

# SYSTEMATIC ARCHITECTURE

---

What is architecture?

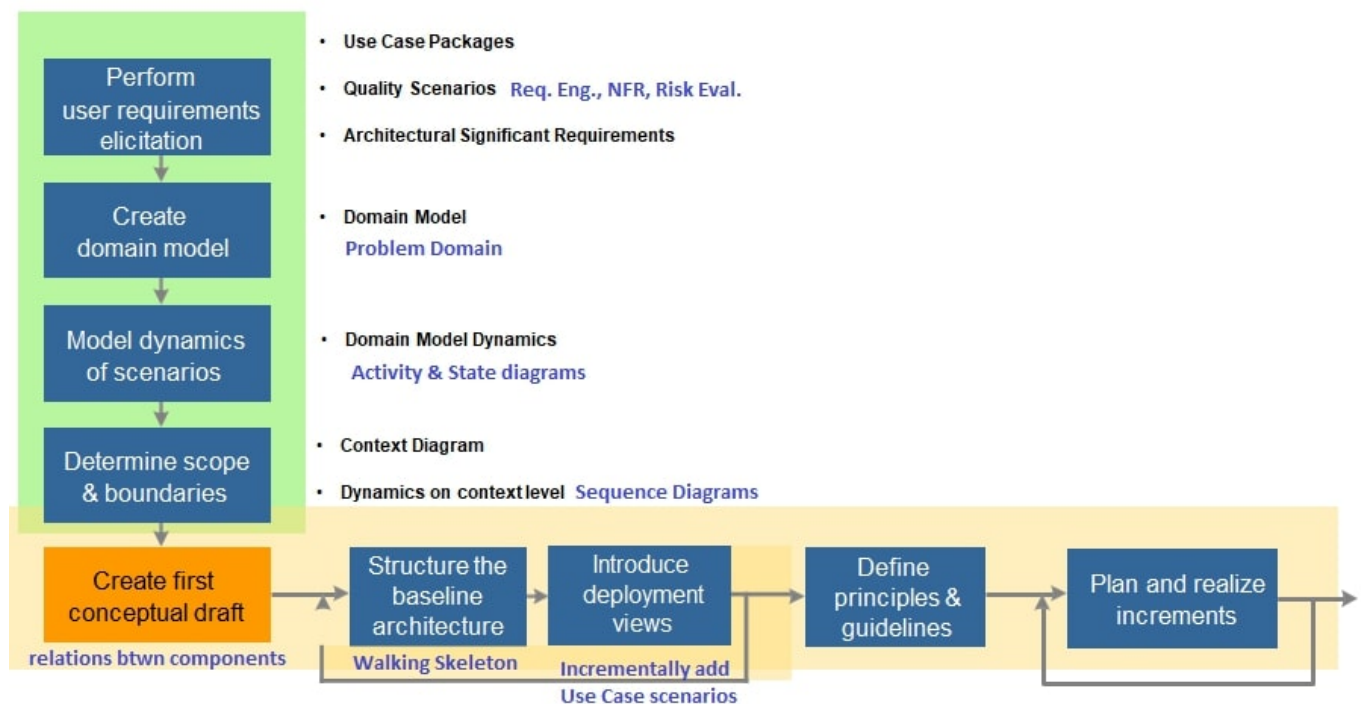
- Abstraction of reality
- Focus on fundamental & critical concepts
- Rationels / Why

Architecture Design = Creative freedom - Forces: (*func./non-func. reqs* + *org./biz./processes*)

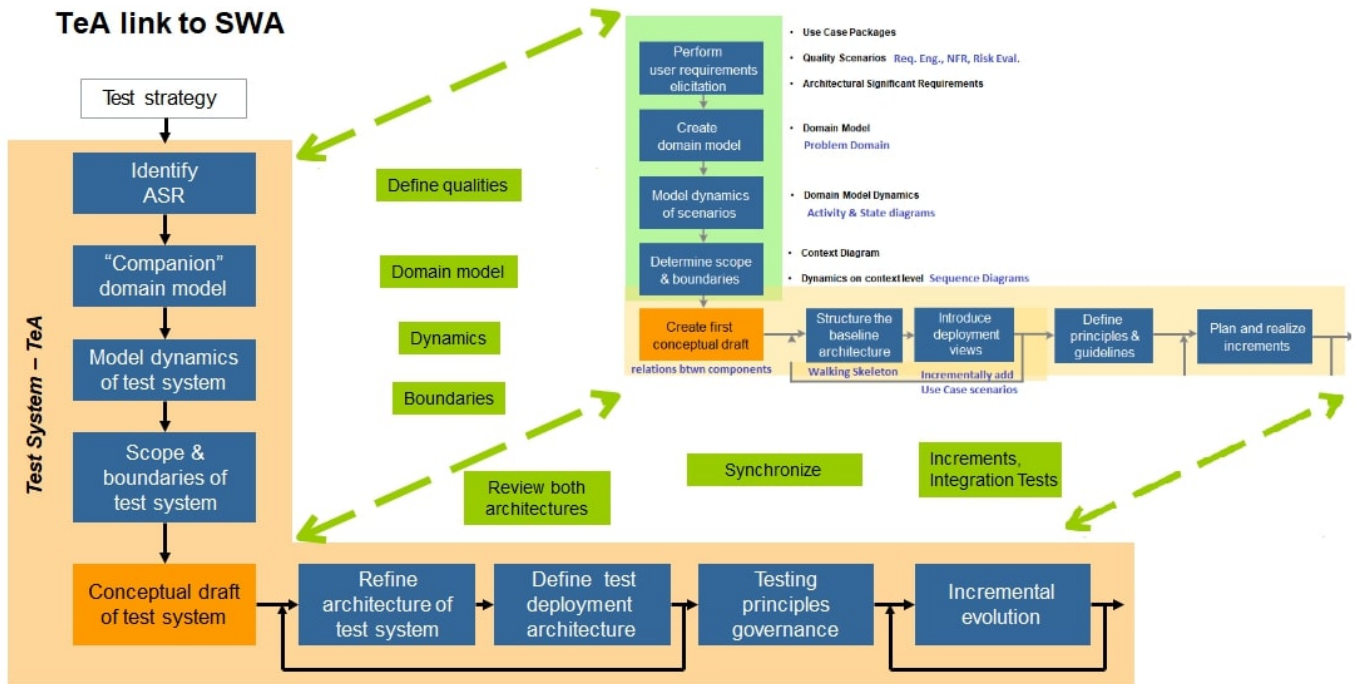
## Architecture Design - step by step

---

### Architecture Design Step by step Overview



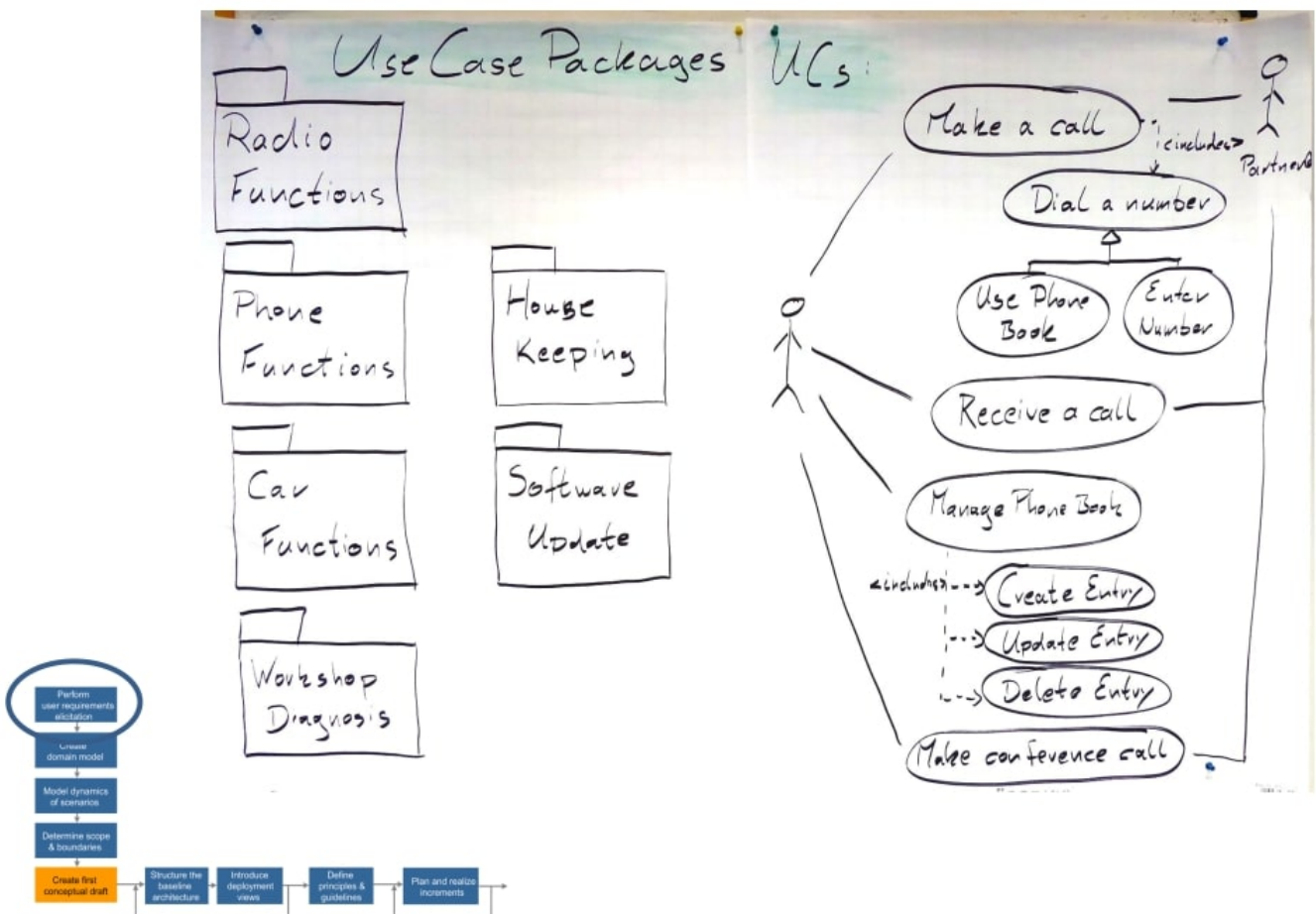
Link between TeA and SWA



## 1) SWA: Requirements Elicitation - TeA: Identify ASR's

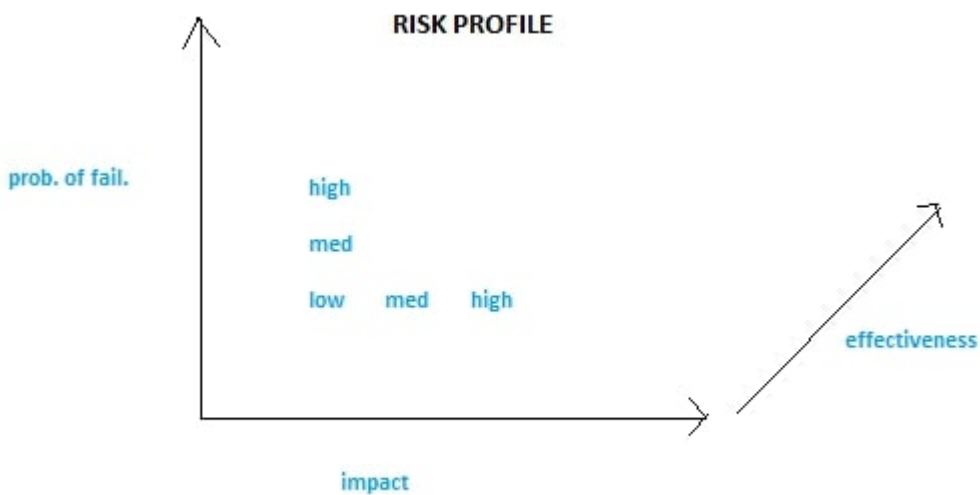
### Use Case Packages

Each use case is a requirement.



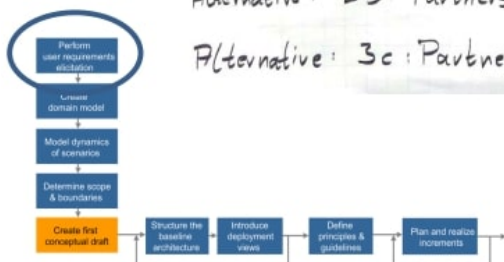
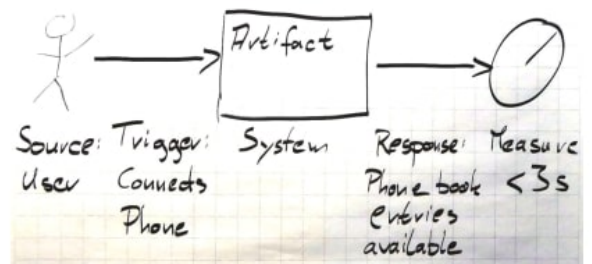
### Quality Scenarios

- Requirements Engineering: Given When Then
- NFRs
- Risk based evaluation: Risk Based Testing



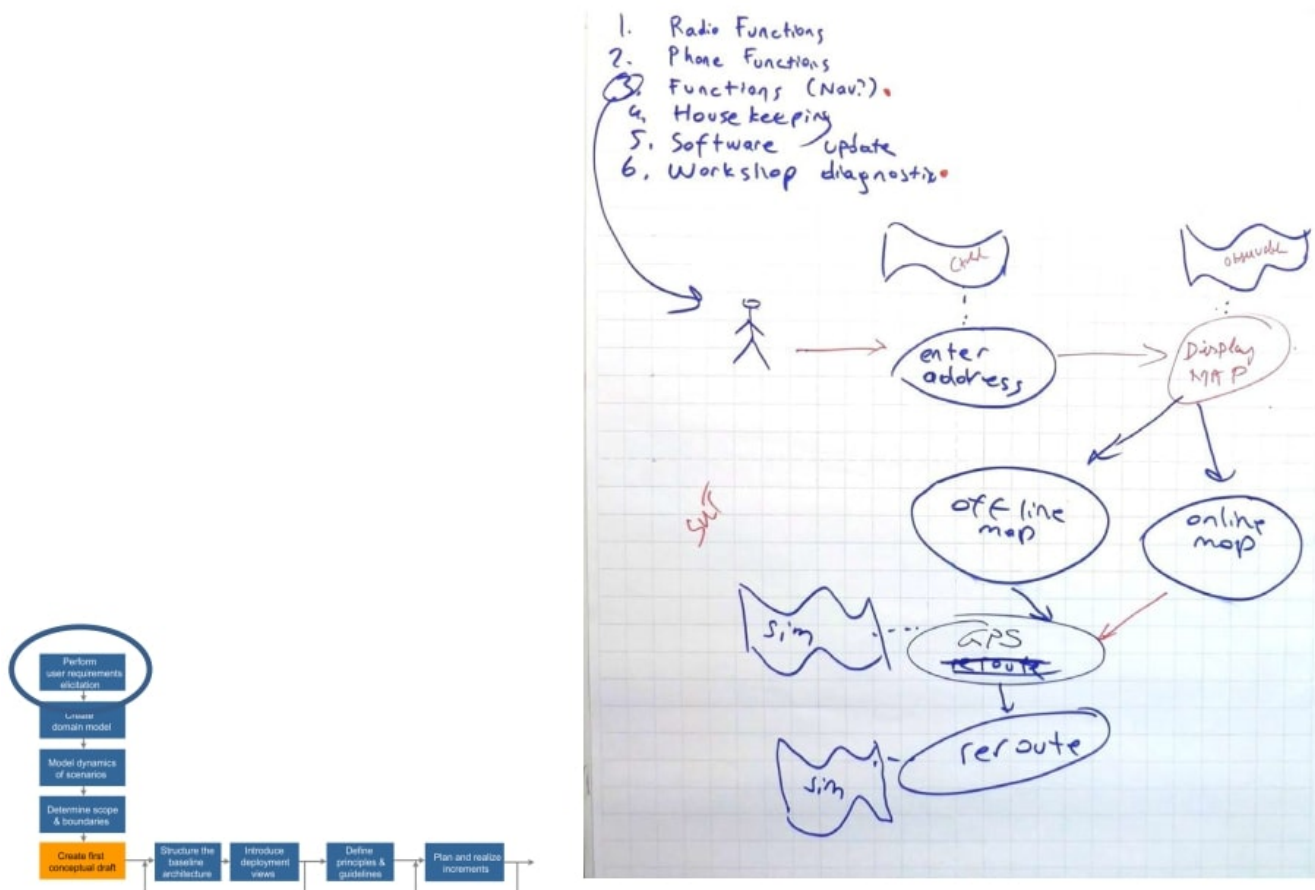
Name: Make a call  
 Actors: User, Call partner  
 Trigger: User indicates call wish  
 Pre Cond: Phone connected, idle  
 Post Cond: idle  
 Success Scen.: 1. User dials a number  
 2. System establishes call  
 3. Call partner accepts  
 4. User & Partner talk  
 5. User hangs up  
 Alternative: 2a: Call cannot be established  
 Alternative: 2b: Partners line is busy  
 Alternative: 3c: Partner rejects

## Quality Scenarios



## Architectural Significant Requirements (ASR's)

## ASR's - Architectural Significant Requirements



## 2) SWA: Domain Model - TeA: "Companion" Domain Model

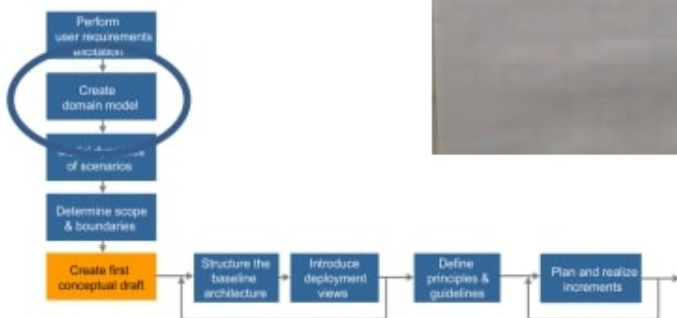
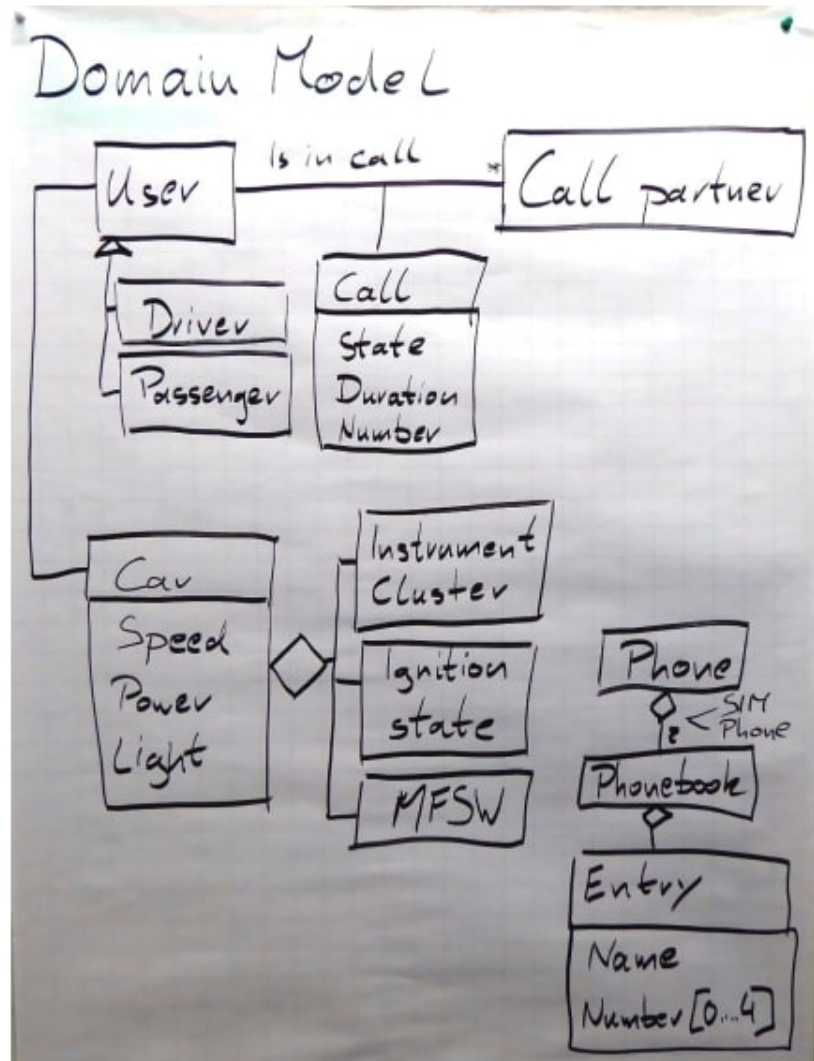
This is not a class diagram, but the **Problem Domain**. It is a **conceptual model that incorporates behavior and data**.

Question: what is the relation between a Domain Model and a requirement?

There can be long discussions on requirements – features – use cases – epics – domain models – context diagrams – package diagrams – component diagrams – etc. These are different perspectives/views/notations used to specify something with different intent, purpose, scope, level of detail.

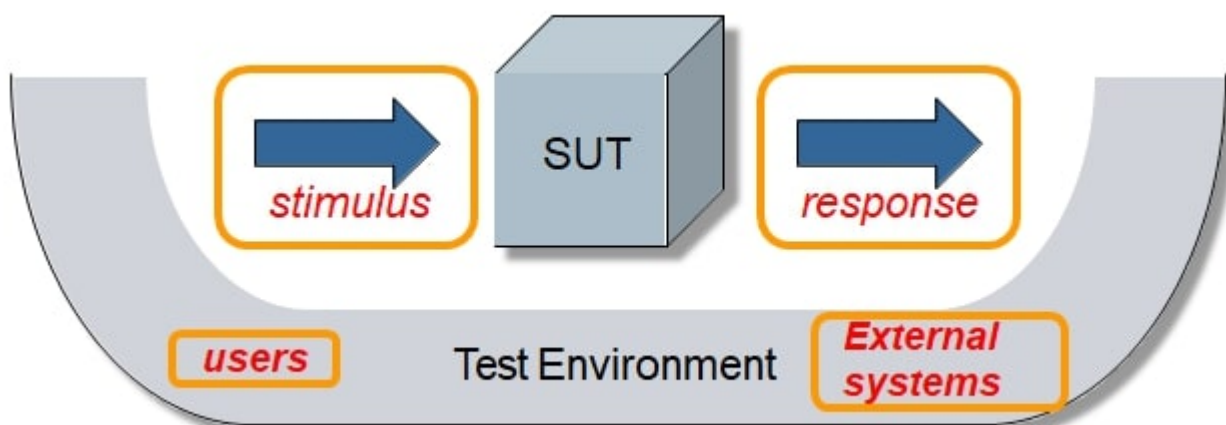
- *Requirement* : capability or condition needed by a stakeholder
- *Domain Model* : conceptual model that incorporates behavior and data
- *Context diagram* : focuses on what is in/out of scope, boundary between system and environment





Your Test System is the "companion" Domain Model, you have to cover all the other elements:

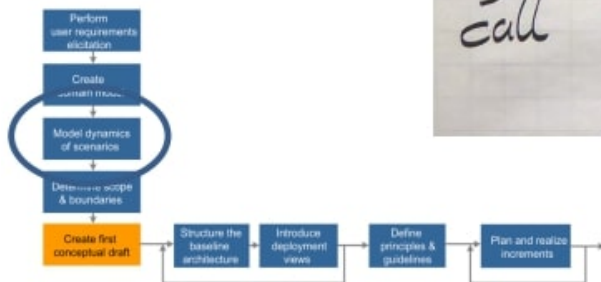
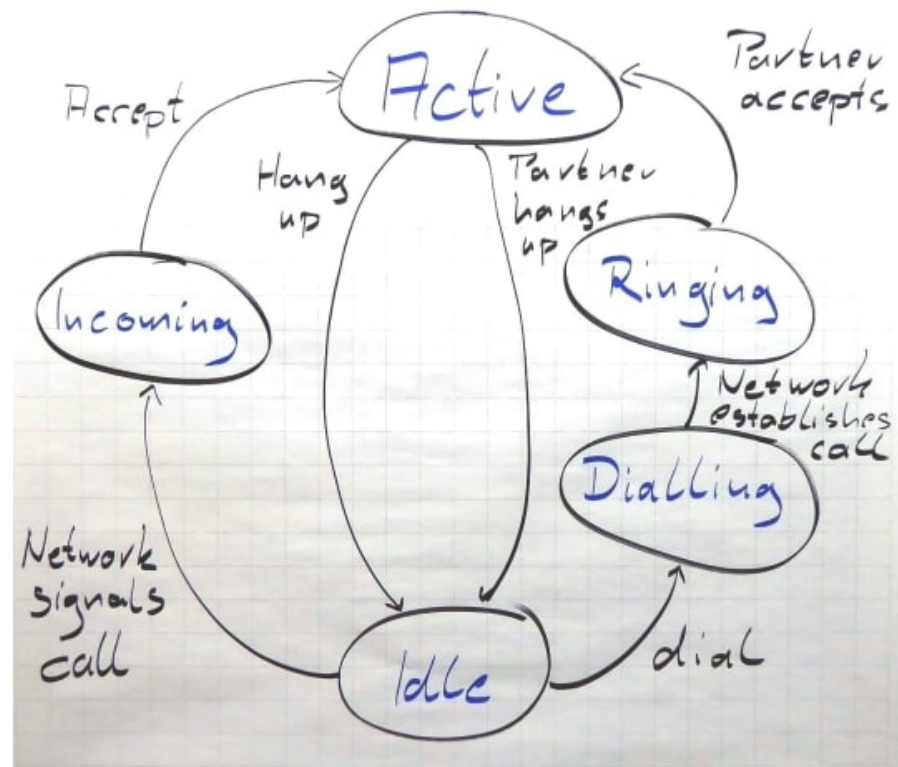
- Human user interaction (simulated or real)
- Communication with other actors (external systems, ...)
- Identify capabilities of these elements and design test cases accordingly



### 3) SWA: Domain Model Dynamics - TeA: Test System's Model Dynamics

#### Activity & State Diagrams

## Domain Model Dynamics



### 4) SWA: Determine Scope Boundaries - TeA: Test System's Scope & Boundaries

#### Context Diagram

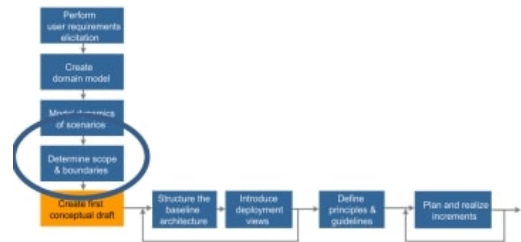
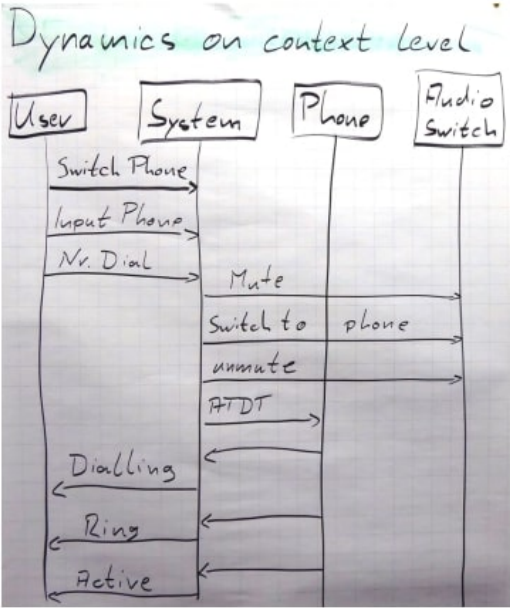
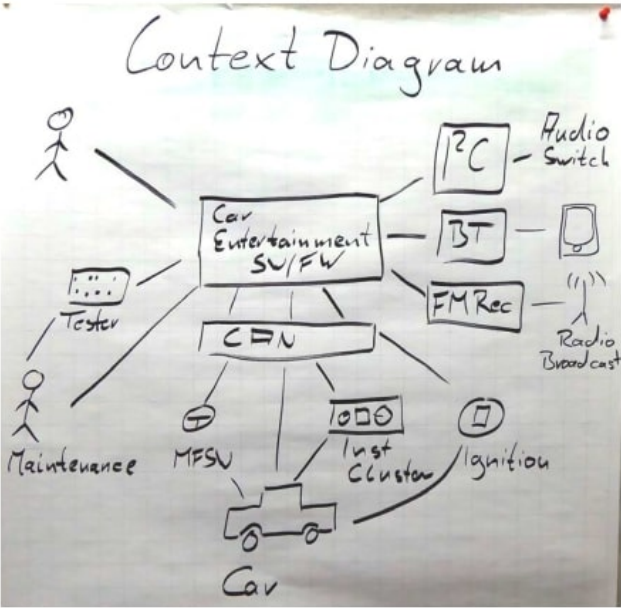
For the product/ system, clearly define the boundaries: **What is IN and what is OUT of scope, boundary between system and environment.**

Context diagram may look like a use case package diagram but has different intent:

- context diagram: describes what is in/out of scope, the so-called "context", the interaction between system and the environment around.
- use case package diagram: shows packages and relationships between them, but can be more an "internal only" view.

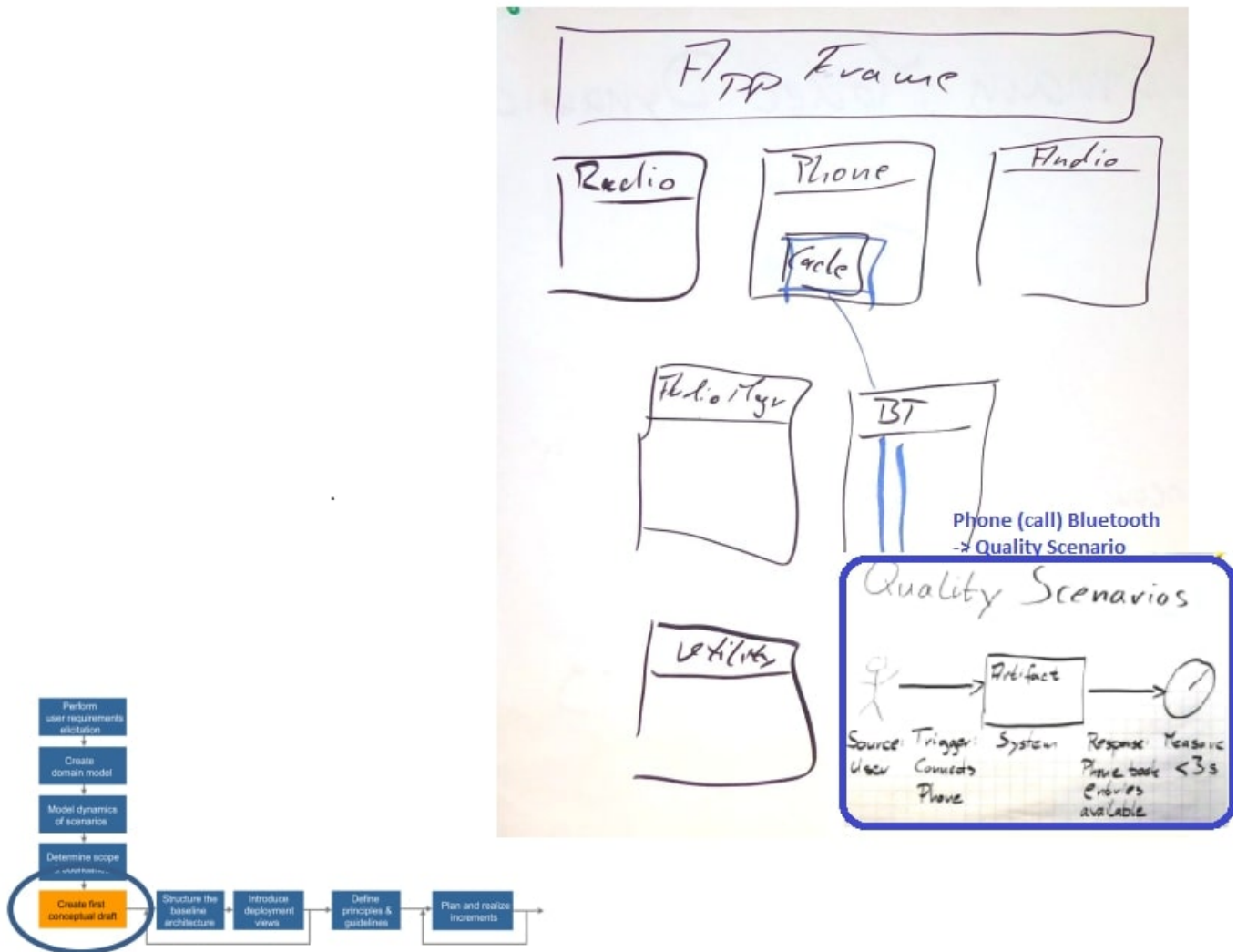
#### Dynamics on Context level

#### Sequence Diagrams



5) SWA: Conceptual Draft - TeA: Test System's Conceptual Draft

Relations between components.



## 6+) SWA: Structure the baseline Architecture & Introduce Deployment Views - TeA: Refine test architecture & Define test deployment architecture

- Walking Skeleton : most important & highest risk
- Incrementally add use case scenarios

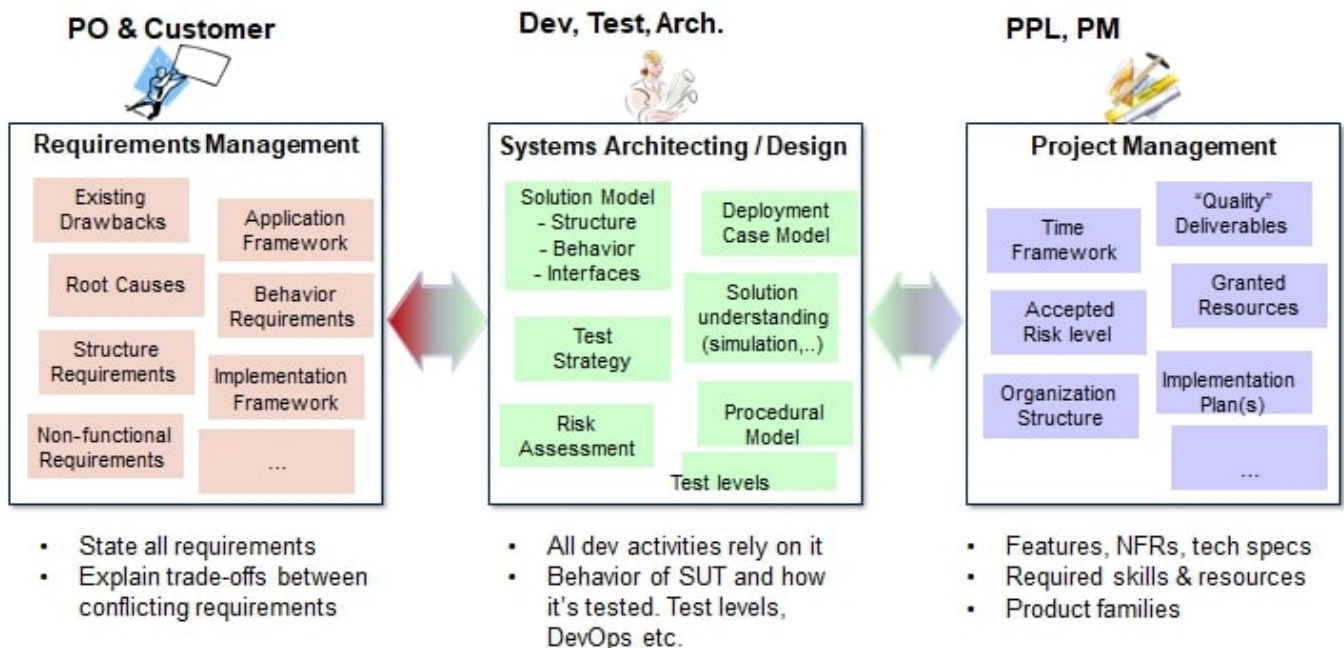
## Architectural Views & Documentation

What should be documented?

- Context & boundaries
- UML views of the architecture itself
- Design rationale
- How the architecture addresses FRs & NFRs & cross-cutting concerns



## Architectural views & documentation Stakeholders



Architectural Views: Kruchten, Zachmann...

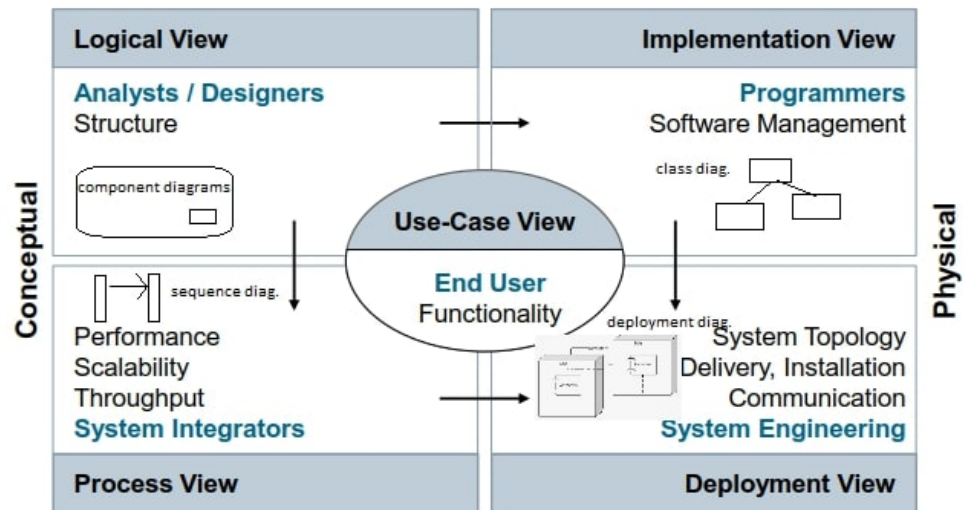
4+1 View explained:

- User view: all possible scenarios the user expects from SUT
- Functional aspects:
  - Logical View: how the functionality use cases are modeled. *Component diagrams*
  - Implementation/Development View: how the functionality is implemented (source code, libs, executables etc.) *Class diagrams*
- Non-functional aspects:
  - Process View: how the artifacts will be executed in terms of concurrency, scalability, synchronization. *Sequence, Activity, State diagrams*
  - Deployment View: maps software artifacts to hardware entities and shows the distribution of functionality. *Deployment diagrams*. one view in the 4+1 views by Kruchten, sometimes also

called physical view, see [[https://en.wikipedia.org/wiki/Deployment\\_diagram](https://en.wikipedia.org/wiki/Deployment_diagram)]

## 4+1 View (Kruchten)

Rational Unified Process 4+1 View introduced by Philippe Kruchten



## Siemens SW Architecture Doc Template

System Architecture Description	Version:	6.5
System_Architecture_Description_TEMPLATE.docx	Page:	3 / 34
	Date:	2017-01-11
	Status:	preliminary

### Contents

<b>1 PREFACE</b>	<b>8</b>
1.1 Purpose and Audience	8
1.2 Responsibilities	8
1.3 Document Structure	8
1.4 Definitions of Terms and Abbreviations	9
1.5 References	9
<b>2 &lt;PRODUCT/SYSTEM&gt;SCOPE</b>	<b>10</b>
2.1 Business Context	10
2.2 Architecturally Significant Requirements	10
2.3 Quality Attributes	10
2.3.1 Overview	10
2.3.2 Example Quality Attribute: Performance	11
2.3.3 <Quality Attribute #2>	11
2.3.4 <Quality Attribute #N>	11
2.4 Standards	11
2.5 Boundary conditions for development	12
2.5.1 Runtime Environment	12
2.5.2 Programming Environment	12
2.5.3 Conventions	13
2.5.4 Teams and Sites	13
2.5.5 Third-party Components	13
2.5.6 Reuse of Components	13
2.6 Patents	14
2.6.1 Foreign Patents	14
2.6.2 New SIEMENS Patent Candidates	14
2.7 Architectural Risks	14
2.7.1 Risks Introduced by Boundary Conditions	15
2.7.2 Risks Introduced by Implementation Decisions	15
<b>3 OUTSIDE VIEWS</b>	<b>16</b>
3.1 Domain Model	16
3.2 System Context	16
3.2.1 High-Level View	16
3.2.2 Interfaces	17
3.2.3 Collaboration	18
3.3 Use Case View	18
<b>4 INSIDE VIEWS</b>	<b>20</b>

SIEMENS

Org Unit

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

<https://wiki.ct.siemens.de/display/MembersSA/Tools+and+Templates>

System Architecture Description	Version:	6.5
System_Architecture_Description_TEMPLATE.docx	Page:	4 / 34
	Date:	2017-01-11
	Status:	preliminary

4.1 Logical View	20
4.1.1 System Decomposition	20
4.1.2 <Component #1>	21
4.1.3 <Component #N>	21
4.1.4 <Interface #1>	21
4.1.5 <Interface #N>	22
4.2 Runtime View	22
4.2.1 Component Distribution and Concurrency	22
4.2.2 Component Communication	22
4.2.3 Processes and Tasks	23
4.2.4 Scheduling and Deterministic Behavior	24
4.2.5 Start-up, Shut-down, and Recovery	25
4.2.6 Exception Handling and Error Processing	25
4.2.7 Persistent Memory	25
4.2.8 Logging, Tracing, Reporting	25
4.2.9 SW-Update	26
4.2.10 Transactions	26
4.2.11 Sessions	26
4.3 Development View	26
4.4 Deployment View	26
4.5 User Interface View (additional)	27
4.6 Data View (additional)	27
4.7 Variability View (additional)	28
4.8 Migration View (additional)	28
<b>5 ARCHITECTURAL QUALITIES</b>	<b>30</b>
5.1 Example Quality Attribute: Security	30
5.1.1 Security Concepts	30
5.1.2 Patch Concepts	30
5.2 <Quality Attribute #2>	30
5.3 <Quality Attribute #N>	31
<b>6 ARCHITECTURAL DECISIONS AND ALTERNATIVES</b>	<b>32</b>
6.1 <Requirement/Decision #1>	32
6.2 <Requirement/Decision #N>	32
6.3 Other Significant Decisions	32
<b>7 PRINCIPLES AND GUIDELINES</b>	<b>33</b>
<b>A APPENDIX</b>	<b>34</b>
A.1 List of open issues	34
A.2 Index	34

SIEMENS

Org Unit

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

## Test Architecture Documentation

Driven by Architecture documentation; you must understand & review it as the TeA.

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

- Part 1: Concepts and Definitions
- Part 2: Test Process
- Part 3: **Test Documentation**
- Part 4: Test Techniques

### Test Documentation

- **Organizational test process**
  - Test strategy (*test levels, test goals, who does what*)
- **Test management processes**
  - Test plan
  - Test status
  - Test completion report

*(Test Exit Criteria: test coverage, test progress, defects)*

- **Dynamic test processes**
  - Test Designs
  - Test case/specs
  - Test data
  - Test environment
  - Test execution log
  - Defects

### Test Plan Template

Example: Test Concept for <XYZ> Template

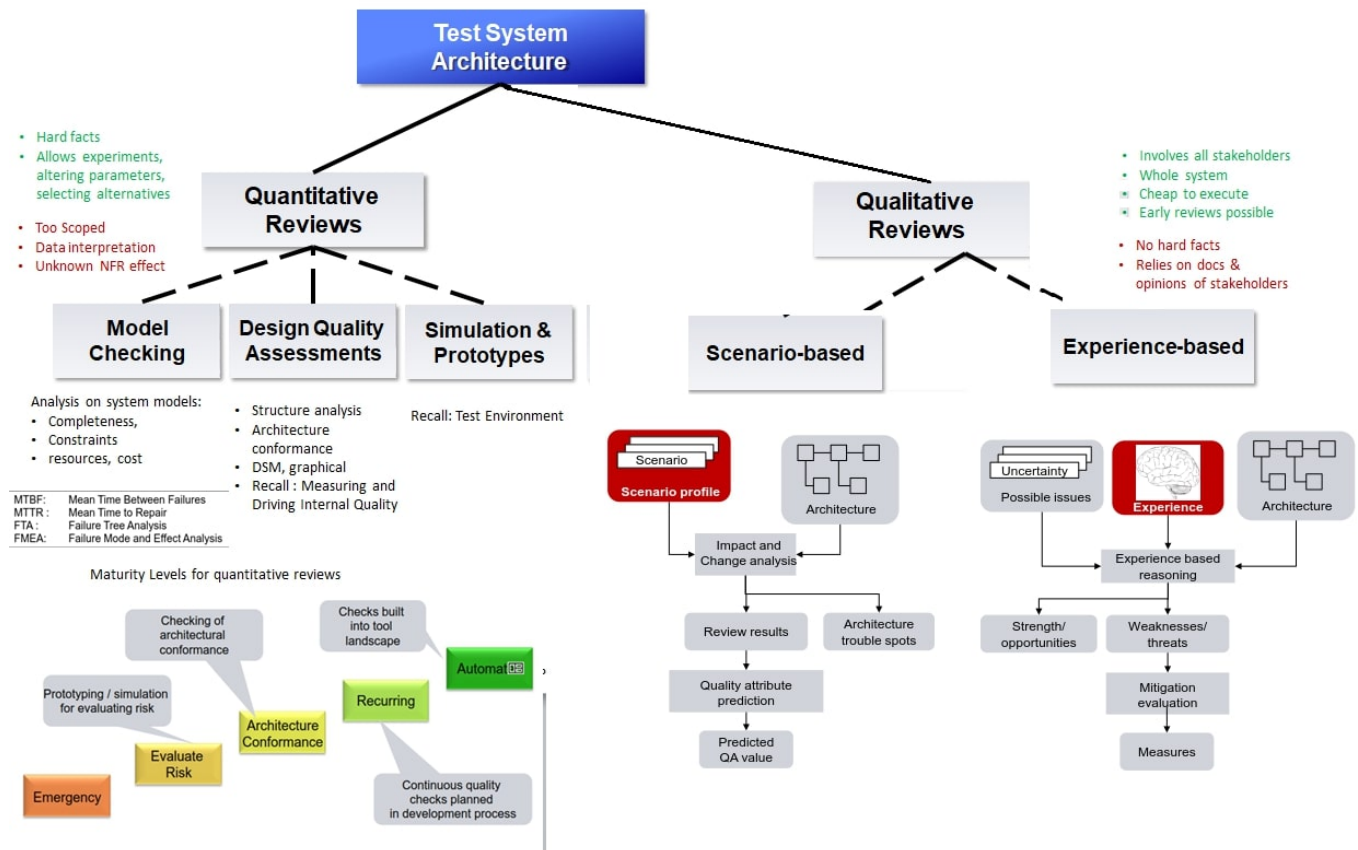
Ingenuity for life

1.	Scope .....	5.4	Test data .....
1.1	Test basis .....	5.5	Test deliverables and documentation .....
1.2	Open issues and topics to be clarified .....	5.6	Defect and change management .....
2.	Stakeholders .....	6.	Test environment .....
3.	Test goals .....	6.1	Hardware / Software / Operational environment for testing...
3.1	Goals and responsibilities of testing .....	6.2	Test configurations and setup .....
3.2	Assumptions, dependencies, and constraints .....	6.3	Deployments and releases .....
4.	Test scope .....	7.	Test automation .....
4.1	Test objects, test items, system under test (SUT) .....	7.1	Development environment, CI .....
4.2	Major product risks (risk-based testing) .....	7.2	Frameworks .....
4.3	Features to be tested .....	7.3	Test tools .....
4.4	Features not to be tested.....	7.4	Test scripts .....
4.5	Non-functional requirements (NFRs) to be tested .....	8.	Test logistics .....
4.6	Non-functional requirements (NFRs) not to be tested ....	8.1	Schedule, timeline .....
4.7	Test data requirements .....	8.2	Staffing.....
5.	Test strategy and approach .....		
5.1	Test levels, entry/exit criteria .....		
5.2	Test types.....		
5.3	Test techniques .....		

# Architecture Quality & Reviews

Criteria for good architecture (recall NFRs and Quality Characteristics) :

- Reliable
- Maintainable
- Scalable
- Performance
- Security



## Qualitative Review Phase

- Prep: clarify review goals / *reviewers*
- Collect: interviews with stakeholders, docs, source *reviewers & stakeholders*
- Elaborate / *reviewers*
  - SWOT analysis: Strengths, Weaknesses, Opportunities, Threats
  - Dealing with Weaknesses
  - ATAM (SWOT alternative:) : Architectural Tradeoff Analysis
- Consolidation : final report / *reviewers*
- Presentation to stakeholders / *reviewers*
- Workshop (optional) *reviewers & stakeholders*



## Qualitative Review Toolbox

*Ingenuity for Life*

	Active design review	Industry practice	Architecture Tradeoff Analysis Method	System Architecture Analysis for the Field
Type	Experience-based, scenario-based	Experience-based	Scenario-based	Scenario-based
Intention	Improve design, find errors	SWOT analysis, identify measures	Clarify and prioritize requirements, evaluate suitability of architecture for change scenarios	Improve design, find errors, identify measures for mitigating typical lifecycle issues

## Error classification

*Ingenuity for Life*

Error type	Examples
<b>Inconsistencies</b>	<ul style="list-style-type: none"> <li>Places where the design will not function properly</li> <li>Contradicting behavior</li> </ul>
<b>Inefficiencies</b>	<ul style="list-style-type: none"> <li>Unnecessary complexity</li> <li>Inefficient use of resources, channels, etc</li> </ul>
<b>Ambiguities</b>	<ul style="list-style-type: none"> <li>Unclear requirements</li> <li>Undocumented assumptions</li> <li>Elements of the design specification which can be understood in different ways</li> </ul>
<b>Inflexibility</b>	<ul style="list-style-type: none"> <li>Elements making change requests difficult</li> <li>Obstacles for dealing with lifecycle issues (e.g. configuration, service, upgrade, etc.)</li> </ul>

## Comparison of qualitative reviews

	Active Design Review	Industry Practice	Architecture Tradeoff Analysis Method	System Architecture Analysis for the Field
<b>Interaction</b>	Designer, reviewer	Interviews	Workshop	Workshop
<b>Phase</b>	Detailed component / module design ready	After architecture has been designed	Architecture design complete enough for walkthroughs	Detailed component / module design ready
<b>Strength</b>	Focused on finding defects in design	Concrete measures	Bring stakeholders together, requirement prioritization	Concrete measures
<b>Key restriction</b>	Small scale	No common understanding of requirement priorities	No measures	Focus on system in operational conditions
<b>Duration</b>	2 days / reviewer	Four weeks regular 1 day flash	Two weeks	Two weeks

## Overview of qualitative architecture reviews

	SAAM	ATAM	ADR	Industry practice
<b>Type</b>	Scenario-based	Scenario-based	Experience-based, scenario-based	Experience-based
<b>Intention</b>	Clarify and prioritize requirements, evaluate suitability of architecture for change scenarios	Clarify and prioritize requirements, find risks, sensitivity points, tradeoffs	Improve design, find errors	SWOT analysis, identify measures
<b>Interaction</b>	Workshop	Workshop	Designer, reviewer	Interviews
<b>Phase</b>	Architecture design complete enough for walkthroughs	Architecture design complete enough for walkthroughs	Detailed component / module design ready	After architecture has been designed
<b>Strength</b>	Bring stakeholders together, requirement prioritization	Like SAAM, but deeper architectural evaluation	Focused on finding defects in design	Concrete measures
<b>Key restriction</b>	No risks, no measures	No measures	Small scale	No common understanding of requirement priorities
<b>Duration</b>	2–3 days	Two weeks	2 days / reviewer	Four weeks regular 1 day flash