Fraunhofer IESE

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

SSA 02 –Architecture Documentation

Dr. Pablo Oliveira Antonino
pablo.antonino@iese.fraunhofer.de

TU Kaiserslautern, SS2018
Lecture "Software and System Architecture (SSA)"

Discussion

- **RECAP LAST LECTURE**

- Explain the contents of the last lecture
  - What were the topics?
  - Why do we need it?
  - How does it work?
  - How is it created, used, and/or evolved?

Documentation

Documentation

- **The ultimate goal of documentation is to enable stakeholders to gain knowledge**

- Knowledge is the dynamic capacity that enables a stakeholder to
  - Increase confidence
  - Understand the context
  - Perform a task
  - Solve problems
  - Use and adapt information for a specific purpose

- **However, only information can be documented**

Data, Information, Knowledge

- Data
  - "Data consists of discrete, objective facts about events and entities but nothing about its own importance or relevance; it is raw material for creating information"

[Rus & Lindvall, 2002]

- Information
  - "Information is data that is organized to make it useful for end users who perform tasks and make decisions"

[Rus & Lindvall, 2002]

- Knowledge
  - "Knowledge is the result of a learning process and can be seen as a function of (task-related) information, experience, skills and attitude at a given moment in time"

[Weggeman, 1999]

Real Life: “I Can Always Explain How the System…”

[Source: dreamstime.com]

Real Life: Architecture Documents

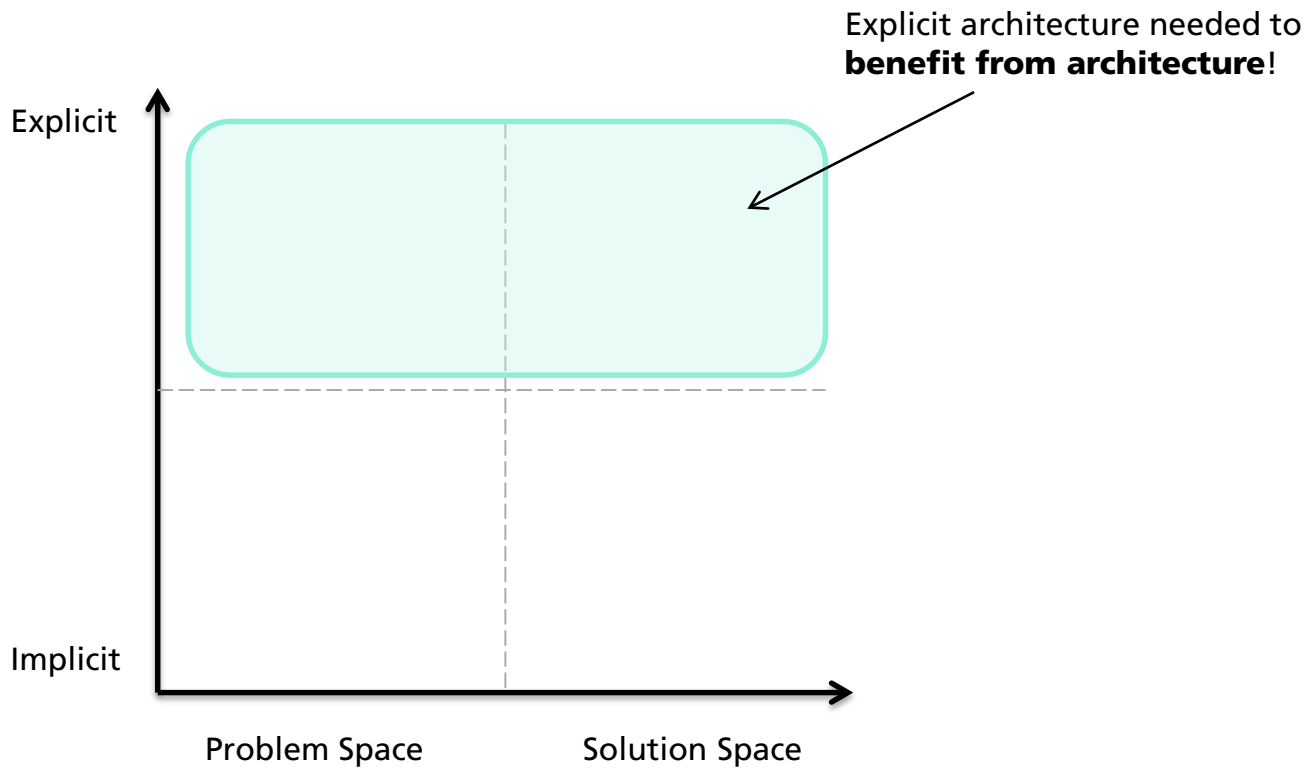
An Ideal Architecture Documentation...

... describes what the code itself does not!

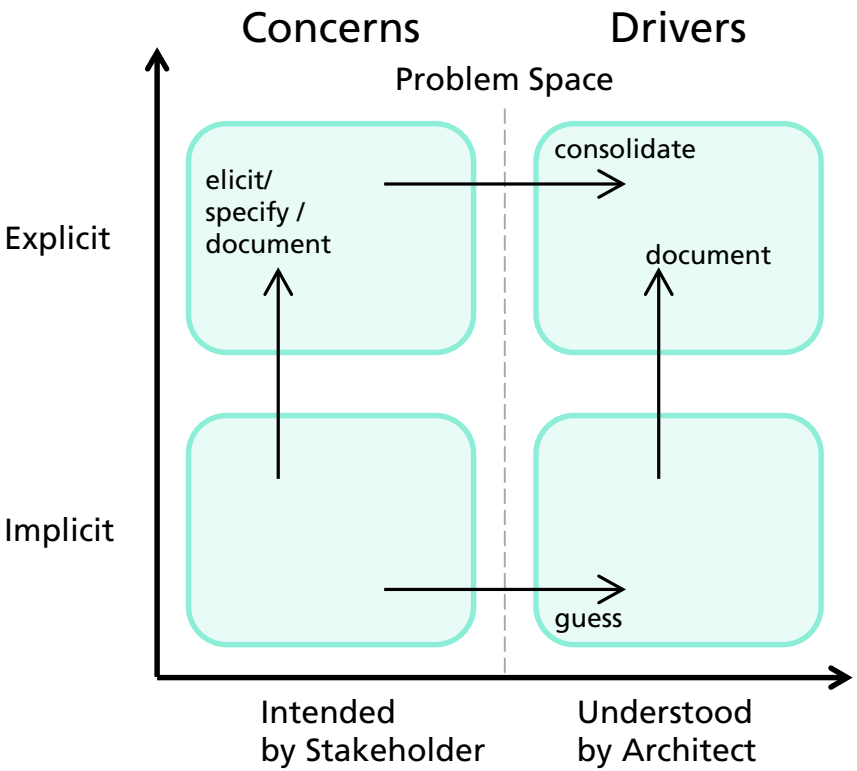
e.g.

- What are the design decisions?
- What is the rationale for the decisions?
- What are the discarded alternatives? Why?
- …

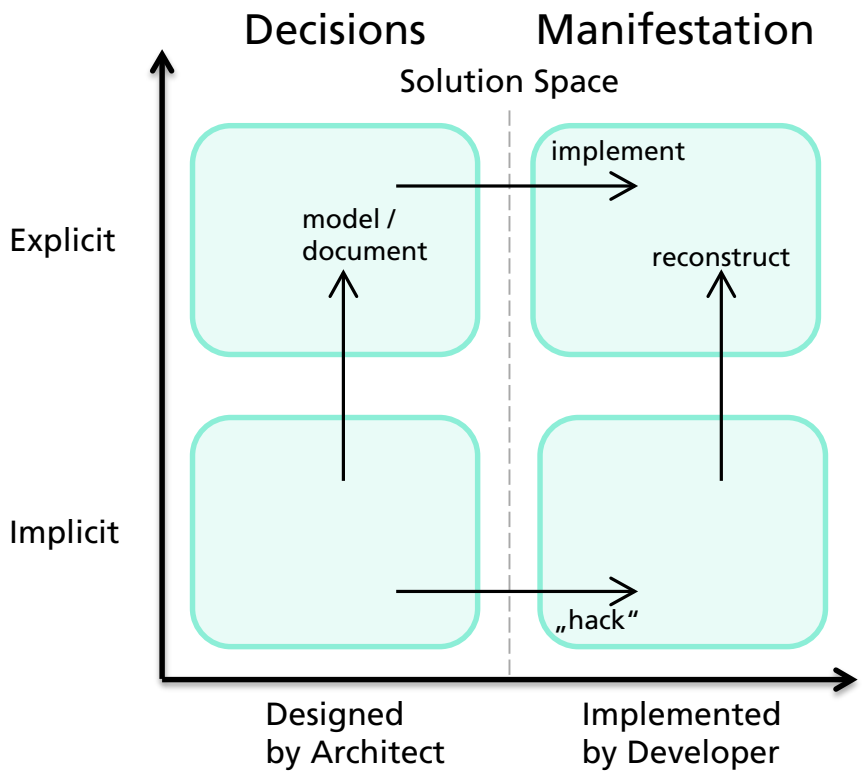
What do We Need in Terms of Architecture?



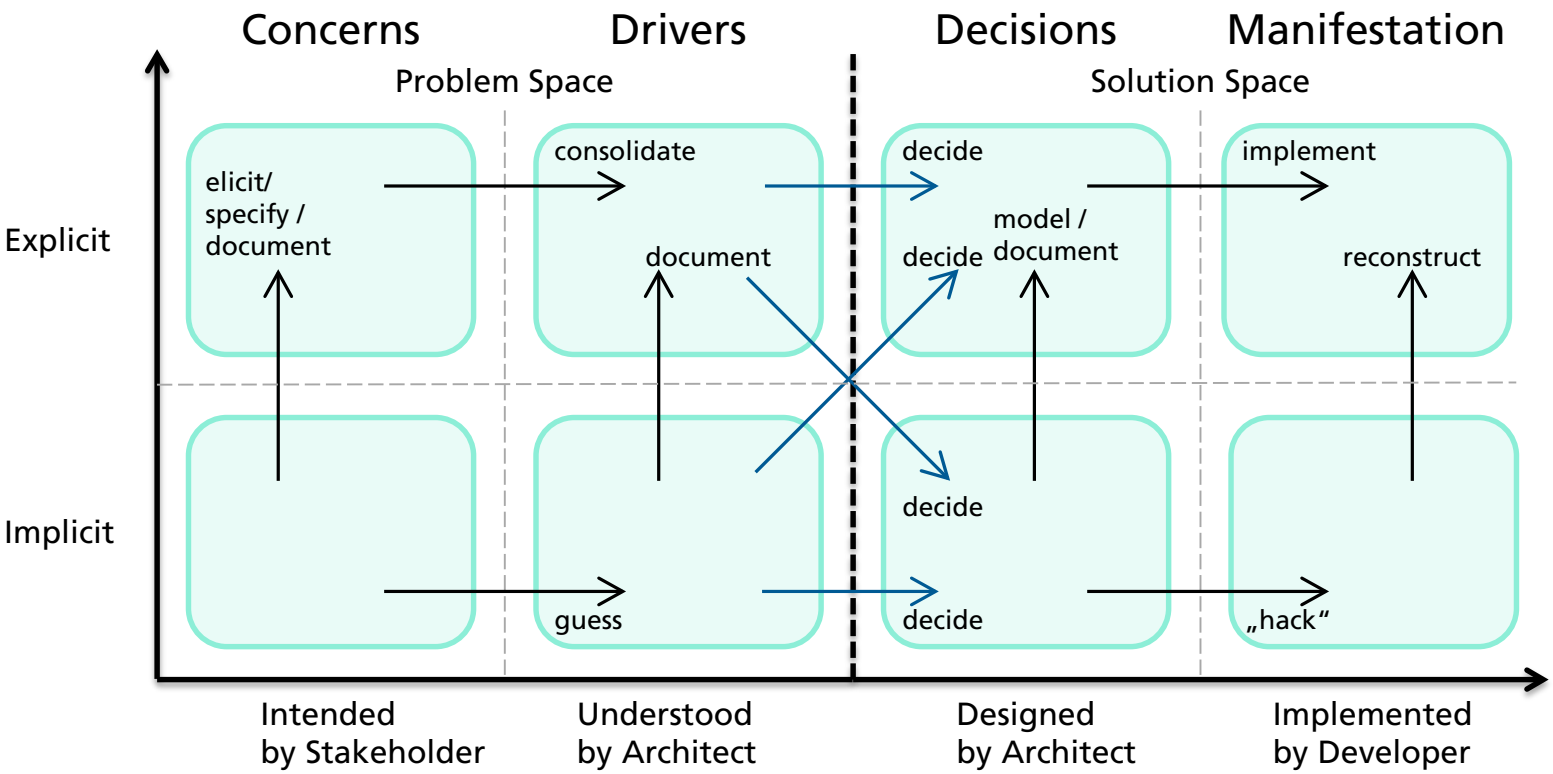
Explicit vs. Implicit Architecture Problem Space



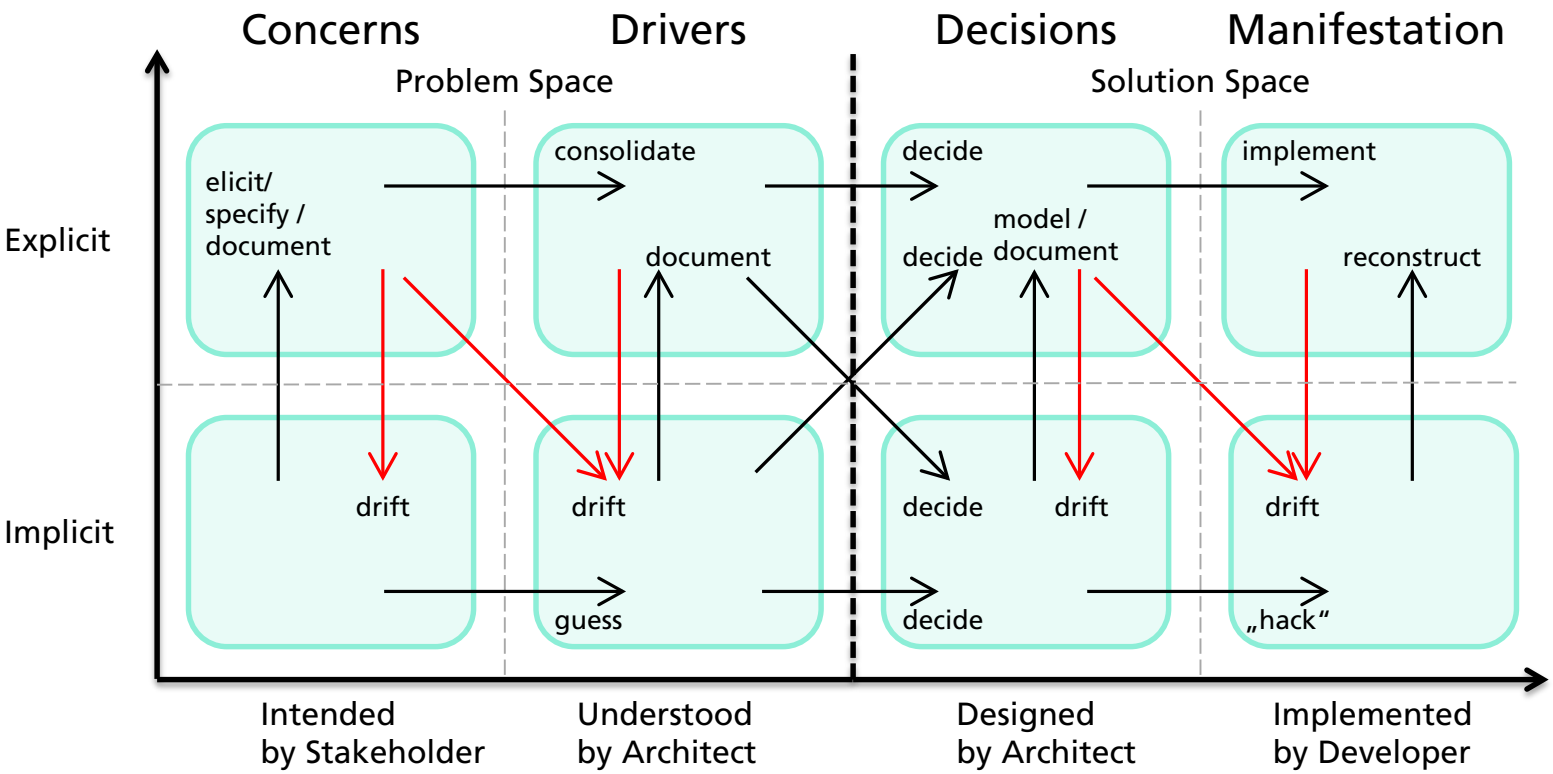
Explicit vs. Implicit Architecture Solution Space



Explicit vs. Implicit Architecture
Problem Space vs. Solution Space



Evolution and Drift



Software Architecture Document – Example

RESCUER

2.2. Conceptual Architecture

The main goal of the RESCUER system is to improve emergency management in the field. **Figure 2** shows the conceptual architecture taking this goal into consideration. We have the emergency situation comprising of cause of emergency (fire/ explosion/ gas-leak etc.), injuries, damages of property, and so on. The *Context Sensor* component is responsible for sensing the environment. The *Information Processing and Decision Making* component receives the sensed data, analyses them and visualizes them efficiently in the command and control centre. The command and control centre decides on some actions to improve the situation. Messages are sent back to the *Effector* component which does something physically on the emergency environment to improve the situation. The components are described in detail below.

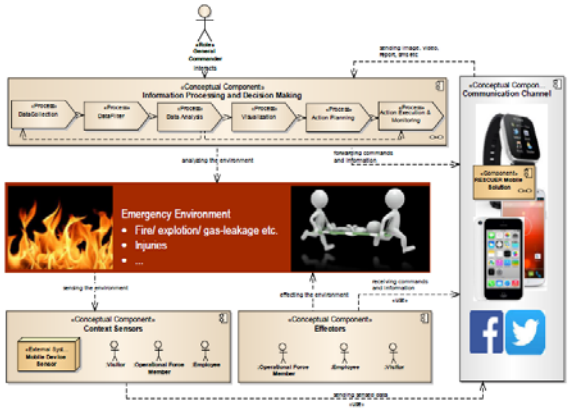


Figure 2: RESCUER conceptual architecture

- *Context Sensors*: This conceptual component represents the stakeholders and systems that do sensing of the environment. Visitors in large scale events, employees in the industrial parks, operational forces' members are the people who are truly on the spot during the incident. They are the people who can observe the situation and report. Mobile device sensors can also sense the location and movement of the people on the spot.
- *Effectors*: This conceptual component represents the stakeholders and systems that do something on the spot to handle the emergency situation in practice. The sensing stakeholders

RESCUER

A user of RESCUER ERT interacts with it web-interface and enters arbitrary data in the input fields. The web application is robust and does not crash.

- ***ASR.ROBUSTNESS.03: Robustness against unstable network connections***
  A user of the RESCUER mobile solution interacts with his/her RESCUER app and experiences an unstable network connection with low bandwidth. The app is robust and does not crash. No data is lost and eventually is sent to the RESCUER backend.

- ***ASR.ROBUSTNESS.03: Robustness against no network connections***
  A user of the RESCUER mobile solution interacts with his/her RESCUER app and experiences a network outage. The app is robust and does not crash. Basic emergency reports (sensor data) can still be sent through Ad-hoc p2p network. Any other data is not lost and eventually is sent to the RESCUER backend, once the network connection comes back.

- ***ASR.ROBUSTNESS.05: Robustness against crashes of mobile application***
  A user of the RESCUER mobile solution interacts with his/her RESCUER app and the app crashes. The app is able to restart its operation from where it crashed. It means that the app persists all administrative messages from the server, all partial reports and profile information. Whenever the app starts, it starts working based on the last saved administrative messages.

3.7.3. Scalability

Assumptions:

The number of industrial parks or large-scale events covered by one installation of the RESCUER backend can increase or decrease based on the region.

Requirements:

- ***ASR.SCALABILITY.01: Initial load during first evaluation***
  The RESCUER backend is running in an initial version for 100-200 test users in the first evaluation. It is covering either one event or one industrial park scenario.

- ***ASR.SCALABILITY.02: Scaling for large number of apps***
  The number of users of the RESCUER app or ERT can increase. The backend has to scale in a way that it does not need to compromise its performance. In addition to the increment in number of users, the multimedia data (image and video) can also increase.

- ***ASR.SCALABILITY.03: Scaling over large number of events or industrial parks***
  The RESCUER backend is intended to be evaluated in one event or one industrial park in Europe or in Brazil. Later, the solution is intended to be offered in multiple events. The

Fraunhofer IESE

Software Architecture Document – Example

RESCUER

4. Key Architectural Concepts

4.1. Context Delineation

Figure 5 shows the RESCUER system and the external systems and stakeholders around it.

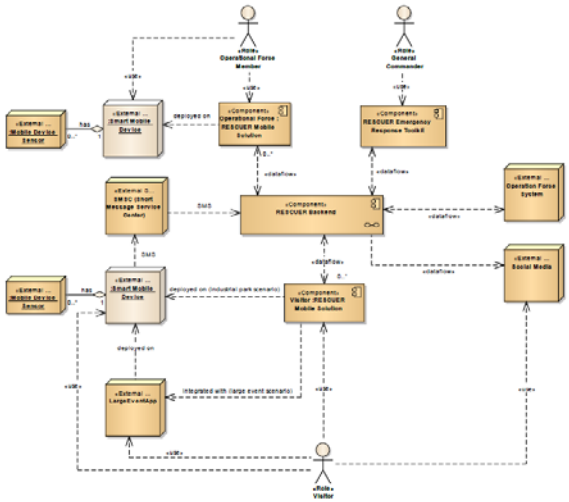


Figure 5 : Context delineation

4.2. Internal Structure

The overall RESCUER system is divided into several layers. Figure 6 shows the layers of the RESCUER system. Figure 7 shows the components inside the layers.

- *Mobile Application Layer*: This layer is responsible for collecting sensor data, user interaction data, multimedia data (image, video), and unstructured text report from the

32

RESCUER

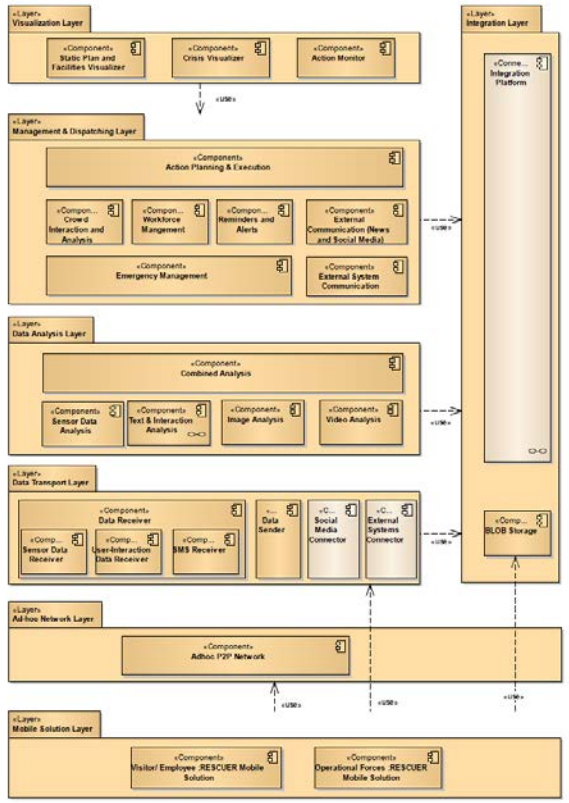


Figure 7 : Layers including functional components

34

Fraunhofer IESE

Software Architecture Document – Example

RESCUER

- Image Analysis Result Topic
- Video Analysis Result Topic
- Combined Analysis Result Topic

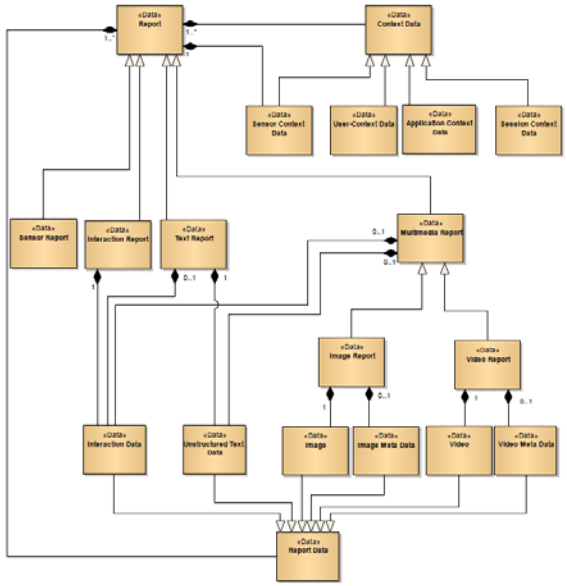


Figure 12 : Report data model

42

RESCUER

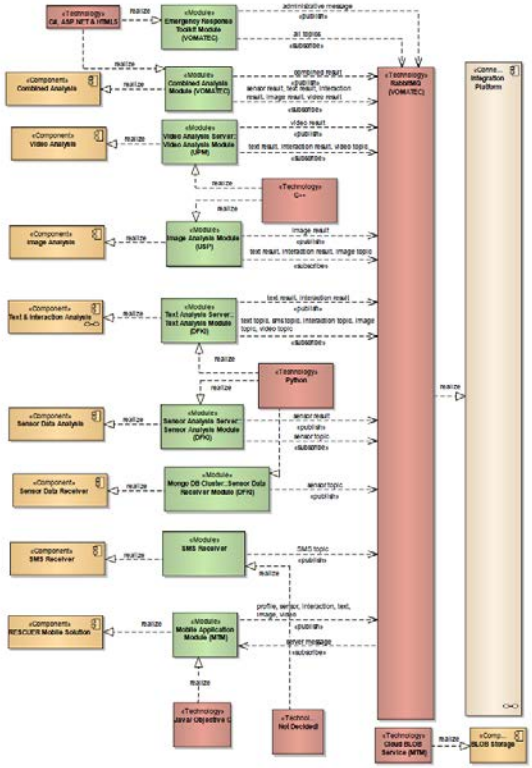


Figure 30 : Development of the Integration Platform

60

Fraunhofer IESE

Software Architecture Document – Example

RESCUER

Table 3 : Development task allocation among partners

Partner	Modules to build
MTM	*Mobile Solutions (MS), Cloud BLOB Service (BLOB)*
DFKI	*Ad-hoc P2P Network (ADHOC), Sensor Data Recorder (SDR), Sensor Data Receiver (SEN_RECV), Sensor Data Analysis (SDA), SMS Data Receiver (SMS_RECV), Text Analysis (TA)*
VOMATEC	*Emergency Response Toolkit (ERT), Combined Analysis (CA), Integration Platform (IP), Social Media Connector (SMC), Legacy System Connector (LSC)*
UPM	*Video Analysis (VA)*
USP	*Image Analysis (IA)*

5.7. Design Decisions

Architecture Significant Requirements	Realisation	Responsible Components
Devtime Requirements		
ASR.DEVTIME.01:Documentation of design and code	Overall system architecture is being documented by this deliverable. Individual project partners have been communicated to document their own design and code.	All
ASR.DEVTIME.02:Distributed development	System is designed in a modular way, and interfaces and data exchanges are made clear among them.	IP
Integration Requirements		
ASR.INTEGRATION.01:New components should be integrated to the RESCUER platform without much effort	Generic integration mechanism publish-subscribe is used which is not bound to any technology and provides asynchronous communication.	All, mostly IP
ASR.INTEGRATION.02:Integration with social media	Social Media Connector in the Data Transport layer is responsible for integration with the social media. This connector makes the overall RESCUER platform not tied to any social media. Realisation concepts will be built in next iterations.	SMC
ASR.INTEGRATION.03:Integration among internal components	Generic integration mechanism publish-subscribe is used which is not bound to any technology and provides asynchronous communication.	IP

69

RESCUER

Glossary

Command and Control Centre Group of people and tools assigned to evaluate risks and make decisions in an emergency and/or crisis in an industrial area or at a large-scale event, usually at the same physical place.

Communication Infrastructure Component of the RESCUER platform whose goal is to support the information flow between the crowd and the command centre.

Data Analysis Solutions Component of the RESCUER platform whose goals are 1) fusing similar data coming from different eyewitnesses, 2) analysing photos, videos, and text messages in order to extract information such as the type of incident, the position and dimensions of the affected area, people density, surrounding sources of further danger, evacuation routes, and possible approach routes for the formal responders.

Emergency Critical situations caused by incidents, natural or man-made, that require measures to be taken immediately to reduce their adverse consequences to life and property.

Emergency Response Toolkit Component of the RESCUER platform whose goals are to: 1) get contextual information about the emergency, 2) ask eyewitnesses and formal responders for relevant missing information, 3) give instructions to eyewitnesses, first responders and potentially affected people or companies, and 4) communicate the emergency to the media, public authorities, and the general public in a context-aware way. The emergency response toolkit is meant to be used primarily by the command and control centre staff.

Mobile Crowdsourcing Solution Component of the RESCUER platform whose goal is to support eyewitnesses and formal responders in providing the command and control centre with information about an emergency situation, taking into account the different smartphones that might be used and how people interact with smartphones under stress.

Abbreviations

RESCUER Reliable and Smart Crowdsourcing Solution for Emergency and Crisis Management

UI User Interface

ASR Architecture Significant Requirements

C&C Command and Control Centre

ERT Emergency Response Toolkit

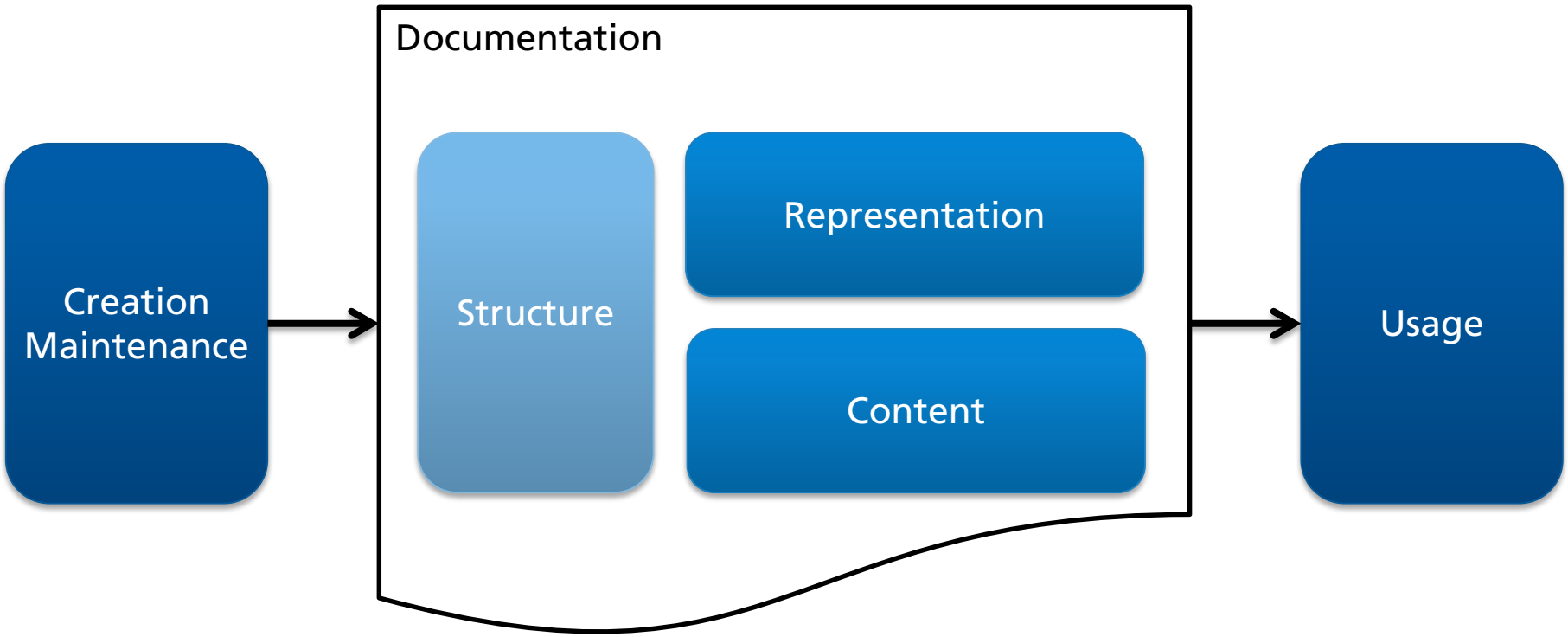
ACES Architecture Centric Engineering Solutions

ADF Architecture Decomposition Framework

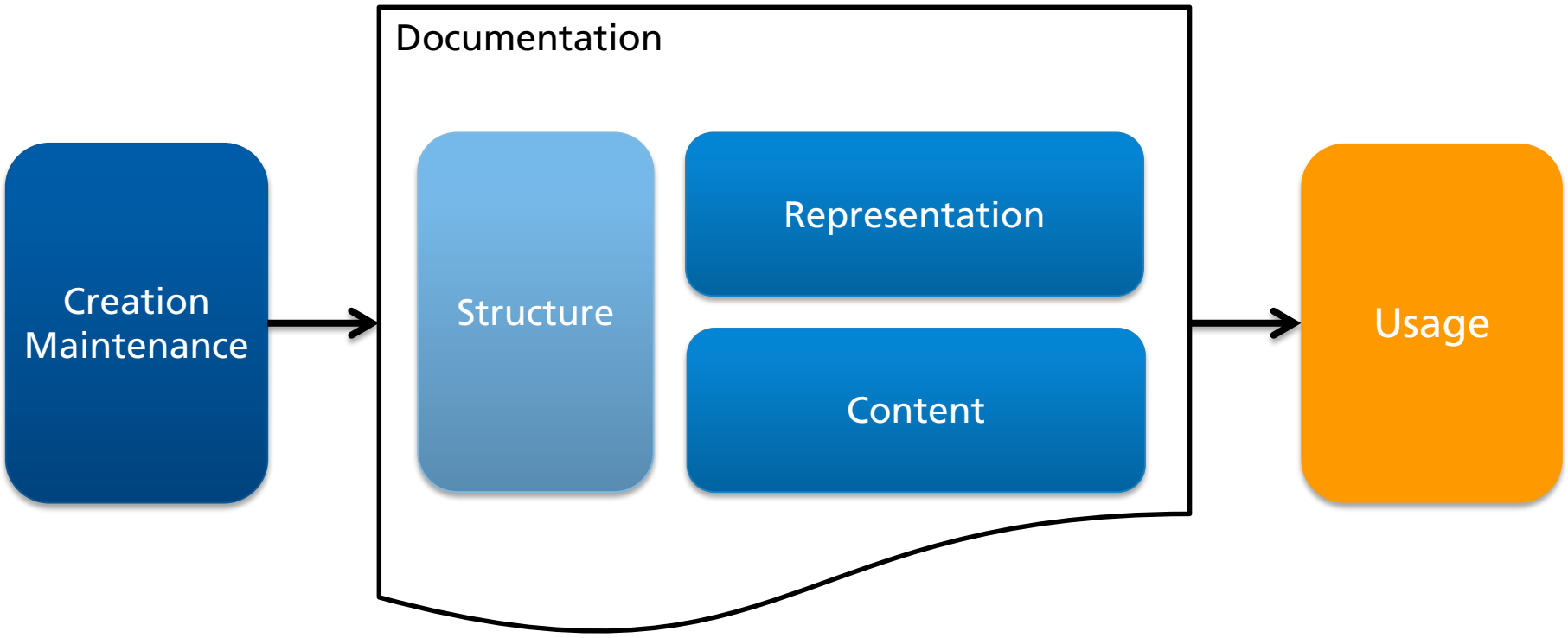
SMSC Short Message Service Centre

78

Architecture Documentation



Architecture Documentation



Architecture documentation has to be adequate for its purposes

Who uses the Architecture Documentation?

Daniel Developer

Arnold Architect

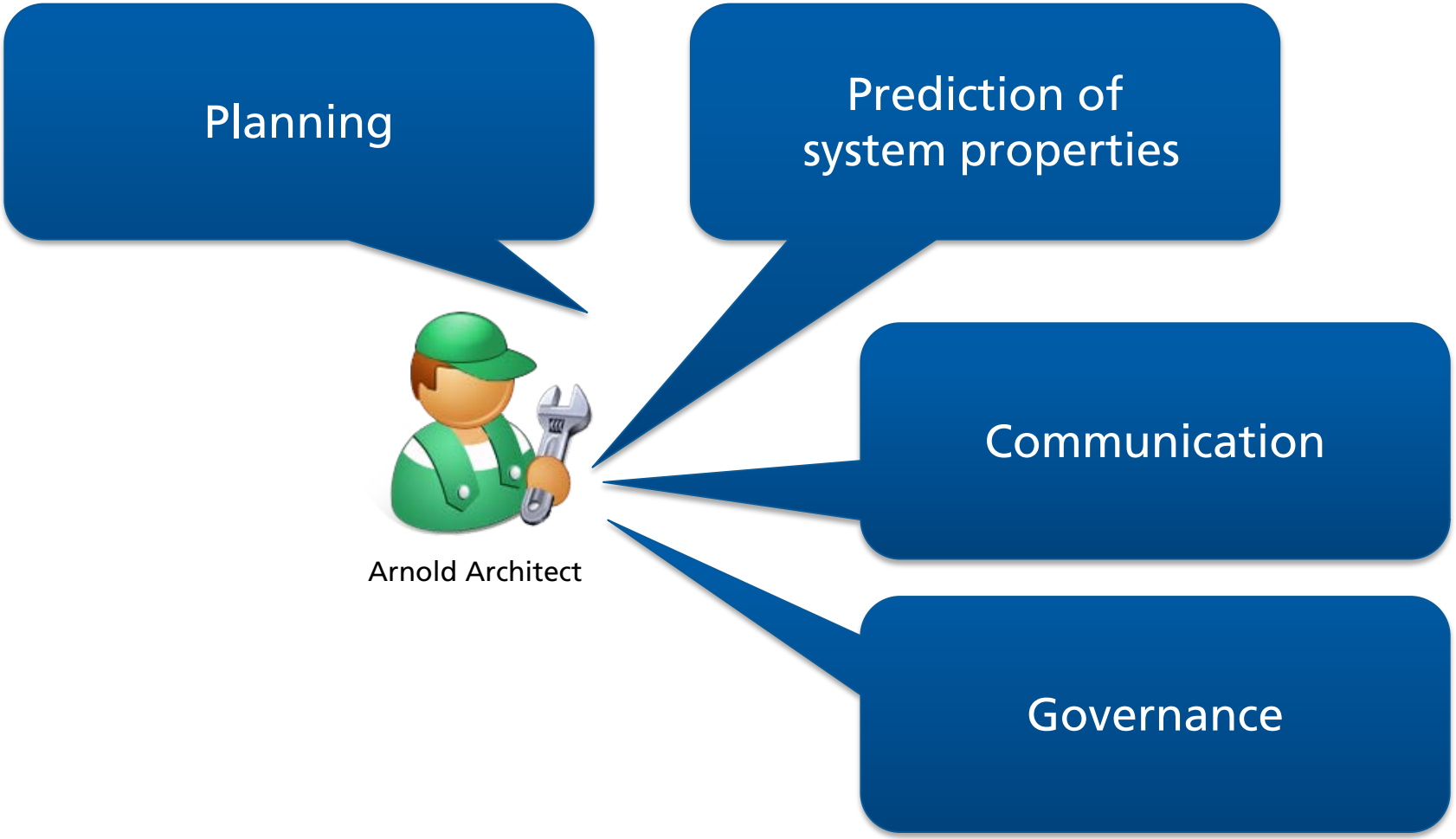
Paul Projectleader

Mike Manager

Quincy Quality

Marcus Marketing

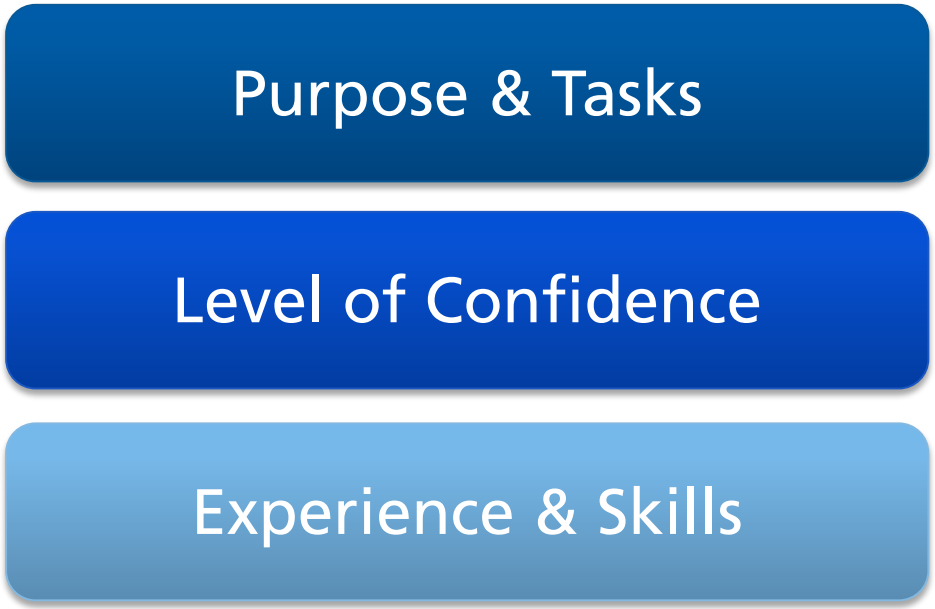
For what?



For what?

For what?

How *detailed* should an Architecture Documentation be?



Skills and Experience of Developers

Architectural Concepts

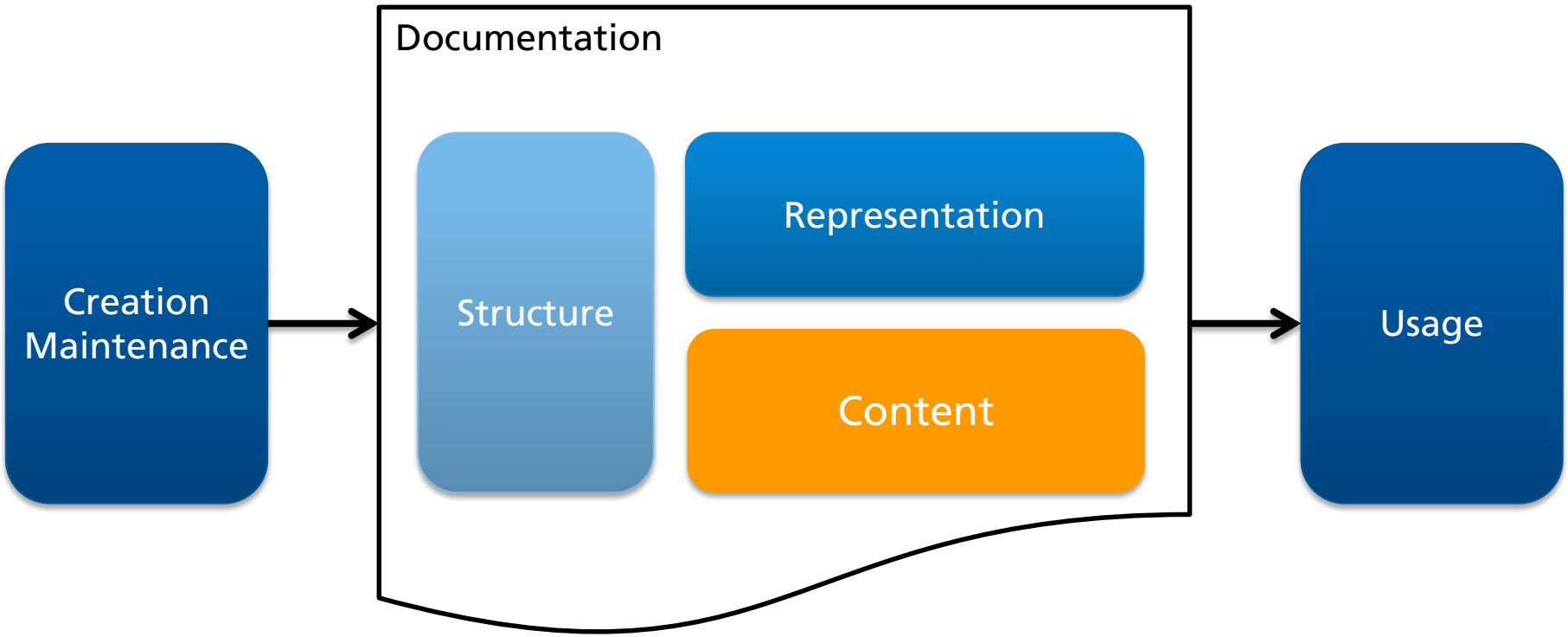
Domain

Daniel Developer

Development Process

Technologies

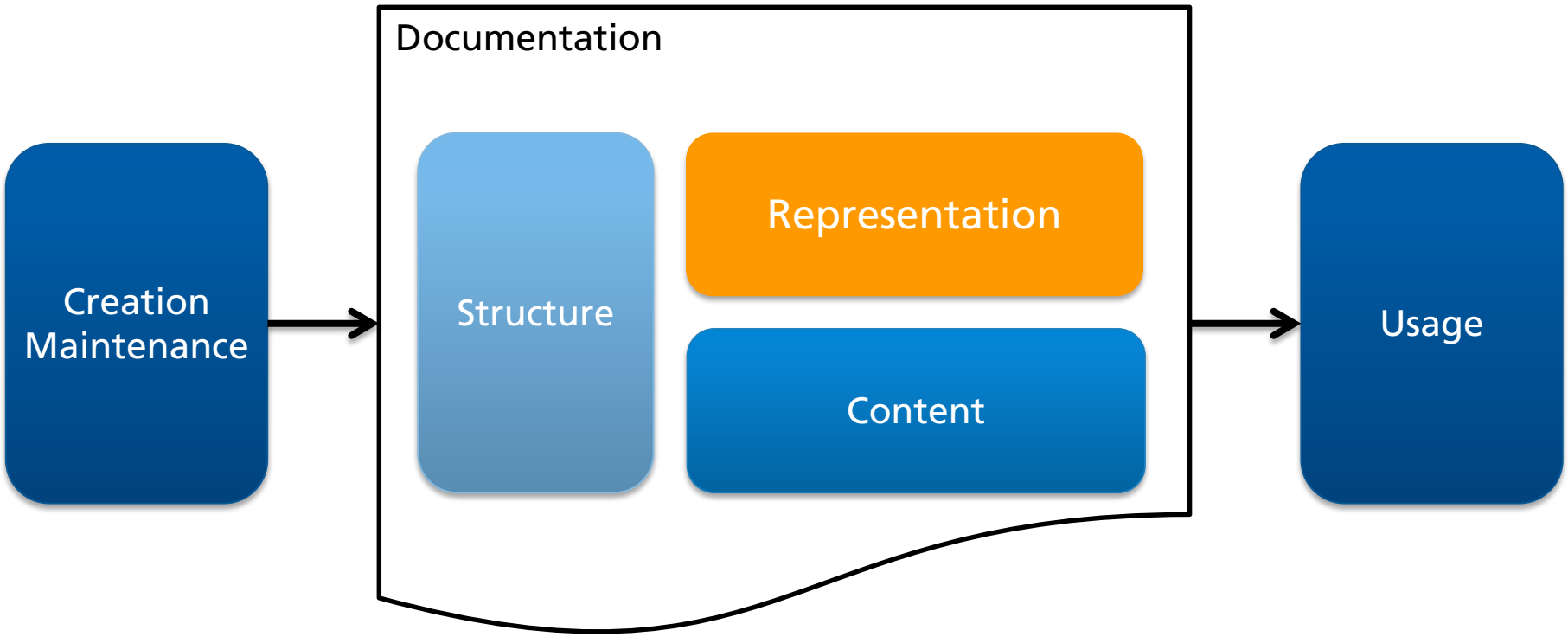
Architecture Documentation



Typical Content of Architecture Documentation

- Overview
  - System Overview
  - Context Delineation
- Architecture Drivers
  - Business Goals
  - Key Features
  - Quality Attributes
  - Constraints
- Design Decisions
  - Rationales
  - Traceability
  - Alternatives
- Evolution
  - Principles
  - Maintainability/Extensibility
  - Built-In Flexibility
- Solution Concepts
  - Runtime
    - Usage
    - Behavior
    - Structure (RT)
    - Technologies (RT)
    - Deployment
    - Configuration
  - Devtime
    - Structure (DT)
    - Technologies (DT)
    - Dev Process
    - Responsibilities
    - Production
    - Variant Management
  - Operation
    - Quality of Service (QoS)
    - Service Level Agreements (SLAs)

Architecture Documentation



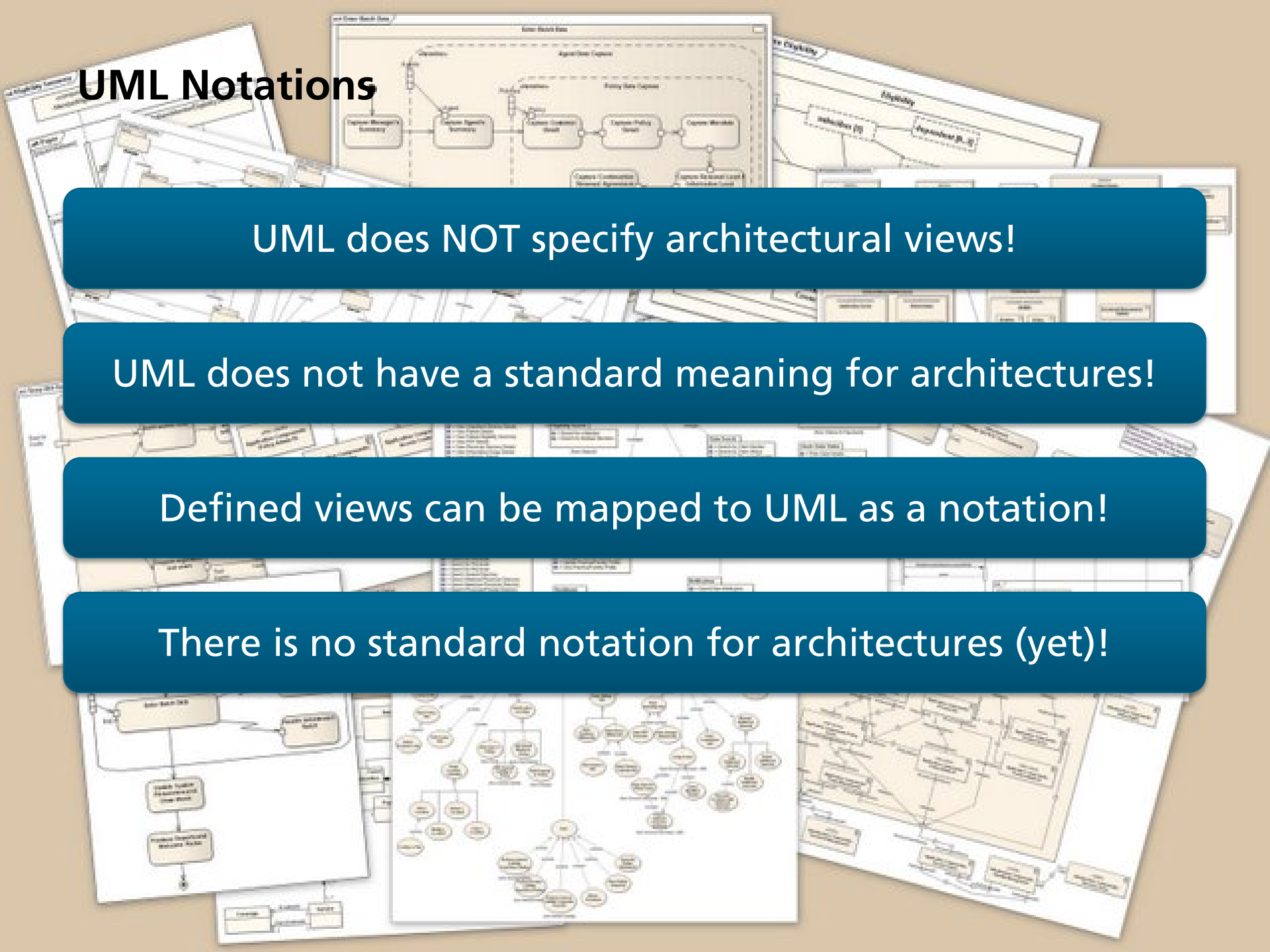
General Properties of Representation

- Readability
- Understandability
- Memorability
- Uniformity
- Consistency (Internal and External with other Documents)
- Compactness
- Completeness
- Correctness
- Suitability for reader
- Look and Feel (Usability)
- …

Representation

- **Graphical**, **textual** and **tabular** notations can be used to represent views
- Most organizations use the Unified Modeling Language (**UML**)
  - Different diagrams and element types
  - Textual specialization through **stereotypes** (e.g. «hardware», «task»)
- No direct and visual support for special aspects (like variability)
  - Manual definition of **UML profiles** or extension of the UML meta-model
  - E.g. variant elements are represented using different colors and/or stereotypes

UML Notations



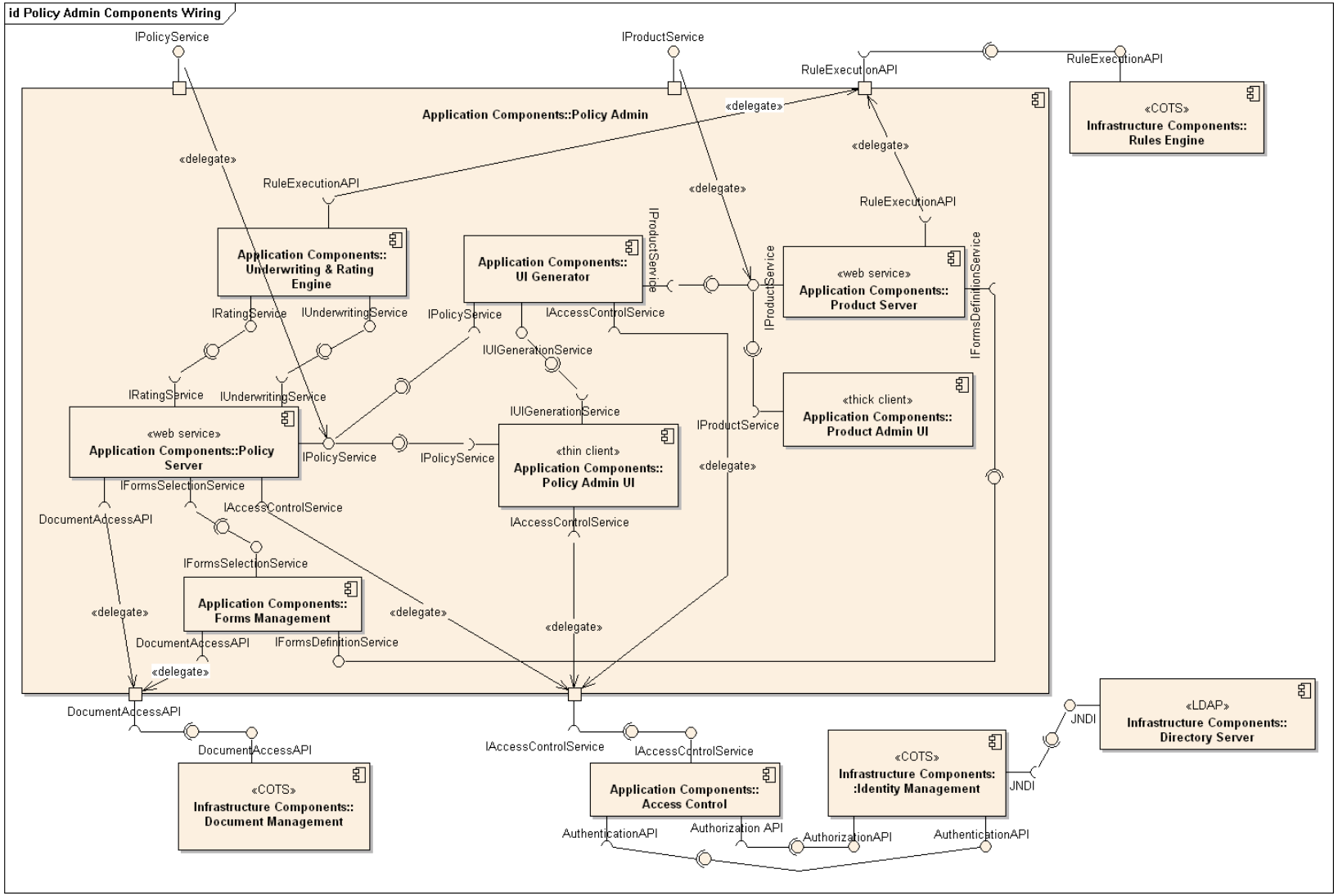
UML does NOT specify architectural views!

UML does not have a standard meaning for architectures!

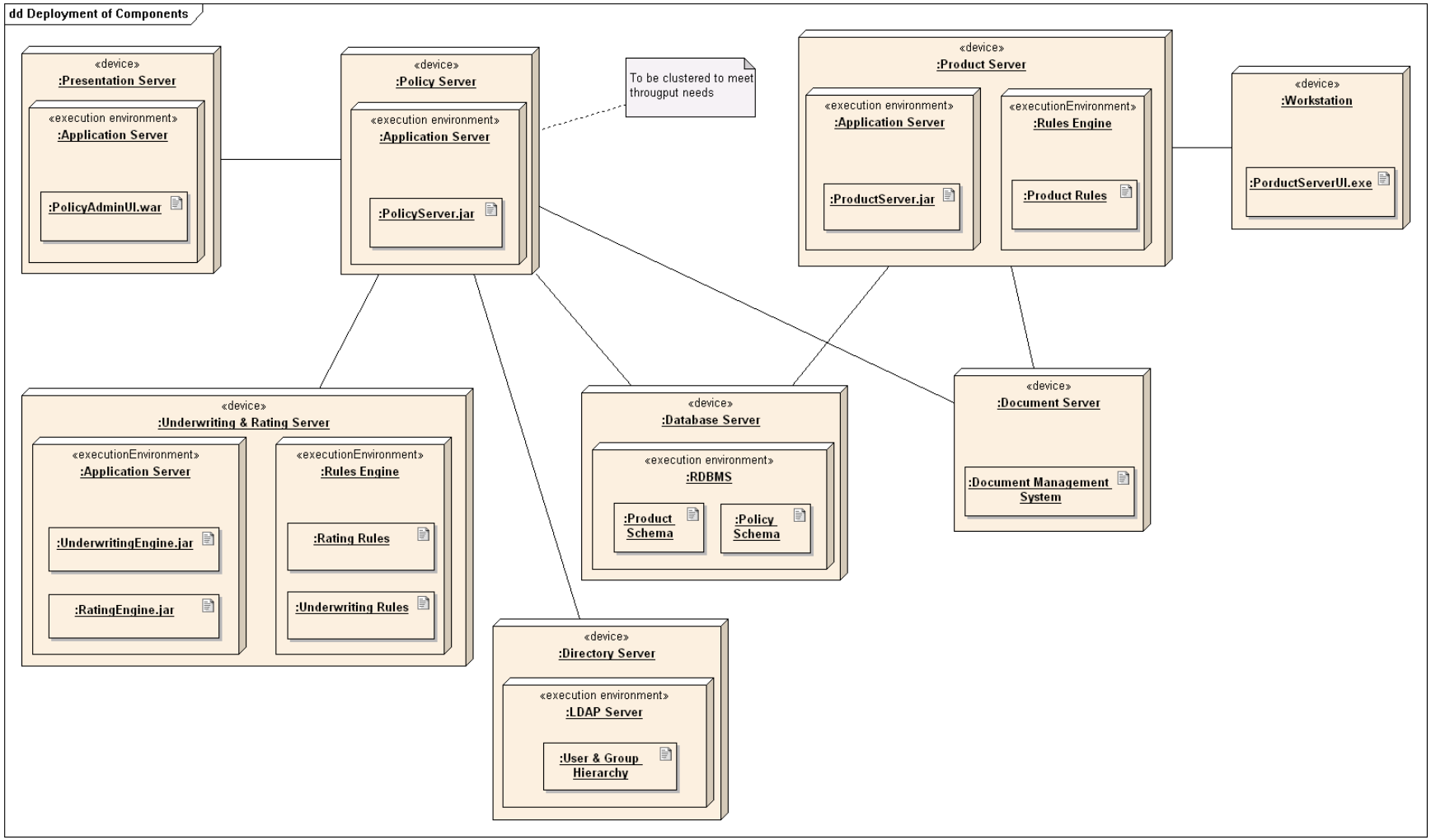
Defined views can be mapped to UML as a notation!

There is no standard notation for architectures (yet)!

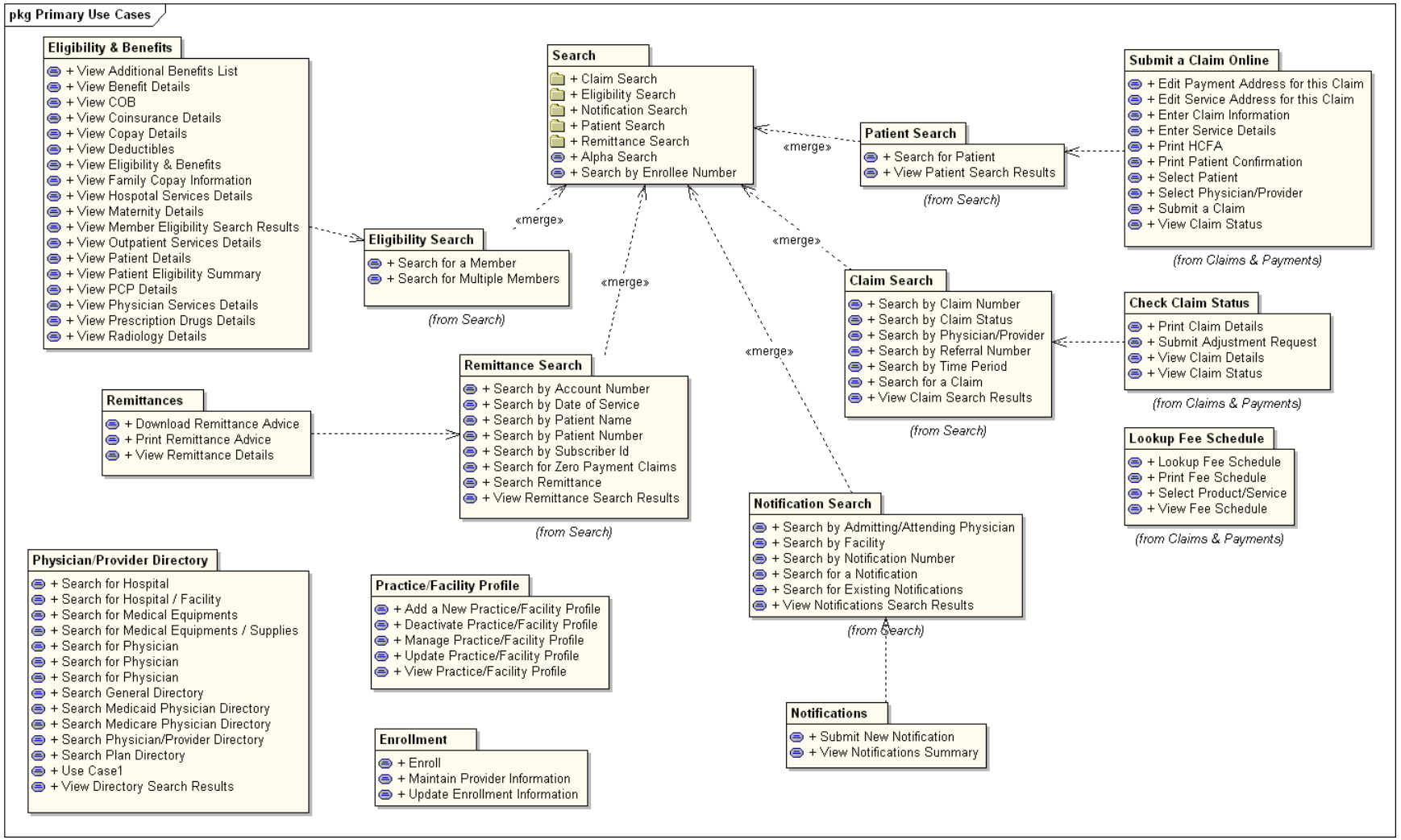
Component Diagram



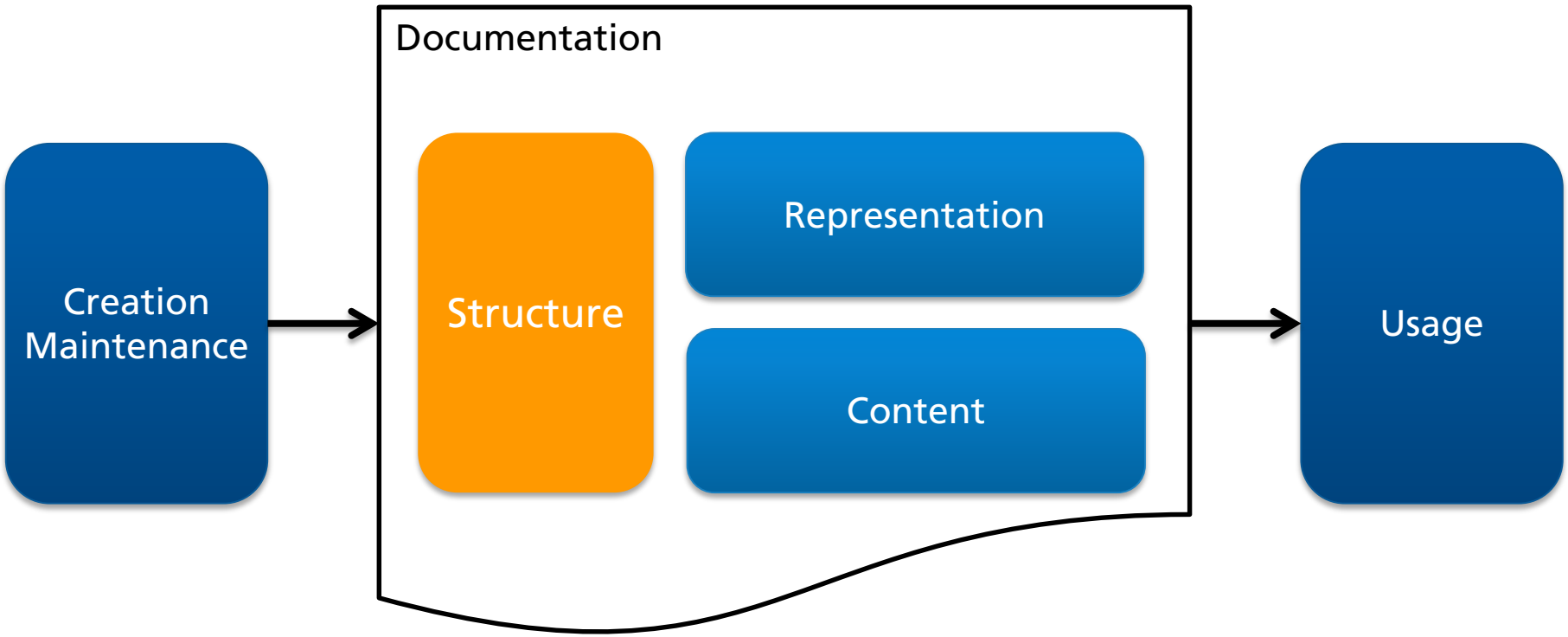
Deployment Diagram



Package Diagram



Architecture Documentation



Software Architecture Document – Example

RESCUER

Contents

1. Introduction 5
1.1. Purpose of this Document 5
1.2. Approach to Create System Architecture 5
1.3. Partners' Roles and Contributions 6
1.4. Document Overview 6
2. Conceptual Overview of the RESCUER System 7
2.1. Mission 7
2.2. Conceptual Architecture 9
2.3. Scope and Iteration 10
2.4. Locations and Stakeholders 11
2.4.1. Stakeholders 11
2.4.2. Locations 13
3. Architectural Drivers 14
3.1. Prioritisation 14
3.2. Business Goals 14
3.3. Key Functional Requirements 15
3.4. Devtime Requirements 21
3.5. Integration Requirements 21
3.6. Operation Requirements 22
3.6.1. Operability 22
3.6.2. Evaluation and Demonstration Capability 23
3.6.3. Installability 23
3.7. Runtime Quality Requirements 23
3.7.1. Availability 23
3.7.2. Robustness 24
3.7.3. Scalability 25
3.7.4. Safety 26
3.7.5. Upgradeability 26
3.7.6. Auditability 26
3.7.7. Usability 26
3.7.8. Variability 27

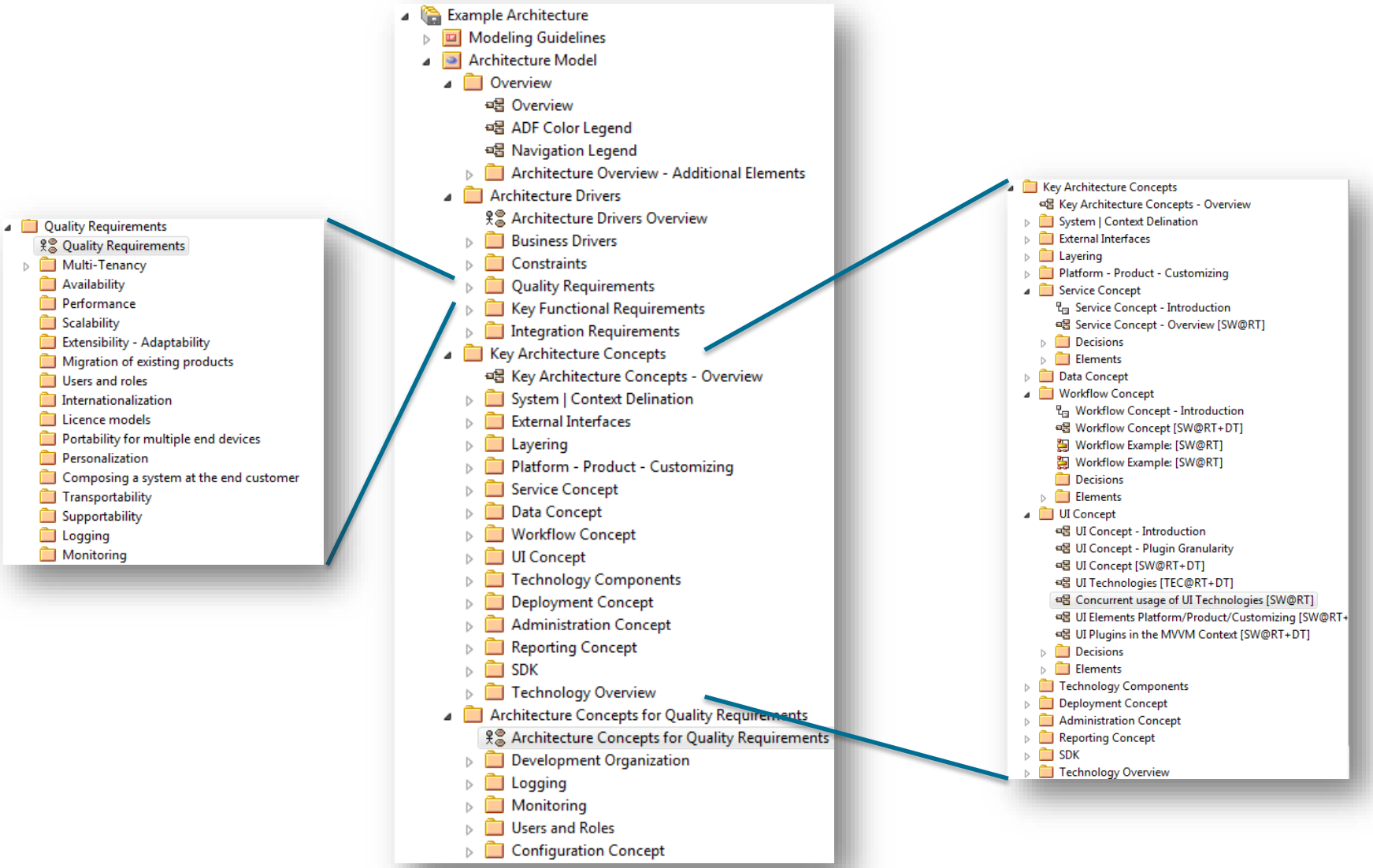
3

RESCUER

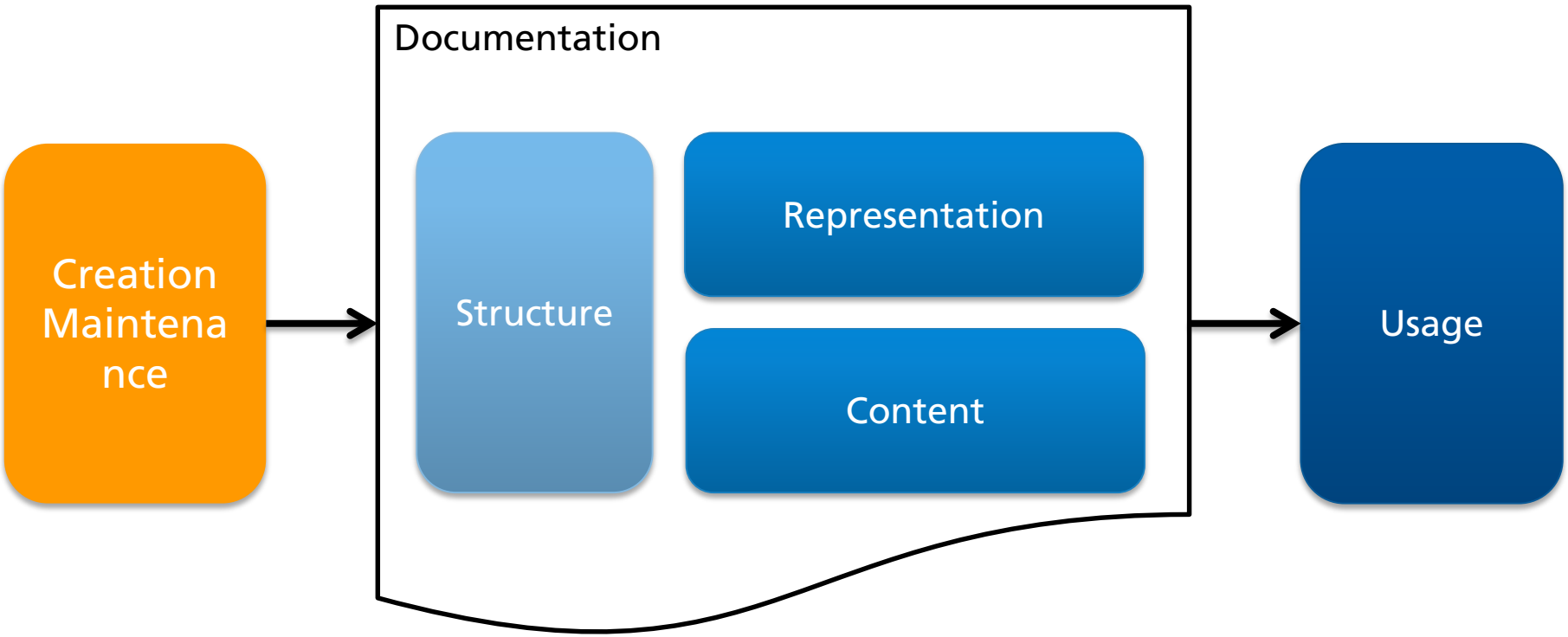
3.7.9. Credibility 28
3.7.10. Performance 28
3.7.11. Security 29
3.8. Constraints 30
4. Key Architectural Concepts 31
4.1. Context Delineation 31
4.2. Internal Structure 31
4.3. Client Server Separation 34
4.4. Conceptual Component Realisation 36
5. Perspectives 39
5.1. Data Perspective 39
5.2. Sub-Systems Perspective 43
5.2.1. Mobile Solutions 45
5.2.2. Communication Infrastructure 49
5.2.3. Data Analysis 52
5.2.4. Emergency Response Toolkit 55
5.2.5. Integration Platform 57
5.3. Development Perspective 61
5.4. Deployment Perspective 64
5.5. Usage Perspective 65
5.6. Task Distribution 66
5.6.1. MTM 66
5.6.2. DFKI 67
5.6.3. VOMATEC 67
5.6.4. UPM 67
5.6.5. USP 67
5.7. Design Decisions 68
6. Conclusion 75
Bibliography 76
Glossary 77
Abbreviations 77
Acronyms for the Modules to be developed 78

4

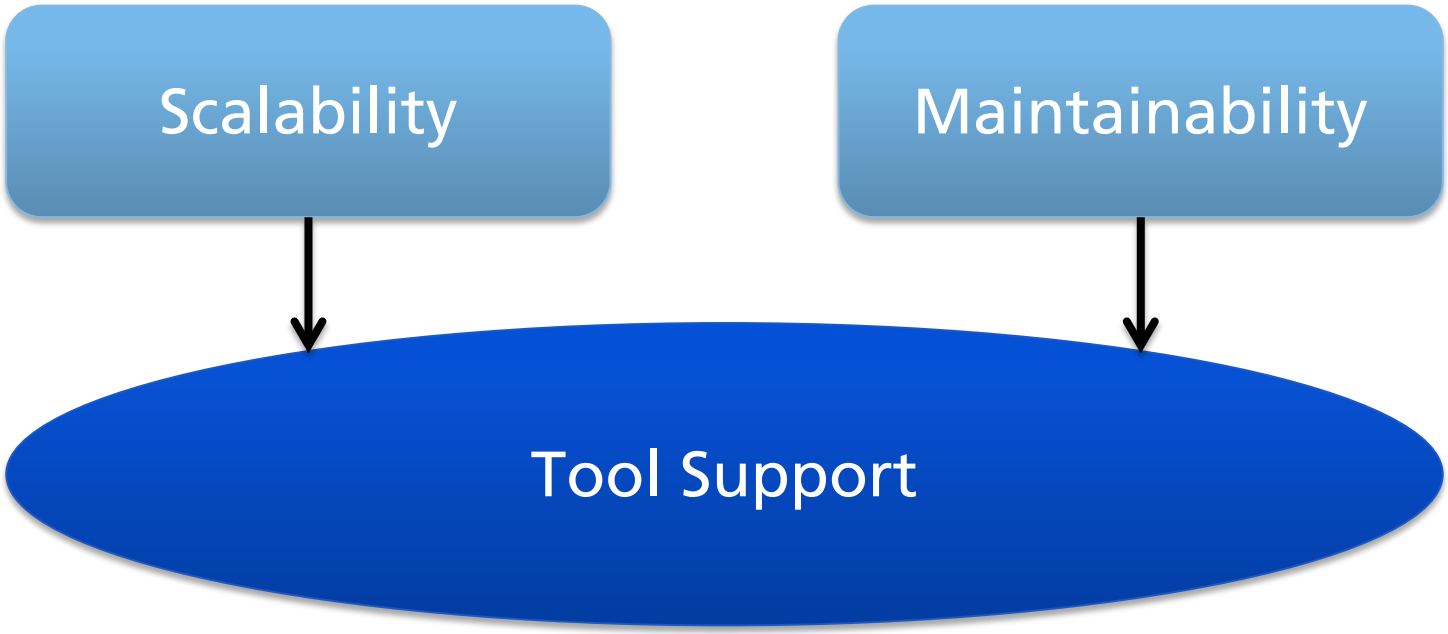
Structure - Example

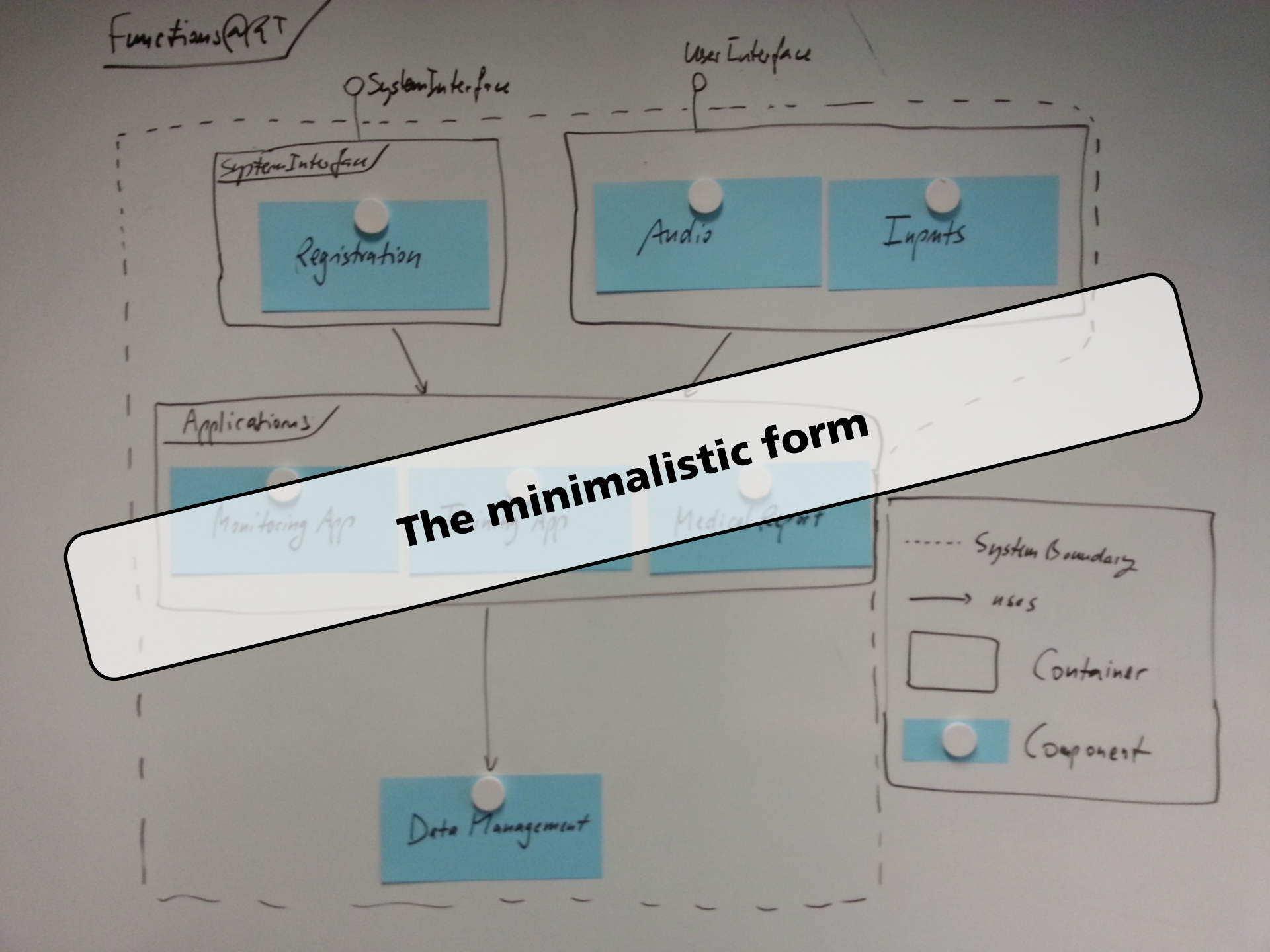


Architecture Documentation

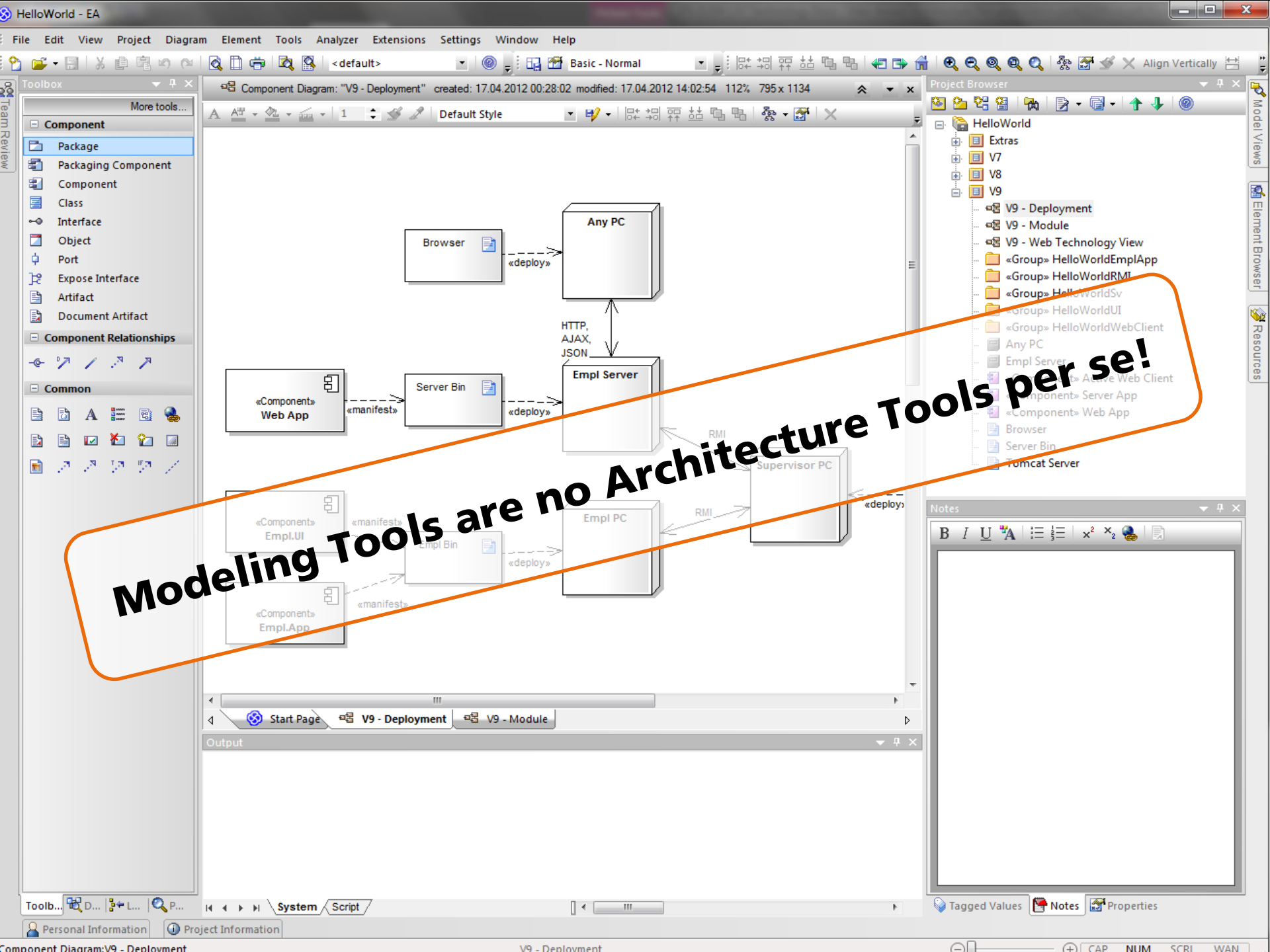


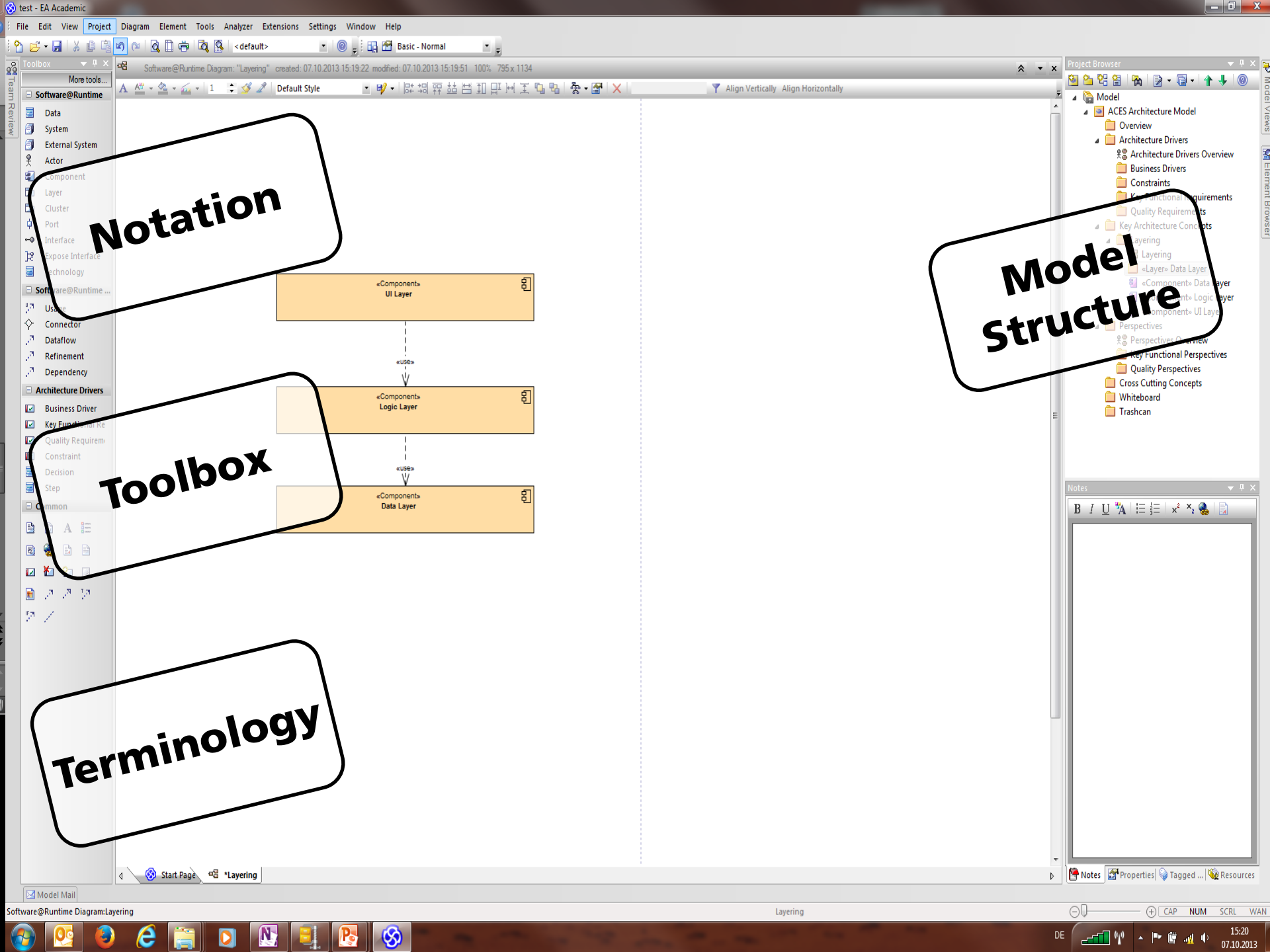
Key Challenges of Architecture Documentation





The minimalistic form


HelloWorld - EA
File Edit View Project Diagram Element Tools Analyzer Extensions Settings Window Help
Toolbox
More tools...
Component
Package
Packaging Component
Component
Class
Interface
Object
Port
Expose Interface
Artifact
Document Artifact
Component Relationships
Common
Component Diagram: "V9 - Deployment" created: 17.04.2012 00:28:02 modified: 17.04.2012 14:02:54 112% 795 x 1134
Default Style
Browser
Any PC
«deploy»
HTTP,
AJAX,
JSON
Empl Server
«Component»
Web App
«manifest»
Server Bin
«deploy»
RMI
Supervisor PC
«deploy»
«Component»
Empl.UI
«manifest»
Empl Bin
«deploy»
Empl PC
RMI
«Component»
Empl.App
«manifest»
Start Page
V9 - Deployment
V9 - Module
Output
System
Script
Project Browser
HelloWorld
Extras
V7
V8
V9
V9 - Deployment
V9 - Module
V9 - Web Technology View
«Group» HelloWorldEmplApp
«Group» HelloWorldRMI
«Group» HelloWorldSv
«Group» HelloWorldUI
«Group» HelloWorldWebClient
Any PC
Empl Server
«Component» Server App
«Component» Web App
Browser
Server Bin
Tomcat Server
Notes
Tagged Values
Notes
Properties
Personal Information
Project Information
Model Views
Element Browser
Resources
Modeling Tools are no Architecture Tools per se!


Notation
Toolbox
Terminology
Model Structure
«Component»
UI Layer
«use»
«Component»
Logic Layer
«use»
«Component»
Data Layer

Software Architecture Document – Example

RESCUER

4. Key Architectural Concepts

4.1. Context Delineation

Figure 5 shows the RESCUER system and the external systems and stakeholders around it.

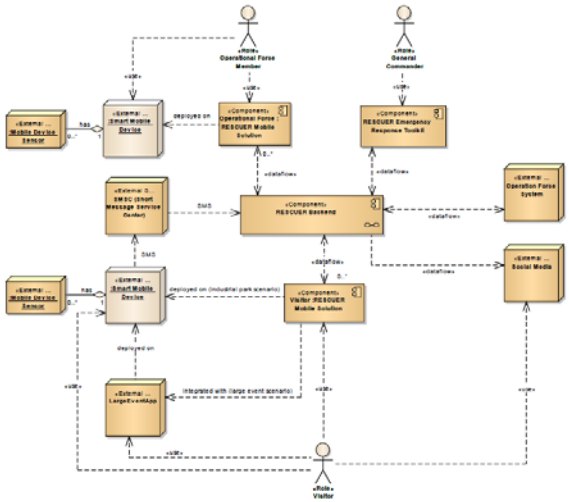


Figure 5 : Context delineation

4.2. Internal Structure

The overall RESCUER system is divided into several layers. Figure 6 shows the layers of the RESCUER system. Figure 7 shows the components inside the layers.

- *Mobile Application Layer*: This layer is responsible for collecting sensor data, user interaction data, multimedia data (image, video), and unstructured text report from the

32

RESCUER

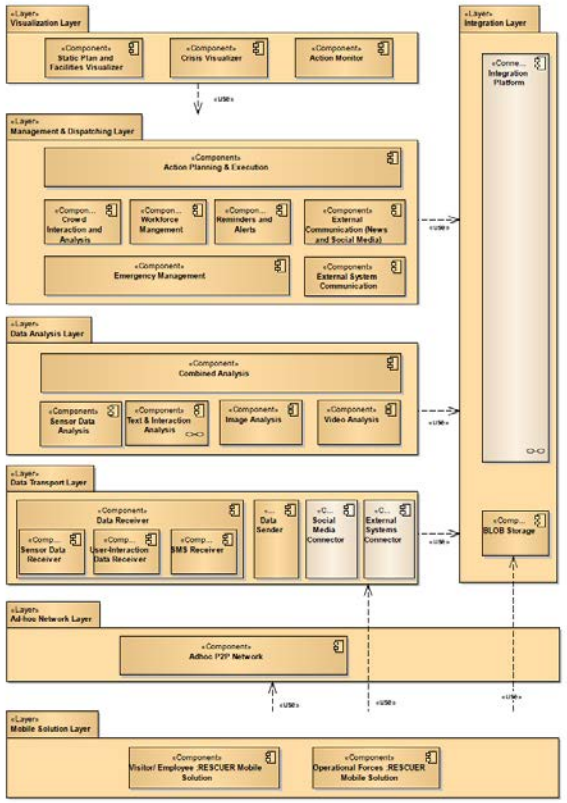
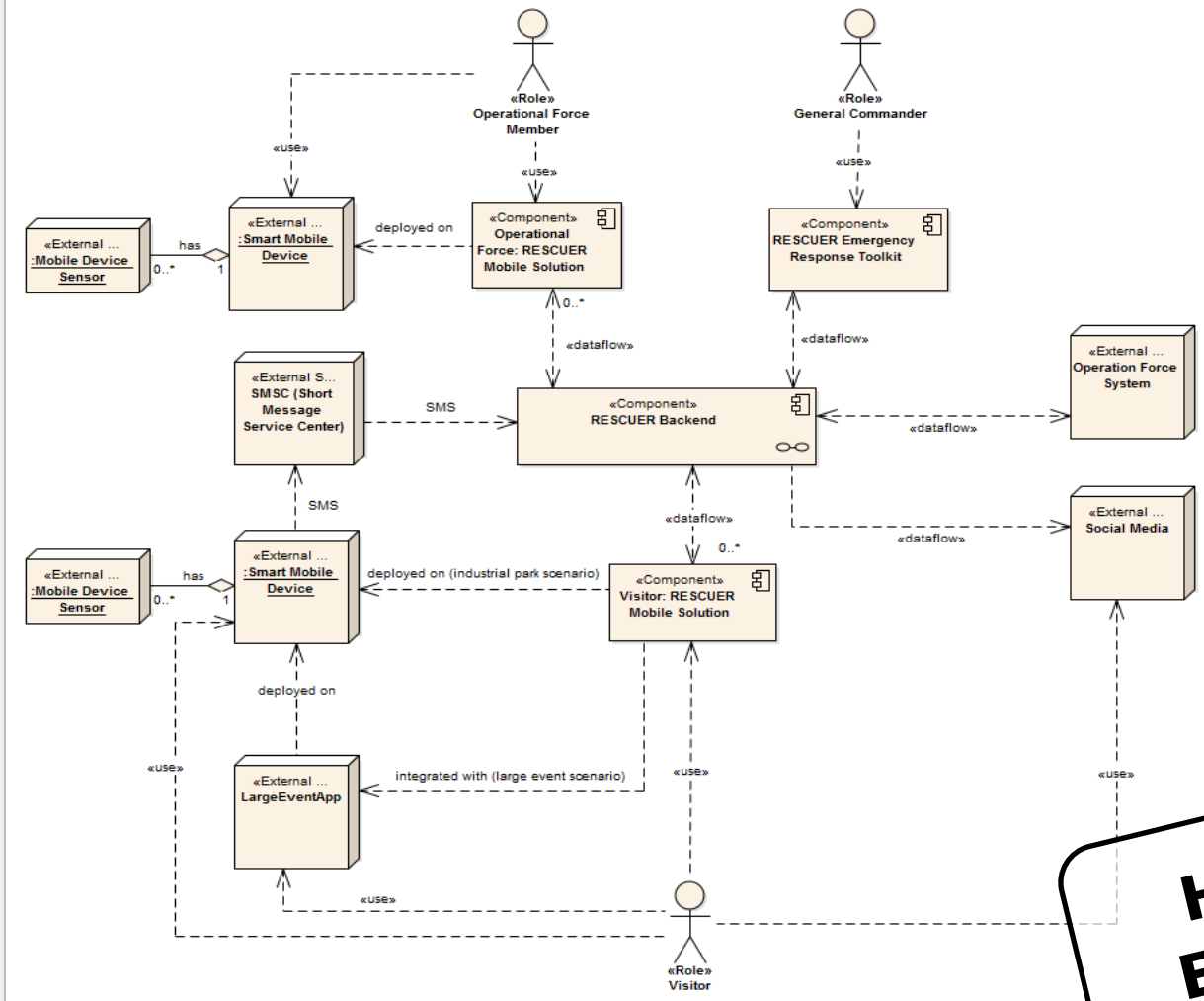


Figure 7 : Layers including functional components

34

Fraunhofer IESE

RESCUER
Rescuer Architecture Model
Overview
Overview
Overview
Overview - Emergency Situation
Overview - Mission
Stakeholders
Overview - Conceptual Architect
Overview - Scope
Overview - Usage
Architecture Drivers
Architecture Drivers Overview
Constraints
Key Functional Requirements
Quality Requirements
Business Drivers
Key Architecture Concepts
Process
Process
Context Delineation
Context Delineation
Internal Structure
Internal Structure - layers
Internal Structure - layers inc
«Connector» AMQP
«Connector» AMQP Connec
«Connector#» External Syst
«Component» Reminders ar
«Connector#» Social Media
Client Server Separation
Datatypes
Design Decisions
Perspectives
Elements

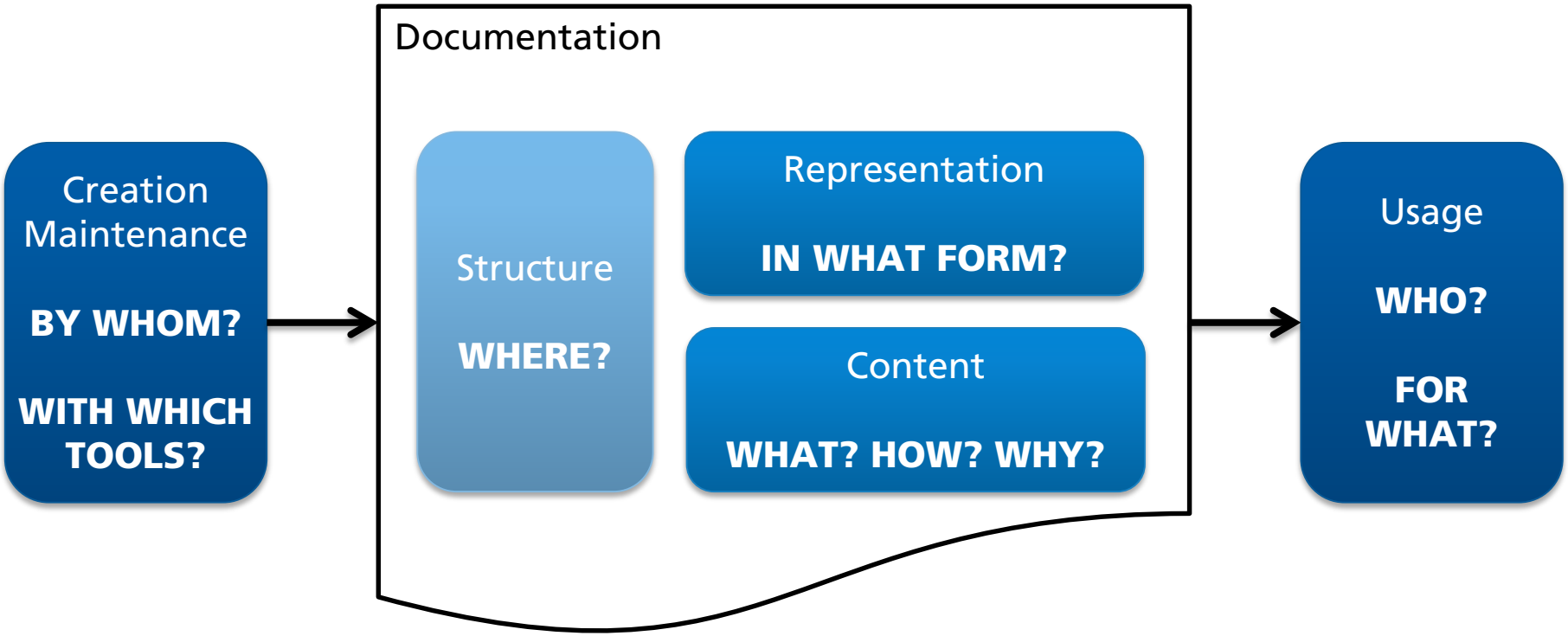


Context Delineation : Deployment diagram

Created: 30.04.2014 15:57:39
Modified: 30.04.2014 15:57:39
Project:
Advanced:

HTML Export

Architecture Documentation



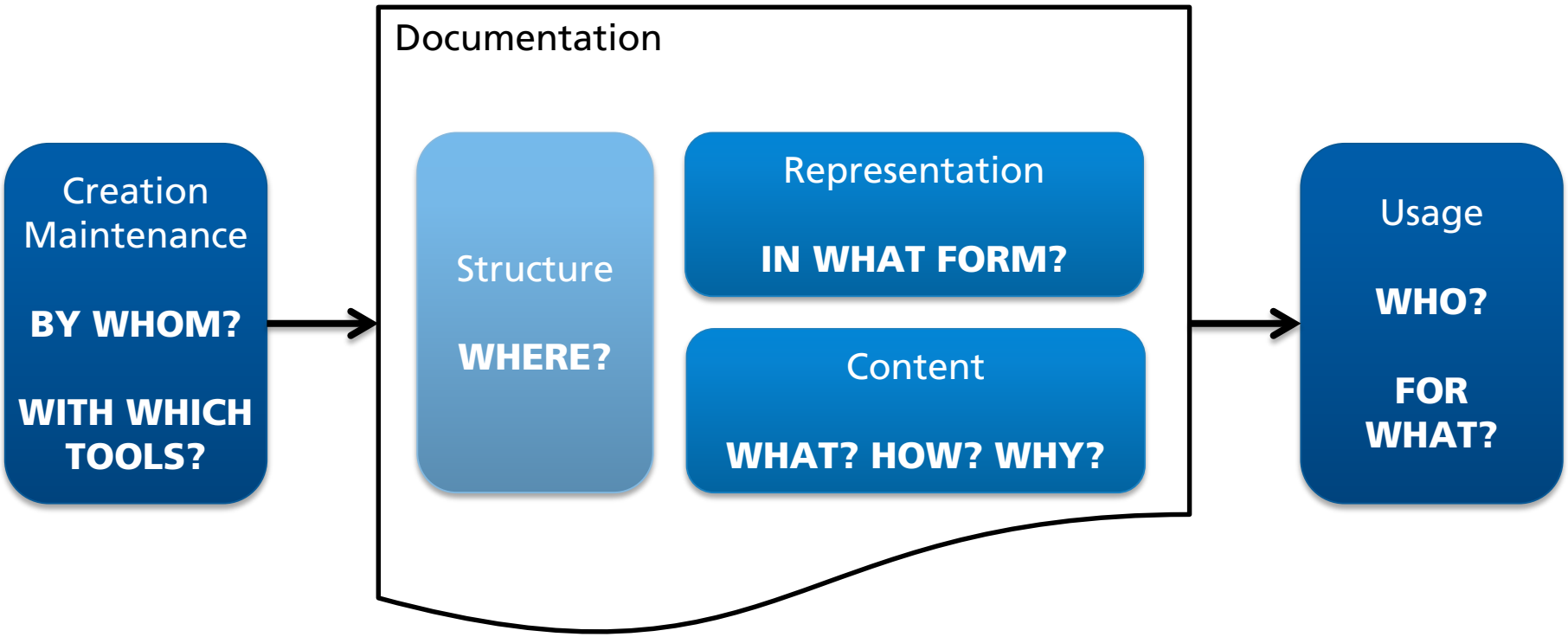
Documentation Quality

Rating		Severity of findings: Critical	Harmful	Minor	Harmless / Advantageous
Balance of findings	Mainly negative findings				
	Negative findings predominate				
	Positive findings predominate				
	Mainly positive findings				

Legend
N/A
1 – NO
2 –PARTIAL
3 –LARGE
4 –FULL

Wrap Up

Architecture Documentation



Typical Content of Architecture Documentation

- Overview
  - System Overview
  - Context Delineation
- Architecture Drivers
  - Business Goals
  - Key Features
  - Quality Attributes
  - Constraints
- Design Decisions
  - Rationales
  - Traceability
  - Alternatives
- Evolution
  - Principles
  - Maintainability/Extensibility
  - Built-In Flexibility
- Solution Concepts
  - Runtime
    - Usage
    - Behavior
    - Structure (RT)
    - Technologies (RT)
    - Deployment
    - Configuration
  - Devtime
    - Structure (DT)
    - Technologies (DT)
    - Dev Process
    - Responsibilities
    - Production
    - Variant Management
  - Operation
    - Quality of Service (QoS)
    - Service Level Agreements (SLAs)

General Properties of Representation

- Readability
- Understandability
- Memorability
- Uniformity
- Consistency (Internal and External with other Documents)
- Compactness
- Completeness
- Correctness
- Suitability for reader
- Look and Feel (Usability)
- …


architecture
development
software
systems
Architecture
requirements
technologies
decisions
quality
system
product
technology
architectural
Systems
organizations
practices
design
support
architectures
challenges
activities
context
competencies
key
time
SAVE
evaluation
engineering
IESE
results
help
order
compliance
business
Fraunhofer
products
architecture-centric
basis
aspects
Tekla
making
particular
like
projects
evolution
processes
generation
services
provide
make
clear
level
many
terms
models
customers
information
usage
one
use
need
implementation
organization
packages
goals
Therefore
Engineering
documentation
control