UNIVERSITÄT DES SAARLANDES

System Vision, System Idea & Views

Architectural Thinking for Intelligent Systems

Winter 2020/2021

Prof. Dr. habil. Jana Koehler

Agenda

- Formulate the System *Vision*
- Develop a first *System Idea*
- How to work with *Views, Viewpoints & Perspectives*
  - Context view
  - Component view
  - Runtime view
  - Distribution view
- Creating work products and documenting architectures
  - Relationship between application/solution architecture and enterprise architecture frameworks

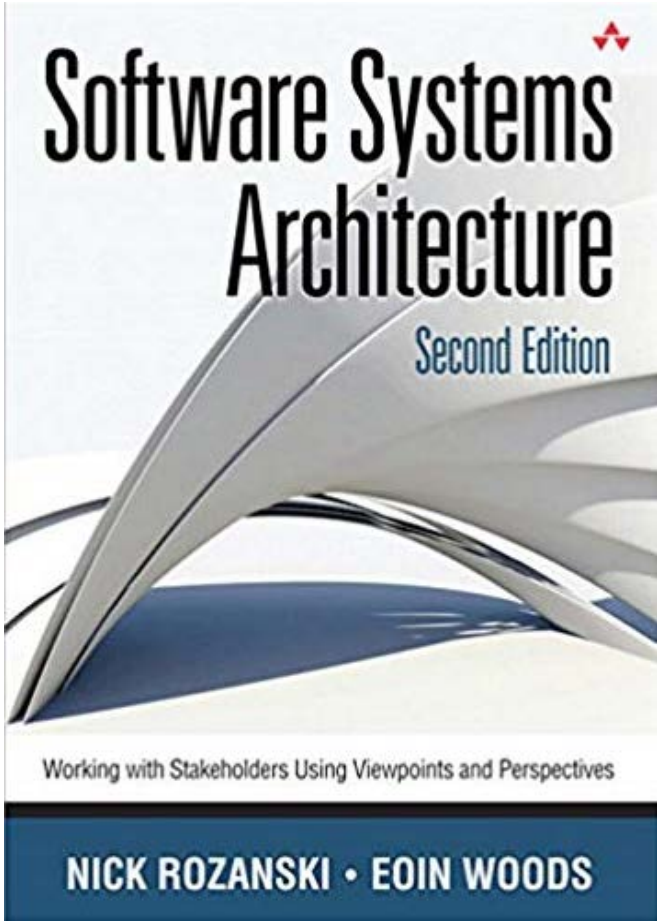
Assignment for this Lecture

- We formulate the vision for our system and sketch a first idea of its architecture expressed in several views
  - Context view
  - Component view
  - Runtime view
  - Distribution view

Recommended Reading

- Rozanski/Woods:
  - Chapter 3 (pages 31-43)
  - Chapter 4 (pages 45-61)

- Optional reading:
  - Vogel et al. Chapter 4, notably Chapter 4.1 (pages 76-92) introduces and explains relevant architecture frameworks defining views



4 Work Products an Architect must do Right

1. **Vision**
   - What goals will the system achieve, what values provide to customers, and how differentiates itself?
2. **Context View**
   - In which environment do we want to be successful?
   - How do stakeholders interact with the system?
3. **Architecture overview**
   - How is the system structured?
   - AS-IS vs. TO-BE in case of system evolution
4. **Operational Model**
   - How will the system "live“?
   - What infrastructure wil be needed? How much will it cost? Can it scale in case of success?

Writing a Vision Statement

For [customer]

Who [needs... or has opportunity]

The [product]

Is [category]

That [major capability, benefit, ...]

Unlike [alternative, current system, current practice]

Our Product [differentiation, advantages, ...]

Developing a VISION – Port Technology Example

- High-rise building with genuine mixed use containing office, residential and public areas
- Individual & efficient mobility
- Offer passengers an optimal route through the building to the desired destination. The seamless journey starts upon entering the building with the assigned elevator already waiting at the ground floor.
- The intelligent combination of traffic control and access control creates maximum efficiency and meets the requirements of the individual user groups in the building.
- Whether it is a design improvement that enables a new building to achieve far more than originally envisioned or an upgrade to give an existing building a new lease on life, The PORT Technology can act as a major enabler of positive change.

https://www.theporttechnology.com/page/solutions.html
https://www.schindler.com/com/internet/en/media/behind-the-scenes/customer-projects/2018/omniturm-frankfurt.html

Developing *The* System Idea

- We find the system idea by applying principles and tactics to our architectural problem and develop adequate solutions

- Creative process
- There is more than one solution
- Find compromises
- Ask the right questions and reuse proven solutions

- The system idea can be expressed in a component view

Find clear answers to these questions

- Which NFR are the most important ones for the architecture?
- Which use cases at level “white” should we put into the main focus?
- What will be the core task of the system?
  - What are the most important elements of the business domain?
- How will the system be used?
- How will the system be controlled?
- Who uses the system?
  - What kind of user interface will the system need?
  - Which business processes and user interactions will the system support?
- What interfaces to other systems will the system need?
- How will data be managed & processed?
  - What kind of data access is necessary?

➢ **Which structural components and relations can best implement the answers to these questions?**

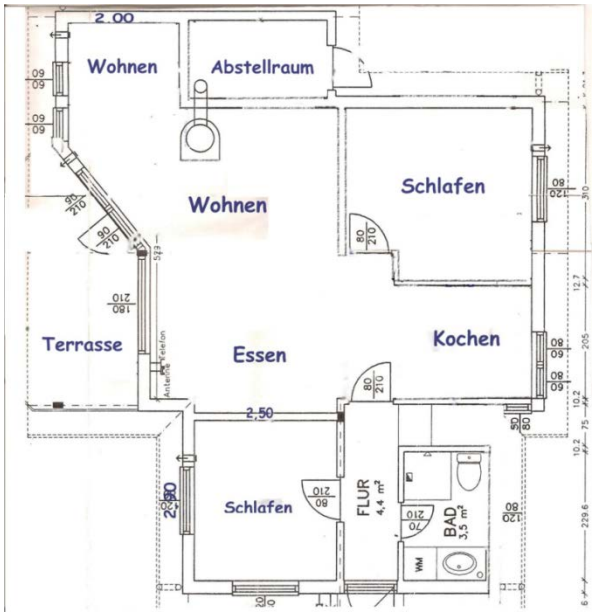
Views and Viewpoints [ISO/IEC/IEEE 42010:2011]

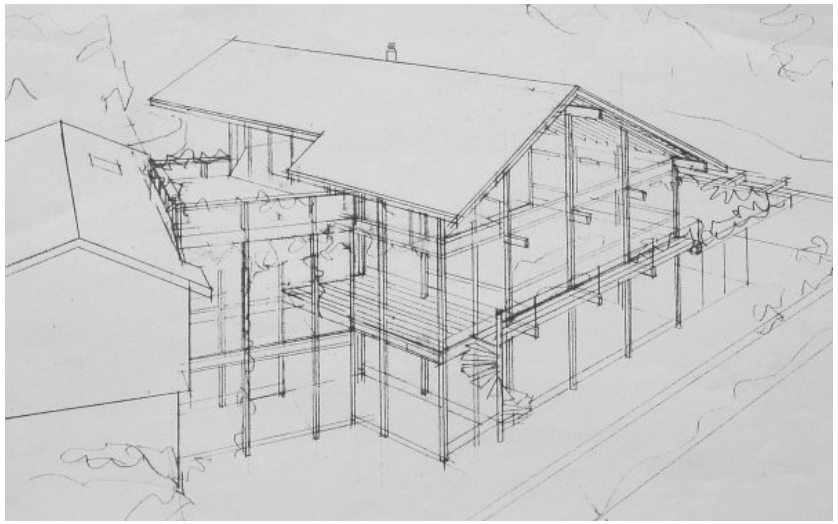
- A *view* is a work product expressing the architecture of a system from the perspective of specific system concerns
  - View as a set of related architecture-relevant elements that is created and used by the stakeholders of a system

- A *viewpoint* is a work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns
  - collection of patterns, templates and conventions for constructing one type of view. It defines the stakeholders, guidelines and principles and template models for constructing the views governed by the viewpoint

Views in Building Architecture

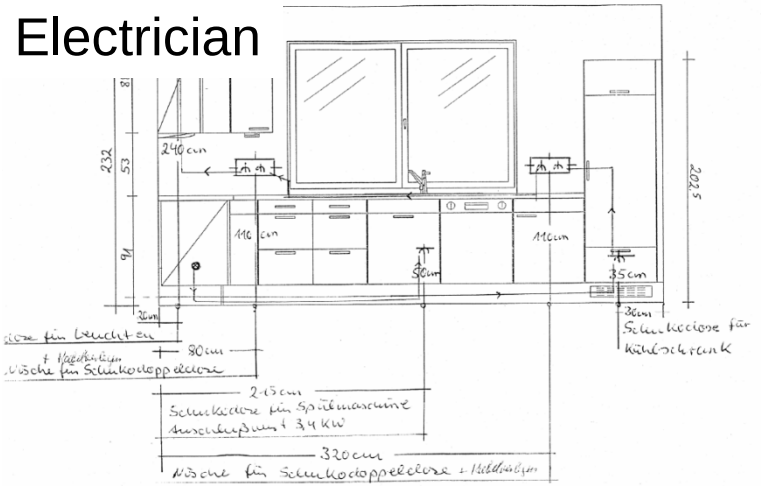
Floor plan



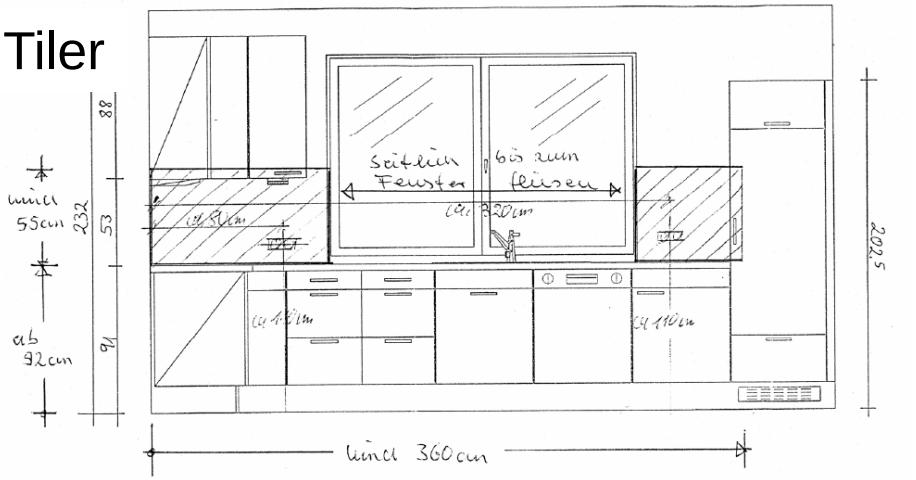


Vertical plan

Electrician

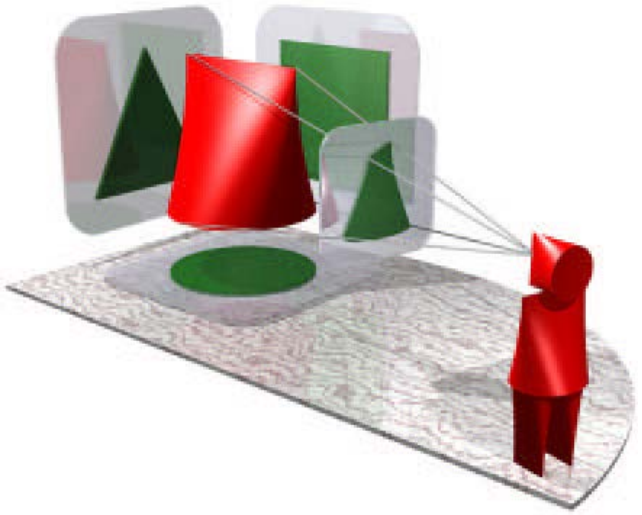


Tiler



Why Views are Important

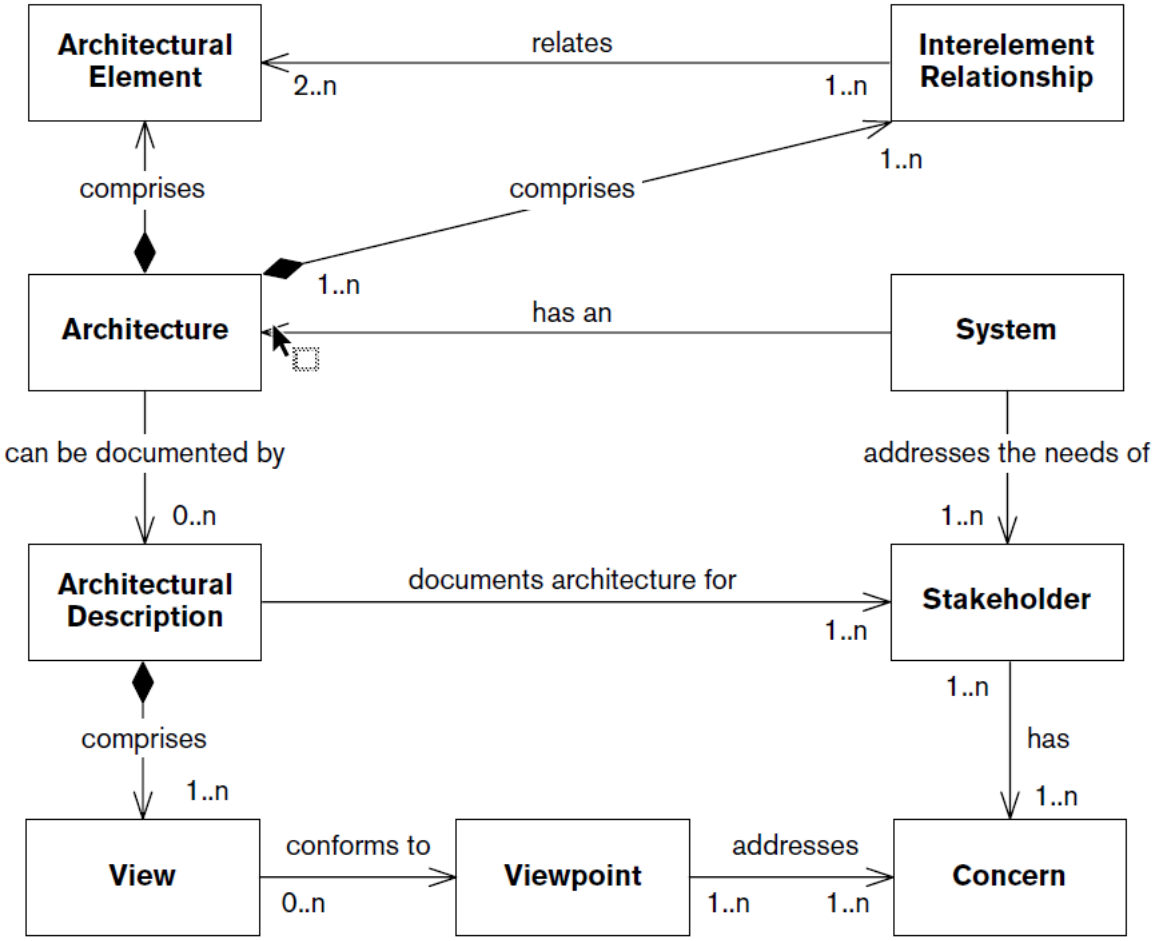
- System structure and architectural decisions must be documented and communicated



- What?
- For whom?
- Why?
- How?

"If it is not written down, it does not exist."
Philipp Kruchten

Role of Views and Viewpoints



Rozanski/Woods, p. 37

Why Views?

- Modern software is often too complex to be viewed and understood as a whole
- We need to focus our attention on one (or a few) parts of the system structure to facilitate understanding
- In order to be able to communicate clearly, we need to make clear which structures, relationships or behavior focus on during discussion

- VIEW as a representation of the selected structures
- Architects develop structures and document these structures through views

Creating GOOD Views

- Meeting stakeholder´s needs
  - What information does a stakeholder need?
- As little formalism as possible, as much as necessary!
- The more risk involved with a decision, the more detailed the view illustrating it
- Work top-down/bottom-up and in iterations to refine and revise views

How can we Assess the Quality of a View?

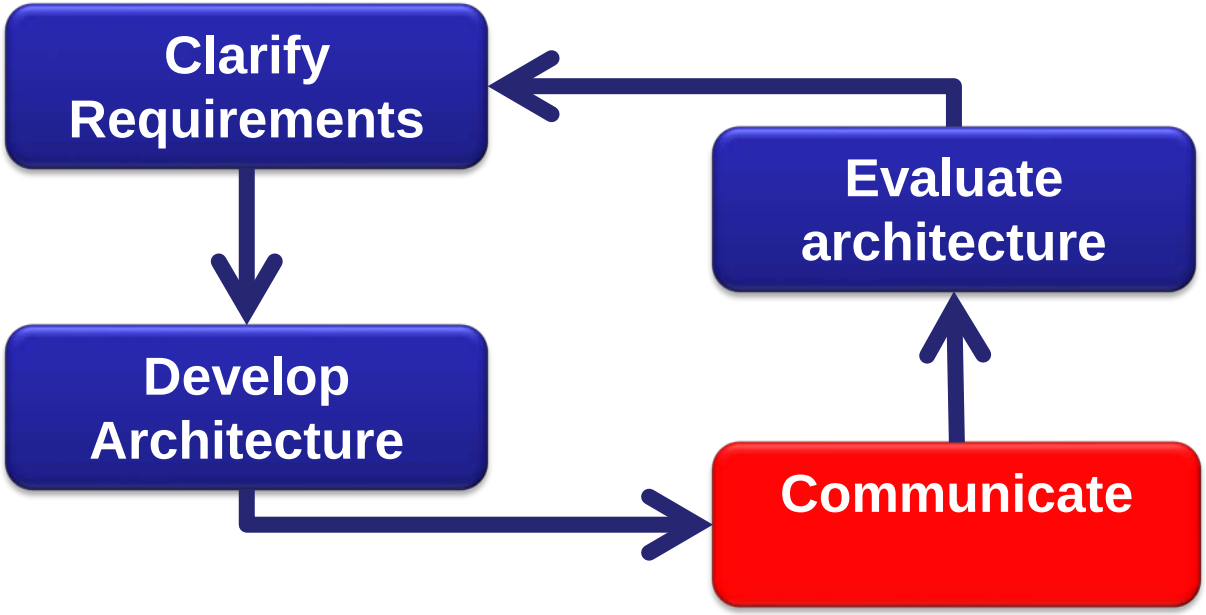
- Need to ask *"good for which purpose?«*

- What are we trying to achieve?
- What do we need to communicate to whom?

- If the view delivers our intended message correctly and successfully to the stakeholder …
  - … and we reach the goal of our message

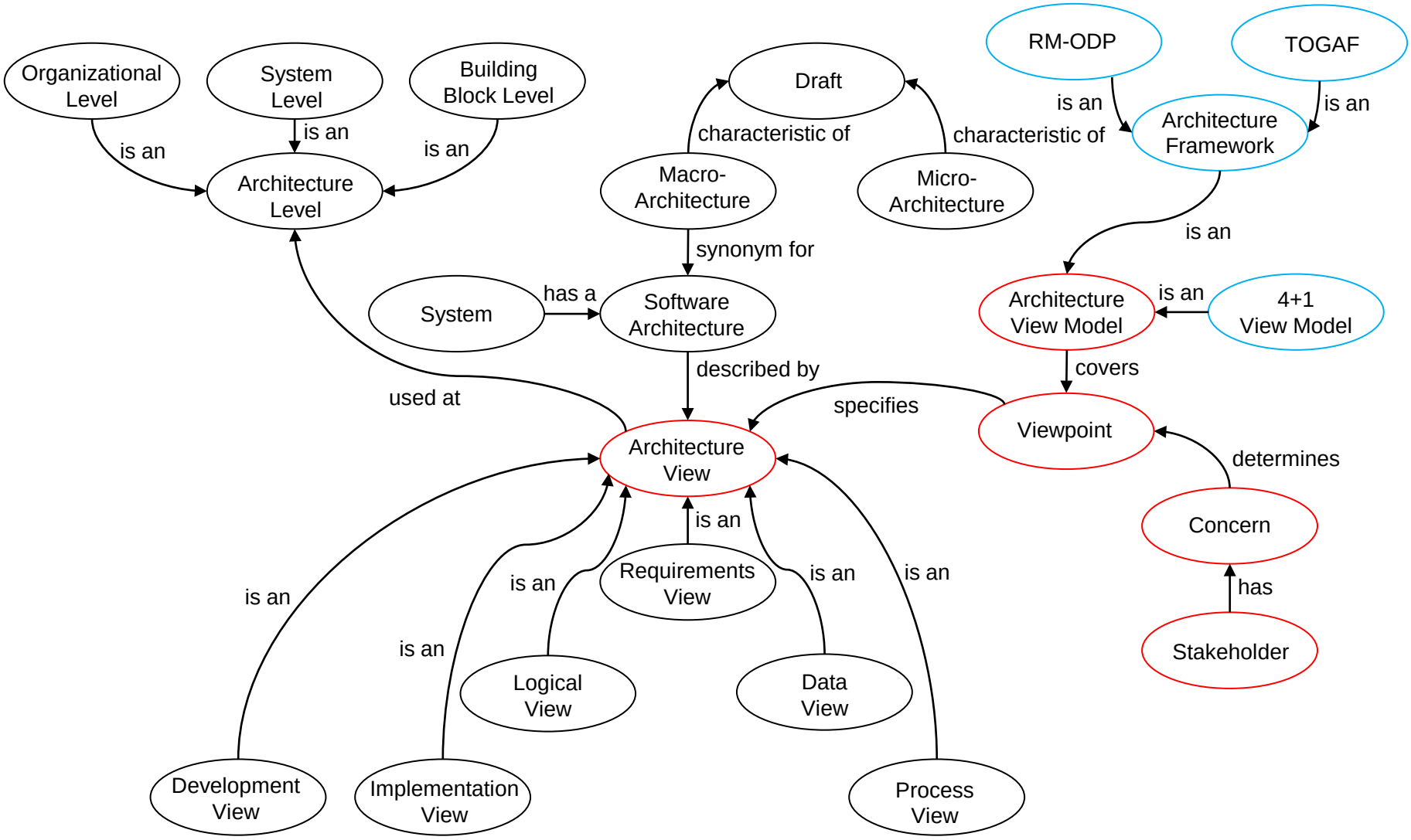
➢ Then it is a good view!

Views in the Communication of the Architect



- Motivate decisions
- Propagate drafts
- Communicate architecture to the team

- Different views (abstractions) for different participants



Vogel et al: Software Architektur

The 4+1 Views Model by Kruchten

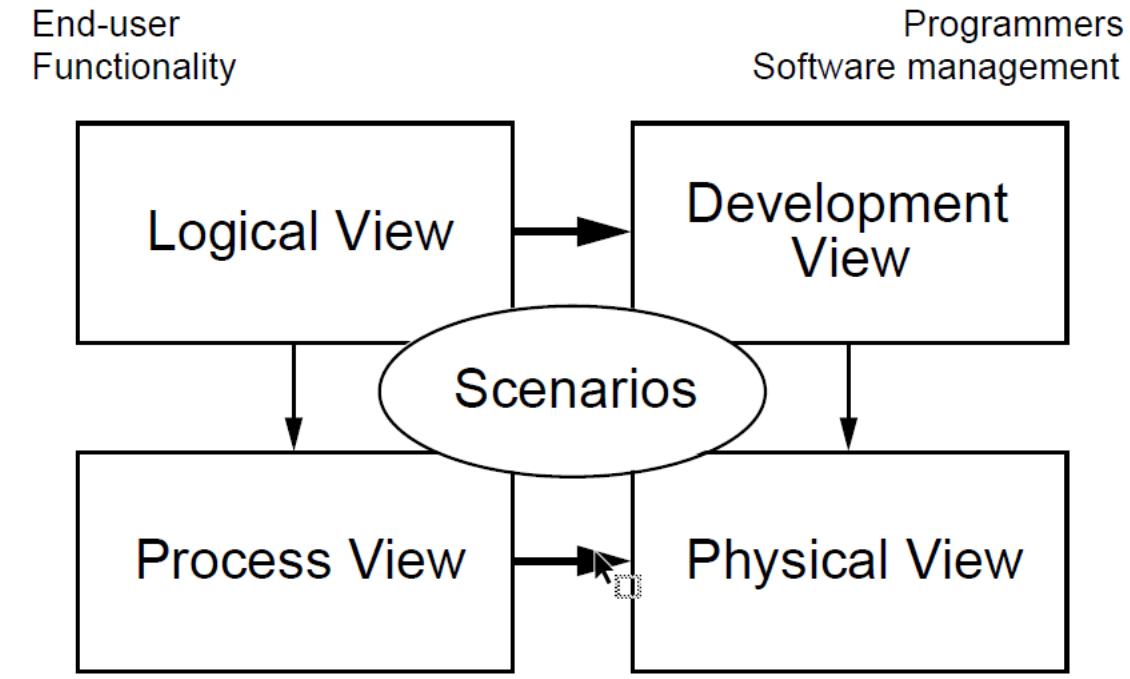


Figure 1 — The “4+1” view model

Philippe Kruchten: Architectural Blueprints—The “4+1” View Model of Software Architecture IEEE Software 12 (6) November 1995, pp. 42-50

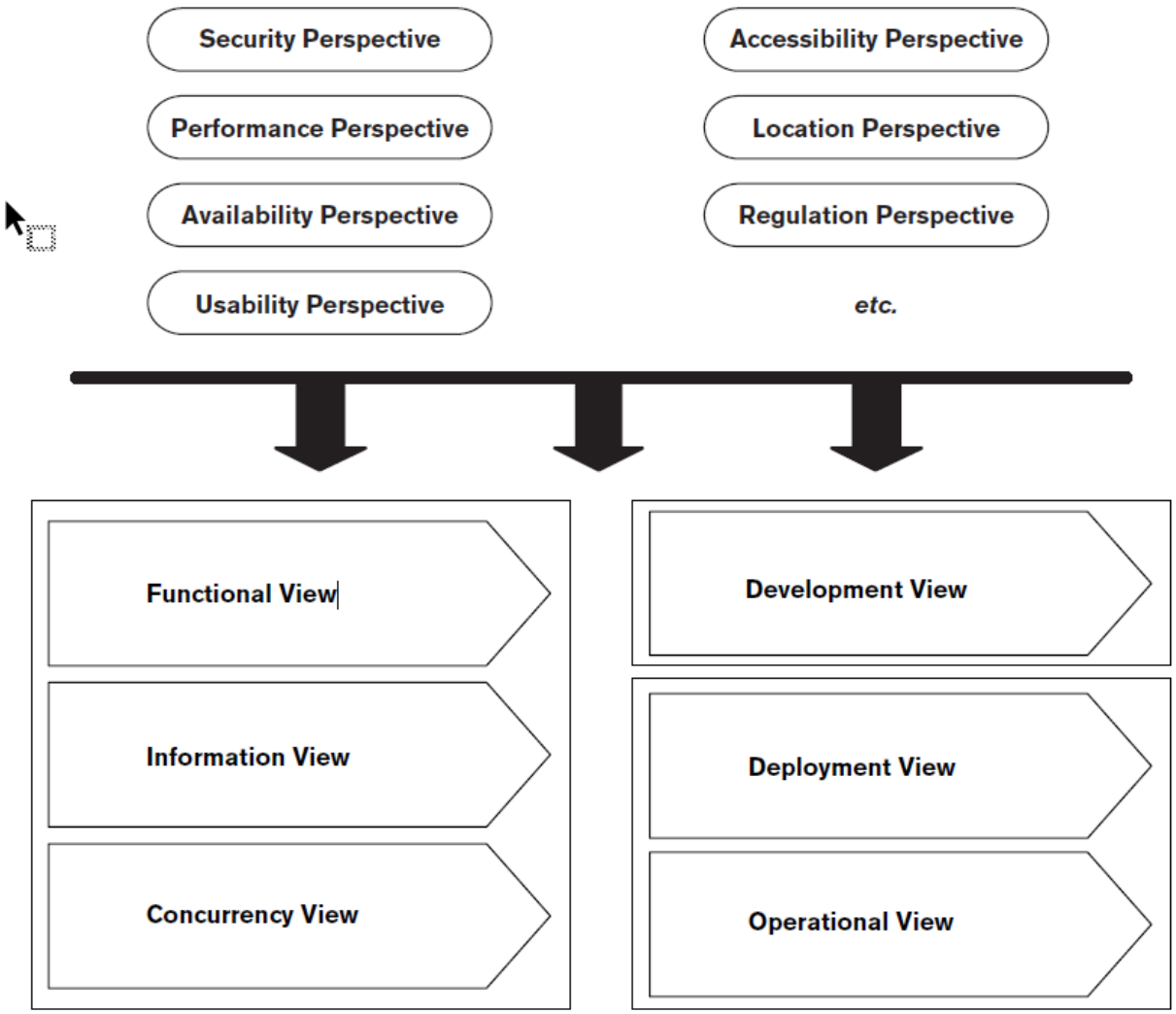
Perspectives in Architectural Thinking

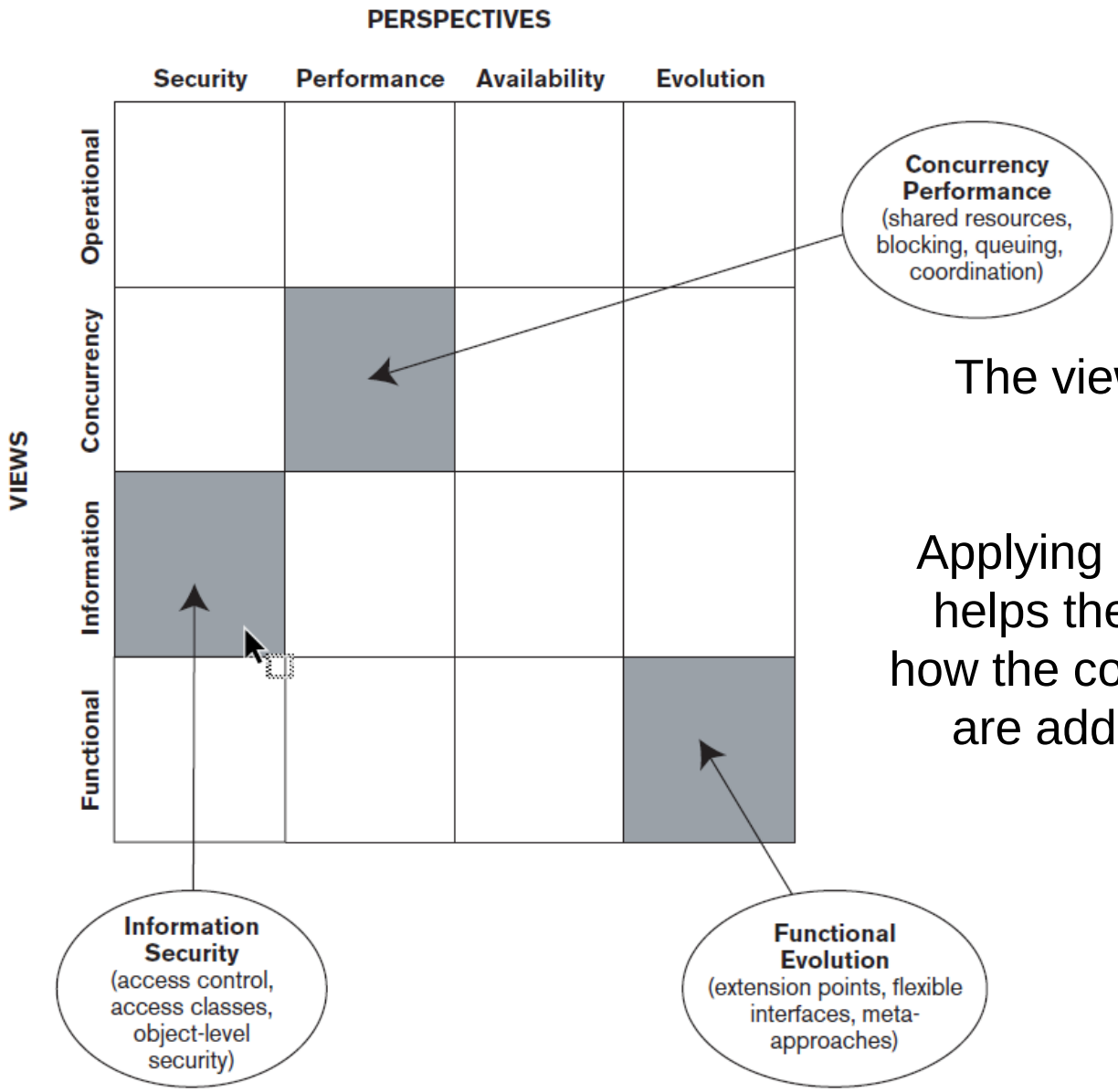
- Many architectural decisions address concerns that are relevant for all views, e.g. security
  - „cross-cutting concern“ - mostly quality attributes
- Architectural perspective as a means to express these cross-cutting concerns orthogonal to views
  - apply perspective to a view to make sure that the system as documented in the view exhibits the desired quality attribute expressed by the concern

- „An architectural perspective is a collection of architectural activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the systems´s architectural views.”

Rozanski/Woods: 2nd edition, page 47

Applying Perspectives to Views





The view gets shaped by the perspective!

Applying a perspective to a view helps the architect to reflect on how the concerns/quality attributes are addressed by the system architecture.

4 Views We Focus on

- Context View
- Component View
- Runtime View (behavioral model)
- Distribution View (operational model)

The Context View

- **How is the system embedded into its environment?**
  - Interfaces to to all(!) neighboring **systems**
  - Interfaces to all **users**
  - All **events** that can enter/leave the system
  - Any **data** the system **sends to /receives from environment**

- The system itself is a black box
- Our attention concentrates on the environment and how the system must interact with all elements in it
- All incoming and outgoing data and events
  - Interactions of the system with any stakeholder
  - Relevant parts of the external infrastructure

Importance of the Context View

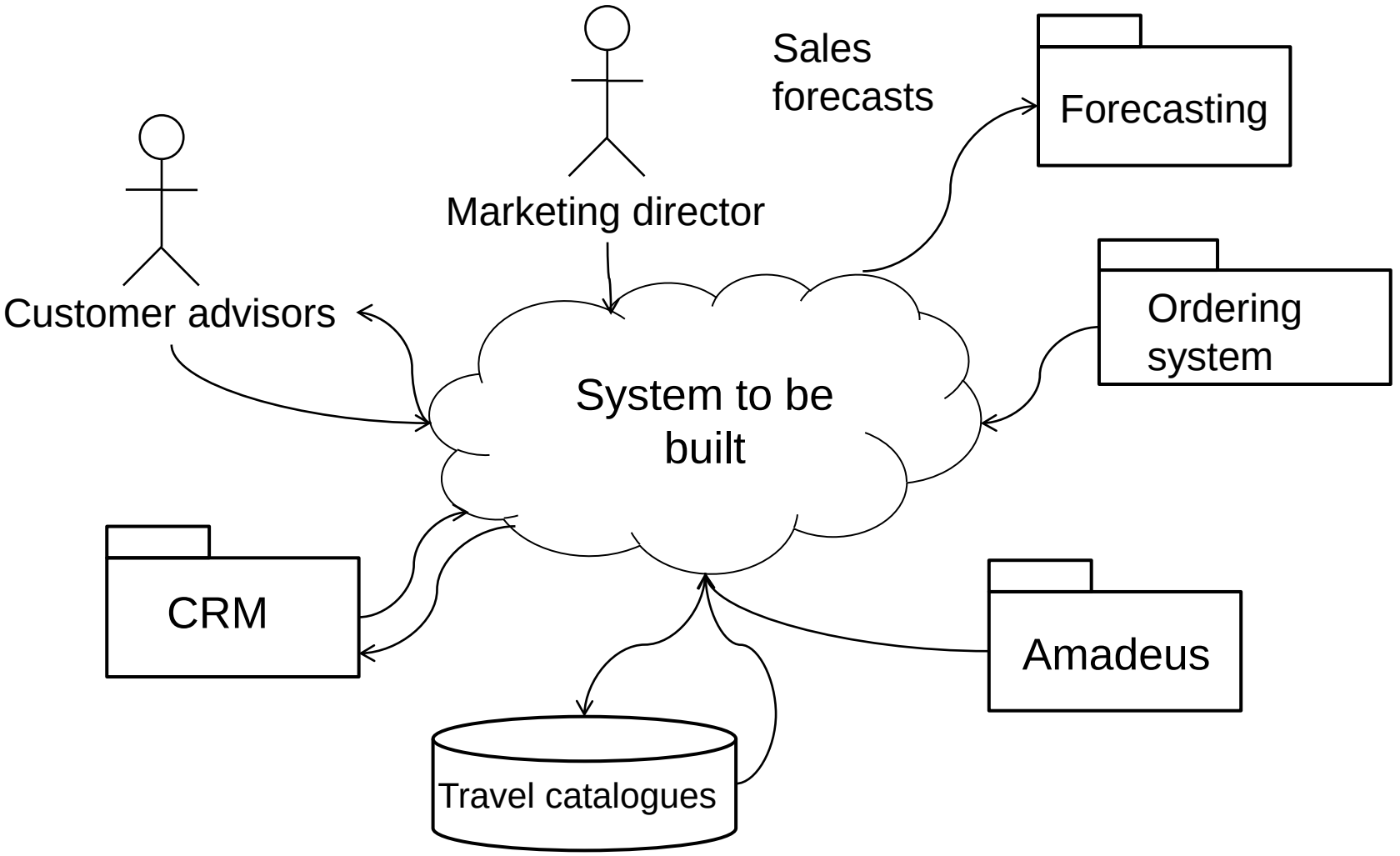
- As an overview diagram of the system environment

AND

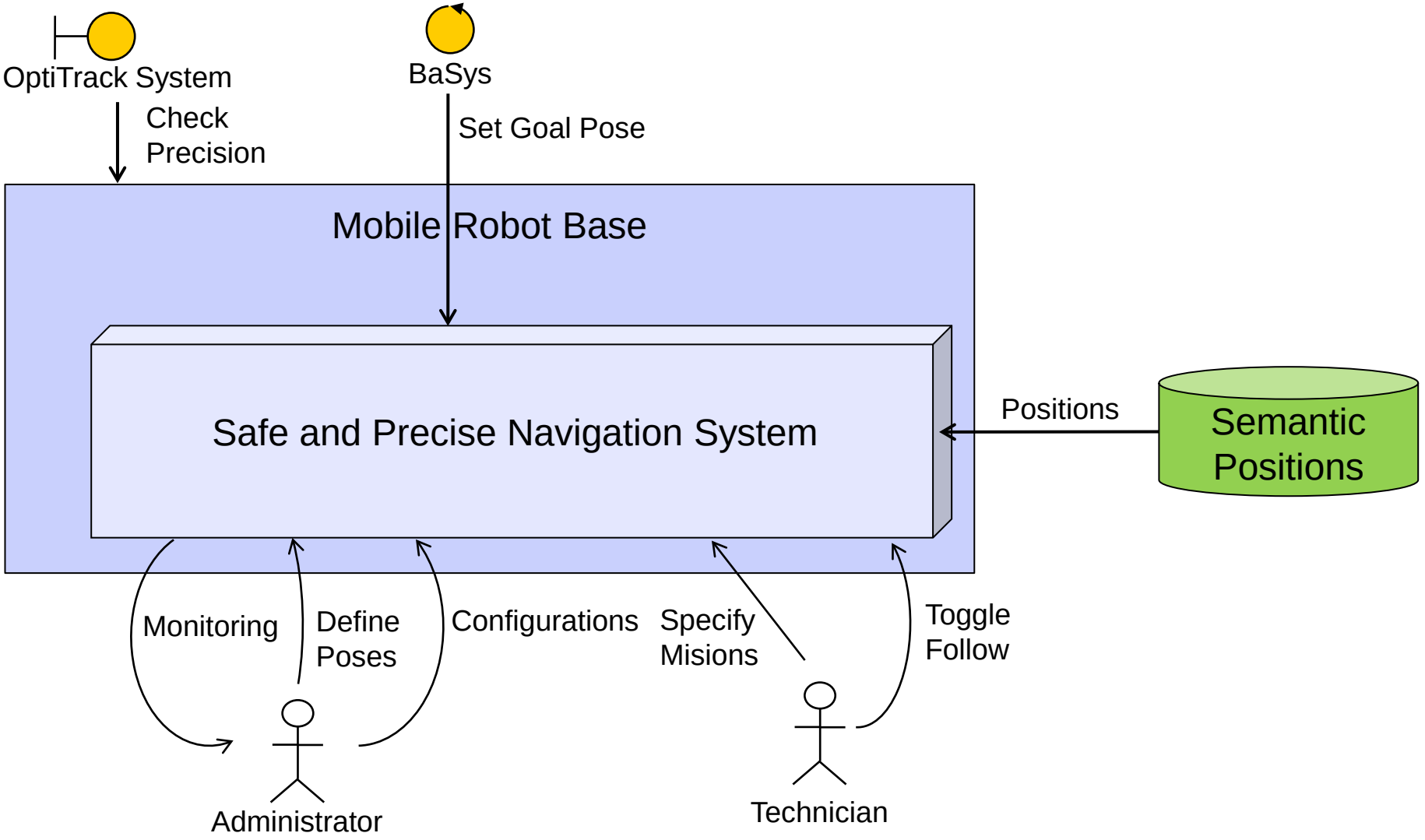
- As the view for resource planning and risk management
  - Number & type of external interfaces
  - Number & type of dependencies on external systems
  - Number & type of stakeholders involved

- Context views with different levels of detail
  - Always create a detailed view when negotiating the scope of the project!

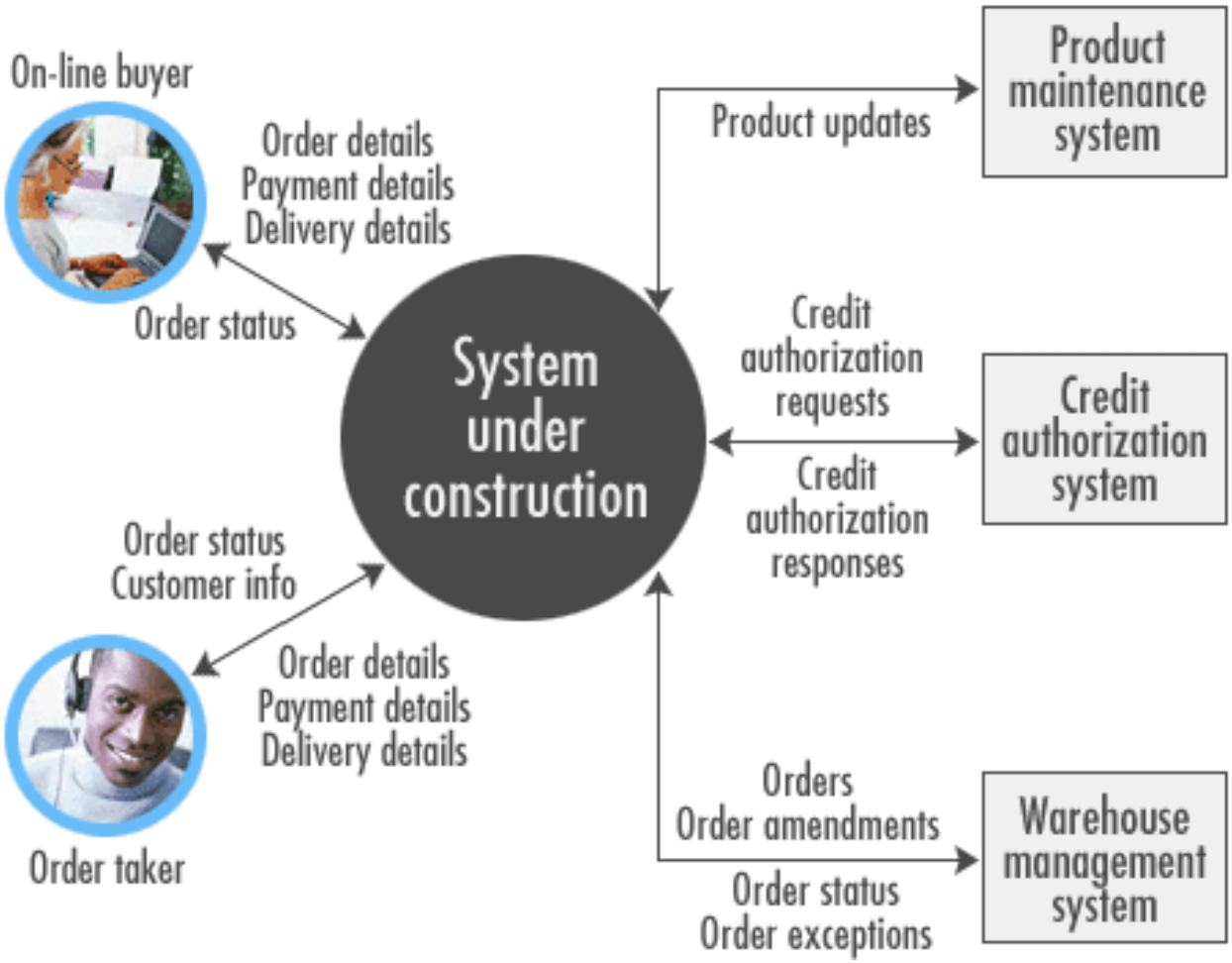
Example Context View - Antipattern: Missing Data & Events



Context View - Antipattern: Activity Focus instead of Data & Event Focus

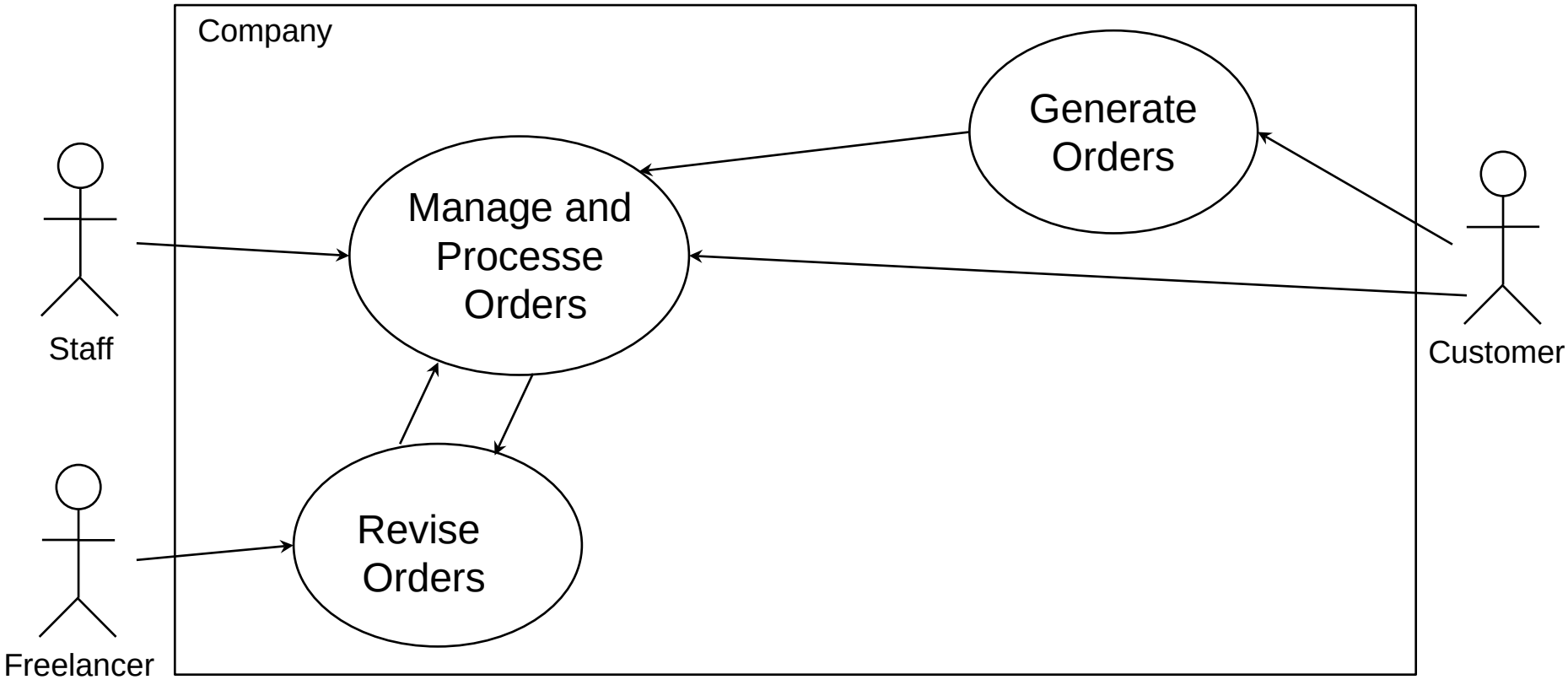


What can we say about the System based on this Context View?

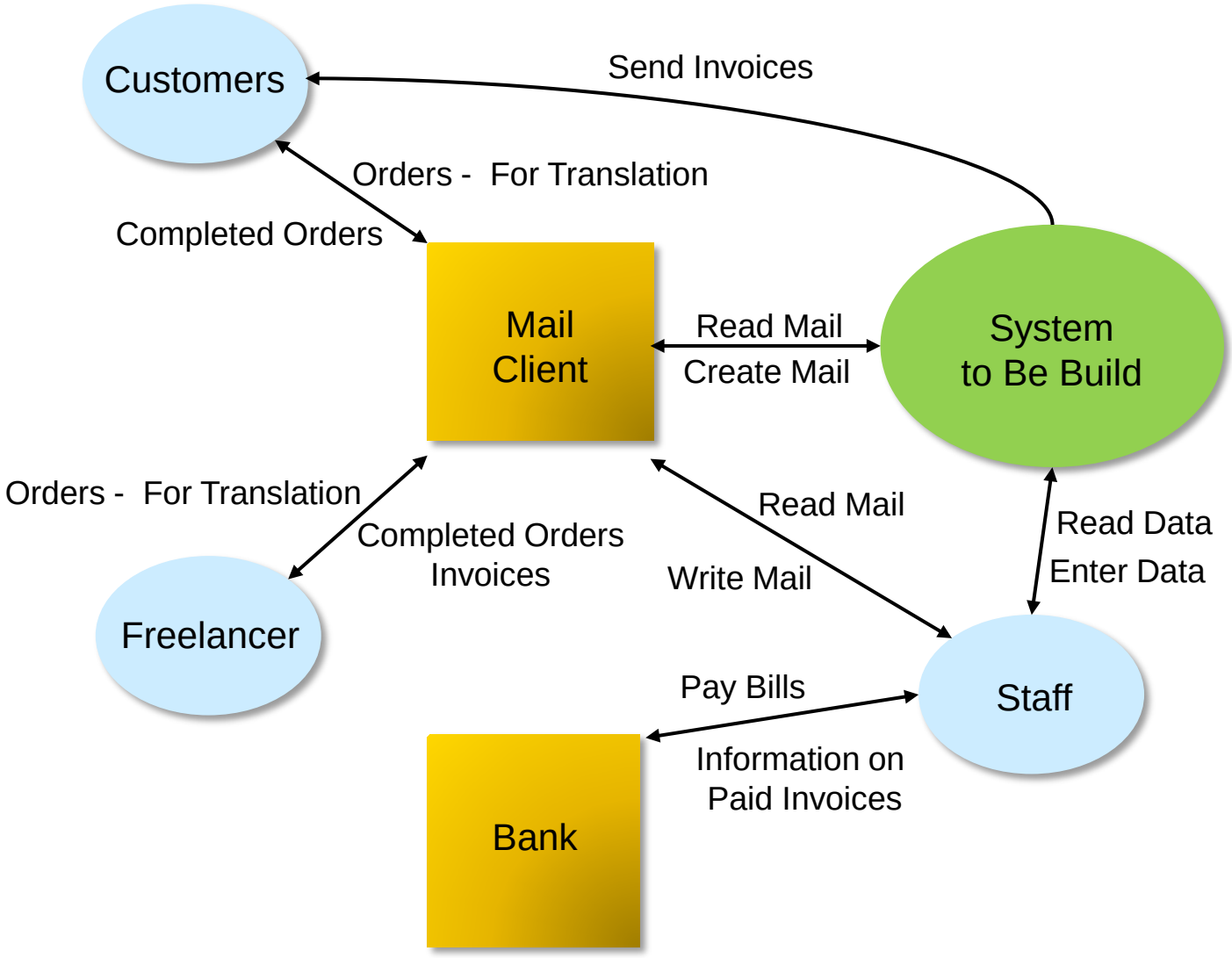


Quelle: IBM

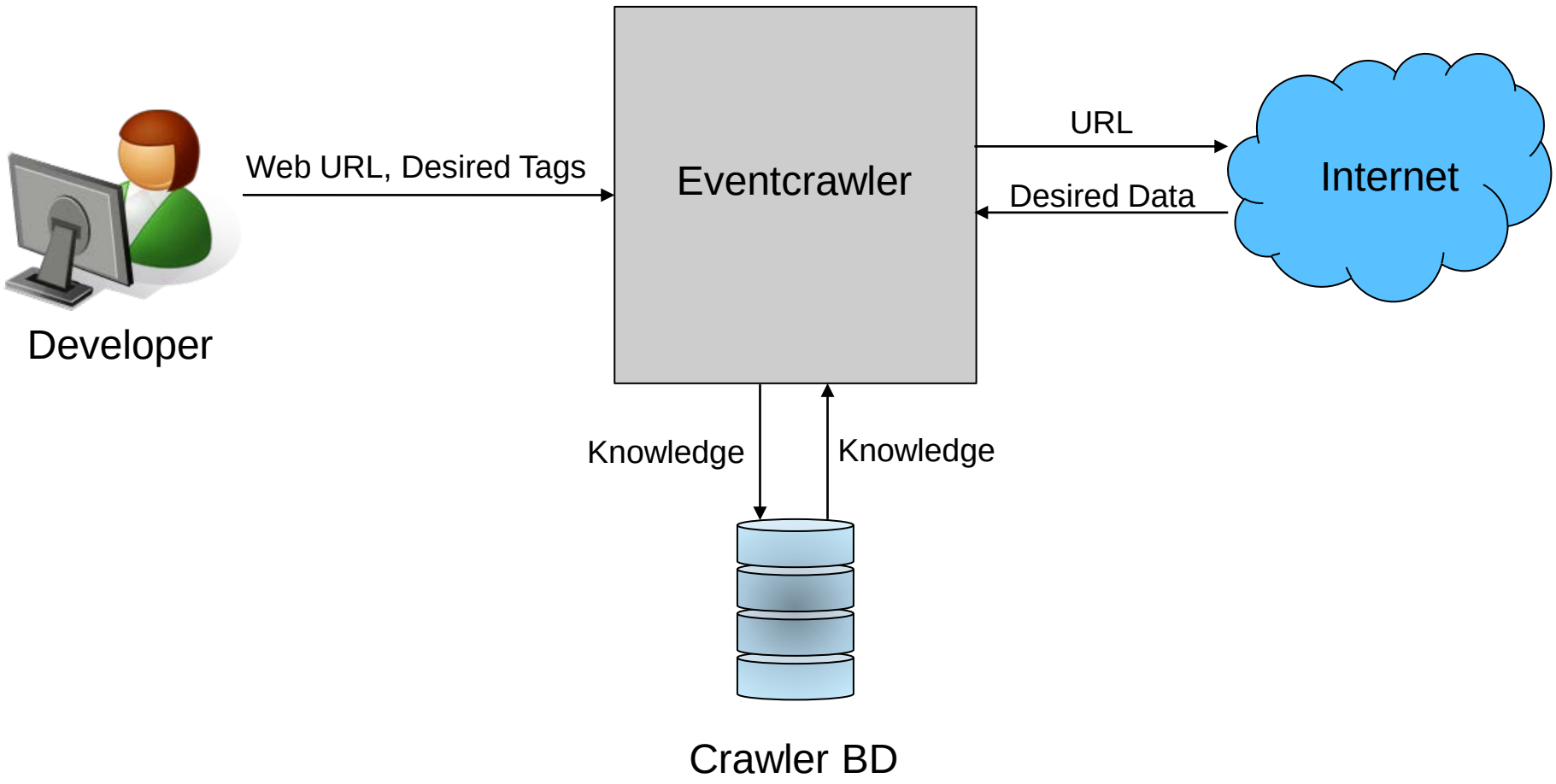
Context View - Antipattern: Believing it is Simple



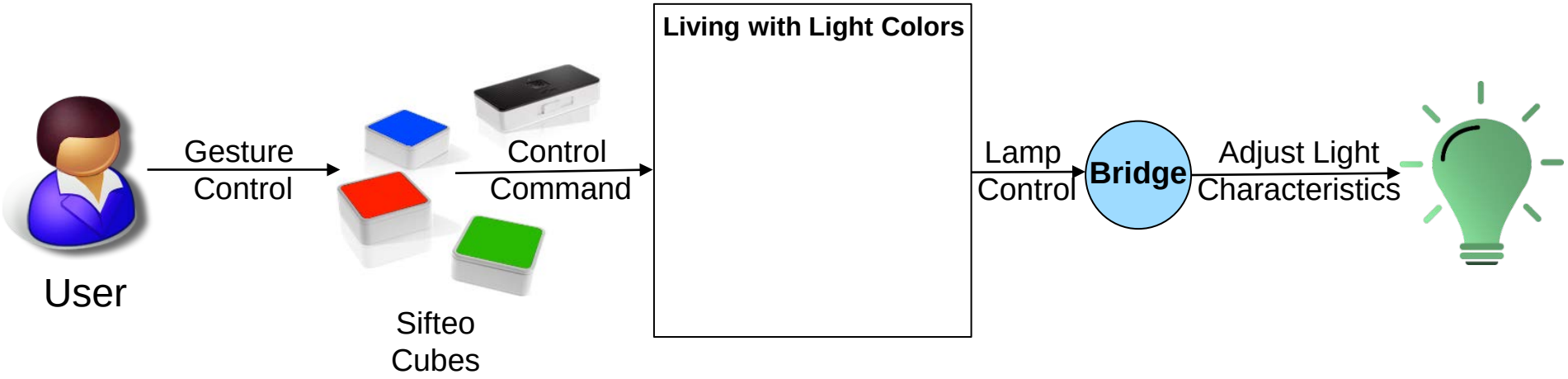
2nd Iteration



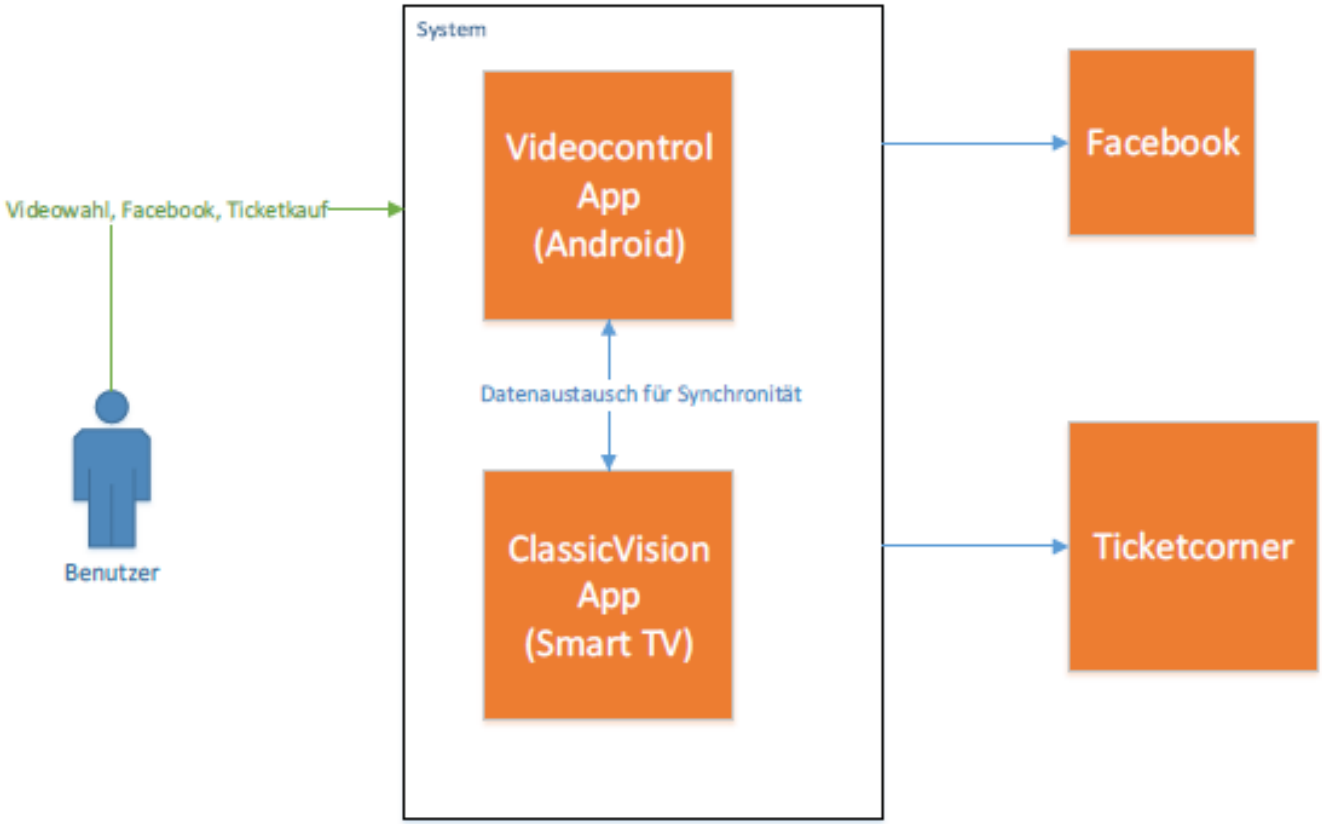
Context View - Antipattern: Abstract Generic Naming



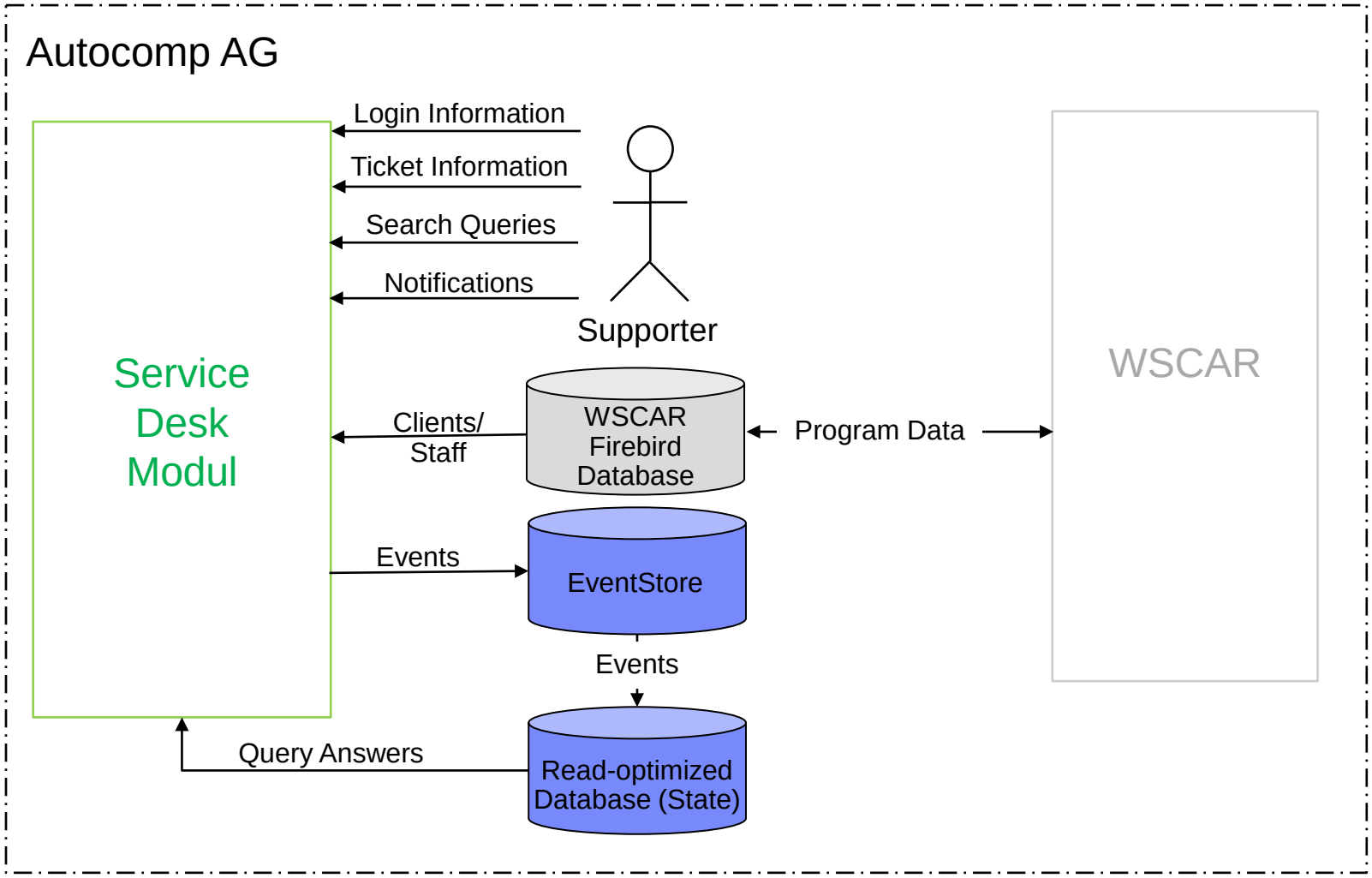
Context View - Antipattern: 2nd Level Environment Elements



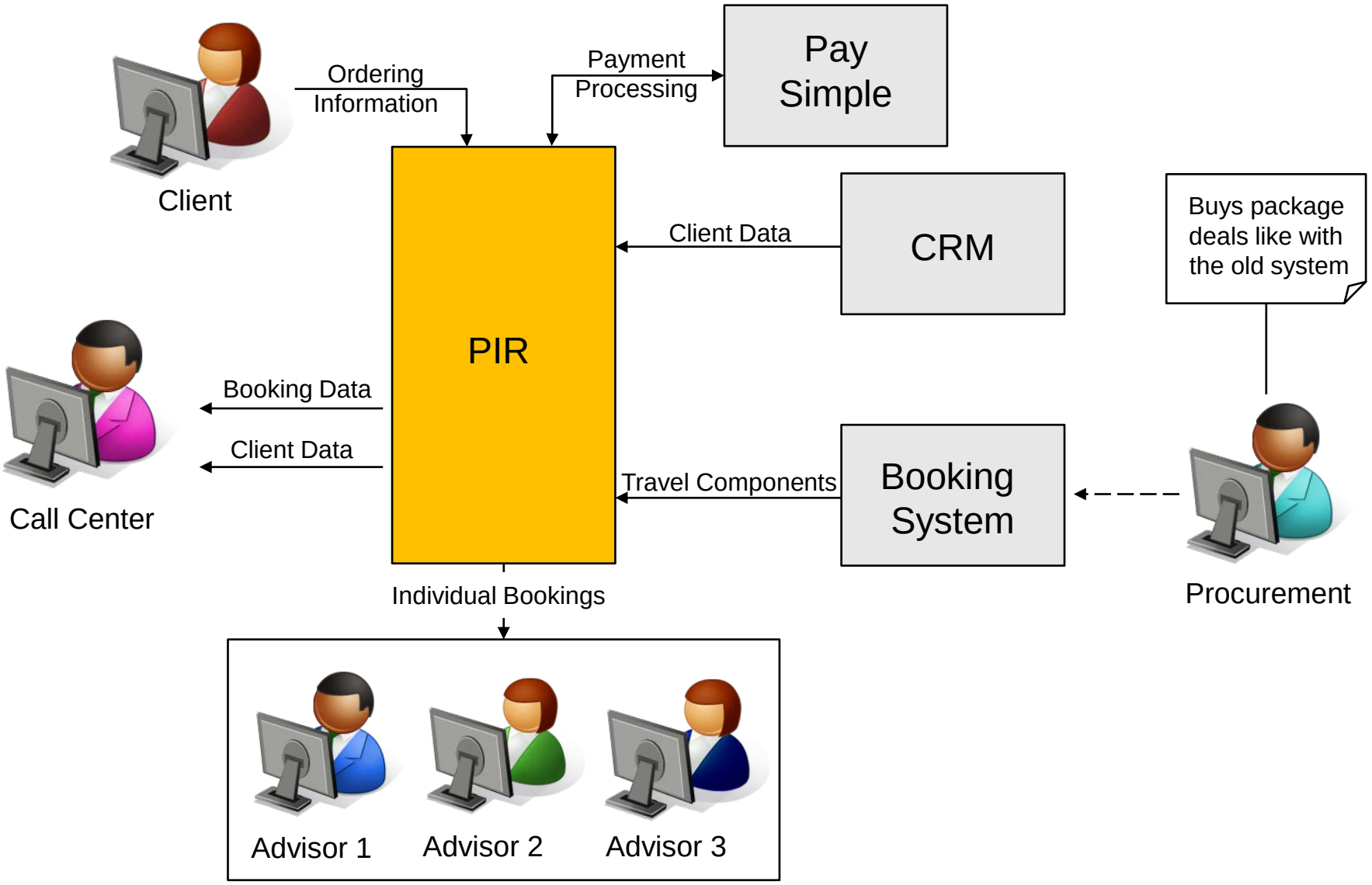
Context View - Antipattern: Showing System Internals



Context View - Antipattern: Unclear System Boundaries



Context View - Antipattern: Forgetting Bidirectional Data Flows



Component View

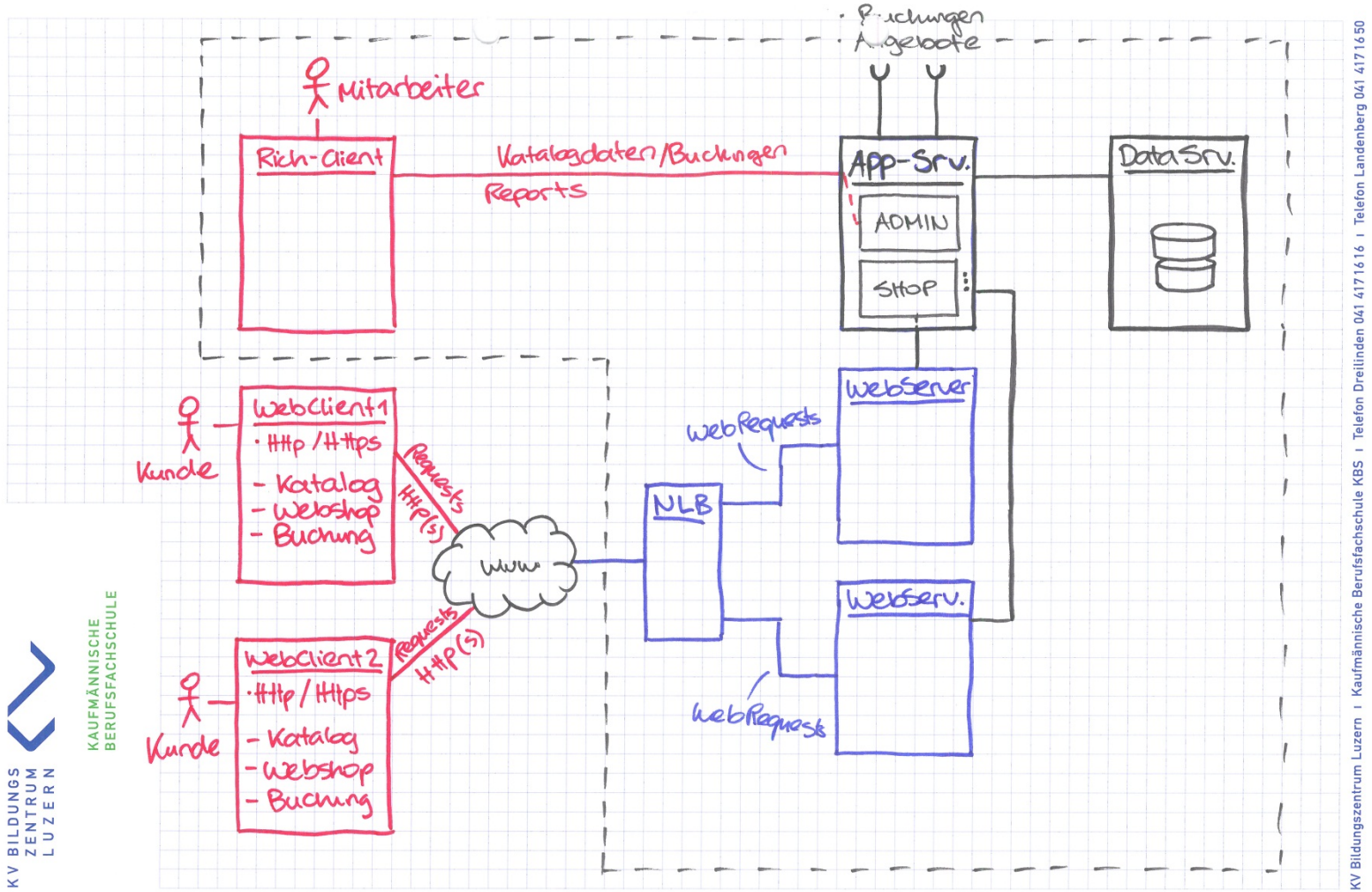
- **Shows the decisions made wrt. the internal structure of the system?**

- Static structures of the architectural building blocks of the system
  - Subsystems, components, their interfaces and relations

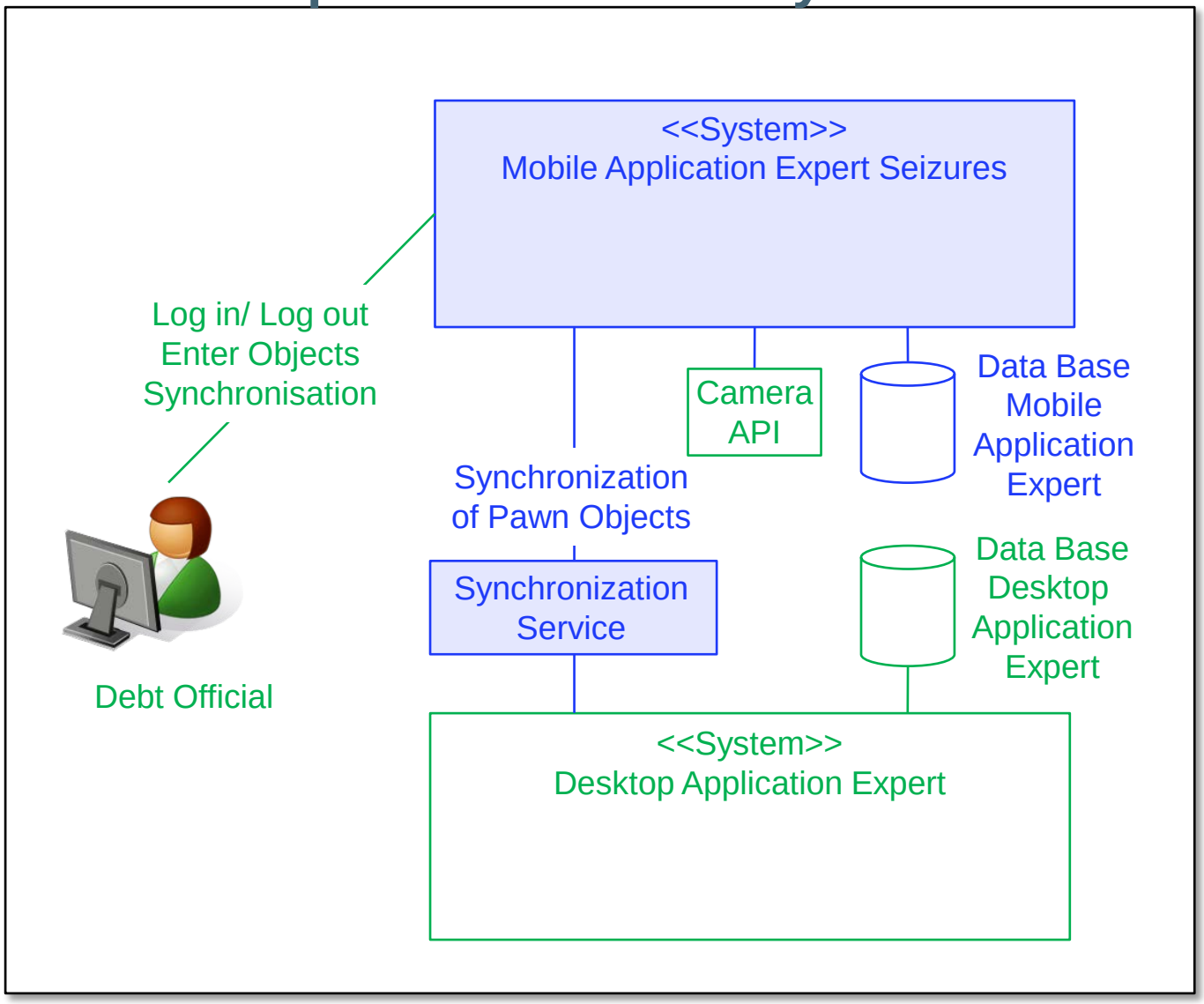
- Support project managers and clients in project monitoring
- Allocate work packages (architecture modules) to teams and employees

- Top-down refinement
  - Last (possible) level of refinement is the source code

System Idea - Antipattern: Cryptic Naming

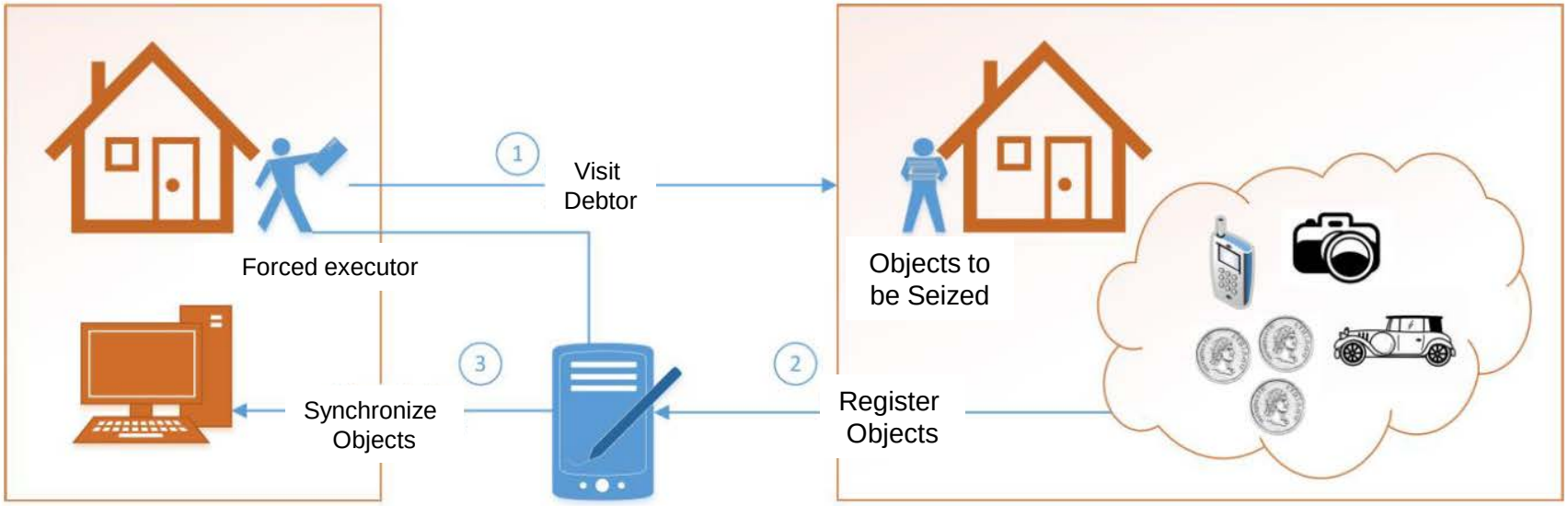


System Idea - Antipattern: Unclear System Boundaries



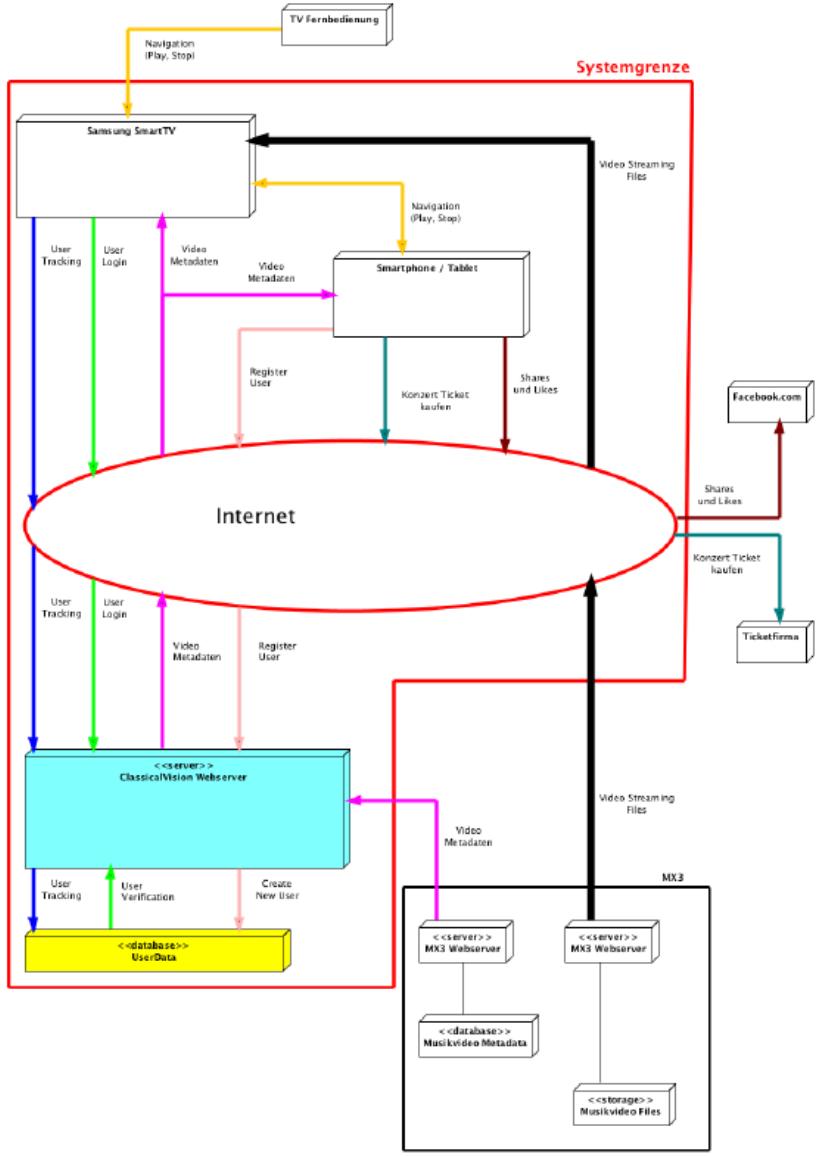
Second Iteration of the System Idea

Levy of Execution (*)

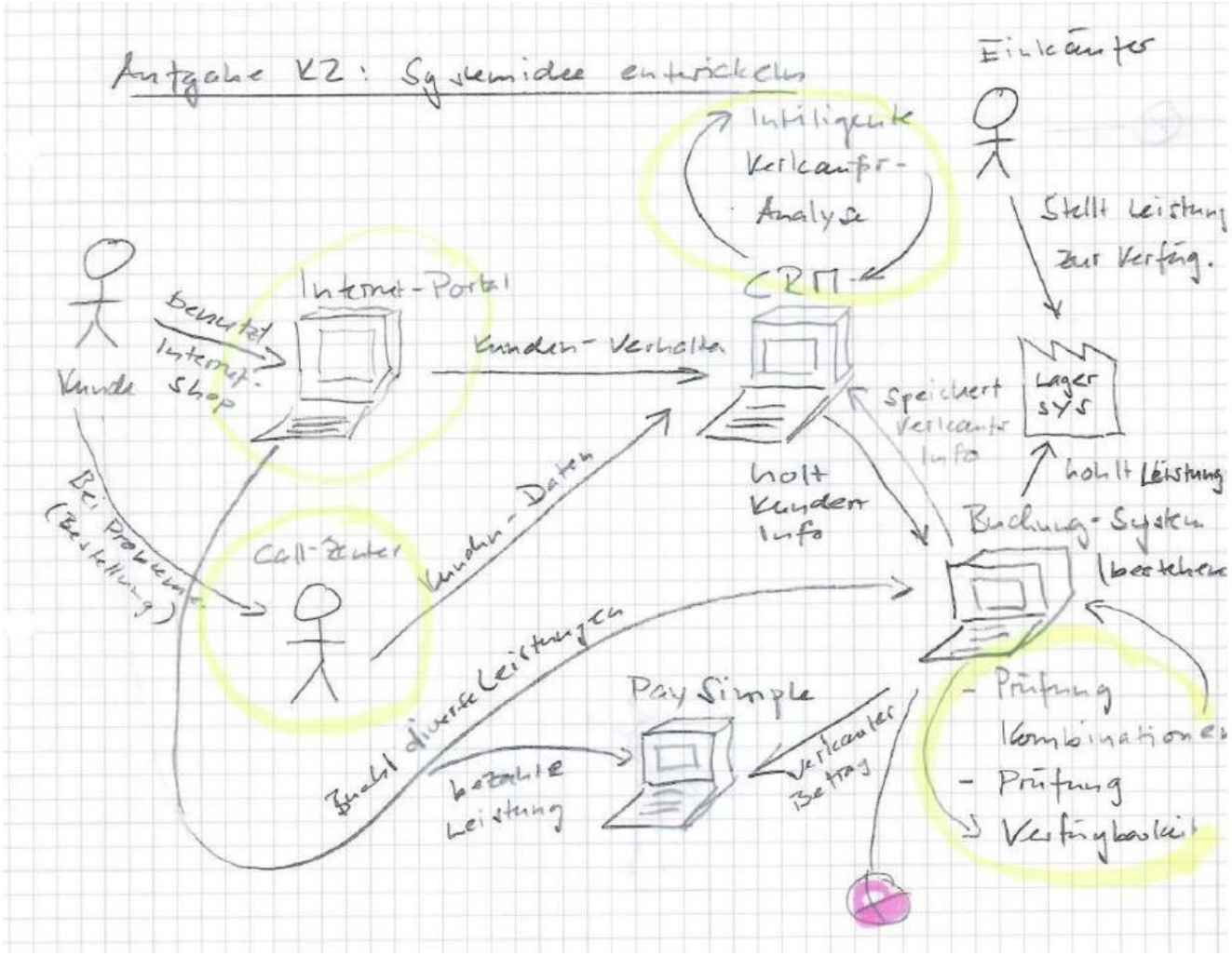


(*) Zwangsvollstreckung

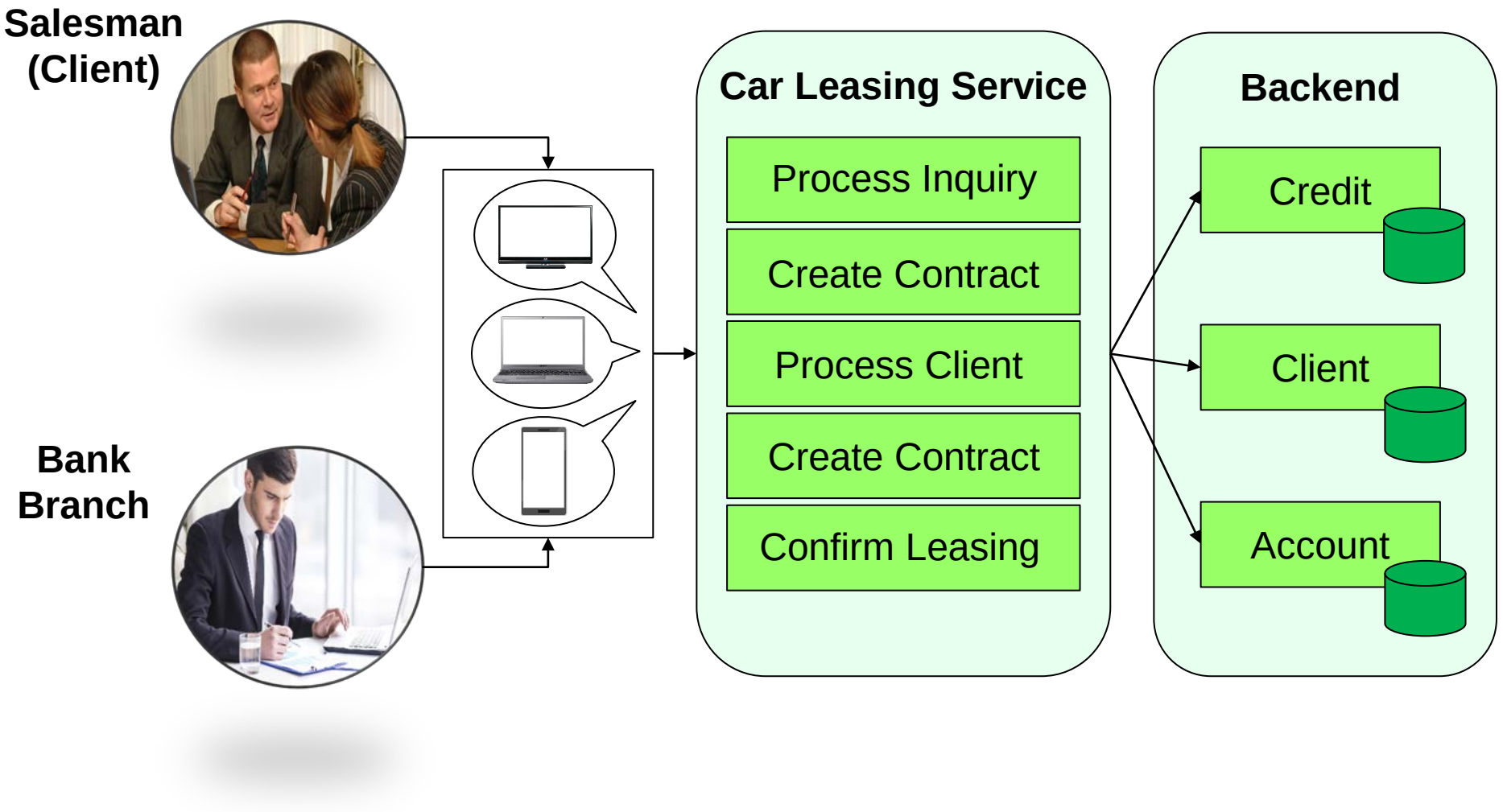
System Idea - Antipattern: Being Overwhelmed by Complexity



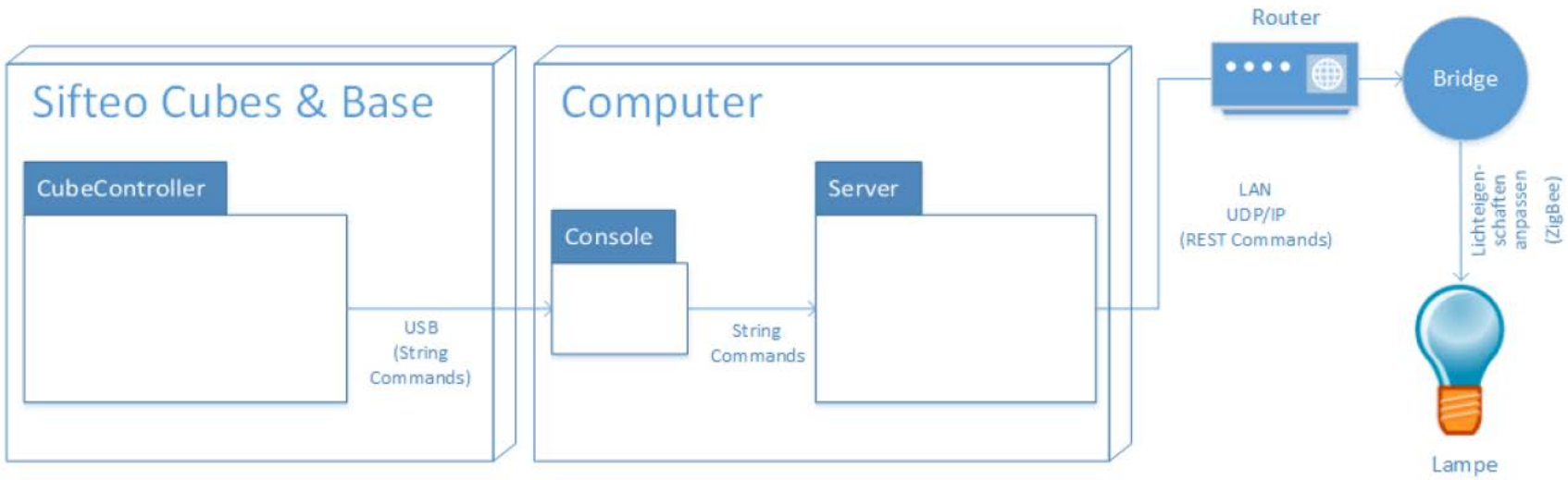
System Idea - Antipattern: Wild Sketching



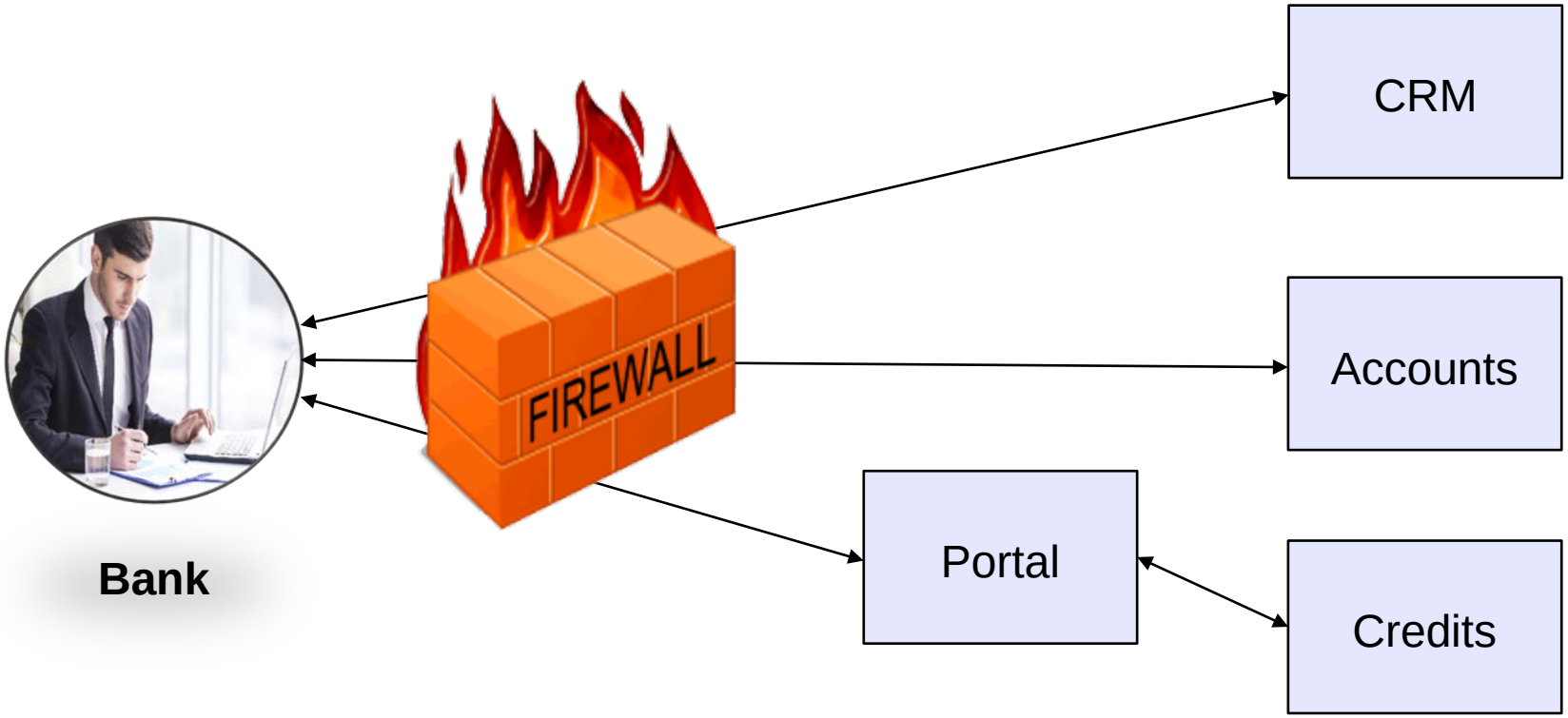
System Idea - Antipattern: Undetected Duplications



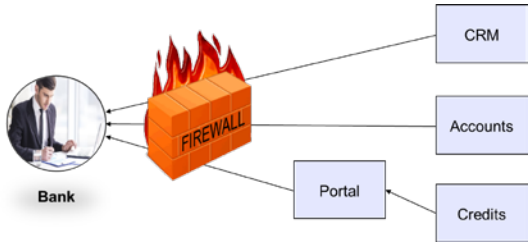
System Idea - Antipattern: Unclear Level of Abstraction

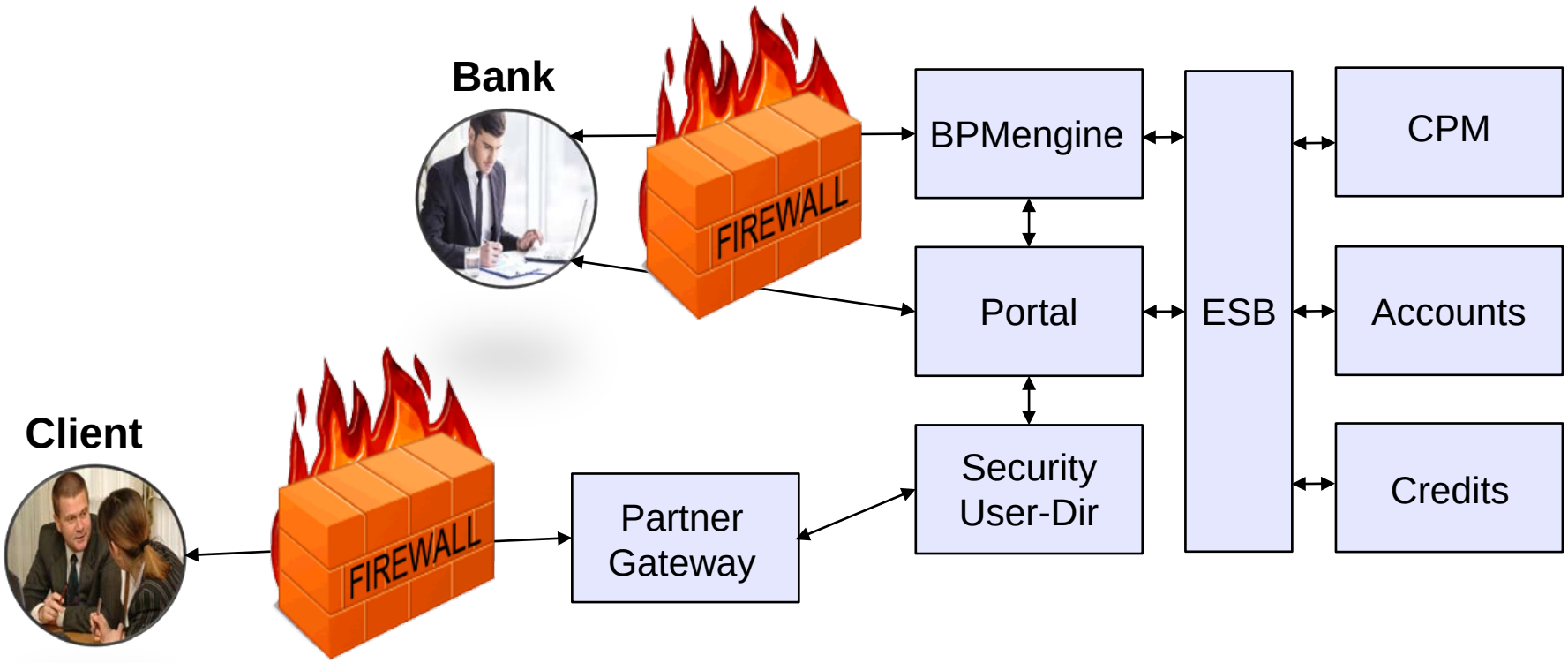


AS-IS View

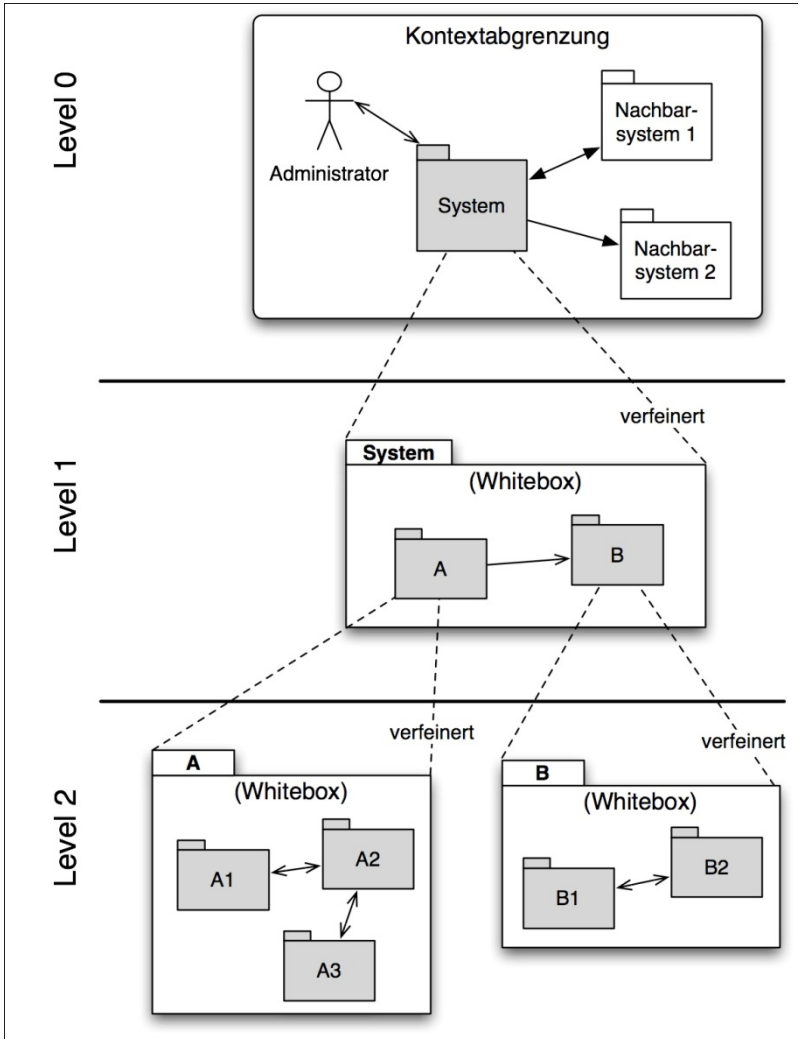


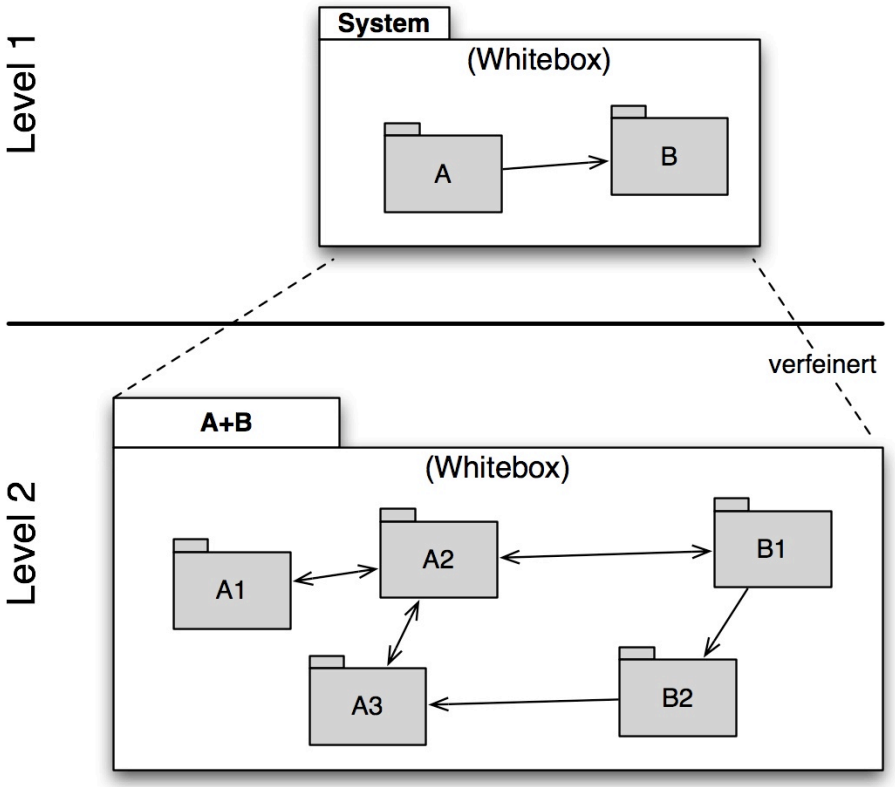
TO-BE View


CRM
Accounts
Portal
Credits
FIREWALL
Bank

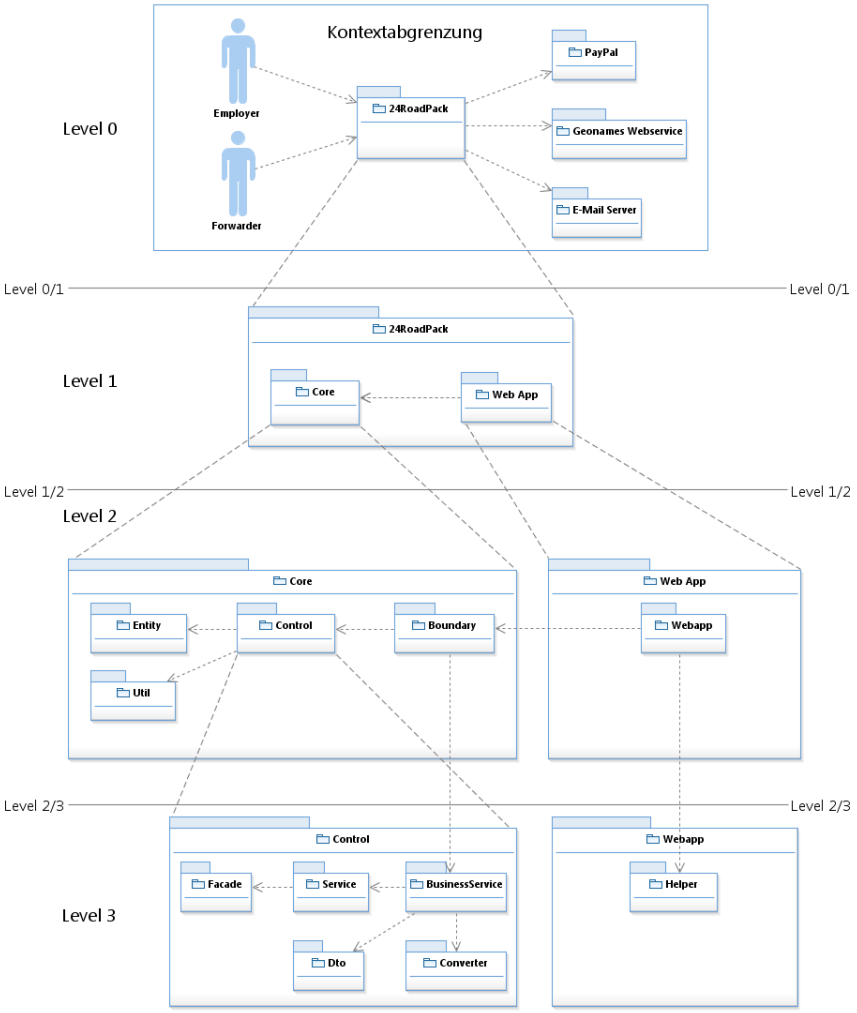

Bank
FIREWALL
BPMengine
CPM
Portal
ESB
Accounts
Client
FIREWALL
Partner Gateway
Security User-Dir
Credits

Component Views at Different Levels of Abstraction



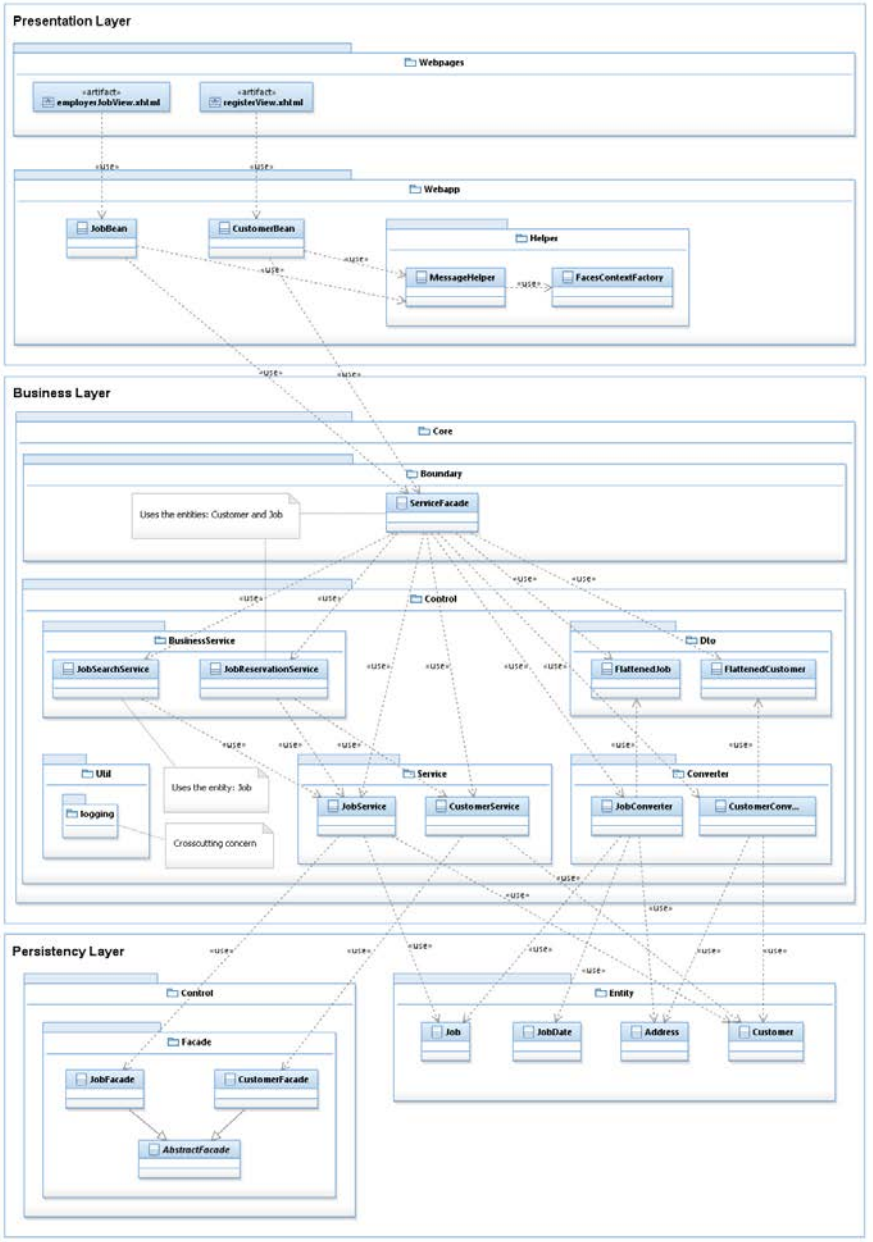


Component View - Antipattern: Lifting the Code to the Model Level



Component View - Antipattern: Lifting the Code to the Model Level

Not really clear any more, but you can see weaknesses in the component structure ….



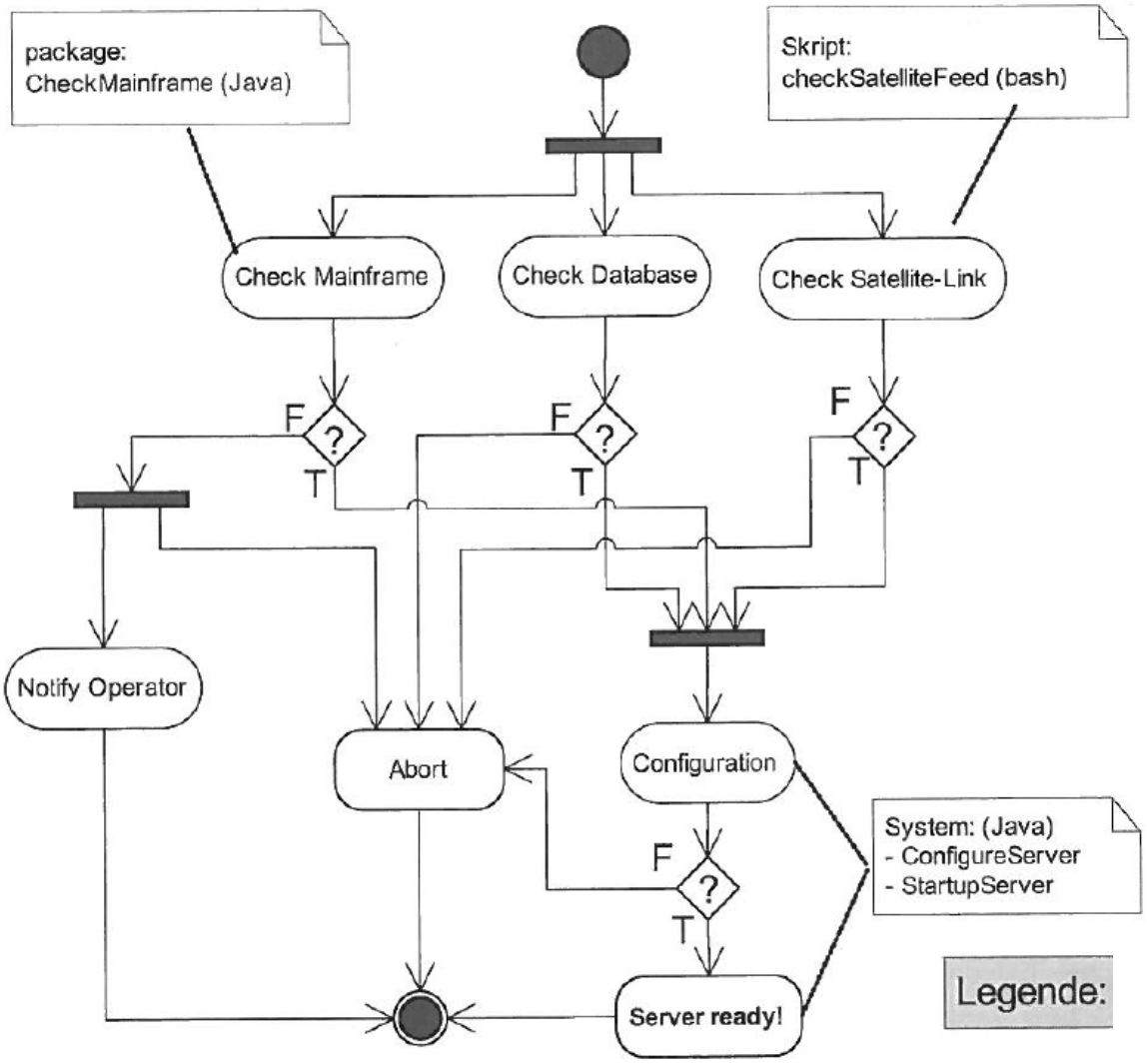
Runtime View

- **How do processes run in the system from a user's point of view?**

- Dynamic structures and behaviors in the system
  - Which components of the system exist at runtime?
  - How do they interact?

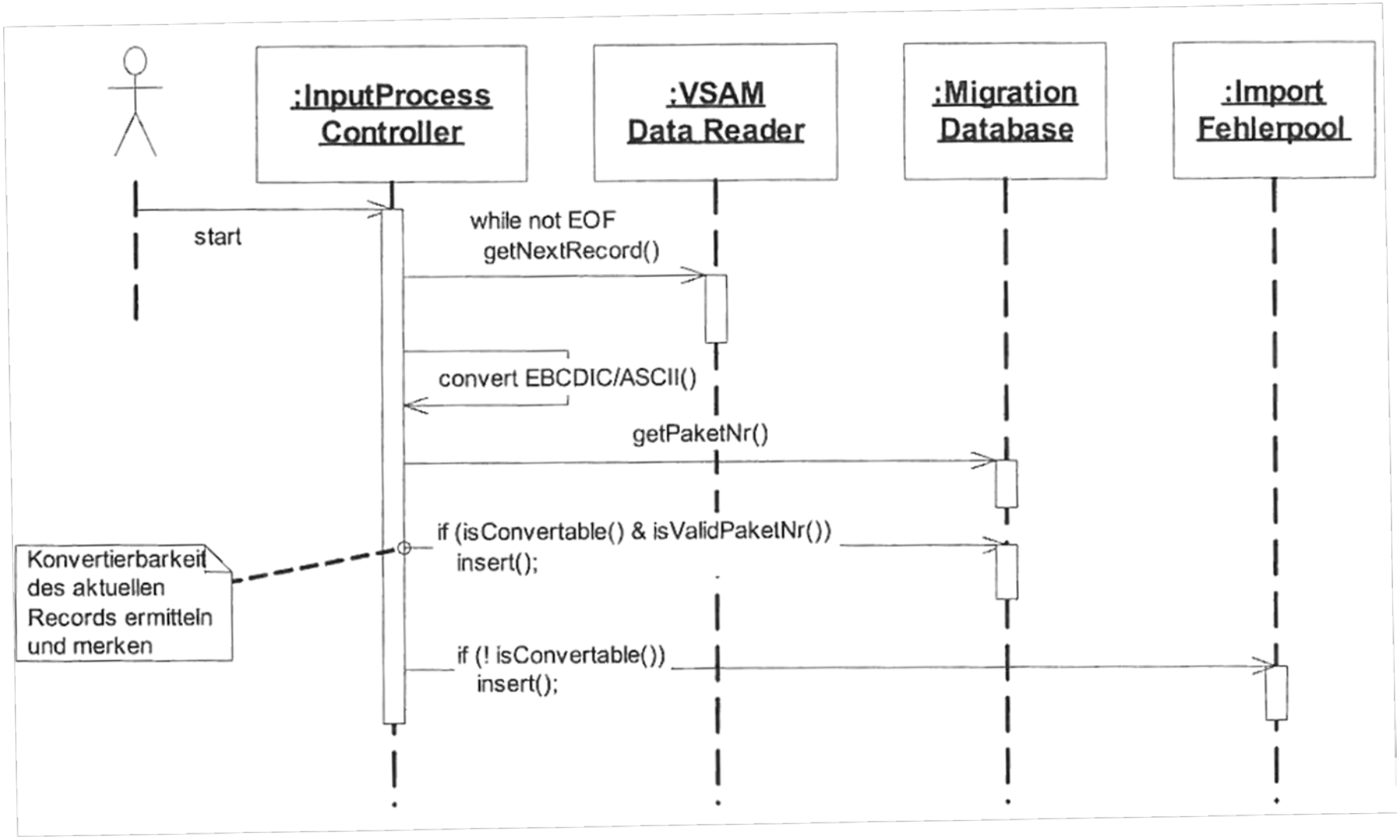
- But also :
  - Where and how do users interact with the system?
  - Where are security risks?
  - How are system qualities such as scalability or performance achieved?

Example of a Runtime View using a UML Activity Diagram



With some effort, I also see this in the code!
Where are the architecture-relevant answers to our questions?

Example of a Runtime View as a UML Sequence Diagram



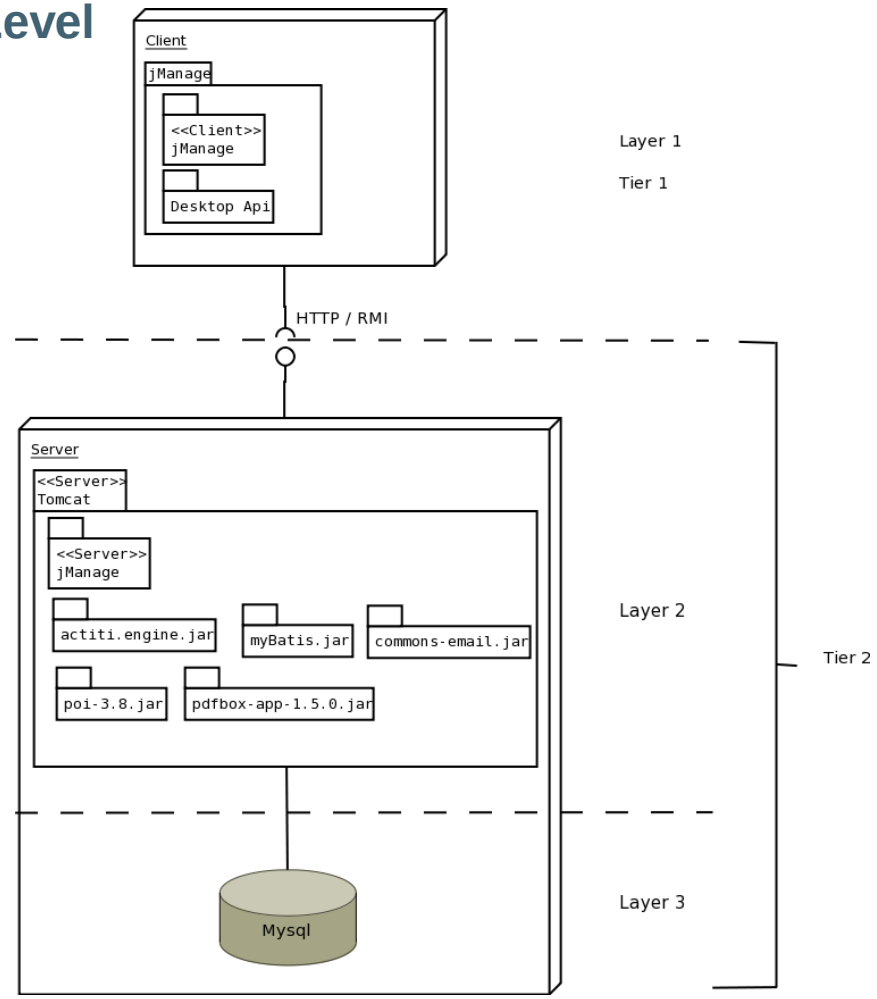
Distribution View - Operational Model - Operational View Infrastructure View

- **In which environment does the system run?**
  - System from the operator's point of view

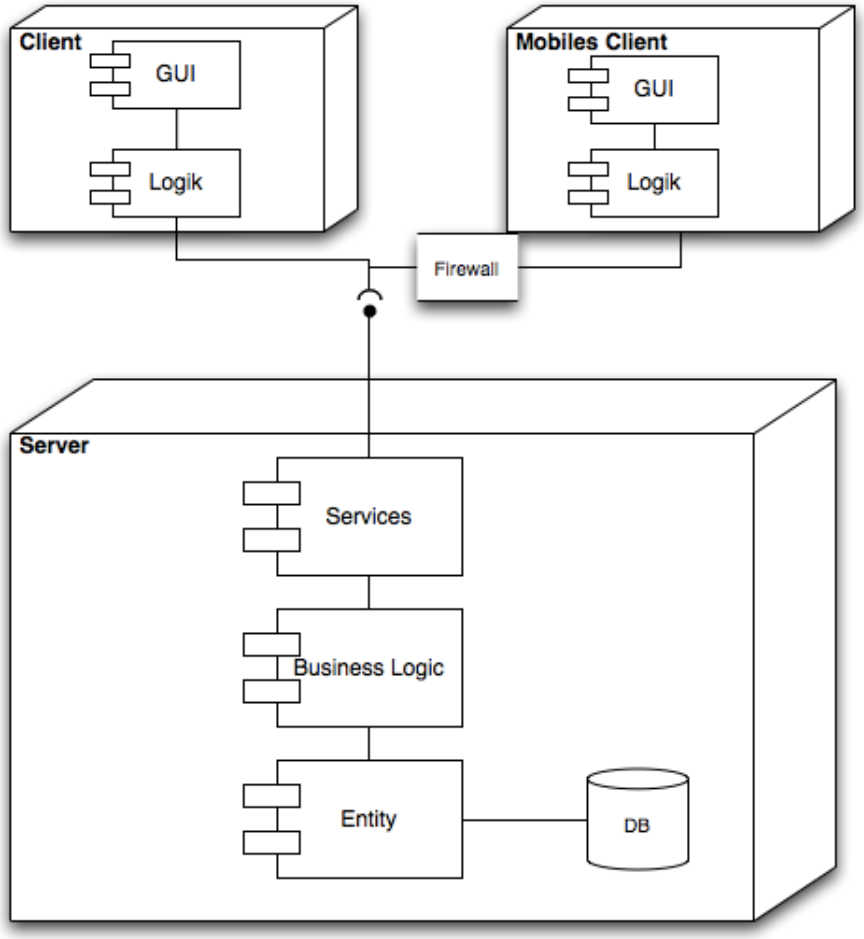
- Distribution of the software system to the physical systems
  - Hardware components on which the software runs
  - Computers, processors, network topologies and protocols
- But also:
  - How are infrastructure requirements satisfied?
  - How expensive will the system be when fully expanded?
  - How do we operate and maintain it?
  - Can it evolve to foreseeable future requirements?

2 Distribution Views of the same System

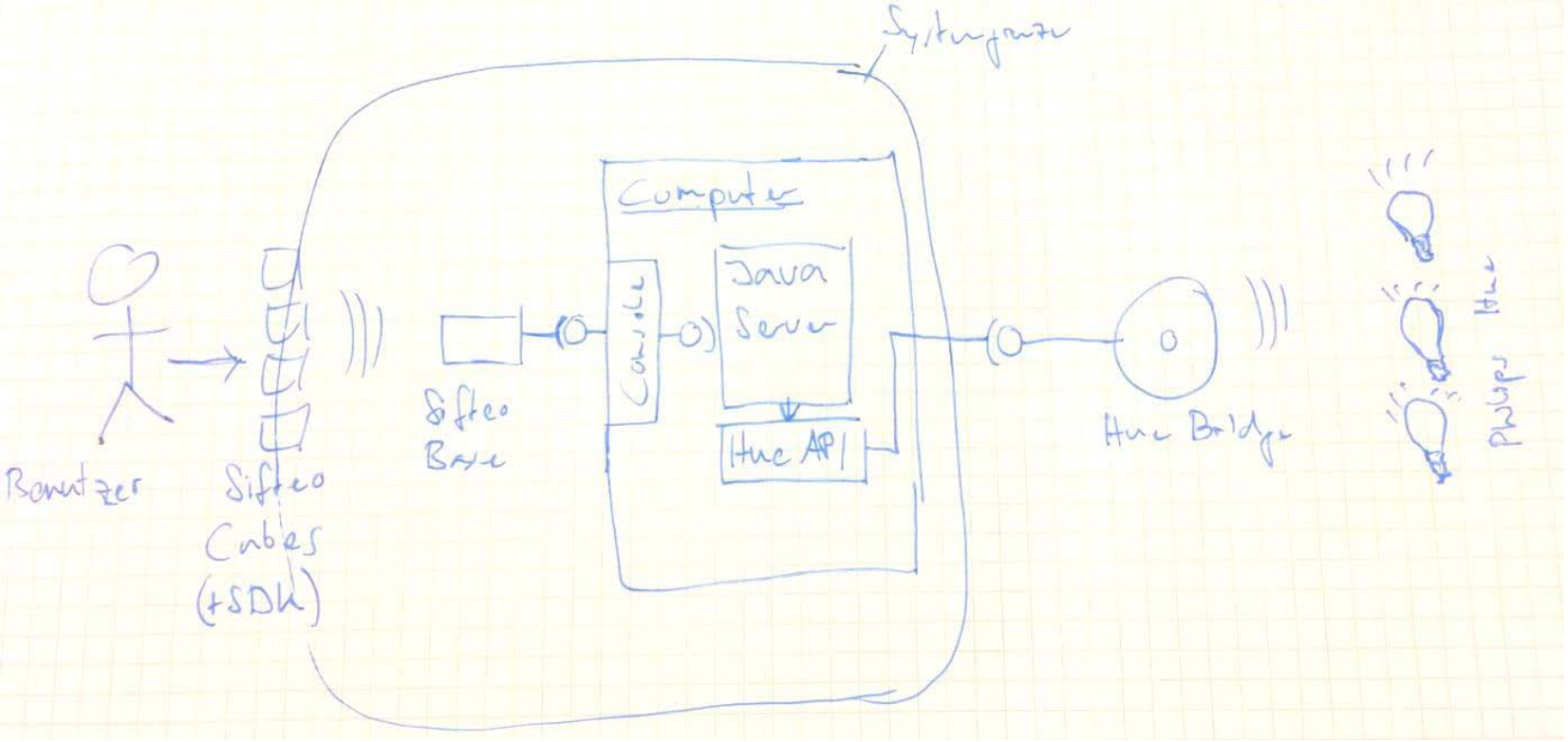
Anti-Pattern: Lifting Code to the Model Level



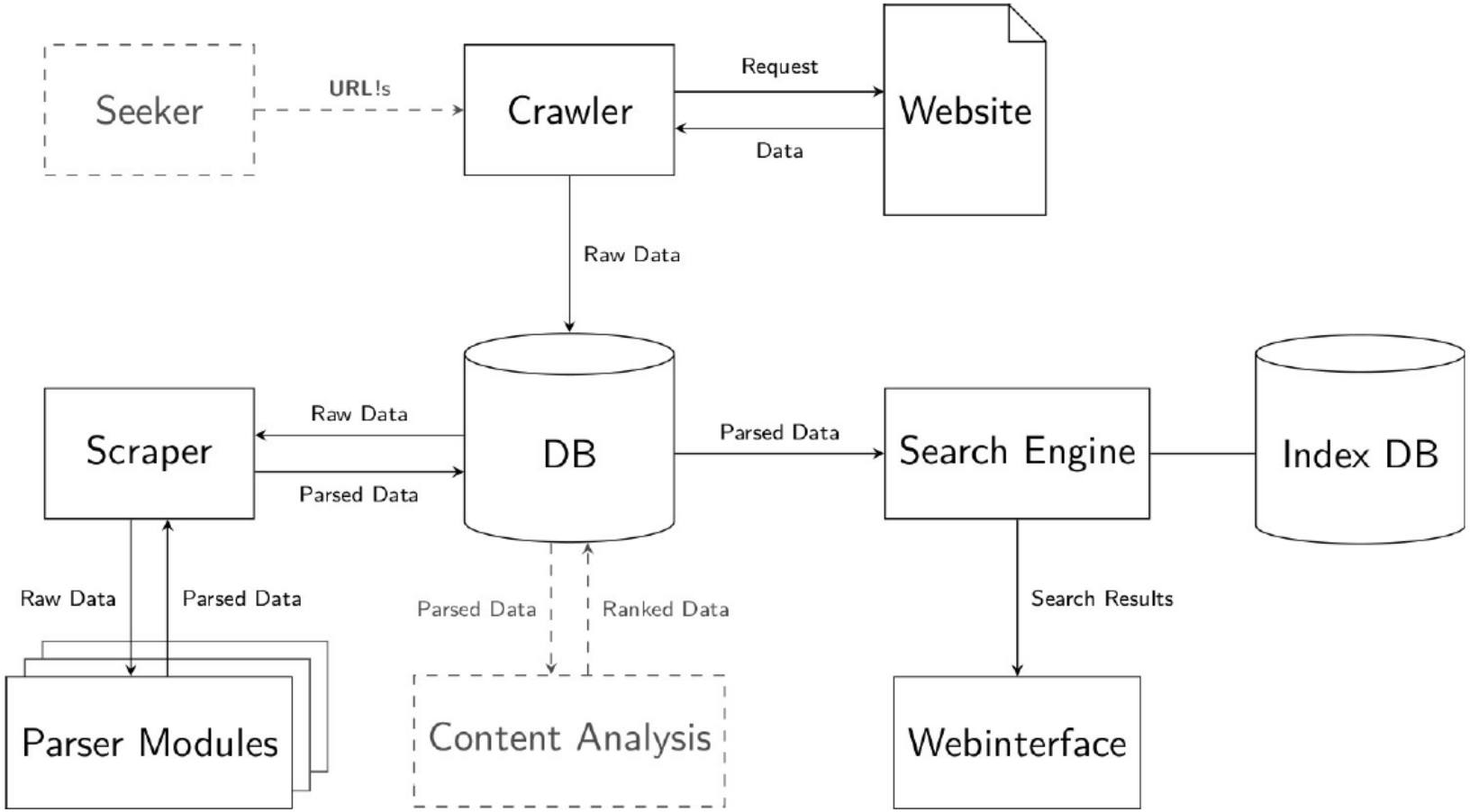
Antipattern: Abstract Generic Naming

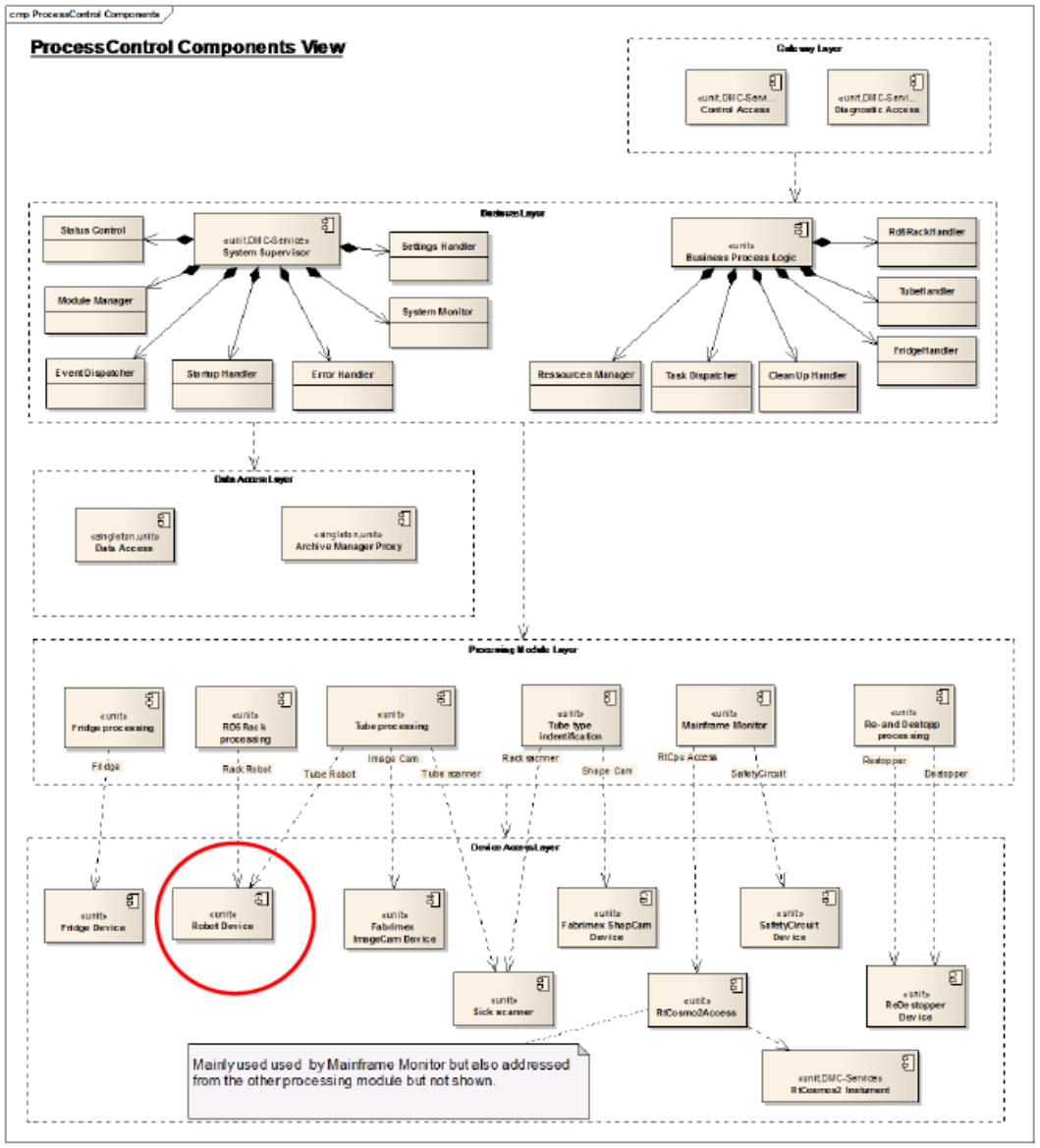


Antipattern: Hidden Type of View



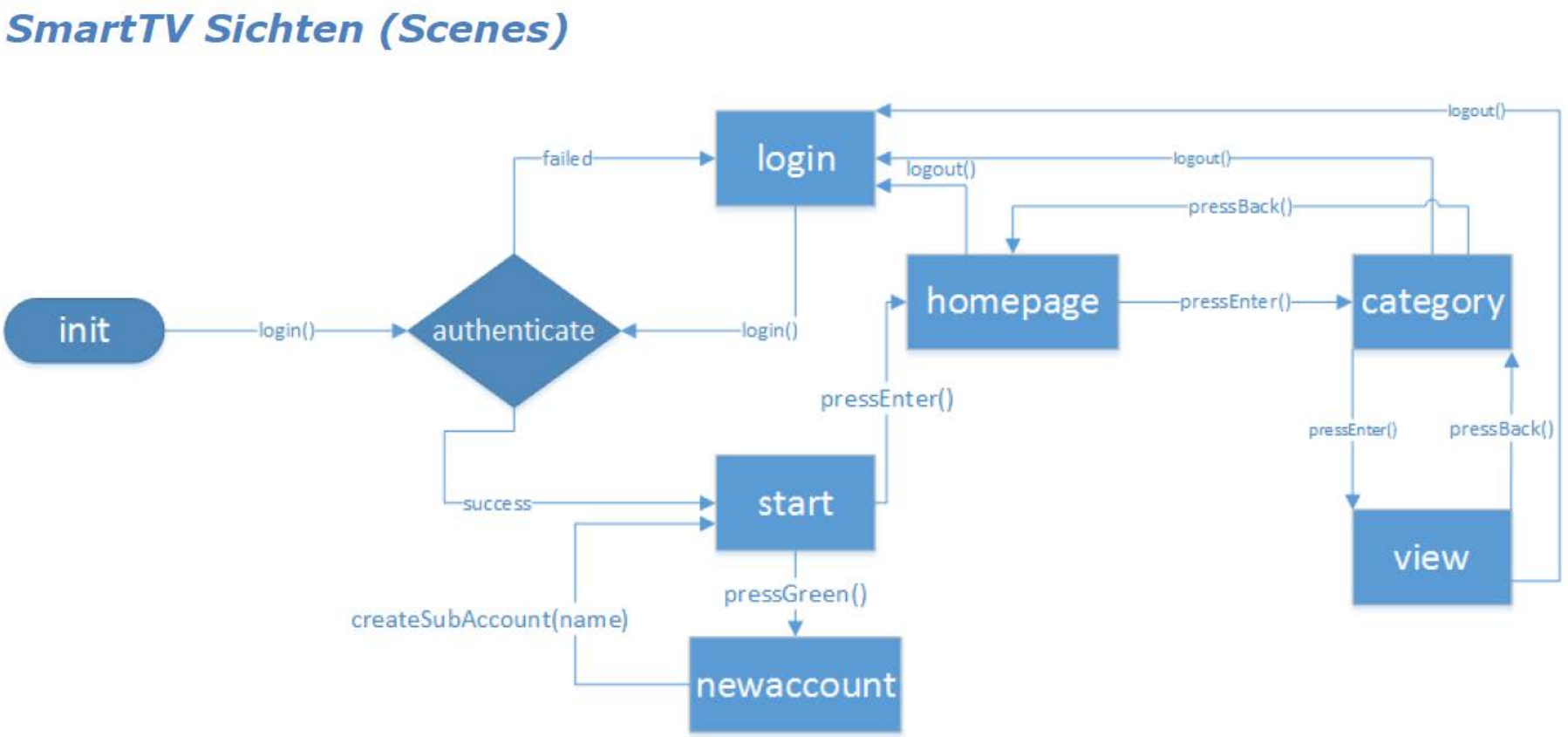
And this?


Seeker
URL!s
Crawler
Request
Data
Website
Raw Data
Scraper
Raw Data
Parsed Data
DB
Parsed Data
Search Engine
Index DB
Raw Data
Parsed Data
Parsed Data
Ranked Data
Search Results
Parser Modules
Content Analysis
Webinterface

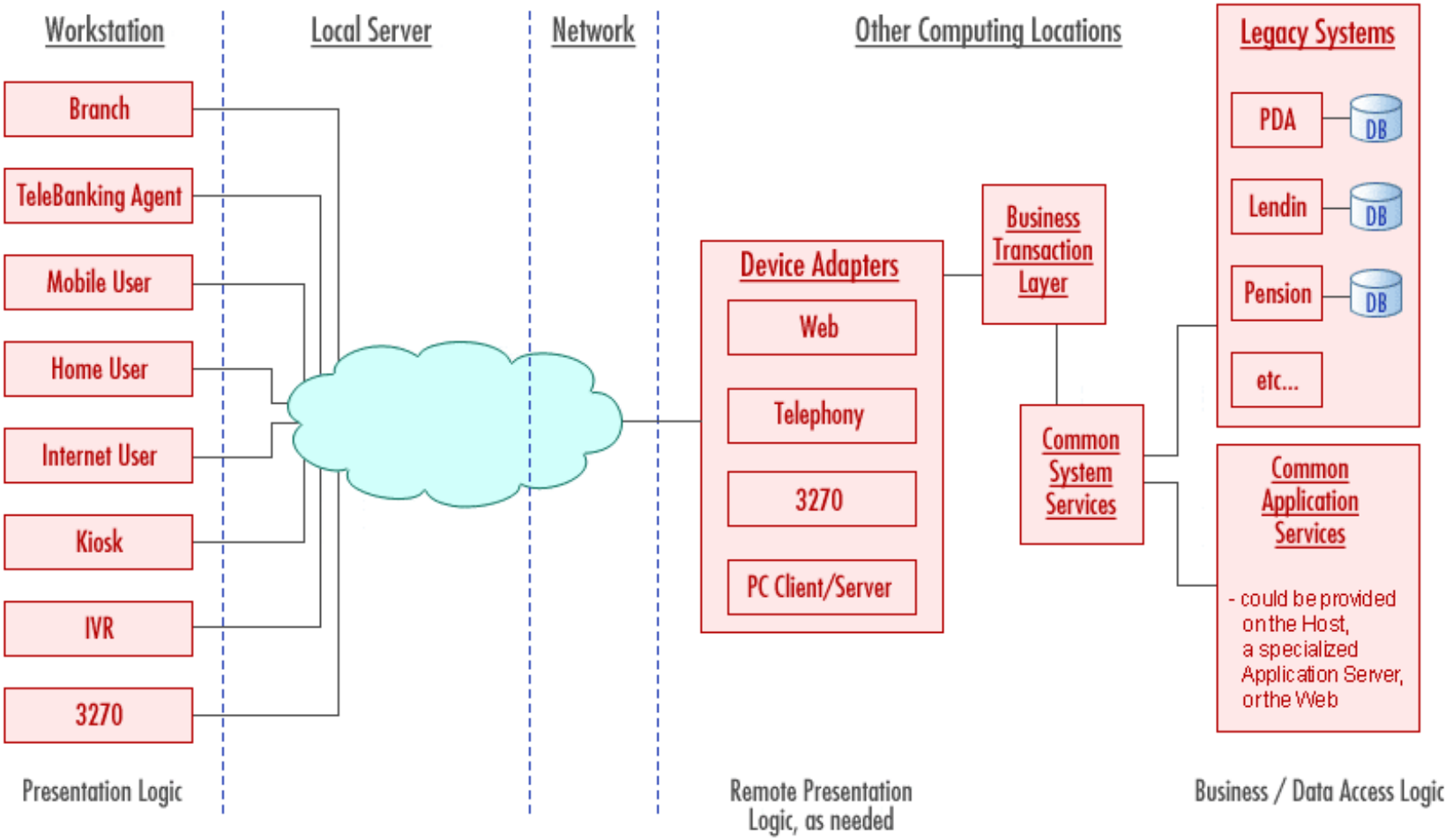


Is this communicated clearly and comprehensibly?

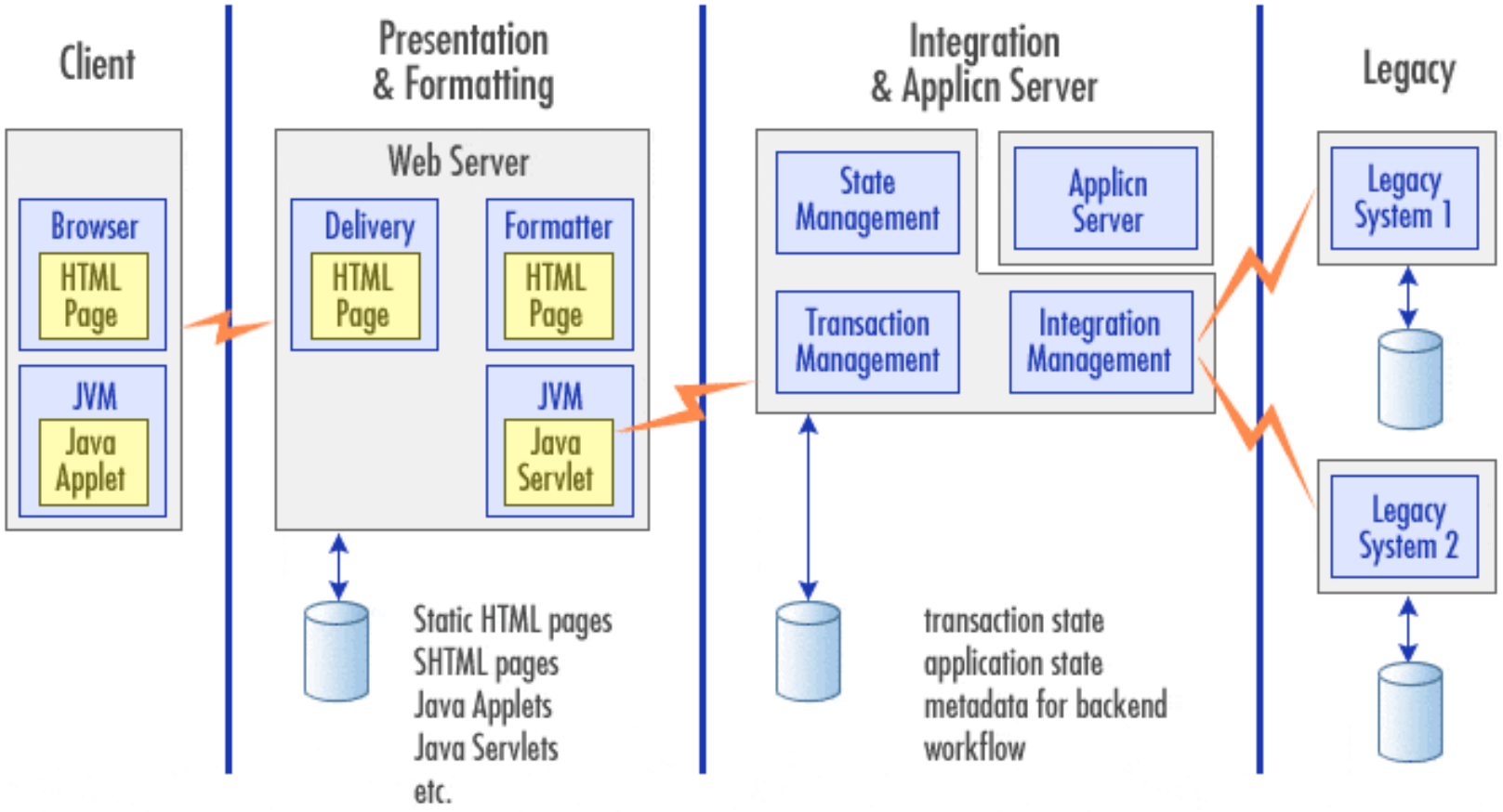
What Type of View is this?



Which View is represented by this Diagram?



Which View is represented by this Diagram?



Summary

- Context, Context, Context!!
- Context view is the most underestimated view by novice architects
- Omissions in the system context are the main source of risks
- Spend time in creating a system vision and review it regularly
- Think in alternatives when creating the system idea, the first idea might not be the best
  - Start implementing the most risky parts
- Real projects require additional relevant views, in particular:
  - Security view & Information view
  - Mockups and story boards are common views to document the user experience (GUI)

Working Questions

1. How do we document structures and behaviors of a system?
2. What do we understand by a view and which views do you know?
3. What constitutes a “good” view?
4. How much detail needs to be put into a view?
5. Why do component views often only represent the structure of the code?
6. Why is the operational view one of the most important views alongside the context view?
7. Why is it so difficult to create a good context view?
8. What is a viewpoint?