FZI

FZI FORSCHUNGSZENTRUM INFORMATIK

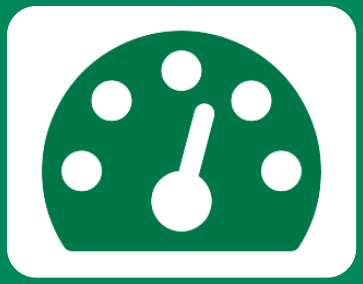
Datenstrommodellierung in Software-Architekturen: Konzepte, Analysen und offene Fragen

Prof. Dr. Ralf Reussner, Stephan Seifermann

GI Architekturen 2016

Quality as Key of Success

Quality Engineering

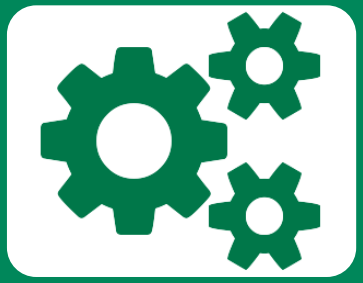


Quality attributes

- Performance
- Reliability
- Security
- Compliance

Quality by design

- Modeling and analyzing of design
- Determine quality before deployment
- Cost-efficient fixing of issues



Data flows as first class entities enable or enhance quality analyses

- Security (e.g. confidentiality)
- Compliance (e.g.handling of personal information)
- Performance (e.g. big data stream processing)

State of Data Flow Modeling

Threat modeling

- High level of abstraction
- Often solely done for threat analysis
- Security experts create and interpret models

[Swiderski and Snyder 2004]

Implementation analyses

- Low level of abstraction
- Break down of high level to low level goals
- High precision results regarding information flows, inference, …

[Ahrendt et al. 2005]
[Snelting et al. 2014]

Architectural data flows

- Inferred from control flow via parameters
- Separate models

[Kramer et al. 2014]
[Schmieders et al. 2015]

Data is no first class entity in architectural design phase

PIBA

Problem

- Data and their properties are highly relevant for certain quality analyses
- No annotation of architectural elements with data possible

Idea

- Introduce data and data flows as first class entities
- Use data properties in quality analyses

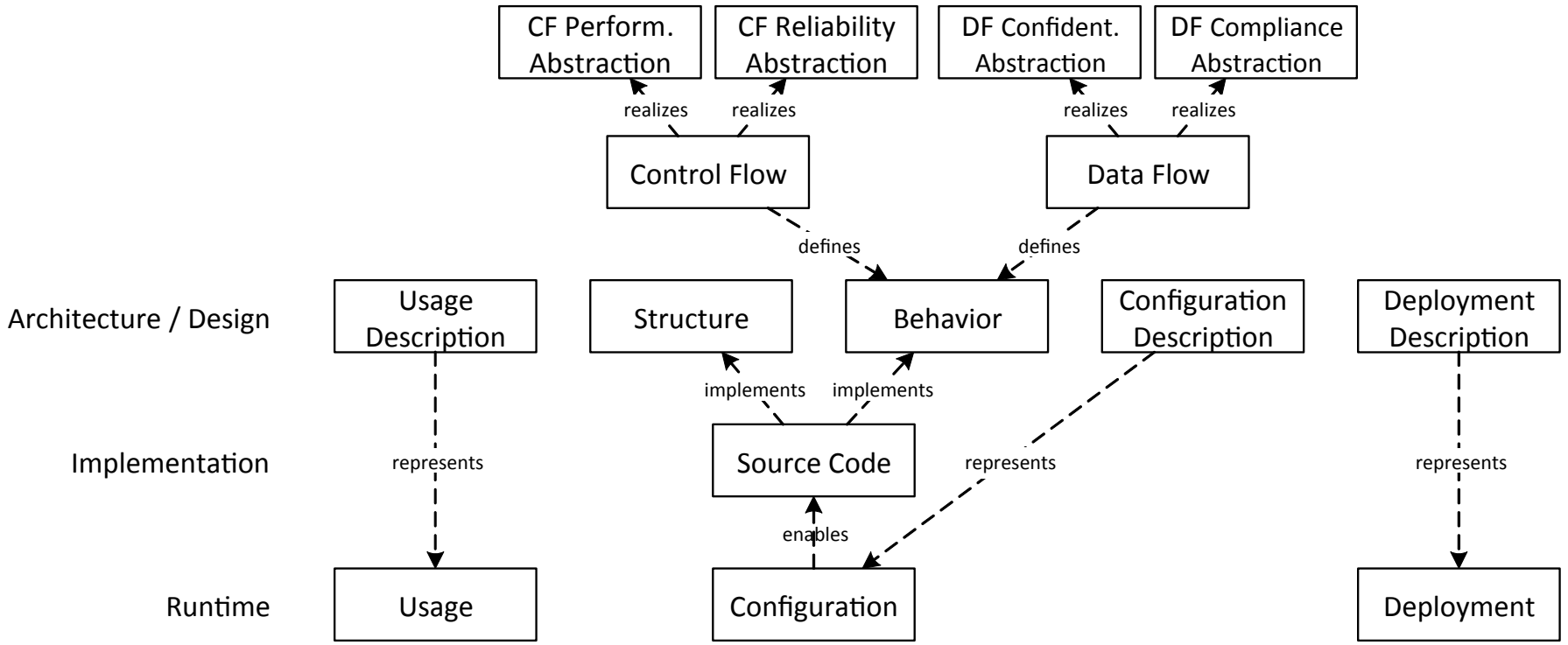
Benefit

- Directly adopt information from requirements engineering in architecture
- Express quality in natural way

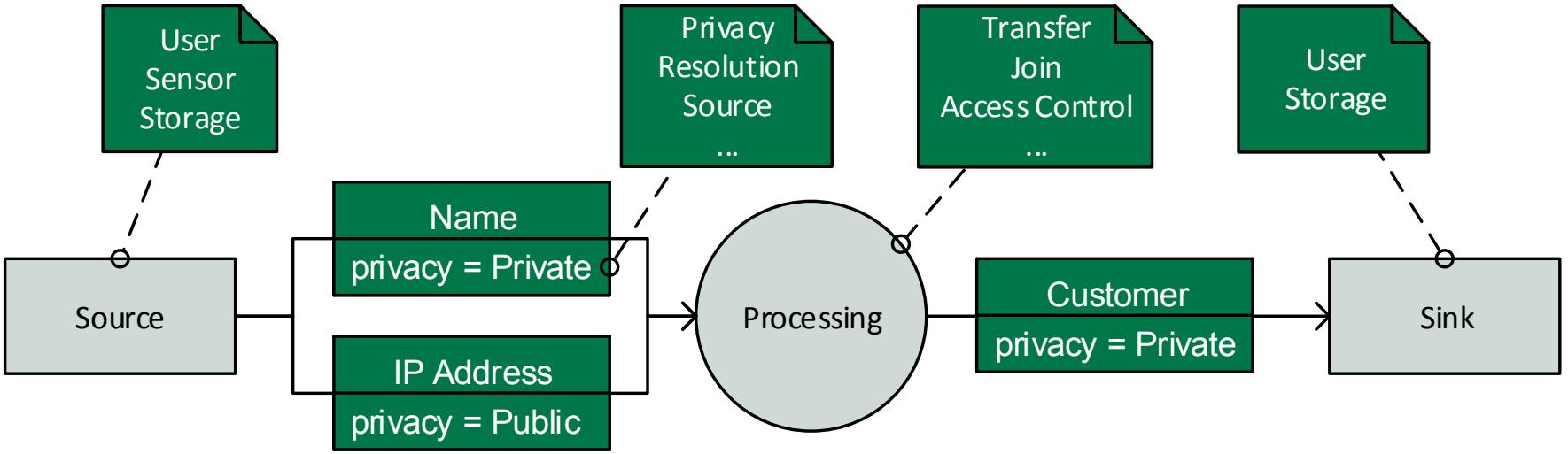
Action

- Extend Palladio with data flow constructs
- Define quality analyses for common use cases using data flows

Fitting of Data Flows into Software Architecture



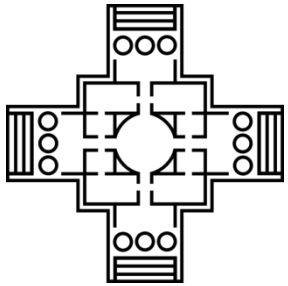
ADL Modeling Extension

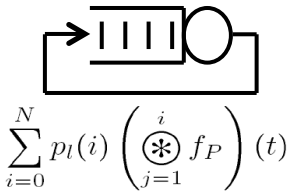


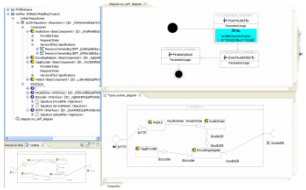
ADL extensions

- Classes of data and relevant meta-data
- Data processors
- Data sources and sinks
- Data flow
  - Inside components: chained processors
  - Between components: transfer processors

The Palladio Approach







Palladio Component Model

- Quality analyses of component-based systems
- Reusable specification

Analyses

- Performance
- Reliability
- Maintainability / costs

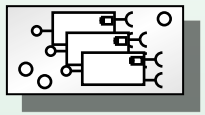
Development Process and Tools

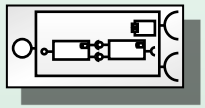
- Process for development of component-based software
- IDE for development and analyses

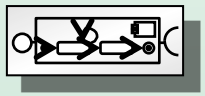
[Reussner et al. 2016]

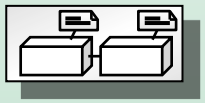
Views on System Models in Palladio

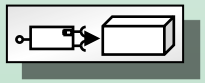
- Holistic view on the system
  - Software and hardware
  - Static and dynamic views
  - Allocation and usage profile

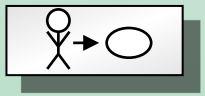

Software components (static)


Hierarchies (static)

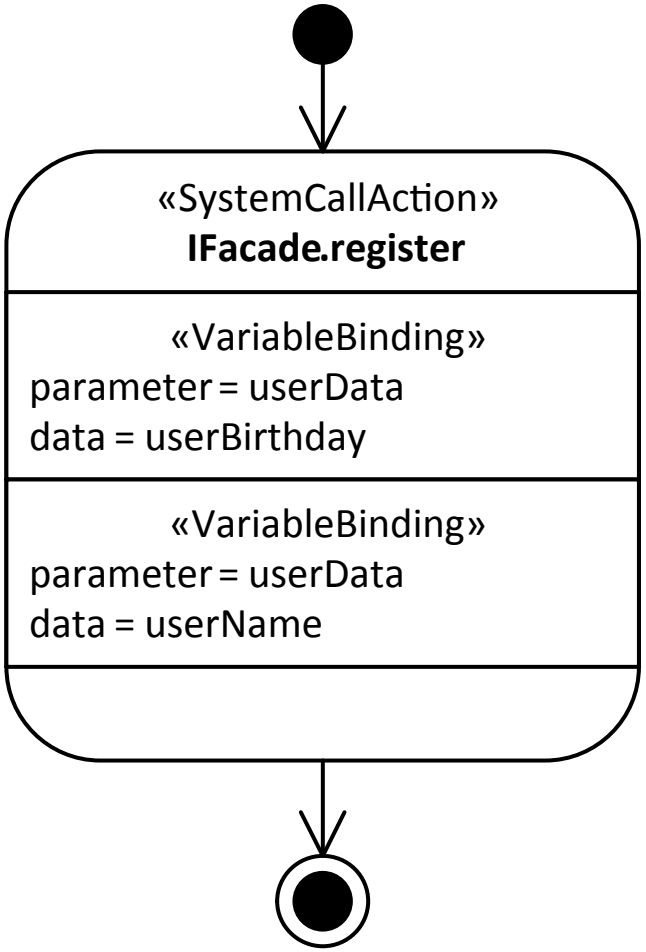

Software services (dynamic)

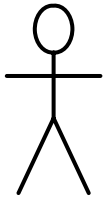

Resource environment


Allocation / deployment


Usage profile

Palladio Modeling Extension
Usage Profile





User

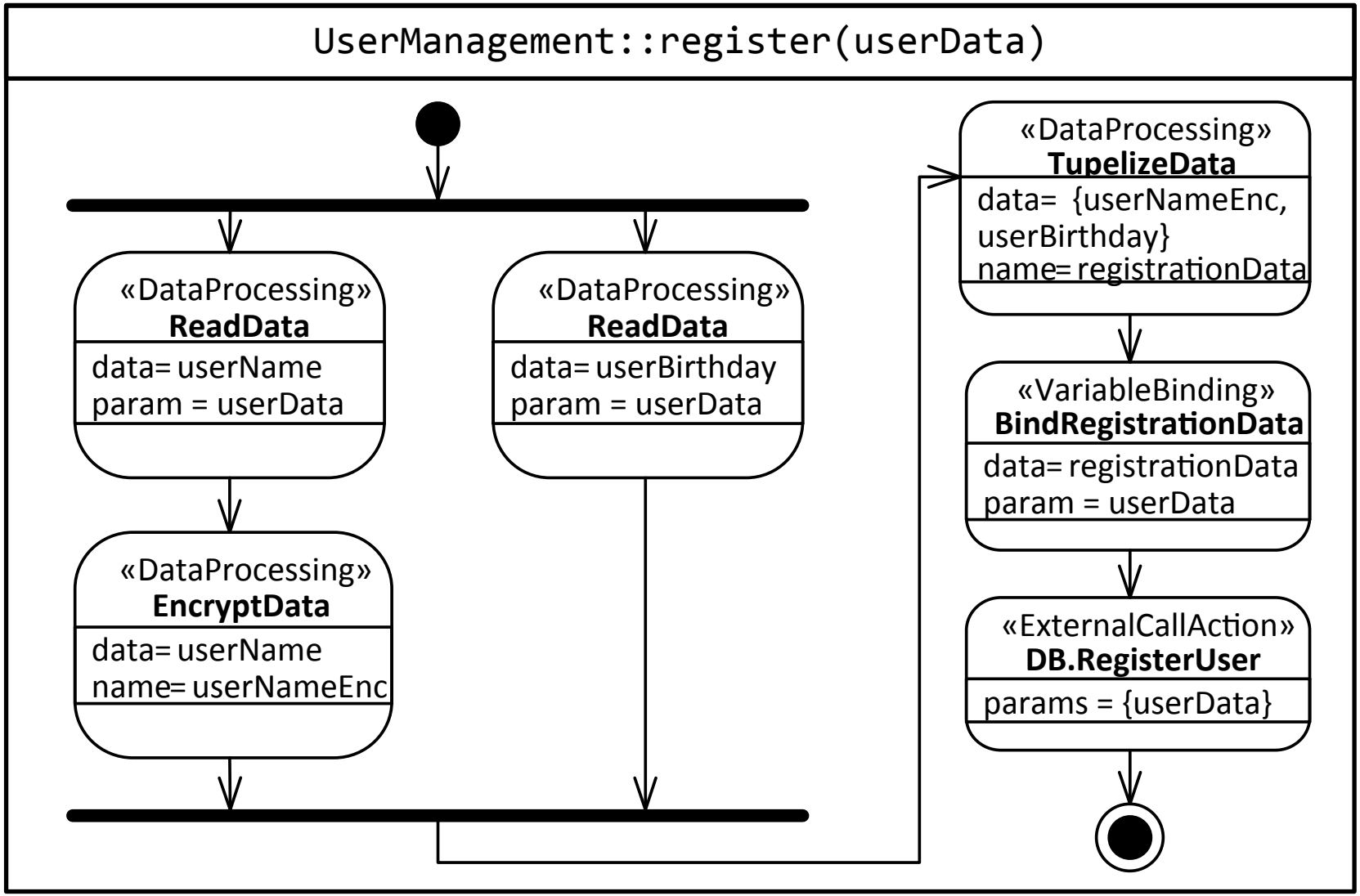
«DataDefinition»
Birthday

id = userBirthday
value = User.birthday

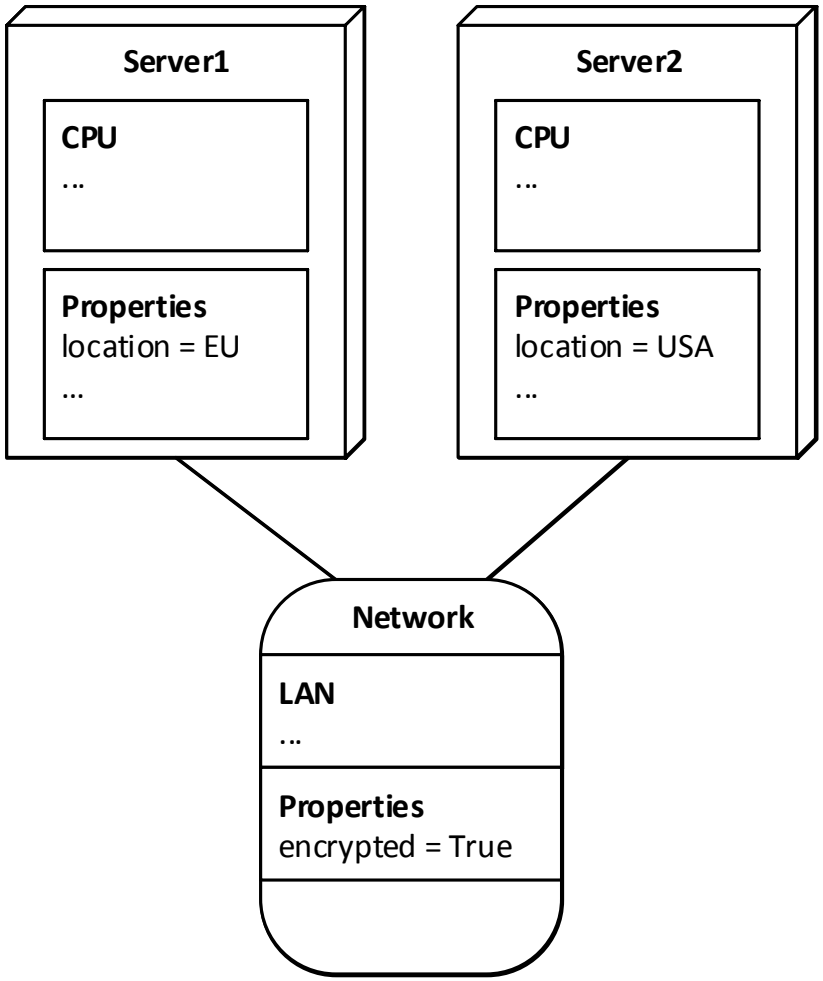
«DataDefinition»
Name

id = userName
value = User.name

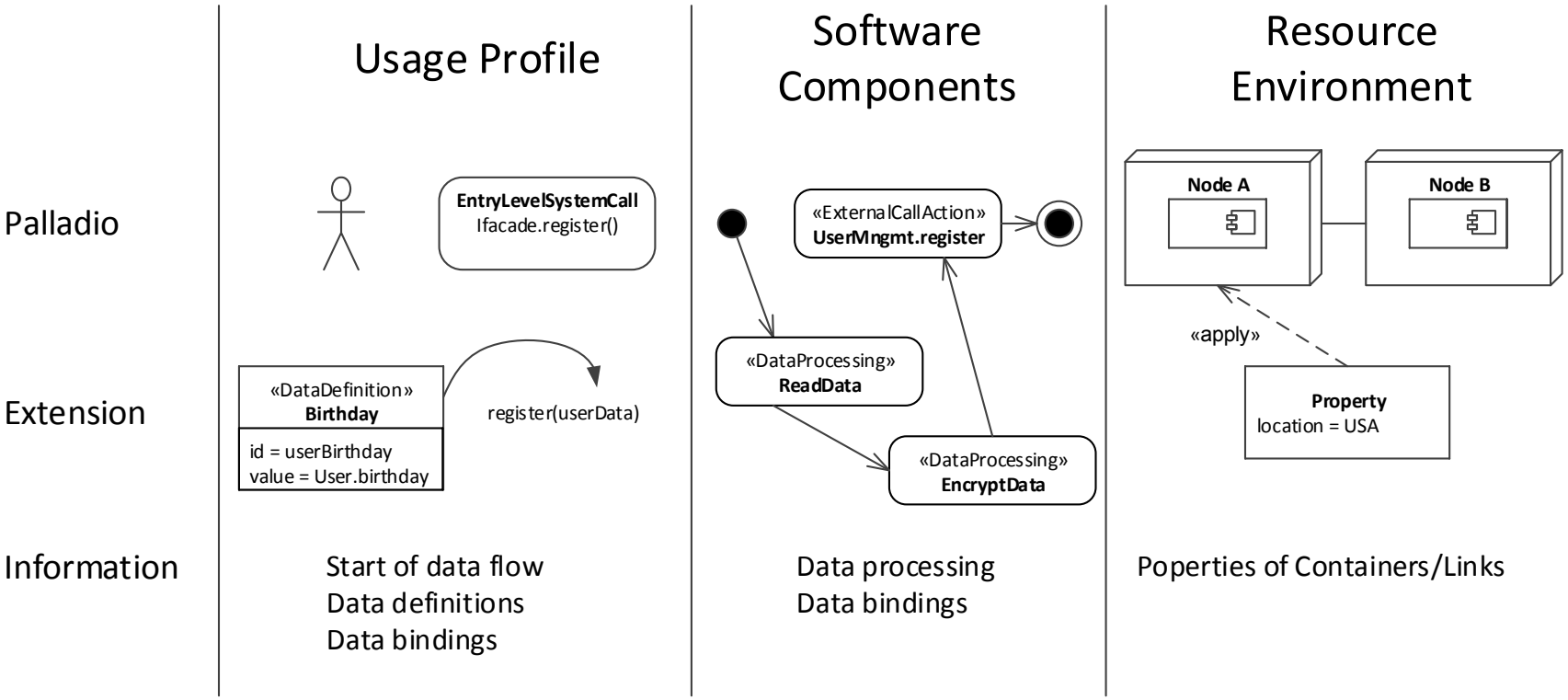
Palladio Modeling Extension
Software Components



Palladio Modeling Extension
Resource Environment

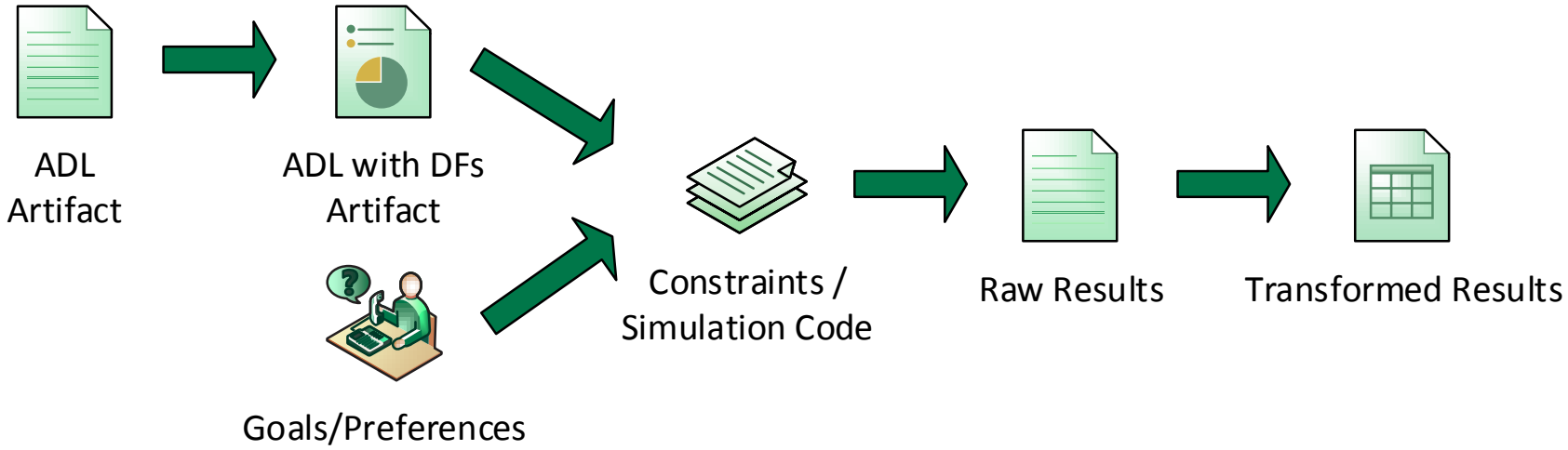


Palladio Modeling Extension Summary



Data Flow Analysis Approaches

	Analytical	Simulative
Input	Extended ADL artifact, analysis specification	
Method	• Transform inputs to constraints • use constraint solver	• Transform ADL artifact to simulation code • use resuls to answer analysis
	• Translate results back to ADL artifact elements	
Areas of Applicability	• Worst-case analyses	• Concurrent behavior • State-depending behavior



Analyses Based on Data Flows

Horizontal

- Infer offered service quality for interface
- Determine fulfillment of user goals

Vertical

- Determine requirements on infrastructure

Compliance

- Location of storage or processing of personal data
- Approval before transmitting personal data

Confidentiality

- Influence of data on other data
- Confidentiality during transmission

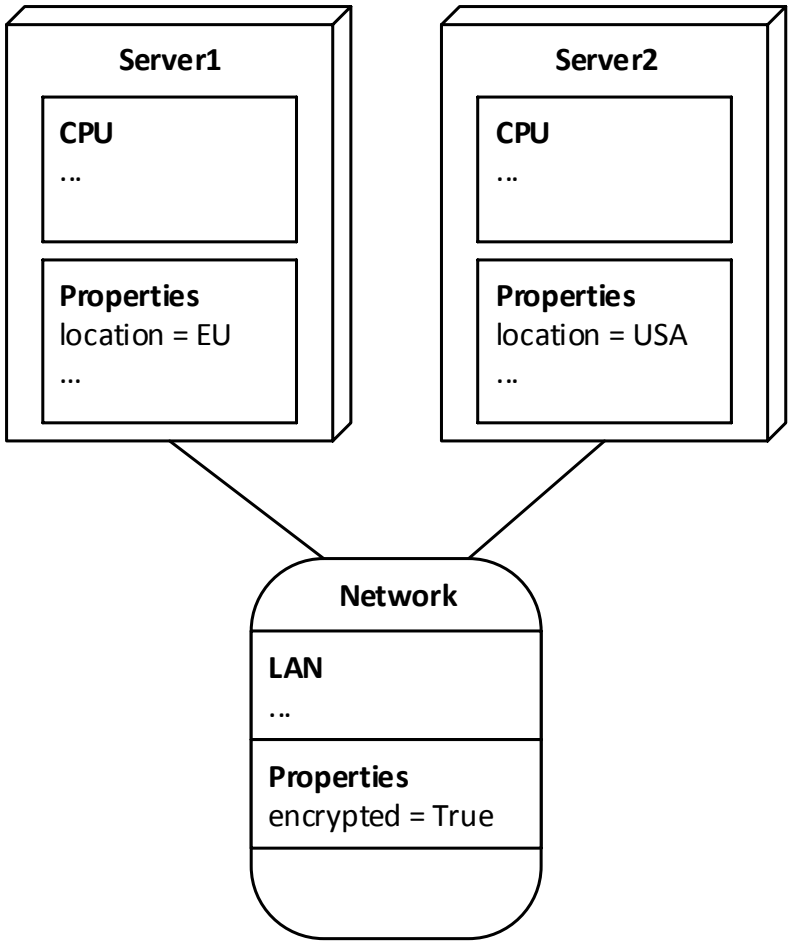
Transformation into Constraint Logic Problem Example

Resource Environment

- Properties of resources `resourceProperty(Server1, location, EU)`
- Linking between component and linking resource `resourceLink(Server1, Network)`

Allocation View

- Mapping between resources and components `allocated(Facade, Server1)`



Transformation into Constraint Logic Problem Example

Usage Model

- Data Definition
  `dataDefinition(userBirthday)`
- Variable Binding
  `bind(userBirthday, userData)`

Usage Model / Component Specification

- Call actions
  `call(Facade, register, UserManagement, register, [userData])`
- Data processor
  `applyEncryption(UserManagement, register, userName, userNameEnc)`
- Ordering of actions
  `post(bind(…), call(…))`

Transformation into Constraint Logic Problem Example

Analysis goal

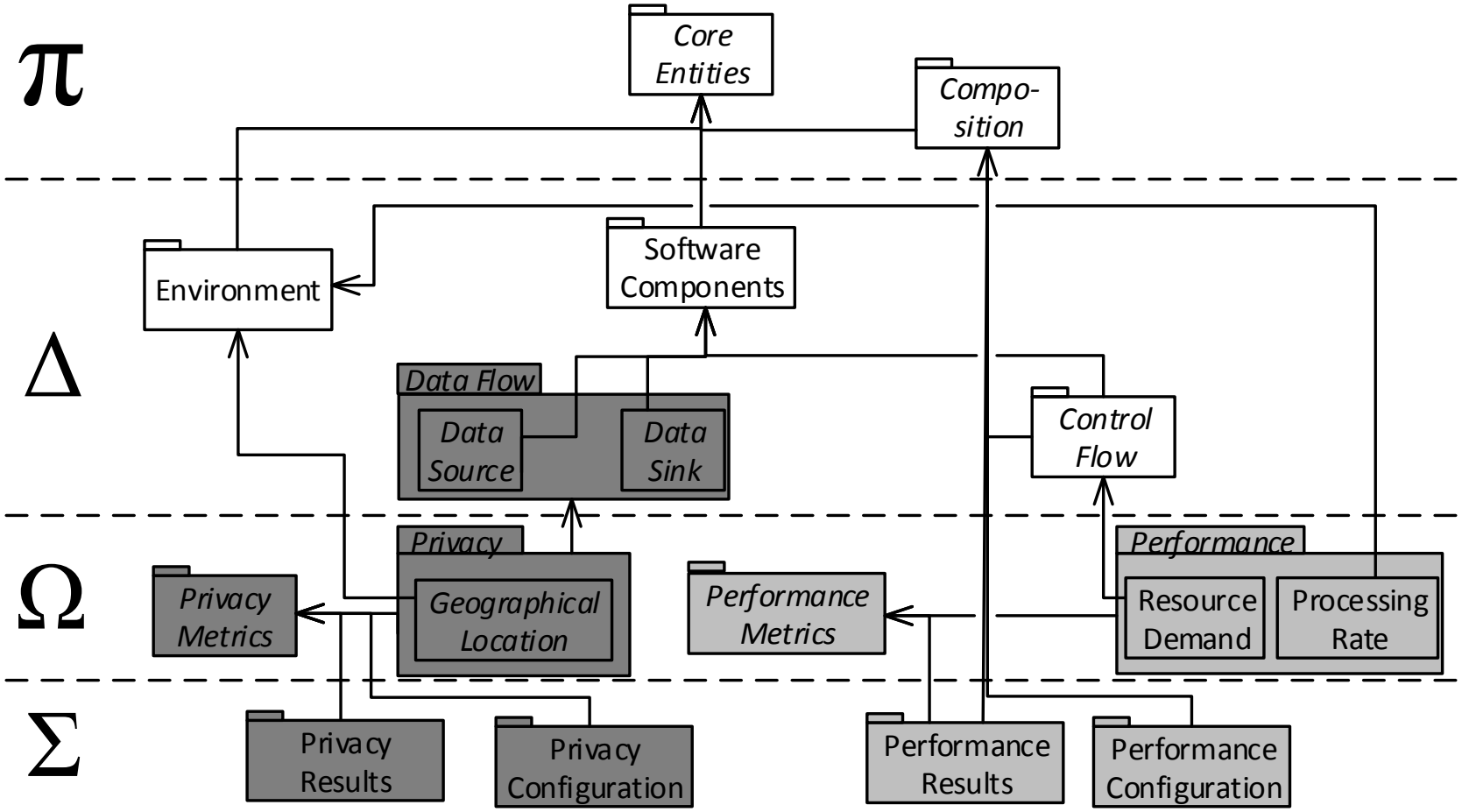
- No personal data d is transferred from the EU to other country.

	no call x to y with parameters P
$allocated(x,s\downarrow 1\ ) \wedge allocated(y,s\downarrow 2\ ) \wedge s\downarrow 1 \neq s\downarrow 2$	x, y deployed on different server
$\wedge$	
$resourceProperty(s\downarrow 1\ ,location, l\downarrow 1\ ) \wedge resourceProperty(s\downarrow 2\ ,location,l\downarrow 2\ ) \wedge l\downarrow 1 \neq l\downarrow 2$	servers in different locations
$\wedge$	
$l\downarrow 1 \in \{EU\} \wedge l\downarrow 2 \notin \{EU\}$	x inside, y outside EU
$\wedge$	
$\exists bind(d,p)\ with\ p \in P \wedge personal(d)$	personal data is transferred via one of the parameters

Benefits of Data Flows as First Class Entities in ADLs

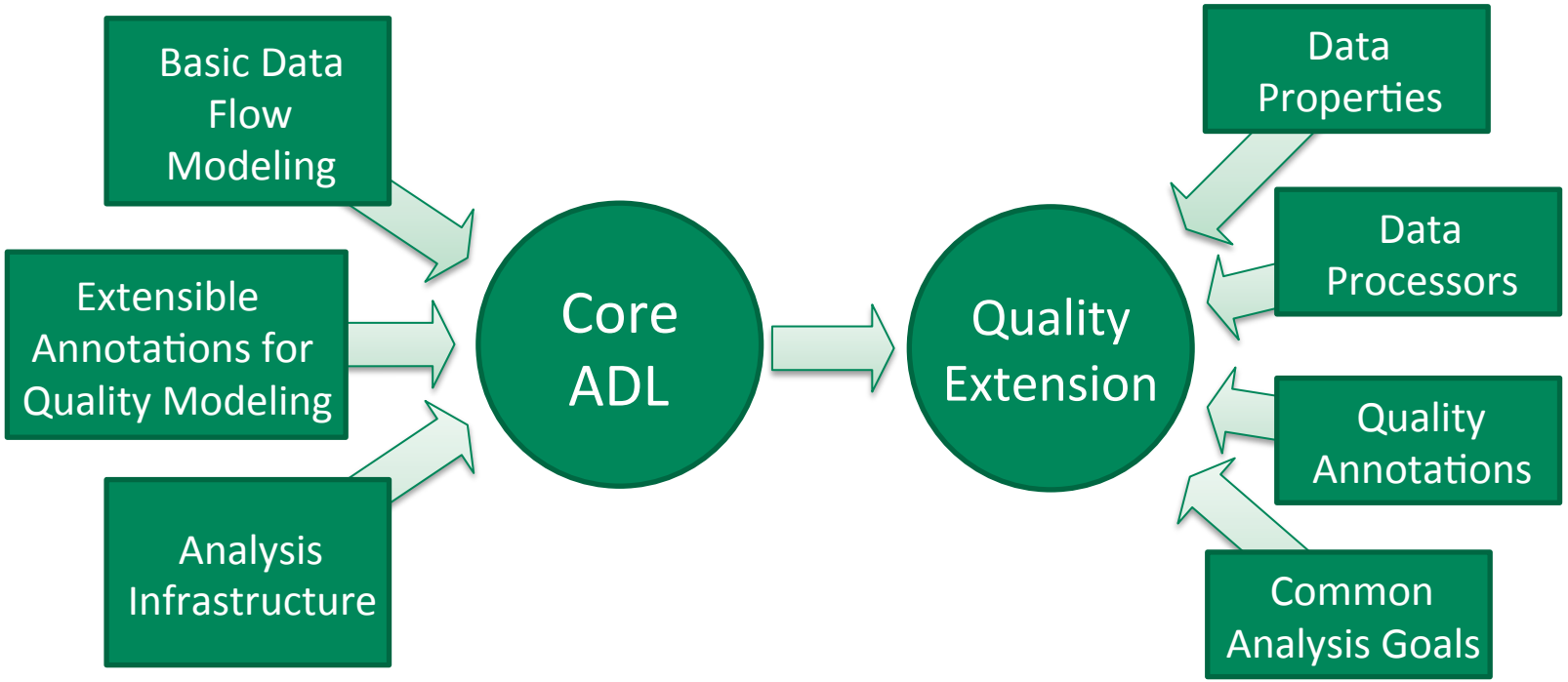
Low modeling effort	• More natural expression of quality requirements • Reuse of existing specifications • Consistency given automatically
Comprehensible results	• Data flows defined in similar terms as architecture • Clear mapping from results to known elements
Usable by non-experts	• Stakeholders fokus on well known parts • No separate analysis models
Early issue detection	• No implementation required • Low cost what-if scenarios • Stable quality properties even after evolution

Extensible Quality Prediction Meta-Models



[Strittmatter et al. 2015]

Extensible Quality Analyses Based on Data Flows



Open Questions

Extensible analyses

- How to combine extensions to lower duplicated information?

Analyses

- What does composability for data flows mean?
- What are the limits of the analytical analysis approach?

Modeling

- How to find an appropriate abstraction for runtime configuration?
- Is including runtime configuration via data processors sufficient?
- Can non-experts find the right granularity for the models?

Conclusions

Quality attributes	• Heavily affect business value • Should be predicted before deployment
Data flows as first class entities	• Allow expressing quality attributes in natural way • Enable prediction of many qualities
Integration in ADL beneficial	• Reuse of existing models • Improved comprehensibility
Analyses	• Analytical approach with constraint solving • Simulative approach

References

- **[Ahrendt et al. 2005]**
  W. Ahrendt et al. The KeY tool. SoSyM, 4:32-54, 2005.
- **[Kramer et al. 2014]**
  M. E. Kramer, A. Hergenröder, M. Hecker, S. Greiner, and K. Bao. Specification and verification of confidentiality in component-based systems. Poster at the 35th IEEE Symposium on Security and Privacy, 2014.
- **[Reussner et al. 2016]**
  Reussner, Ralf H. et al., editor. Modeling and Simulating Software Architectures - The Palladio Approach. MIT, 2016. ISBN: 978-0-262-03476-0, to appear.
- **[Schmieders et al. 2015]**
  E. Schmieders, A. Metzger, and K. Pohl. Architectural runtime models for privacy checks of cloud applications. In PESOS'15, pages 17-23. IEEE, 2015.

References

- **[Snelting et al. 2014]**
  G. Snelting et al. Checking probabilistic noninterference using joana. it - Information Technology, 56:280-287, 2014.
- **[Strittmatter et al. 2015]**
  M. Strittmatter, K. Rostami, R. Heinrich, and R. Reussner. A modular reference structure for component-based architecture description languages. In ModComp‘15, pages 36-41. CEUR, 2015.
- **[Swiderski and Snyder 2004]**
  F. Swiderski and W. Snyder. Threat Modeling. Microsoft, 1st edition, 2004.

Palladio Modeling Extension Summary

Usage profile

- Data definitions
- Binding of data classes to system calls

Software components

- Data processors
- New SEFF containing chained data processors

Resource environment

- Annotations for infrastructure elements

Runtime configuration model (new)

- Abstractions of runtime configuration
  - Access control
  - Firewall rules