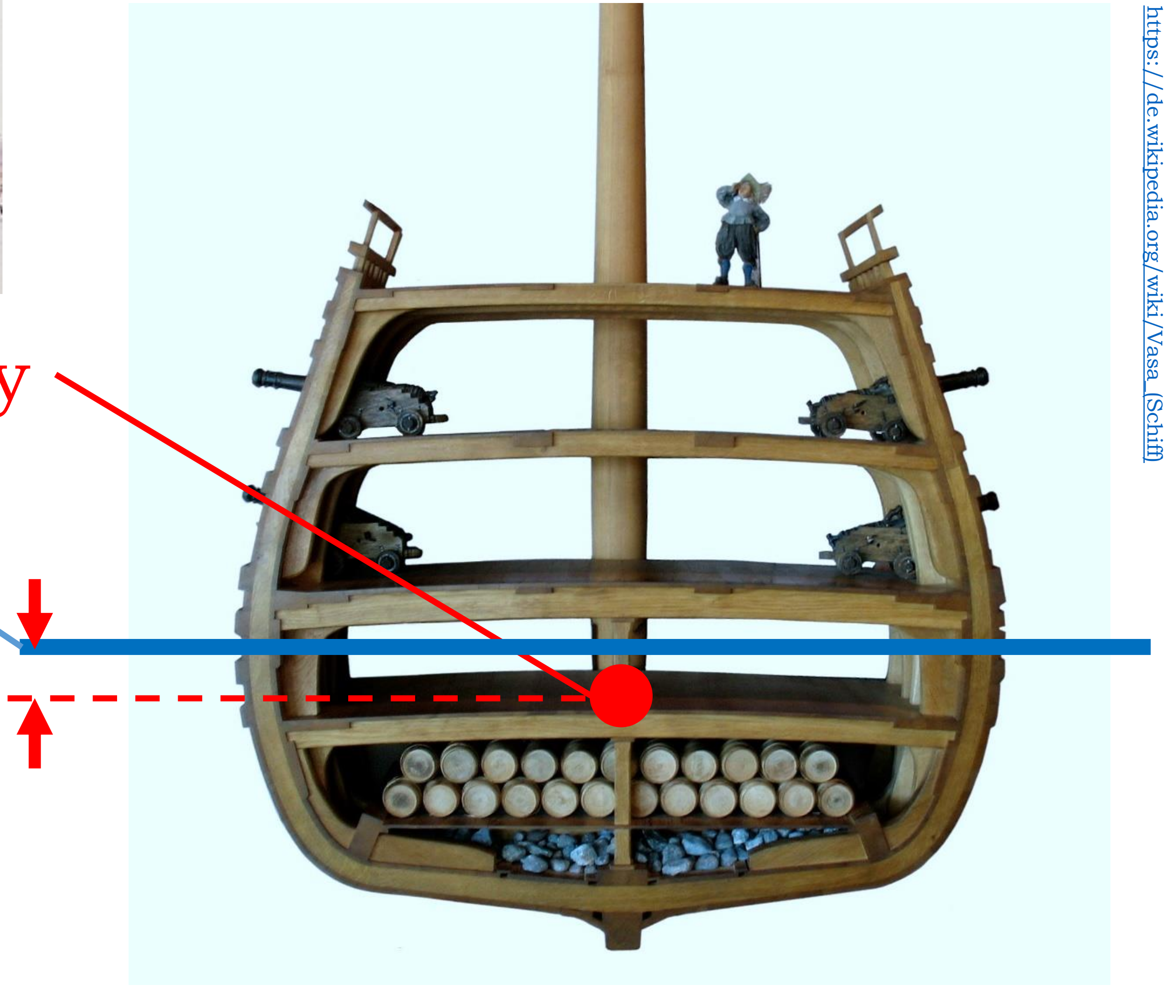
After sailing about 1’300 meters, a light gust of wind caused the Vasa to heel over on its side. Water poured in through the gun portals and the ship **sank**

http://www.hochschule-rhein-waal.de

Example: Vasa (3/3)

What happened?

http://www.hochschule-rhein-waal.de



https://de.wikipedia.org/wiki/Vasa_(Schiff)

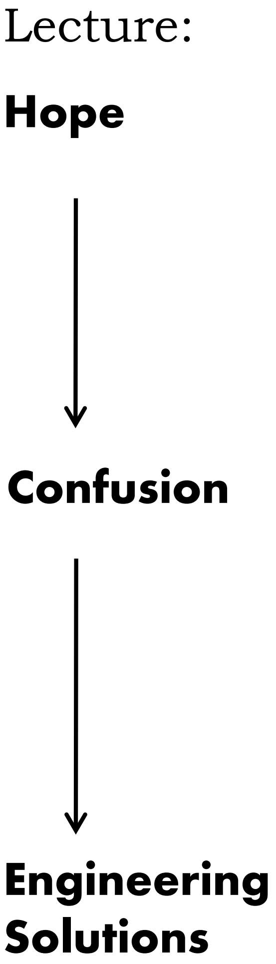
A simple **model** would have shown that the ship was not seaworthy (unstable)!

Modeling of IT-Systems

1. Motivation
2. Definitions
3. State of the Art
4. Engineering Solutions

http://www.domesticatingit.com

http://www.motorauthority.com



Lecture:

Hope

Confusion

Engineering Solutions

Motivation

"All models are wrong - but some are useful"

http://museumvictoria.com.au/treasures

- ⇒ Models *simplify* the real world
- ⇒ Models *abstract* the real world
- ⇒ Models *focus* the real world

Why wrong?

Why useful?

http://museumvictoria.com.au/treasures

Why wrong?

- Oversimplified
- Distances very wrong
- Planet sizes completely wrong
- Movement circular (not elliptical)

Why useful?

- Basic movements understandable
- Important details shown
- Synchronized operation (rotation)
- Projections possible (e.g. distances)

Why models?

http://www.portlandart.net

Adequate Models provide:

- √ **Clarity**
- √ **Committment**
- √ **Communication**
- √ **Control**

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

C The **4** C‘s of models

C1

Clarity

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

C2

Committment

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

C3

Communication

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

C4

Control

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

C The **4** C‘s of models

Before starting any modeling activity, clearly define:

Purpose of the Model

Which is the objective of the model? Which solutions shall the model facilitate? For what shall the model be used? How fine-granular shall the model be? Which is the modeling boundary?

Who is the owner of the model? Which process shall be used to evolve and maintain the model?

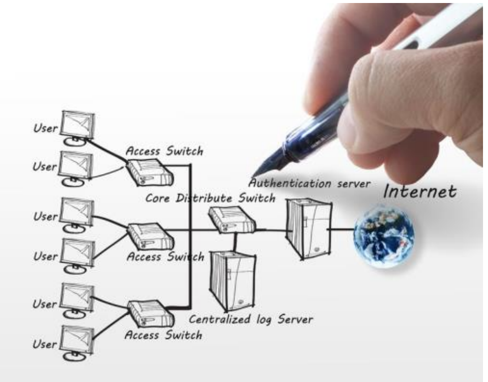
Audience of the Model

Who benefits from the model (stakeholders)? Who needs to agree to the model? Who needs to influence or accept the model? Who finances the modeling activity and what is the model‘s business case?

Definitions

Model: ?

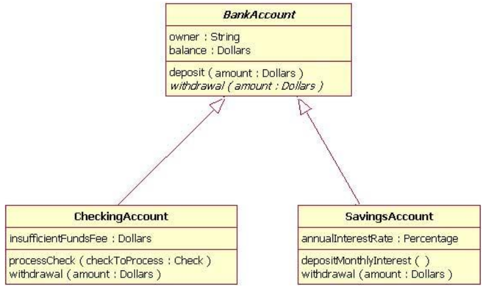
Informal Modeling ⇒



Syntax: Intuitive
Semantics: Intuitive

Informal discussions

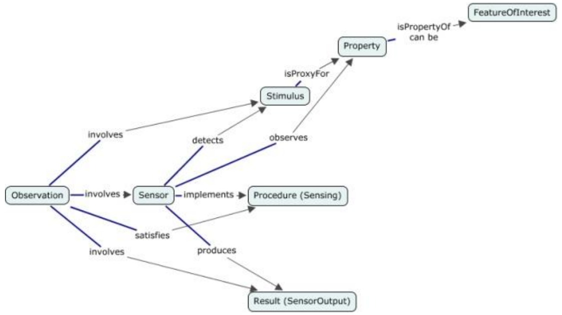
Semi-formal Modeling ⇒



Syntax: Formalized
Semantics: Semi-formal

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

Formal Modeling ⇒



Syntax: Formalized
Semantics: Formalized

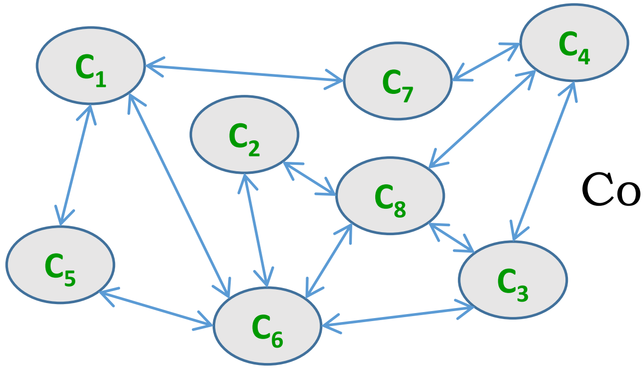
Formal discussions
Extensive Model Checking
Reasoning

Power of model

low

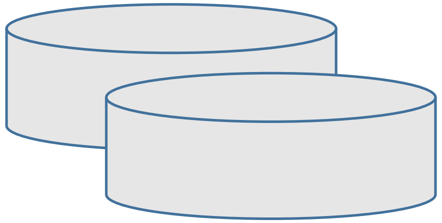
high

Model Typology



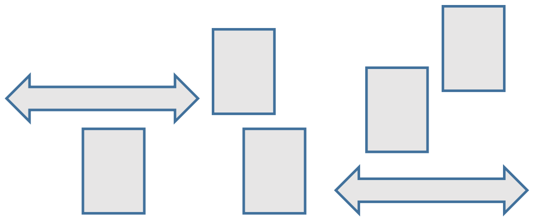
Conceptual Model(s)

Models the concepts, their relationships and their behaviour of a *specific domain*



Database Model(s)

Models the elements and the structure of databases



Technology Model(s)

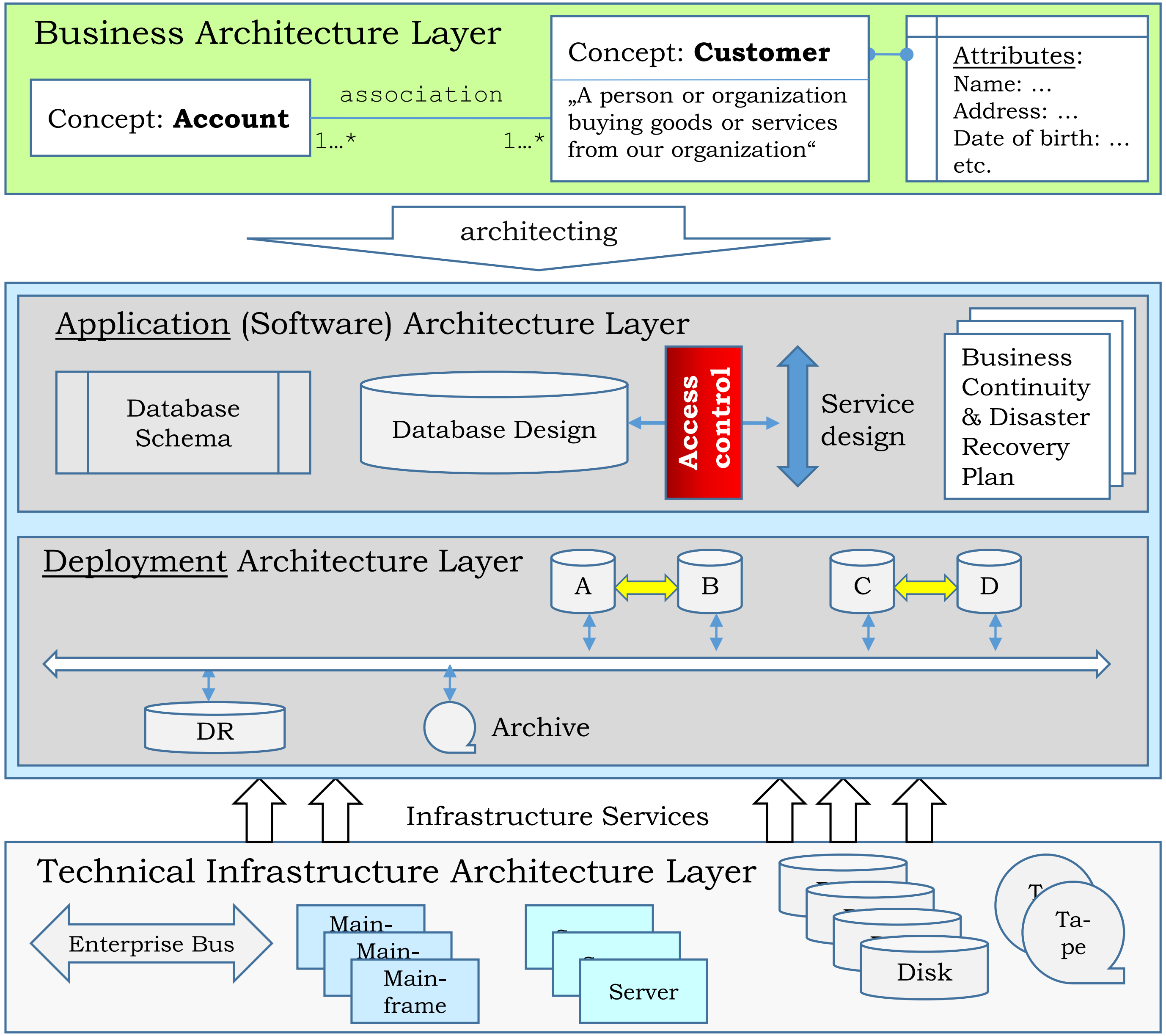
Models the technology elements (servers, networks, busses, system software, backup and disaster recovery configurations etc.)

Future-Proof Software-Systems [Part 3B]

Example: „Customer“

Business area:
Financial institution

Concept:
Customer



Business Architecture Layer

Concept: **Account**

association

1…* 1…*

Concept: **Customer**

„A person or organization buying goods or services from our organization“

Attributes:
Name: …
Address: …
Date of birth: …
etc.

architecting

Application (Software) Architecture Layer

Database Schema

Database Design

Access control

Service design

Business Continuity & Disaster Recovery Plan

Deployment Architecture Layer

A B C D

DR

Archive

Infrastructure Services

Technical Infrastructure Architecture Layer

Enterprise Bus

Main-
Main-
Main-frame

Server

Disk

Ta-pe

DEFINITIONS

Definition

Formal Model:

Abstraction of a specific domain using:

- a precise *syntax*,
- rich *semantics*,
- based on a formal *logical* foundation,

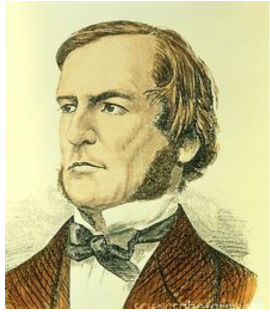
enabling model checking, validation and reasoning

[at least to a certain degree]

Model Expressivity

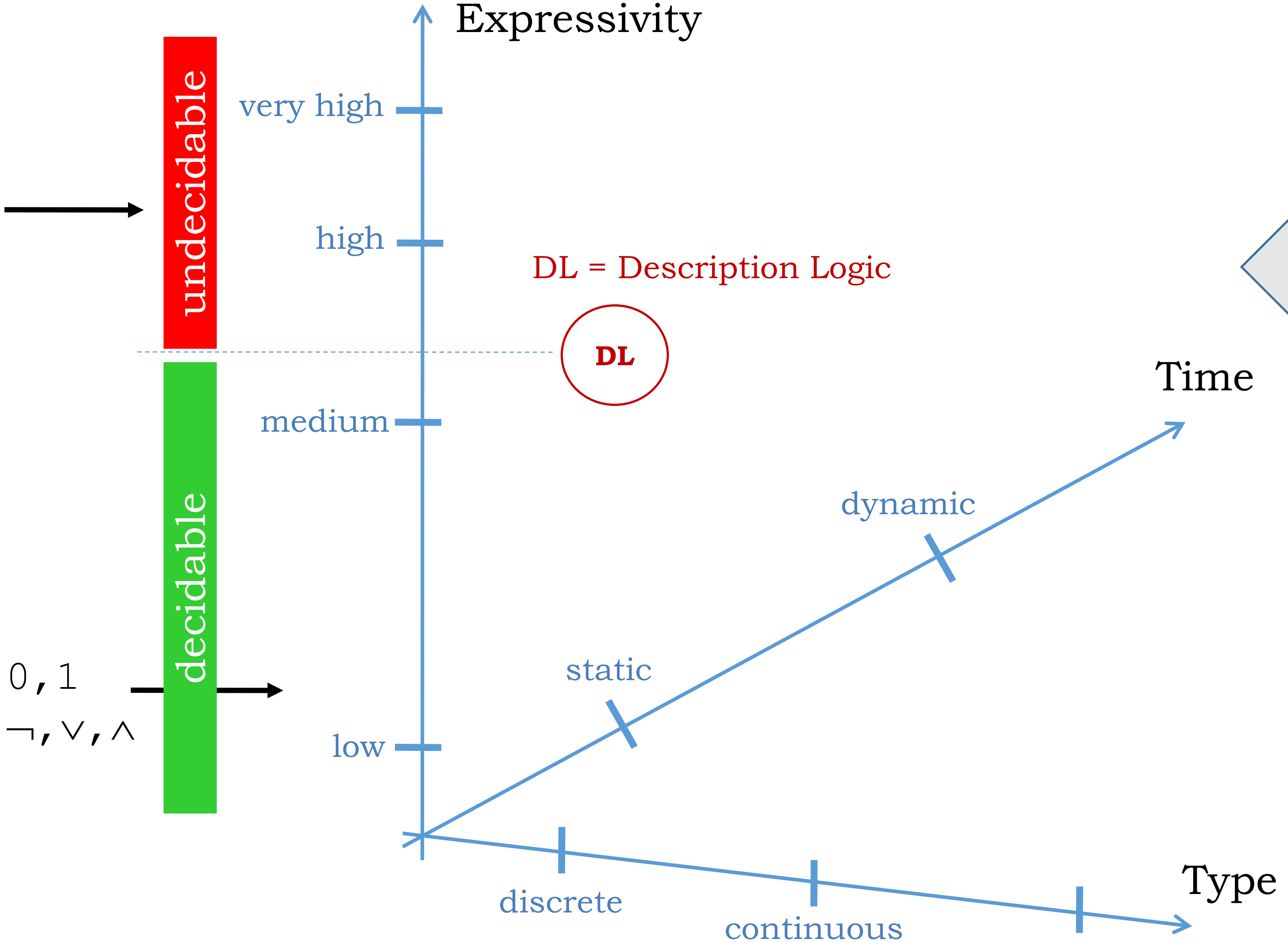
http://www.name-list.net/img/images.php/Godel_5.jpg

Kurt Gödel

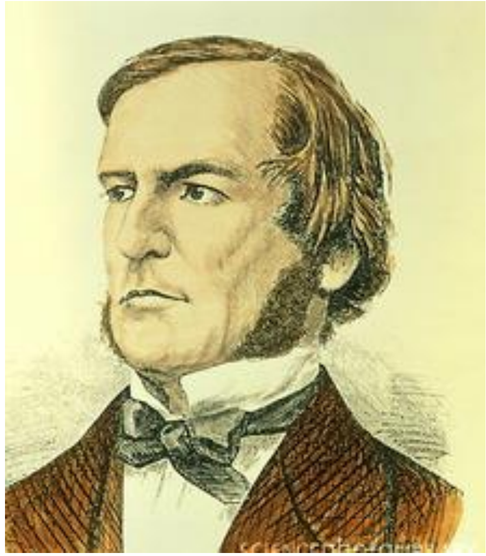


http://ro.math.wikia.com/wiki/George_Boole

George Boole



Engineering use:
Decidable logic
in finite time

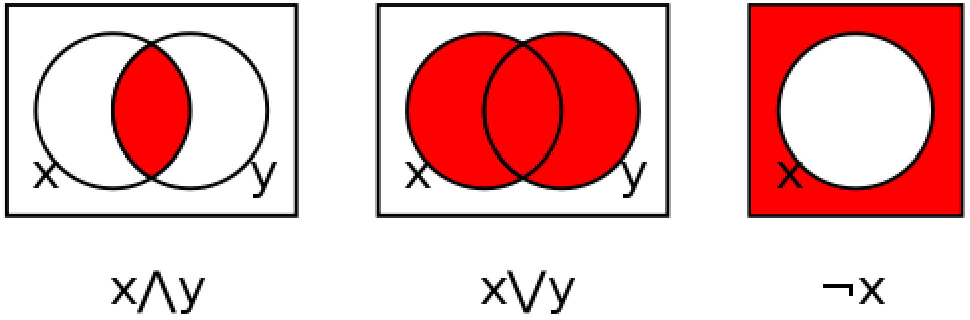


George Boole

Example: Boolean Logic

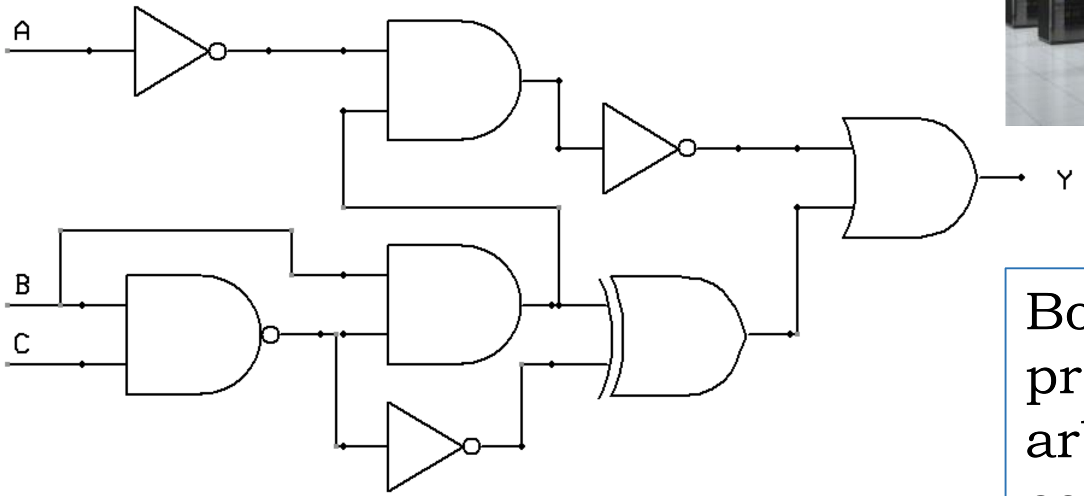
Variables: x ∈ [0,1]

Operators: and, or, not



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

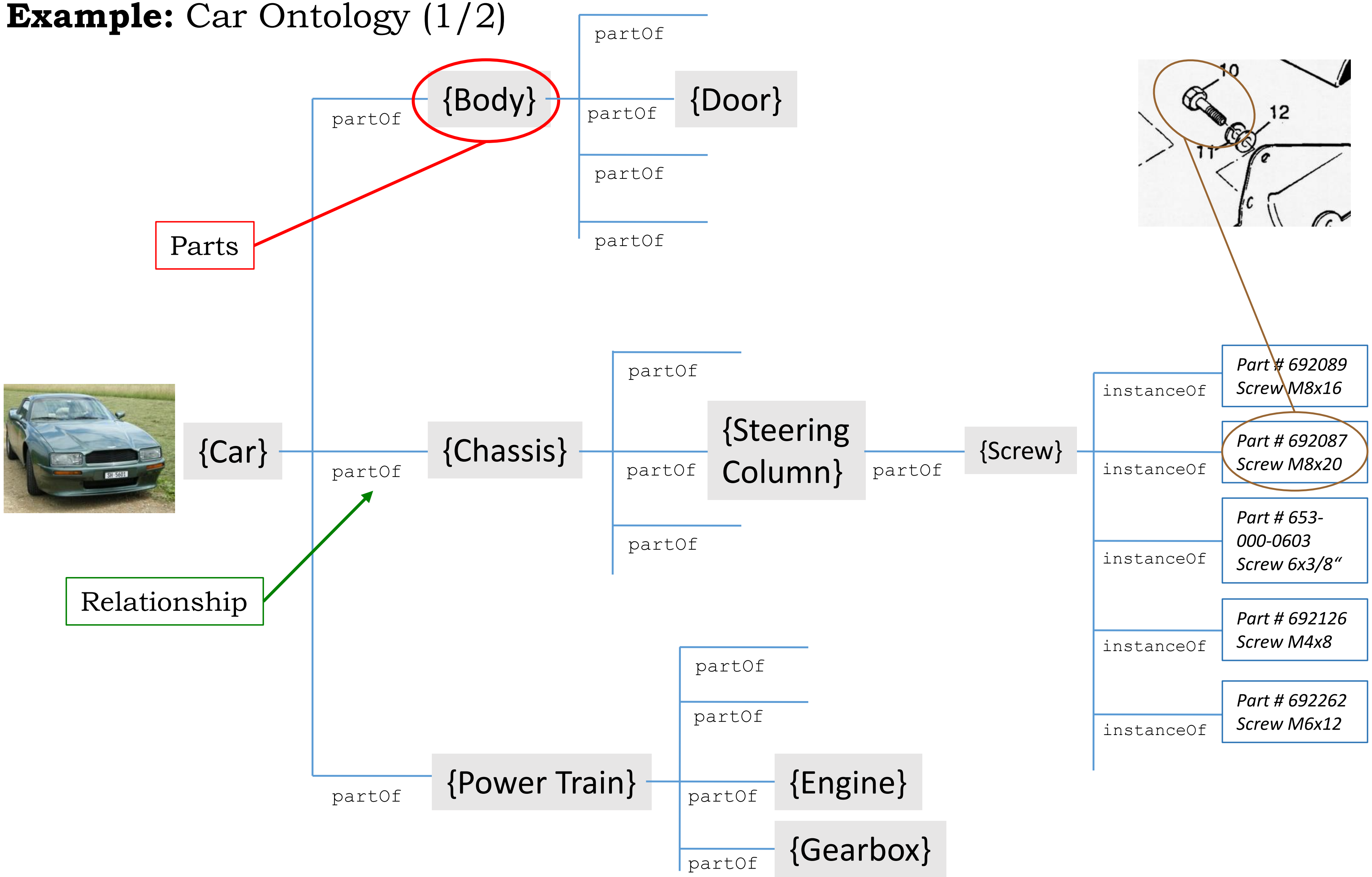
A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



http://drstienecker.com/tech-332

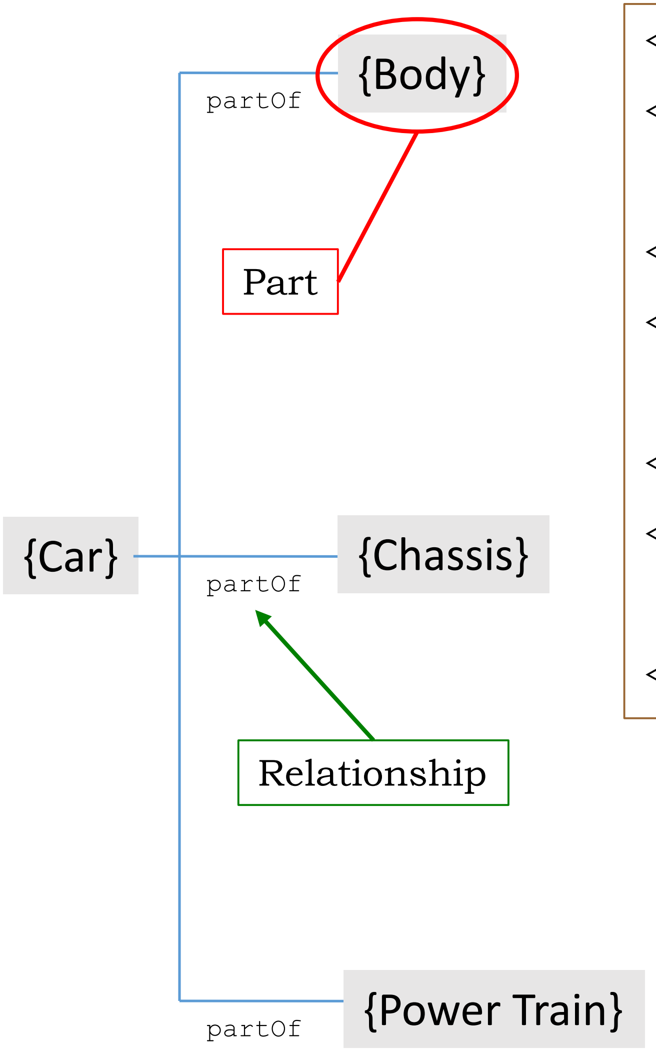
Boolean logic allows the precise modeling of arbitrarily large digital computers

Example: Car Ontology (1/2)



An **ontology** formalizes the complete *structural knowledge* of a specific domain

Example: Car Ontology (2/2)



OWL-**DL** (Web Ontology Language) Representation:

```
<owl:Class rdf:ID=“Car”/>
<owl:Class rdf:ID=“Body”>
     <rdfs:subClassOf rdf:resource=“Car”/>
</owl:Class>
<owl:Class rdf:ID=“Chassis”>
     <rdfs:subClassOf rdf:resource=“Car”/>
</owl:Class>
<owl:Class rdf:ID=“PowerTrain”>
     <rdfs:subClassOf rdf:resource=“Car”/>
</owl:Class>
```

Kurt Gödel

Formal, machine-processable, decidable representation of the domain structure knowledge

http://www.esa.int

Modeling is a powerful instrument.

It provides:

- √ **Clarity**
- √ **Committment**
- √ **Communication**
- √ **Control**

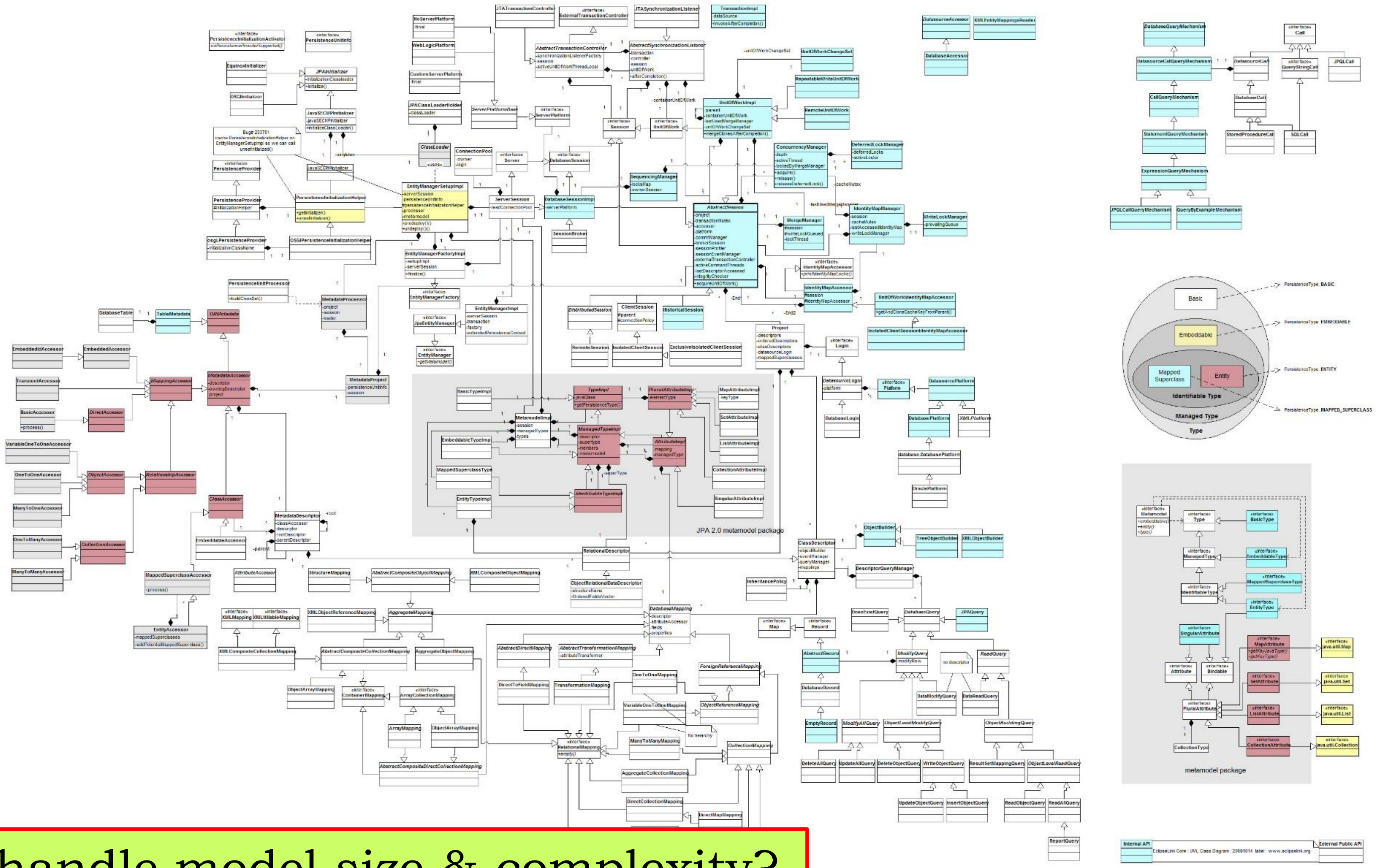
$C_{1..4}$

The **4** C‘s of models

… during the whole life-cycle of an IT-system

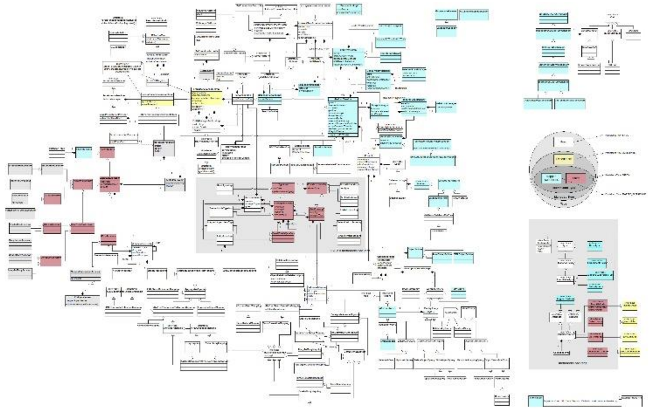
However:

Models can become very large and complex!



http://wiki.eclipse.org

How can we handle model size & complexity?



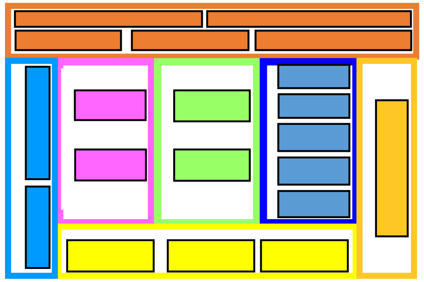
How can we handle model size & complexity?

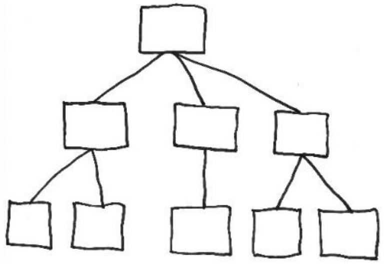
Domains

Hierarchical refinement

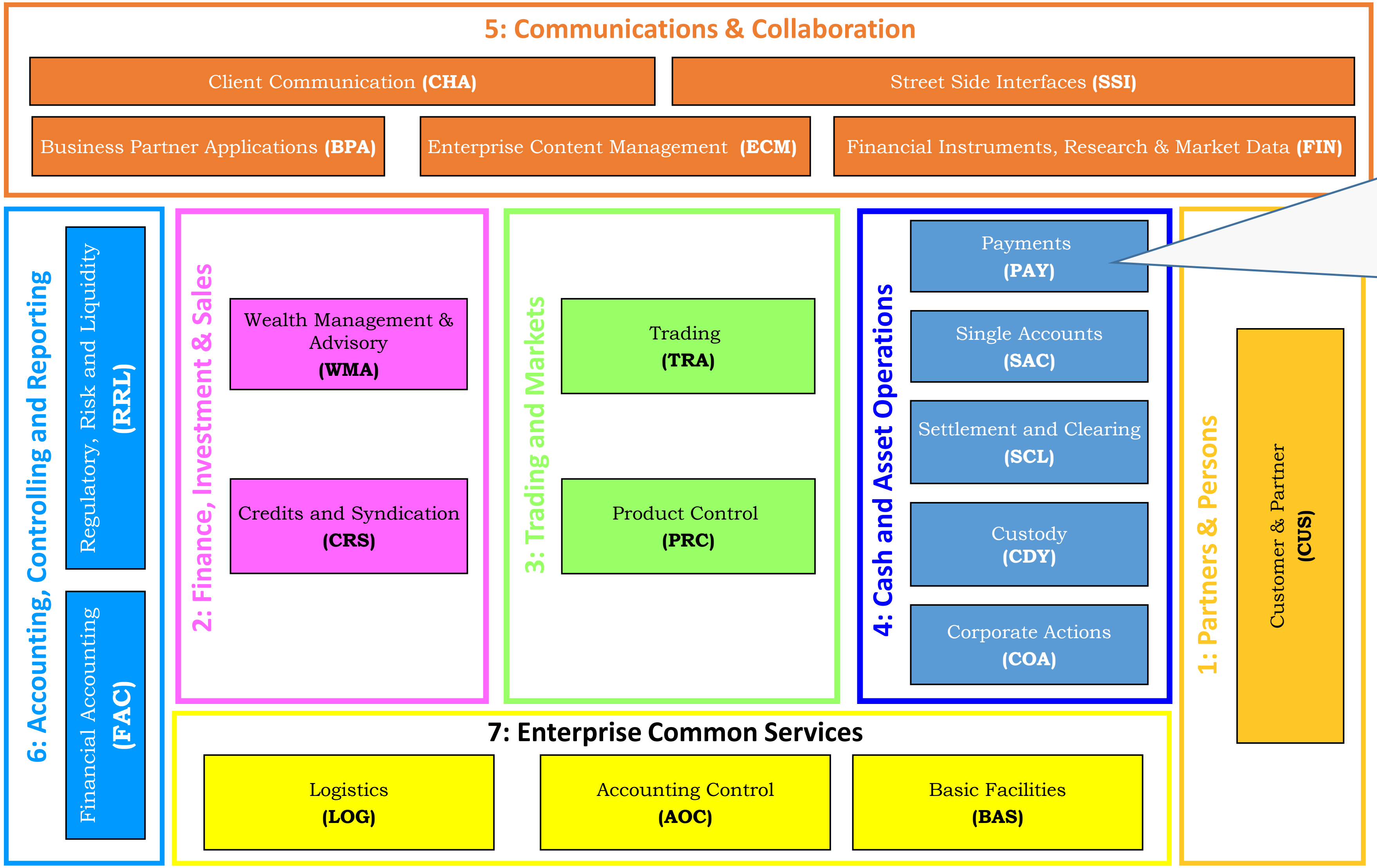
Views

Tools





Model Refinement «Domains»



5: Communications & Collaboration

- Client Communication **(CHA)**
- Street Side Interfaces **(SSI)**
- Business Partner Applications **(BPA)**
- Enterprise Content Management **(ECM)**
- Financial Instruments, Research & Market Data **(FIN)**

6: Accounting, Controlling and Reporting

- Regulatory, Risk and Liquidity **(RRL)**
- Financial Accounting **(FAC)**

2: Finance, Investment & Sales

- Wealth Management & Advisory **(WMA)**
- Credits and Syndication **(CRS)**

3: Trading and Markets

- Trading **(TRA)**
- Product Control **(PRC)**

4: Cash and Asset Operations

- Payments **(PAY)**
- Single Accounts **(SAC)**
- Settlement and Clearing **(SCL)**
- Custody **(CDY)**
- Corporate Actions **(COA)**

1: Partners & Persons

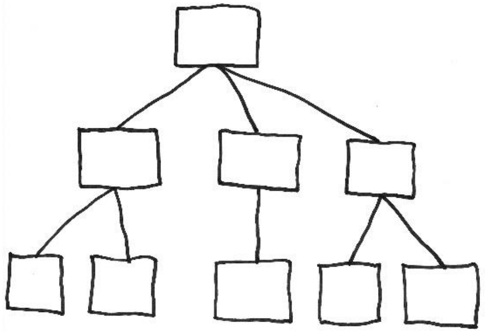
- Customer & Partner **(CUS)**

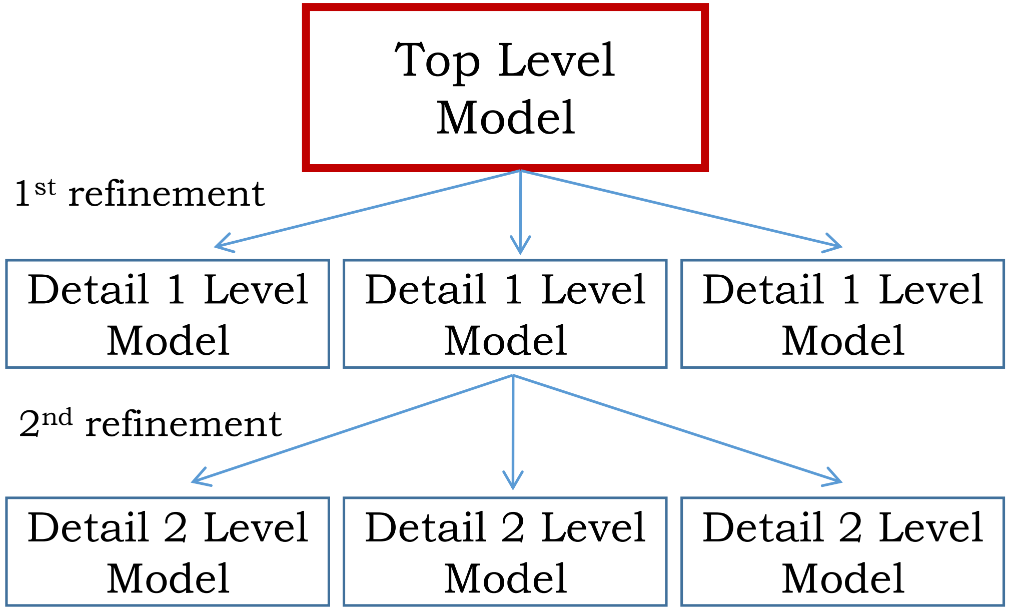
7: Enterprise Common Services

- Logistics **(LOG)**
- Accounting Control **(AOC)**
- Basic Facilities **(BAS)**

Partitioning the system into «**domains**» and modeling each domain individually ⇒ *massive complexity reduction*

Model Refinement (Model hierarchy):

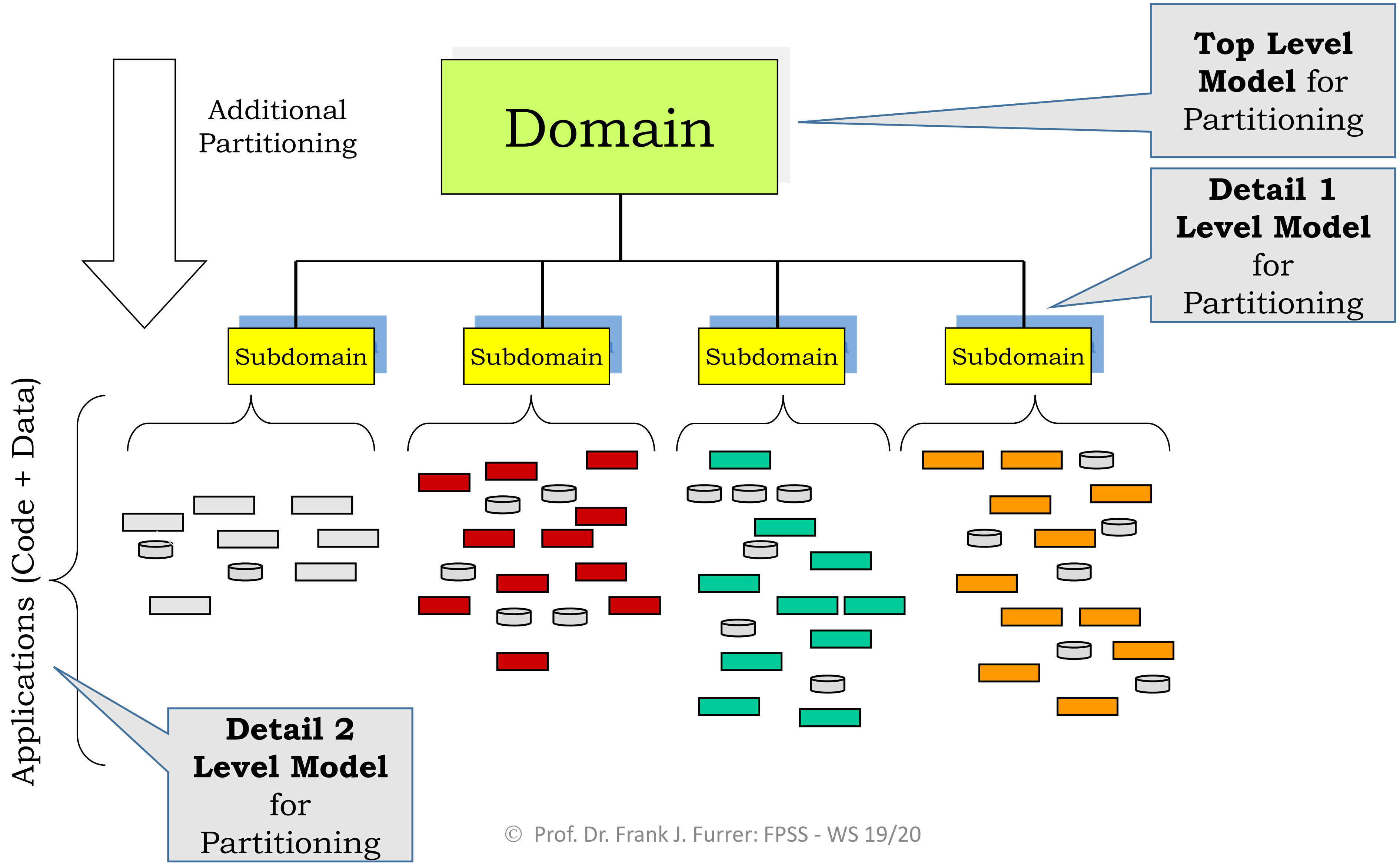




Increasing level of detail

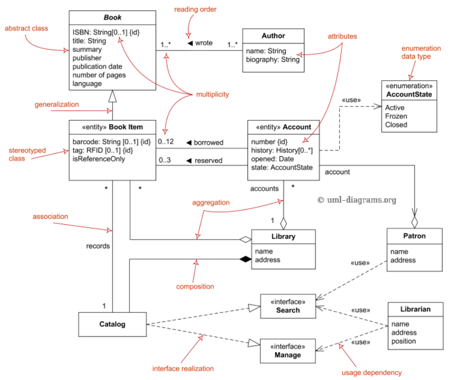
Paradigm: **Inheritance**

Example: Domain & Hierarchy Model for a Financial Institution



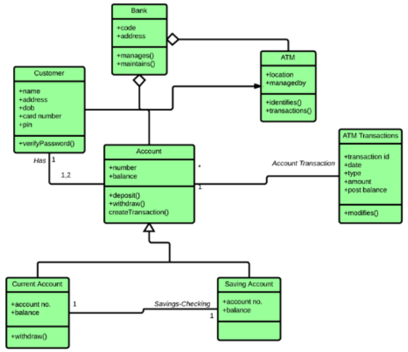
Future-Proof Software-Systems [Part 3B]

Model Views:



Structure

Top Level Model



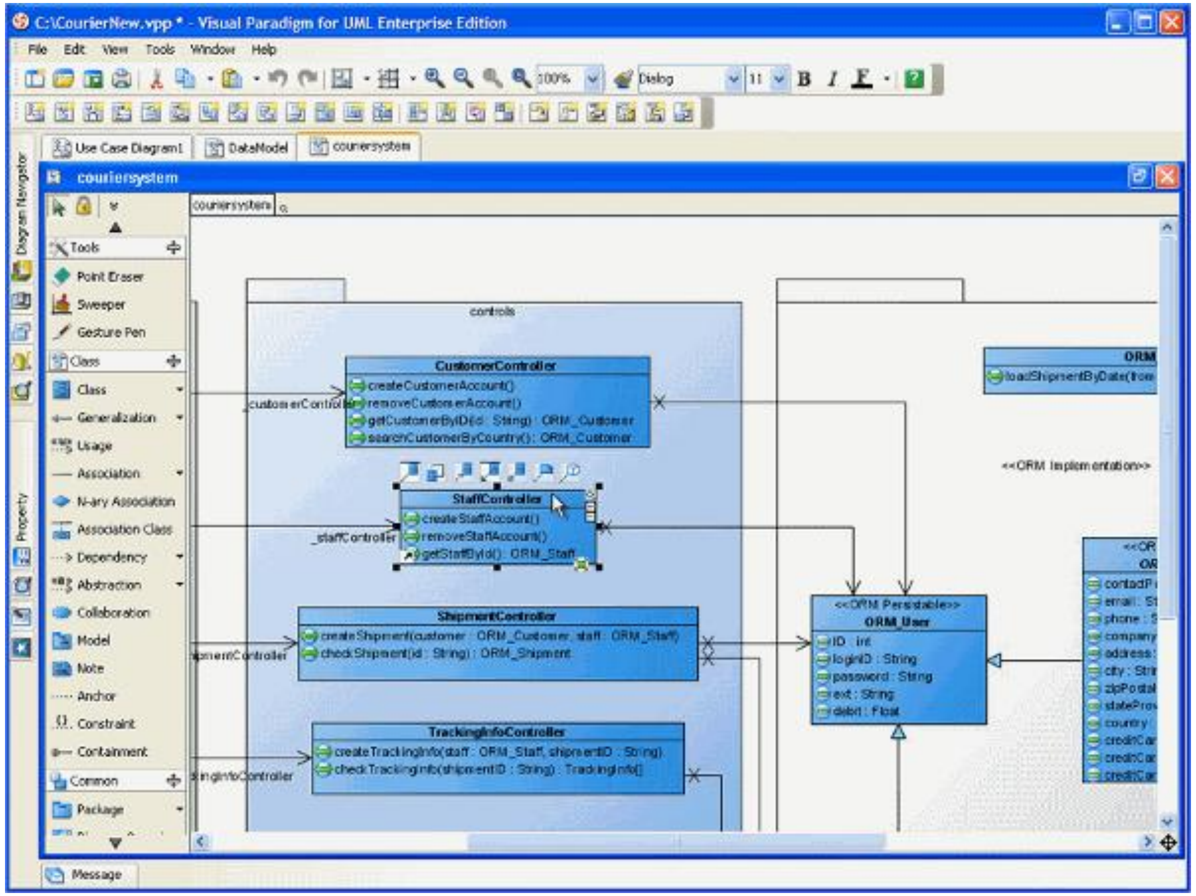
Security

etc.

The complete model is segmented into specific, consistent **views**

Modeling Tools:

- Language Editor
- Syntax Check
- Composition
- Libraries
- Administration
- Exchange
- Graphics
- Profiles/Extensions
- Views



Tools supporting the full modeling cycle

Modeling of IT-Systems


www.123rf.com

Business Stakeholders

… need to model:

- *Business processes*
- *Data/Information content & structure*
- *Future business scenarios*
- *External relationships*


http://creattica.com

IT Stakeholders

… need to model:

- *IT system structure*
- *IT system behaviour*
- *IT system interaction with the environment*
- *IT system evolution*
- *IS system operation*

Conceptual Integrity
Consistency

State of the Art

Modeling Universe

Hierarchy

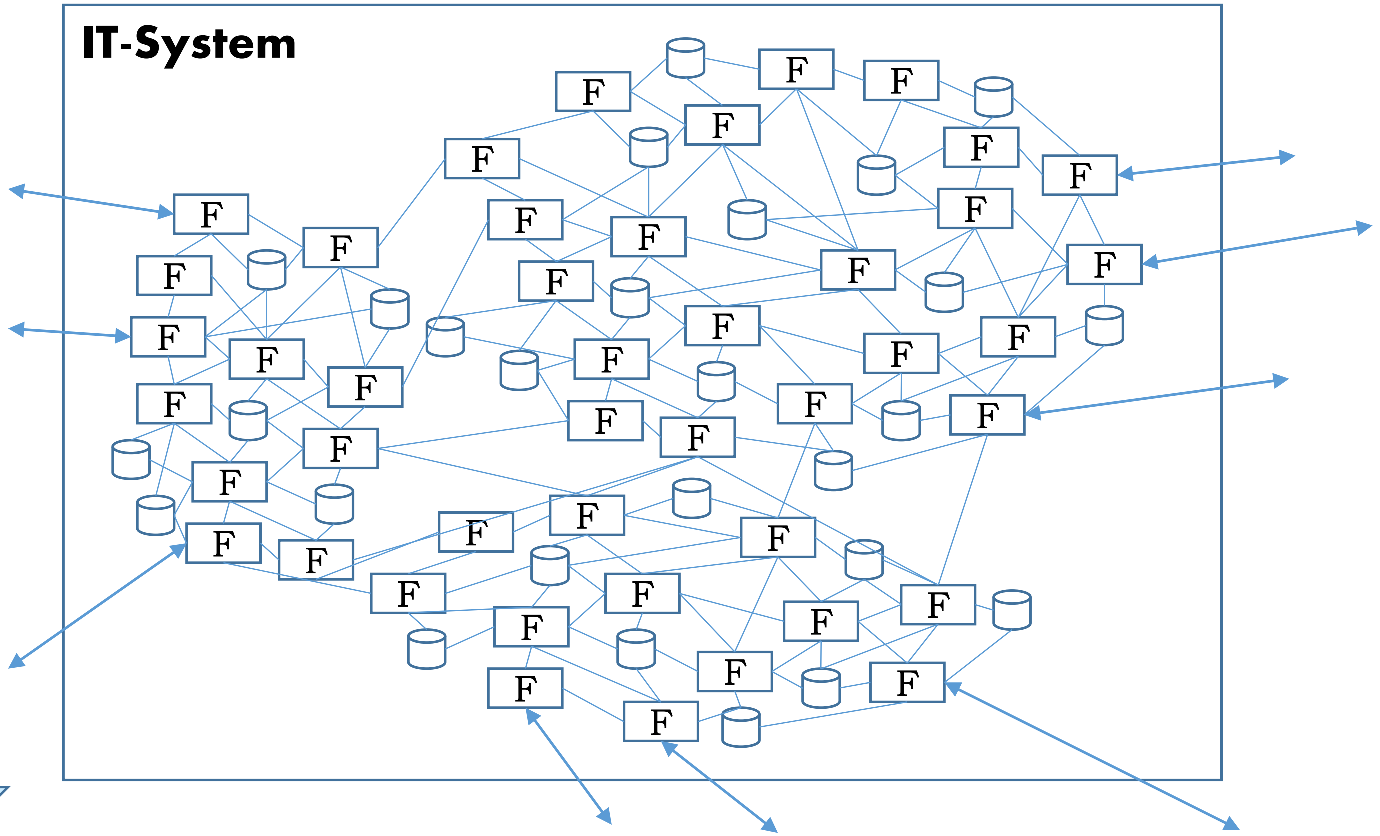
World	System-of-Systems (SoS)			Application Domain	System		
	Structure	Behaviour	Interaction with the environment		Structure	Behaviour	Interaction with the environment

Modeling **Hierarchy**

Level

- **Metamodel** [→ Modeling elements & Notation]
- **Conceptual Model** [→ Concepts & relationships in the real application domain world]
- **Architecture Model** [→ Structure, i.e. parts and their connections]
- **Implementation Model** [→ Planned deployment of the parts + connections]

Modeling **Level**



State of the Art

Modeling Instruments: Overview

Modeling Level	**World**	**System-of-Systems (SoS)**			**Application Domain**	**System**		
		Structure	**Behaviour**	**Interaction with the environment**		**Structure**	**Behaviour**	**Interaction with the environment**
Metamodel	*Boundary Definition*	*SoS Metamodel*	*SoS Interaction Model*	*SoS Interaction Model*	*Domain Metamodel*	*OMG Metamodel & Profiles, Graphs*	*Component Model*	*Interface theories, Contract Models*
Conceptual Model	*Upper (World) Ontology*	*UML, SysML*	*System Black Box Model, Governance Model*	*SoS-Model, UML, SysML, Contracts (Technical & Legal)*	*Domain Ontology, Business Object Model, Application Domain Model (DSL), Business Process Models*	*UML, SysML*	*Hybrid Components, Business Process Orchestration*	*SoS-Model, UML, SysML, Contracts*
Architecture Model	-	*UML, SysML, Petri-Nets Frameworks,*	*Contracts, Web-Stds (e.g. WSDL)*	*Contracts, Web-Stds (e.g. WSDL)*	*Reference architecture, Architecture Framework*	*UML, SysML, Petri-Nets Frameworks, Contracts*	*State-machines, timed automata, Simulink, Contracts, Web-Stds (e.g. WSDL)*	*Contracts, Web-Stds (e.g. WSDL)*
Implementation Model	-	*Annotated, directed Hypergraphs*	-	*Annotated, directed Hypergraphs*	-	*Annotated, directed Hypergraphs*	-	*Annotated, directed Hypergraphs*
Run-Time Model	-	*model@run-time*	-	*model@run-time*	-	*model@run-time*	-	*model@run-time*

Modeling of IT-Systems: **State of the Art**

ERD

Frameworks

Domain Model

Contracts

UML

Simulink Models

Taxonomy

SysML

Business Object Model

Directed Hypergraph

Timed Automata

Petri Net

State Machines

Ontology OWL

model@runtime

http://www.ubizoo.de

Modeling Zoo

Modeling of IT-Systems: **State of the Art**

Why the confusion?

- Modeling is an *evolving* science (Many papers/books published every year)
- Modeling instruments depend heavily on *purpose/audience*
- The standardization bodies (OMG, W3C, ietf, ISO, …) are *slow*
- Strong – and conflicting – interest of major industry players (Divergence)

Future-Proof Software-Systems [Part 3B]

http://www.ubizoo.de

Which are today's engineering modeling solutions?

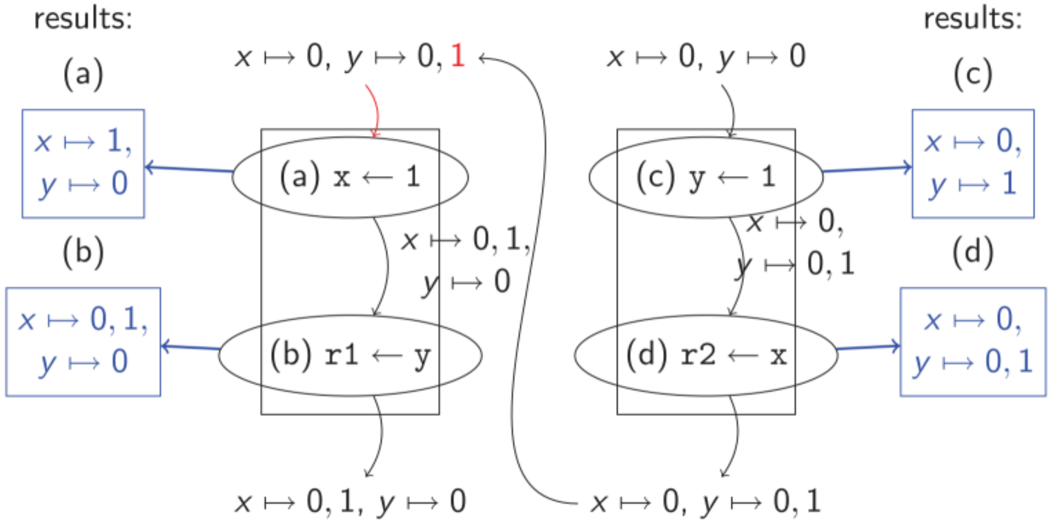
Mature and in wide use:

- ✓ Domain Models
- ✓ Business Object Models
- ✓ Web-Standards (WSDL, …)
- ✓ OCL
- ✓ Ontologies (OWL-DL)
- ✓ UML, SysML + Profiles
- ✓ State machines
- ✓ Timed automata
- ✓ Simulink Models
- ✓ ERD for Databases

Emerging and in selected use:

- ✓ Domain Specific Languages
- ✓ Contracts (CSLs)
- ✓ (Coloured) Petri Nets
- ✓ Annotated, directed hypergraphs
- ✓ Graph rewriting

Model Checking & Verification



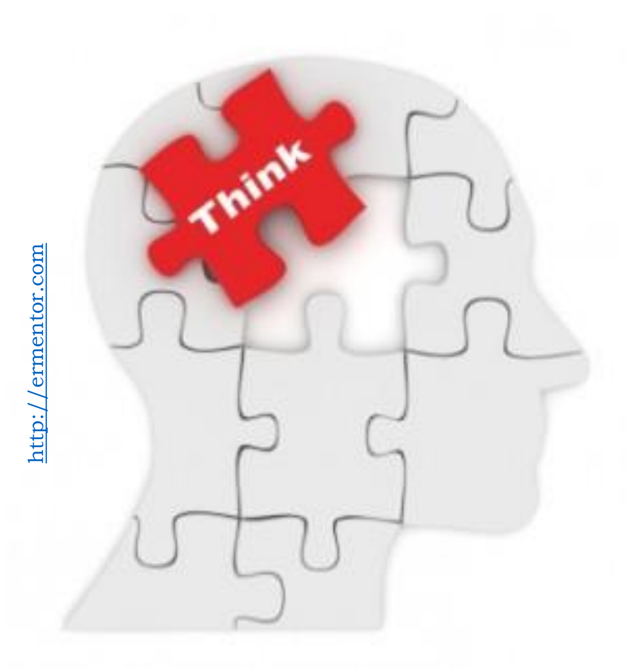
http://www.cprover.org/wmm/

A formalized model based on a *formal logical foundation* allows automatic verification of:

- Syntactical correctness
- Semantic correctness
- Behavioural correctness

Model Quality

Future-Proof Software-Systems [Part 3B]



http://ermentor.com

A formalized model based on a *formal logical foundation* allows reasoning:

- extracting implicit knowledge (reasoning)
- deciding statements (true/false)

Example: Reasoning in an Ontology

Reasoning: From the *explicitly* formulated knowledge in an Ontology (= model) *implicit* knowledge is extracted via defined rules

Nontrivial example (http://owl.man.ac.uk/2003/why/latest):

Content of the ontology:

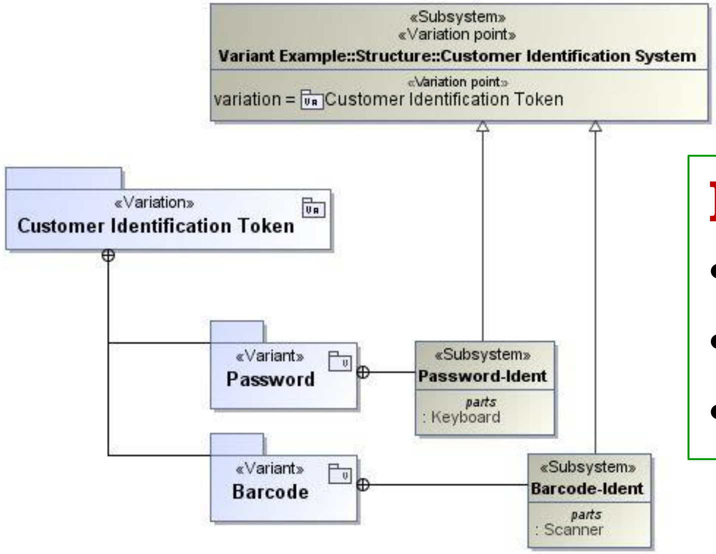
- „Cat owners have cats as pets“ ← Statement in the ontology
- „has pet“ ← Subproperty of „loves pet“ (Statement in the ontology)

Reasoning Conclusion:

- „Cat owners love their cats“

→ deduction
→ checking

A view into the future:



http://model-based-systems-engineering.com

Model:
- Structure
- Behaviour
- Constraints

Automatic Code Generation

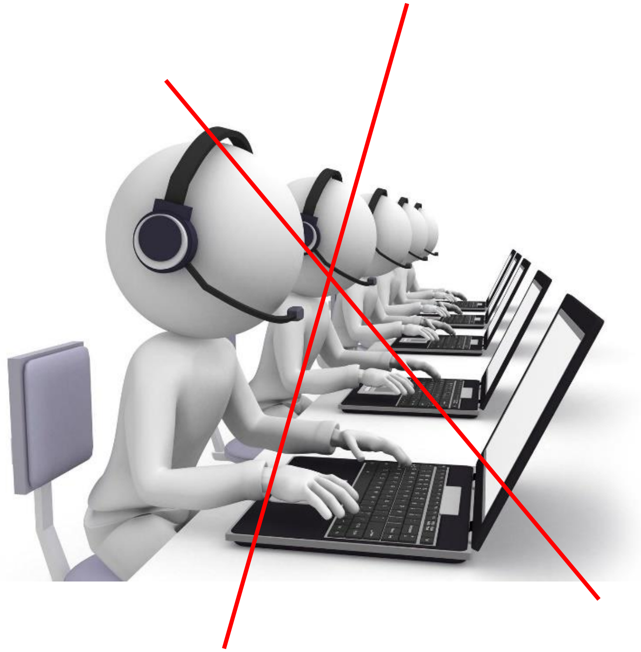
```
    long t1 = (new Date()).getTime();
    double q1 = Math.random();
    byte[] protected1 = Protection.mak
    long t2 = (new Date()).getTime();
    double q2 = Math.random();
    byte[] protected2 = Protection.
    out.writeUTF(user);
    out.writeInt(protected1.lengt
    out.write(protected2);
    out.flush();
}

public static void main(St
    String host = args[0];
    int port = 7999;
    String user = "John";
    String password = "Sh
    Socket s = new Socke

    Client client = new
    client.sendAuthent
```

http://www.i-consortium.com

Code:
- executable
- checked
- ⇒ framework



Model-based System Engineering

Engineering Solutions

Modeling of IT-Systems: **Engineering Solutions**

Which instruments can we use in today's SW-engineering work?

http://www.choicebond.com

Future-Proof Software-Systems [Part 3B]

http://www.ubizoo.de

Which are today‘s engineering modeling solutions?

Mature and in wide use:

- ✓ Domain Models
- ✓ Business Object Models
- ✓ Web-Standards (WSDL, …)
- ✓ OCL
- ✓ Ontologies (OWL-DL)
- ✓ UML, SysML + Profiles
- ✓ State machines
- ✓ Timed automata
- ✓ Simulink Models
- ✓ ERD for Databases

Emerging and in selected use:

- ✓ Domain Specific Languages
- ✓ Contracts (CSLs)
- ✓ (Coloured) Petri Nets
- ✓ Annotated, directed hypergraphs
- ✓ Graph rewriting
- ✓ Role-based modeling (RoSI)

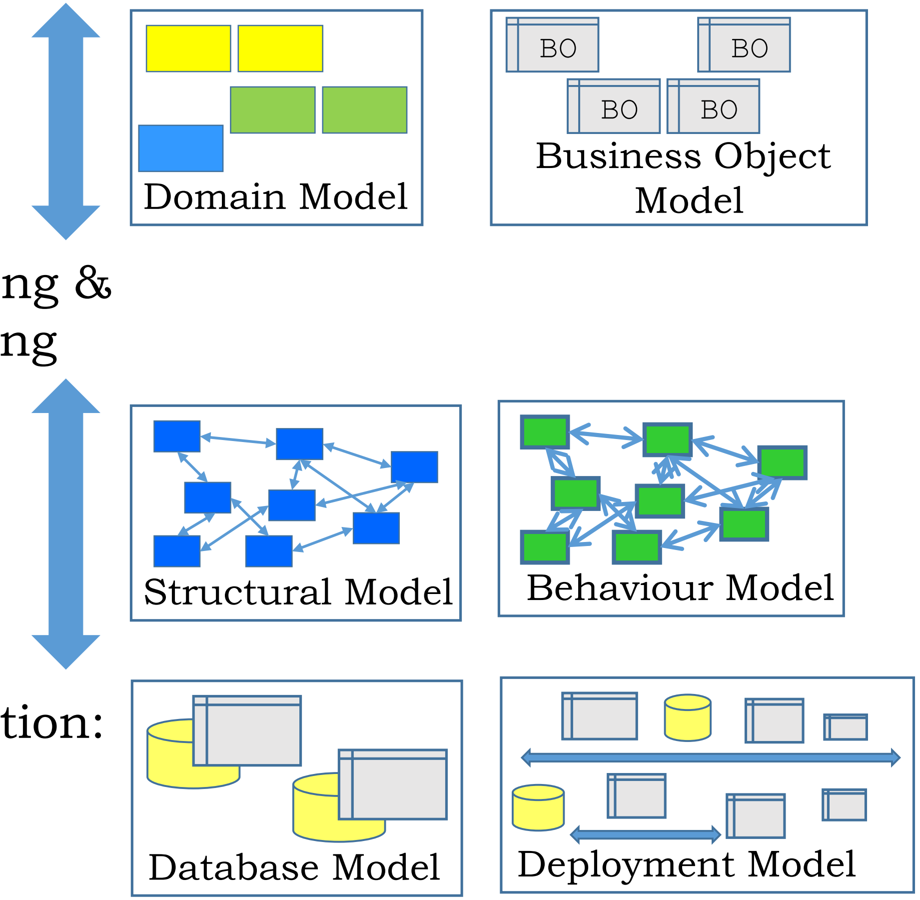
Waiting in the trenches:

- ✓ «Z»-Language
- ✓ «Event-B» Language
- ✓ Certified Code generators
- ✓ Correctness provers

Modeling of IT-Systems: Engineering Solutions

www.123rf.com

Stakeholders:
Business People, …

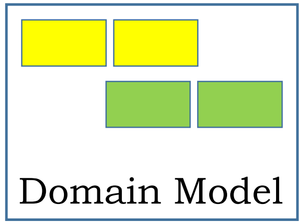


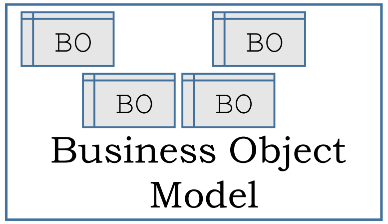
Architecting &
Engineering

Implementation:
SW-People

http://creatica.com

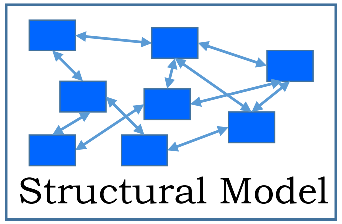
Modeling of IT-Systems: **Engineering Solutions**

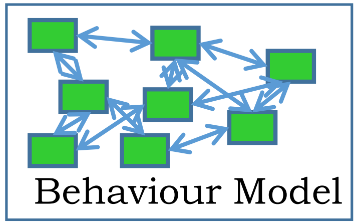

Domain Model



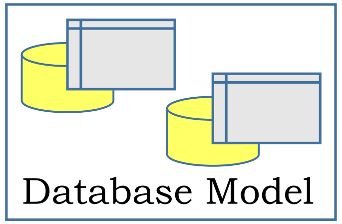
Business Object Model

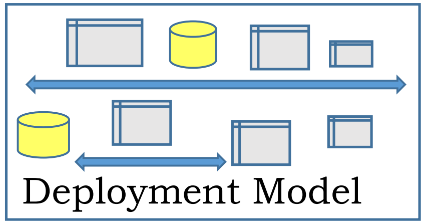
Domain-Model,
Business Object Model
Domain Ontology
UML + Profile Model


Structural Model


Behaviour Model

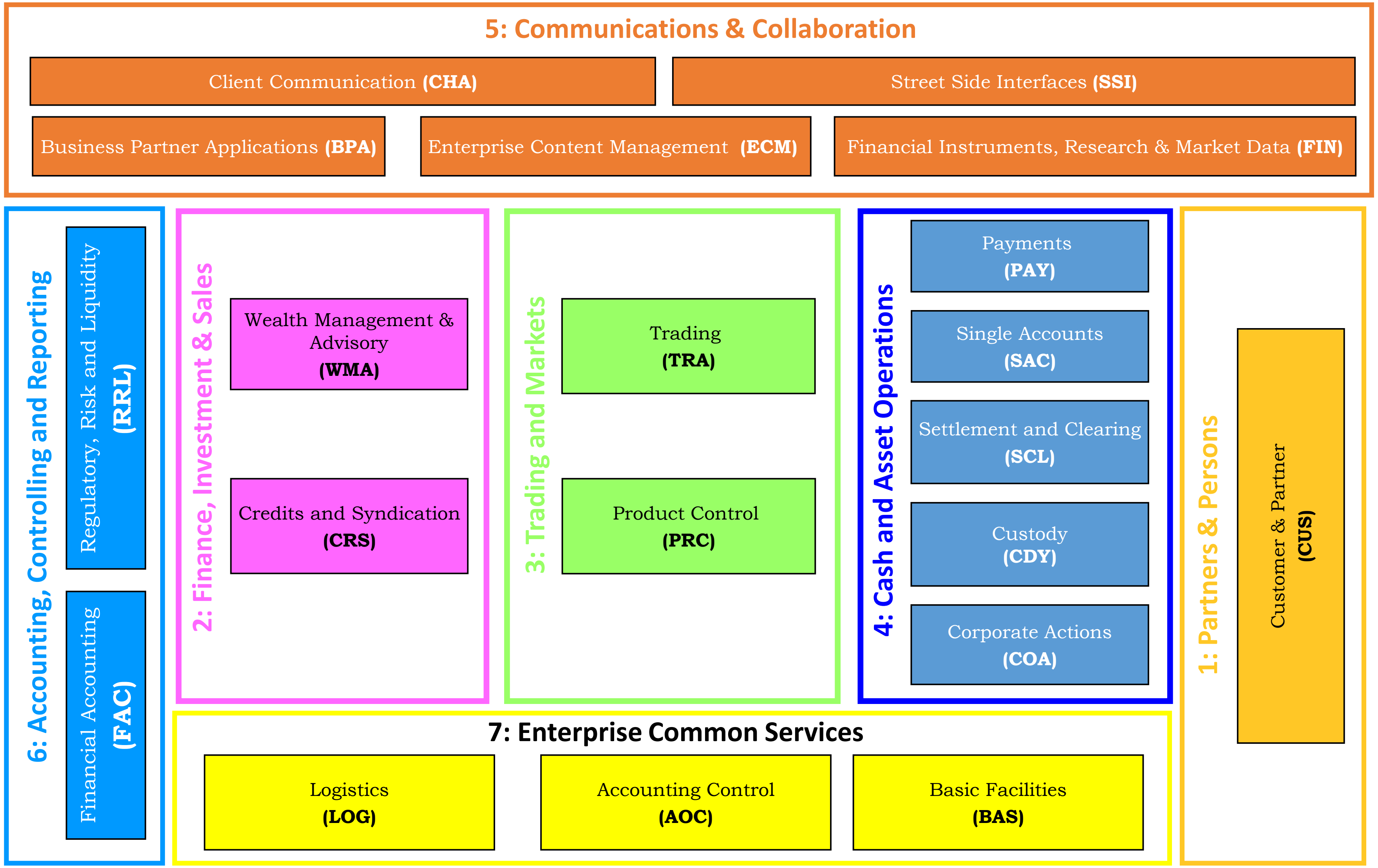
Application Taxonomy
UML + Profile(s) Model
[Interface Contract Model]


Database Model


Deployment Model

Data Dictionary
ERD-Model
Graphs/Petri Nets

Example: Domain Model for a Financial Institution



5: Communications & Collaboration

Client Communication **(CHA)**

Street Side Interfaces **(SSI)**

Business Partner Applications **(BPA)**

Enterprise Content Management **(ECM)**

Financial Instruments, Research & Market Data **(FIN)**

6: Accounting, Controlling and Reporting

Regulatory, Risk and Liquidity **(RRL)**

Financial Accounting **(FAC)**

2: Finance, Investment & Sales

Wealth Management & Advisory **(WMA)**

Credits and Syndication **(CRS)**

3: Trading and Markets

Trading **(TRA)**

Product Control **(PRC)**

4: Cash and Asset Operations

Payments **(PAY)**

Single Accounts **(SAC)**

Settlement and Clearing **(SCL)**

Custody **(CDY)**

Corporate Actions **(COA)**

1: Partners & Persons

Customer & Partner **(CUS)**

7: Enterprise Common Services

Logistics **(LOG)**

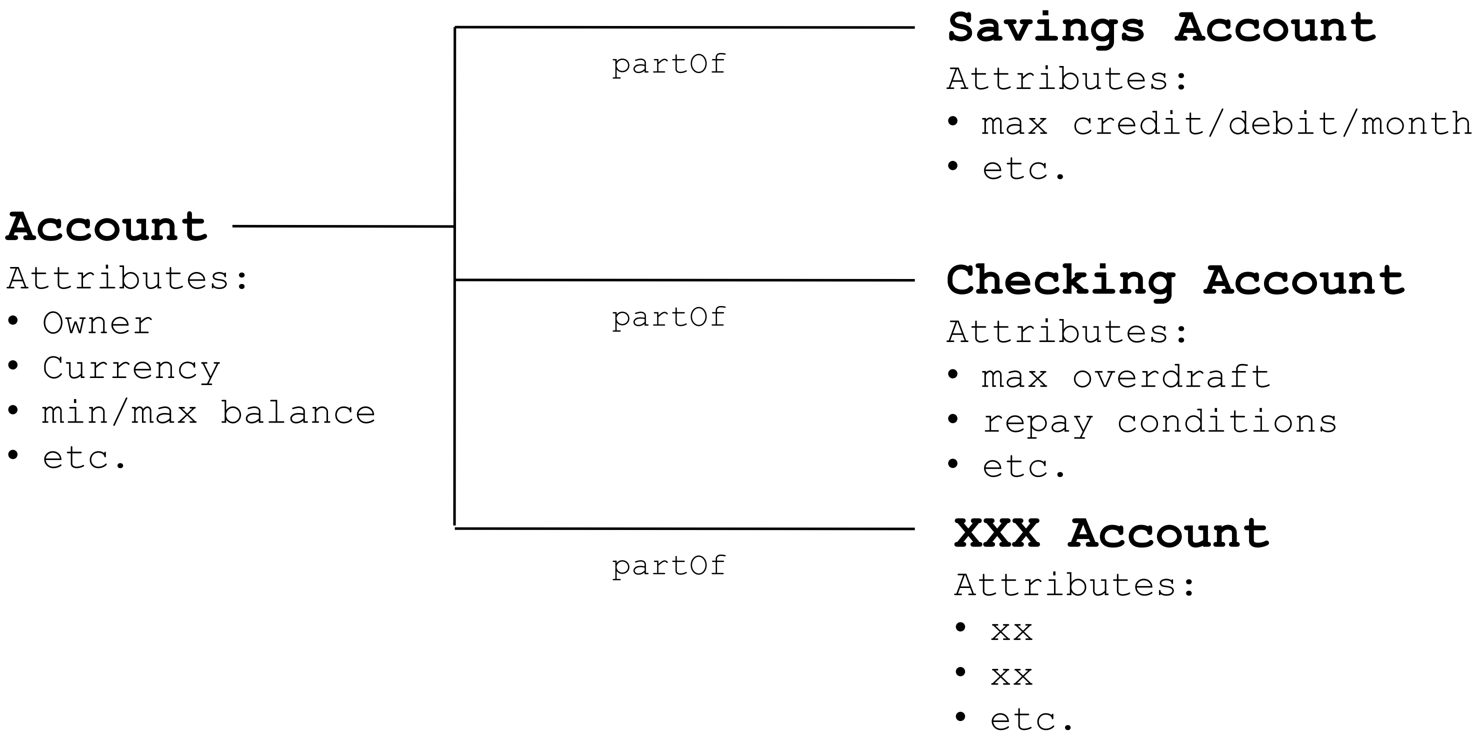
Accounting Control **(AOC)**

Basic Facilities **(BAS)**

Modeling of IT-Systems: Engineering Solutions

Domain Ontology

Single Accounts
(SAC)



Account

Attributes:
- Owner
- Currency
- min/max balance
- etc.

partOf

Savings Account

Attributes:
- max credit/debit/month
- etc.

partOf

Checking Account

Attributes:
- max overdraft
- repay conditions
- etc.

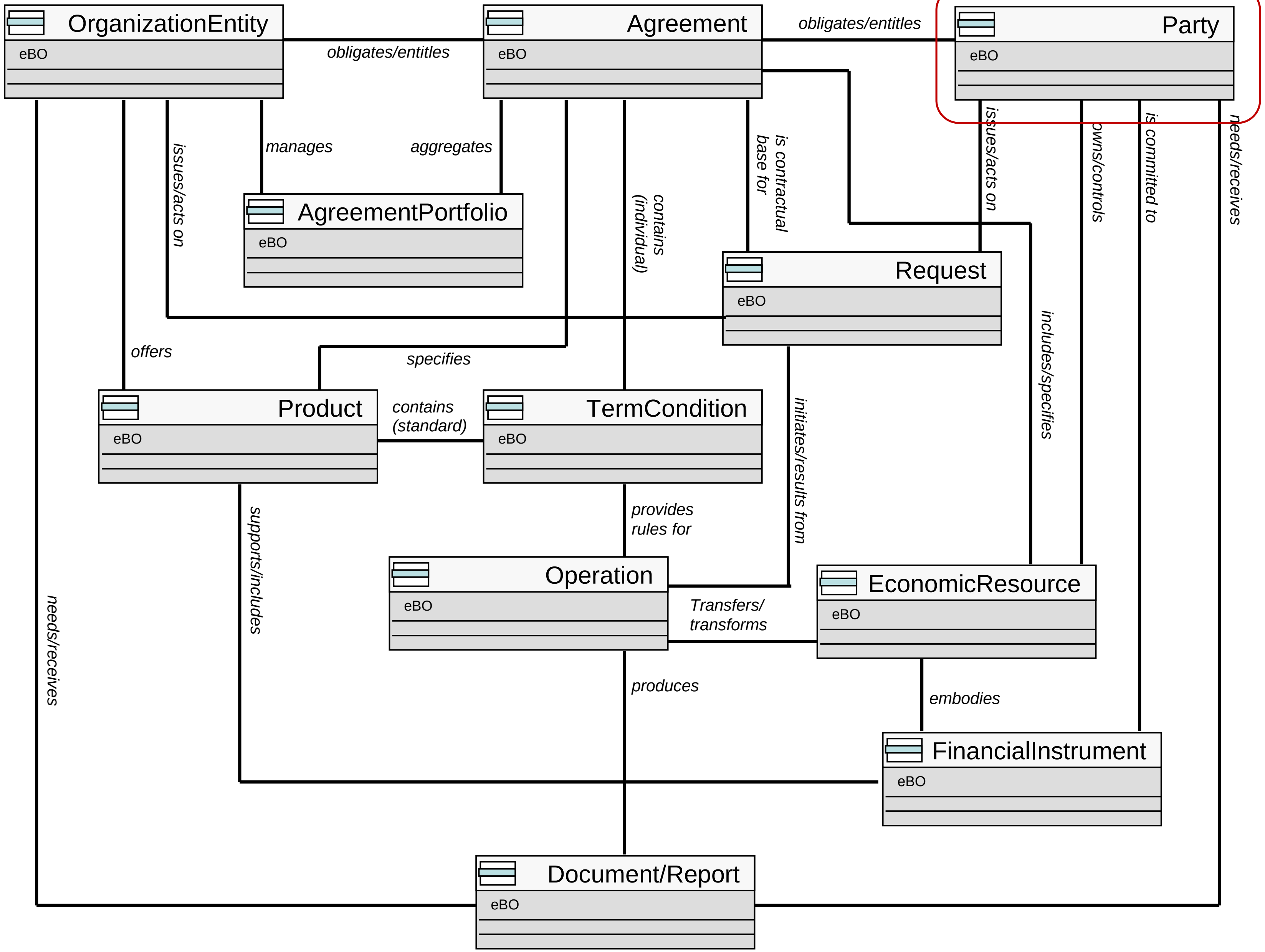
partOf

XXX Account

Attributes:
- xx
- xx
- etc.

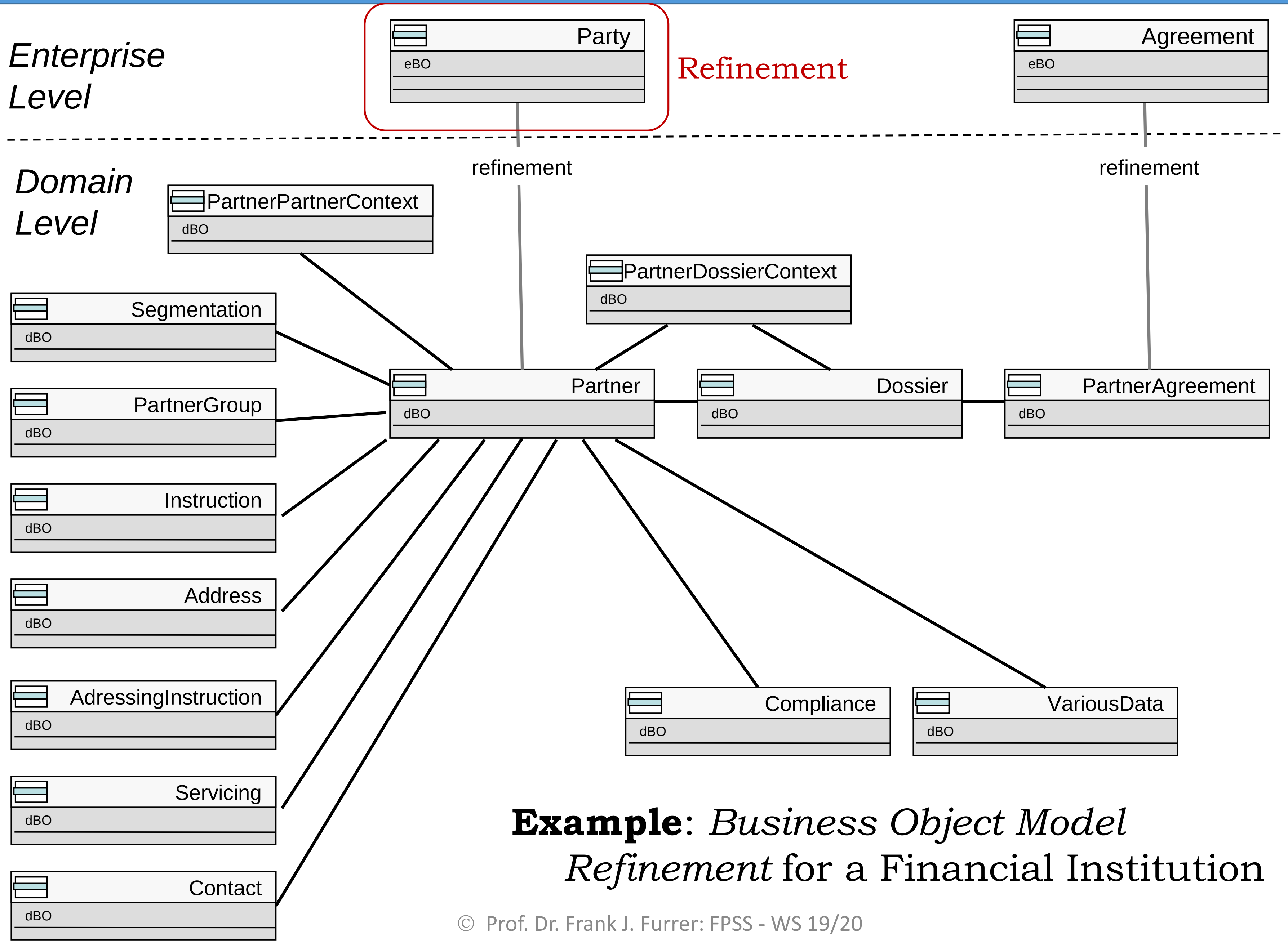
Future-Proof Software-Systems [Part 3B]

Top Level Business Object Model (*Enterprise Level*)



Refinement

Future-Proof Software-Systems [Part 3B]



Example: *Business Object Model Refinement* for a Financial Institution

Modeling of IT-Systems: **Engineering Solutions**

Mature and in wide use:

- Domain Models √
- Business Object Models √
- Web-Standards (WSDL, …)
- OCL
- Ontologies (OWL-DL) √
- UML, SysML + Profiles
- State machines
- Timed automata
- Simulink Models
- ERD for Databases

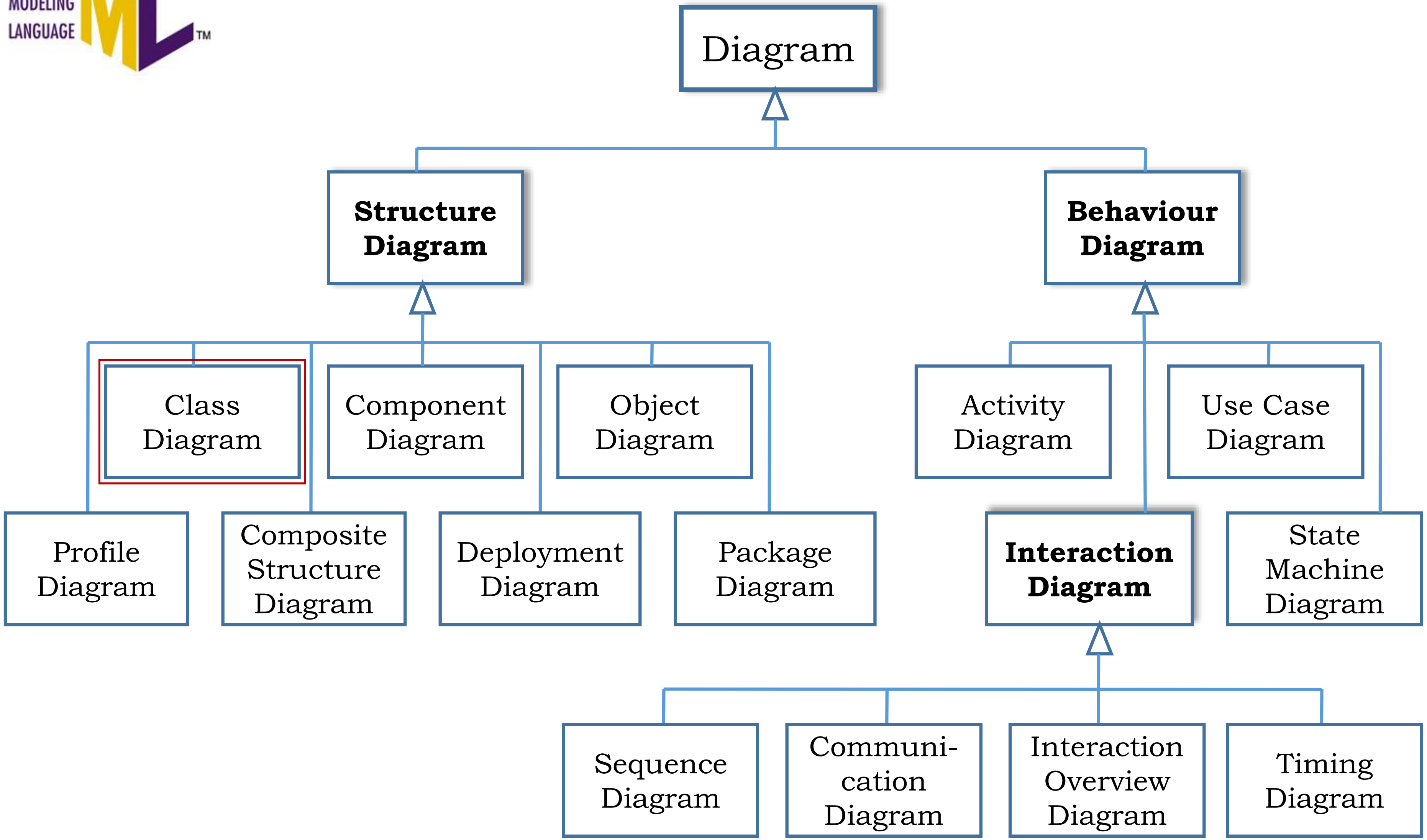
The Object Management Group (OMG®) is an international computer industry standards consortium

Founded in 1989, OMG standards are driven by vendors, end-users, academic institutions and government agencies

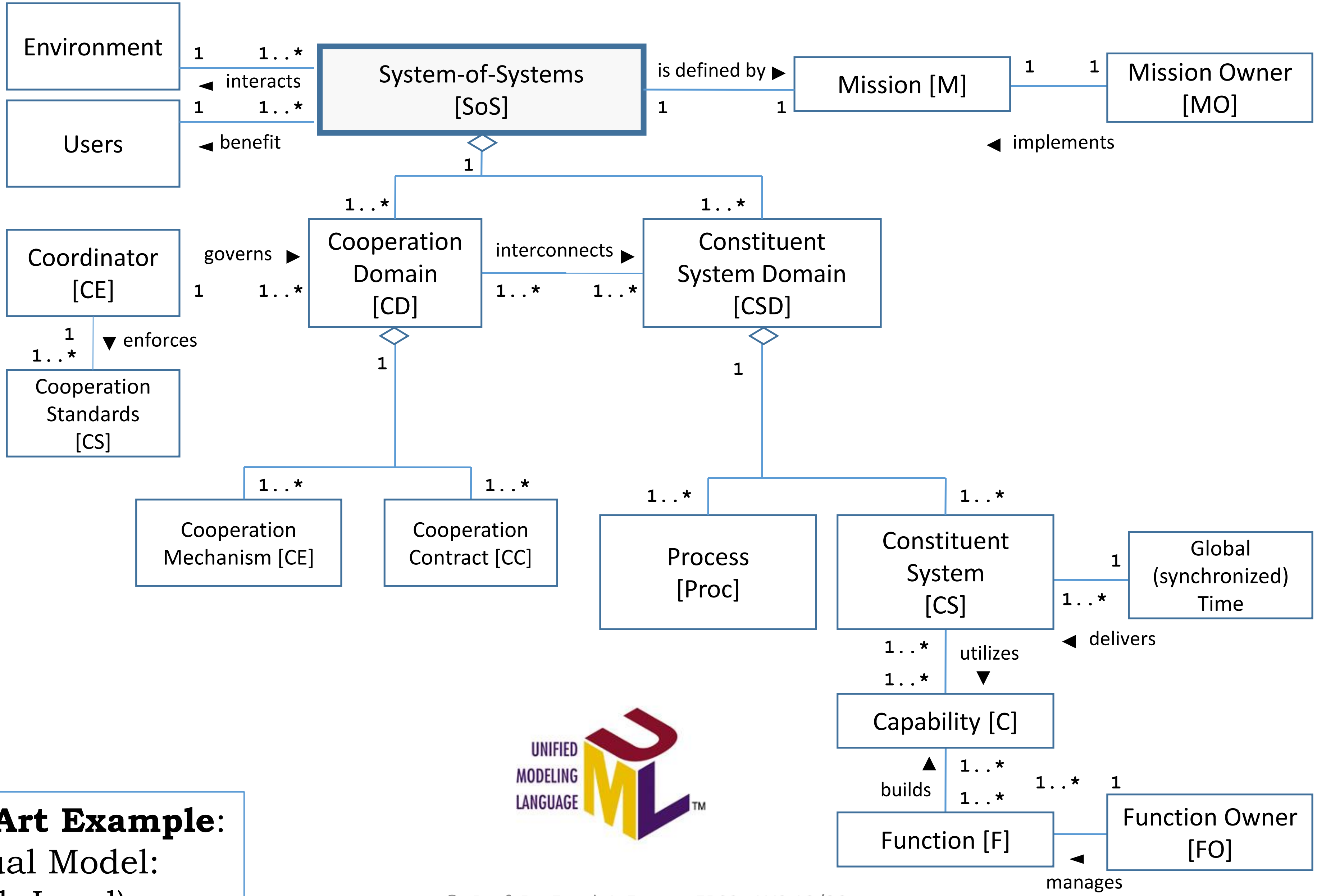
OMG's modeling standards, including the Unified Modeling Language (UML) and Model Driven Architecture (MDA), enable powerful visual design, execution and maintenance of software and other processes

http://www.omg.org

Modeling of IT-Systems: Modeling Instruments



Future-Proof Software-Systems [Part 3B]



Environment 1 — ◄ interacts — 1..* System-of-Systems [SoS]

Users 1 — ◄ benefit — 1..* System-of-Systems [SoS]

System-of-Systems [SoS] 1 — is defined by ► — 1 Mission [M]

Mission [M] 1 — ◄ implements — 1 Mission Owner [MO]

System-of-Systems [SoS] 1 — 1..* Cooperation Domain [CD]

System-of-Systems [SoS] 1 — 1..* Constituent System Domain [CSD]

Coordinator [CE] 1 — governs ► — 1..* Cooperation Domain [CD]

Cooperation Domain [CD] 1..* — interconnects ► — 1..* Constituent System Domain [CSD]

Coordinator [CE] 1 — ▼ enforces — 1..* Cooperation Standards [CS]

Cooperation Domain [CD] 1 — 1..* Cooperation Mechanism [CE]

Cooperation Domain [CD] 1 — 1..* Cooperation Contract [CC]

Constituent System Domain [CSD] 1 — 1..* Process [Proc]

Constituent System Domain [CSD] 1 — 1..* Constituent System [CS]

Constituent System [CS] 1..* — ◄ delivers — 1 Global (synchronized) Time

Constituent System [CS] 1..* — utilizes ▼ — 1..* Capability [C]

Capability [C] 1..* — ▲ builds — 1..* Function [F]

Function [F] 1..* — ◄ manages — 1 Function Owner [FO]

State of the Art Example:
SoS Conceptual Model:
Structure (High Level)

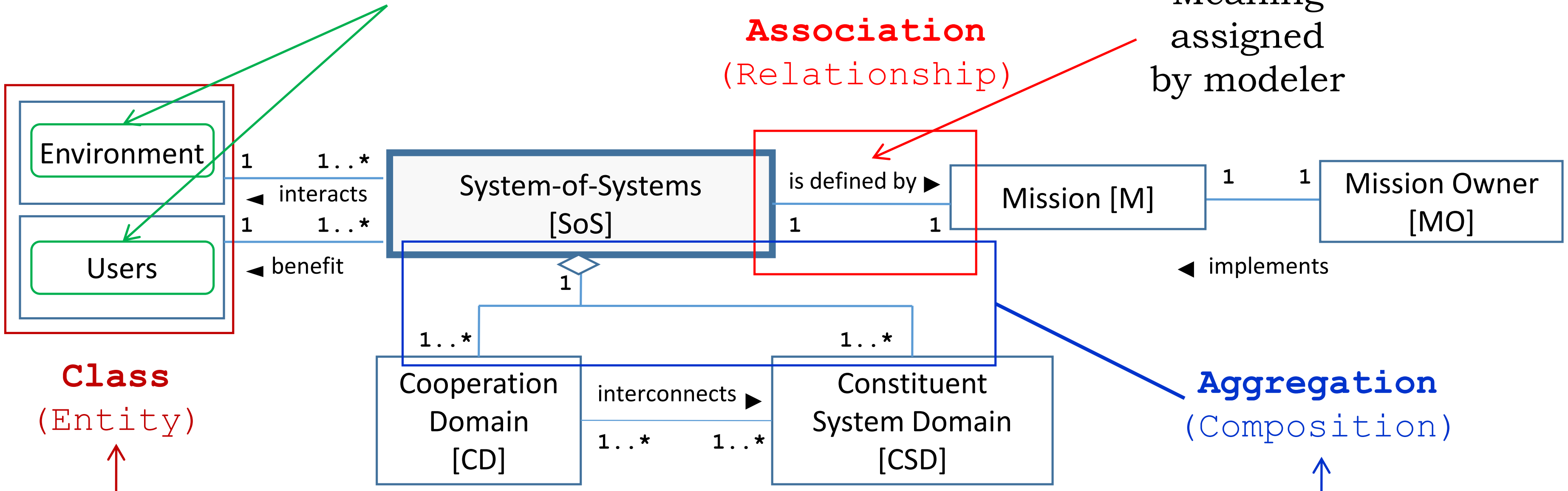
Modeling of IT-Systems: Modeling Instruments

UML and Semantics

Meaning
assigned
by modeler

Association
(Relationship)

Meaning
assigned
by modeler



Class
(Entity)

„meaningless“
container

Aggregation
(Composition)

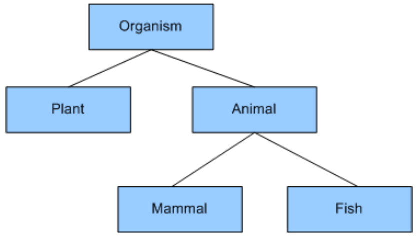
Meaning
defined
in UML

UNIFIED MODELING LANGUAGE ™

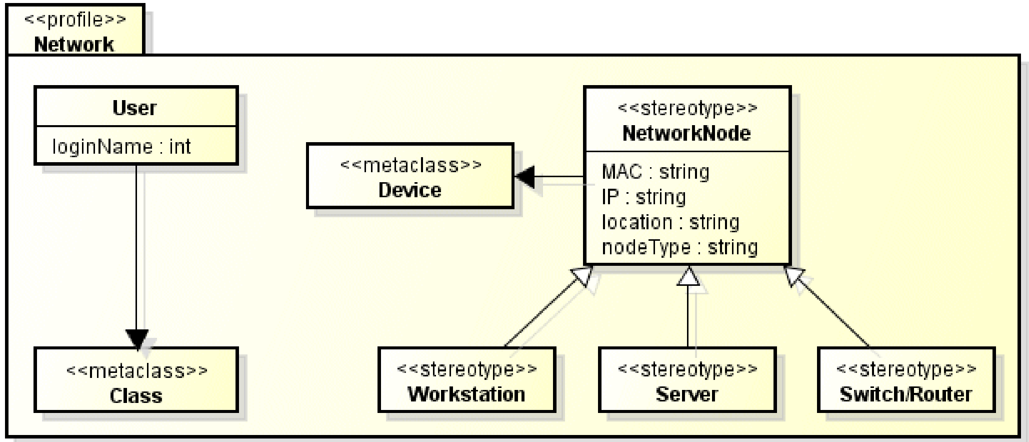
Modeling of IT-Systems: Modeling Instruments

UML and Semantics

How can we define semantics (**meaning**) in UML diagrams?



a) By building an **ontology** based on a **domain-model** which formally defines the meaning of all concepts (classes), relationships (associations) and their attributes



http://www.technologyuk.net

b) By defining an **UML-profile**, extending UML with a domain-specific vocabulary (including relationships)

Modeling of IT-Systems: Engineering Solutions

UML and Semantics

DEFINITIONS

Definition:

An **UML-profile** allows UML to model systems intended for use in a particular domain (for example medicine, financial services or specialized engineering fields, such as safety-critical embedded systems or systems-of-systems).

A profile extends the UML to allow user-defined *stereotypes*, *meta-attributes*, and *constraints*. The vocabulary of the UML is thus extended with a domain-specific vocabulary that allows more meaningful names to be assigned to model elements.

UML-profiles allow the formalized exchange of domain-knowledge between different users and enforce a standardization of UML models.

Modeling of IT-Systems: Engineering Solutions

UML and Semantics

Modeling Language

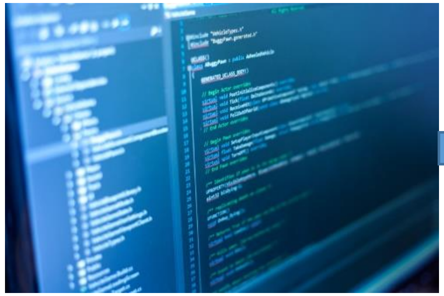
Domain/Application-specific concepts & semantics

Example: Important UML-profiles (Standardized by the OMG)

MARTE (**M**odeling and **A**nalysis of **R**eal-**T**ime and **E**mbedded Systems): MARTE is an UML profile intended for model-based development of real-time and embedded systems

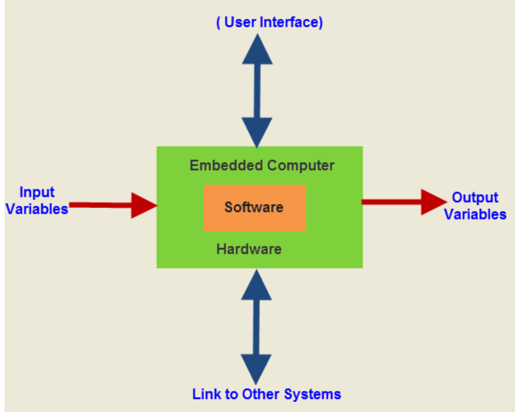
UDMP (**U**nified Profile for **D**oDAF and **M**ODAF **P**rofile): Profile for enterprise and system of systems (SoS) architecture modeling

Future-Proof Software-Systems [Part 3B]



Software

UNIFIED MODELING LANGUAGE



System

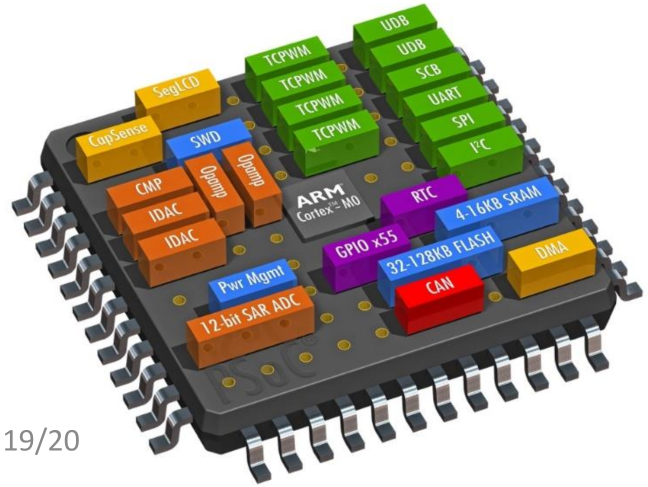
OMG SYSTEMS MODELING LANGUAGE

SysML:
Systems Modeling Language

DEFINITIONS

The Systems Modeling Language (**SysML**) is a general-purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems.

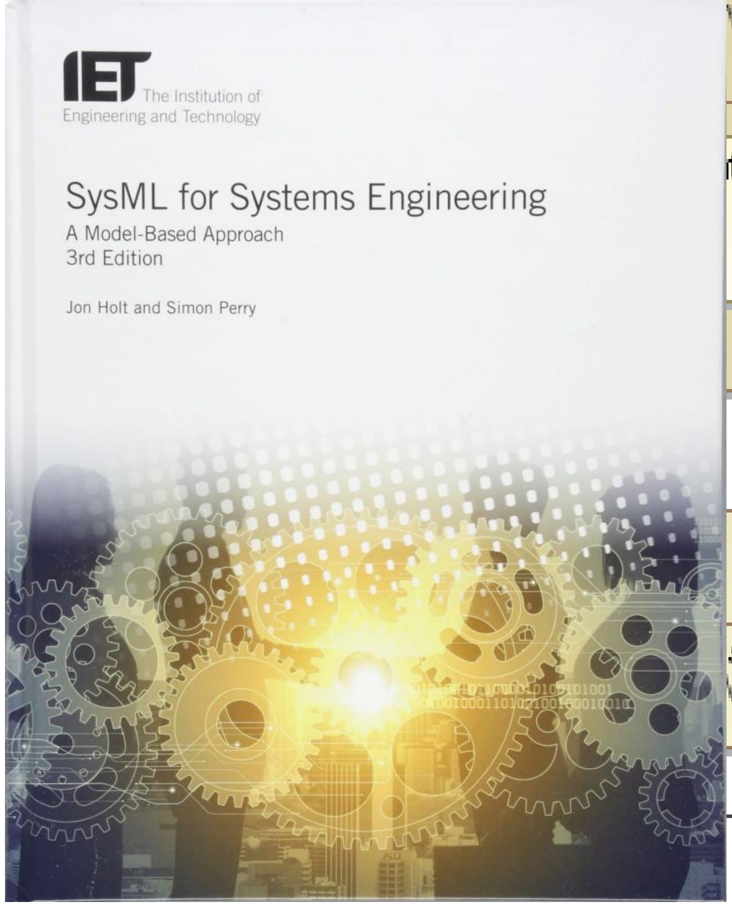
SysML expresses **systems engineering** semantics (interpretations of notations) better than than UML. SysML is smaller and easier to learn than UML. Since SysML removes many software-centric constructs, the overall language is smaller as measured in diagram types (9 vs. 13) and total constructs.



https://www.thegeniusworks.com

Future-Proof Software-Systems [Part 3B]

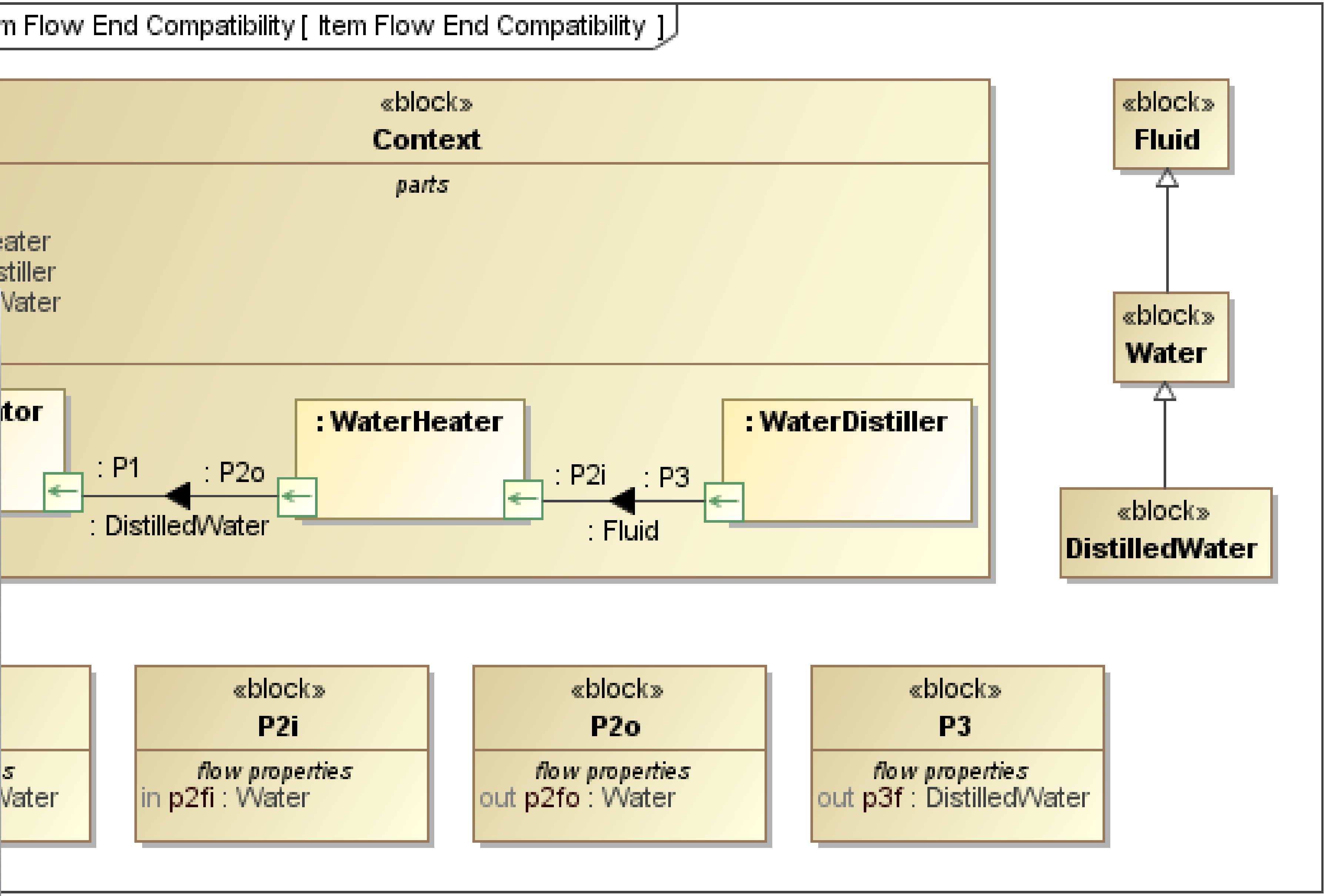
SysML:
Block Definition Diagram
https://docs.nomagic.com



IET The Institution of Engineering and Technology

SysML for Systems Engineering
A Model-Based Approach
3rd Edition

Jon Holt and Simon Perry



Quality of Models

PREMIUM QUALITY

The **quality of a model** can be expressed as follows:

Syntactic Quality: The model does not violate any syntactic rules of the modeling language

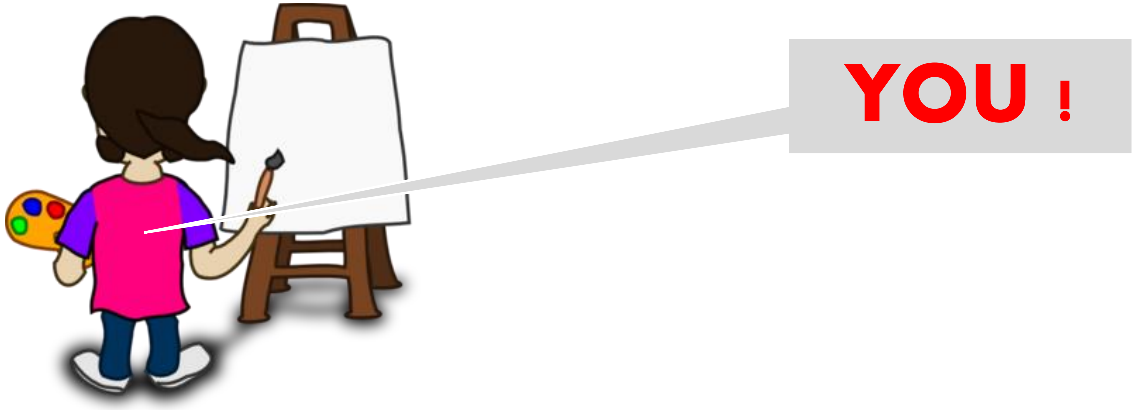
Semantic Quality: All the elements in the model have a unambigously specified and agreed meaning

Pragmatic Quality: The interpretation by the human stakeholders is correct with respect to what is meant to be expressed by the model. The interpretation by the tool(s) is correct with respect to the intended functionality

Social Quality: The model has sufficient agreement by all stakeholders

Completeness Quality: The model contains sufficient information to fullfill its role „clarity, committment, communication, control“ for the intended goal

The System Modeler



A good *system modeller* needs:

- A strong theoretical background of the choosen modeling instrument
- An excellent fluency in the modelling language and the modeling tools
- Good skills to extract the knowledge from the stakeholders in the domain
- Mediation skills to reach agreement for the model between the stakeholders
- A „touch of art“ – to make simple and beautiful, rich models
- A good and reliable memory to have the full model present at all times

The Future: Contract-Based Systems Engineering

http://www.yellowjacketdisposal.com

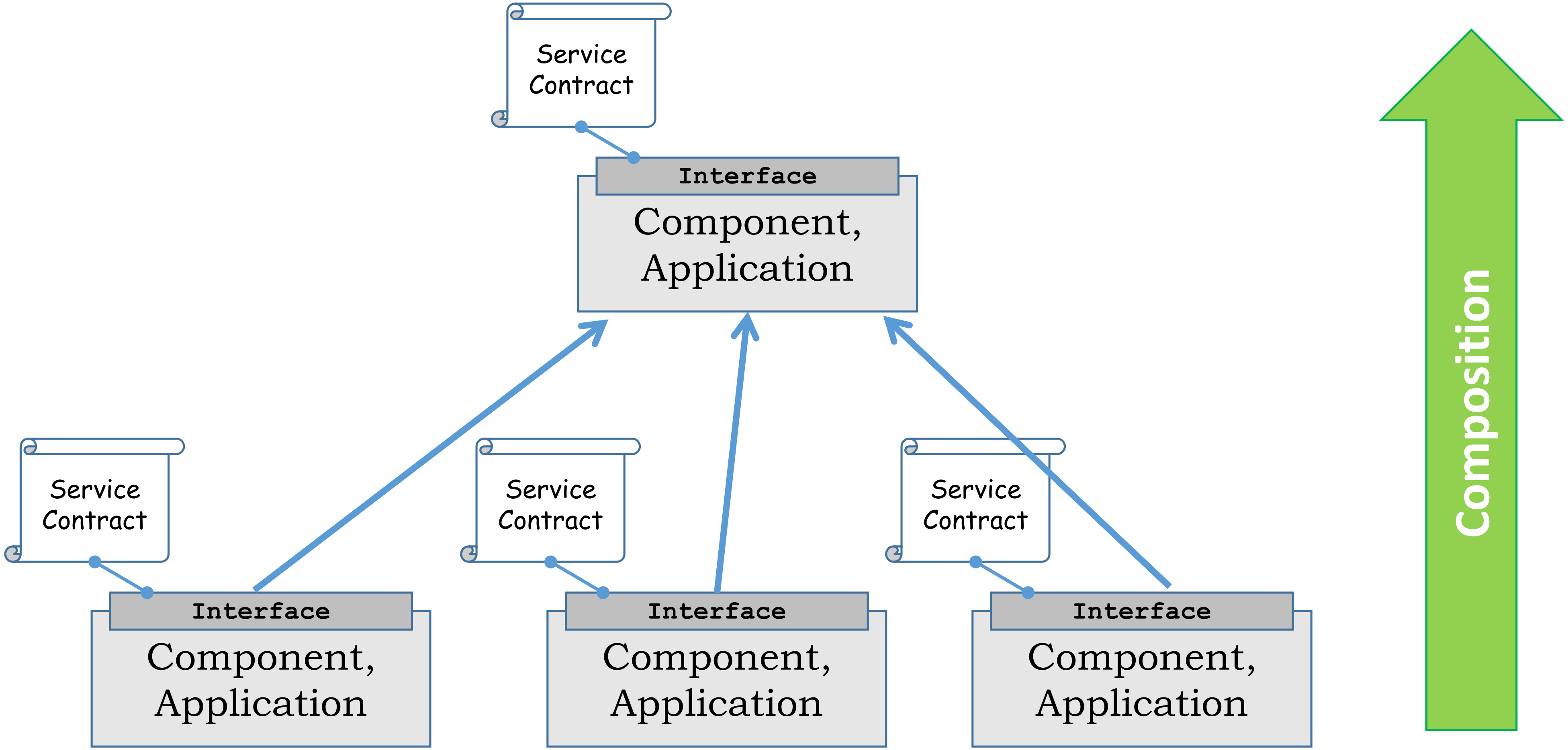
DEFINITIONS

Definition:

Contracts are formal, binding agreements between a service provider and a service consumer.

They cover the *functional* interface specifications (functionality and data), the *non-functional* properties (timing, security etc.) and in some cases also the commercial conditions (terms of use, guarantees, liability etc.)

The Future: Contract-Based Systems Engineering



www.publicdomainpictures.net

The Future: Contract-Based Systems Engineering

Example: Emergency Services

"All FireStation host at least one Fire Fighting Car"

SoS.itsFireStations->forAll(fstation | fstation.hostedFireFightingCars->size() >= 1)

"Any district cannot have more than 1 fire station, except if all districts have at least 1"

SoS.itsDistricts->exists(district | district.containedFireStations->size() > 1) implies
SoS.itsDistricts->forAll(containedFireStations->size() >= 1)

"The fire fighting cars hosted by a fire station shall be used all simultaneously at least once in 6 months"

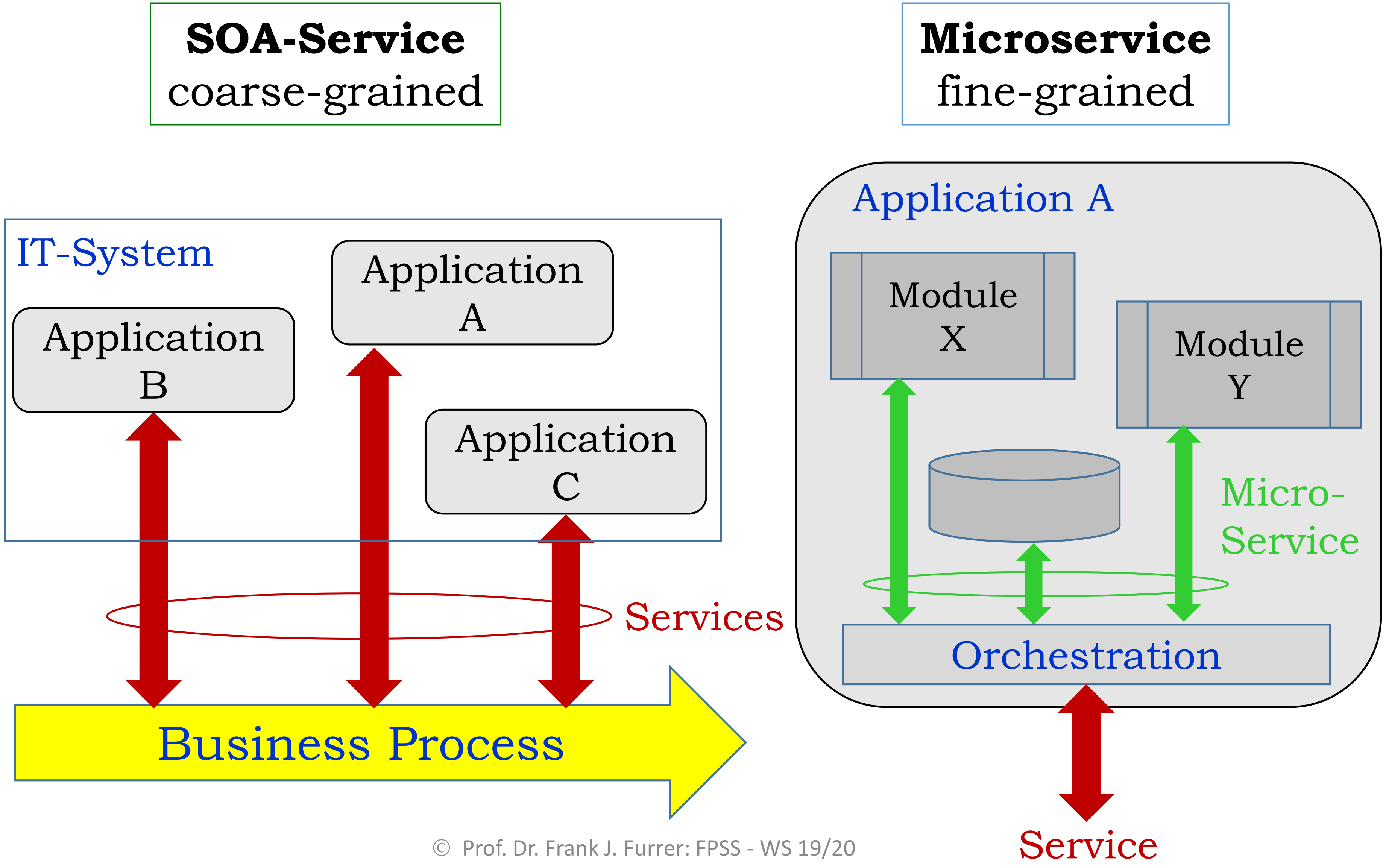
SoS.itsFireStations->forAll(fireStation |
Whenever [fireStation.hostedFireFightingCars->exists(isAtFireStation)] **occurs,**
[fireStation.hostedFireFightingCars->forall(**isAtFireStation** = false)]
occurs within [6 months])

---

red = identifiers from the model / blue = OCL constraints / **bold black** = temporal operators

https://www.danse-ip.eu/home/images/deliverables/DANSE_D6.3.2_GCSL.pdf

Granularity (Size) of Services



Micro-Services

DEFINITIONS

„The *microservice architectural* style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.“

Martin Fowler:
http://martinfowler.com/articles/microservices.html

Microservices have emerged from:

- Domain-driven design
- Continuous delivery
- On-demand virtualization
- Infrastructure automation
- Small autonomous teams
- Systems at scale

Sam Newman: **Building Microservices**, 2015

Future-Proof Software-Systems [Part 3B]

https://cdn1.lockerdome.com

«The glamour lies in software»

«The future lies in modeling»

http://www.mortgageorb.com

Future-Proof Software-Systems [Part 3B]

A11

Architecture Principle A11:

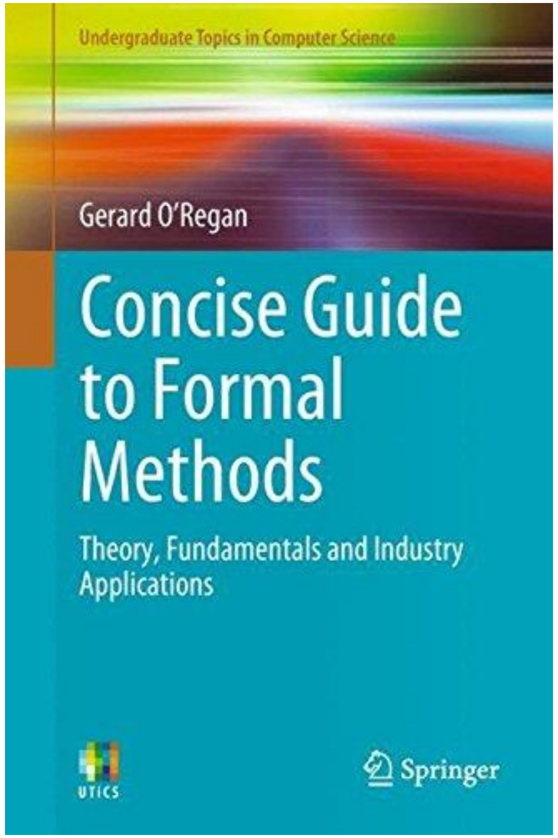
Formal Modeling

1. Model as many parts of your IT-system as possible (organization & skills contraints?)
2. Use the highest possible degree of formalization
3. Use industry-standard modeling instruments & tools
4. Treat models as a long-term, highly valuable assets in your company and maintain them in a repository
5. Keep models complete& up-to date

Justification: The 4 C's – **C**larity, **C**ommittment, **C**ommunication and **C**ontrol

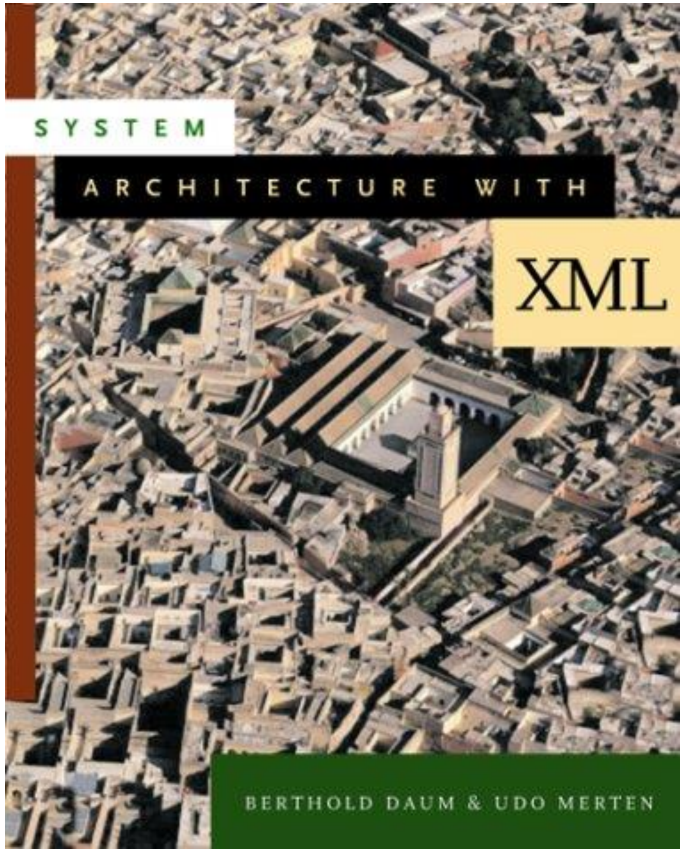
Future-Proof Software-Systems [Part 3A]

Textbook



Gerard O'Regan:
Concise Guide to Formal Methods – *Theory, Fundamentals and Industry Applications*
Springer-Verlag, Germany, 2017. ISBN 978-3-319-64020-4

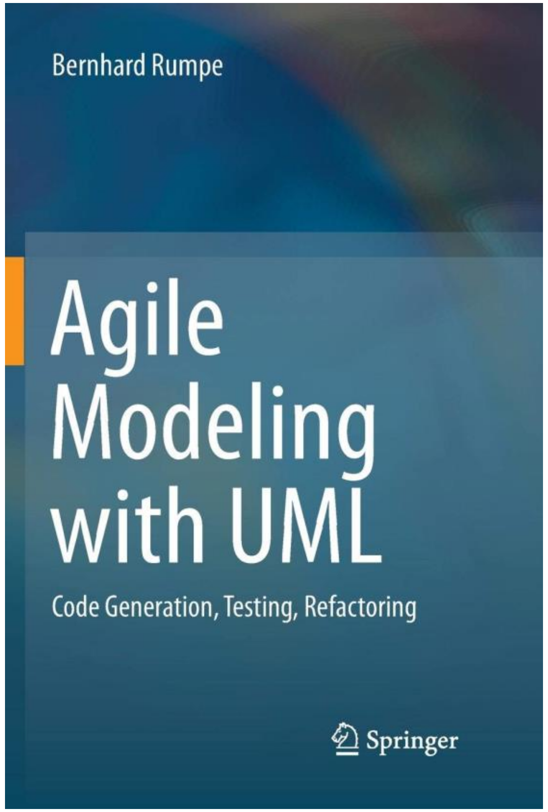
Textbook



Berthold Daum, Udo Merten:
System Architecture with XML
Dpunkt Verlag, Germany, 2002. ISBN 978-3-8986-4196-8

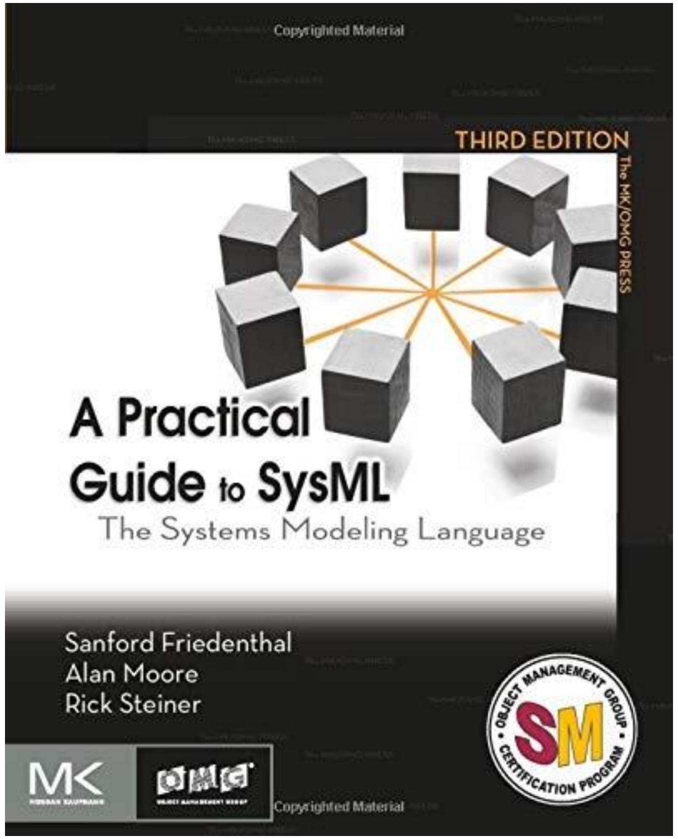
Future-Proof Software-Systems [Part 3A]

Textbook



Bernhard Rumpe:
Agile Modeling with UML – *Code Generation, Testing, Refactoring*
Springer International Publishing, Cham, Switzeralnd, 2017. ISBN 978-3-319-86494-5

Textbook



Sanford Friedenthal, Alan Moore, Rick Steiner:
A Practical Guide to SysML –*The Systems Modeling Language*
Morgan Kaufmann Publishing (OMG Press), 3rd edition, 2014. ISBN 978-0-128-00202-5

Future-Proof Software-Systems [Part 3B]

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

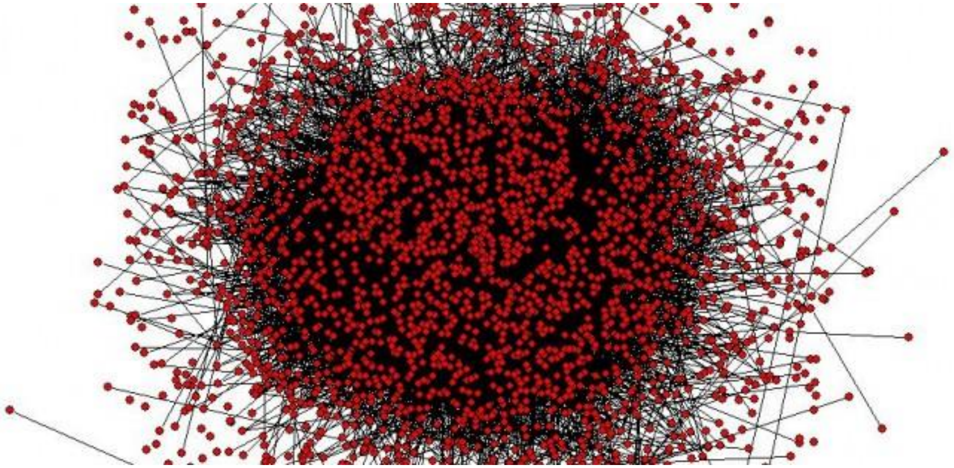
A12

Architecture Principle A12:

Complexity and Simplification

Complexity

- Biology
- Sociology
- Astronomy
- Physics
- …
- Information Technology (IT)



http://blog.digital.telefonica.com

“Complexity is that property of an IT-system which makes it difficult to formulate its overall behaviour, even when given complete information about its parts and their relationships“

DEFINITIONS

Complexity = (IT-) Risk

Example: U.S. FAA Air Traffic Control System

http://www.informationweek.com/664/64iufaa.htm

1995: The FAA (US Federal Aviation Agency) admits the colossal modernization failure of the Advanced Automation System (AAS). That effort took *16 years* of effort and cost taxpayers *$23 billion*

http://clarioncontentmedia.com

"FAA did not recognize the *technical complexity* of the effort, realistically estimate the resources required, adequately oversee its contractors' activities, or effectively control system requirements"

Complexity = (IT-) Risk

good

- Complexity makes large, useful systems possible
- It forces us to develop science for dealing with complexity
- it is a highly interesting and fruitful area of research

bad

- It is the single most important reason for disasters in IT
- It makes understanding, explaining and evolving IT-systems very hard
- It may lead to unpredictable (= emergent) behaviour

Complexity must be managed !

- Identify it
- Understand it
- Avoid and reduce it as much as possible

Future-Proof Software-Systems [Part 3B]

Essential complexity

… is the *inherent* complexity of the system to be built.

Essential complexity for a given problem *cannot* be reduced.

It can only be lessened by *simplifying* the requirements for the system extension.

⇒ However, essential complexity can be *managed* and its negative effects can be *minimized* by good architecture

Accidental Complexity

… is introduced in addition to the essential complexity by our development activities or by constraints from our environment.

This is unnecessary and threatening complexity!

⇒ *Avoiding* and *eliminating* accidental complexity is a continuous task in the development process – from requirements to deployment!

Classification of Complexity

Necessary or desired complexity: **Essential** complexity

… is caused by the problem to be solved. Nothing can remove it. Represents the inherent difficulty

http://gregmaciag.typepad.com

Unnecessary or undesired complexity: **Accidental** Complexity

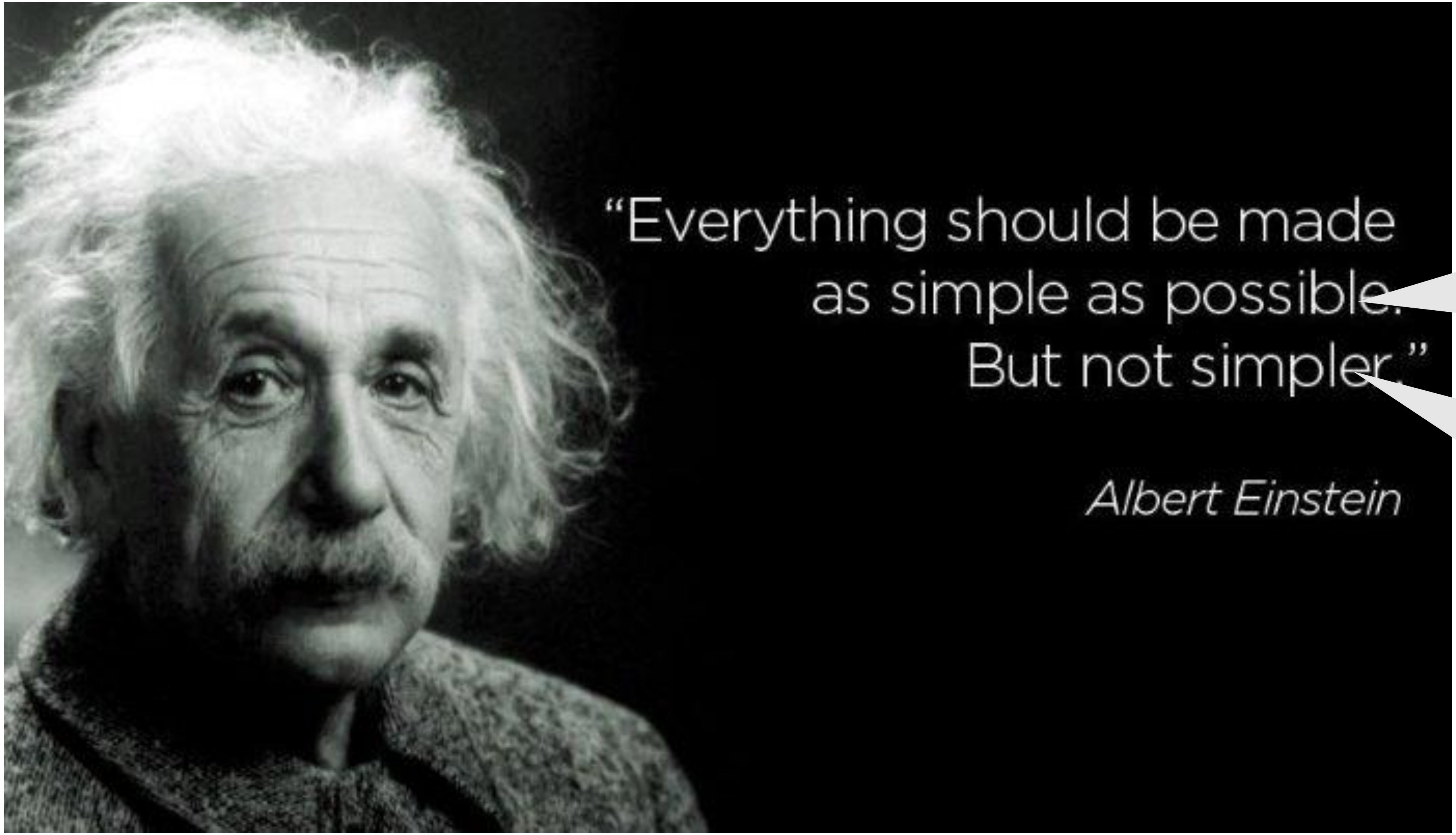
… is caused by solutions that we create on our own or by impacts from our environment

$$[(27/3)/3] - 1$$

$$= 1 + 1$$

$$\boxed{= 2}$$

Future-Proof Software-Systems [Part 3B]



“Everything should be made as simple as possible. But not simpler.”

Albert Einstein

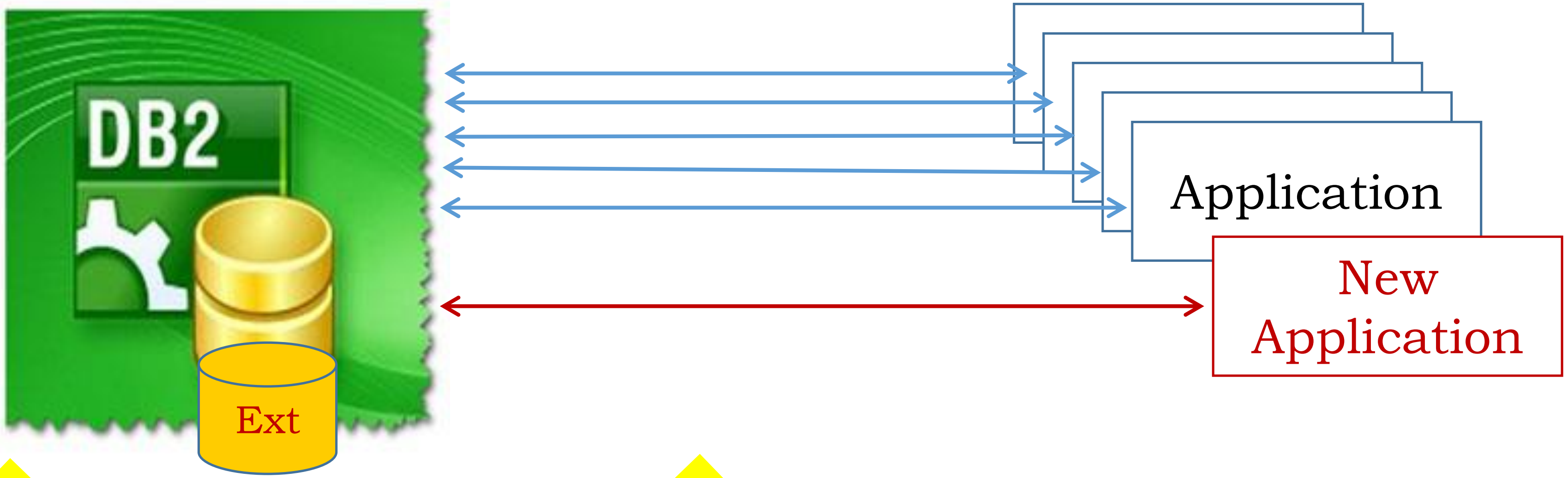
https://media.licdn.com

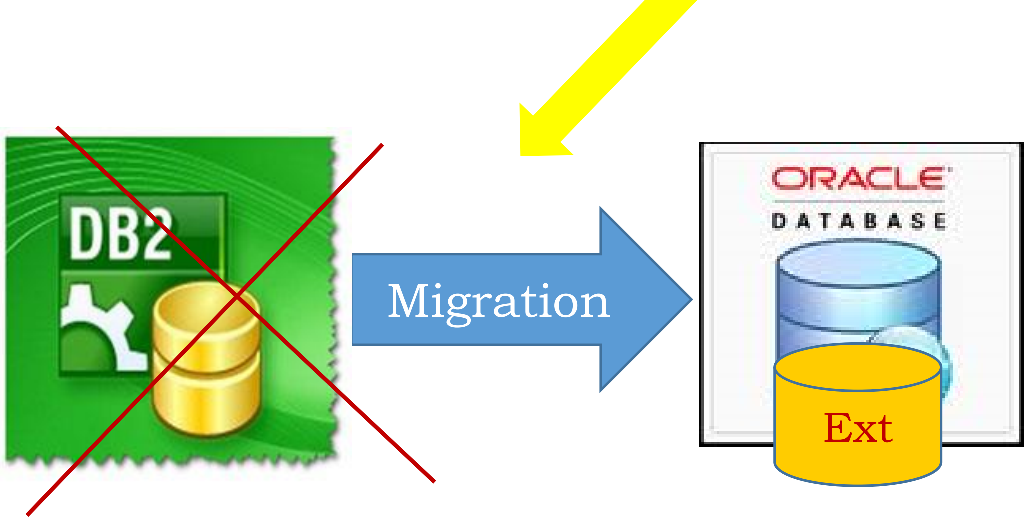
Avoidance of ***accidental complexity***

Minimization of ***essential complexity***

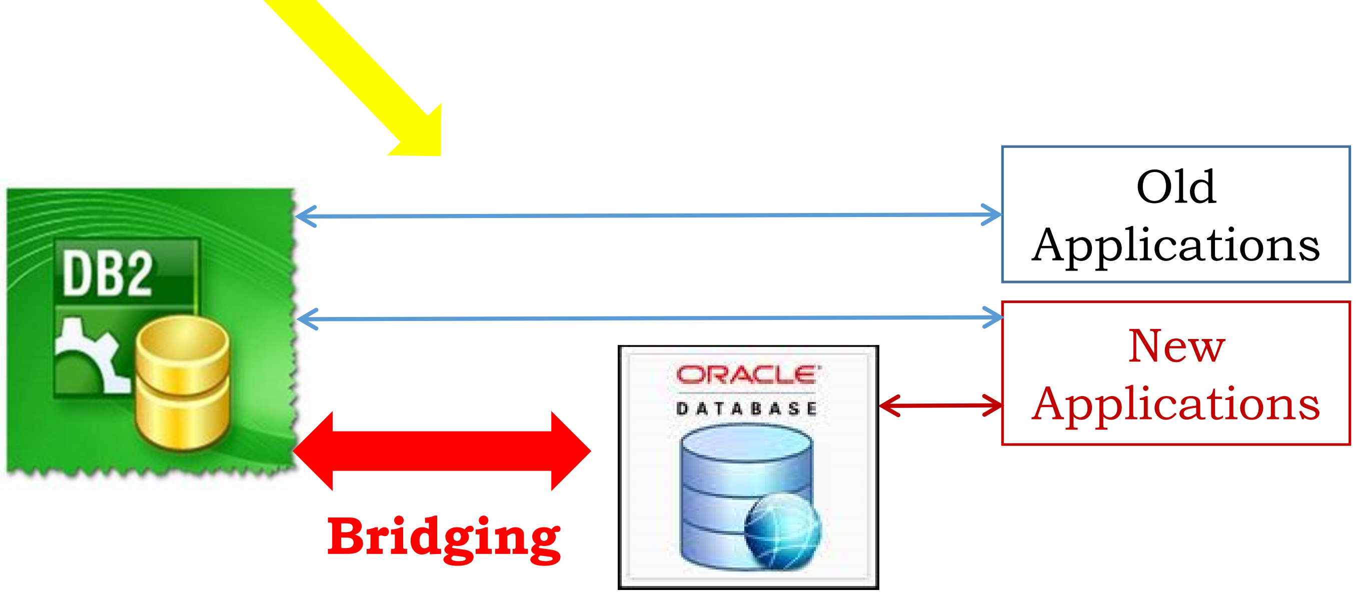
Example:
Database Extension

Problem:
New database standard
= **ORACLE**





Essential complexity: minimized
Accidental complexity: none



Essential complexity: high
Accidental complexity: high

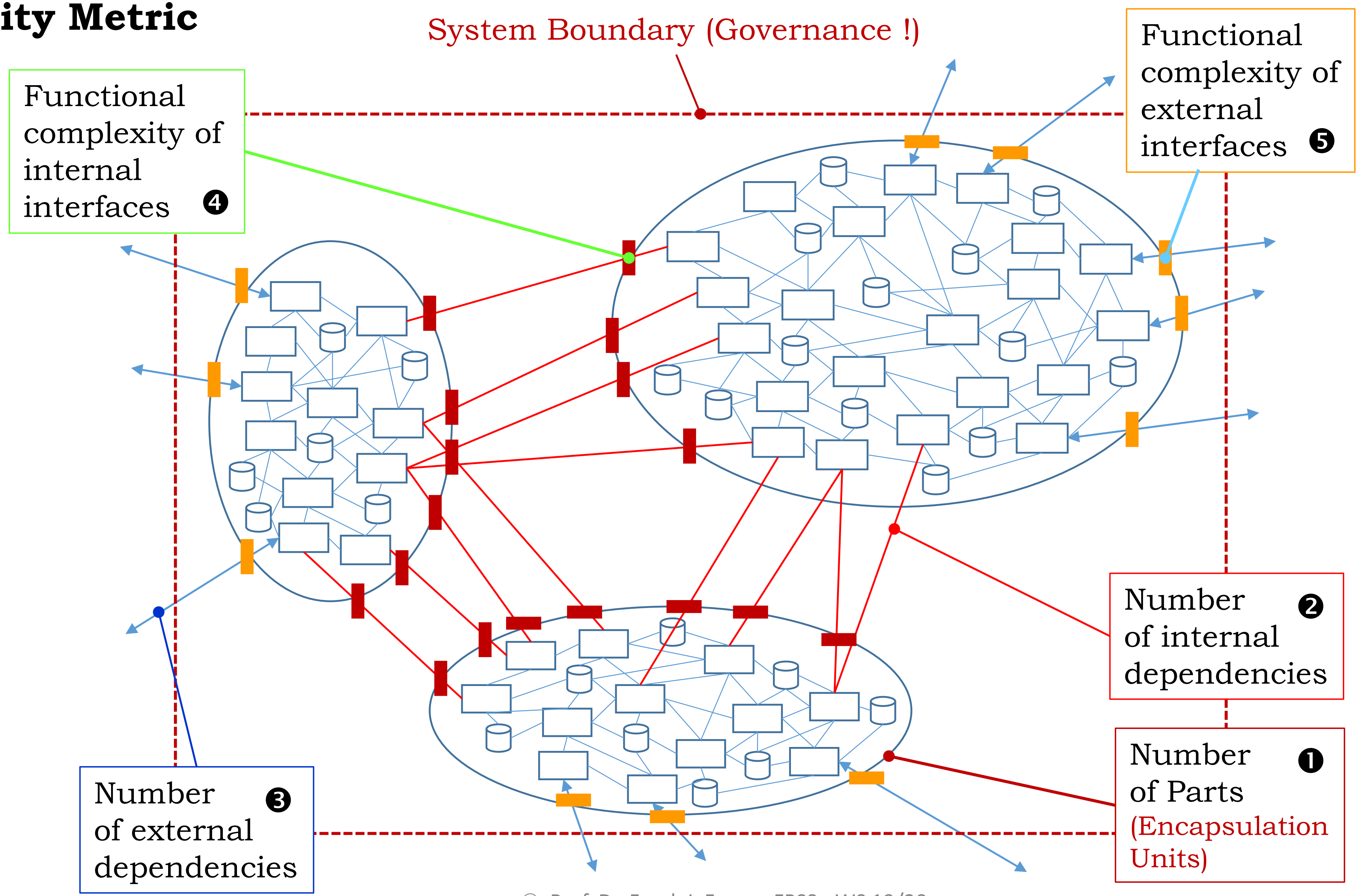
Managing Complexity

Complexity	*Known* (identified) Complexity	*Unknown* (hidden) Complexity
Necessary (desired) Complexity [*Essential* Complexity]	**manage it**	**use it carefully**
Unnecessary (undesired) Complexity [*Accidental* Complexity]	**avoid it** **eliminate it**	DANGER-KEEP OUT DANGER DANGER-KEEP OUT **attack it**

- OS
- DBMS
- TCP/IP Stack
- etc.

- Technical debt
- Architecture erosion

Complexity Metric



System Boundary (Governance !)

Functional complexity of internal interfaces ❹

Functional complexity of external interfaces ❺

Number of internal dependencies ❷

Number of Parts ❶ (Encapsulation Units)

Number of external dependencies ❸

Complexity Contributors:

Contributor	Metric
❶ Number of Parts (*Structural* complexity)	Integer number (N_P)
❷ Number internal dependencies (*Structural* complexity)	Integer number (N_{iD})
❸ Number external dependencies (*Structural* complexity)	Integer number (N_{eD})
❹ *Functional* complexity of internal interfaces	# of FP, UCP (Fi_{Ij})
❺ *Functional* complexity of external interfaces	# of FP, UCP (Fe_{Ik})

Complexity Metric:

$$\boxed{\mathrm{SysCompl} = f[N_P, N_{iD}, N_{eD}, \Sigma Fi_{Ij}, \Sigma Fe_{Ik}]}$$

No distinction between essential complexity and accidental complexity

A number of *complexity metrics* exist in the literature.

However, none of them is satisfactory for engineering system complexity

→ Interesting open research question (PhD-Level) !

Sources accidental IT-complexity:

- **Specifications**: overlaps, duplication
- **Redundancy**: functional, data & interface redundancy
- **Lack of conceptual integrity**: diverging concepts, misunderstandings
- **Disregard of (industry) standards**: technology explosion
- **3rd party software**: forced, incompatible concepts, redundancy
- **Inconsistent housekeeping**: „dead“ code & data
- **Diversity in vertical architectures**: proliferation of solutions
- **Neglected legacy systems**: old technology, out-of-use components

If you don‘t properly manage complexity, it may kill your system

(… most probably: it will)

The nasty ways of complexity:

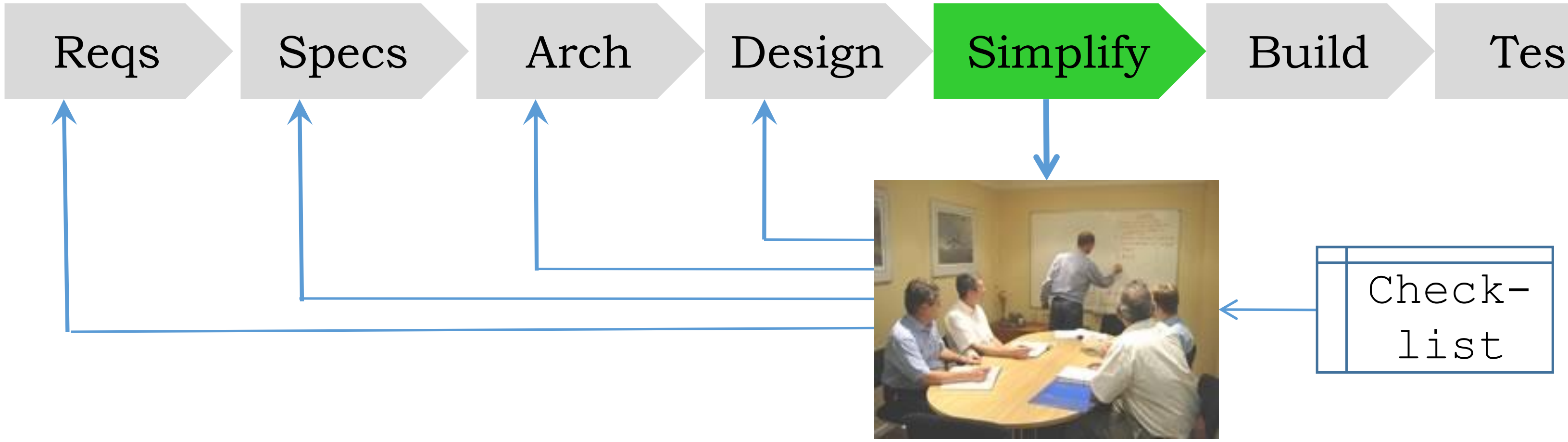
- Complexity creeps up, incrementally growing over long time
- Complexity occurs locally in many different specifications, programs and interfaces, but its impact is global
- Complexity may grow to such a state, that the IT-ystem becomes unmanageable or commercially unviable
- Containing complexity growth requires continuous and substantial architectural intervention and strong management committment

How can we manage complexity ?

a) Implement a process step „*simplification*“ in your development process

b) Periodically carry out re-architecture programs „*complexity reduction*“

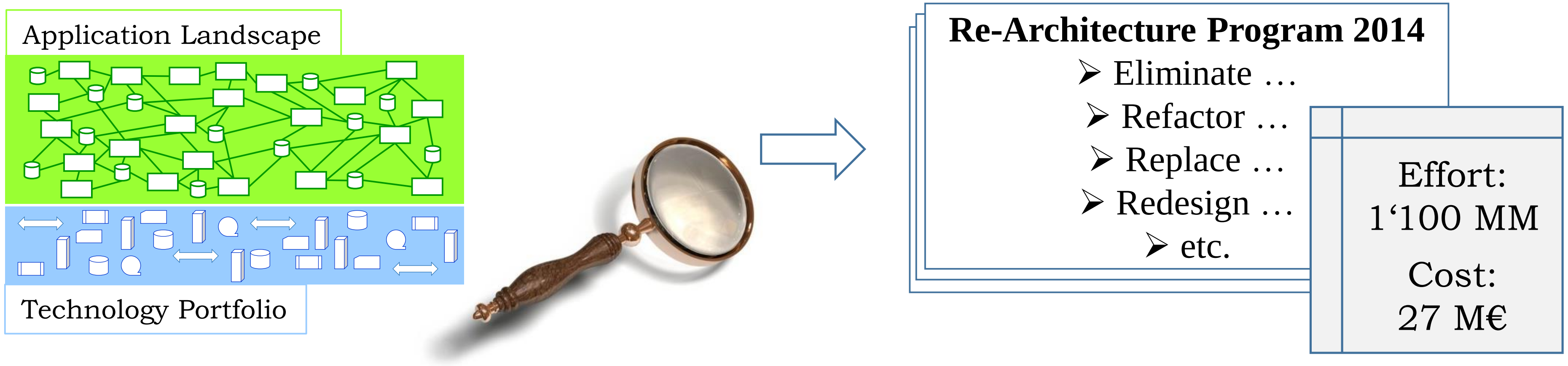
Implement a process step „*simplification*“ in your development process



Reqs → Specs → Arch → Design → **Simplify** → Build → Test

Check-list

Periodically carry out re-architecture programs „*complexity reduction*“



Application Landscape

Technology Portfolio

Re-Architecture Program 2014

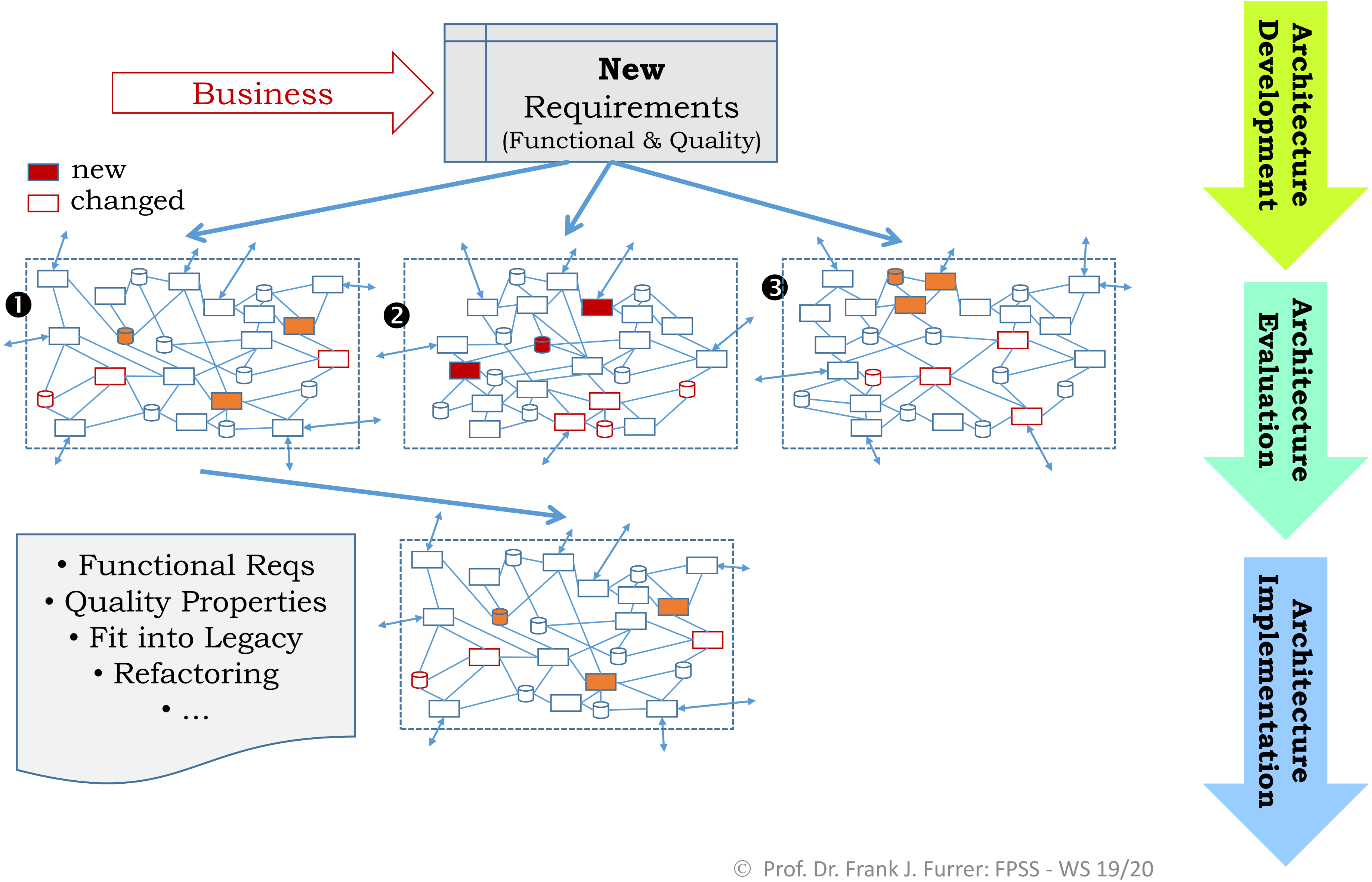
- ➢ Eliminate …
- ➢ Refactor …
- ➢ Replace …
- ➢ Redesign …
- ➢ etc.

Effort: 1‘100 MM

Cost: 27 M€

Future-Proof Software-Systems [Part 3B]

Complexity Reduction → **Simplification Process** → **Architecture Exploration**



- Functional Reqs
- Quality Properties
- Fit into Legacy
- Refactoring
- …

The architecture options are evaluated, assessed and the best one is selected

Future-Proof Software-Systems [Part 3B]

A12

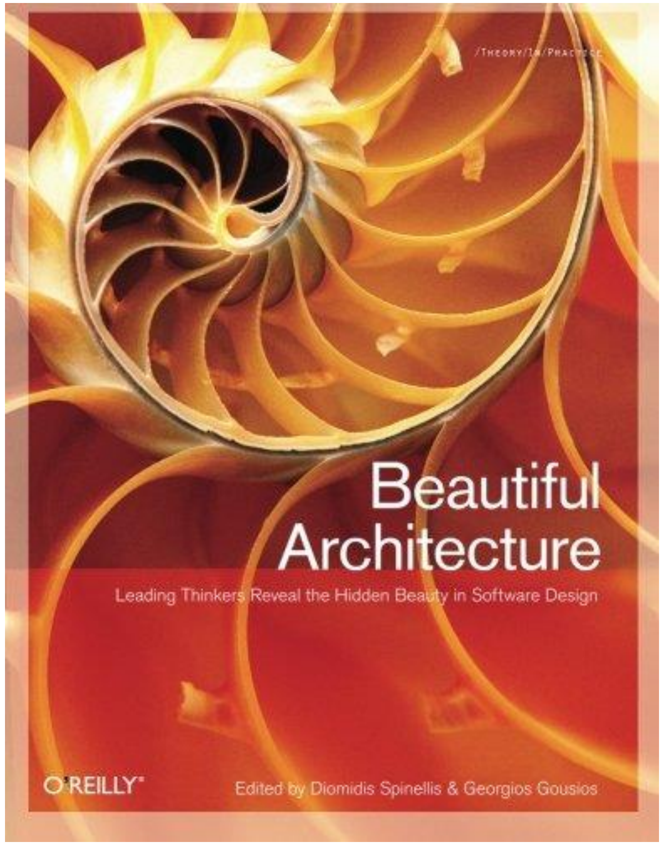
Architecture Principle A12:

Complexity and Simplification

1. Actively manage the complexity in your system:
    - Identify it
    - Understand it
    - Avoid and reduce it as much as possible (especially the *accidental complexity*)
2. Install a formal, controlled *process step* „simplification“ in your design and evolution procedures
3. For any (substantial) set of requirements develop several possible architectures and use an architecture assessment method to select the most suitable
4. Periodically execute *re-architeture programs* with the objective to reduce the complexity of your IT-system

Justification: Complexity is the largest single risk in IT-systems. By managing complexity, the unwanted or unnecessary complexity can be reduced – thus making the IT-system more changeable, manageable and dependable.

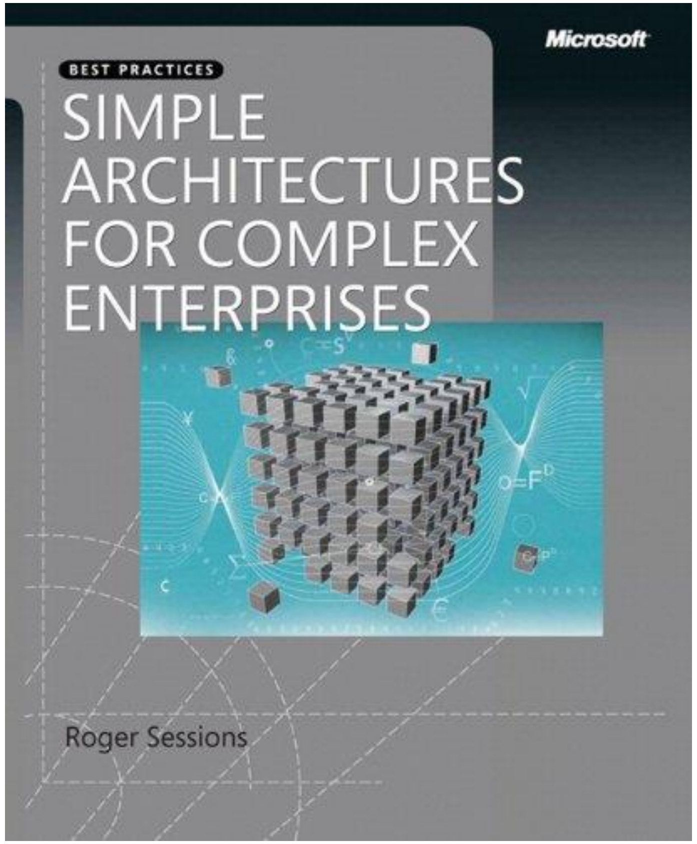
https://www.pinterest.com



Diomidis Spinellis, Georgios Gousios:
Beautiful Architecture – *Leading Thinkers Reveal the Hidden Beauty in Software Design*
O'Reilly and Associates, USA, 2009. ISBN 978-0-596-51798-4

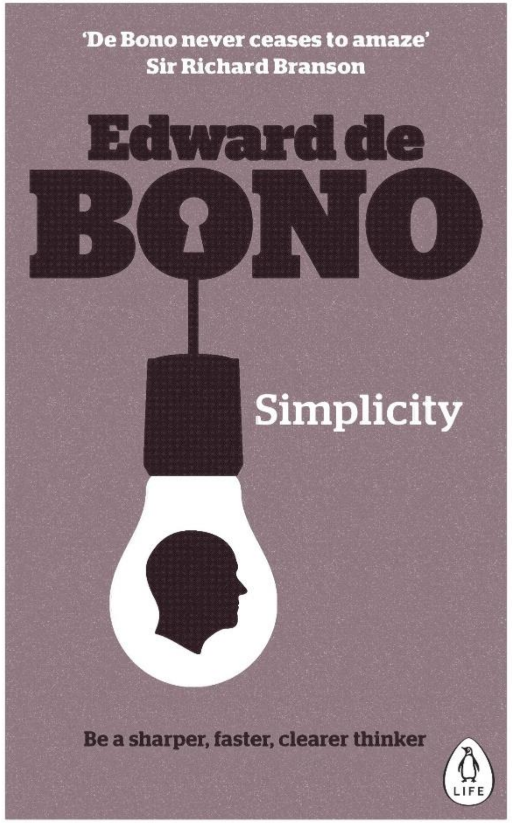
Future-Proof Software-Systems [Part 3A]

Textbook



Roger Sessions:
Simple Architectures for Complex Enterprises
Microsoft Press, USA, 2008. ISBN 978-0-735-62578-5

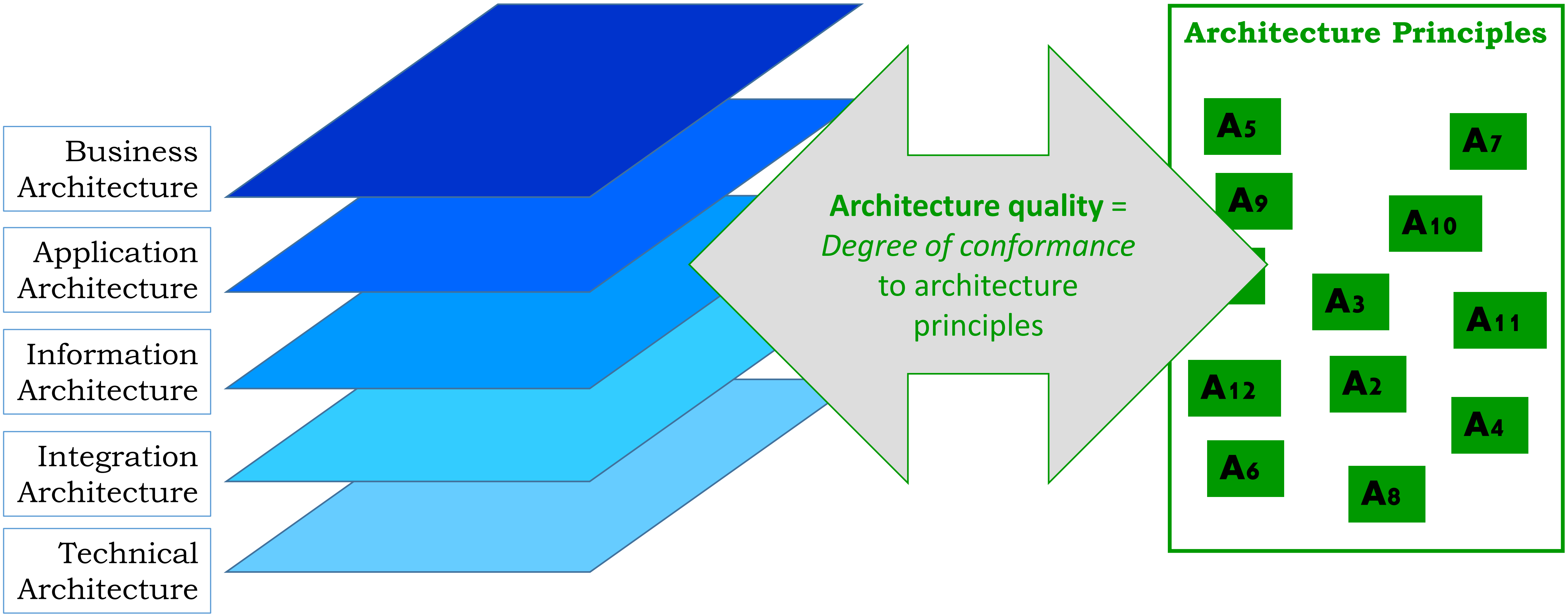
Textbook



Edward de Bono:
Simplicity
Penguin Life, 2015. ISBN 978-0-241-25748-7

Architecture Quality

Architecture Quality



Business Architecture

Application Architecture

Information Architecture

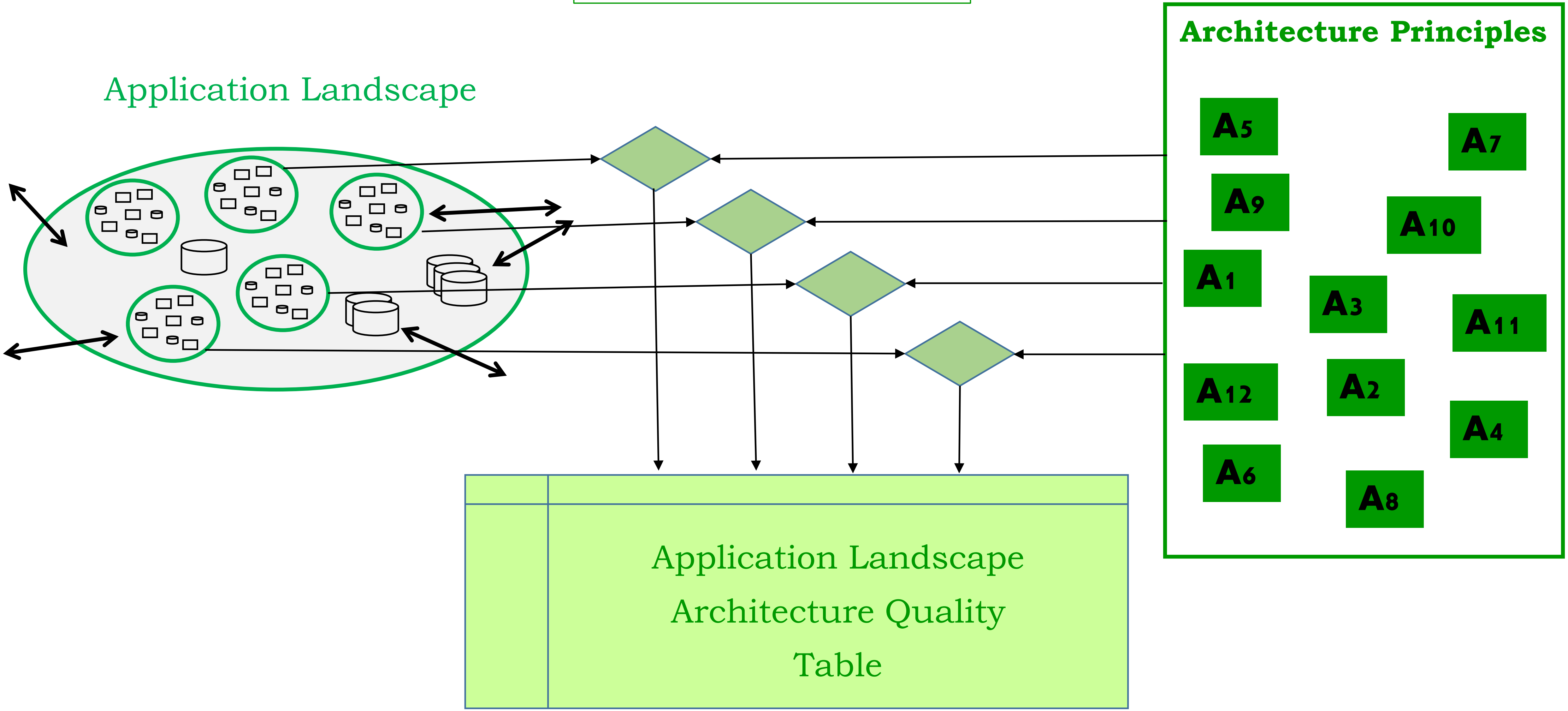
Integration Architecture

Technical Architecture

Architecture quality =
Degree of conformance
to architecture principles

Architecture Principles

Architecture Quality



Beautiful Architecture

Requirements
- Req A
- Req B
- …
- Req Y

Functionality (Business Value)

Architecture Properties:
- **Changeability**
- **Dependability**
- Performance
- …

Architecture-Greatness:
- **Simplicity**
- **Elegance**

Future-Proof Software-System Engineer

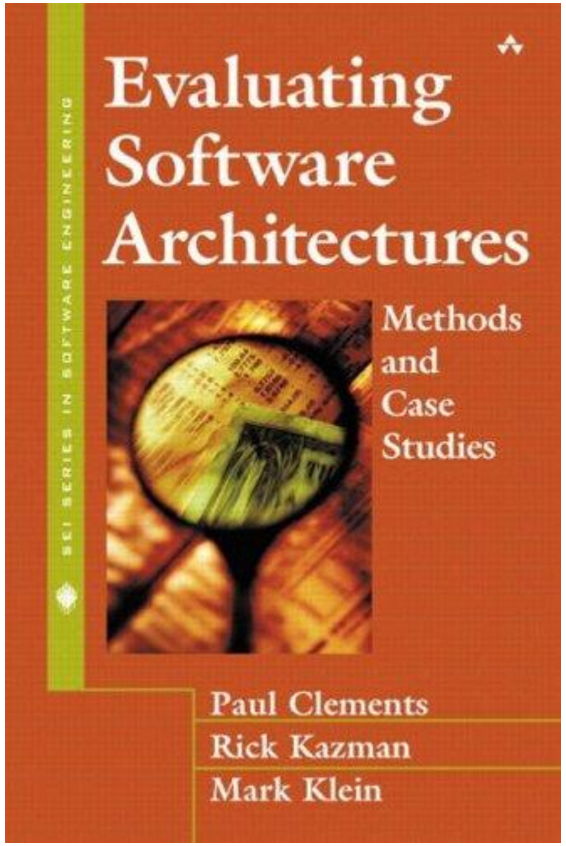
Architecture Principles

A5 A9 A7 A1 A3 A10 A12 A2 A11 A4 A8 A6

http://de.123rf.com

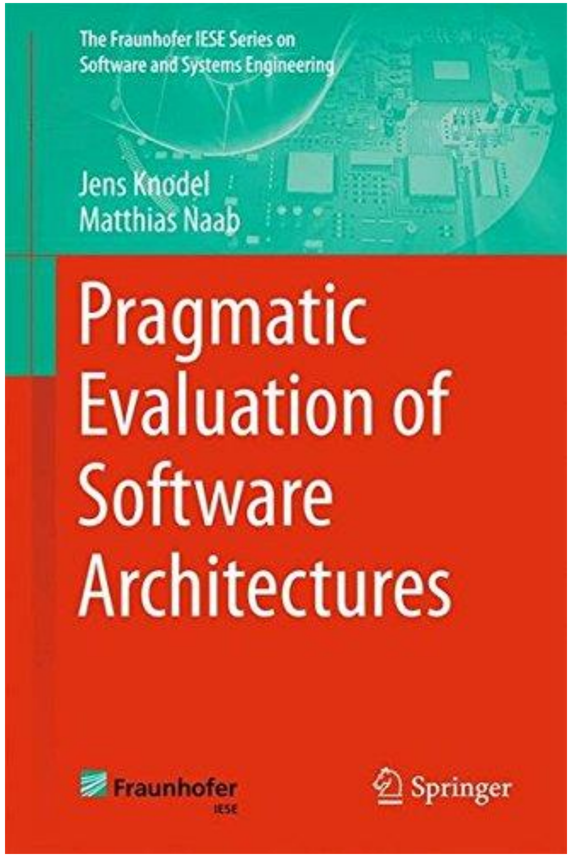
Future-Proof Software-Systems [Part 3A]

Textbook



Paul Clements, Rick Kazman, Mark Klein:
Evaluating Software Architectures – *Methods and Case Studies*
SEI Series in Software Engineering, Addison-Wesley, USA, 2001. ISBN 978-0-201-70482-2

Textbook



Jens Knodel, Matthias Naab:
Pragmatic Evaluation of Software Architectures
Springer-Verlag, Germany, 2016. ISBN 978-3-319-34176-7

Part 3B