Universität Karlsruhe (TH)
Research University • founded 1825

SD Software Design and Quality

Software Architecture: Basics and Performance Engineering

Guest Lecture
Saarbrücken, 9th June 2009

Ralf Reussner (reussner@ipd.uka.de)

http://sdq.ipd.uka.de

1

Overview on today's lecture

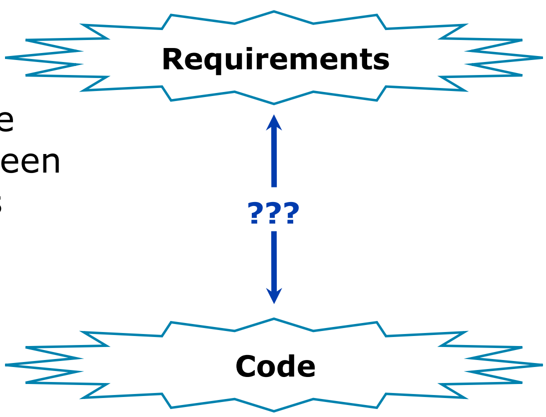
- What is a software architecture?
- What are its benefits?
- The Use of Architectures for Software Performance Prediction

2

PDF-Folien für Studis:
- Benutzername stud
- Passwort sw-architektur
-**aus dem Wiki verlinkt, nach der Vorlesung freigeschaltet**

The Problem

- How to bridge the gap between requirements and code?



3

The traditional Answer

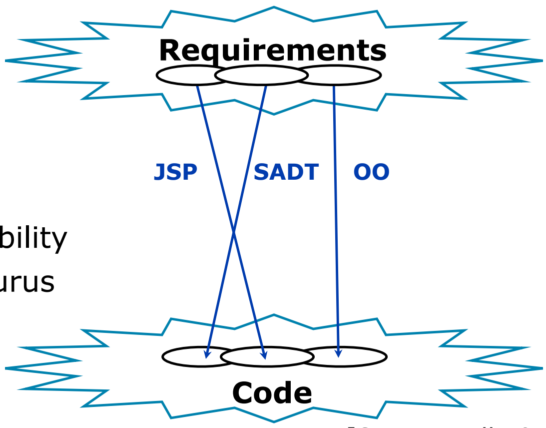
- Ad hoc
- Requires gurus
- Unpredictable
- Costly

Software Development Methods

- More predictable processes
- Some design guidance

BUT

- Limited applicability
- Still requires gurus
- Weak support for design analysis

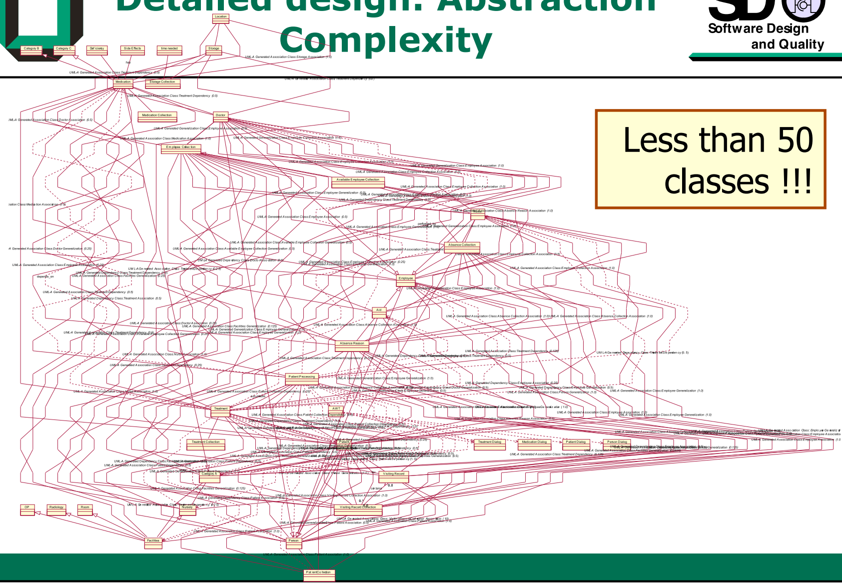


[Sommerville 04]

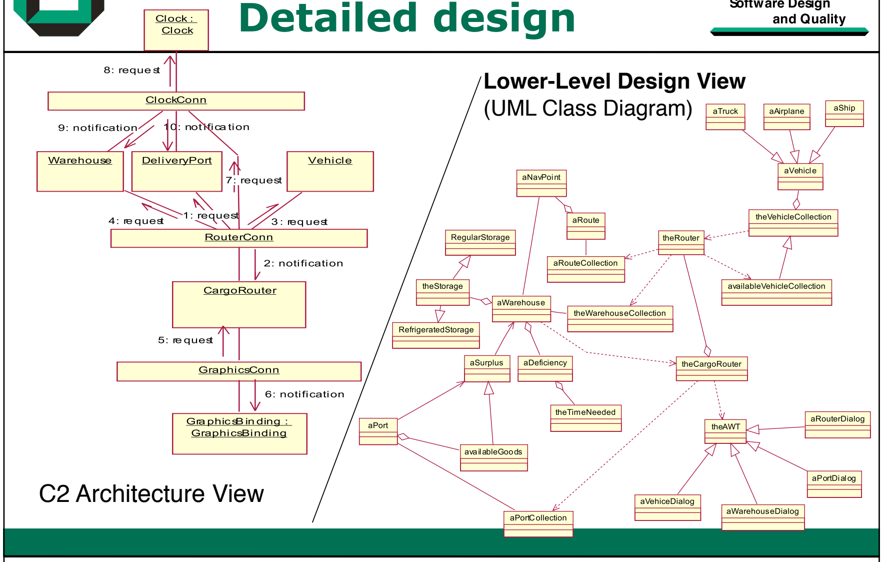
JSP (Jackson Structured Programming)

SADT (Self Accelerating Decomposition Temperature)

Detailed design: Abstraction Complexity



Architecture vs. Detailed design



What is an Architecture?

Several definitions exist:

- A software architecture defines the coarse-grained structure of the system.
- A software architecture captures design decisions which are hard to revert or which have to be made early.

Architectural Design Decisions (1)

- Architectural design is a creative process so the process differs depending on the type of system being developed.
- However, a number of common decisions span all design processes.

Architectural Design Decisions (2)

- Is there a generic application architecture that can be used?
- Which kinds of distribution are possible and appropriate?
- What architectural styles are appropriate?
- What approach will be used to structure the system?
- How will the system be decomposed into subsystems (modules, components)?
- What management and evolution strategy should be used?
- How will the architectural design be evaluated?
- What are realistic evolution scenarios?

Architectural Design Decisions (3)

- How should the architecture be documented?
- Which components can or must be bought?
- How to include legacy software?
- How to communicate with existing software?
- How to access existing data?
- How does the architecture fit into the existing portfolio?
- What can be re-used from older project?
- What should be re-used in the next project?
- Is a product-line architecture appropriate?
- …

Architectural Design

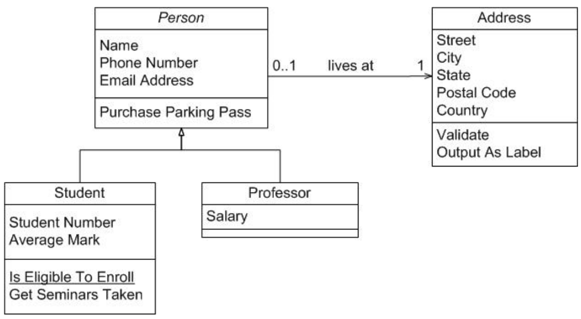
- An early stage of the system design process.
- Represents the link between specification and design.
- Often carried out in parallel with some specification activities.

- It involves identifying major system components, their communications and mapping to hardware or software resources.

What constitutes a Software Architecture?

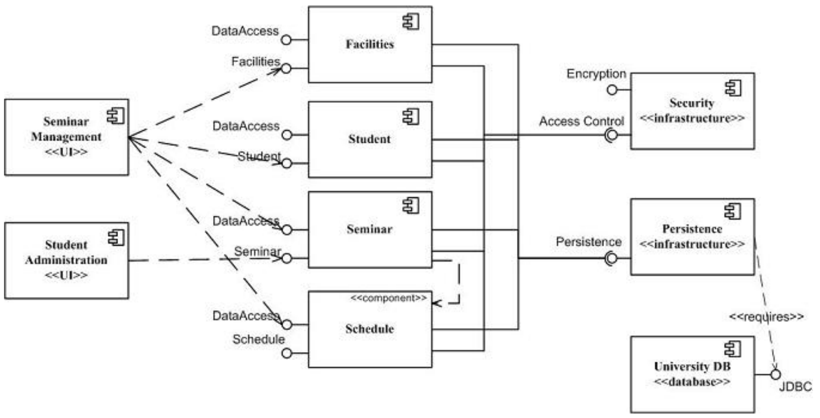
- Static structural model that shows the major system components.
  - Interface model that defines sub-system interfaces.
- Dynamic process model that shows the process structure of the system.
  - Relationships model such as a data-flow model that shows sub-system relationships.
- Deployment model that shows how sub-systems and connections are mapped to resources, such as processors or network connections
  - distribution across computers.

Static View (Data Objects)



http://www.agilemodeling.com/artifacts/

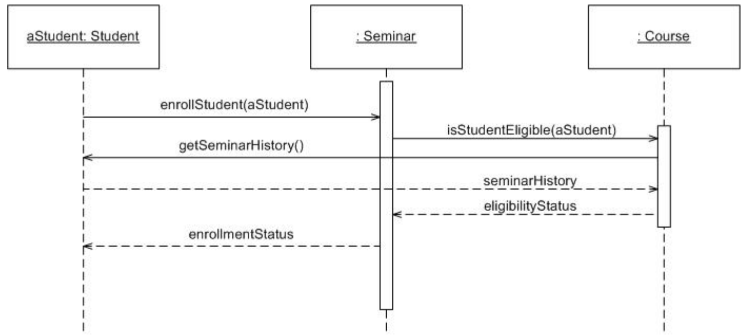
Static View (Architecture)



http://www.agilemodeling.com/artifacts/

Dynamic View (inter component dynamism)

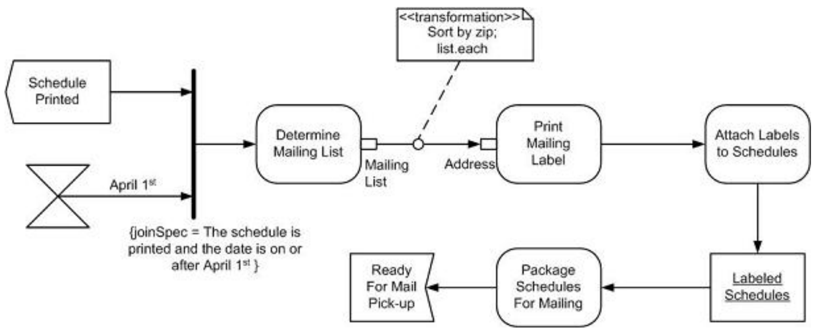
Software Design and Quality



http://www.agilemodeling.com/artifacts/

Dynamic View (intra component dynamism)

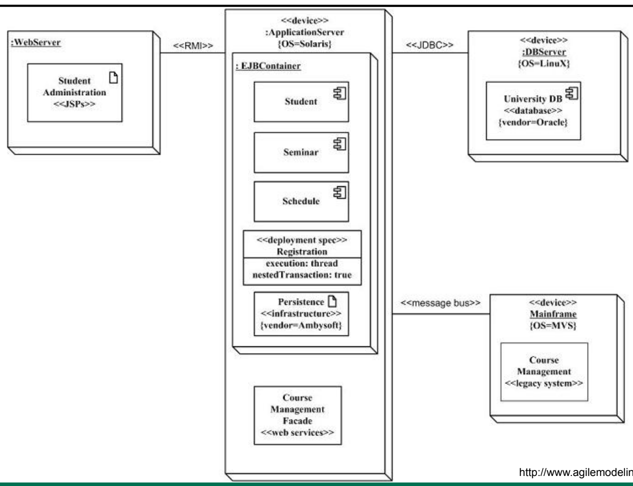
Software Design and Quality



http://www.agilemodeling.com/artifacts/

Deployment View

Software Design and Quality



http://www.agilemodeling.com/artifacts/

Overview on today's lecture

- What is a software architecture?
- **What are its benefits?**
- The Use of Architectures for Software Performance Prediction

PDF-Folien für Studis:

- Benutzername stud

- Passwort sw-architektur

-aus dem Wiki verlinkt, nach der Vorlesung freigeschaltet

Advantages of an explicit Architecture

- Stakeholder communication
  - Architecture may be used as a focus of discussion by system stakeholders.
- System analysis
  - Analysis of whether the system can meet its non-functional requirements.
- Large-scale reuse
  - The architecture may be reusable across a range of systems.
  - Existing components can be considered during design
    - COTS, in-house components, commissioned / off-shore
- Project planning
  - Cost-estimation, mile stone organisation, dependency analysis, change analysis, staffing

Predicting the quality attributes of an artefact during design is a core property of any engineering discipline.

Architecture and System Characteristics

- Performance
  - Localise critical operations and minimise communications. Use large rather than fine-grain components. Lower resource usage.
- Security
  - Use a layered architecture with critical assets in the inner layers.
- Safety
  - Localise safety-critical features in a small number of sub-systems.
- Availability
  - Include redundant components and mechanisms for fault tolerance.
- Maintainability
  - Use of fine-grain, replaceable components, localisation of design decisions which are likely to change

Relation between Architectural Quality Properties

- **Intrinsic**: definition of property A involves property B.
  - "The system is considered available if the reaction time is below 5 ms."
  - Performability: the performance of a system, including its performance during failures
- **Extrinsic**: improvement of property A decreases property B in an architecture C
  - Using large-grain components improves performance but reduces maintainability.
  - Introducing redundant data improves availability but makes security more difficult.
  - Note the influence of the Architecture on the relationship:
  - The duplication of components can increase performance and reliability in one architecture while it can decrease performance in another one.

Factors Influencing the Architecture

- Requirements
- Re-Use
  - Architectures
  - Subsystems / Components
  - Guidelines
- Organisation (Conway's law)
  - team size, team number, experience, organisation structure

Some Terms

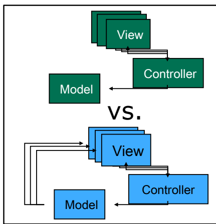
- Meta-Model: Model to model a model: which elements having which attributes.
- Model: Abstraction of the modelled entity – with a given abstraction aim. Instance of a meta model.
- Style:
  (a) [Reussner] Cross-cutting principles (object-oriented style, modular style), independent of application, should not be mixed
  like in building: baroque-style, classicist-style
  (b) Synonymously used for Pattern
- Pattern: Solution to recurring problem / situation where several forces have to be balanced. Often application specific, often mixed
- View: Commonly emphasises certain aspects of a model (distribution, componentisation, dynamic behaviour). Is a mean of structuring an instance of the meta-models. Hence a view is usually defined for a subset of elements of the meta-model.

Overview on today's lecture

- What is a software architecture?
- What are its benefits?
- **The Use of Architectures for Software Performance Prediction**

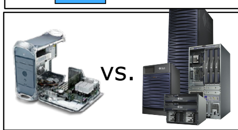
PDF-Folien für Studis:
- Benutzername stud
- Passwort sw-architektur
-aus dem Wiki verlinkt, nach der Vorlesung freigeschaltet

Why do we want to predict quantitative Properties?

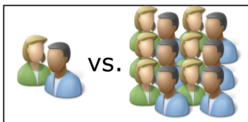


Evaluation of Design Alternatives

- the quantifiable best of a list of many
- trade-off decisions
  - cost vs. benefits
  - QA a vs. QA b

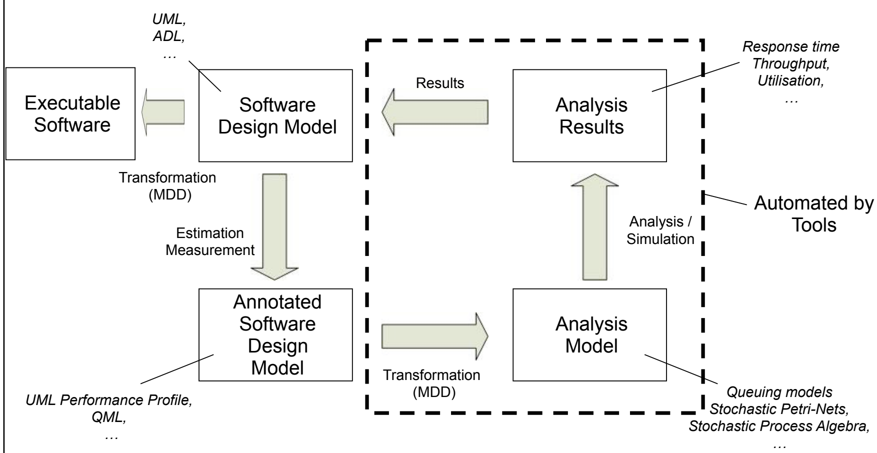


Dimensioning of Resources ("Sizing")

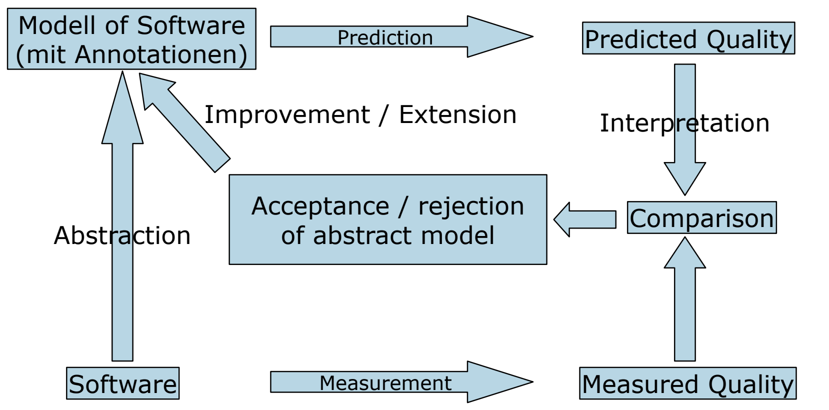


Changes of usage profile – Scalability

Model-based Prediction of Quantitative Properties

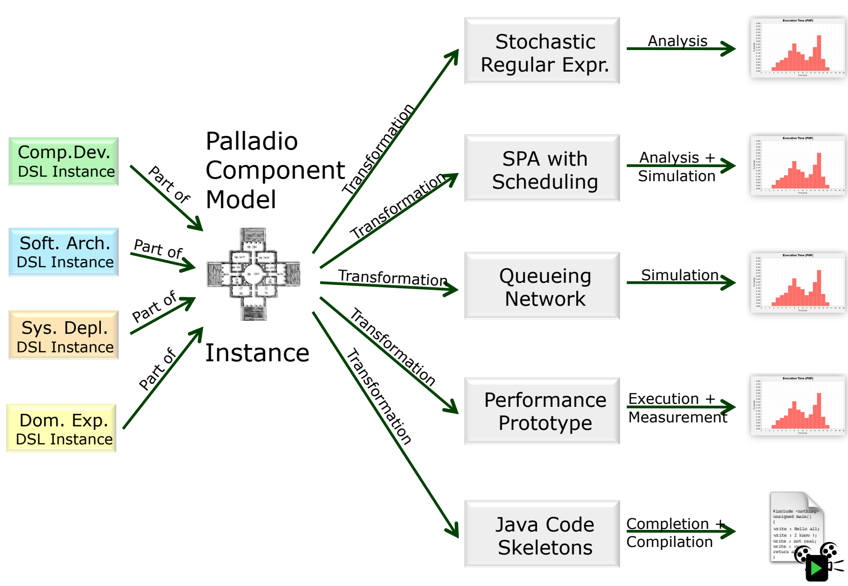


Scientific Approach to Create Quantitative Models

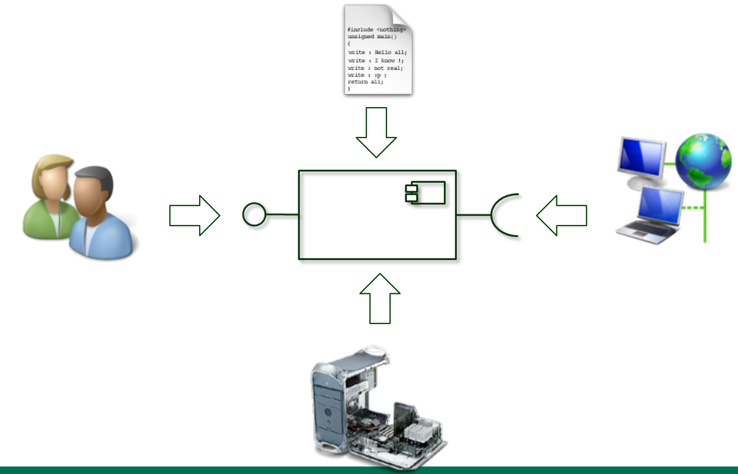


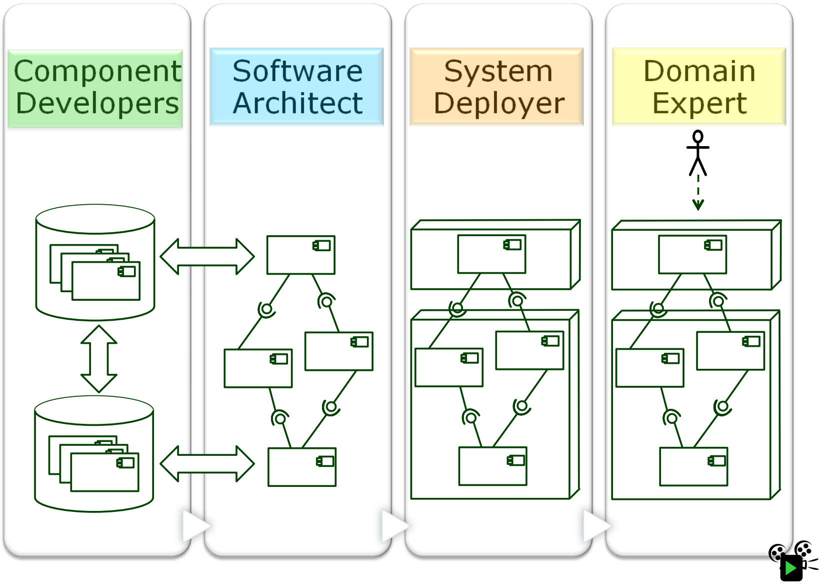
Validation of Quantitative Models

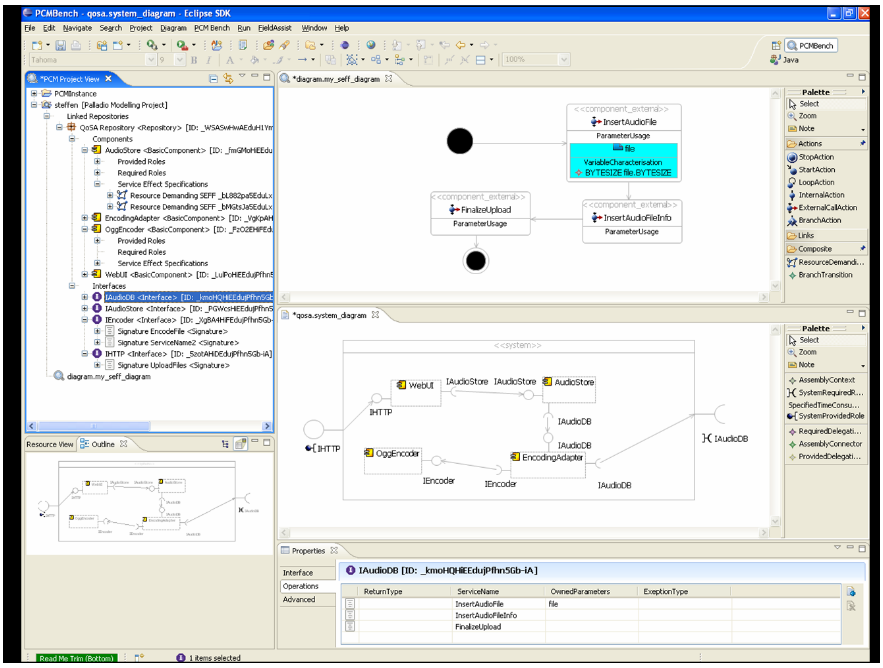
- Type 1: Validation of Prediction Model
- Type 2: Validation of Applicability
  - Case Studies and Controlled Experiemts with Students
- Typ 3: Validation of Benefits
  - in comparison to different methods
  - Limitations of the Approach
  - Required prerequisites
  - FZI
  - Industrial Partners



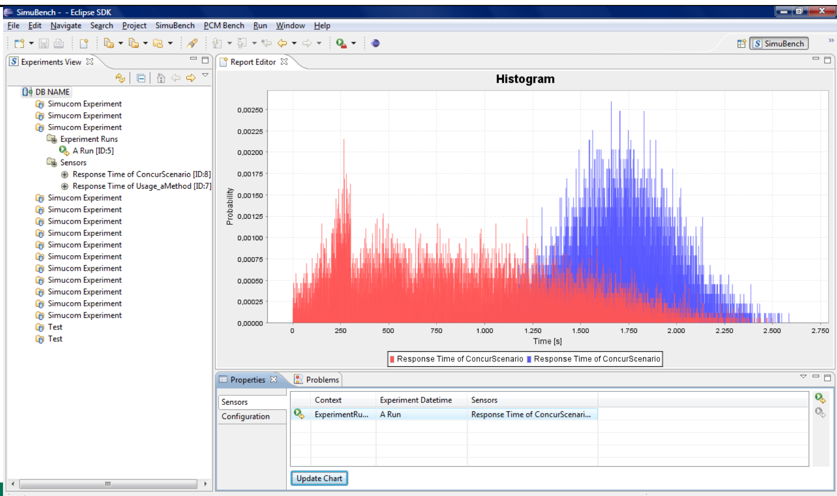
Factors on Quantitative Component Properties



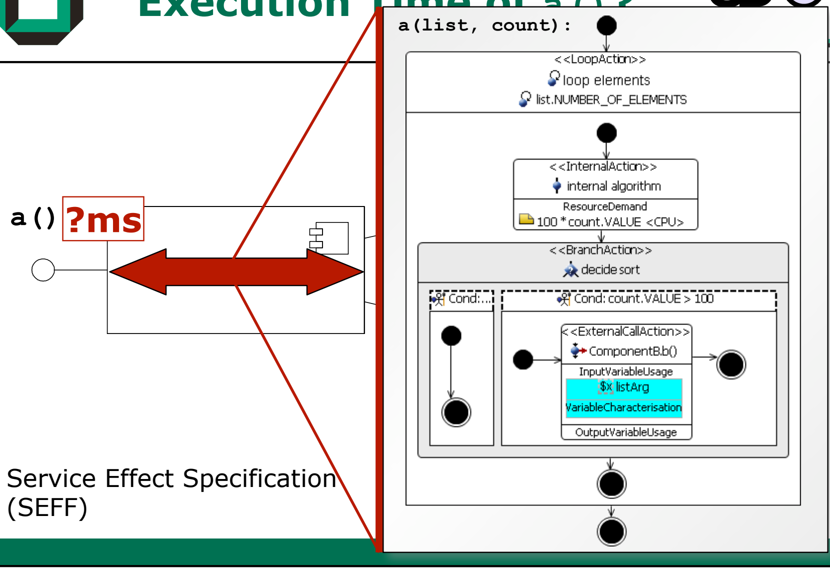




Tool Support



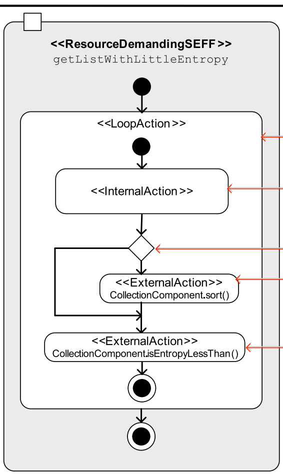
Execution Time of a()?



Service Effect Specification (SEFF)

Syntax comparable to UML activity charts

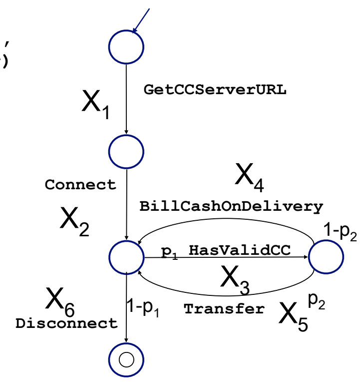
Service Effect Specification (1)




```
public List getListWithLittleEntropy
        (List listToSort, int count) {

    while(mode) {

      // some simple internal action
      for(int x = 0; x < count; x++) {
          listToSort.add(new Integer(x));
      }

      if(count > 100) {
          //external call:
          collectionComponent.sort(listToSort);
      }

      //external call:
      mode = collectionComponent
          isEntropyLessThan(listToSort, count);

    }
    return listToSort;
}
```

Annotierter SE-Automat eines Dienstes `DoOrderBilling`

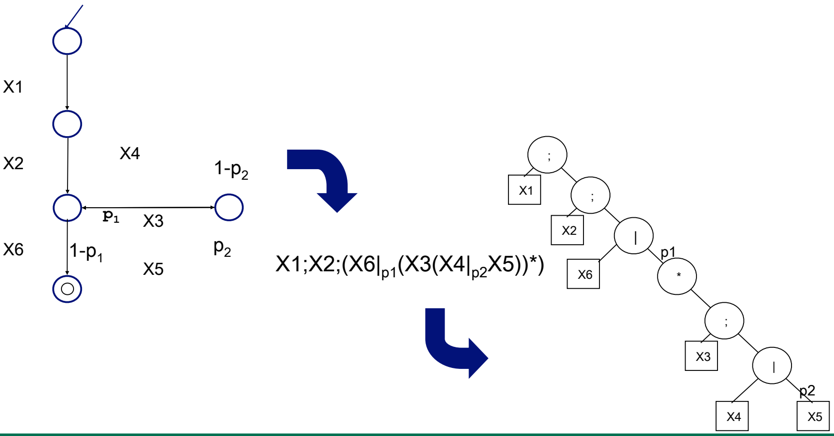
```
void DoOrderBilling (ListOfOrders orders,
                     CCServer myCCServer)
{
  myCCServer.Connect(
        resources.GetCCServerURL());
  foreach (Order o in orders)
  {
    if (!o.HasValidCC())
    {
      BillCashOnDelivery(o);
    }
    else
    {
      myCCServer.Transfer(o);
    }
  }
  myCCServer.Disconnect();
}
```



37 As an example consider the following code and its associated service effect automaton. It can be seen, that transitions correspond to external calls, while any internal computation is abstracted away within nodes. Nodes represent internal computation.

The ps on the branchings are the probabilities for controll flow forking.

Systematische Berechnung der Verteilungsfunktion

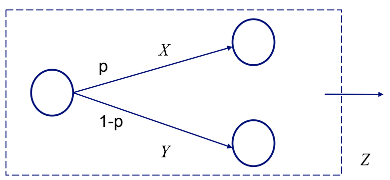


38 For this purpose the service automaton is translated into a regular expression. Afterwards the parse tree of the regular expression is created.

This parse-tree gives us the order of how to apply the basic operators of alternative, sequence and loop to the distribution functions.

By stepwise using of

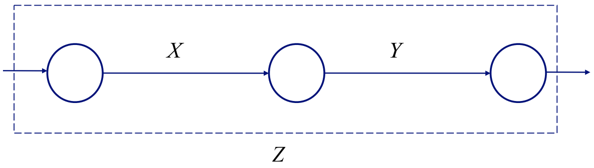
Komposition: Alternative



$$x_{alt}[n] = px[n] + (1-p)y[n]$$

39 A random variable associated to an alternative is represented as a sum of the alternative paths weighted with the call probabilities. The associated probability mass function is therefore the weighted sum of single probability mass functions. The weights are the probabilities of the alternative transitions

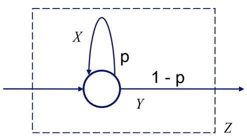
Komposition: Sequenz



$$
\begin{aligned}
(x+y)[n] &= P(X+Y=n)\\
&= \sum_{k=1}^{n} P(X=k)P(Y=(n-k))\\
&= \sum_{k=1}^{n} x[k]y[n-k] =: x \circledast y[n]
\end{aligned}
$$

For a sequential execution of services the time consumption of the whole sequence is the sum of time consumption for each external call- Therefor the random variable associated to a call sequence is represented as a sum of the random variables assigned to the individual edges. The probability mass function results from the

Komposition: Schleife



$$x_{loop}[\cdot] = (1-p)y[\cdot] \circledast \sum_{k=0}^{\infty} p^k \underset{l=1}{\overset{k}{\circledast}} x[\cdot]$$

$$\underset{l=1}{\overset{0}{\circledast}} x[\cdot] = \delta[\cdot]$$

$$
X_{loop} = \begin{cases}
Y & \text{with probability} \quad 1-p\\
X+Y & \text{with probability} \quad p(1-p)\\
\vdots & \\
\underbrace{X+\ldots+X}_{k}+Y & \text{with probability} \quad p^k(1-p)\\
\vdots &
\end{cases}
$$

A loop is either run again with probability p or left with probability 1-p. Therefore one can represent a loop as a choice of an infinite number of alternative paths.

The associated probability mass function is given by the infinite series. If k is zero, the convolution is defined to be unity impulse which is

$x_{loop}[\cdot] = (1-p)y[\cdot] \circledast \sum_{k=0}^{\infty} p^k \underset{l=1}{\overset{k}{\circledast}} x[\cdot])$

$$x_{loop}[\cdot] = (1-p)\mathcal{F}^{-1}\left\{\mathcal{F}\{y[\cdot]\} \sum_{k=0}^{\infty} (p\mathcal{F}\{x[\cdot]\})^k\right\}$$

Konvergenz-prüfung $\quad |p\mathcal{F}\{x[\cdot]\}| = p|\mathcal{F}\{x_\alpha[n]\}| = p < 1$

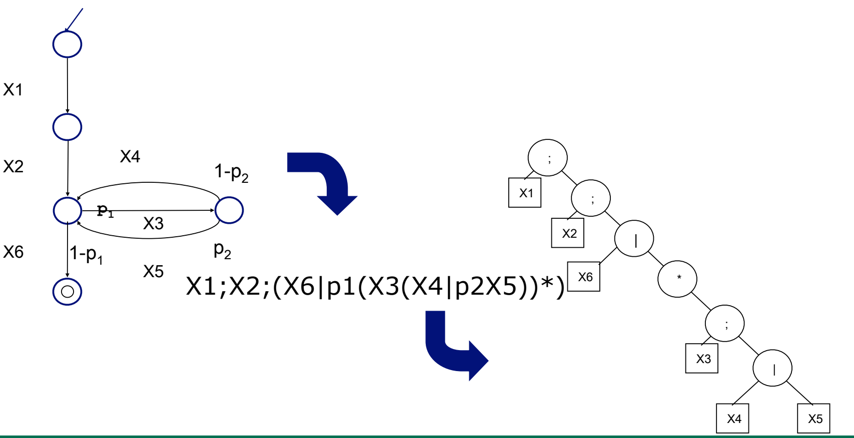
$$|f_j| = \sum_{n=0}^{N-1} x[n] \underbrace{|e^{-\frac{2\pi j}{N}ni}|}_{=1} = 1 \qquad j = 0,\ldots,N-1$$

$$= (1-p)\mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{y[\cdot]\}}{1-p\mathcal{F}\{x[\cdot]\}}\right\}$$

Above is the expression of the probability mass function for the loop once more.

We use the Fourier transform to prove the existence of the limit. The advantages of the Fourier transform is that the convolution becomes a product in the Fourier space. The discrete Fourier transforms for x and y exist, so we can

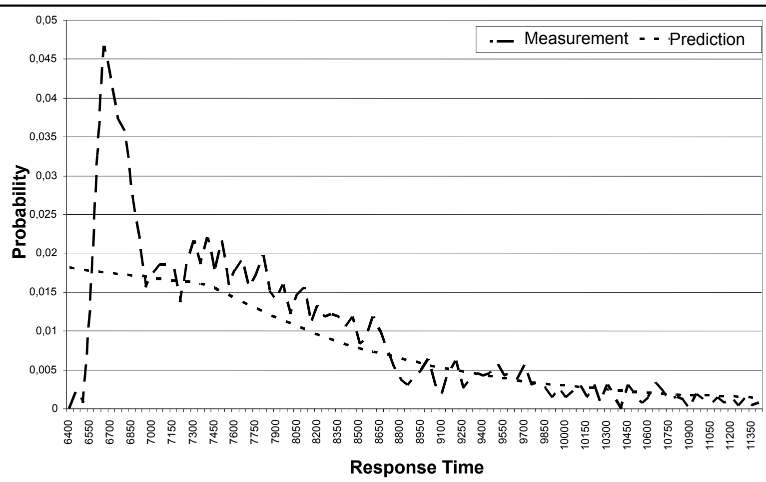
Systematische Berechnung der Verteilungsfunktion



43

On the slides above we have seen how to calculate the basic operators: alternative, sequence and loop. In this way we can subsequently calculate the probability mass function respectively the distribution functions of the method described by the service effect automaton. And this is the response time distribution of the whole

Validierung (1)



44

Modellverbesserung: Beliebige VF für Anzahl von Schleifendurchläufen

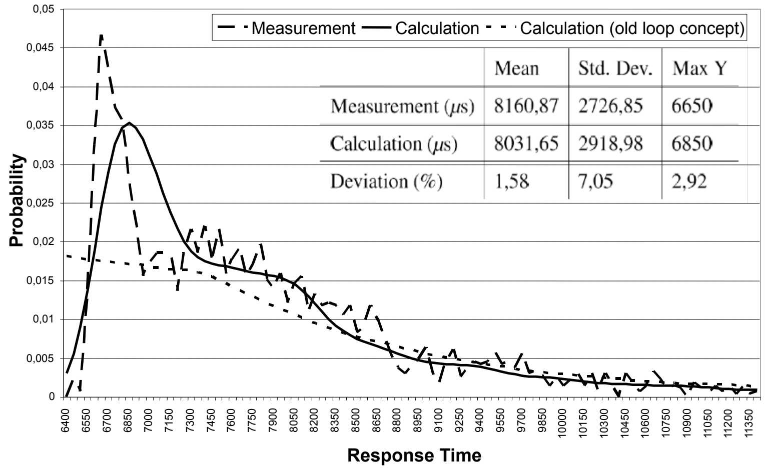
- Sequenz: $x_{R_1 \cdot R_2}(n) = x_{R_1} \circledast x_{R_2}[n]$
- Alternative: $x_{R_1 + R_2}(n) = p_1 x_{R_1}[n] + p_2 x_{R_2}[n]$
- Schleife:

$$X_{R^l} = \begin{cases} X_R & \text{with probability} \quad l(1) \\ X_R + X_R & \text{with probability} \quad l(2) \\ \vdots & \\ \sum_{i=1}^{N} X_R & \text{with probability} \quad l(N) \end{cases}$$

$$x_{R^l}(n) = \sum_{i=1}^{N} l(i) \overset{i}{\underset{j=1}{\circledast}} x_R[n]$$

45

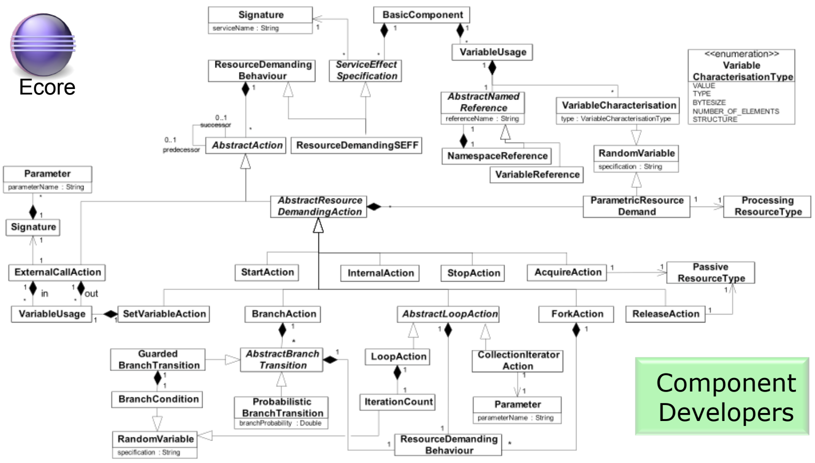
Validierung (2)



	Mean	Std. Dev.	Max Y
Measurement (µs)	8160,87	2726,85	6650
Calculation (µs)	8031,65	2918,98	6850
Deviation (%)	1,58	7,05	2,92

46

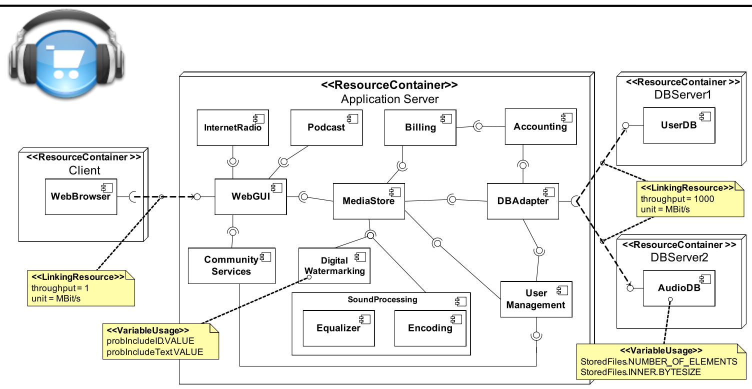
Service Effect Specification (2)



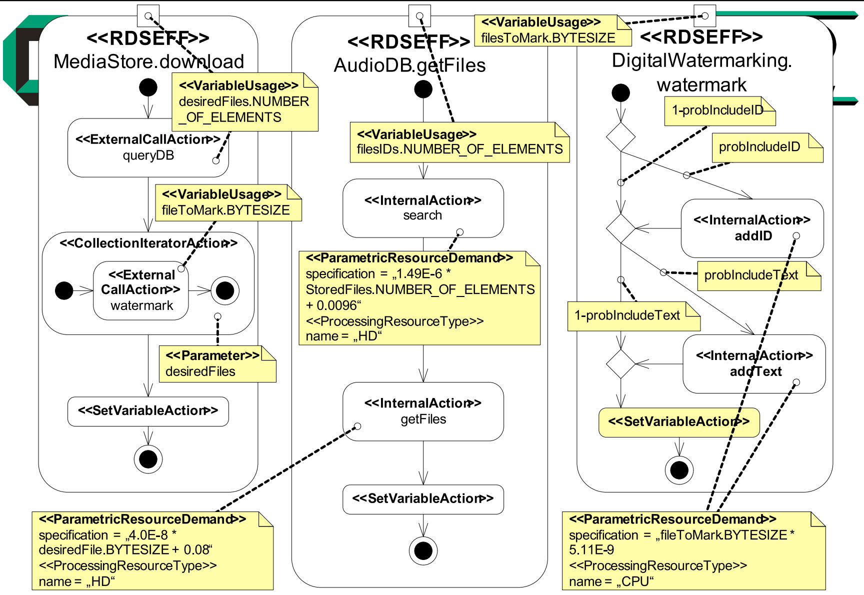
Component Developers

47

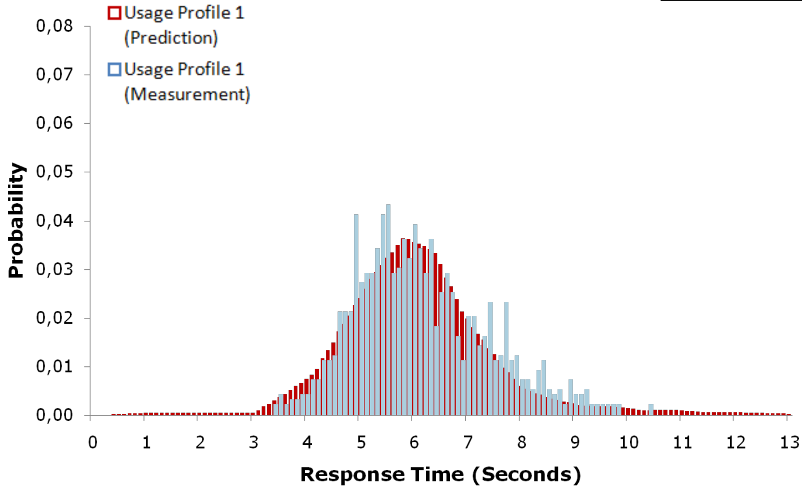
MediaStore - Architecture



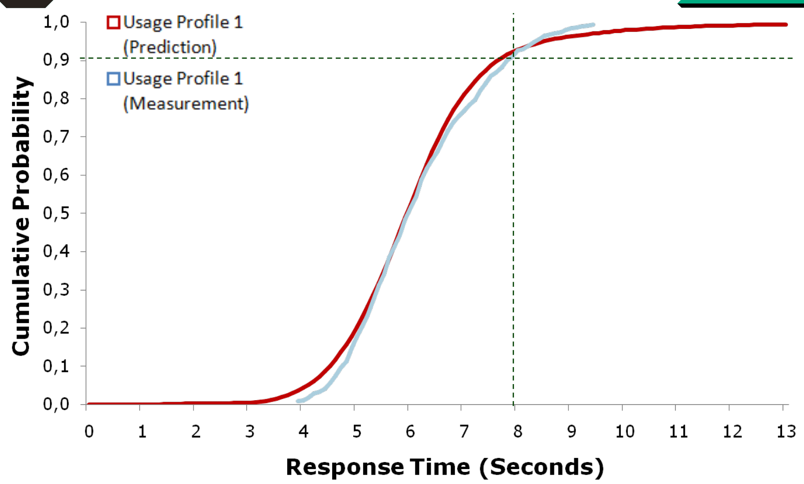
48



Results



Results



Lessons Learned

- Model-centric development instead of code-centric development
- Without an architecture you won´t have fun in re-use, evolution, organisation, planning, non-functional properties
- But front-end costs are increased
- You have to be familiar with modelling and certain techniques to benefit from architectures
  – reuse (pattern, product-lines, etc), planning
- and do not forget the three views:

52

Architectural Views

- Used to document an architectural design.
- Static structural model that shows the major system components.
  – Interface model that defines sub-system interfaces.
- Dynamic process model that shows the process structure of the system.
  – Relationships model such as a data-flow model that shows sub-system relationships.
- Deployment model that shows how sub-systems and connections are mapped to ressources, such as processors or network connections
  – distribution across computers.

53

Copyright Notice

- Some of the slides are taken from Sommerville, Software Engineering 7th Ed.

54

Conclusions

- Prediction and Understanding of the Consequences of Design Decsions is THE central characteristic of an engineering discipline.
- Components and MDD lower the degrees of freedom in implementation
- Creativity is on design-model level
- Quality-driven Design requires prediction models
  - automatically generated from design models
- Definition of design and prediction models follows the scientific process of the natural sciences.
  - No proofs possible, but empirical validations necessary

Software Engineering becomes “architecture-centric”.
Code-centration is as meaningful as
“**brazing solder-centration**” of an Electrical Engineer

http://www.palladio-approach.net

http://sdq.ipd.uka.de