

Chapter 13



Prof. Dr. Harald Störrle
Danmarks Tekniske Universitet (DTU)

Chapter 13: Closure

DTU course 02264

Abstract

- In this chapter we will be looking back at the whole subject we have covered this term.
- In summarizing it, we will establish or reinforce the connections between the different fields and issues.
- Also, we will try and identify some trends that might change the way we do Requirements Engineering in the future.

Contents

1. Requirements Revisited
2. Requirements Engineering Techniques & Topics
3. Where do we go from here?



Prof. Dr. Harald Störrle
Danmarks Tekniske Universitet (DTU)

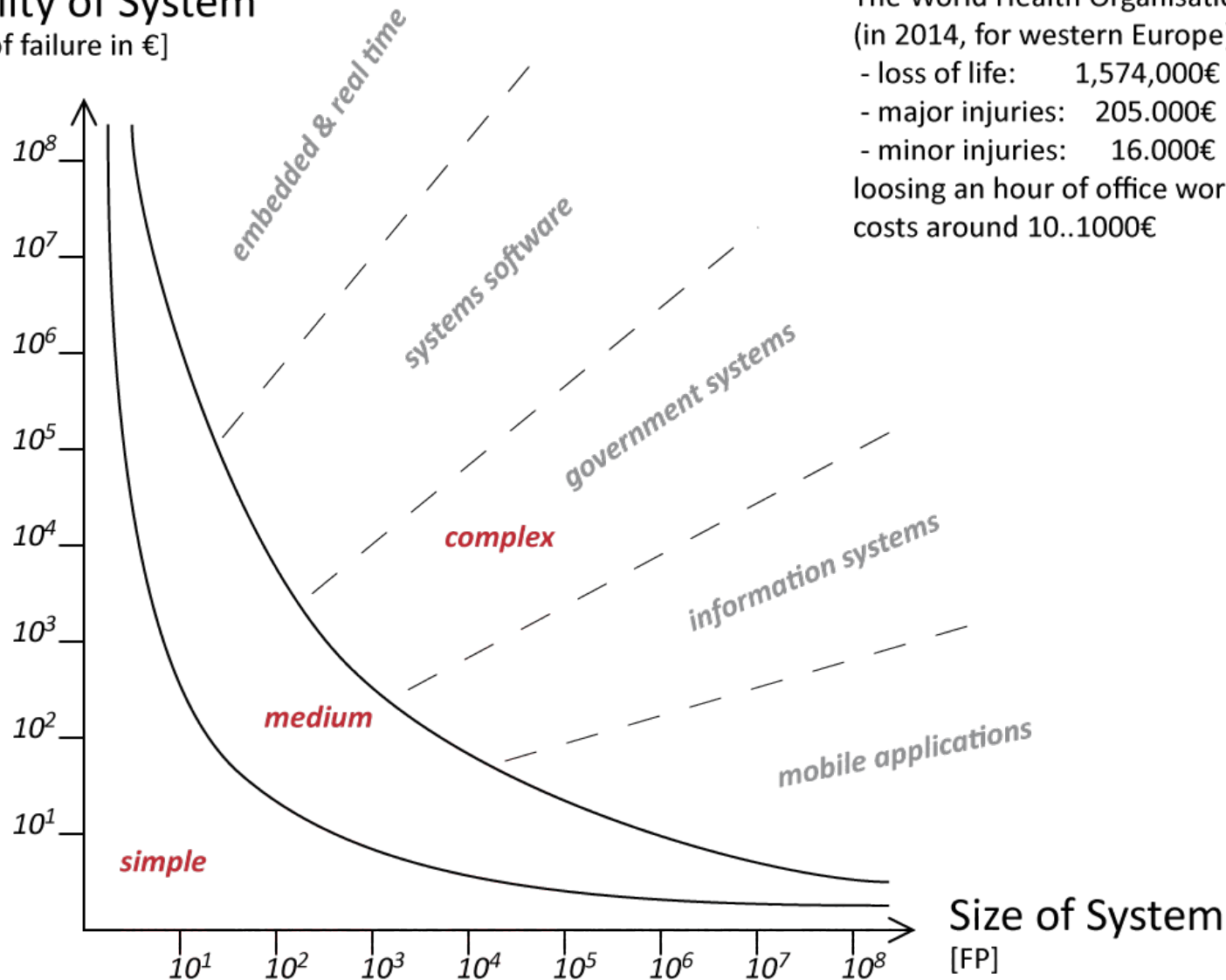
Chapter 13.1:

Requirements Revisited

DTU course 02264

Complexity of RE of software systems

Quality of System
[cost of failure in €]



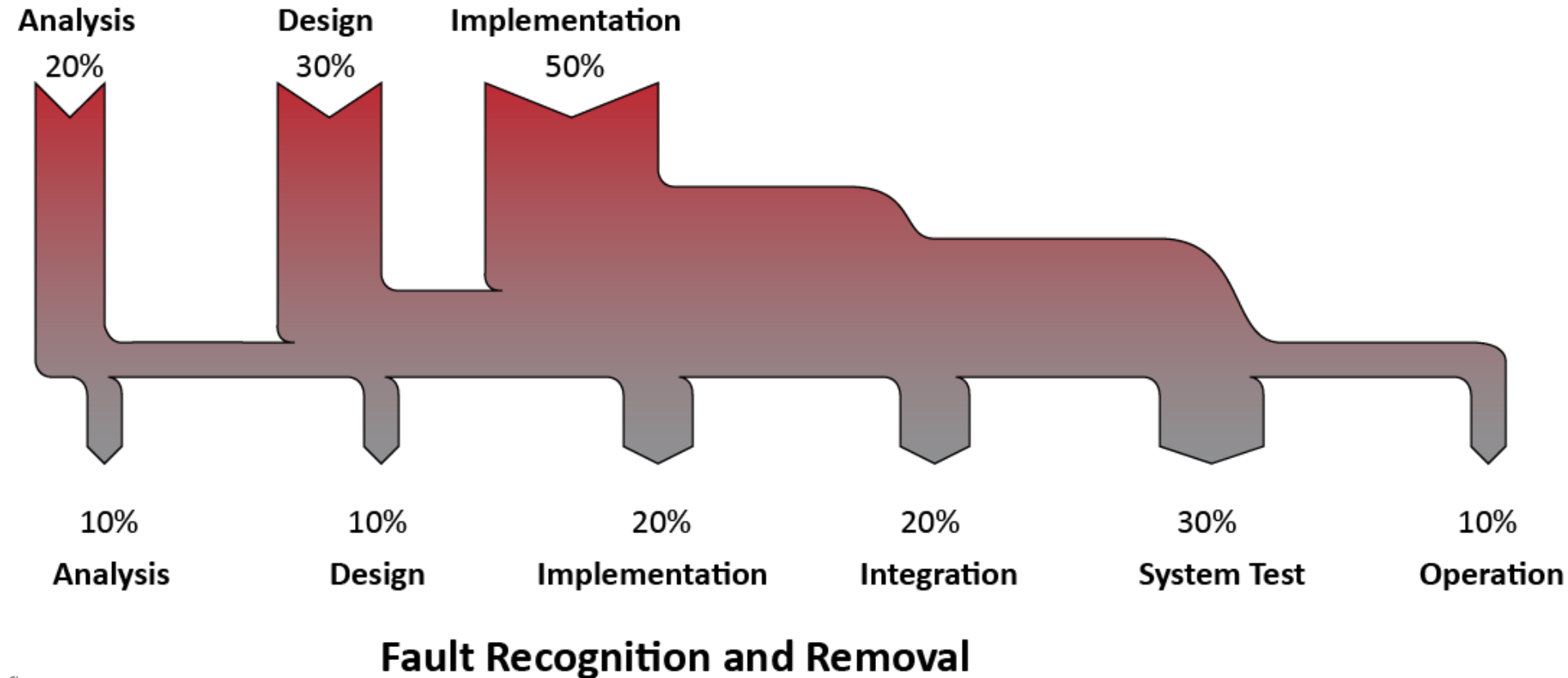
The World Health Organisation estimates
(in 2014, for western Europe) the cost of

- loss of life: 1,574,000€
- major injuries: 205.000€
- minor injuries: 16.000€

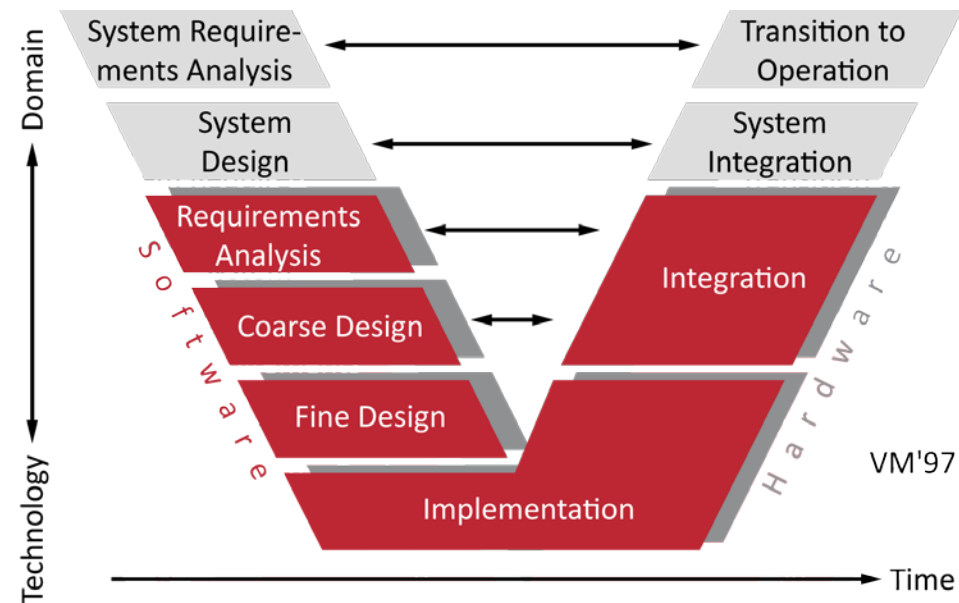
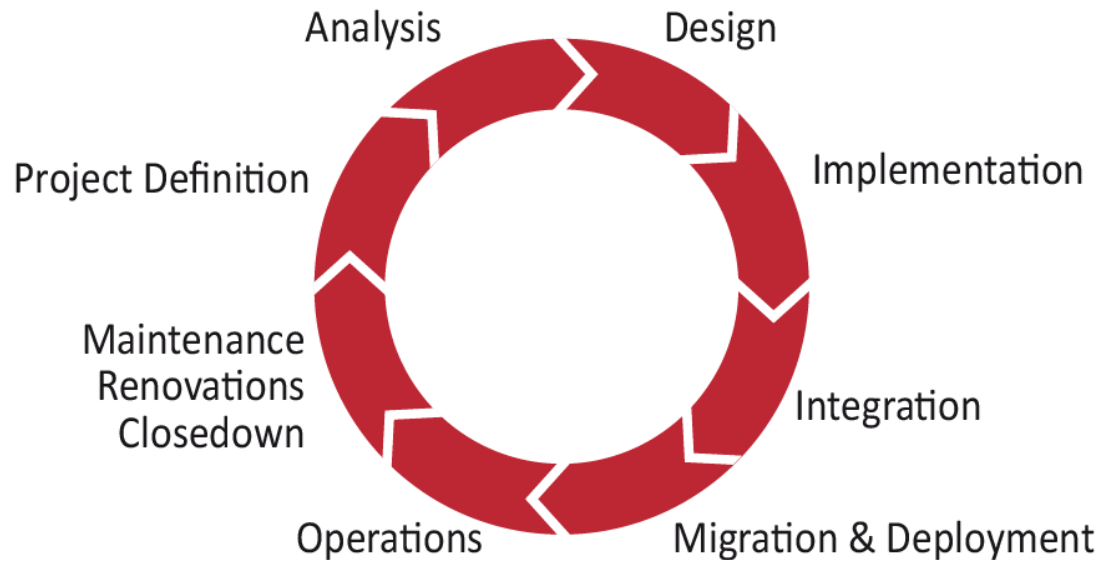
loosing an hour of office work
costs around 10..1000€

Putting in effort early pays back

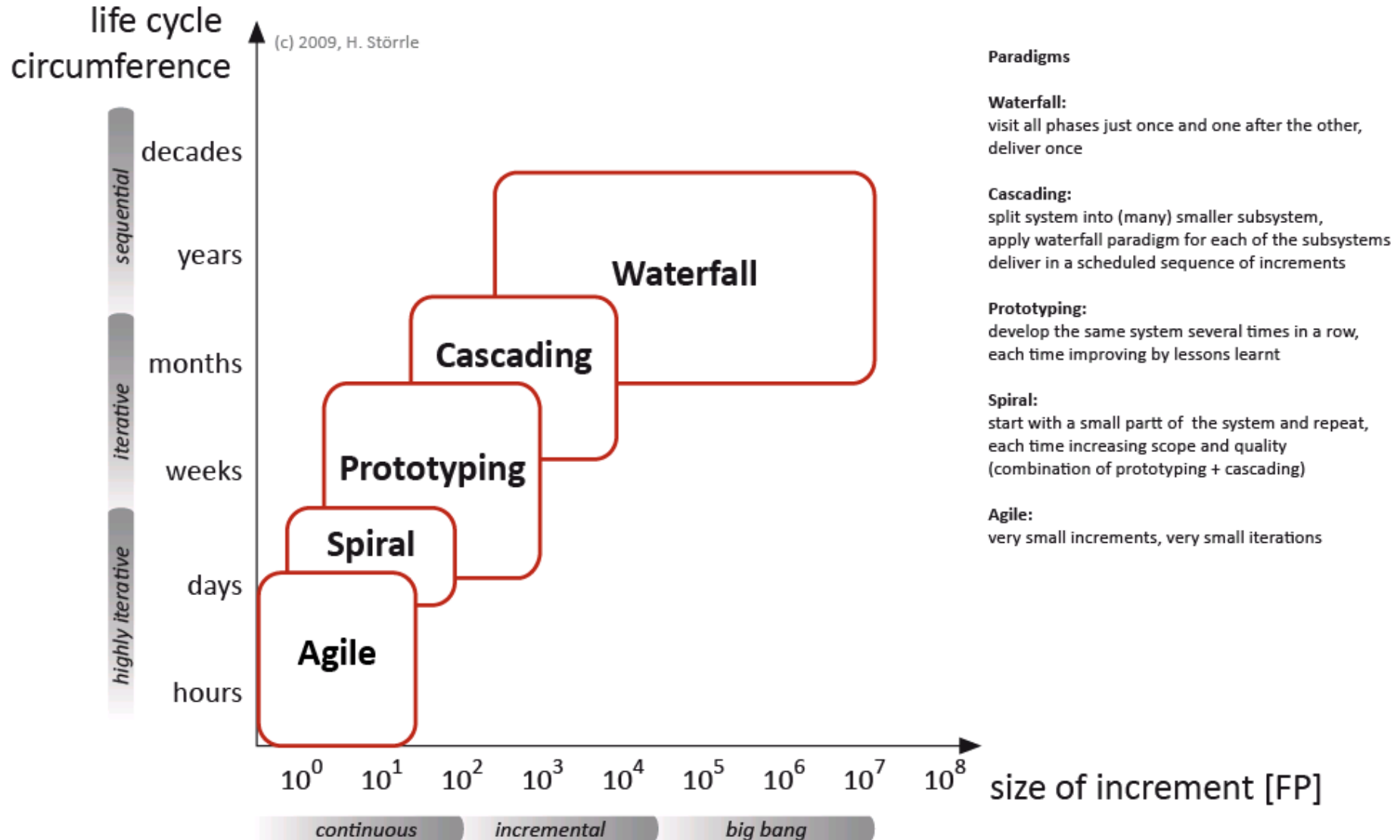
Introduction of Faults



The Software Lifecycle



Software Development Paradigms



Organizing Truth



Requirement Types

Requirement

Goal

Product-related

Constraint

Process-related

Feature

(Functional R.)

actions, functions, interactions, interfaces, and services a system shall provide cross cutting features (e.g. Undo/Redo).

Quality

(Non-functional R.)

desired system properties like throughput, performance, usability, maintainability, portability, safety & security

Organization

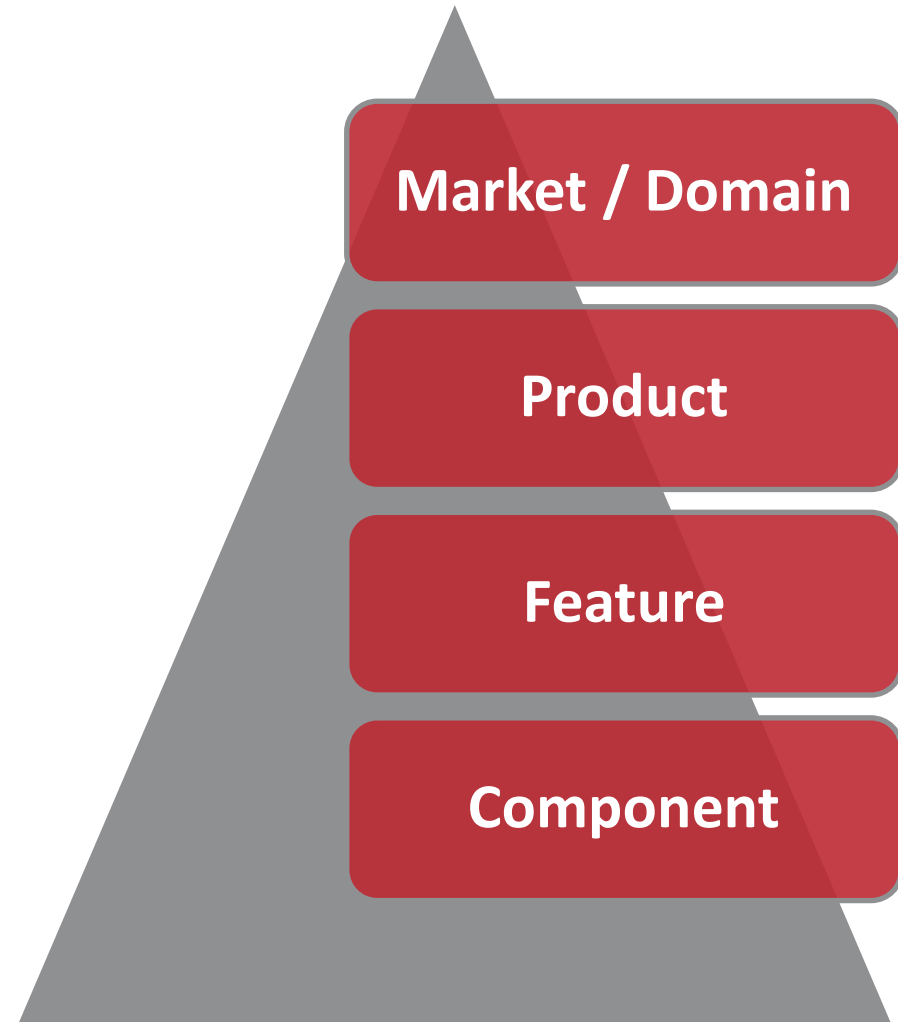
Available Staff, qualification, organisation maturity, legal and company regulations

Project

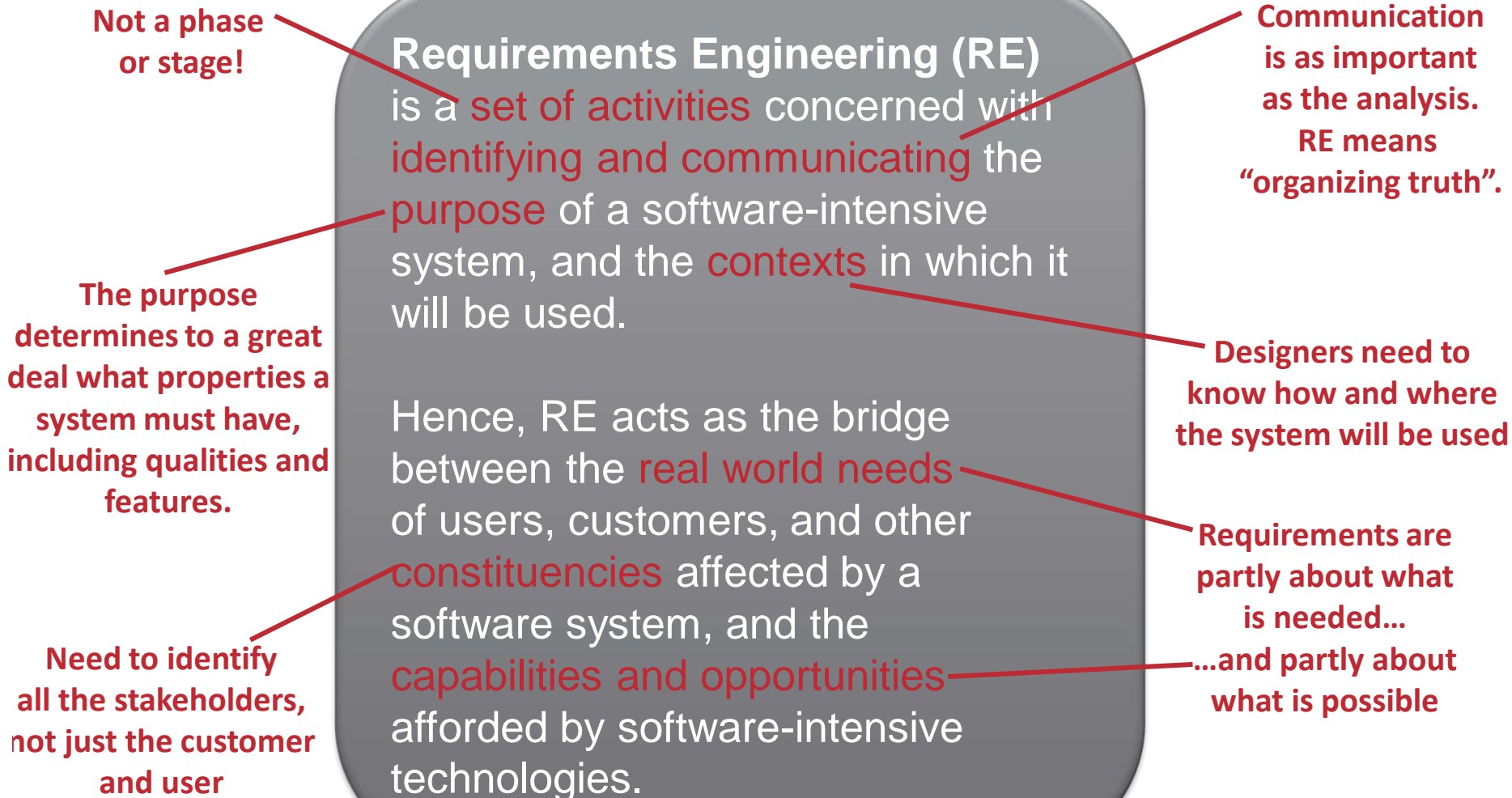
Time to market, budget, other projects, technology

Requirement Granularities

- 1. Requirements can also be grouped by granularity, i.e. the solution entity they address.**
- 2. Most, though not all combinations of size and type make sense.**



Definition “Requirements Engineering”



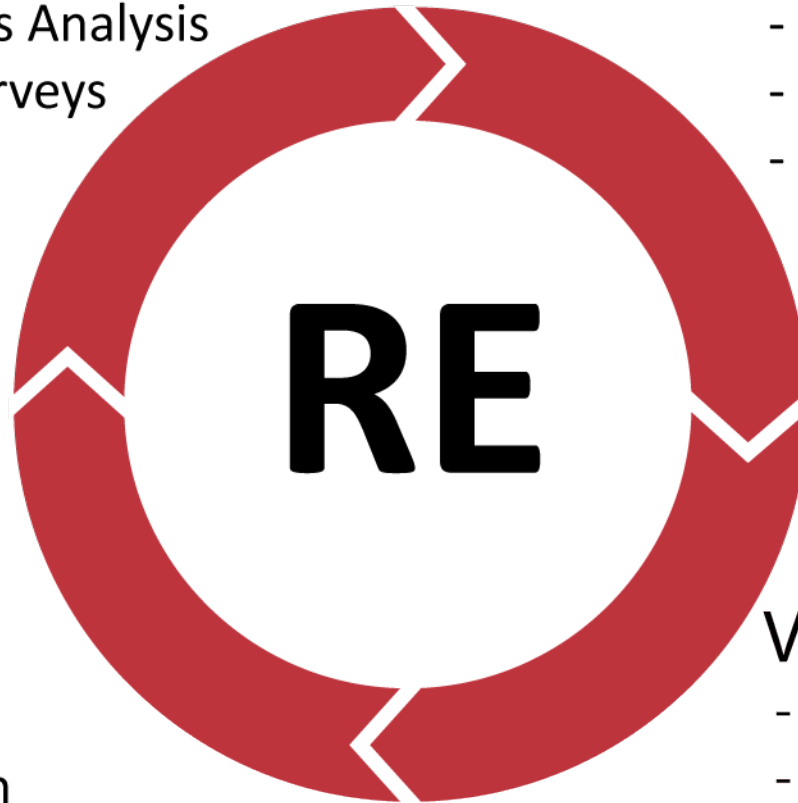
The Requirements Lifecycle

Ideation & Elicitation

- Business Process Analysis
- Interviews & Surveys
- Data Analysis

Elaboration

- Dataflow Analysis
- Domain Modeling
- Structured/Controlled Text



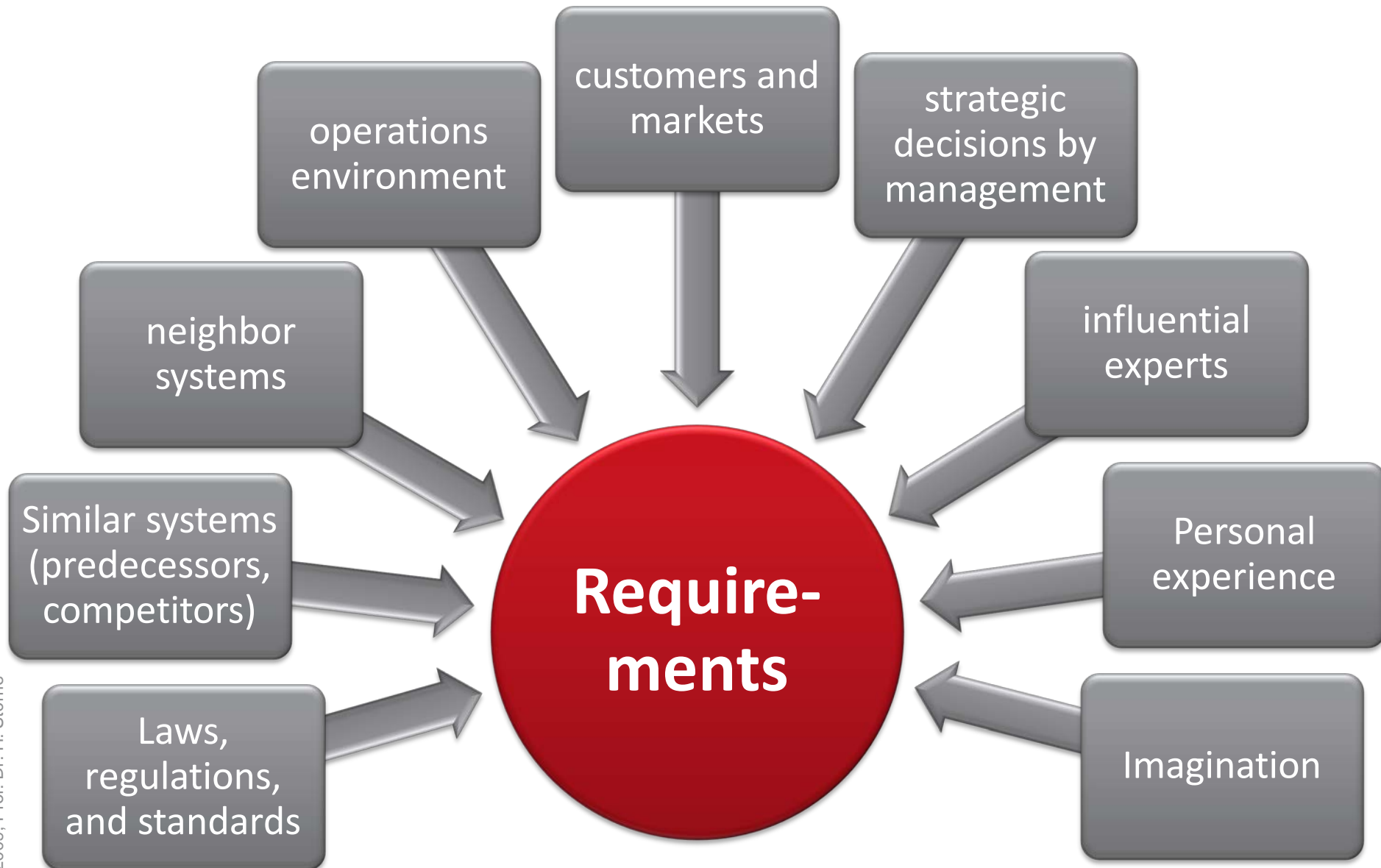
Management

- Tracing
- Effort Estimation
- Version Control

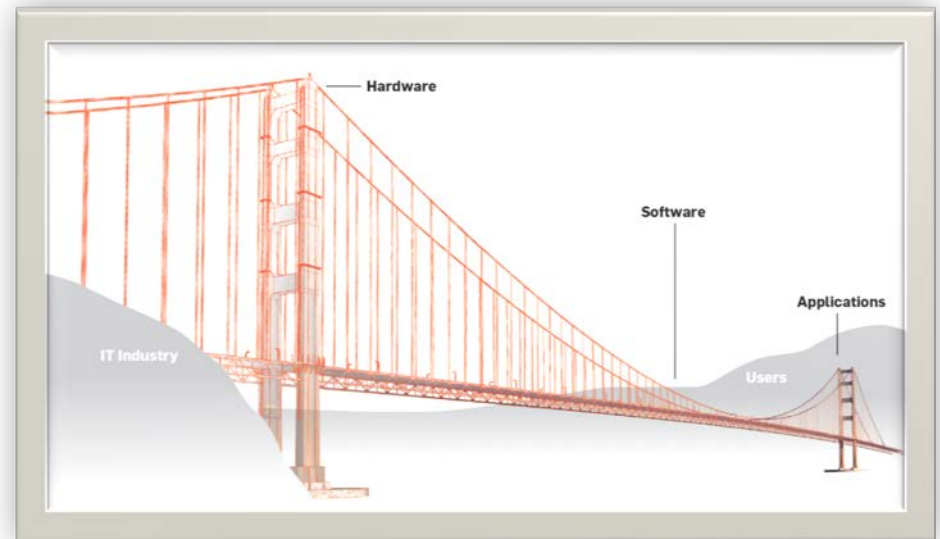
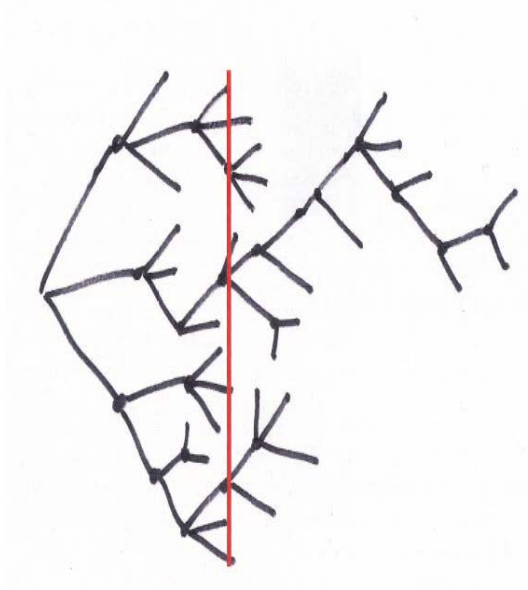
Validation

- Reviews & Inspections
- Formal Methods
- Prototyping

Sources of Requirements



Sketching & Ideation





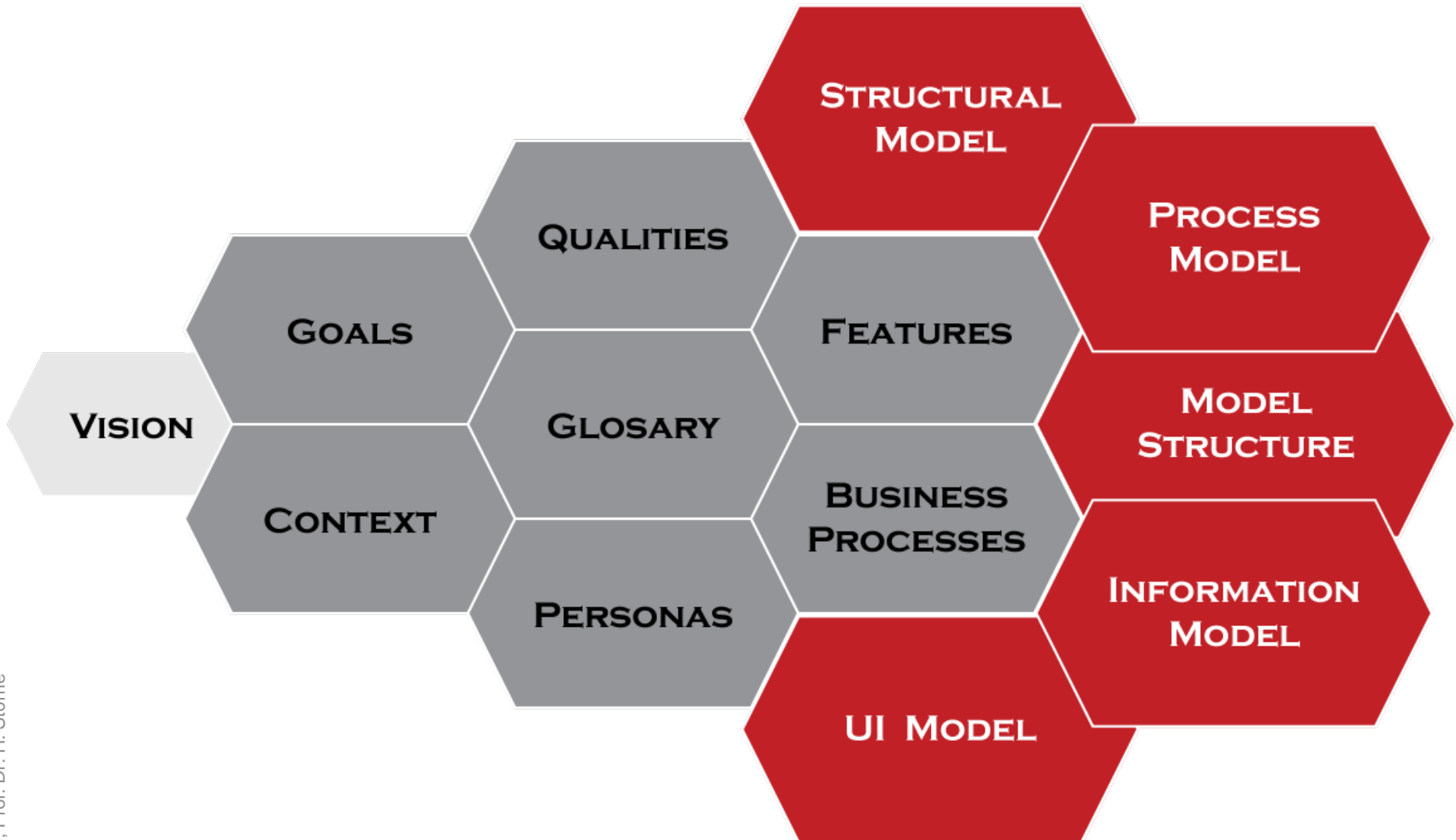
Prof. Dr. Harald Störrle
Danmarks Tekniske Universitet (DTU)

Chapter 13.2:

Requirements Engineering Techniques & Topics

DTU course 02264

Looking Back at Techniques This Term



Elicitation Technique Overview

Objective Techniques

- Data Analysis
- Background Reading
- System Archaeology
- Laws & Regulations

Observational Techniques

- Ethnographic field studies
- Protocol Analysis
- Apprenticing
- Participant Observation

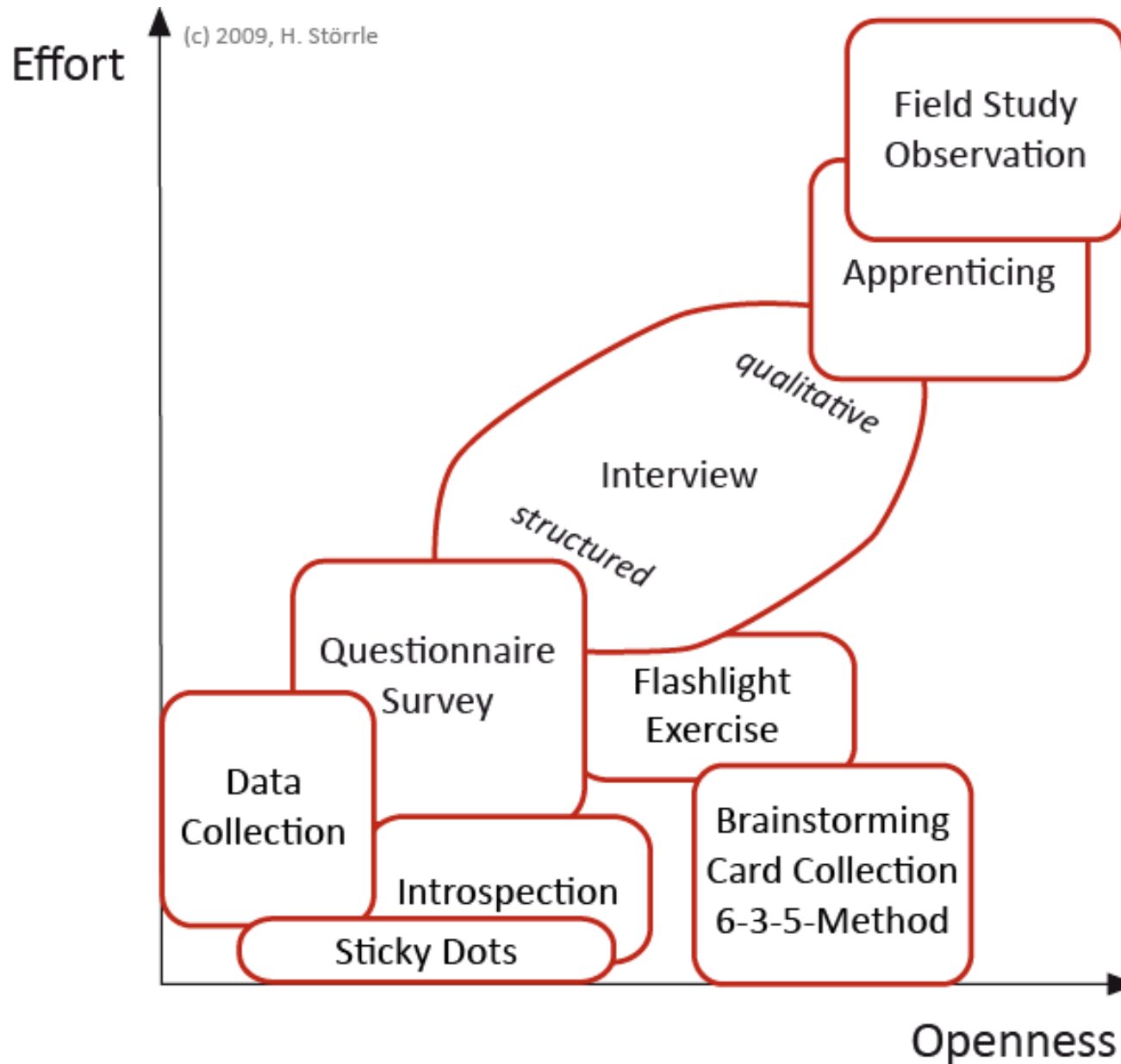
Conversational Techniques

- Interviews
- Surveys, Focus Groups
- Group dynamics
- Role Playing

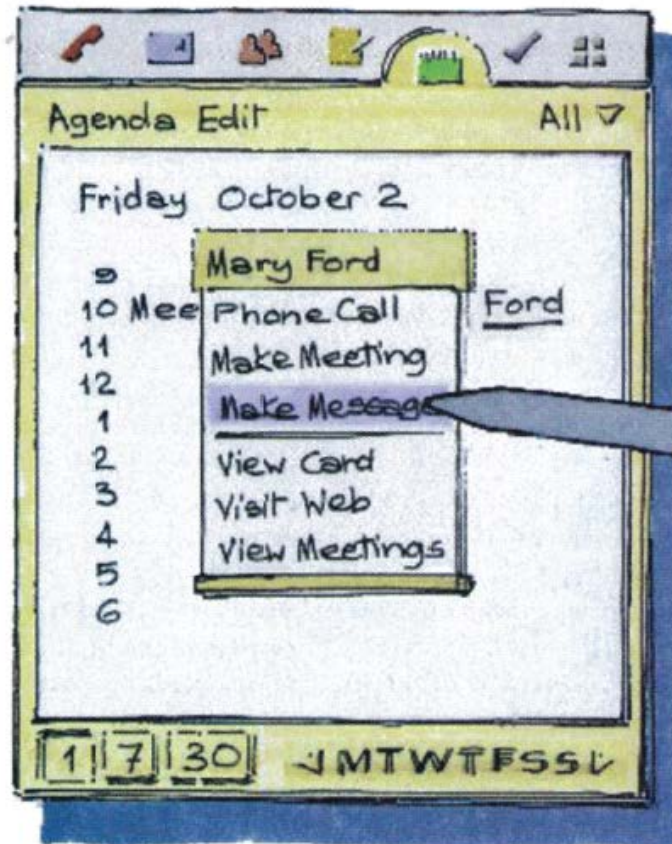
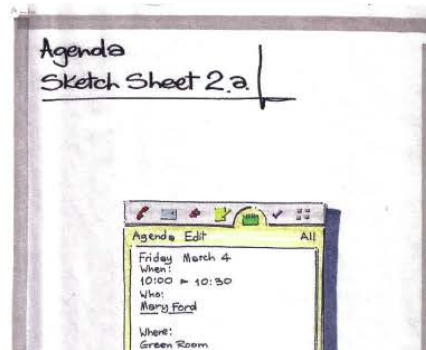
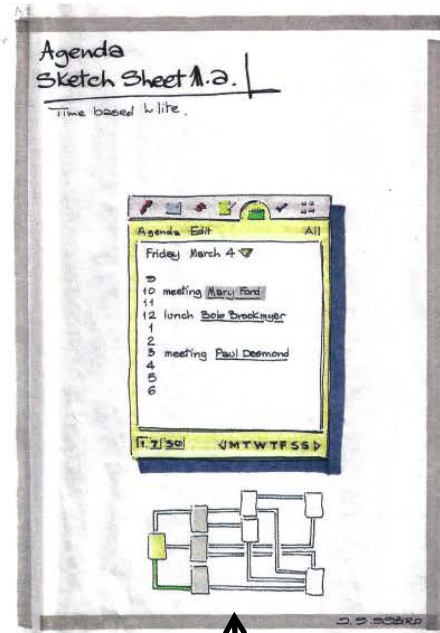
Introspective Techniques

- Storytelling
- Personas
- Brainstorming
- Mind-Mapping

Effort vs. Openness



Maintaining the Overview in Storyboards



Links are automatic

What a link can do

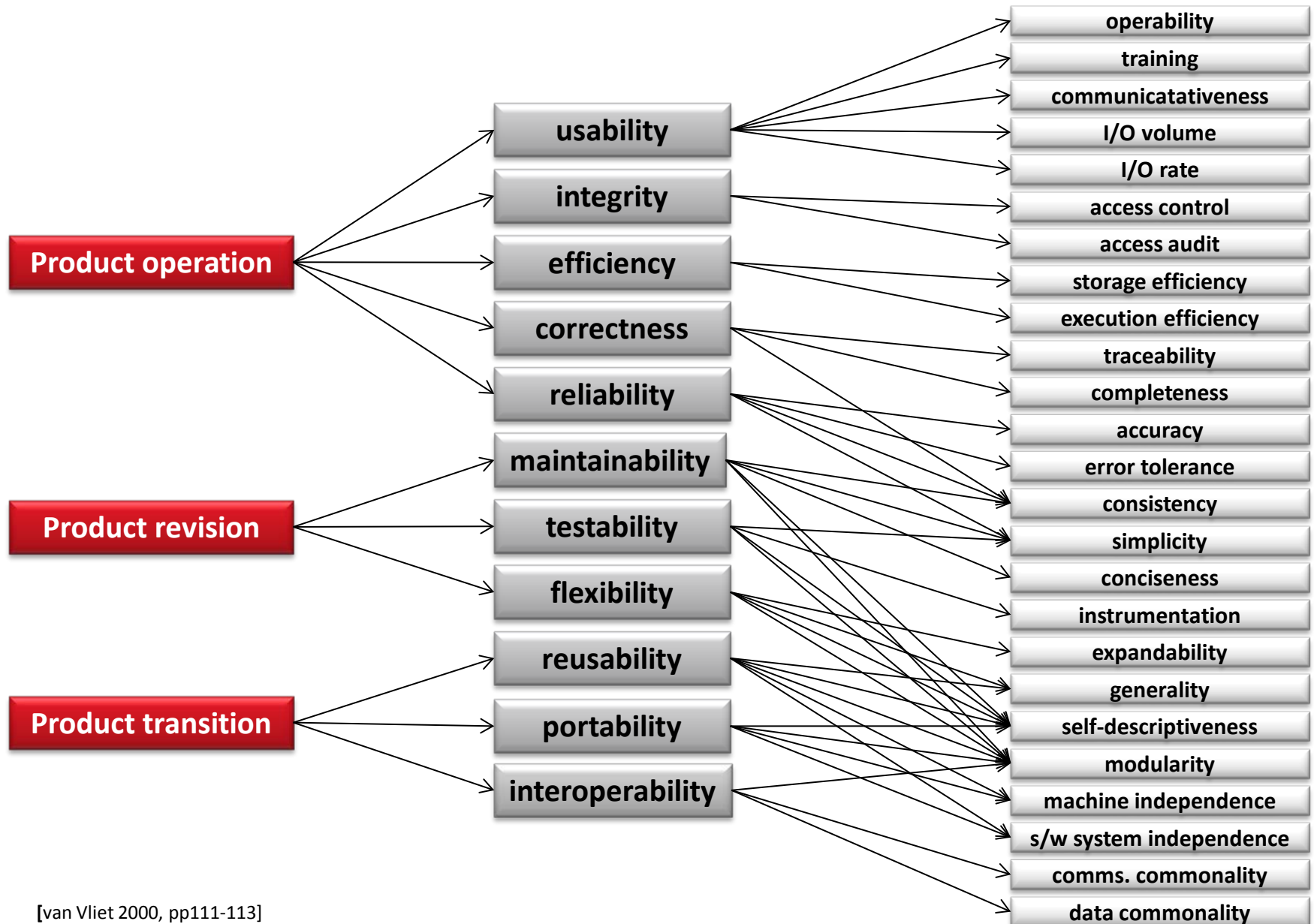
Links auto build

Links speed searches

Links make chains

Links chain text too

Software Quality Attributes (McCall)



RQAs for different system types

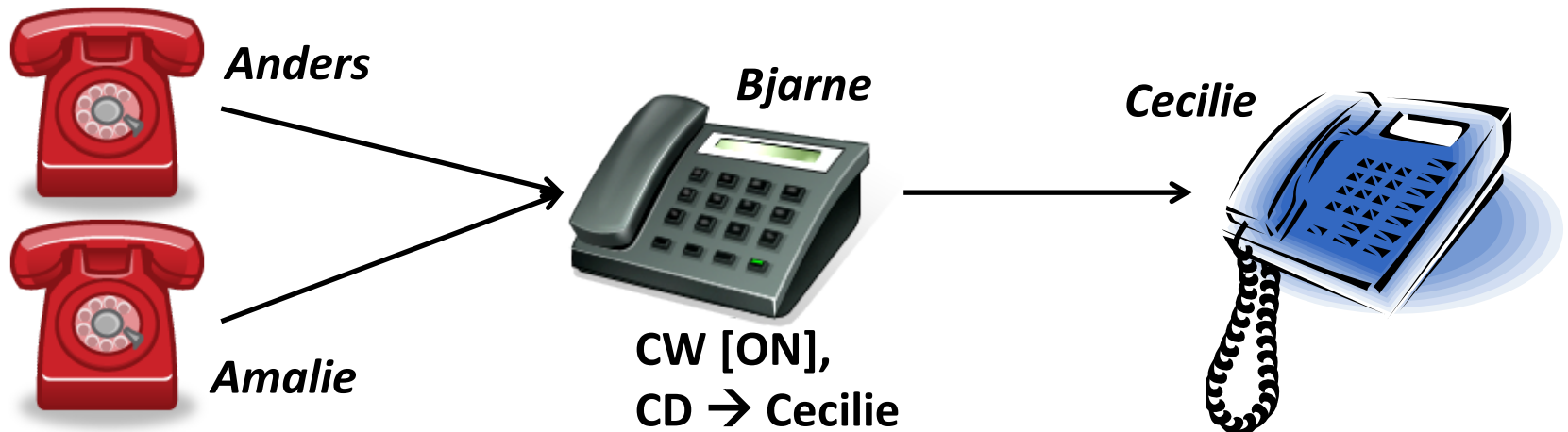
Quality Attribute	Word	Magic Draw	ESP	AMADEUS	Compiler	Excel Macro	ERH	iPod
Reliability Re-startability Availability	--	-	++	+	--	--	+	-
Efficiency Capacity Performance	-	+	++	+	+	--	+	++
Integrity Safety Security Privacy	-	--	--	++	--	--	++	--
Usability Understandability Learnability	++	+	--	--	-	-	-	++
Operability Interoperability Robustness	++	+	--	++	-	-	++	--
Maintainability Portability Testability	+	+	+	-	-	--	++	--

Feature Interaction CD/CW

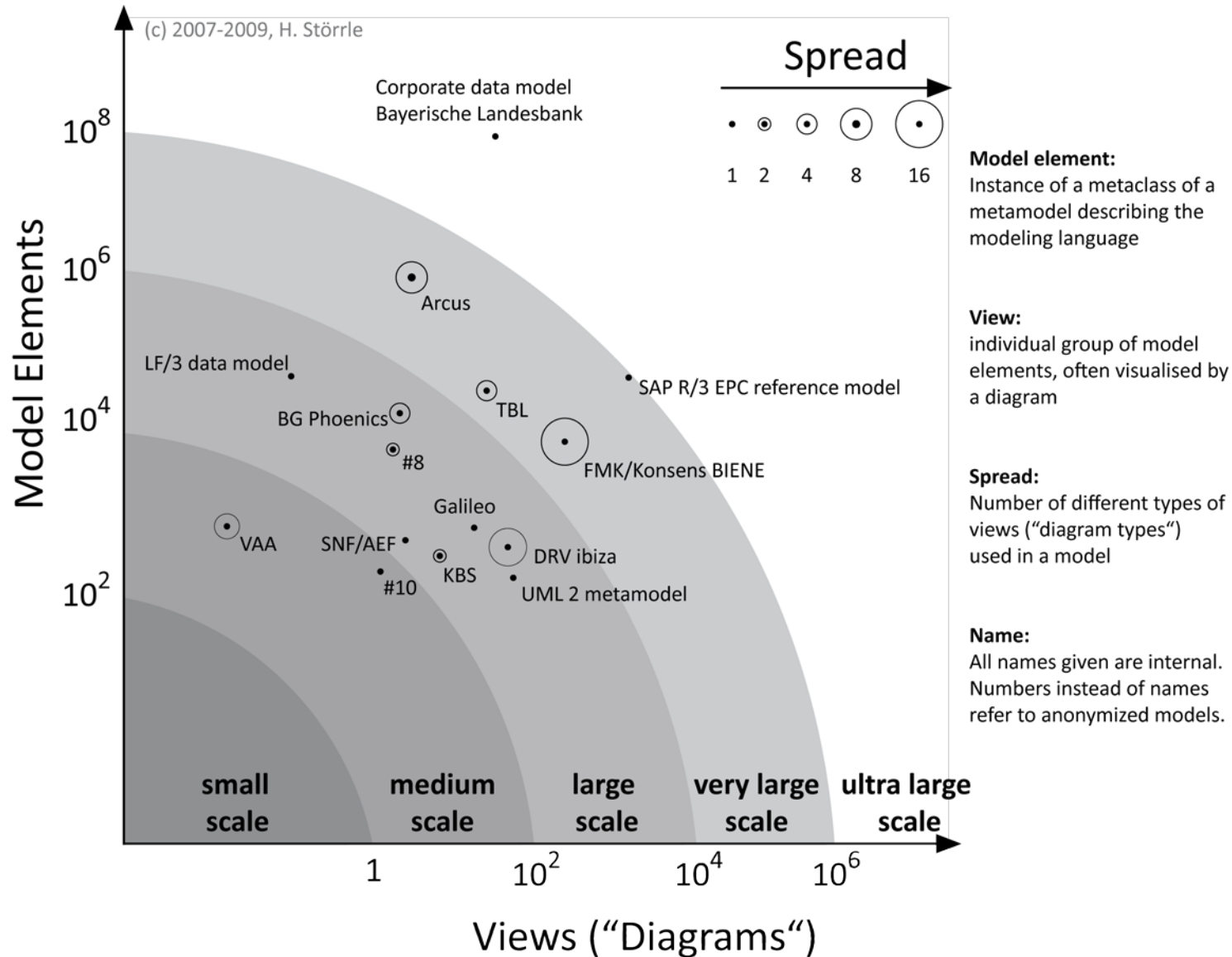
- There are many problems related to requirements, for instance feature interaction.

CW (Call Waiting):

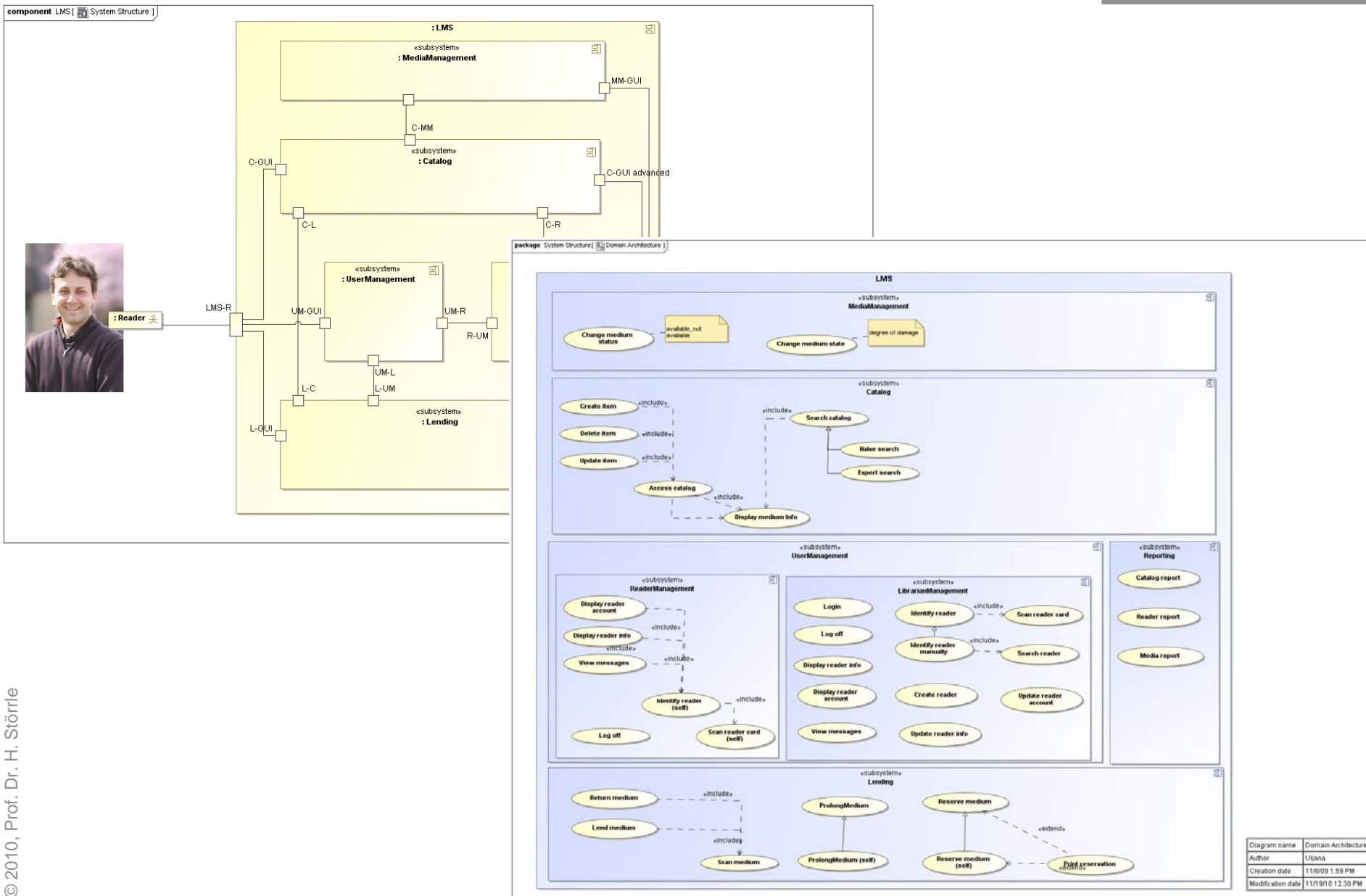
If the line is busy and a new call arrives, the callee is notified and decides which call to continue.



Models may get large

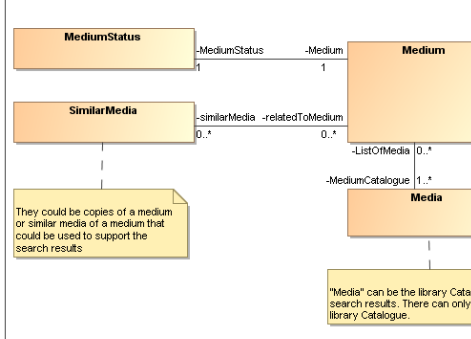
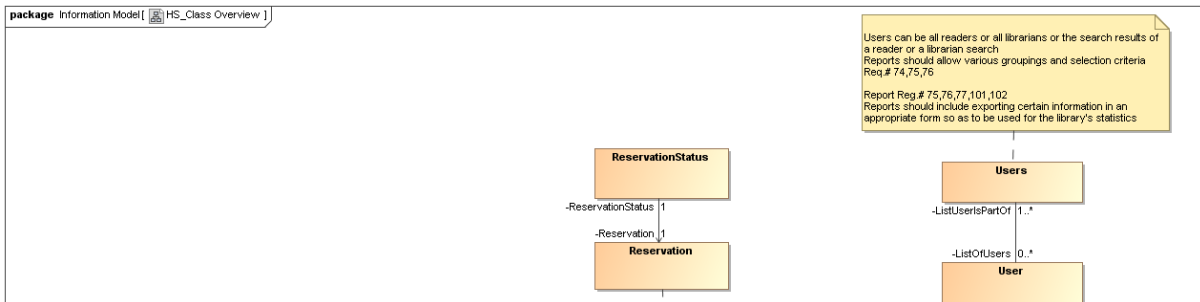


System Structure Models

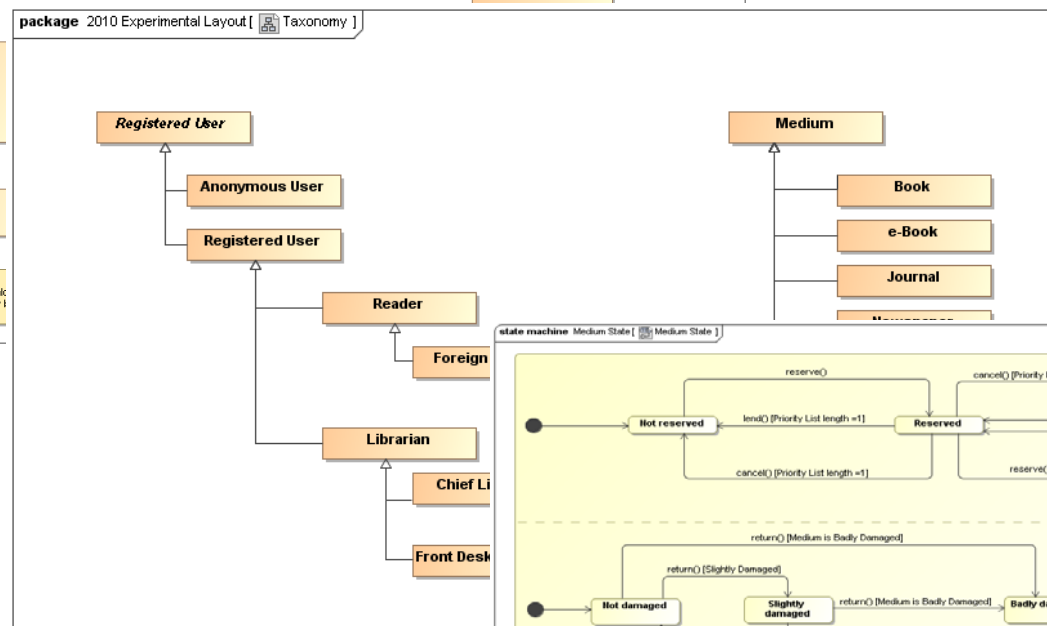


Information Models

package Information Model [HS_Class Overview]



package 2010 Experimental Layout [Taxonomy]



state machine Medium State [Medium State]

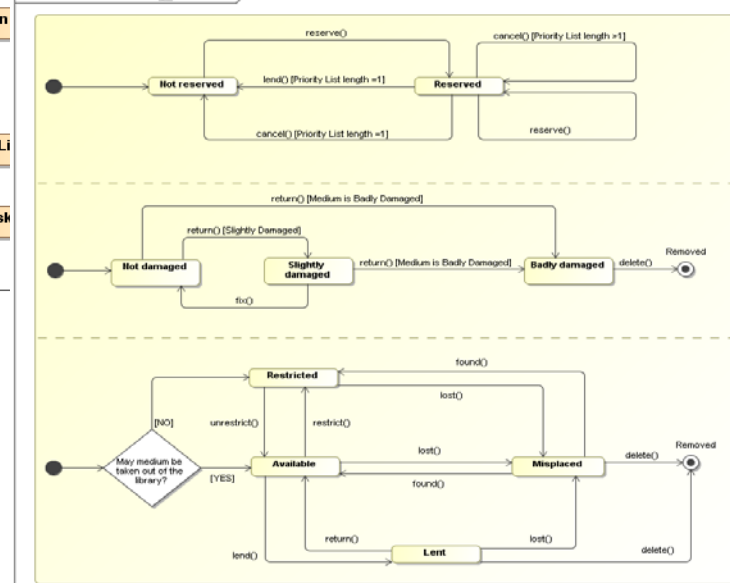


Diagram name	Medium State
Author	enigma39139
Creation date	11/01/09 11:00 PM
Modification date	11/22/10 3:06 PM

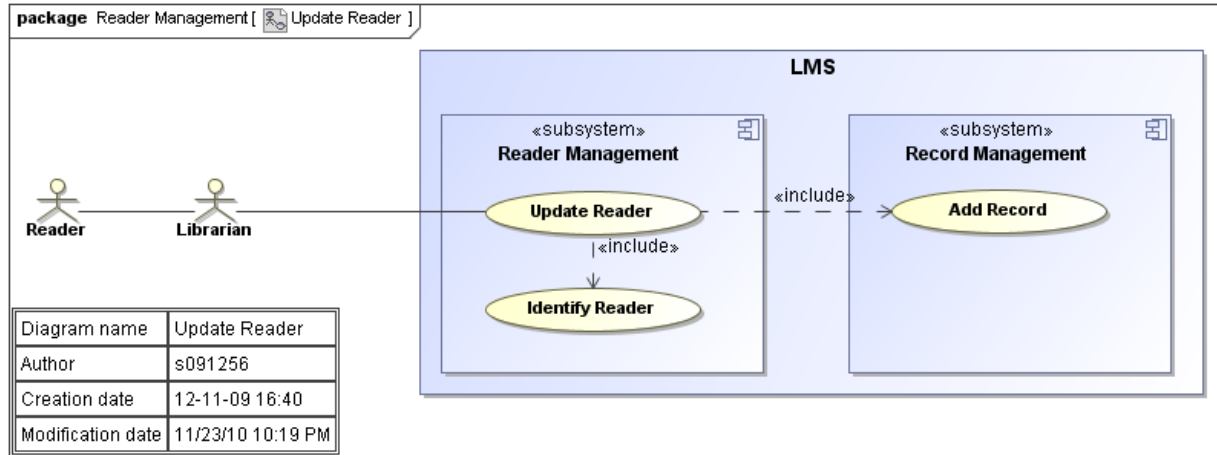


Diagram name	Update Reader
Author	s091256
Creation date	12-11-09 15:11
Modification date	11/24/10 3:31 PM

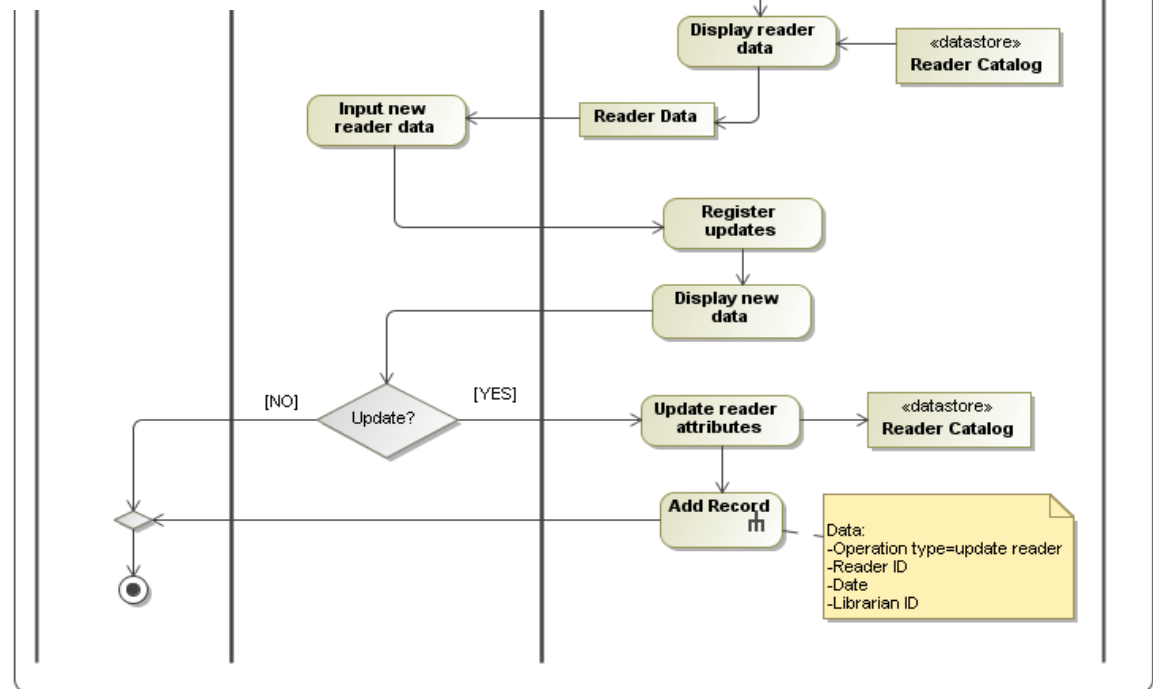
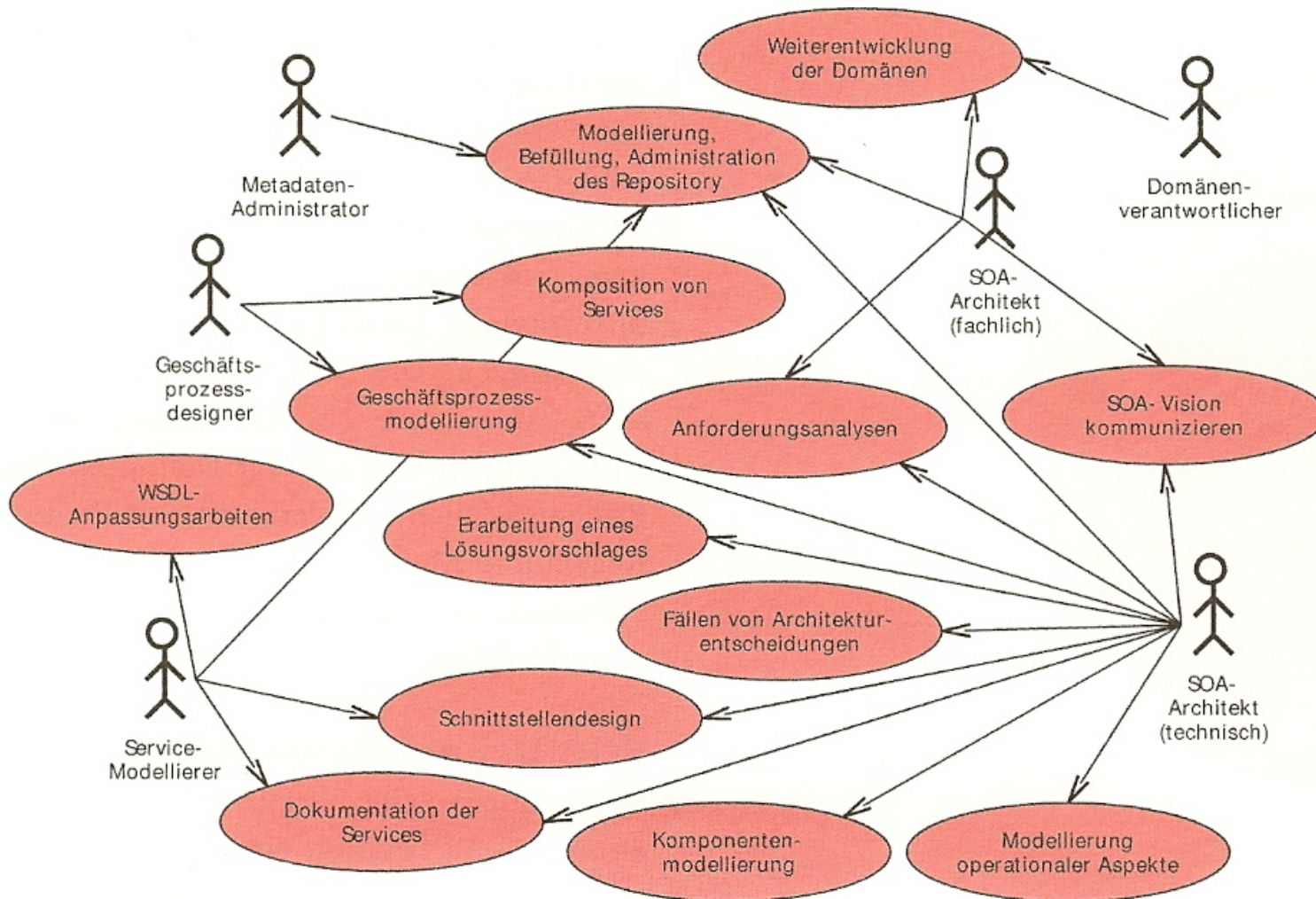


Diagram Layout: The Good, The Bad & The Ugly



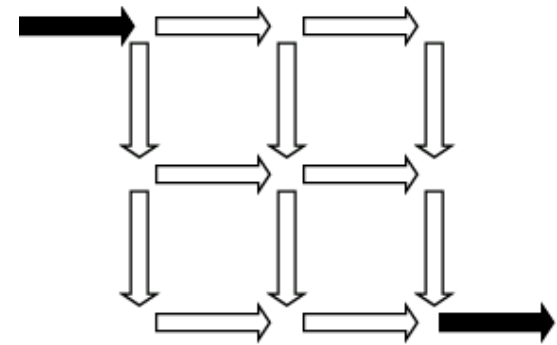
Many roads lead to Rome (or Lyngby)

1. There is no single “right” sequence of steps to create the “best” model.

1. However, models are more or less well-suited to their purpose, and paths are more or less well-suited to getting there.
2. The results of following different paths will often converge in rather surprising ways.

2. If there are several possible next steps in modeling, take the cheapest.

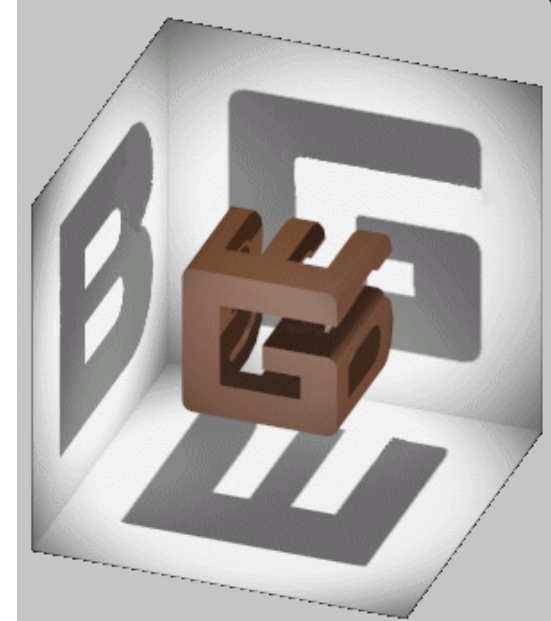
1. By “cost” we mean the modeler’s effort to go ahead a step. This effort is determined, for instance, by the modeler’s command of the modeling language, the available knowledge, the model developed so far, the available tools, and many



1. Do not hesitate to switch between several models to capture just the information that you have available.
2. If in doubt, use comments to capture the idea and return later to figure out the right way of modeling it.

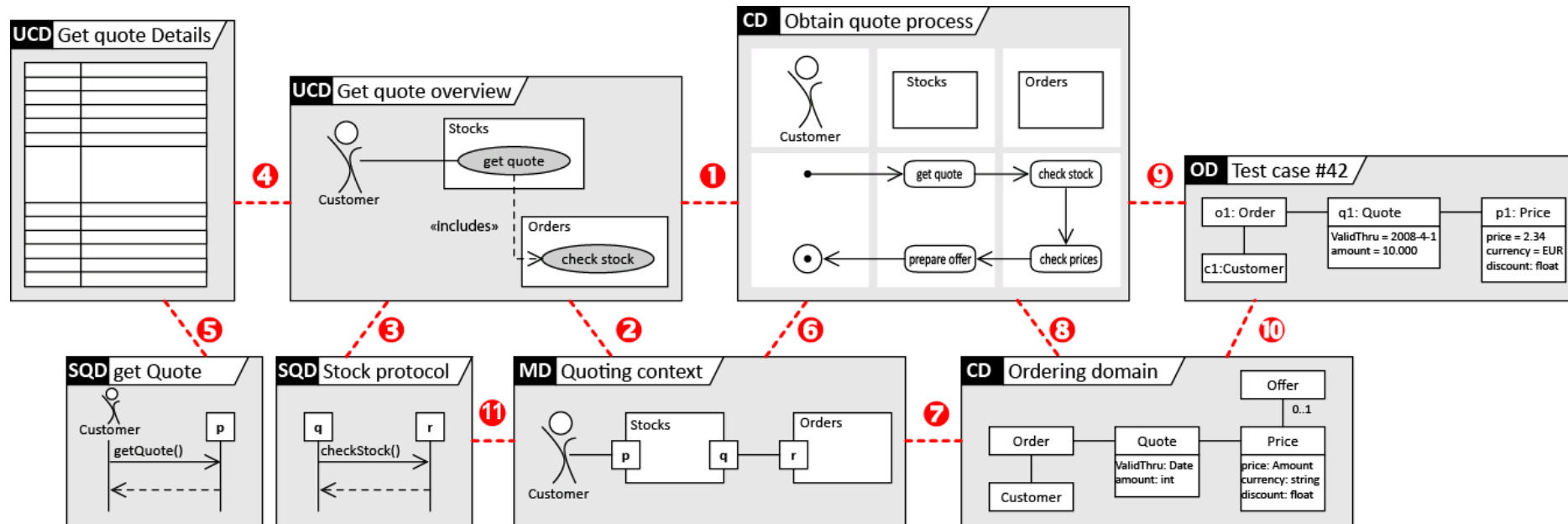
Multiple Views and Overlap

1. Every modeling language has its strengths and weaknesses: some type of phenomena are easily captured and conveniently presented, and some are not.
2. Complex systems exhibit many different phenomena of all kinds, many of which will be relevant for requirements analysis and should thus be captured.
3. However, a single model or diagram will not suffice to achieve this.
4. So, most likely we need several views working together, for different aspects.



1. In order for several views to talk about the same thing, there must be some “common ground”, that is, some conceptual overlap between different views.

Derivations between model types



Add/enrich model elements

0 * → Tab, UCD, AD, MD from interviews, observations, and reverse engineering

- 1 UCD → AD swimlanes & actions
- 1 AD → UCD systems & use cases, A-U-associations & includes/extends
- 2 UCD → MD actors, parts, ports, and connectors
- 2 MD → UCD actors, systems, A-U-associations
- 3 IAD → UCD use cases
- 3 UCD → IAD create interaction partners, messages
- 4 UCD → Tab rows (actors, title, trigger/precondition, exceptions)
- 4 Tab → UCD actors, systems, use cases, extension points, includes/extends
- 5 IAD → Tab rows (steps, exceptional steps, conditions, triggers, results)
- 5 Tab → IAD default interaction partners, create messages



Prof. Dr. Harald Störrle
Danmarks Tekniske Universitet (DTU)

Chapter 13.3:

Requirements Engineering in the 21st Century

DTU course 02264

Software is not just the desert: it's the whole menu

- 20 years ago, IT systems were few, far apart and isolated.
- Today, computers (and their Software) are plentiful, everywhere, all of the time, and they are globally networked (“ubiquitous”).
 - Many consumer products are small computers (mobile phones, MP3 players, digital cameras, ...).
 - But also in less visible places computers abound (cars, machines, household appliances, pacemakers, ...).
- *"For every 25% increase in problem complexity, there is a 100% increase in complexity of the software solution."*
[Robert L. Glass: Sorting Out Software Complexity. CACM November 2003, 45(1), pp. 19-21]
- *„If knowledge is the key asset of the future, then the ability to create or discover that knowledge is the key wealth-generation activity. Since that knowledge will be stored mostly in software, software development should (and will) become the driving force of the new economy."*
[Phillip G. Armour: "Owning and Using." CACM June 2014, 57(6), pp. 29-30]
- *"The automotive industry has become part of the software industry."*
[M. Würtenberger: "Changing Automotive Industry", CEO, BMW Car IT Proc. Sw. Eng. & Mgmt., LNI-239, Ges.f.Informatik, p. 264]
- **IT is now a critical infrastructure.**

Software Ecology

- **From 1950 to the 1990's, most projects used to be “green field development”, integration happened in data centers.**
- **Today it's all about the integration: cross-cutting processes, global supply chains, and interaction with .**
 - In public and private administrations, almost all software lives in an ecosystem of applications. Large corporations have “Enterprise Architectures” of hundreds or thousands of interacting applications, databases, web sites, and so on.
 - In manufacturing, plants are more like giant networks with some robots attached (“Industry 4.0”).
- **Software affects everything, and everything affects software.**
- **Every product, system, legislation or regulation can not be thought without software support.**

Progress in Software Engineering

- **We have also changed the ways we build and deploy software.**
 - Web applications, Cloud Computing, and SOA transform the software market.
 - New languages and approaches are thought up all of the time (MDA, agile, ...)
 - If you don't know something, Google will.
 - End users create a growing portion of all software.
- **The organizations that create software and their processes change.**
 - We have outsourcing and offshoring, global software development, virtual organizations, open source development, and so on.
 - Many people have Smartphones, we have Wikis, Blogs, Skype, NetMeeting and free WLAN in many Airports and Hotels so that constantly being in touch is easy.
- **We have made fantastic progress in many areas which give us fabulous tools today.**
 - Model Checkers and Theorem Provers are becoming practical tools.
 - Today's IDEs (e.g. Eclipse, NetBeans, IntelliJ) are light-years ahead of Xemacs.

The 21st century is the conceptual age

1. It has been argued that we are now living in the beginning of the “Conceptual Age”:

“an era where creativity, design, and innovation become the major factors of global economic and cultural competitiveness.”

[Daniel Pink: “A whole new mind”]

1. Even if this is one of the usual exaggerations, it is obvious that companies like Apple obviously fit well into this theory.

Agricultural Age

In the 18th century, most people’s life was effectively dominated by agriculture: 90% of the population were farmers working the fields, like millennia before.

Industrial Age

In the 19th century, inventions like the steam machine, artificial fertilizer and the telegraph allowed a growing proportion of the population to worked in factories, mines, ship yards etc.

Information Age

In the 20th century, electronic computers and communications globally moved societies into the information age dominated by knowledge workers.

Conceptual Age

In the beginning of the 21st century, it looks like We are entering the conceptual age, where creativity, design, and innovation are the main Contributions of conceptual workers. ...

RE in the 21st Century

- 1. So, accelerating technological progress and globalization affect Requirements Engineering in many ways.**
- 2. Traditional centralized requirements engineering is neither effective nor necessary.**
- 3. However, that does not mean that traditional RE has become worthless over night and we can throw it all away.**
 1. In fact, very often seasoned methods and tools are far more adequate than what is currently fashionable – Software Engineering and the IT Industry are very much driven by recurring hypes following businesses motives.
 2. Historical thinking is an exception, often laughed or smiled at, but useful at times.
- 4. However, we need to face the new challenges and evaluate how they affect our field, where they're just passing fads or recurring fashions, where they are genuine and how we may adapt, and where they are actually detrimental, so we have to put up a fight to live up to our professional responsibility.**

Better, Faster, Cheaper

- **Today, most organizations are not very professional about creating software, and requirements engineering is often one of the weak spots.**
 - It would be very beneficial if we could increase the level of professionalism.
- **There are many known problems that could be addressed today, or that could be addressed in the near future with a little scientific research.**
- **Requirements specifications expressed in modeling languages like UML could be turned into formal specifications.**
 - We could verify and check such specifications much better than we can today.
 - We could use them much better to create application systems, possibly as a sequence of automatic and semi-automatic transformations.
 - By raising the abstraction level of system creation from code to models, productivity might increase drastically (cf. Assembly, C, Pascal, and 4GLs).

Requirements for Product Design

- **More and more, software is an integral part of a great variety of products. Many of the distinguishing features of products will be determined to a large degree or even exclusively by software.**
 - Even today, 70% of added value and over 90% of all innovations in upper class-cars are driven by software.
- **Software development will not start after a product has been defined, but together with product design (as an integral part), or even before, driving product design, triggering innovations.**
- **That means that Software Engineers will be concerned with more general features and requirements such as marketing, business models, and production.**
- **Thus, Requirements Engineering will also cover topics like the ecological footprint, industrial design, user experience, or ethical issues.**
 - Philips focuses on lighting.

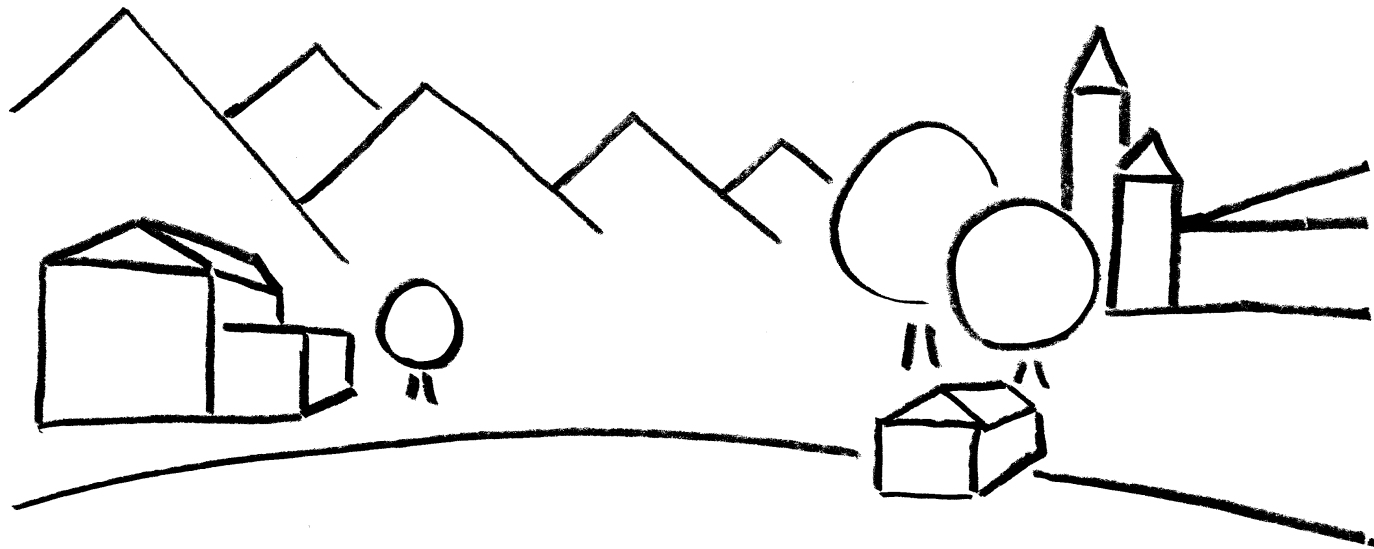
Pervasive Multimedia Requirements

- **With digital technology, many people are continuously taking pictures, films, and sound recordings – everywhere, all the time. This has been dubbed “life-logging”.** [cf. CACM 5/2001, (5)53]
 - Often, they also upload their recordings to Web 2.0 platforms like flickr, FaceBook, Twitter, and so on.
 - And they can do so as they go along. “Dead time” like waiting for the bus is being used to process and make up such data.
- **Before long, people will record each and every impression they ever have, for their whole life: we will be our own ethnographers.**
 - There will be software to index, search, mine, and analyze such life-tapes, and this could be used for requirements elicitation.
- **Just imagine the elicitation of work processes by capturing all the recordings of all the respective workers and automatically mine it for recurring patterns, important tasks, problems, time wasters, and so on.**
 - If privacy is ensured, a small reward might suffice and people might cooperate. Actually, people don't seem to care about privacy all that much...

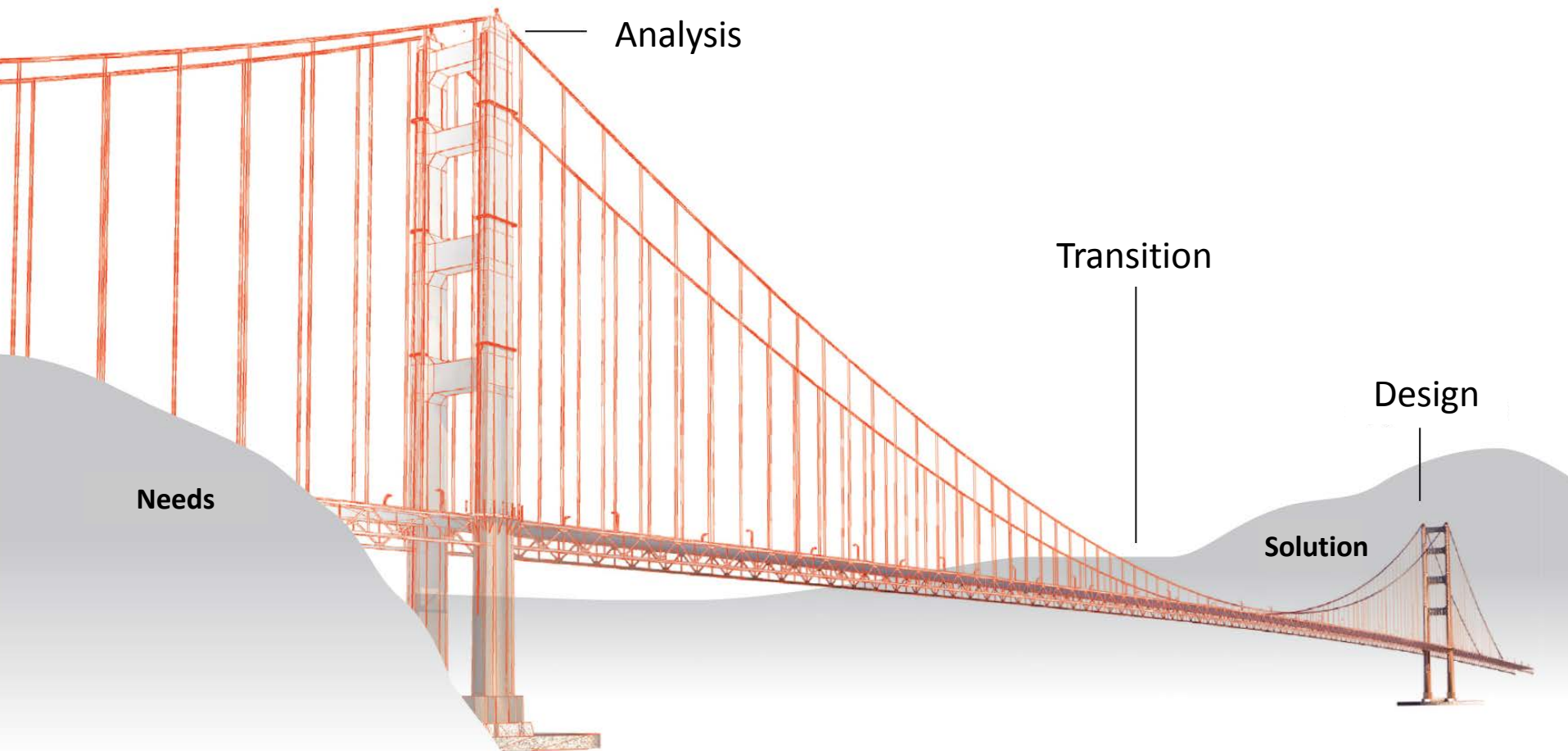
Federal Agency of Requirements Engineering

- **There has been a sad history of failures of large scale projects of all kinds, with substantial losses.**
 - If this happens to a company – too bad. If this happens to a public authority, however, it's the tax-payers money.
 - Also, these projects tend to be very big, and there are frequently heavy regulations making it ever more difficult to succeed.
- **The competencies for this kind of project could be bundled into a Federal Agency for Requirements Engineering (FARE).**
 - The people working there would be former consultants and project managers that have switched sides and have become civil servants.
 - Their services would be available to any agency or authority that is conducting a software procurement or development project.
 - They would also have a department to monitor and study the projects supported by FARE.
 - Public authorities would deal with the IT giants on a level playing field.

Adding details, step by step



It's a long road from needs to solutions





Prof. Dr. Harald Störrle

Software Engineering Section
Department of Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads
Building 322, Room 024
DK-2800 Kgs. Lyngby

hsto@imm.dtu.dk
www.imm.dtu.dk/~hsto

