SYSTEMS ENGINEERING STUDY IN GERMANY
PART 2 - BEST PRACTICES FOR SYSTEMS ENGINEERING

Dr. Martin Becker

Fraunhofer IESE
Kaiserslautern, Germany

IPA/SEC Special Seminar
Tokyo (Oct 24, 2016) and Osaka (Oct 26, 2016)

SYSTEMS ENGINEERING STUDY IN GERMANY
PART 2 – BEST PRACTICES FOR SYSTEMS ENGINEERING

- **Model-driven System Development**
- System Requirements Engineering
- System Verification and Validation
- Integrated Tool Chains
- Virtual Engineering of Systems

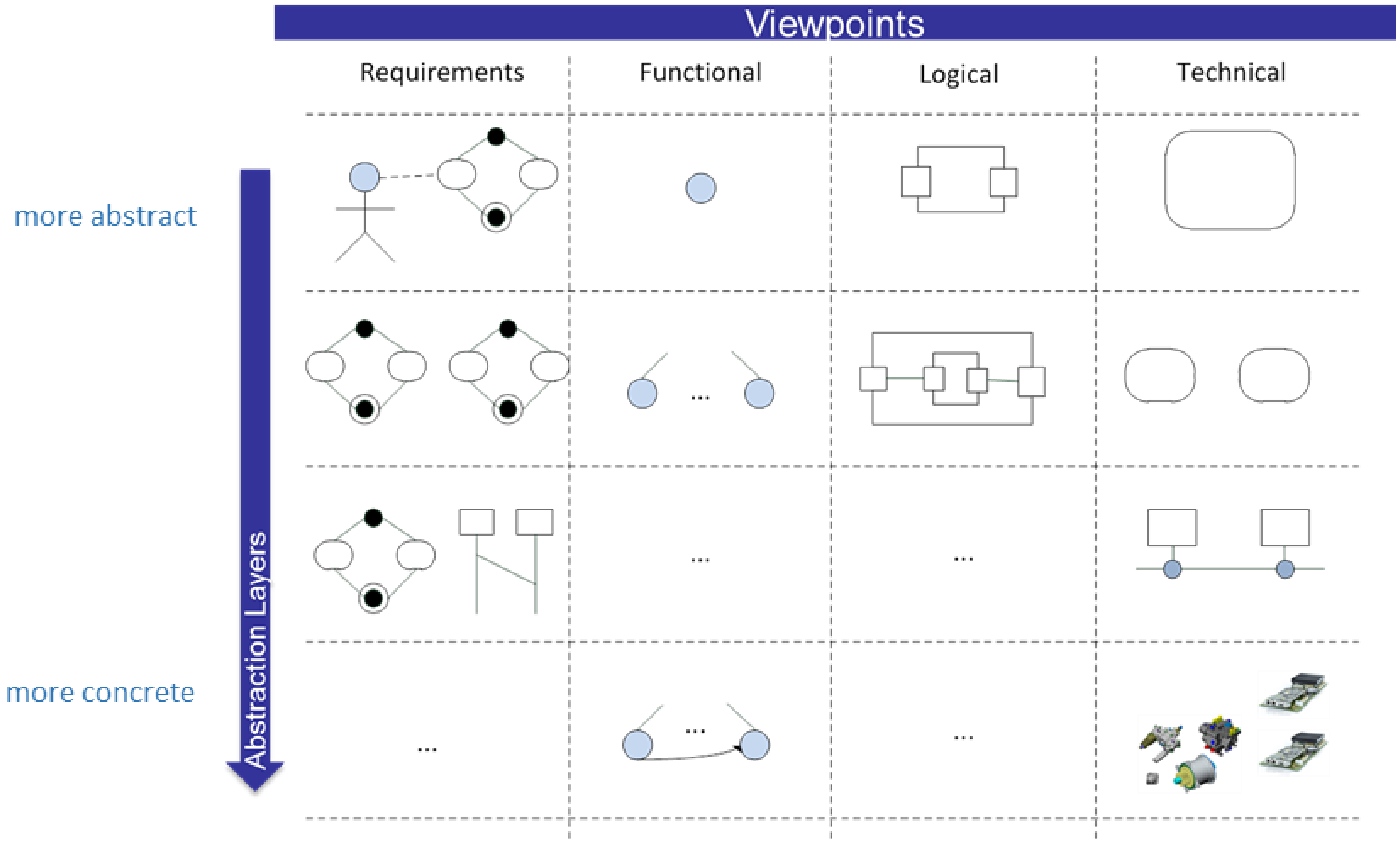
Fraunhofer
IESE

Model-driven System Development
Definition

- Ideally Provides a set of integrated modeling techniques
  - Model based requirements engineering
  - Model based design
  - Model driven implementation
  - Model based certification (e.g. Fault Trees)
- Models are refined on different abstraction levels

Fraunhofer IESE

Model-driven System Development
The SPES Methodology



Fraunhofer IESE

SPES Design Process

- Black-box system
- External entities
- Environmental properties
- Services

1 - Delineate System and its context

2 - Functional decomposition and orchestration to grant the services

- Functions
- Functional Data
- Functional Data Flow

3 – Reason on the realization (technology-independent)

- Logical Entities
- Assignment of functions to logical entities
- Logical Signals

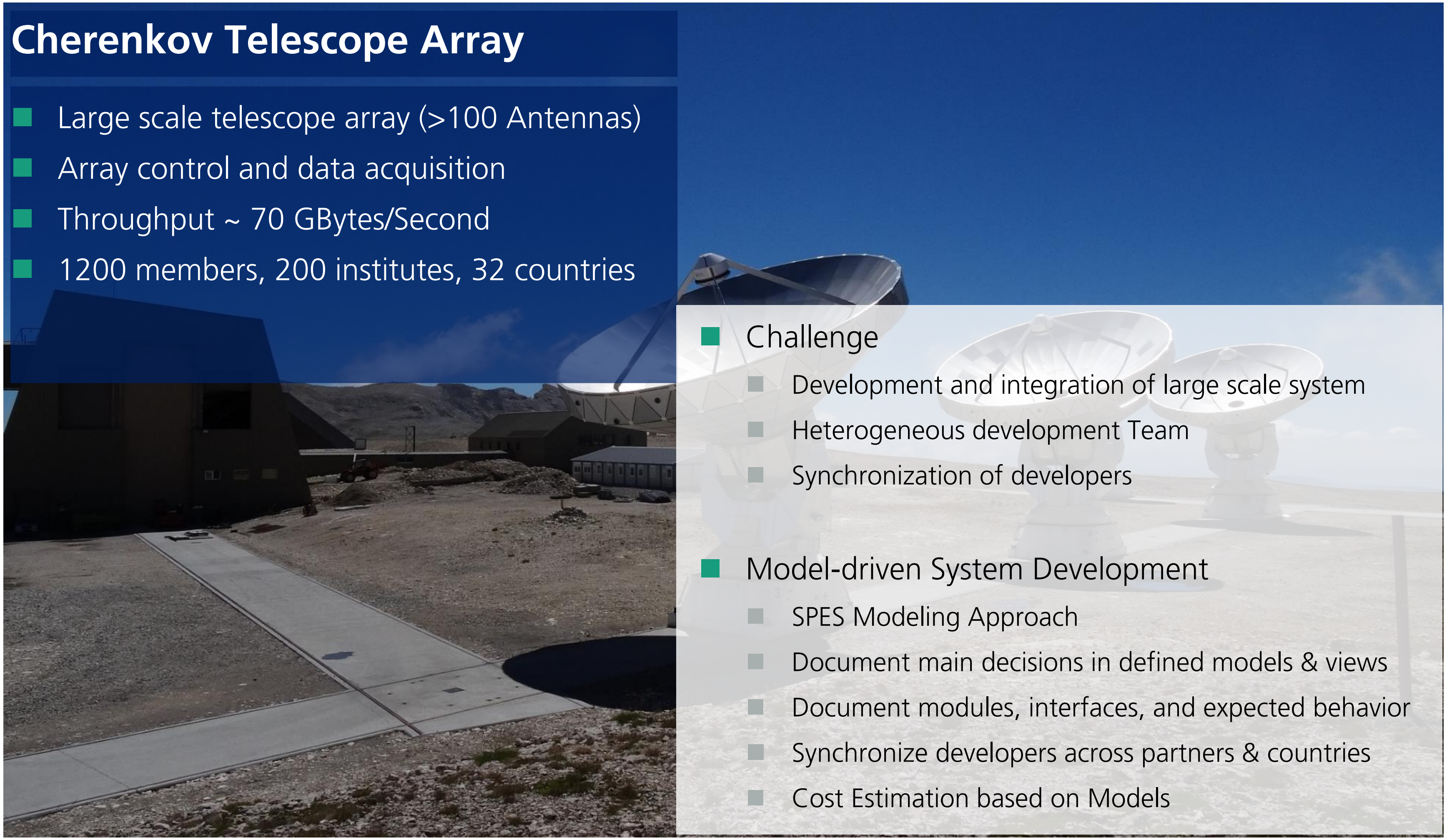
4 – HW/ Software Realization of Functions

- Software Components
- Software Interfaces
- Attributes
- Operations
- Devices
- Networks
- Buses

Fraunhofer IESE

Model-driven System Development

Practical Case



Cherenkov Telescope Array

- Large scale telescope array (>100 Antennas)
- Array control and data acquisition
- Throughput ~ 70 GBytes/Second
- 1200 members, 200 institutes, 32 countries

- Challenge
  - Development and integration of large scale system
  - Heterogeneous development Team
  - Synchronization of developers
- Model-driven System Development
  - SPES Modeling Approach
  - Document main decisions in defined models & views
  - Document modules, interfaces, and expected behavior
  - Synchronize developers across partners & countries
  - Cost Estimation based on Models

Fraunhofer IESE

Model-driven System Development
Lessons Learned and Recommendations

- Approach needs tailoring
  - e.g. modeling elements for special software entities
- Functional Model: good communication medium for non-computer scientists (such as physicists or electrical engineers)
- Training is essential
- Cost Estimation still needs time, but models make it easier to reason about the system
- Documentation based on models perceived very helpful from developers

SYSTEMS ENGINEERING STUDY IN GERMANY
PART 2 – BEST PRACTICES FOR SYSTEMS ENGINEERING

- Model-driven System Development
- **System Requirements Engineering**
- System Verification and Validation
- Integrated Tool Chains
- Virtual Engineering of Systems

Fraunhofer
IESE

System Requirements Engineering
Definition

- Several institutions provide definitions of the term Requirements Engineering (RE)
  - IREB (International Requirements Engineering Board): Requirements engineering is the systematic and methodologically sound approach to requirements analysis and management
  - IEEE: Requirements Engineering is the branch of systems engineering concerned with managing desired properties and constraints of software-intensive systems and with goals to be achieved in the environment. It is concerned with these aspects from the problem analysis stage to the implementation and maintenance stages of a system.
- Goal: Develop good requirements and manage them during development considering risks and quality
- RE bridges the entire life cycle and thus determines the success or failure of a product or project

System Requirements Engineering
Example Approach (1)

- Processes used in RE vary widely, depending on
  - the application domain
  - the people involved
  - the organization developing the requirements
- Common generic activities can be found in all RE processes:
  - Requirements Elicitation: The process of discovering, reviewing, documenting, and understanding the user’s needs and constraints for a system.
  - Requirements Analysis: The process of refining the user’s needs and constraints.
  - Requirements Validation: The process of ensuring that the system requirements are complete, correct, consistent, and clear.
  - Requirements Specification: The process of documenting the user's needs and constraints clearly and precisely.
  - Requirements Management: The process of scheduling, coordinating, and documenting the requirements engineering activities (that is, elicitation, analysis, specification, and verification)

System Requirements Engineering

Example Approach (2)

- Lots of different techniques can be applied to execute generic activities
- Example: Selection of appropriate requirements elicitation technique using selection matrix
- Model-based requirements engineering has been invented to overcome inadequacies of common techniques such as:
  - misunderstanding in communications
  - continuously changing requirements
  - low quality because of time pressure

Key:

–	not recommended
0	no influence => may be used
+	recommended
++	highly recommended

	Brainstorming	Method 6-3-5	Change of perspective	Field observation	Apprenticing	Questionnaire	Interview	System archeology	Reuse
Human influences									
Stakeholders lack motivation (to participate actively)	–	–	–	+	–	0	+	++	++
Lack of communication skills	–	–	–	++	++	–	+	++	++
Abstract thinking ability deficient	–	–	–	++	++	0	+	++	++
Many different opinions	+	++	+	++	++	+	0	0	0
Imbalance in power between involved parties	–	+	–	0	0	0	0	0	0
Problematic group dynamics	–	+	+	0	0	0	0	0	0
Organizational influences									
Development for a complex market	++	+	+	–	–	++	0	+	0
Fixed, tight project budget	++	++	+	+	–	–	+	–	++
Stakeholders physically far apart from each other	–	0	–	0	0	++	0	0	0
Poor availability of the stakeholders	+	+	–	+	–	+	++	++	++
High number of stakeholders	+	–	+	0	–	++	0	0	0
Technical influences									
Highly critical system	0	0	+	++	–	+	+	++	+
System has a large scope	0	0	0	+	–	–	+	++	+
No previous experience in the domain	0	0	0	–	+	–	+	++	+
Trying to find rough requirements	++	++	+	+	0	+	++	–	0
Trying to find detailed requirements	+	+	+	+	++	–	+	++	+
Non-functional requirements wanted	0	0	0	0	+	–	+	+	+
Very complex system	0	0	0	+	–	–	+	+	+

SOPHIST

Fraunhofer IESE

System Requirements Engineering
Practical Case

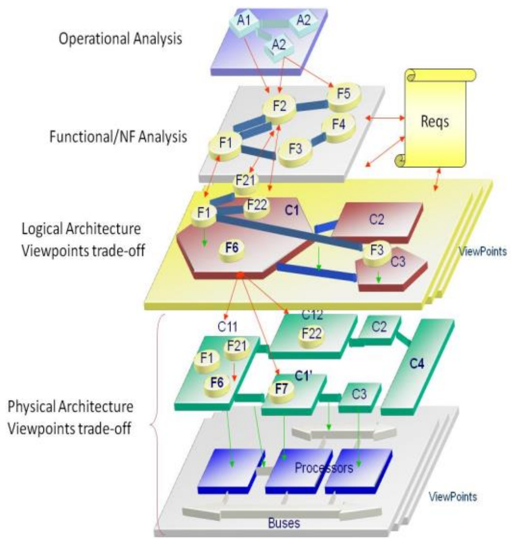
- German Ministry of Defense: Analysis phase for a new modular multipurpose combat ship class MKS 180
  - Integrated RE and System Architecture Model

Requirements Model

What?

How?

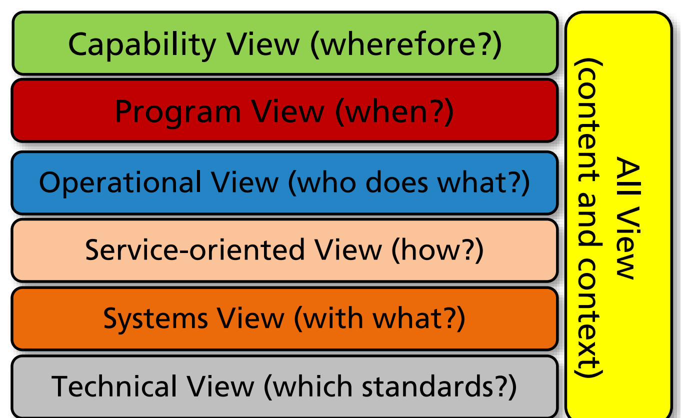
With what?



Customer Product Management

Synchronization with integrated tool chain

System Model (Views)



Strategic Context

Operational Context

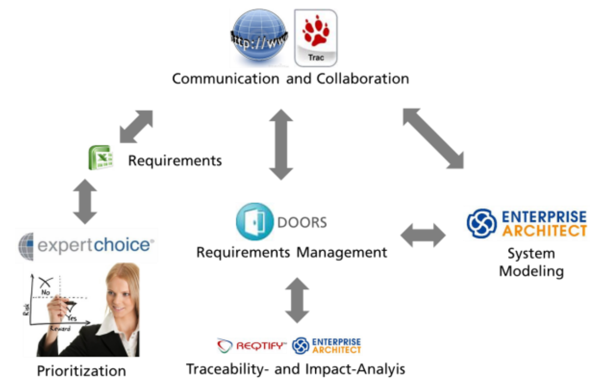
Technical Context

NATO Architecture Framework

Fraunhofer IESE

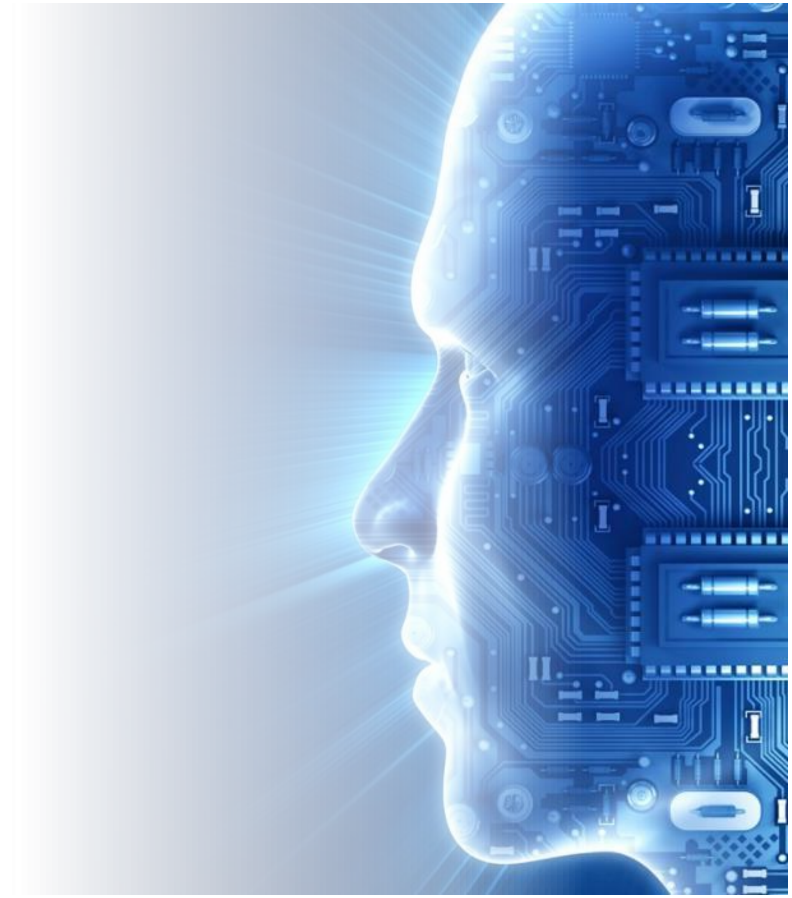
System Requirements Engineering
Lessons Learned and Recommendations

- Lessons learned
  - Selection of appropriate requirements elicitation techniques
  - Analyzing and refining users needs and constraints involves original stakeholders again ⇒ Support by tool chain
  - Validation can be supported by standardized quality measures such as IEEE:830 ⇒ Support by tool chain
  - RE-tool should be customizable and extensible to fit into tool chain
  - Requirements change management with specific processes and tools ⇒ tool chain
- Recommendation
  - Integrated tool chain facilitates project processes and RE/System modeling processes



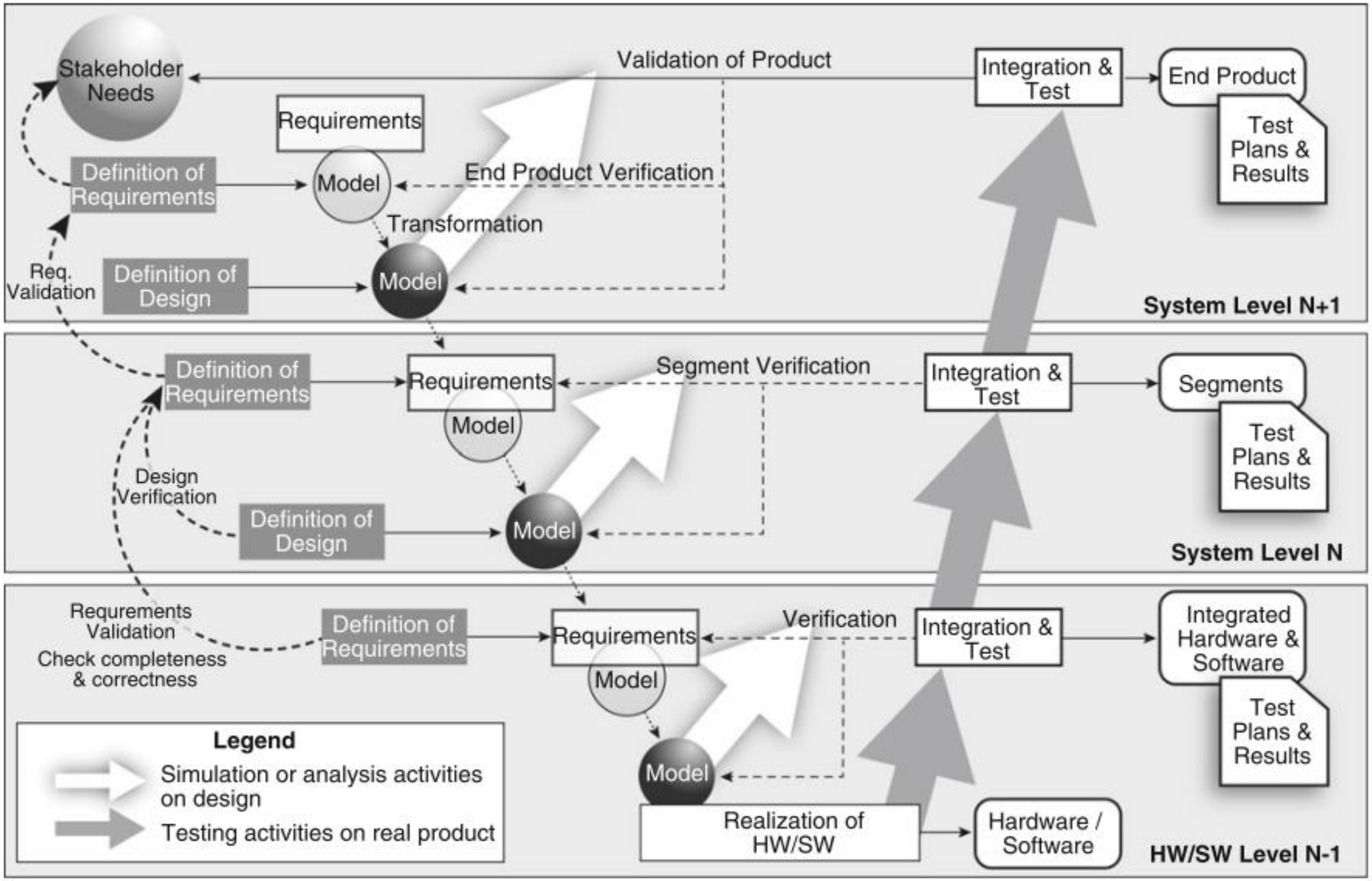
SYSTEMS ENGINEERING STUDY IN GERMANY
PART 2 – BEST PRACTICES FOR SYSTEMS ENGINEERING

- Model-driven System Development
- System Requirements Engineering
- **System Verification and Validation**
- Integrated Tool Chains
- Virtual Engineering of Systems



Fraunhofer IESE

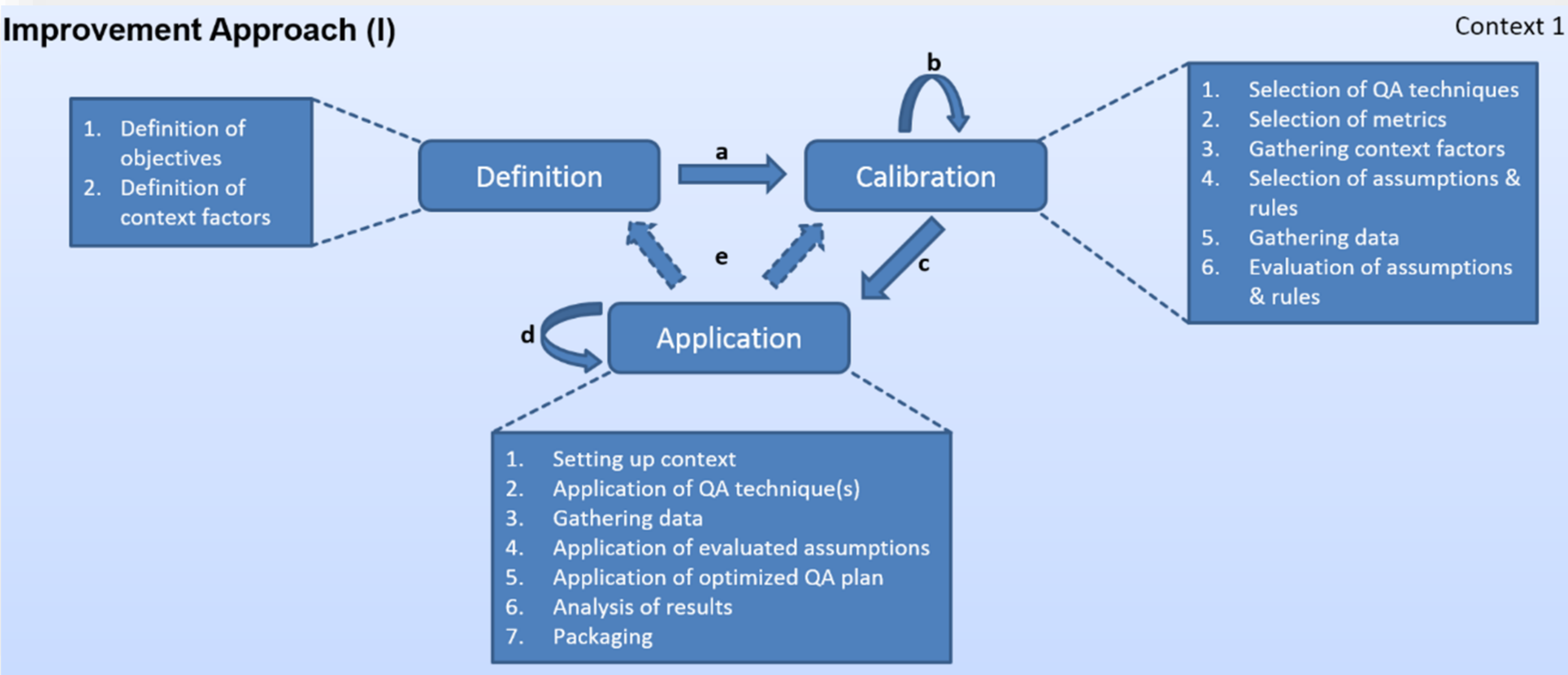
System Verification and Validation
Definition



[Source: CESAR Book, Springer]

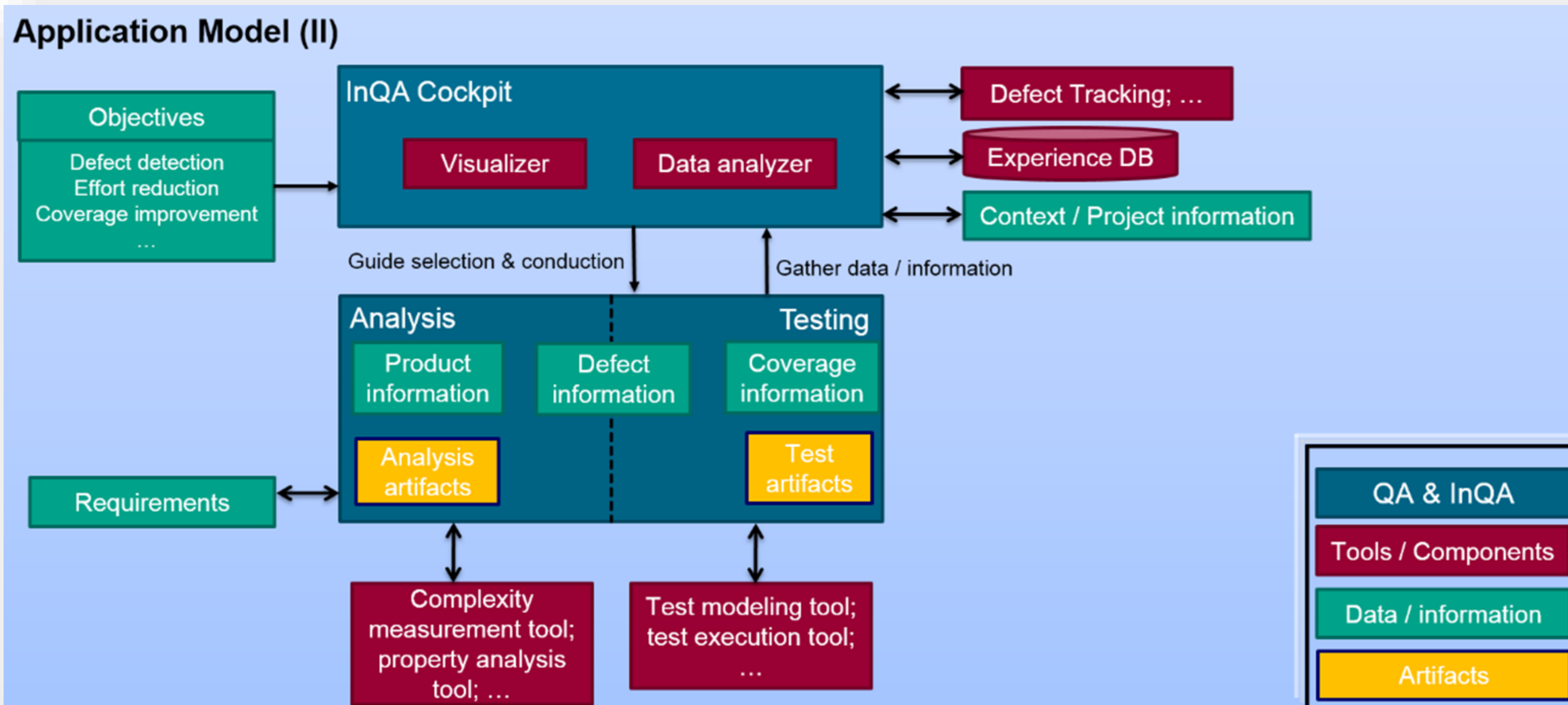
System Verification and Validation
Example Approach

- Approach: Integrated Quality Assurance approach (InQA)

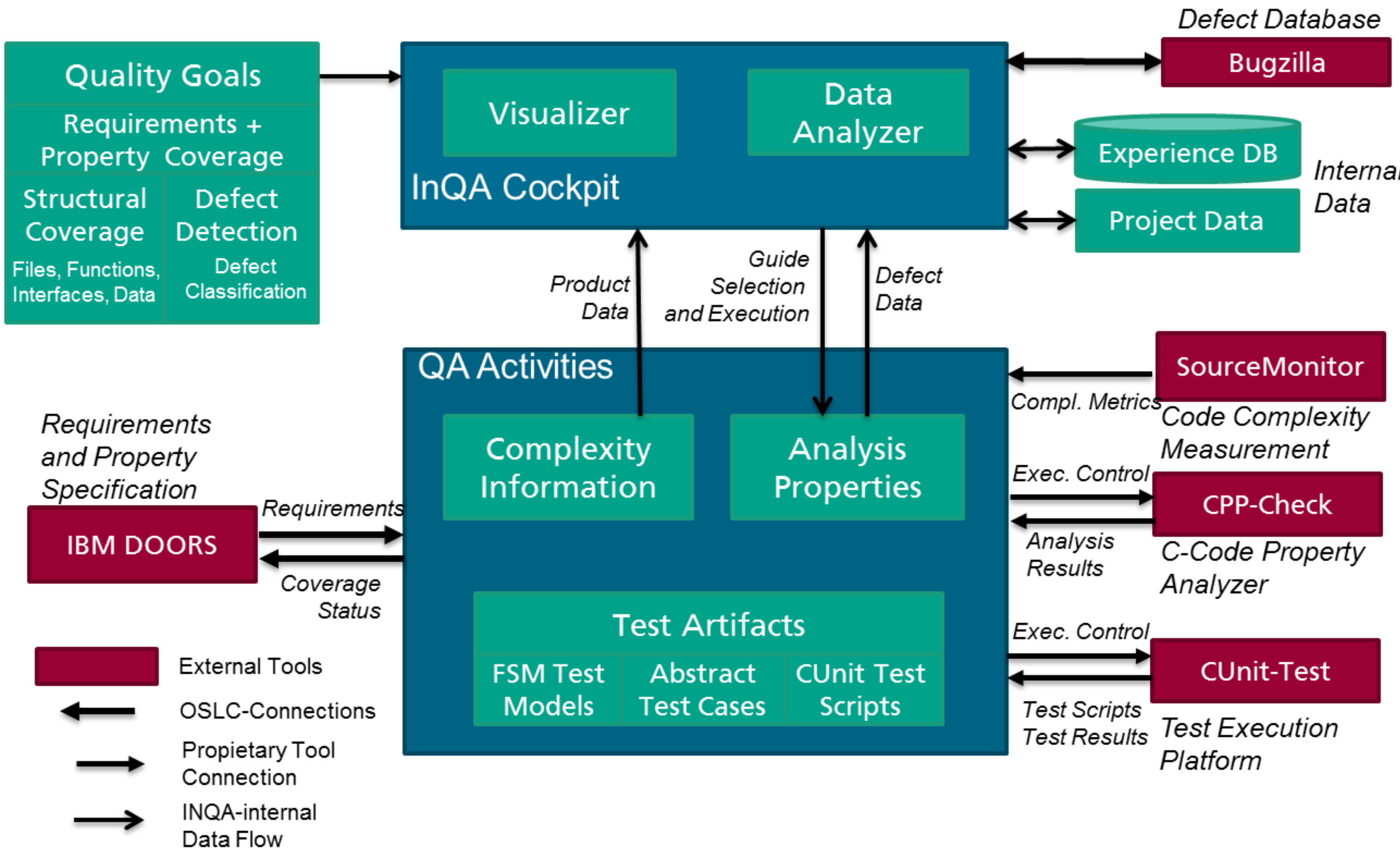


System Verification and Validation
Example Approach

- Approach: Integrated Quality Assurance approach (InQA)



System Verification and Validation
Practical Case



System Verification and Validation
Practical Case

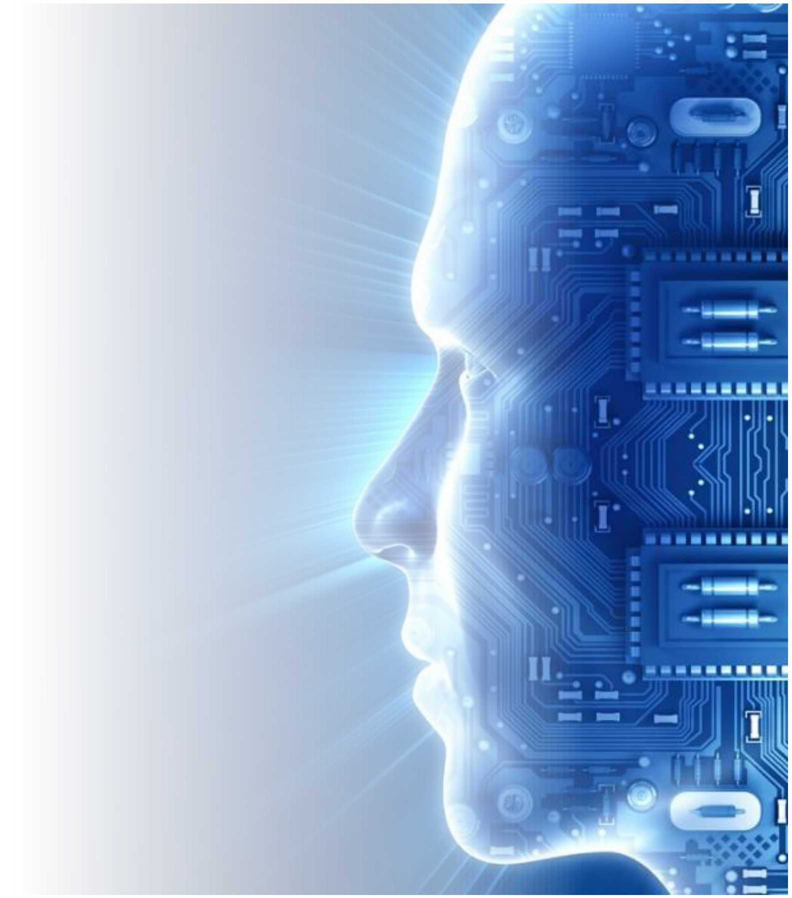
- Applied in European research project MBAT (model-based analysis and testing)
- 39 organizations, 8 countries
- Goal: improve quality assurance process by combining analysis, verification, and testing techniques
- 13 industrial use cases
  - Domains: automotive, avionics, and rail systems
  - Different contexts, settings, problems, quality properties, QA stages
  - Examples:
    - Daimler (light control subsystem)
    - Volvo (brake-by-wire subsystem)

System Verification and Validation
Lessons Learned and Recommendations

- Large scale industrial evaluation (13 case studies)
- Only aggregated evaluation data was published
- Detailed use case-specific data was classified as confidential and not published outside the project.
- Significant cost reduction
  - Costs for application of verification and validation techniques could be significantly reduced by 32%
  - Costs caused by remaining defects in subsequent development stages could be also reduced by an average of 27%
- System quality was also improved
  - e.g. test coverage and post-release defects were improved by at least 8%

SYSTEMS ENGINEERING STUDY IN GERMANY
PART 2 – BEST PRACTICES FOR SYSTEMS ENGINEERING

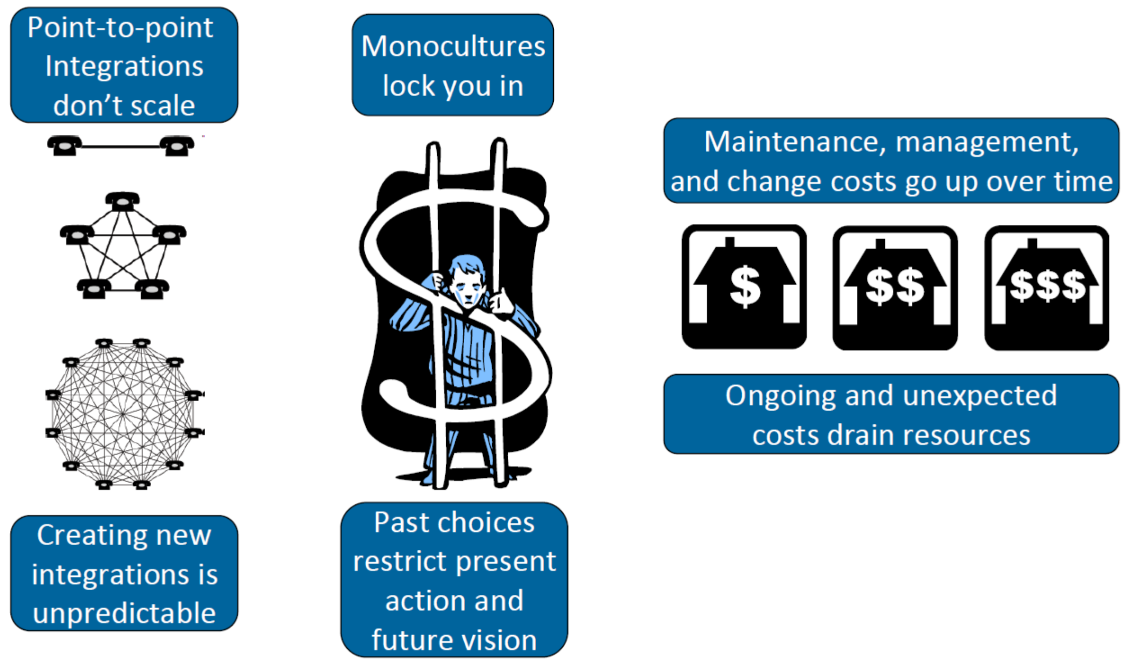
- Model-driven System Development
- System Requirements Engineering
- System Verification and Validation
- **Integrated Tool Chains**
- Virtual Engineering of Systems



Fraunhofer IESE

Integrated Tool Chains
Definition

- Broad heterogeneity of engineering methods, tools, and data involved
- Bridge the gap between development platforms and operational ones
- Distributed and multi-tier nature of development teams



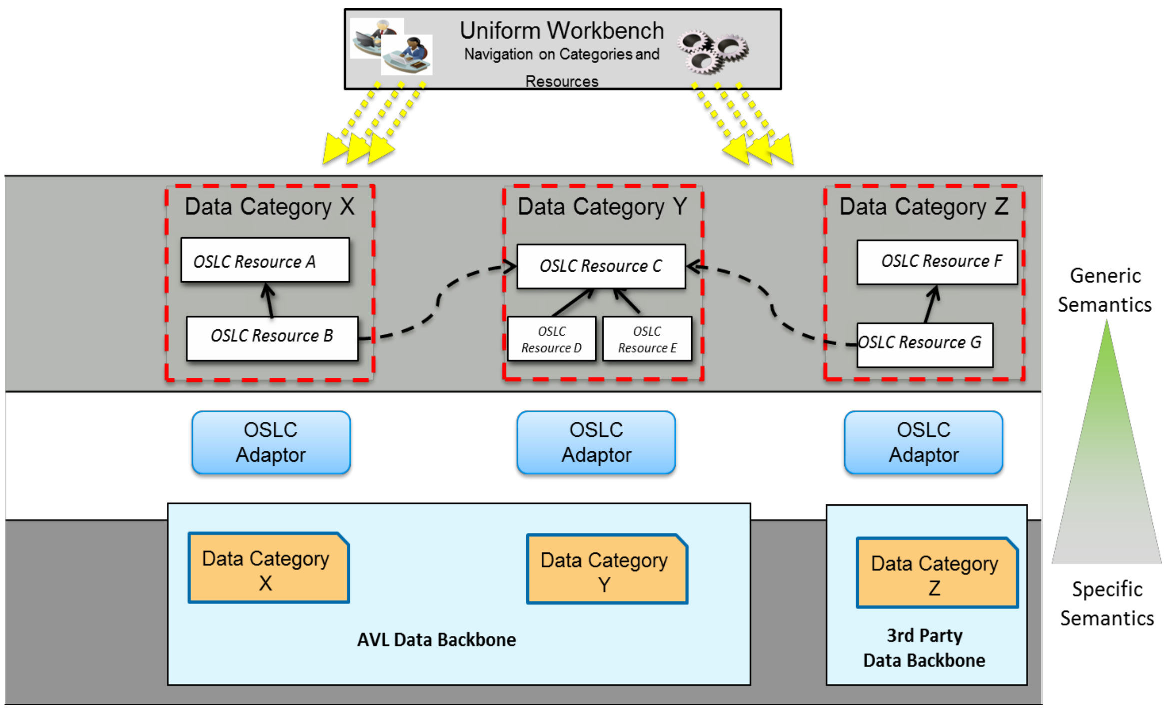
[Source: http://www.crystal-artemis.eu]

Need to interoperate seamlessly in today's (still fragmented) tool landscapes

Fraunhofer IESE

Integrated Tool Chains
Example Approach

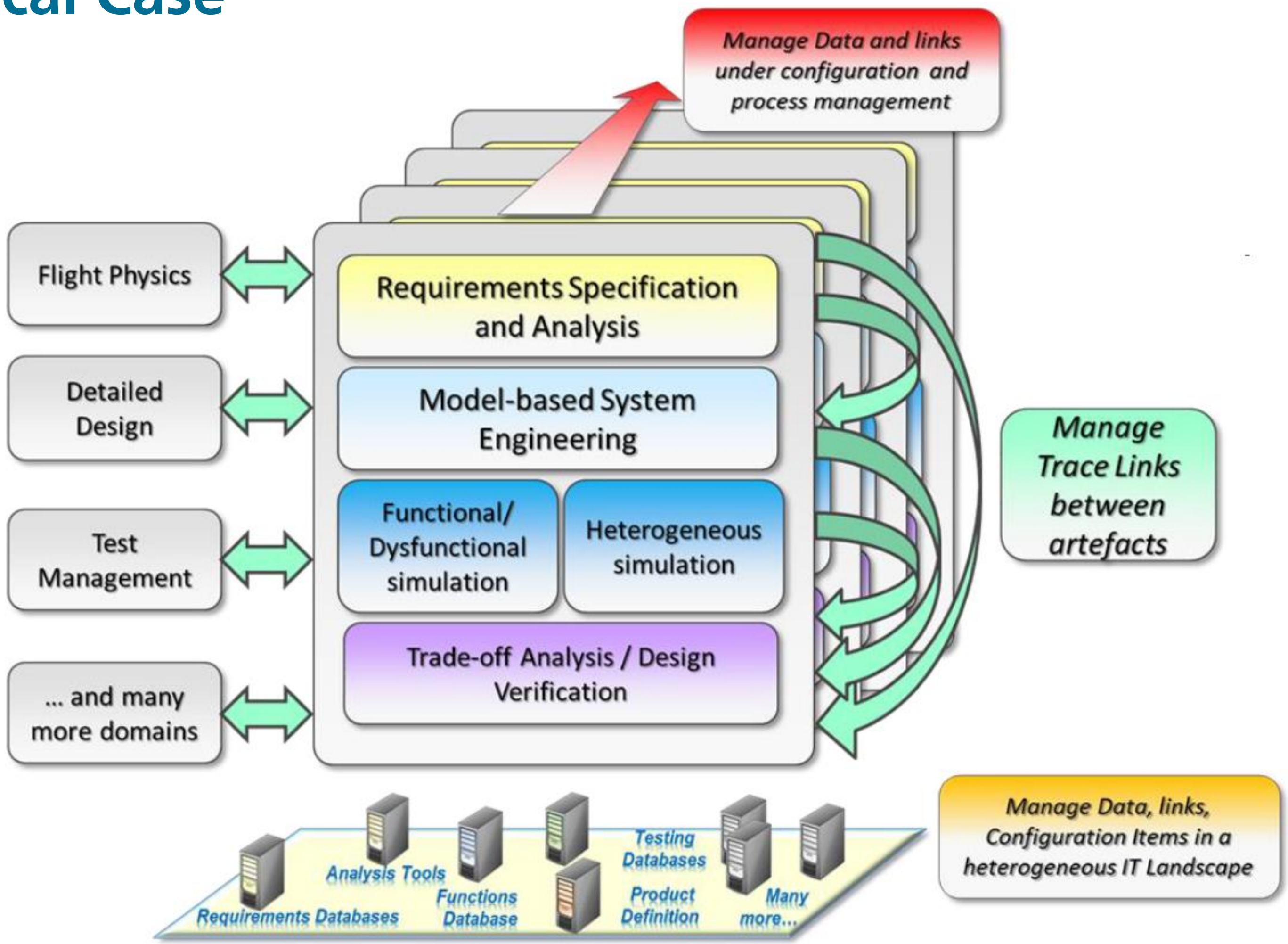
- OSLC (Open Services for Lifecycle Collaboration)
- OSLC defines a set of specifications focusing on the support of life cycle activities



[Source: http://www.crystal-artemis.eu]

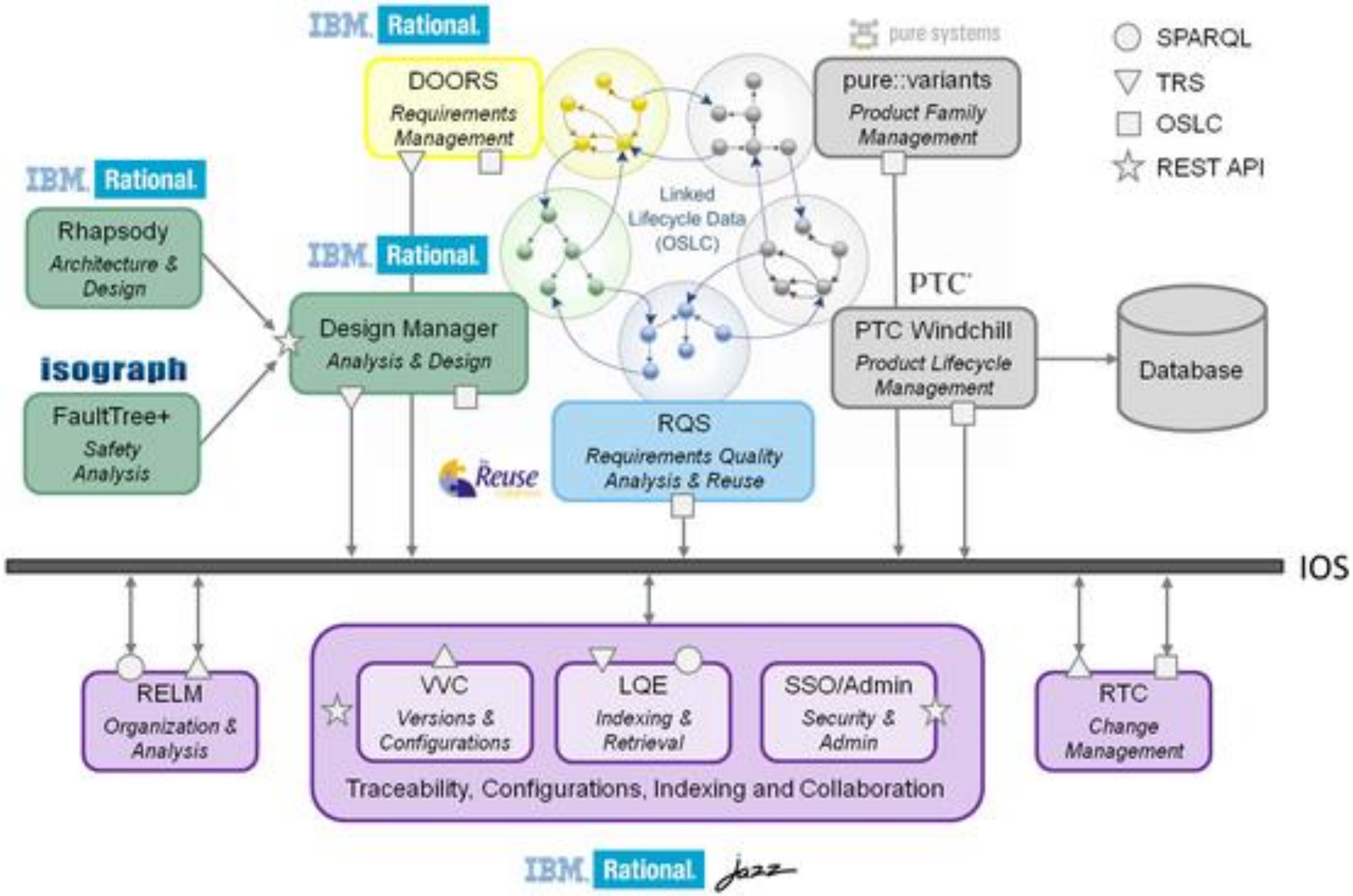
Fraunhofer IESE

Integrated Tool Chains
Practical Case



[Source: http://www.crystal-artemis.eu/fileadmin/user_upload/Deliverables/CRYSTAL_D_208_903_v3.03.pdf]

Integrated Tool Chains
Practical Case



[Source: http://www.crystal-artemis.eu/application-domains/aerospace/work-package-203.html]

Fraunhofer IESE

Integrated Tool Chains
Lessons Learned

- Tool interoperability can be improved substantially based on OSLC, and has advanced significantly in recent years
- An increasing number of tool providers such as IBM, PTC, PureSystems, Siemens, etc., are providing standardized interfaces to interoperate with other tools and have shown compelling tool interoperation scenarios among tools from different tool providers
- Certain complexities regarding the setup and maintenance of the tool adapters and data also became apparent in the use cases
- Link management: OSLC does not really specify where links should be managed and how – this is completely up to the developer of the interfaces
- It is quite challenging to implement an OSLC interface for an existing tool because the availability of the source code is a prerequisite

Fraunhofer IESE

Integrated Tool Chains
Recommendations

- Tool interoperability still remains an issue to be investigated in detail on a tool-by-tool basis in a concrete setting. The generalization of meta models and tool interfaces for the typical interoperation scenarios is work in progress and it's unclear whether this will be achieved at all
- Following a use-case-driven approach to improve tool interoperation is a good practice to identify shortcomings and value-adding improvements in the tool chains in a systematic and measurable way
- Beyond tool interoperation, open data formats can also help to archive important data without facing the challenge of having to reinstall complicated tool infrastructures in order to access the data of past projects

SYSTEMS ENGINEERING STUDY IN GERMANY
PART 2 – BEST PRACTICES FOR SYSTEMS ENGINEERING

- Model-driven System Development
- System Requirements Engineering
- System Verification and Validation
- Integrated Tool Chains
- **Virtual Engineering of Systems**

Fraunhofer IESE

Virtual Engineering of Systems
Definition

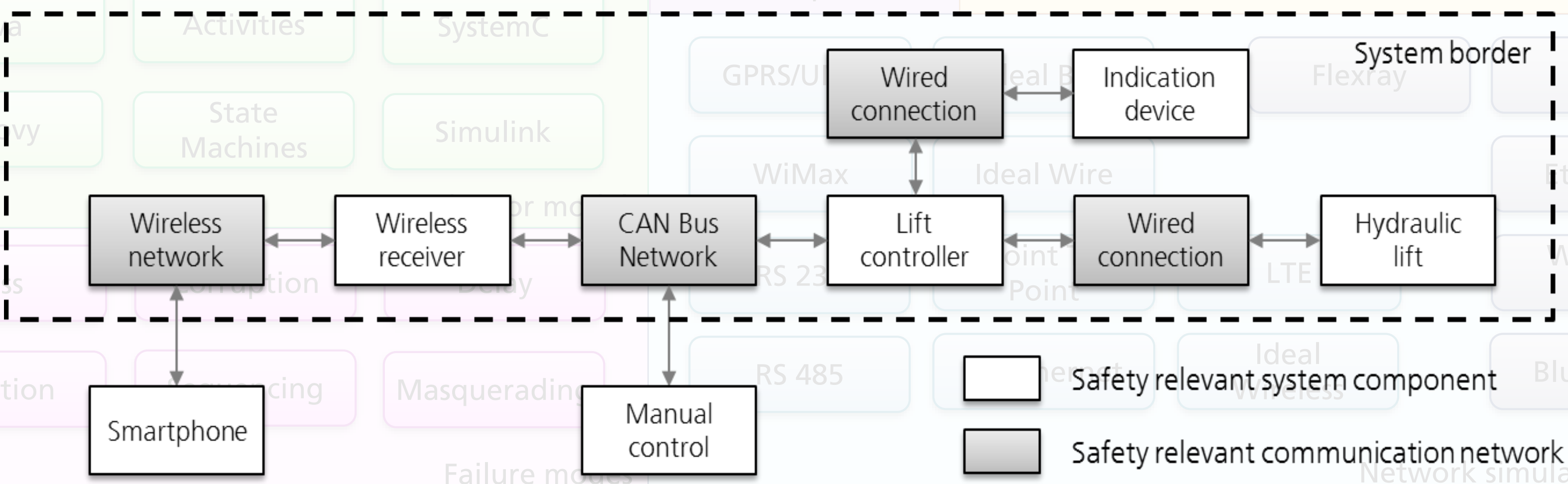
- Virtual engineering substitutes real artifacts with simulation models
- Substituted artifacts may be mechanical parts that are substituted by CAD models, hardware platforms under development that are substituted by virtual platforms and software implementations that are substituted e.g., by Simulink behavior model
- Main purpose
  - ensure important properties and features of artifacts under development. Virtual prototypes are available much earlier.
  - enables quantitative evaluations much earlier than if implementations/realizations
- Often different simulators need to be coupled into holistic system simulations

Virtual Engineering of Systems
Example Approach

- Simulator coupling is difficult because the models of computation and communication (MOCCs) of simulation models often differ
- Three common MOCCs are Discrete Time, Discrete Event, and Continuous Time models.
- Common application in industry is the virtual evaluation of Electrical/Electronic (E/E) architectures. E/E architectures consist of hardware control units that are connected by networks.
  - Task deployment on a processors
  - Network communication
  - Safety mechanisms

Virtual Engineering of Systems
Practical Case

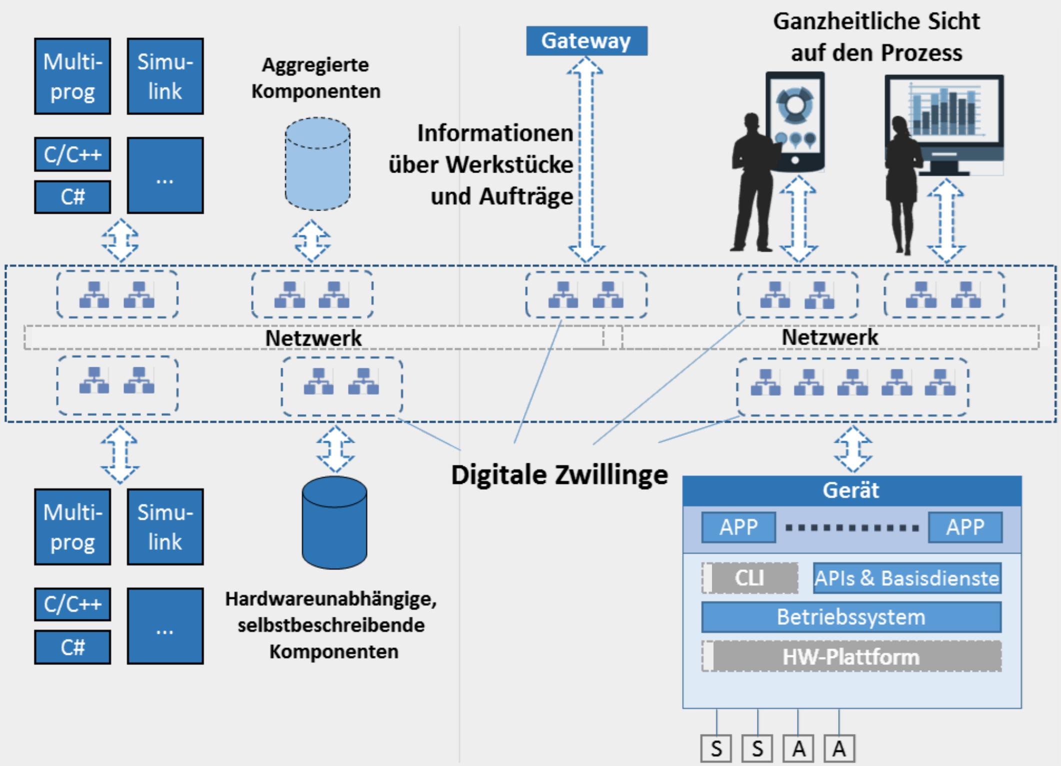
- Remote Lift Control (anonymized application)
  - Remote control of safety function with smart phone over wireless network
  - Virtual evaluation of system and safety concept prior to implementation
  - Gradual integration of models (specification) and C/C++ code (realization)



Virtual Engineering of Systems
Virtual Deployments

Virtual Development of Software Architectures

- System Architecture Prototype
  - High-Level models similar to UML/SysML
  - Executable and unambiguous
  - Synchronize developers
- Development Team Agreement
  - High-Level & Implementation interfaces
  - Data flows in system
  - Main features
- Refinement and integration
  - Change impact analysis
  - Change management support



[Source: BASYS project]

Fraunhofer IESE

Virtual Engineering of Systems
Lessons Learned and Recommendations

- Develop revolutionary concepts that are not an evolution of existing approaches, but rather realize new ideas
- Ability to evaluate critical aspects early and without risks in simulation in conjunction with the increasing speed and accuracy of simulation models continuously increases the importance and applicability of virtual engineering techniques
- Considering the increasing system complexity and architecture, integration testing should start as soon as possible in the development process. Virtual Hardware-in-the-Loop testbeds, created by coupling existing simulators, should be considered as an efficient approach

Thanks!

どうもありがとう

Dōmo arigatō

Dr. Martin Becker

Department Head
Embedded Systems Engineering

Fraunhofer IESE
Fraunhofer-Platz 1
67663 Kaiserslautern
Germany

Phone: +49 (0) 631-6800-2246
Fax: +49 (0) 631-6800-9-2246

martin.becker@iese.fraunhofer.de

© Japanese Garden in Kaiserslautern, 2015
Partner City of Bunkyo-ku, Tokyo

Japanischer Garten Kaiserslautern

[Picture from http://www.japanischergarten.de]

Fraunhofer IESE