SIEMENS

Ingenuity for life

PLM and Innovation Excellence

Learning Campus

Your partner for Business Learning

Siemens Core Learning Program

Architectural Views & Documentation

Authors: Wieland Eckert, CT | Peter Zimmerer, CT | Sylvia Jell, CT | Rüdiger Kreuter, CT

Documentation?

An architectural decision that isn't written down has a lifespan of – …

What is the memory span of a goldfish?

[Ruth Malan]

Architectural views & documentation

Learning objectives

- Understand importance of documenting architectures & testing
- Know methods to specify and document an architecture
- Understand how to specify and document a test architecture
- Know ways to document testing

Architectural Views & Documentation

Agenda

(Software) Architecture Documentation

Test Architecture Documentation

Test Documentation

Summary

Exercise

Brainstorming / Discussion

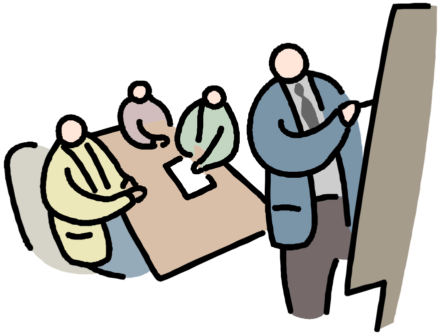
- **WHY** do you document your architecture?
- **WHO** has interests in your architecture documentation?
- **WHAT** should your architecture documentation contain?

Why do we need documentation?

Architecture is for communication – not for backup!

- **Communication** with different stakeholders and different roles in the project
- **Educating** new team members
- **(Re)understanding** the architecture, e.g. for maintenance reasons, integration, change of underlying platform
- Base for **architecture assessment**

War story: In a project for pre-paid mobile phone cards, documentation was either missing or difficult to read. Although the documentation did not lack any information, developers failed to read it.
As a result, at the end the documented and realized architecture differed so much that the system had to be reengineered.



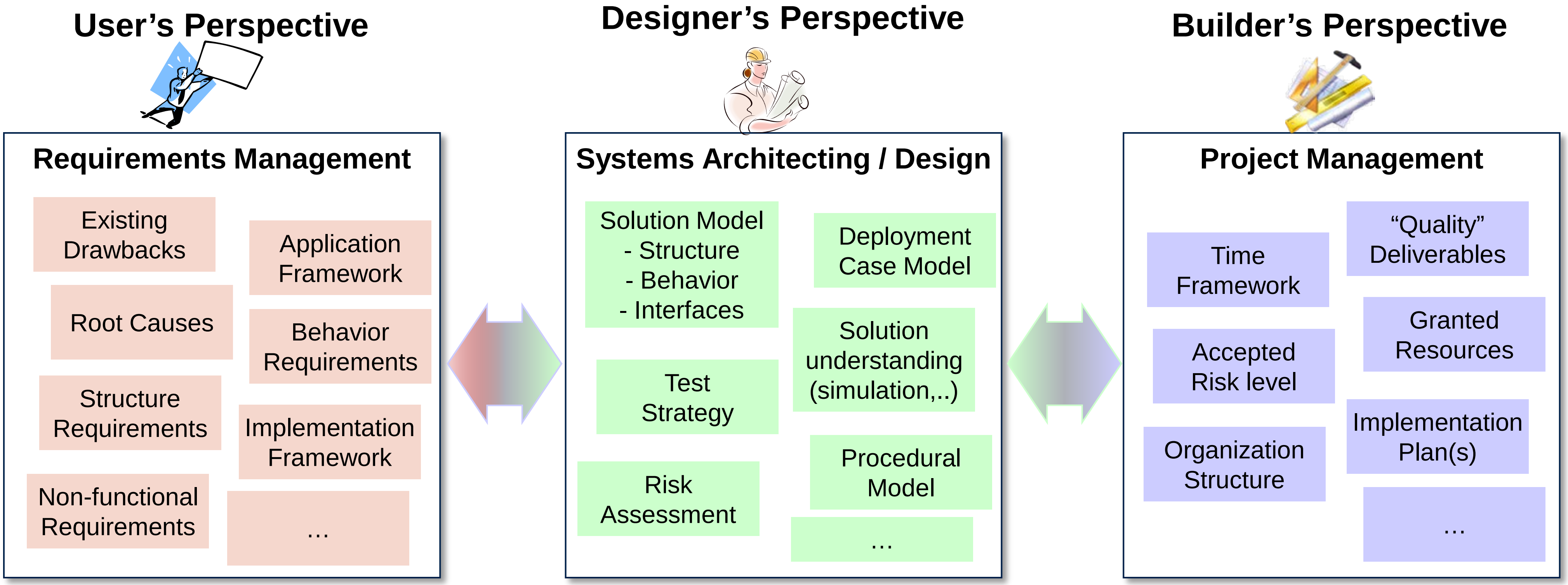
What architecture documentation needs to communicate – and to whom: (I)

Requirements Engineer, Customer	Clearly state all requirements and explain tradeoffs that have been made between conflicting requirements.
Developers	Architecture documentation is the single and decisive source for all development activities. Examples: Requirements, interfaces, logical, deployment, constraints, etc.
Testers	Specify the correct behavior of the system and how it is tested. Include the necessary details for unit, incremental and integration tests.
Product Manager / Marketing	Features, performance, technical specifications, etc.
Architects of other systems	Describe the system’s external interfaces and their Characteristics. Examples: Protocols, Electrical & mechanical attributes, etc.

What architecture documentation needs to communicate – and to whom: (II)

Project Managers	Provide information necessary to set up development teams (required skills, etc.) and to allocate project resources.
PLM	Documentation must enable PLM to decide whether a new member of the product family is in or out of scope.
Production Planner	Function test, mechanical tolerances, bill of materials, etc.
Maintenance, Service, Users, …	▪ Usability and interfaces; interworking topics ▪ Setup / startup, replacement / repair strategies
QA team	The documentation provides the basis for design and code reviews.

Architectural views & documentation
“Stakeholder Stories”



User’s Perspective

Requirements Management

- Existing Drawbacks
- Application Framework
- Root Causes
- Behavior Requirements
- Structure Requirements
- Implementation Framework
- Non-functional Requirements
- …

Conceptual and Context views
Operation and Behavior
Maintenance
Support Systems

Designer’s Perspective

Systems Architecting / Design

- Solution Model
  - Structure
  - Behavior
  - Interfaces
- Deployment Case Model
- Test Strategy
- Solution understanding (simulation,..)
- Risk Assessment
- Procedural Model
- …

Logical Views
Architecture View
Scope
Function

Builder’s Perspective

Project Management

- “Quality” Deliverables
- Time Framework
- Granted Resources
- Accepted Risk level
- Implementation Plan(s)
- Organization Structure
- …

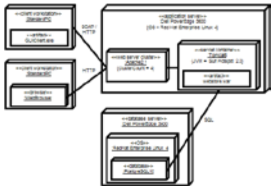
Physical views
Implementation
Manufacturing
Deployment

Exercise: Add „Tester‘s Perspective“

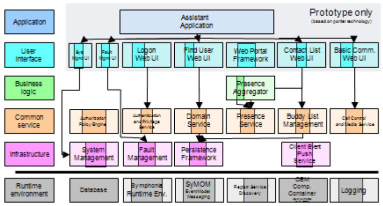
Documenting is part of the design work

Documenting is a continuous process: avoid doing it "post-mortem"!

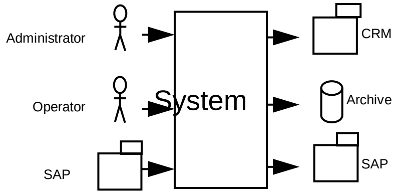
Deployment



Base Line

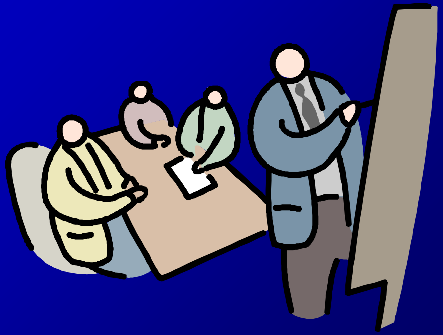


Boundaries



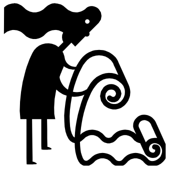
Principles

Always include the decisions and rationale in the documentation!



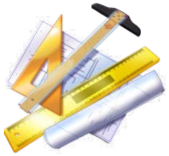
What should be documented?
What would I like to know about the architecture?

SIEMENS
Ingenuity for life



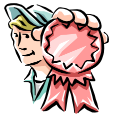
- **Requirements** driving the architecture
- Main **functional** requirements
- **Non-functional** requirements describing the quality attributes

- **Context** – The boundaries of the system



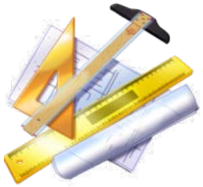
- The **architecture** itself – Structure and behavior in different views of the system, interfaces, protocols

- How the architecture **addresses quality attributes**, **non-functional** requirements and **cross-cutting** concerns



- The **design rationale** behind the architecture

What should be documented?
What would I like to know about the architecture?



- Introduction describing main use case and context of the system.
- Context – boundaries and environment of the system; interworking, upgrade or replacement strategy if necessary.
- The functional and non-functional requirements for the system.
- The architecture itself:
  - Document how the system is decomposed and structured.
  - Detailed description why this decomposition has been chosen.
  - Chapters describing the electrical, mechanical, electronic and software related aspects of the system.
  - Descriptions of the behavior of the system and its subsystems.
  - Description of the interfaces of the system and its subsystems.
- The design rationale behind the architecture.
- Describe how the architecture satisfies the
  - functional requirements and
  - the non-functional requirements.
- Realization strategy for each part and the complete system.
- Estimation on capabilities required for the development team.
- A manual, describing how the system can be used, tested and debugged.
- Description of how the system can be modified (e.g. extended or refurbished).

Documentation approaches

Text Files

Diagrams

Drawings

Models

WORD or WIKI? UML or text2UML …?

Documentation approaches – Helpful questions

Lifecycle / Lifetime of the information

Ongoing Discussion
One-time information
Short-lived
Long-lived

Who is the Audience?

Stakeholder

WORD or WIKI? UML or text2UML …?

Access/Usage Characteristics

Lecture providing background knowledge
Reference manual
Daily use for everyday work
Nearly never read
Access/Navigation characteristics
(direct/random vs sequentially)

Change Characteristics

Lightweight (change anytime / by everone)
One Author
Change process

Best practices in documenting architecture (1)

1. Documents need **product quality** like implementation artifacts: they are subject to versioning, change management, refactoring
2. Write documentation from the **reader's point of view**
3. Avoid unnecessary **repetition**
4. Avoid ambiguity → start writing a **glossary**
5. Use **standard templates**
   - From your organization
   - From reference processes

War Story: In OpenSOA there were different qualities and different view sets in the documentation.
So these different documents couldn't be read consistently.
After a review **these documents had to be refactored to achieve consistency.**

Best practices in documenting architecture (2)

6. Document the **rationale**

7. Keep the document **current** but not too current

8. For each document assign one unique document **owner**

9. A document shouldn't exceed 50 pages to keep it **simple** and **expressive**. Use hierarchical top-down approach to partition
   - Strategic design, principles
   - Cross-cutting concerns
   - Subsystems

10. **Review** documents to assure their quality

War Story: In Sira the new team member **received 4 functional description documents of about 400 pages total – he had no central point of access and no recommended course through the documents.** He needed to take other informal paths to get the required information – the documents couldn't provide this function.

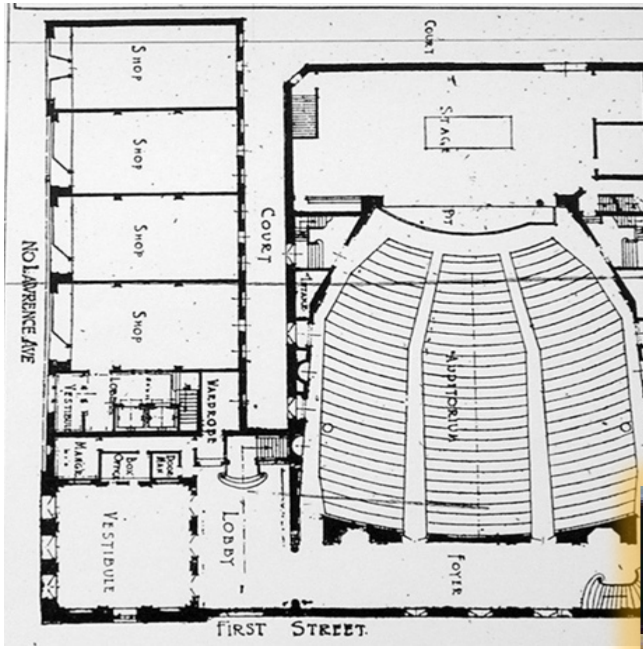
Not all information ends up in the specification – The project diary

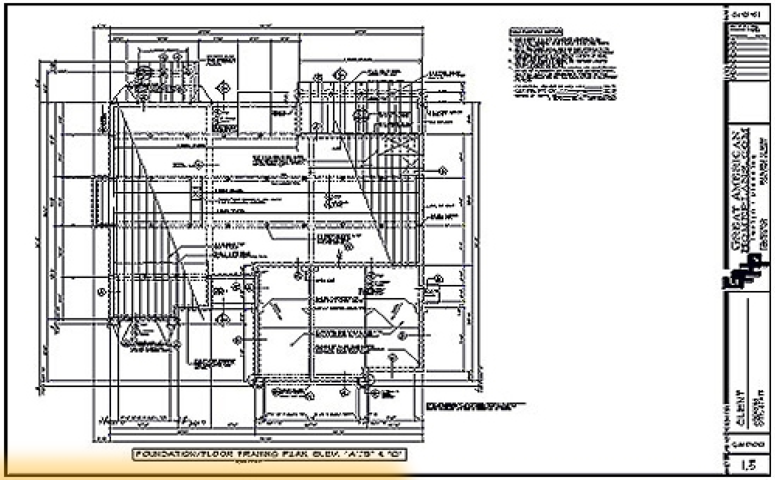
- This diary helps to document the history of
- Major changes in project strategy or planning
- Changes in the environment, including hardware and software – anything that can affect your product
- Ideas for new features
- Problems you may encounter and need to investigate
- Alternative designs and implementations
- Quick descriptions of each issue to be decided – Here you can also describe decisions that won't end up in the specification

Typical styles of diaries: Blogging application, wiki, SharePoint, formless documents

Hint for architects: If there is no official diary, write your own!

Modeling: Views in other engineering disciplines





View sets in system and software development

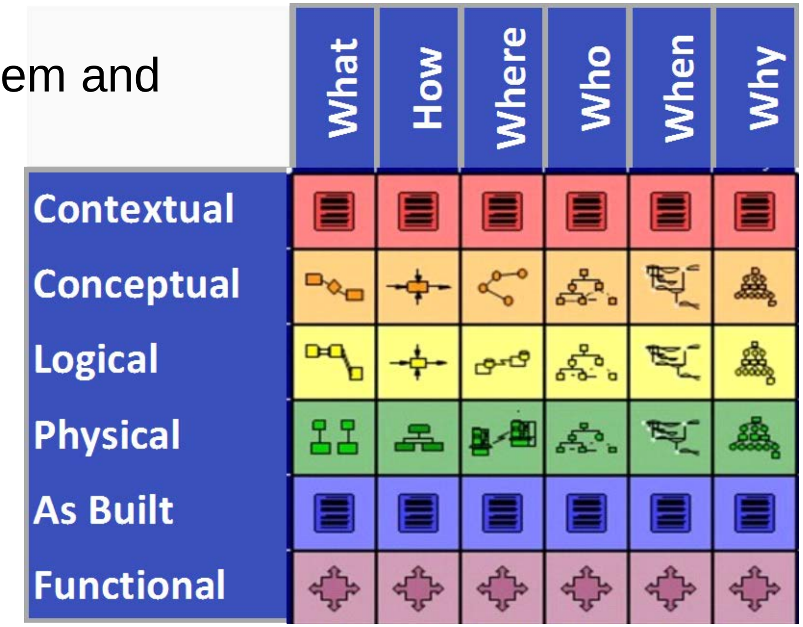
There exist different sets of views in the area of system and software engineering, such as:

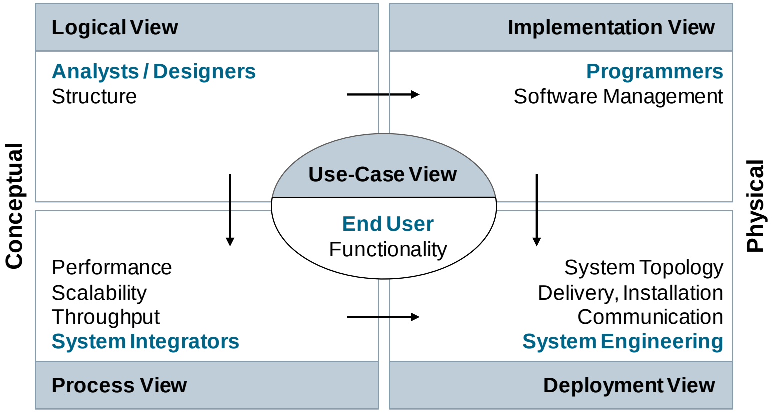
- Zachmann framework
- 4+1 Views by Kruchten
- Reference model for open distributed processing

These address different aspects:

- Informational aspect
- Functional aspect
- Physical aspect
- Implementation aspect
- Technological aspect
- Enterprise aspect

Depending on the goal of the view set there might be some additional views




Logical View

Analysts / Designers
Structure

Implementation View

Programmers
Software Management

Use-Case View

End User
Functionality

Performance
Scalability
Throughput
System Integrators

Process View

System Topology
Delivery, Installation
Communication
System Engineering

Deployment View

Conceptual

Physical

SIEMENS

Ingenuity for life

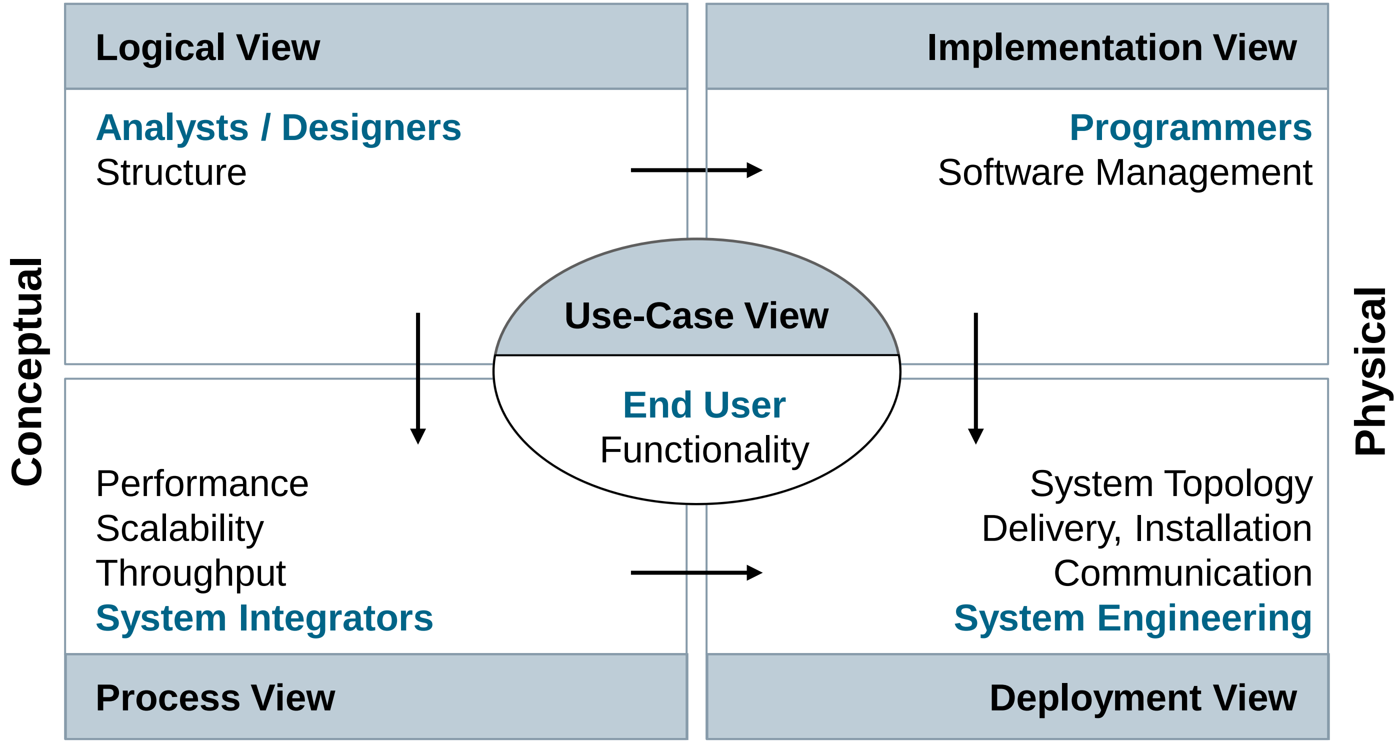
Views according to the Zachmann Framework

“Enterprise Architecture” Views (generic)

Audience / Perspectives	*What*	*How*	*Where*	*Who*	*When*	*Why*	Model Names
Executive (Business Context Planners)	**Contextual View** Scope and Context Identification						**Scope, Context**
Business Management (Business Concept Owners)	**Conceptual View** Business Definition Models						**Business Concepts**
Architect (Business Logic Designers)	**Logical View** System Representation Models						**System Logic**
Engineer (Business Physics Builder)	**Physical View** Technology Specification Models						**Technology Physics**
Technician (Business Component Implementers)	**Implementation View** Tool Configuration Models						**Tool Components**
Enterprise (User)	*The implemented and instantiated operational enterprise*						*Operations Instances (Implementations)*

4+1 View (Kruchten)

Rational Unified Process 4+1 View introduced by Philippe Kruchten



The 4+1 View explained

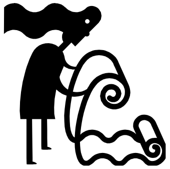
Note: A view is more than a set of diagrams.
It needs graphical representations and textual explanations

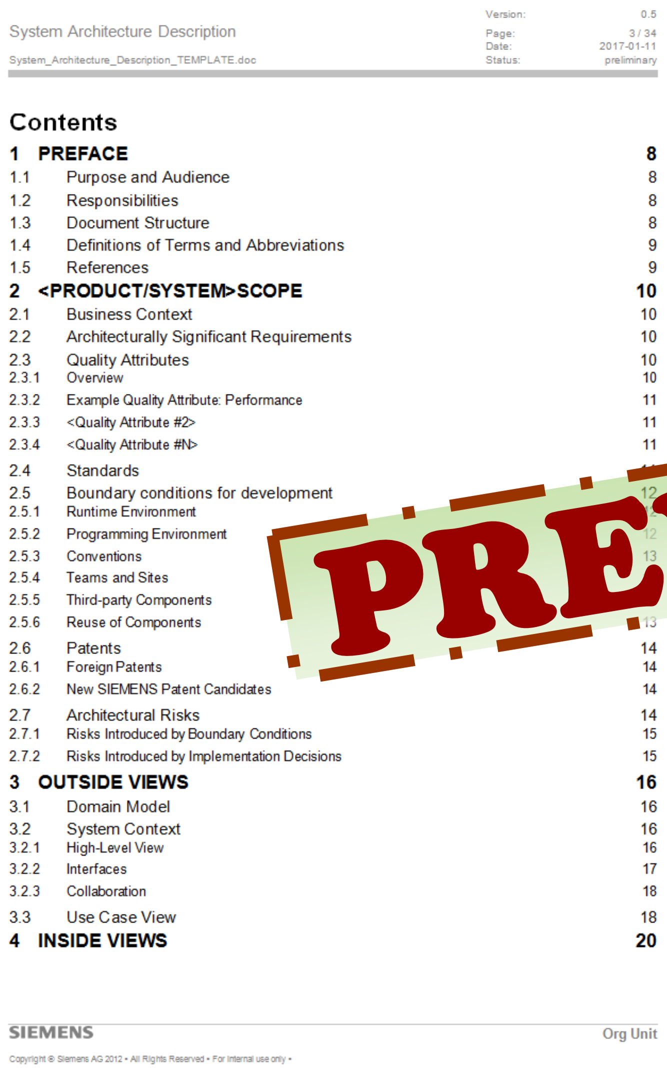
- **User view**
  - The ***Use Case View*** explains all possible scenarios users expect from the system
- **Functional aspects:**
  - The ***Logical View*** shows how the functionality defined in the use cases is modeled, while the
  - ***Development / Implementation View*** shows how the functionality is implemented (using source code, libraries, executables, documents, ...)
- **Non-functional aspects**:
  - The ***Process View*** illustrates how artifacts will be executed in terms of concurrency, scalability, synchronization
  - The ***Deployment View*** maps these software artifacts to concrete hardware entities and shows the distribution of functionality
- Each view can be modeled using diagrams (e.g., UML), but it is more important what is in the view (semantics) than how to express it syntactically
- Likewise, it is important that the stakeholders targeted have knowledge of what the views mean

Proposal for the structure of the software / system architecture documentation

SIEMENS

Ingenuity for life





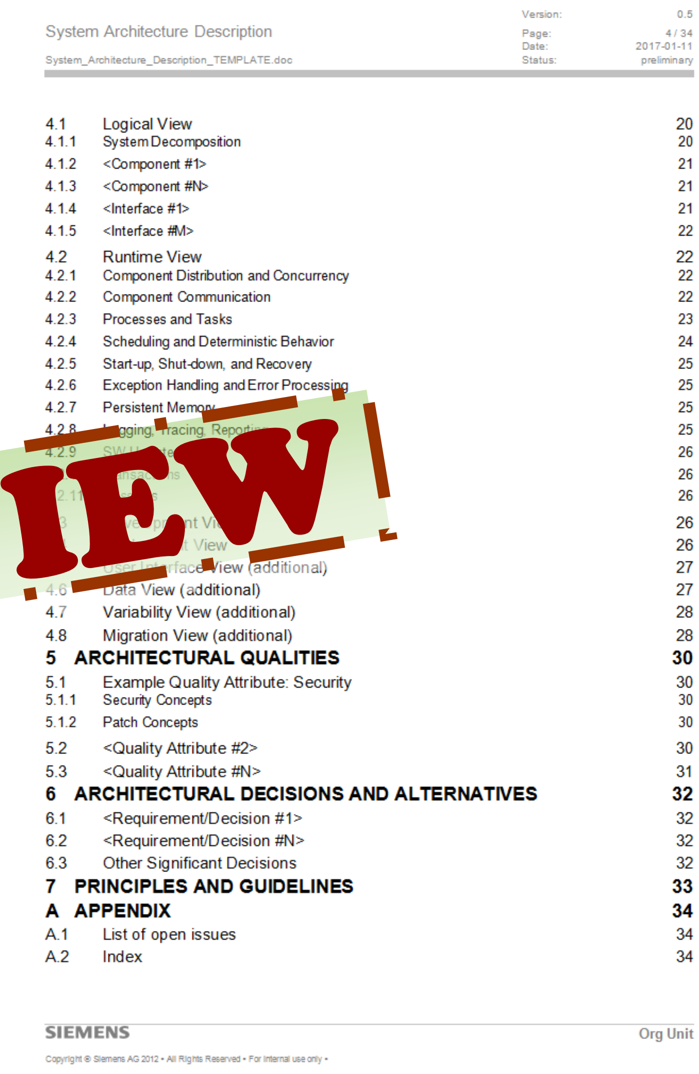
System Architecture Description | Version: 0.5 | Page: 3 / 34 | Date: 2017-01-11 | Status: preliminary

System_Architecture_Description_TEMPLATE.doc

Contents

SIEMENS | Org Unit

Copyright © Siemens AG 2012 • All Rights Reserved • For internal use only •



System Architecture Description | Version: 0.5 | Page: 4 / 34 | Date: 2017-01-11 | Status: preliminary

System_Architecture_Description_TEMPLATE.doc

SIEMENS | Org Unit

Copyright © Siemens AG 2012 • All Rights Reserved • For internal use only •

PREVIEW

Solution-space views

Key Question: How?

White Box views

Logical

- Structure
- Behavior

Physical

- Deployment
- Spatial
- User Interface / HMI

Discipline-Specific views

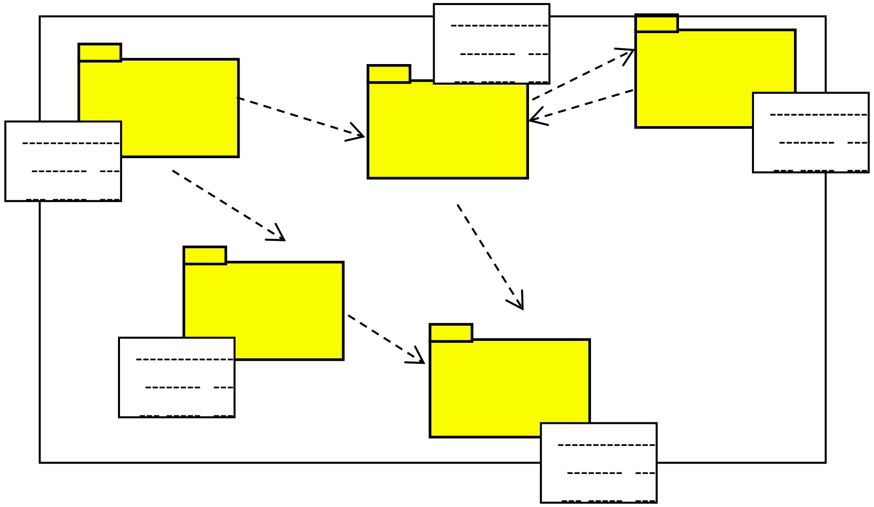
- Electronics, mechanics, software, data, thermal flow, etc.

Logical views (Structure)

A functional decomposition of the system into (logical) elements/building blocks

Consists of

- Overview diagram
- Logical elements and their dependencies
- Specification of each logical element



Specification of a logical element

Name

Description

(i.e. function for which the block is responsible)

Optional: Rationale

(why has the block been formed? What is hidden in it?)

Interfaces - externally visible features

(export and import, a.k.a provided and required)

Optional: Internal structure

(i.e. lower level logical building blocks) and their dependencies

Requirements for the logical element

(allocated from system requirements + newly derived, decomposed and detailed requirements)

Logical views (Behavior)

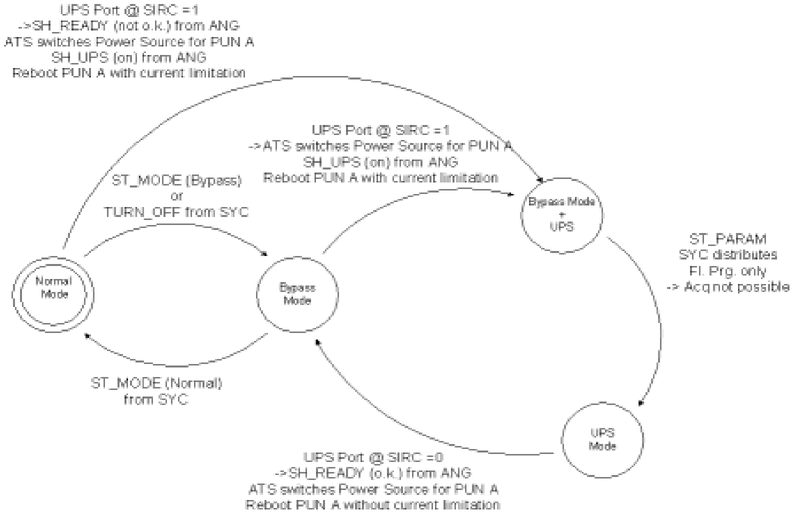
Sometimes called ***Process view***

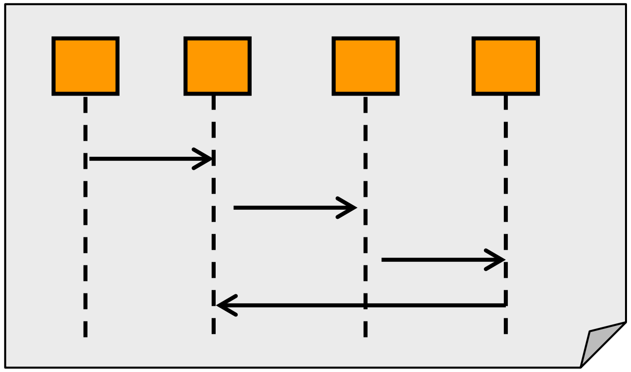
Shows the behavioral aspects of the system

- Intended and unintended
- Static and dynamic

Expressed using:

- Activities
- Sequence diagrams
- State charts
- Timing diagrams
- Graphs
- Etc.

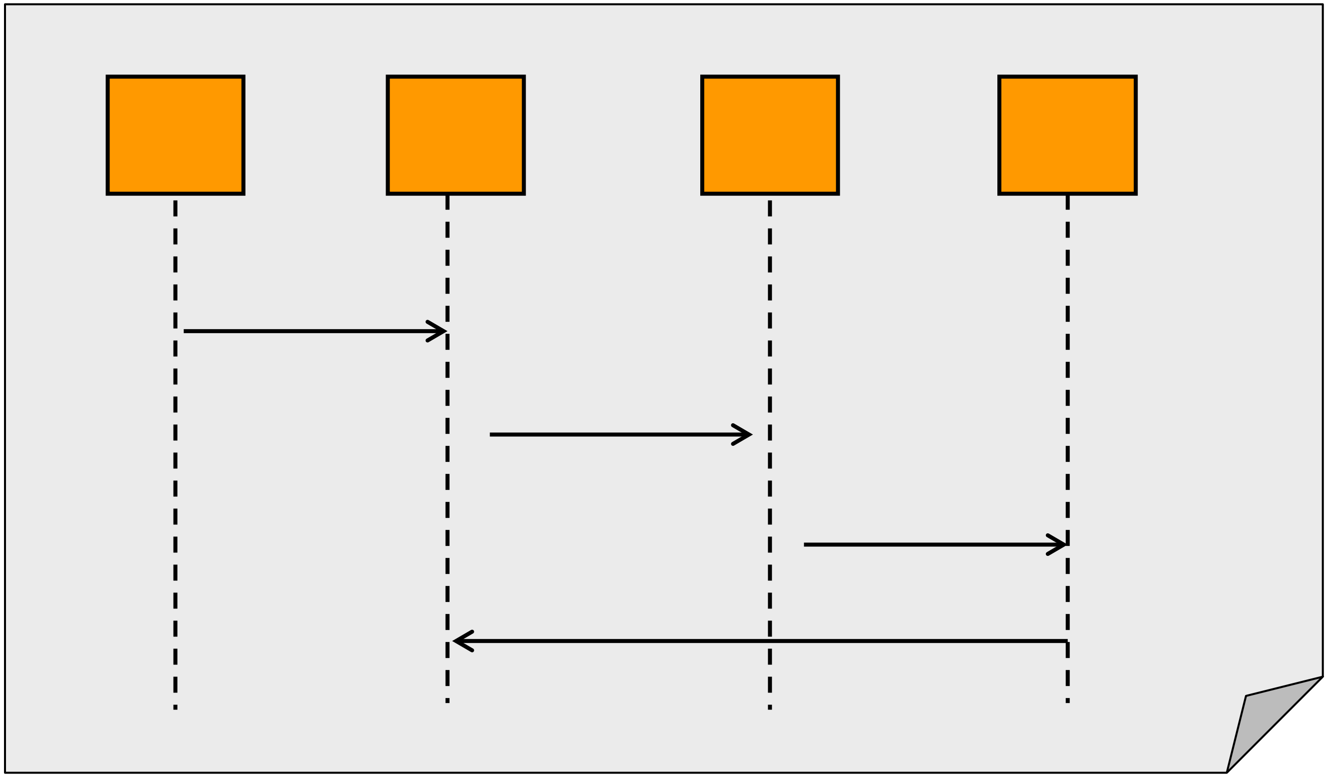

UPS Port @ SIRC =1
->SH_READY (not o.k.) from ANG
ATS switches Power Source for PUN A
SH_UPS (on) from ANG
Reboot PUN A with current limitation
UPS Port @ SIRC =1
->ATS switches Power Source for PUN A
SH_UPS (on) from ANG
Reboot PUN A with current limitation
ST_MODE (Bypass)
or
TURN_OFF from SYC
Bypass Mode
+
UPS
ST_PARAM
SYC distributes
Fl. Prg. only
-> Acq not possible
Normal Mode
Bypass Mode
ST_MODE (Normal)
from SYC
UPS Mode
UPS Port @ SIRC =0
->SH_READY (o.k.) from ANG
ATS switches Power Source for PUN A
Reboot PUN A without current limitation



Sequence diagrams

Used to show specific scenarios

(examples of collaboration between architectural elements/blocks)

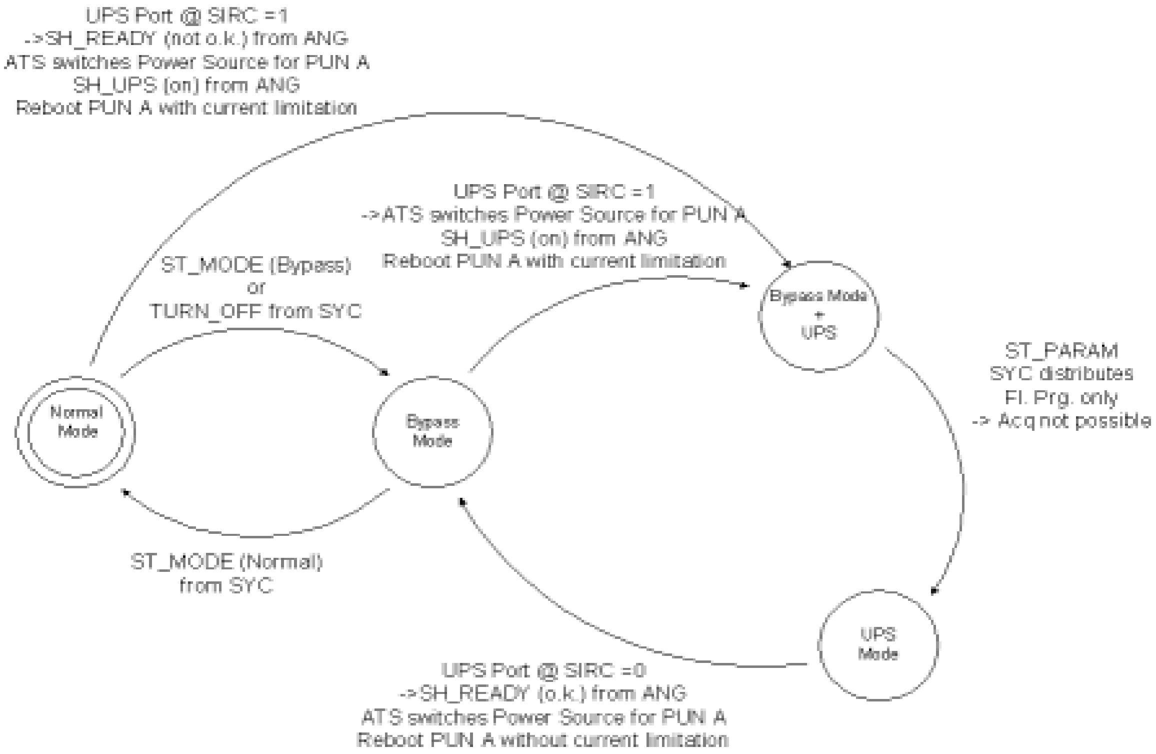


State charts

Useful for reactive systems

whenever it is not possible to define “linear” behaviour

when “modes” are important to determine behaviour


UPS Port @ SIRC =1
->SH_READY (not o.k.) from ANG
ATS switches Power Source for PUN A
SH_UPS (on) from ANG
Reboot PUN A with current limitation
UPS Port @ SIRC =1
->ATS switches Power Source for PUN A
SH_UPS (on) from ANG
Reboot PUN A with current limitation
ST_MODE (Bypass)
or
TURN_OFF from SYC
Bypass Mode
+
UPS
ST_PARAM
SYC distributes
Fl. Prg. only
-> Acq not possible
Normal
Mode
Bypass
Mode
ST_MODE (Normal)
from SYC
UPS
Mode
UPS Port @ SIRC =0
->SH_READY (o.k.) from ANG
ATS switches Power Source for PUN A
Reboot PUN A without current limitation

Deployment view

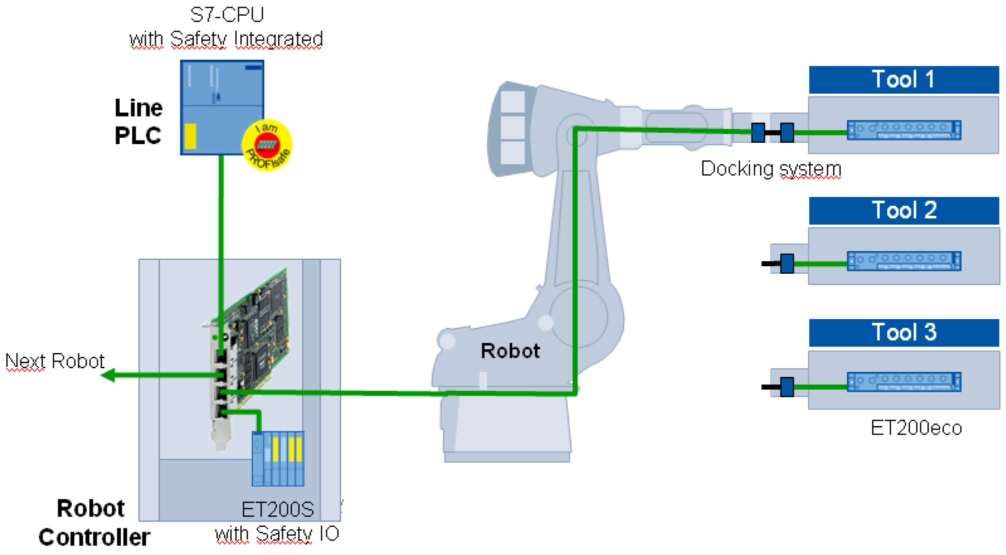
A decomposition of the system into physical building blocks

Shows:

- Physical Building Blocks that can execute (instances of) the logical elements
- Channels: (physical) connections between blocks

Note:

Channels can be electronic, optic, mechanic, pneumatic, etc.



Specification of a (physical) building block

Name

Description

(intended use; overview of its capabilities and capacities)

optional: *Design Justification*

(statement why you think the node will do what you say it will, and how it will meet the requirements allocated to it)

Allocated logical elements

(list of elements from the logical view, allocated to this node)

Specification of the interfaces

(mechanical, electrical, electronic, … interfaces)

Requirements

(allocated from system requirements + newly derived, decomposed and detailed requirements; e.g. required throughput, availability, maximum costs, ….)

Specification of a channel

Name

Description

(intended use; what it is supposed to do, what it is good for)

Characteristics:

(its capabilities and constraints;
often just a reference to an industry standard)

Requirements

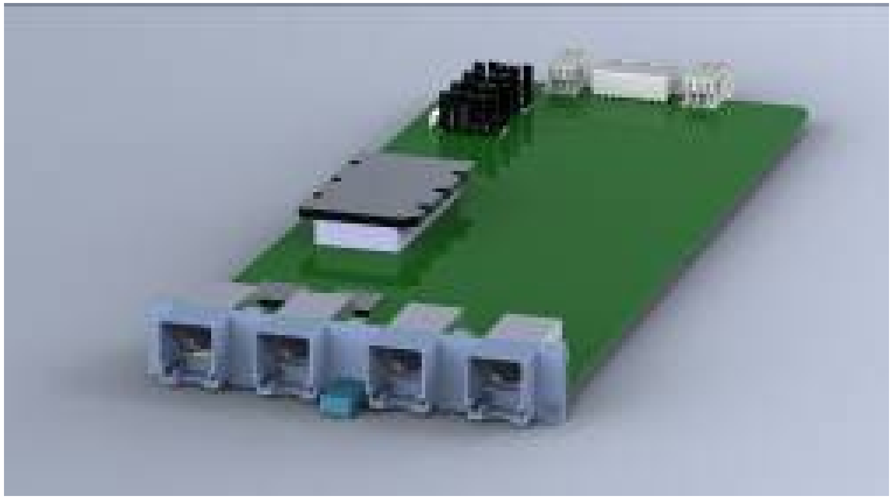
(allocated from system requirements + newly derived, decomposed and detailed requirements; e.g. required throughput, reliability, service requirements)

Spatial view

Shows the spatial relationships between physical elements

Examples:

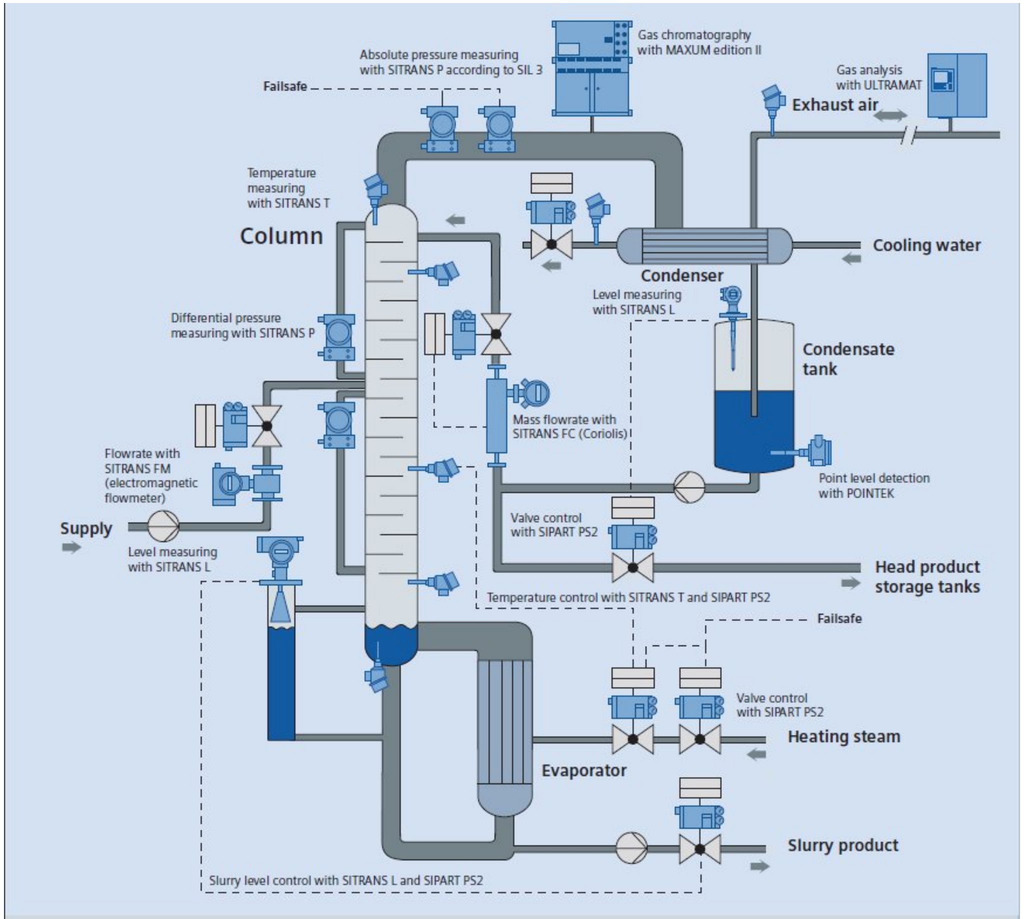
- Mechanical drawings
- 3-D Models
- Assembly instructions



Detail view (Example)

Mechanical sub-systems are usually described according to:

- Energy transfer
- Material transfer
- Signals


Gas chromatography with MAXUM edition II
Absolute pressure measuring with SITRANS P according to SIL 3
Failsafe
Gas analysis with ULTRAMAT
Exhaust air
Temperature measuring with SITRANS T
Column
Cooling water
Condenser
Level measuring with SITRANS L
Differential pressure measuring with SITRANS P
Condensate tank
Mass flowrate with SITRANS FC (Coriolis)
Flowrate with SITRANS FM (electromagnetic flowmeter)
Point level detection with POINTEK
Supply
Valve control with SIPART PS2
Level measuring with SITRANS L
Head product storage tanks
Temperature control with SITRANS T and SIPART PS2
Failsafe
Valve control with SIPART PS2
Heating steam
Evaporator
Slurry product
Slurry level control with SITRANS L and SIPART PS2

Scheme of a multi-level distillation process

Test architect’s stakes in architecture documentation

TeA is an important stakeholder in the software / system architecture documentation

- You need to ***read it***
- You need to ***understand it***
- You need to ***review it***

Most decisions of the test architecture are rooted in the software / system architecture

Test system requires product quality: ***you need to document***

- **Architecturally significant requirements**
- **Decisions** and **rationale**
- **Architecture** of the test system, i.e. the **test architecture**

Other methods used for documentation, e.g.

- Additional views beyond *Kruchten 4+1*
- Modeling: UML, SysML, …

SIEMENS
Ingenuity for life

Exercise

Sketch the system architecture,
i.e. the architecture of your system under test

- Split up in pairs (***A***, ***B***) of participants with similar context:
  - Type system: Software only / System / Solution
  - Domain: Industry, Energy, Healthcare,
  - …
- ***A***: Explain to your partner ***B*** the most important aspects of your system architecture
- ***B***: As a test architect for this system, ensure that **A** covers all information that is important for your test system
- ***Both***: document in an appropriate way on flipchart
  - How do you document these aspects? Why?
  - Which views do you select? Why?
  - Which diagrams are most important? Why?

Exercise:
System Architecture

Pairs of participants, 20min

Architectural Views & Documentation

Agenda

(Software) Architecture Documentation

Test Architecture Documentation

Test Documentation

Summary

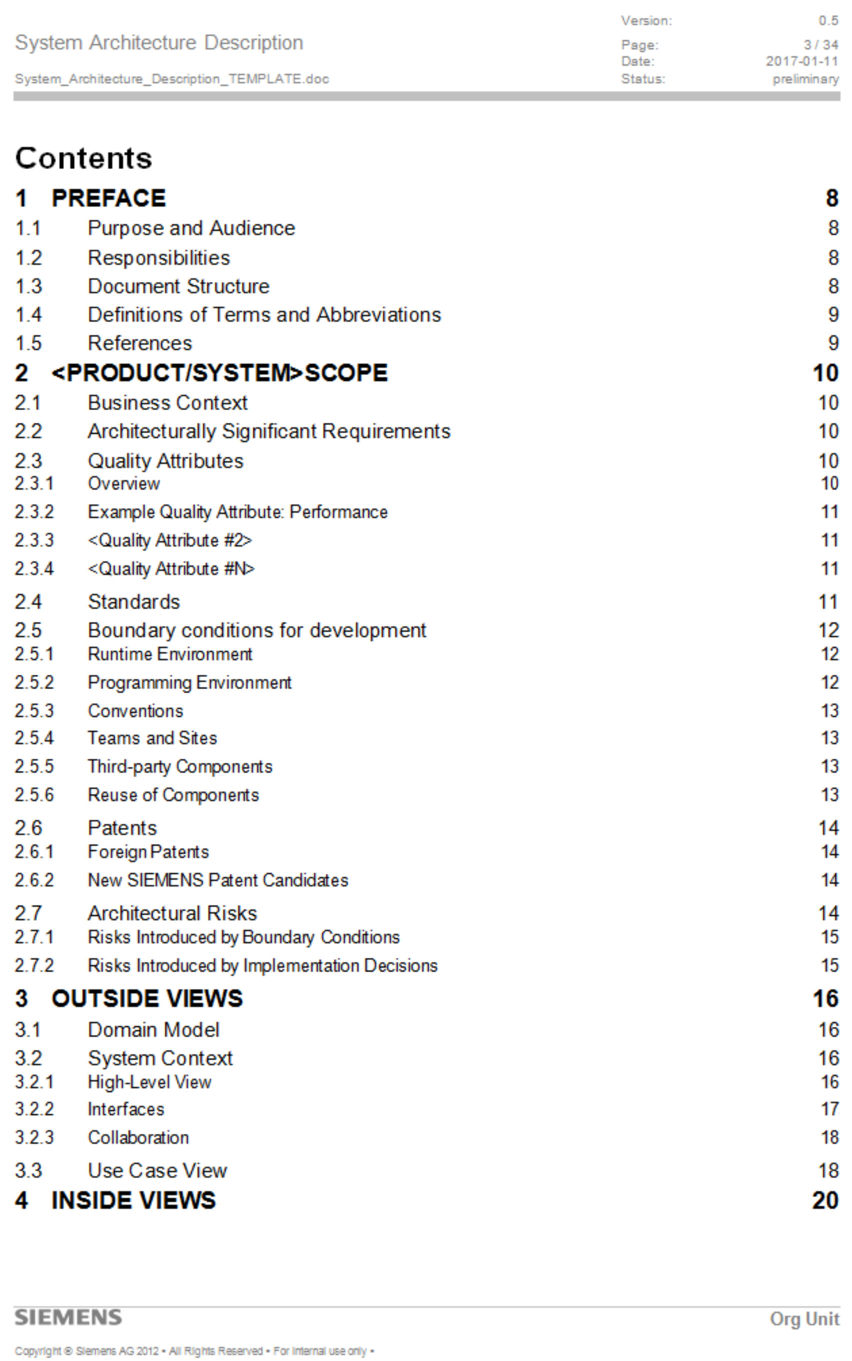
SIEMENS
Ingenuity for life

Exercise

Documenting test architecture – Group work

- Put yourself in the shoes of different **Test Architecture** stakeholders (one per group):
  - Test (Automation) Engineer, Test Designer
  - Test Manager, Head of System Test
  - Head of R&D, Portfolio/Product Manager
  - Software/System Architect, Developer
  - Customer
  - New team member
  - …

- What are their typical concerns?
- What information will they expect to find in your **Test Architecture Specification**?
- What would you provide in your **Test Architecture Specification** to satisfy the stakeholders' information needs?



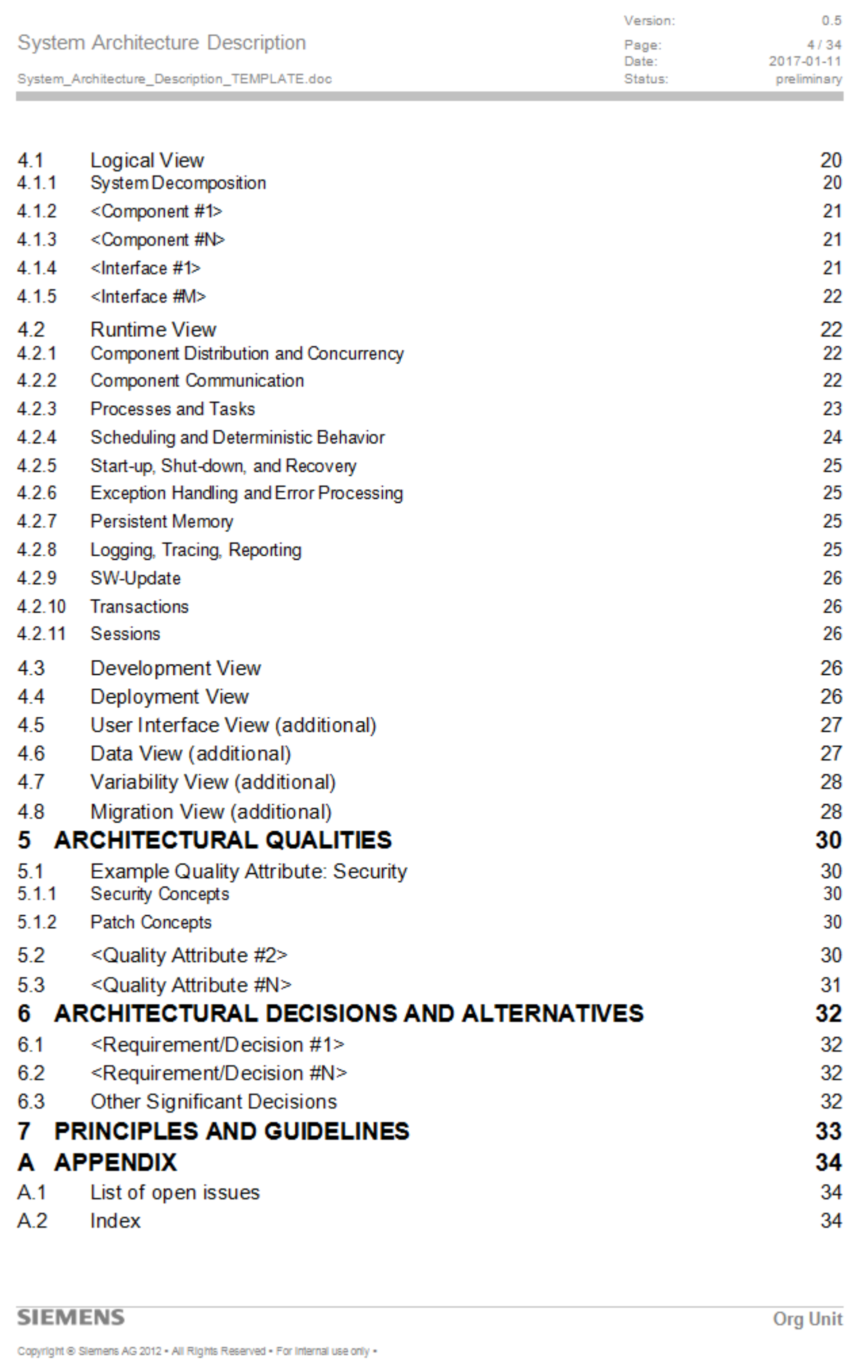
System Architecture Description

System_Architecture_Description_TEMPLATE.doc

Version:	0.5
Page:	3 / 34
Date:	2017-01-11
Status:	preliminary

Contents

SIEMENS Org Unit



System Architecture Description

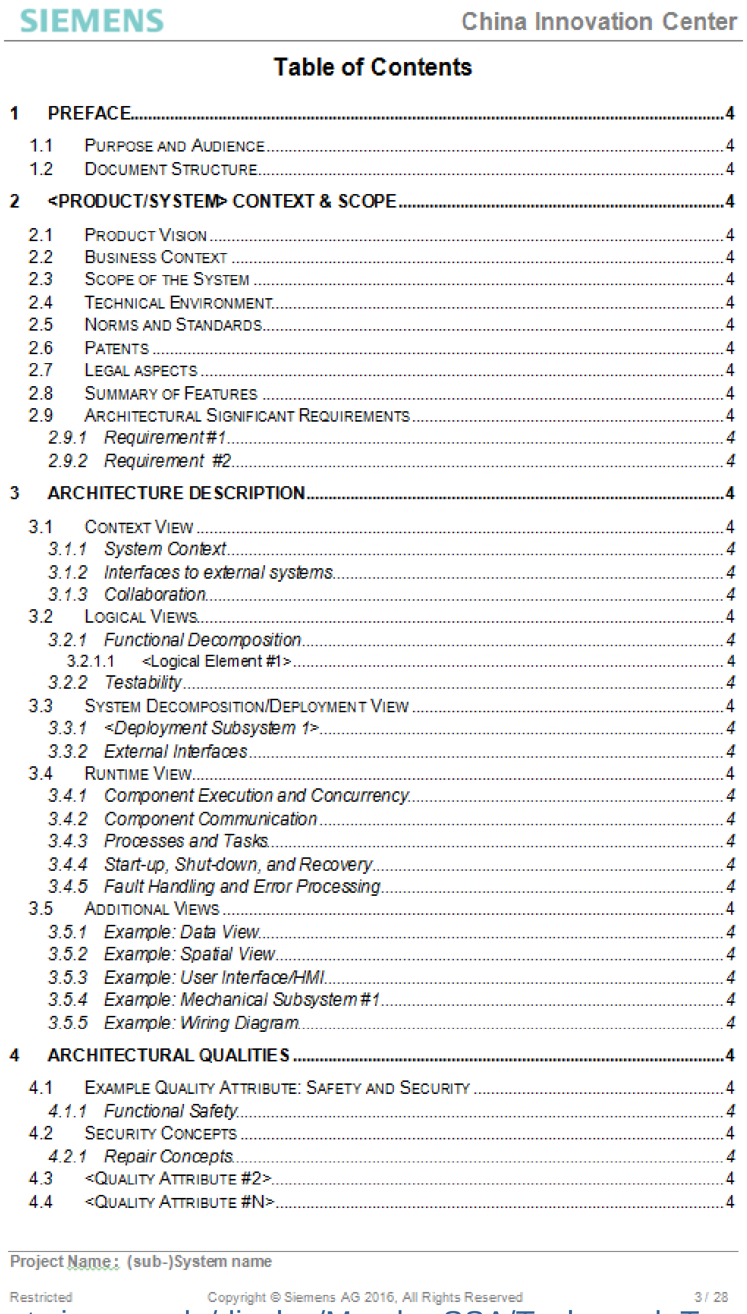
System_Architecture_Description_TEMPLATE.doc

Version:	0.5
Page:	4 / 34
Date:	2017-01-11
Status:	preliminary

SIEMENS Org Unit

https://wiki.ct.siemens.de/display/MemberSSA/Tools+and+Templates

System / Software Architecture Description

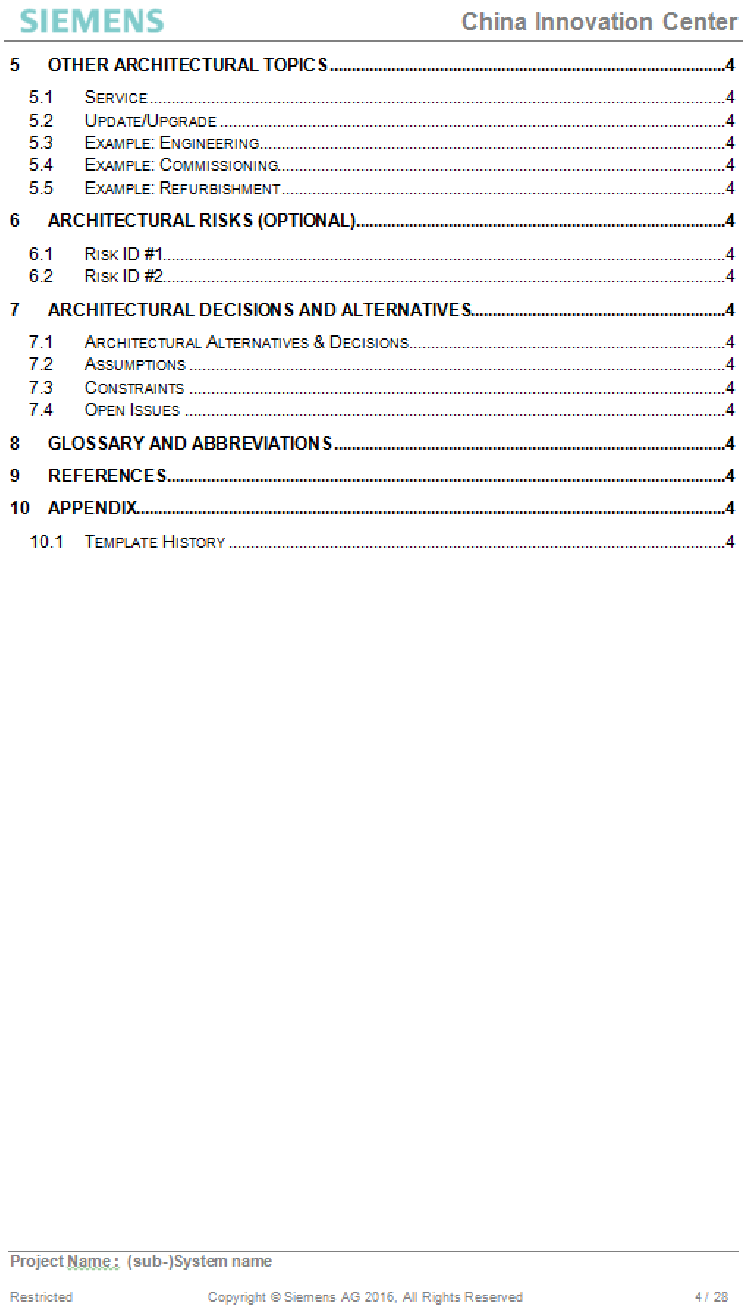


SIEMENS China Innovation Center

Table of Contents

Project Name: (sub-)System name

Restricted Copyright © Siemens AG 2016, All Rights Reserved 3 / 28



SIEMENS China Innovation Center

Project Name: (sub-)System name

Restricted Copyright © Siemens AG 2016, All Rights Reserved 4 / 28

https://wiki.ct.siemens.de/display/MemberSSA/Tools+and+Templates

Review of the test architecture

Typical reviewers

- Software/System Architect
- Product Owner
- Test Engineer
- Test Manager
- Developers
- …

Review topics could be, e.g.

- *Covering the important product NFRs and risks?*
- *Is the defined test environment adequate?*
- *Test system usability for stakeholder: tester*
- *Test system usability for stakeholder: developer (e.g. unit testing)*
- *Supporting test automation, test reporting (dashboards)?*
- *Compatibility with incremental, iterative, agile, lean, … development process?*
- *…*

Architectural Views & Documentation

Agenda

(Software) Architecture Documentation

Test Architecture Documentation

Test Documentation

Summary

Test documentation

ISO/IEC/IEEE 24765 and ISO/IEC/IEEE 25051

Test documentation is the

- documentation describing plans for, or results of, the testing of a system or component
- collection of the documentation inherent to the testing activities

Les Chambers (http://www.chambers.com.au/glossary/test_documentation.php)

Test documentation is the complete suite of artifacts that describe test planning, test design, test execution, test results and conclusions drawn from the testing activity

ISTQB Glossary (http://www.istqb.org/)

horizontal traceability – The tracing of requirements for a test level through the layers of **test documentation** (e.g., test plan, test design specification, test case specification and test procedure specification or test script)

Standards for test documentation:

- ISO/IEC/IEEE 29119-3 (published 2013)
- IEEE Standard 829 (published 1998 and 2008)
- ISO 9000-3 (published 1997)

ISO/IEC/IEEE 29119-3: Test documentation Overview

Organizational test process

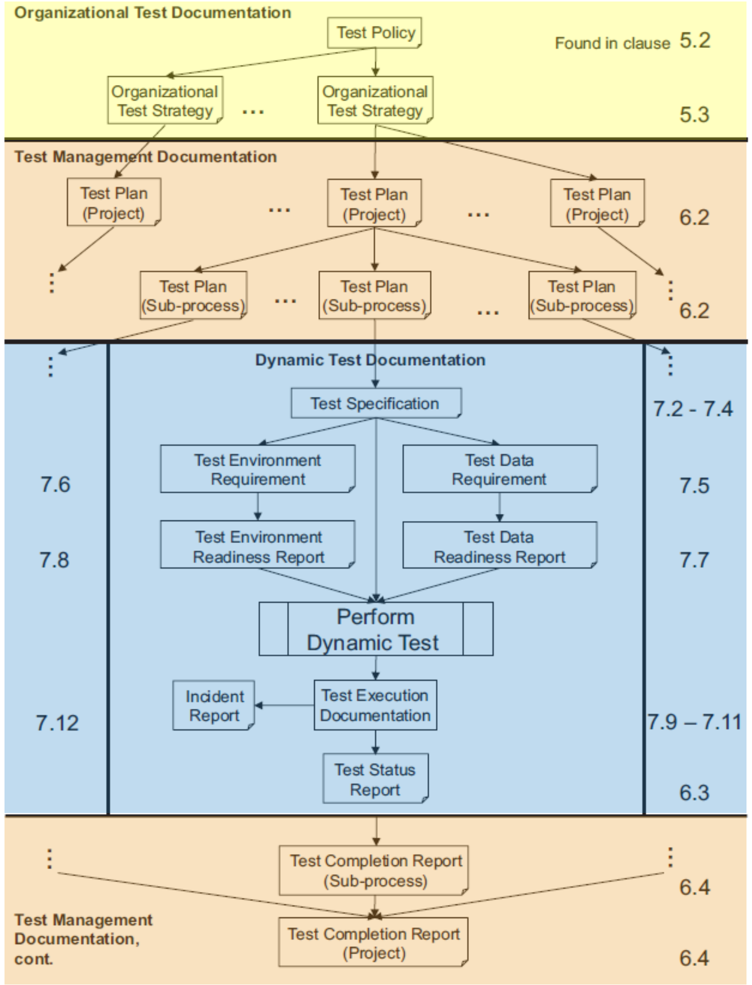
- Test policy
- Organizational test strategy

Test management processes

- Test plan
- Test completion report

Dynamic test processes

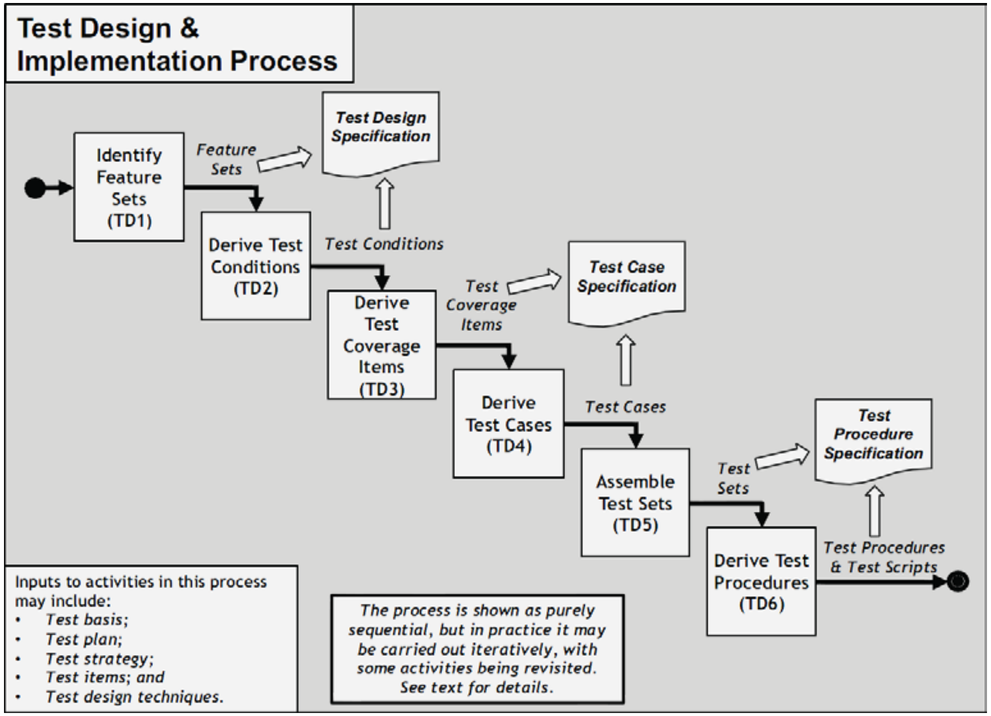
- Test plan
- Test specification
- Test data requirements
- Test data readiness report
- Test environment requirements
- Test environment readiness report
- Test execution log
- Incident report
- Test status report


Organizational Test Documentation
Test Policy
Found in clause 5.2
Organizational Test Strategy
...
Organizational Test Strategy
5.3
Test Management Documentation
Test Plan (Project)
...
Test Plan (Project)
...
Test Plan (Project)
6.2
Test Plan (Sub-process)
...
Test Plan (Sub-process)
...
Test Plan (Sub-process)
6.2
Dynamic Test Documentation
Test Specification
7.2 - 7.4
Test Environment Requirement
Test Data Requirement
7.6
7.5
Test Environment Readiness Report
Test Data Readiness Report
7.8
7.7
Perform Dynamic Test
Incident Report
Test Execution Documentation
7.12
7.9 – 7.11
Test Status Report
6.3
Test Completion Report (Sub-process)
6.4
Test Management Documentation, cont.
Test Completion Report (Project)
6.4

http://www.iso.org/

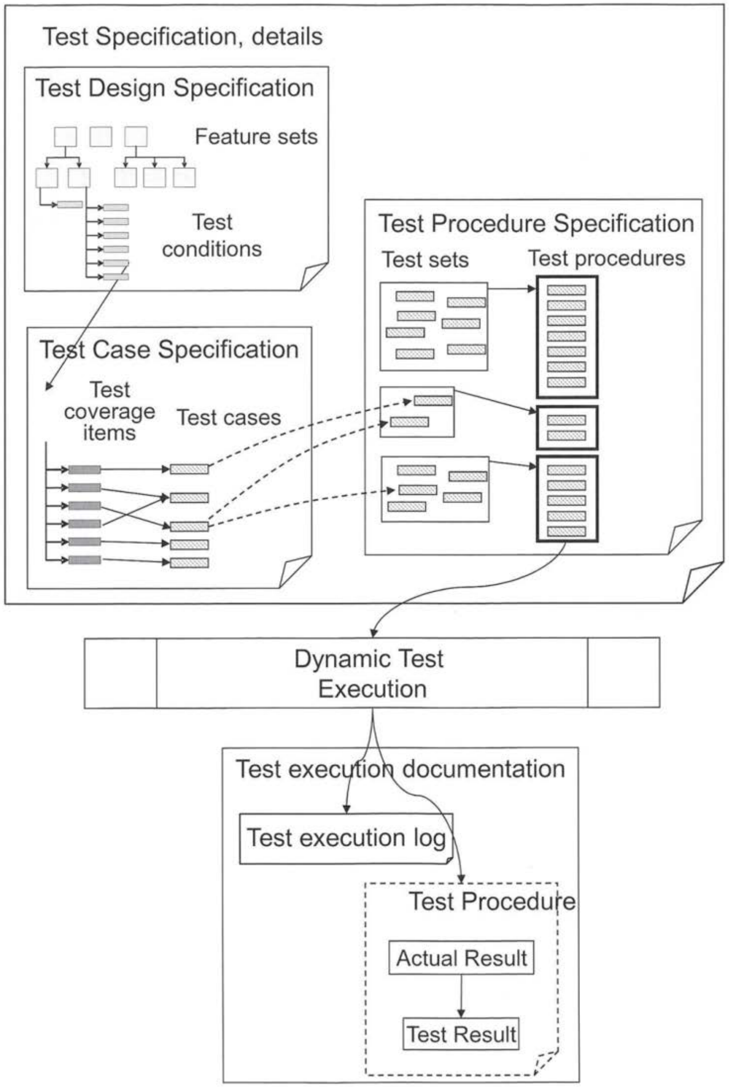
ISO/IEC/IEEE 29119-3: Test documentation Overview

The hierarchy between the documents produced in completing the ***Test Design and Implementation Process*** outlined in ISO/IEC/IEEE 29119-2


Test Design & Implementation Process
Identify Feature Sets (TD1)
Feature Sets
Test Design Specification
Derive Test Conditions (TD2)
Test Conditions
Derive Test Coverage Items (TD3)
Test Coverage Items
Test Case Specification
Derive Test Cases (TD4)
Test Cases
Assemble Test Sets (TD5)
Test Sets
Test Procedure Specification
Derive Test Procedures (TD6)
Test Procedures & Test Scripts

Inputs to activities in this process may include:
- *Test basis;*
- *Test plan;*
- *Test strategy;*
- *Test items; and*
- *Test design techniques.*

The process is shown as purely sequential, but in practice it may be carried out iteratively, with some activities being revisited. See text for details.

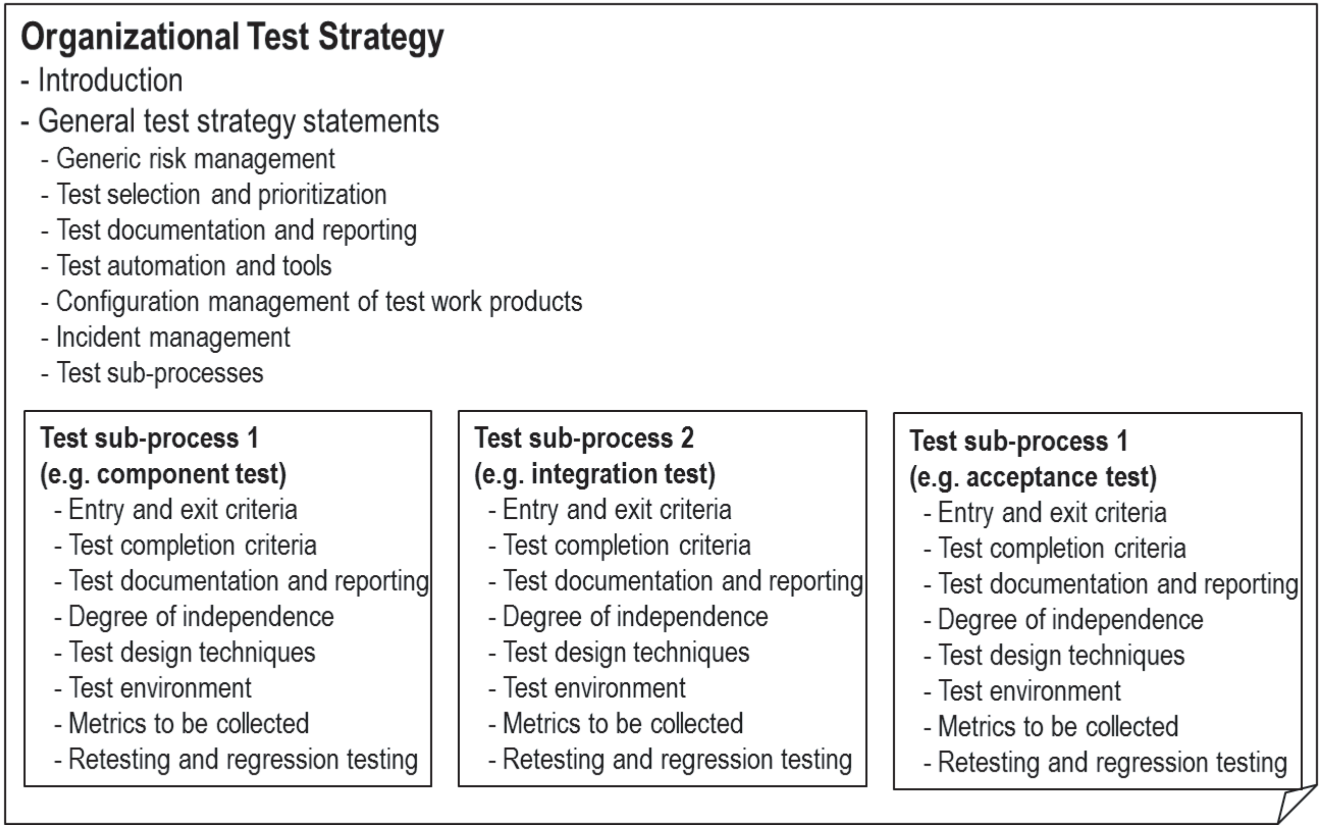


http://www.iso.org/

ISO/IEC/IEEE 29119-3: Test documentation
Example: Organizational Test Strategy

Organizational Test Strategy includes identification of relevant test sub-processes and strategy statements

Document may be partitioned with a sub-section for each of the identified test sub-processes if the test sub-processes strategy statements differ significantly



Organizational Test Strategy

- Introduction
- General test strategy statements
  - Generic risk management
  - Test selection and prioritization
  - Test documentation and reporting
  - Test automation and tools
  - Configuration management of test work products
  - Incident management
  - Test sub-processes

Test sub-process 1
(e.g. component test)

- Entry and exit criteria
- Test completion criteria
- Test documentation and reporting
- Degree of independence
- Test design techniques
- Test environment
- Metrics to be collected
- Retesting and regression testing

Test sub-process 2
(e.g. integration test)

- Entry and exit criteria
- Test completion criteria
- Test documentation and reporting
- Degree of independence
- Test design techniques
- Test environment
- Metrics to be collected
- Retesting and regression testing

Test sub-process 1
(e.g. acceptance test)

- Entry and exit criteria
- Test completion criteria
- Test documentation and reporting
- Degree of independence
- Test design techniques
- Test environment
- Metrics to be collected
- Retesting and regression testing

http://www.iso.org/

Implementation of a standard

"Once you understand your needs, you can invest to meet them, rather than investing to meet a standard that may or may not serve your needs."

Cem Kaner

Prescriptive standards and templates that encourage people to do the same thing in different contexts have led to generate huge test documentation sets

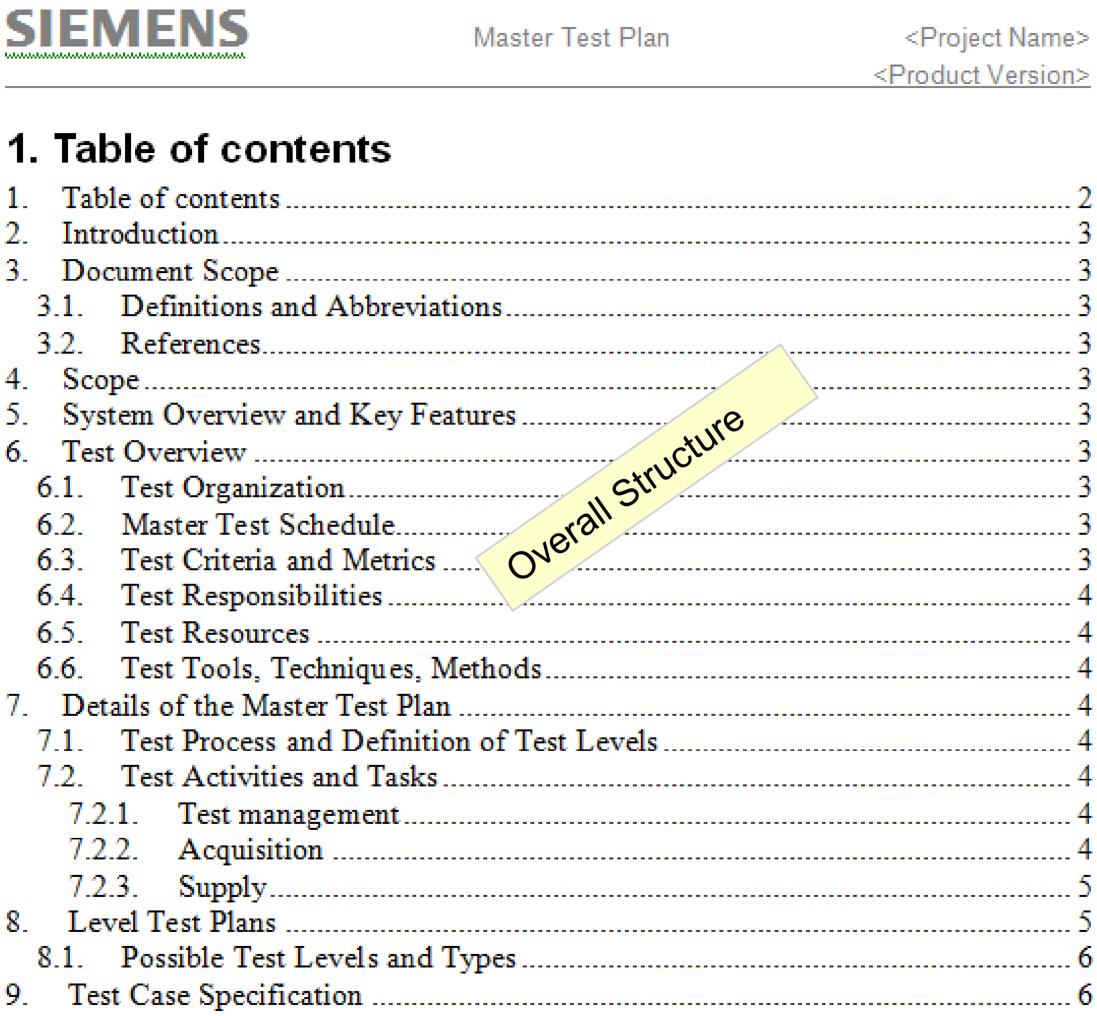
Implementation of the standard typically involves selecting a set of documents *suitable* for the organization or project

Requirements for test documentation should be defined in terms of

- **Stakeholders**: Who would use or be affected by test documentation?
- **Interests**: What interests of theirs does documentation serve or disserve?
- **Actions**: What will they do with the documentation?
- **Objects**: What types of documents are of high or low value?

In parts based on materials provided by Cem Kaner and James Bach

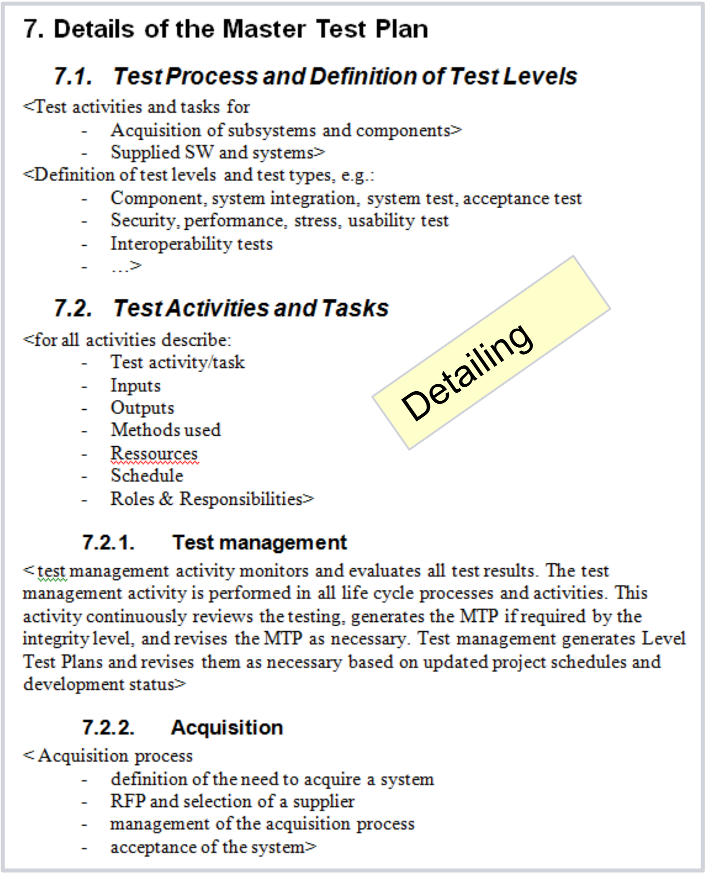
Example: Project Test Plan Template



SIEMENS Master Test Plan <Project Name>
<Product Version>

1. Table of contents

Overall Structure



7. Details of the Master Test Plan

7.1. Test Process and Definition of Test Levels

<Test activities and tasks for
- Acquisition of subsystems and components>
- Supplied SW and systems>

<Definition of test levels and test types, e.g.:
- Component, system integration, system test, acceptance test
- Security, performance, stress, usability test
- Interoperability tests
- ...>

7.2. Test Activities and Tasks

<for all activities describe:
- Test activity/task
- Inputs
- Outputs
- Methods used
- Ressources
- Schedule
- Roles & Responsibilities>

7.2.1. Test management

<test management activity monitors and evaluates all test results. The test management activity is performed in all life cycle processes and activities. This activity continuously reviews the testing, generates the MTP if required by the integrity level, and revises the MTP as necessary. Test management generates Level Test Plans and revises them as necessary based on updated project schedules and development status>

7.2.2. Acquisition

< Acquisition process
- definition of the need to acquire a system
- RFP and selection of a supplier
- management of the acquisition process
- acceptance of the system>

Detailing

Example: Test Concept for <XYZ> Template

1. Scope
- 1.1 Test basis
- 1.2 Open issues and topics to be clarified

2. Stakeholders

3. Test goals
- 3.1 Goals and responsibilities of testing
- 3.2 Assumptions, dependencies, and constraints

4. Test scope
- 4.1 Test objects, test items, system under test (SUT)
- 4.2 Major product risks (risk-based testing)
- 4.3 Features to be tested
- 4.4 Features not to be tested
- 4.5 Non-functional requirements (NFRs) to be tested
- 4.6 Non-functional requirements (NFRs) not to be tested
- 4.7 Test data requirements

5. Test strategy and approach
- 5.1 Test levels, entry/exit criteria
- 5.2 Test types
- 5.3 Test techniques
- 5.4 Test data
- 5.5 Test deliverables and documentation
- 5.6 Defect and change management

6. Test environment
- 6.1 Hardware / Software / Operational environment for testing
- 6.2 Test configurations and setup
- 6.3 Deployments and releases

7. Test automation
- 7.1 Development environment, CI
- 7.2 Frameworks
- 7.3 Test tools
- 7.4 Test scripts

8. Test logistics
- 8.1 Schedule, timeline
- 8.2 Staffing

Regulatory compliance

Systems in specific domains (e.g. medical, rail, …) are subject to official regulations
Corresponding standards specify how test activities have to be documented

Food and Drug Administration (FDA)

FDA guidelines highlight four types of testing activities that need to be supported:

- Structural testing
- Functional testing
- Statistical testing
- Regression testing

Testing activities required:

- Unit testing
- Integration testing
- System testing
- Acceptance testing

Documents required during the testing process include:

- Test plan
- Test procedures
- Test cases
- Test reports
- Test logs

Standards for Railway Applications

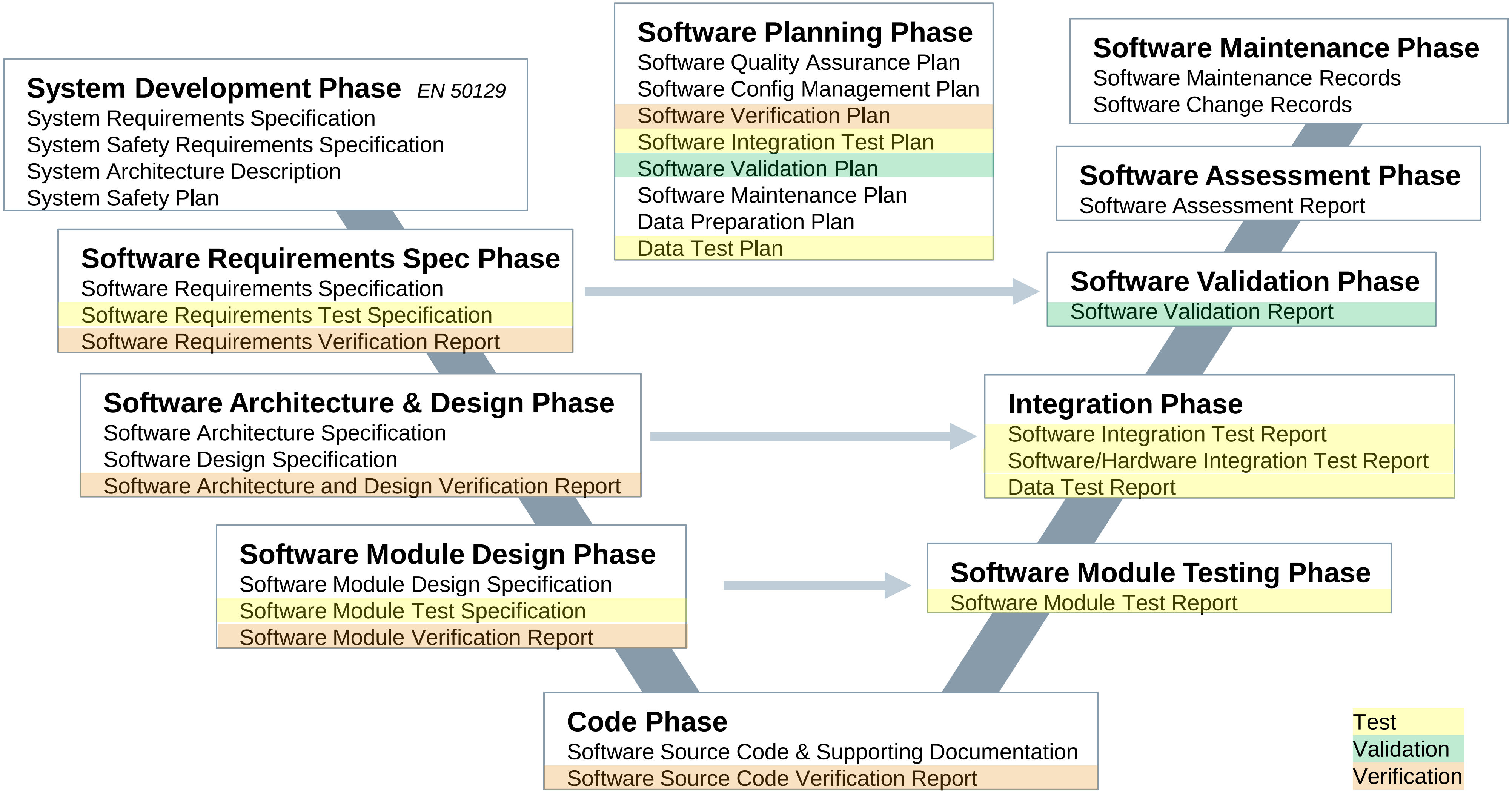
European Committee for Electrotechnical Standardization, *CENELEC*, has produced a number of standards for railway applications.

Testing activities and according documents required depend on Safety Integrity Level (SIL).

Documents for testing indicated by EN 50128 (Software for railway control and protection systems):

- Requirements Test Specification
- Module Test Specification, Module Test Report
- Verification Plan
- Verification Reports (Requirements, Architecture & Design, Module, Source Code)
- Integration Test Plan, Integration Test Report
- Validation Plan, Validation Report
- Data Test Plan, Data Test Report

EN 50128: Documents in the lifecycle



System Development Phase *EN 50129*
System Requirements Specification
System Safety Requirements Specification
System Architecture Description
System Safety Plan

Software Requirements Spec Phase
Software Requirements Specification
Software Requirements Test Specification
Software Requirements Verification Report

Software Architecture & Design Phase
Software Architecture Specification
Software Design Specification
Software Architecture and Design Verification Report

Software Module Design Phase
Software Module Design Specification
Software Module Test Specification
Software Module Verification Report

Code Phase
Software Source Code & Supporting Documentation
Software Source Code Verification Report

Software Module Testing Phase
Software Module Test Report

Integration Phase
Software Integration Test Report
Software/Hardware Integration Test Report
Data Test Report

Software Validation Phase
Software Validation Report

Software Assessment Phase
Software Assessment Report

Software Maintenance Phase
Software Maintenance Records
Software Change Records

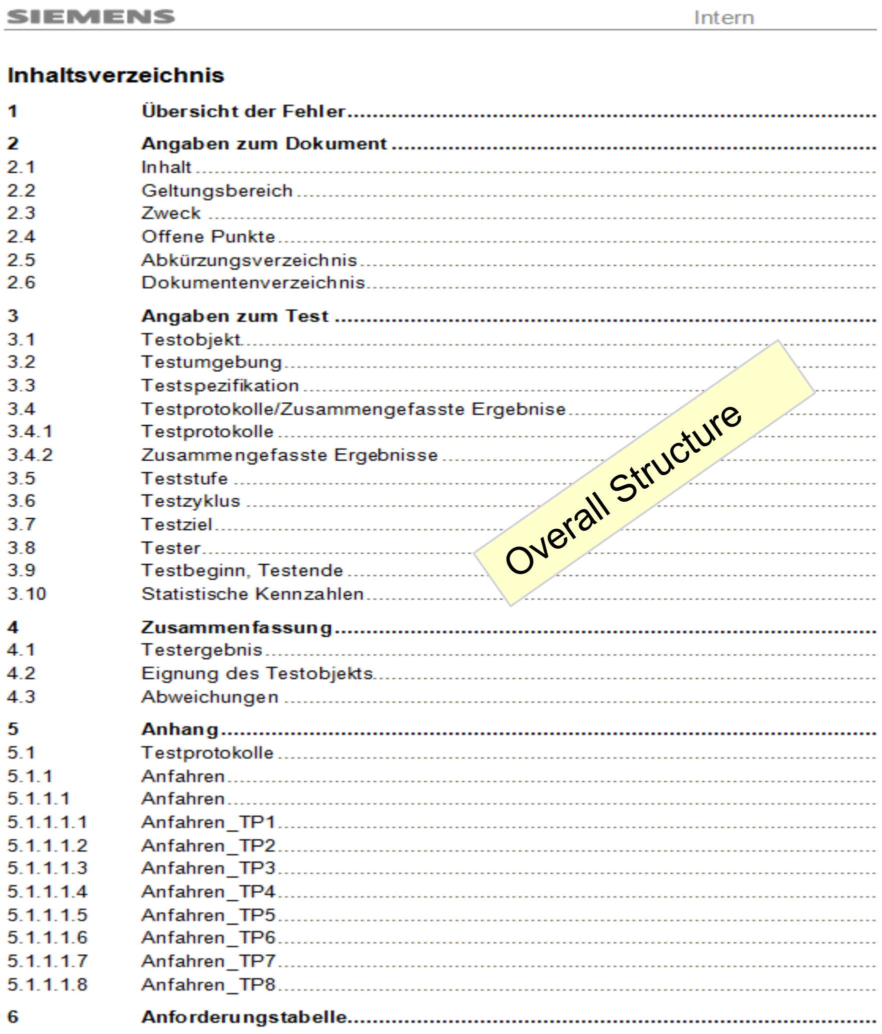
Software Planning Phase
Software Quality Assurance Plan
Software Config Management Plan
Software Verification Plan
Software Integration Test Plan
Software Validation Plan
Software Maintenance Plan
Data Preparation Plan
Data Test Plan

Test
Validation
Verification

Example: Integration Test Report

ICx TrainSIM

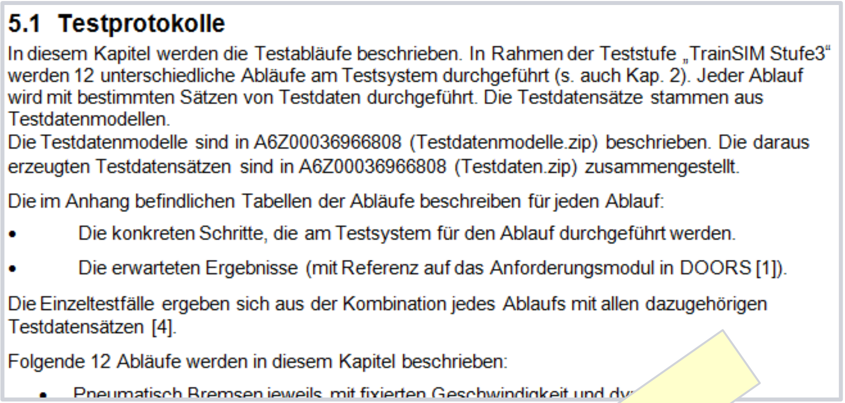
SIEMENS
Ingenuity for life



SIEMENS Intern

Inhaltsverzeichnis

Overall Structure



5.1 Testprotokolle

In diesem Kapitel werden die Testabläufe beschrieben. In Rahmen der Teststufe „TrainSIM Stufe3" werden 12 unterschiedliche Abläufe am Testsystem durchgeführt (s. auch Kap. 2). Jeder Ablauf wird mit bestimmten Sätzen von Testdaten durchgeführt. Die Testdatensätze stammen aus Testdatenmodellen.
Die Testdatenmodelle sind in A6Z00036966808 (Testdatenmodelle.zip) beschrieben. Die daraus erzeugten Testdatensätzen sind in A6Z00036966808 (Testdaten.zip) zusammengestellt.

Die im Anhang befindlichen Tabellen der Abläufe beschreiben für jeden Ablauf:

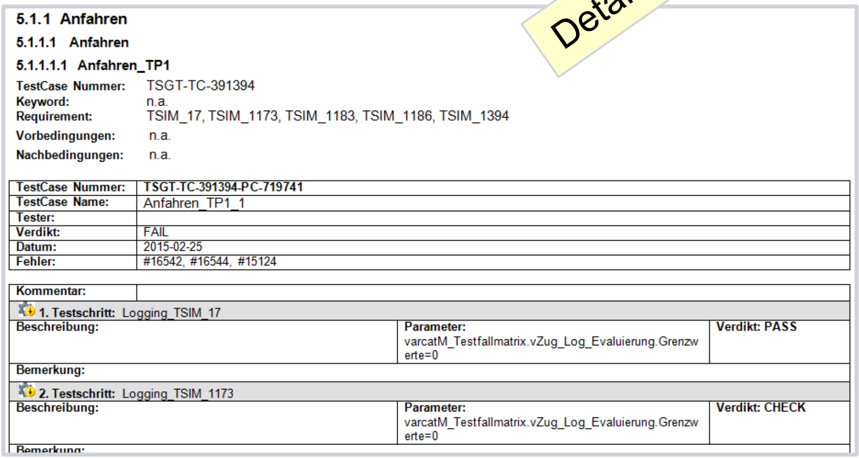
- Die konkreten Schritte, die am Testsystem für den Ablauf durchgeführt werden.
- Die erwarteten Ergebnisse (mit Referenz auf das Anforderungsmodul in DOORS [1]).

Die Einzeltestfälle ergeben sich aus der Kombination jedes Ablaufs mit allen dazugehörigen Testdatensätzen [4].

Folgende 12 Abläufe werden in diesem Kapitel beschrieben:

- Pneumatisch Bremsen jeweils mit fixierten Geschwindigkeit und dy

Details



5.1.1 Anfahren

5.1.1.1 Anfahren

5.1.1.1.1 Anfahren_TP1

TestCase Nummer:	TSGT-TC-391394
Keyword:	n.a.
Requirement:	TSIM_17, TSIM_1173, TSIM_1183, TSIM_1186, TSIM_1394
Vorbedingungen:	n.a.
Nachbedingungen:	n.a.

TestCase Nummer:	**TSGT-TC-391394-PC-719741**
TestCase Name:	Anfahren_TP1_1
Tester:	
Verdikt:	FAIL
Datum:	2015-02-25
Fehler:	#16542, #16544, #15124

Kommentar:		
1. Testschritt: Logging_TSIM_17		
Beschreibung:	**Parameter:** varcatM_Testfallmatrix.vZug_Log_Evaluierung.Grenzwerte=0	**Verdikt: PASS**
Bemerkung:		
2. Testschritt: Logging_TSIM_1173		
Beschreibung:	**Parameter:** varcatM_Testfallmatrix.vZug_Log_Evaluierung.Grenzwerte=0	**Verdikt: CHECK**
Bemerkung:		

Exercise

How do you document testing?

Discuss:

- What are the test documentation practices you are used to ?
- How does the selection of test documents in your organization / project look like?

- Test documents you currently use
- Important test documents, in your opinion

Architectural Views & Documentation

SIEMENS

Ingenuity for life

Agenda

(Software) Architecture Documentation

Test Architecture Documentation

Test Documentation

Summary

What we have learned

A view is a representation of a system from the perspective of a related set of concerns.

Views are for communication, not just documentation!

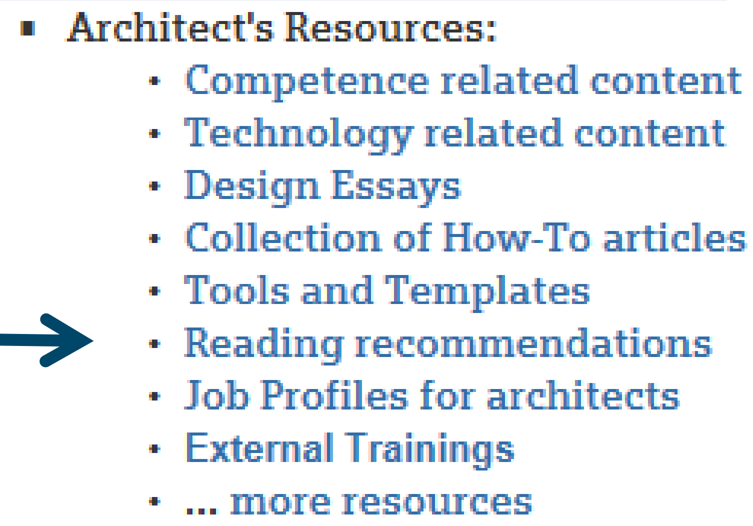
Documentation is part of the design work;
it‘s a continuous process – avoid doing it post-mortem!

Select the *right* set of documents for the organization and for the project when implementing a test documentation standard!

Further readings

Use the SSA Wiki :
https://wiki.ct.siemens.de/x/fReTBQ

and check the “Reading recommendations”:
https://wiki.ct.siemens.de/x/-pRgBg



- Architect's Resources:
  - Competence related content
  - Technology related content
  - Design Essays
  - Collection of How-To articles
  - Tools and Templates
  - Reading recommendations
  - Job Profiles for architects
  - External Trainings
  - ... more resources

Backup

Backup

Views in software engineering

Use case

Logical

Deployment

Runtime

Information

Implementation

Summary of the **most architecturally significant use cases** and their non-functional requirements

That are those use cases which by their implementation

- illustrate significant architectural coverage
- exercise many architectural elements

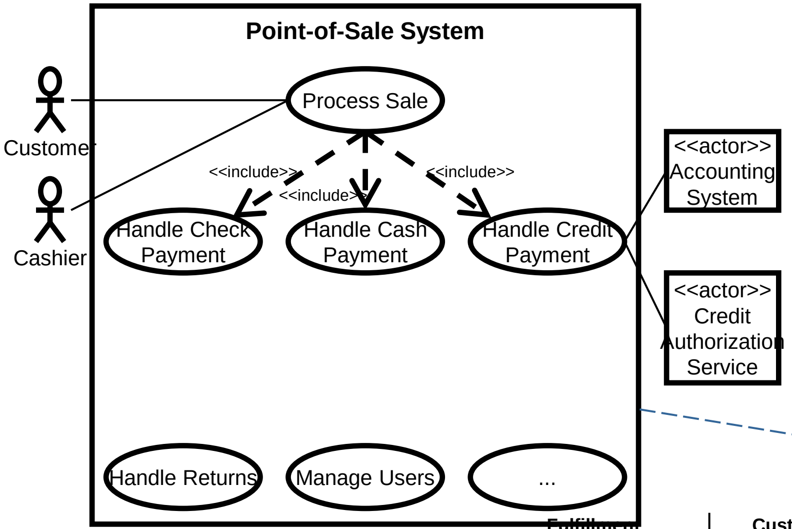
Might be expressed by

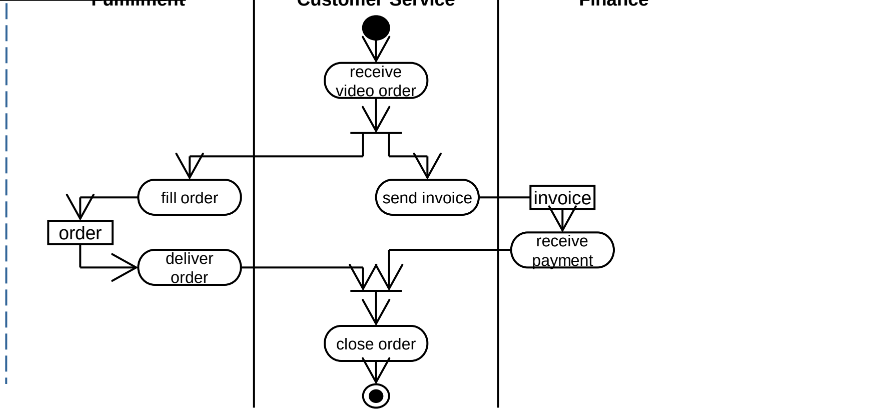
- expressed in **text**
- Overview and relations of use cases in UML use case diagrams
- *Perhaps* with use-case realizations in UML interaction diagrams

SIEMENS
Ingenuity for life

Views in software engineering

- **Use case**
- Logical
- Deployment
- Runtime
- Information
- Implementation





Views in software engineering (cont'd)

- Use case
- **Logical**
- Deployment
- Runtime
- Information
- Implementation

Conceptual organization of the software

- In terms of the most important layers, subsystems, packages, frameworks, classes, and interfaces

Summarizes the **functionality of the major software elements**

Shows **outstanding use case realization scenarios**

- as **UML interaction diagrams**
- that illustrate **key aspects of the system**

Is visualized with

- UML package diagrams
- UML class diagrams
- UML interaction diagrams

Views in software engineering (cont'd)

Use case

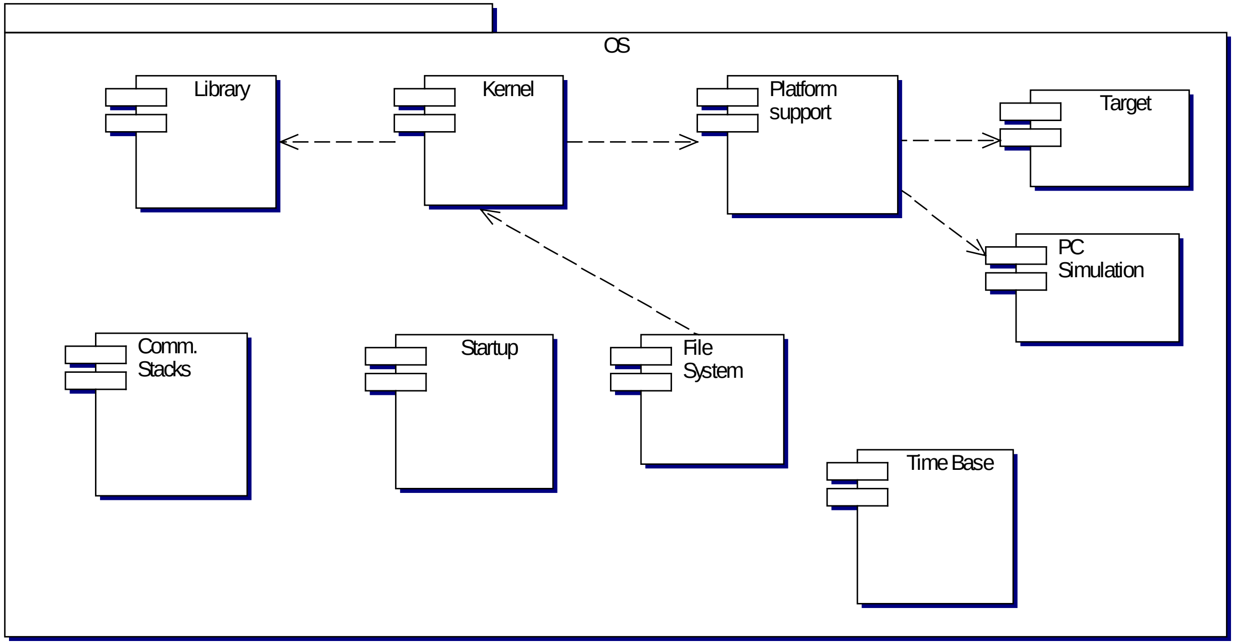
Logical

Deployment

Runtime

Information

Implementation



Views in software engineering (cont'd)

- Use case
- Logical
- **Deployment**
- Runtime
- Information
- Implementation

Shows the **physical deployment** of processes and components to processing nodes

Shows the **physical network configuration** between nodes

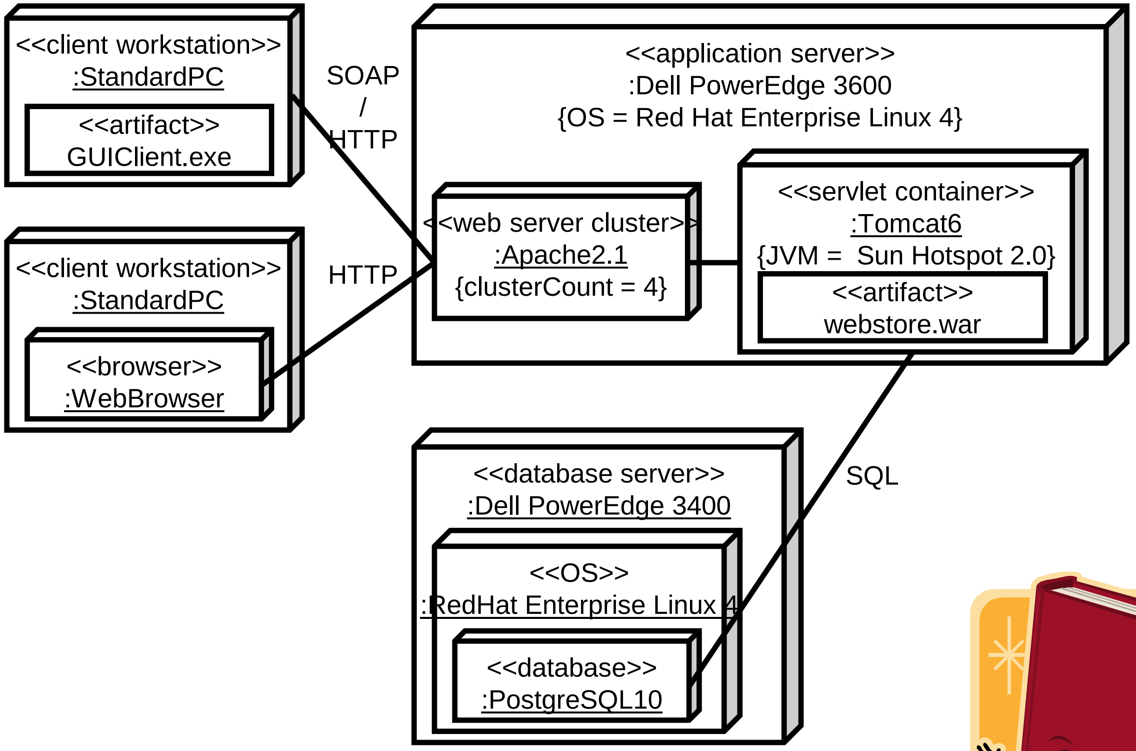
Normally, the deployment view is simply the **entire model** rather than a subset

Might be expressed by

- UML deployment diagrams

Views in software engineering (cont'd)

- Use case
- Logical
- **Deployment**
- Runtime
- Information
- Implementation



Views in software engineering (cont'd)

- Use case
- Logical
- Deployment
- **Runtime**
- Information
- Implementation

Shows dynamic aspects

- Subsystem Distribution and Concurrency
- Subsystem Communication
- Processes and Tasks
- Start-up, Shut-down, and Recovery
- Scheduling and Deterministic Behavior
- Exception Handling and Error Processing
- Logging, Tracing, Reporting

Might be expressed by

- UML interaction diagrams

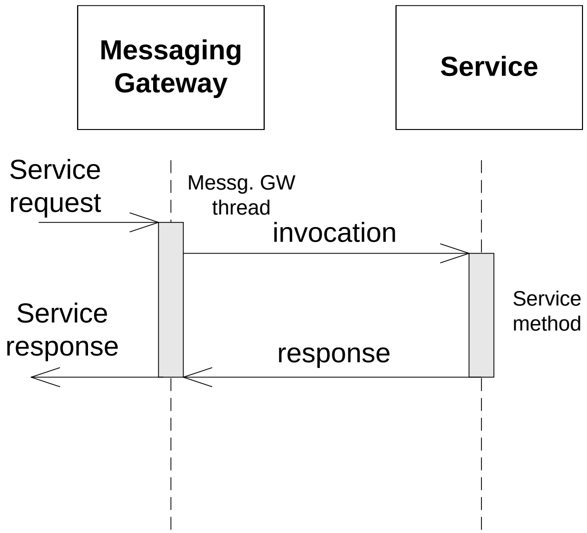
SIEMENS

Ingenuity for life

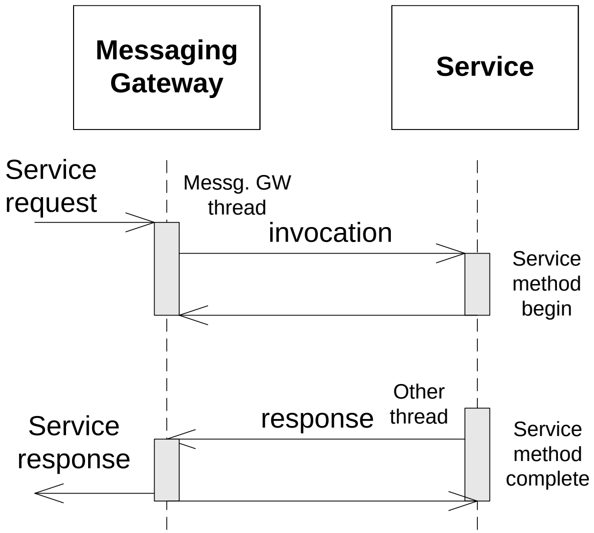
Views in software engineering (cont'd)

- Use case
- Logical
- Deployment
- **Runtime**
- Information
- Implementation

Synchronous invocation



Asynchronous invocation



Views in software engineering (cont'd)

This view is an extension to the 4+1 views by Kruchten

Use case

Logical

Deployment

Runtime

Information

Implementation

Shows the **data scheme**

- Object-oriented/Relational

Shows the scheme mapping **from objects to persistent data**

Shows an overview of the **data flows**

Might be expressed by

- UML class diagrams
- Data flows can be shown with UML activity diagrams

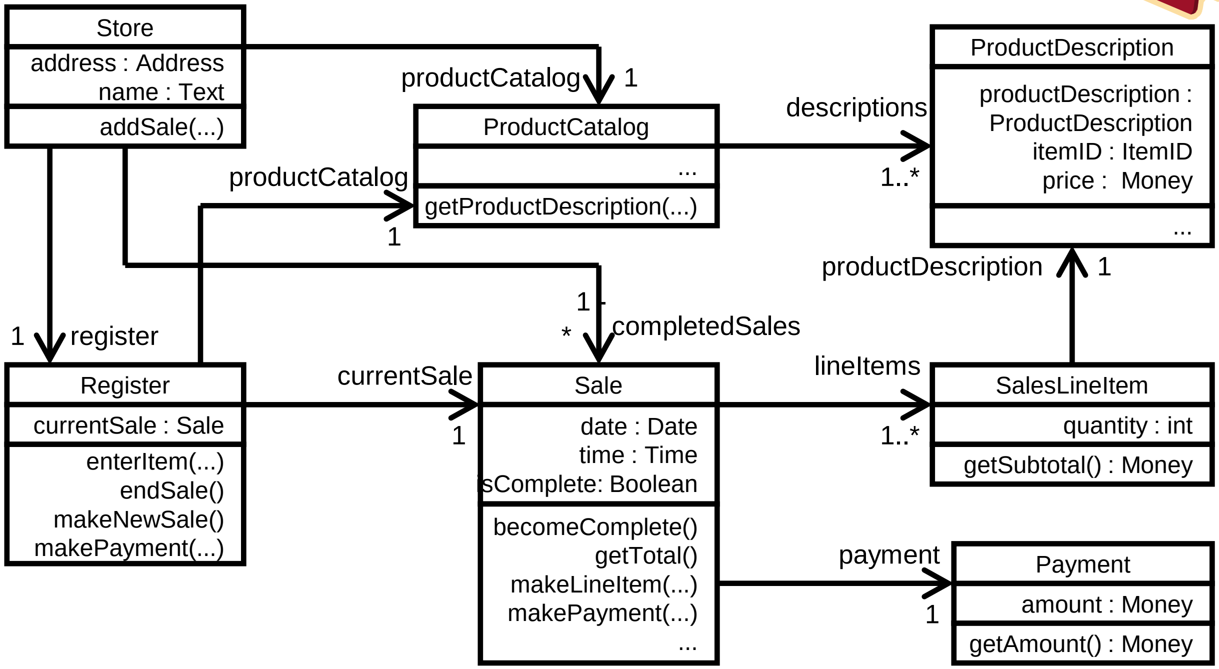
SIEMENS

Ingenuity for life

Views in software engineering (cont'd)

This view is an extension to the 4+1 views by Kruchten

- Use case
- Logical
- Deployment
- Runtime
- **Information**
- Implementation



Views in software engineering (cont'd)

Use case

Logical

Deployment

Runtime

Information

Implementation

The development view summarizes information that developers need to know about the **setup of the development environment**

- For example
  - How are all the files organized in terms of directories, and why ?
  - How does a test run ?
  - How is version control used ?

Is described

- As text