Requirements Engineering for Software vs. Systems in General?

Institut für Computertechnik

ICT

Institute of Computer Technology

Moderator:

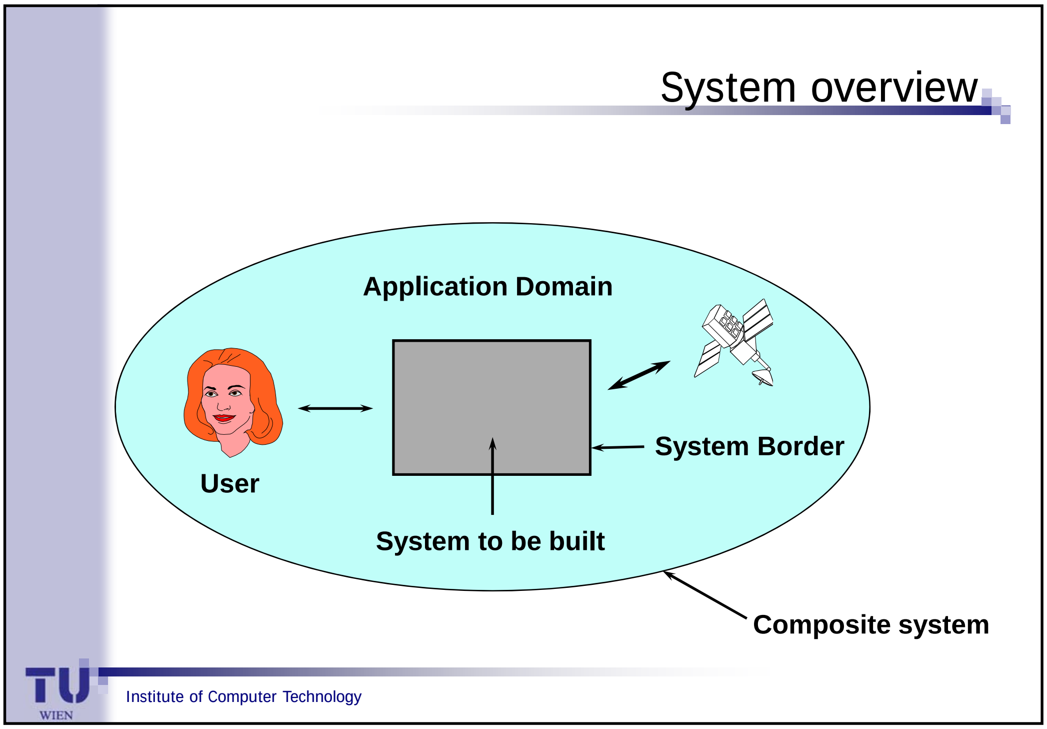
Hermann Kaindl

Vienna University of Technology, ICT

Panelists

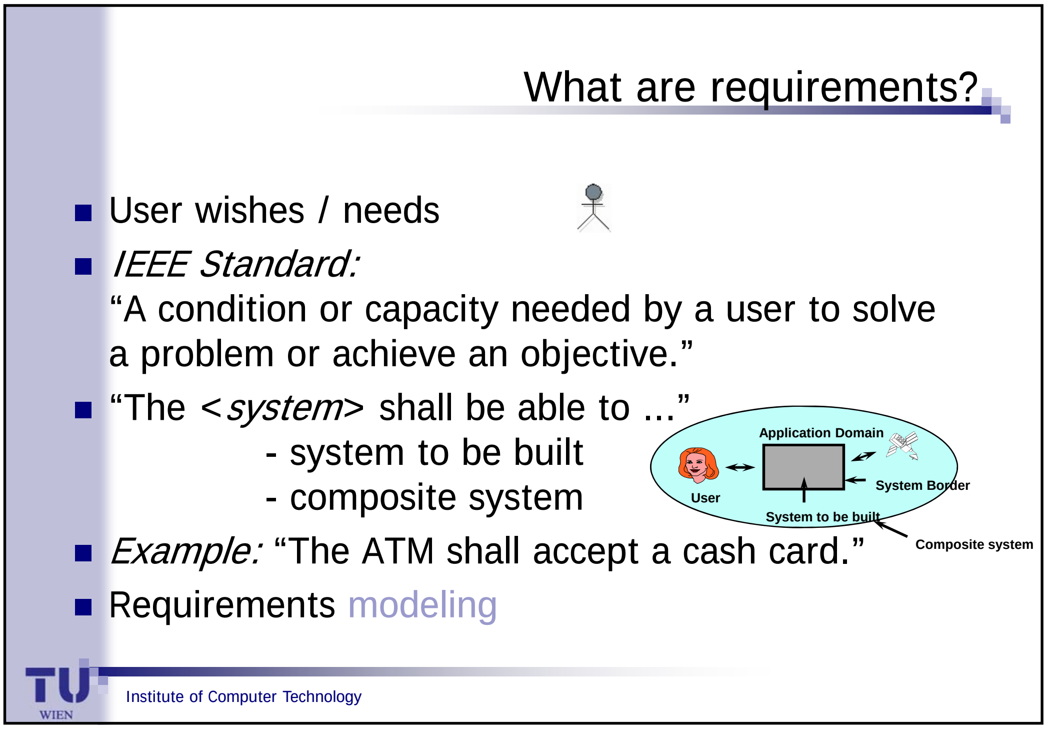
Marko Jäntti	University of Eastern Finland
Herwig Mannaert	University of Antwerp
Kazumi Nakamatsu	University of Hyogo
Roland Rieke	Fraunhofer Institute for Secure Information Tech.

Institute of Computer Technology



System overview

Institute of Computer Technology



What are requirements?

- User wishes / needs
- *IEEE Standard:*
  "A condition or capacity needed by a user to solve a problem or achieve an objective."
- "The <*system*> shall be able to ..."
  - system to be built
  - composite system
- *Example:* "The ATM shall accept a cash card."
- Requirements modeling

Application Domain
User
System Border
System to be built
Composite system

Institute of Computer Technology

Fundamental technical differences?

Types of requirements:

- *Functions:* effects achieved by some entity
- *Behavior:* state change over time
- *Structure:* arrangement or relationship of elements in a system (physical or logical structure)
- *Constraints:* restrictions or limitations

All of them equally relevant for software and systems in general?

Institute of Computer Technology

Structural requirements

- Function to support someone when sitting on a chair only through its physical structure
- Function to fly related to the s (aerodynamics)
- Structural requirements on sof
- Software architecture and subs
- Constraint requirement
- Difference between software a

Institute of Computer Technology

Thank you for your attention!

???

Institute of Computer Technology

Requirements Engineering for Software vs. Systems in General

Reasoning-based Intelligent Systems

Kazumi Nakamatsu
University of Hyogo
JAPAN

Viewpoint as Reasoning

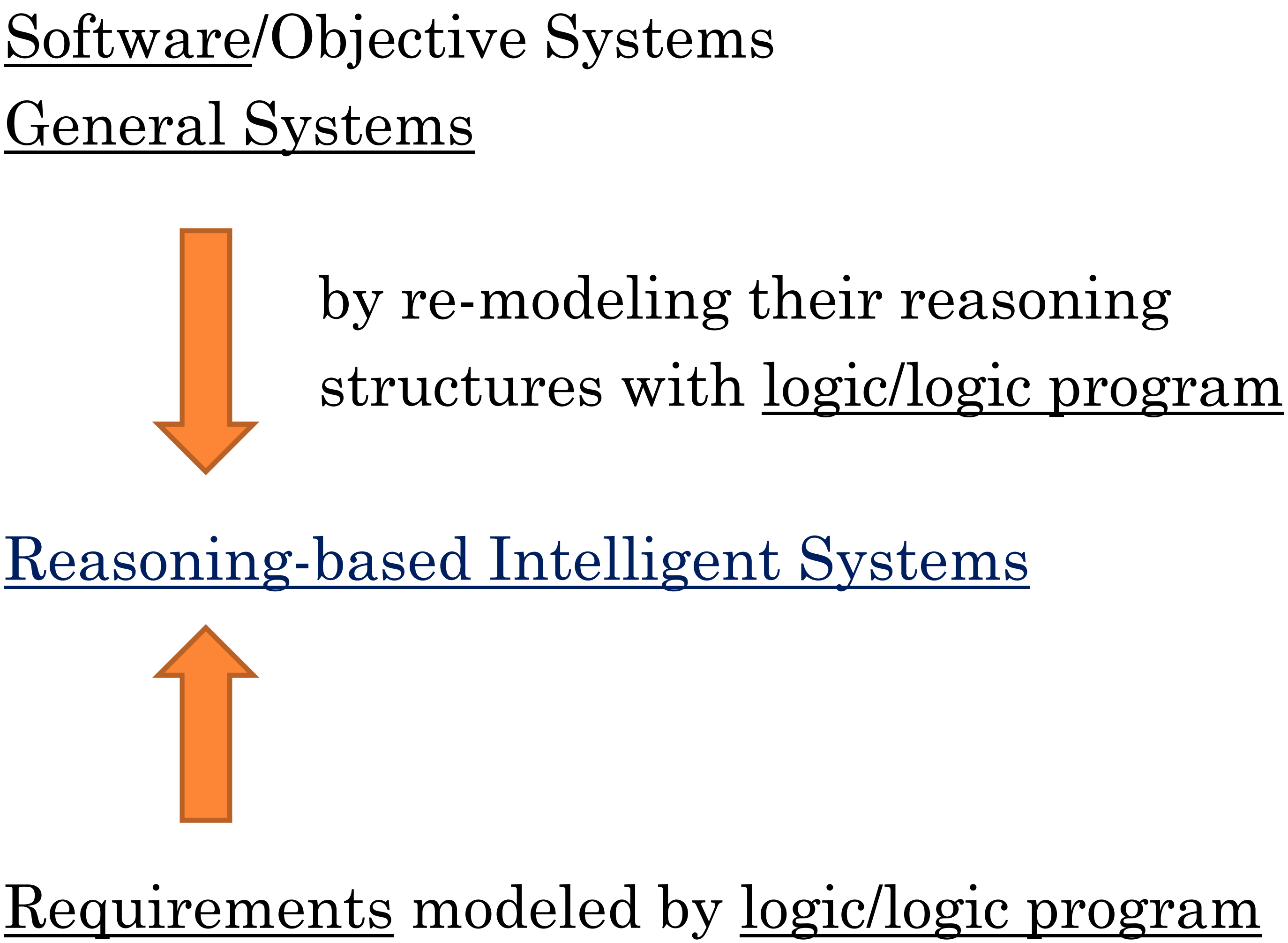
Software

Automated reasoning systems implemented on electronic devices,

which are functional models of objective systems

General Systems

More human-like systems maybe including interactions such as man-machine/man-man



Software/Objective Systems

General Systems

by re-modeling their reasoning structures with logic/logic program

Reasoning-based Intelligent Systems

Requirements modeled by logic/logic program

Examples

*Railway interlocking system

(software on relay/electronic device)

whose basic reasoning part can be modeled by classical logic.

Requirements: assuring safety

*Trial system (decision system: guilty or not)

whose basic reasoning part can be modeled by plausible logic.

Requirements: mutual understanding between professional judges and citizen judges

Development of Annotated Logic Programs

Annotated Logic by da Costa and Subrahmanian

Annotated Logic Program by Subrahmanian et al.

ALPSN (Annotated Logic Program with Strong Negation)

--- non-monotonic reasoning
Eg. default, autoepistemic reasoning, etc.

VALPSN (Vector Annotated Logic Program with Strong Negation)

--- defeasible reasoning Eg. conflict resolving

EVALPSN (Extended Vector Annotated Logic Program with Strong Negation)

--- defeasible deontic reasoning Eg. various controls,
logical verification

Bf-EVALPSN (Before-after EVALPSN)

--- before-after relations between time intervals(processes)

MY OPINION

There is no fundamental difference

between

Requirement Engineering for
Software

and

Systems in General

Marko Jäntti

Panel: Requirements Engineering for Software vs. Systems in General

TEKES

UNIVERSITY OF EASTERN FINLAND

Software / System / Service Requirements

INFORMATION SYSTEM REQUIREMENTS

Hardware

Software

People

Infrastructure

NON-FUNCTIONAL REQUIREMENTS

USABILITY REQUIREMENTS

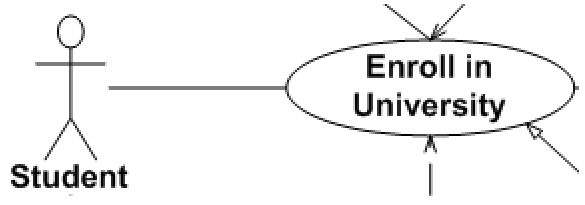
SERVICE REQUIREMENTS

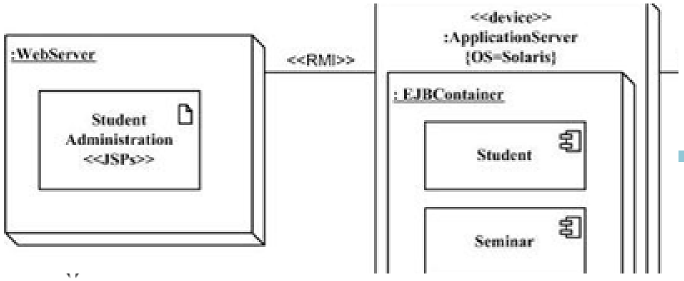
Services

FUNCTIONAL REQUIREMENTS

Service availability

UML




:WebServer
Student Administration <<JSPs>>
<<RMI>>
<<device>> :ApplicationServer {OS=Solaris}
: EJBContainer
Student
Seminar

Service Strategy

Service Design

Service Transition

Service Operation

Continual Service Improvement

IT Infrastructure Library v3

Example: Service desk tool requirements

- **Functional requirement :** Create a support request
- **Non-functional requirement:** Data security. Cases from Customer Y have to be handled by Team T.
- **Usability requirement:** Submitting a support request should be done within 5 minutes
- **Hardware requirement:** User should be able to create a request via smart phone (android, windows phone)
- **Infrastructure/integration:** The system needs to have an interface to the service provider X's service desk tool
- **Service requirement:** Availability of support system 24/7

Panel: Requirements Engineering for Software vs. Systems in General – Security and Reliability –

Roland Rieke

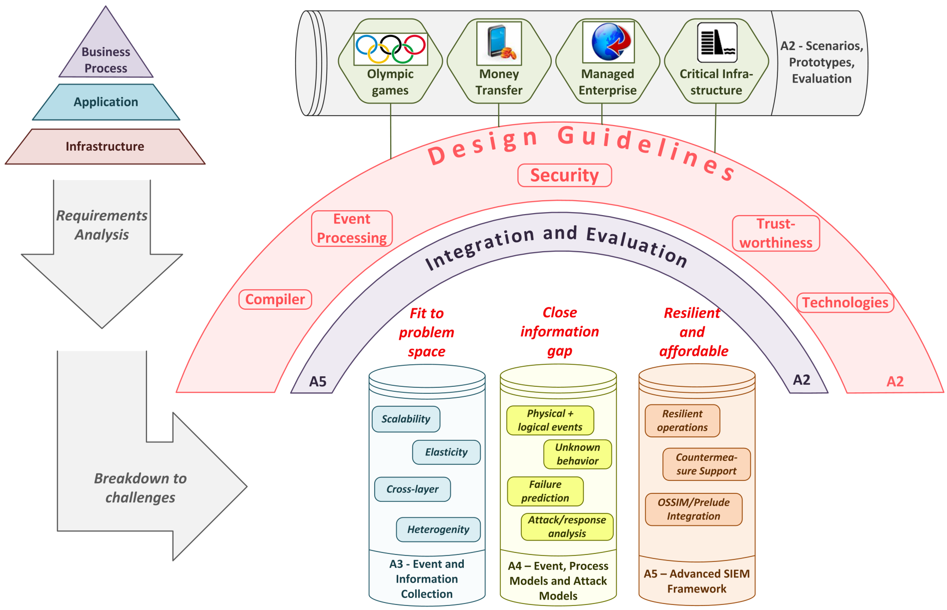
roland.rieke@sit.fraunhofer.de

Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany

ICONS, March 2012

Fraunhofer SIT

Requirements-driven System Design in Project MASSIF



Business Process

Application

Infrastructure

Requirements Analysis

Breakdown to challenges

Olympic games

Money Transfer

Managed Enterprise

Critical Infra-structure

A2 - Scenarios, Prototypes, Evaluation

Design Guidelines

Security

Event Processing

Trust-worthiness

Compiler

Technologies

Integration and Evaluation

A5

A2

A2

Fit to problem space

Close information gap

Resilient and affordable

Scalability

Elasticity

Cross-layer

Heterogenity

A3 - Event and Information Collection

Physical + logical events

Unknown behavior

Failure prediction

Attack/response analysis

A4 – Event, Process Models and Attack Models

Resilient operations

Countermea-sure Support

OSSIM/Prelude Integration

A5 – Advanced SIEM Framework

;

Common Tasks in Security Engineering Methods

Security Requirements Engineering Process

- identification of the target of evaluation & principal security goals
- elicitation of artifacts (e.g. use case and threat scenarios)
- risk assessment
- the actual security requirements elicitation process
- requirements categorisation and prioritisation

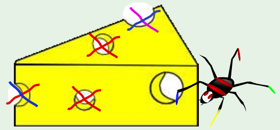
Further steps in Security Engineering

- security requirements (structural) refinement
- mapping of security requirements to security mechanisms → software requirements

Methods to Elicit Security Requirements

- misuse cases (attack analysis), soft systems methodology, quality function deployment, controlled requirements expression, issue-based information systems, joint application development, feature-oriented domain analysis, critical discourse analysis, accelerated requirements method, (cf. SQUARE)
- anti-goals derived from negated security goals,
- use Jackson‘s problem diagrams,
- actor dependency analysis (i^* approach)
- vulnerability analysis (attack graph/surface)
- functional dependencies analysis (Fuchs/Rieke ’09)

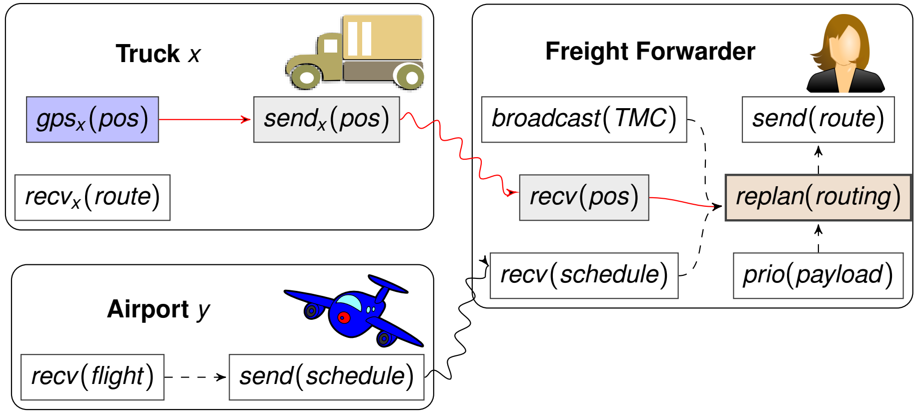
Completeness



Avoid premature architecture constraints

- protocols SSL/TLS/VPN/IPv6
- trust anchor TPM
- infrastructure PKI, PDP/PEP
- end-to-end/hop-by-hop

Example: Security Requirements Elicitation by Functional Dependency Analysis



Truck x

$gps_x(pos)$

$send_x(pos)$

$recv_x(route)$

Freight Forwarder

$broadcast(TMC)$

$send(route)$

$recv(pos)$

$replan(routing)$

$recv(schedule)$

$prio(payload)$

Airport y

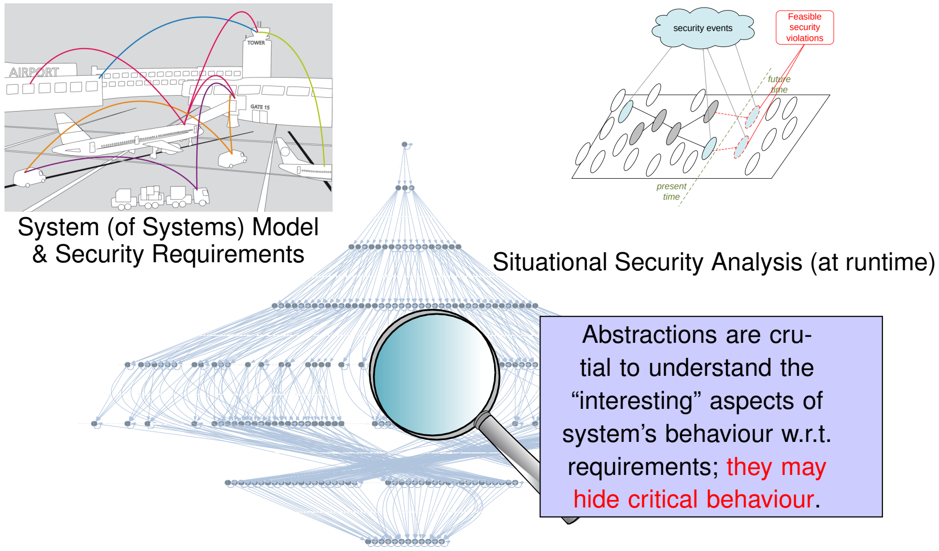
$recv(flight)$

$send(schedule)$

Security Requirement

$auth(gps_x(pos), replan(routing), scheduler)$

Models – Understand System's Behaviour & Predict Effects



System (of Systems) Model & Security Requirements

Situational Security Analysis (at runtime)

Abstractions are crucial to understand the "interesting" aspects of system's behaviour w.r.t. requirements; they may hide critical behaviour.

Operational Model (SoS Sandbox) → Behaviour Analysis (at SoS design time)

Requirements Engineering for Software vs. Systems in General

Herwig Mannaert

University of Antwerp
Department of Management Information Systems
Normalized Systems Institute

The Seventh International Conference on Systems (ICONS) 2012
1 March 2012

Universiteit Antwerpen

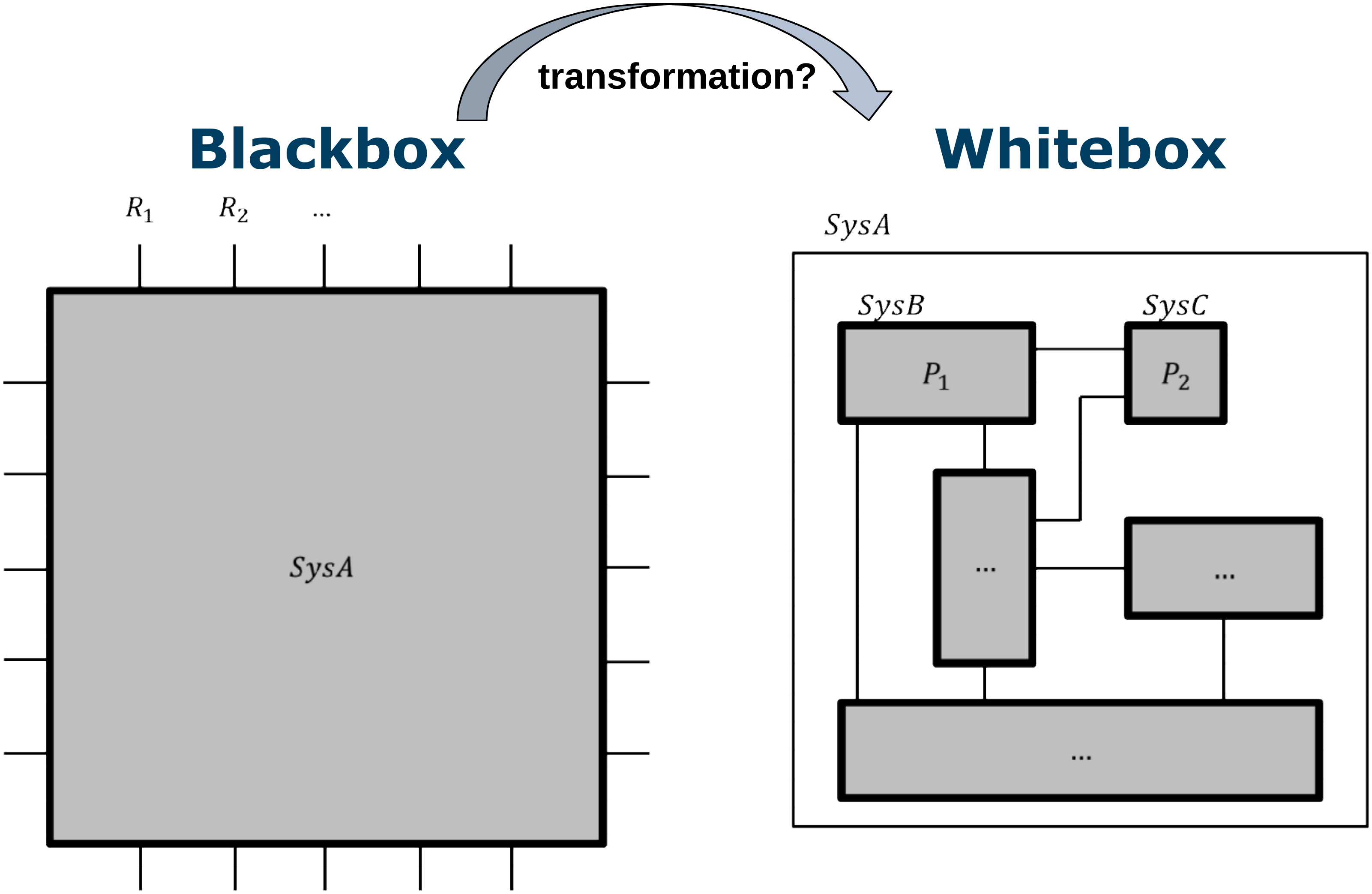
Modular Structures

- Systems in general can be seen as modular structures, i.e.
  - mechanical
  - information systems and software
  - Organizations
- Subdividing a system in subsystems should result in complexity reduction
- Software systems should strive to pay as much attention to modular structure as mechanical counterparts

Functional and Constructional

- Systems have both a black-box or functional view and a white-box or constructional view
- The main issue is that often hidden coupling is present, invisible in interface
- Service oriented architectures are trying to address this

Blackbox vs. whitebox



transformation?

Blackbox

Whitebox

Subject to Change

- Software systems are subject to change, as opposed to their mechanical counterparts
- Requirements will evolve during the development of software systems and through the entire lifecycle
- Normalized Systems theory has shown that it is all but trivial to cope with these changing requirements
- Neither can engineering systems in general, but they are not required to do so