README.md 4/18/2019

REQUIREMENTS ENGINEERING

Sentence Template for System Requirements

Given When Then

1 / 8

Characteristics of Good Requirements

Requirements Traceability

README.md 4/18/2019

2 / 8

Tracing Usage:

**Analysis Description Process**

Derivation *Why is this here?* cost-benefit analysis

Impact *What if this changed?* change management

Coverage *Have I covered all reqs?* management report

NFRs / Quality Attributes

Software Quality Characteristics pdf

Step 1: Utility Tree

README.md 4/18/2019

3 / 8

Step 2: Scenario Description Template / Workflows

To reduce ambiguity and increase testability, each measurable scenario in the Utility Tree gets described with this template.

Step 3: Refine Scenarios

README.md 4/18/2019

4 / 8

Step 4: Implement Design Strategies to support the NFRs

.. and use Design patterns when needed...

README.md 4/18/2019

5 / 8

Testing NFRs

README.md 4/18/2019

6 / 8

KANO Model

For the Product Manager KANO is a tool to find the right mix between the different feature types. For the Test Architect KANO helps to identify risks.

README.md 4/18/2019

7 / 8

Configuration Management

Source Control, configurations, environment...

**Devops**: continious integration, deployment, testing...

Change Request Management

**Change Request**: any issue that cannot go in a patch or point rev, has to go into the product asap.

**Change Management**: because requirements are incomplete, erroneous. ambiguous.

README.md 4/18/2019

**Defect Management**: because tests reveal defects that need correction.

Tool -> Feature -> trace & Test

**CCB**:

8 / 8

README.md 4/17/2019

SOCIAL CAPABILITY

Active Listening

Do not give solution!

1 / 4

Feedback

You can use feedback for blind spots.

README.md 4/17/2019

2 / 4

Stakeholders

Who's buy-in you need?

Situational Leadership

README.md 4/17/2019

3 / 4

Conflict Management

README.md 4/17/2019

4 / 4

README.md 4/18/2019

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

1 / 15

Link between TeA and SWA

README.md 4/18/2019

2 / 15

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

Use Case Packages

Each use case is a requirement.

Quality Scenarios

README.md 4/18/2019

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

3 / 15

Architectural Significant Requirements (ASR's)

README.md 4/18/2019

4 / 15

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

README.md 4/18/2019

5 / 15

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

README.md 4/18/2019

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

Activity & State Diagrams

6 / 15

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

README.md 4/18/2019

7 / 15

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

Relations between components.

README.md 4/18/2019

8 / 15

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

README.md 4/18/2019

9 / 15

Architectural Views: Kruchten, Zachmann...

4+1 View explained:

User view: all posibble 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

README.md 4/18/2019

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

Siemens SW Architecture Doc Template

10 / 15

Test Architecture Documentation

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

README.md 4/18/2019

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

11 / 15

README.md 4/18/2019

12 / 15

Architecture Quality & Reviews

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

Reliable Maintainable Scalable Performance Security

README.md 4/18/2019

13 / 15

**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*

README.md 4/18/2019

14 / 15

README.md 4/18/2019

15 / 15

README.md 4/18/2019

TESTING & QUALITY

Value of Testing

What is Testing? : investigation of the SUT to provide **information** that results in **improvements**.

The 5 Dimensions of Testing

Coverage: an assessment for the thoroughness or completeness of testing with respect to our test model - *Paul Gerrard*

1 / 13

ROI of Testing

README.md 4/18/2019

2 / 13

Cost of Quality

Test Strategy

README.md 4/18/2019

3 / 13

Testing serves a purpose (*test mission*) that has goals (*test policy*) and requires a map (*test strategy*).

Test levels – V model with architecture testing

README.md 4/18/2019

Risk Based Testing

Risk Profile

4 / 13

Risk Based Testing Worksheet. You can download the worksheet in xls .

README.md 4/18/2019

Relations in RBT worksheet

5 / 13

Design for Testability

Goal : Controllable, Observable, Reliable : *Instrinsic Testability* . More On Heuristics of Testability

Why : reduce the cost of testing, diagnosis, maintenance.

Who : system, software and test architects

How : TDD, Loose Coupling, Inversion of control, SOLID, follow the best practices of clean code & architecture.

README.md 4/18/2019

6 / 13

Test Exit Criteria

README.md 4/18/2019

Performance Testing and Scalability

7 / 13

At a certain load, the response time sky-rockets.

TDD

From Req. to unit test level. The most effective way of specifying something is to describe how you would test it.

README.md 4/18/2019

Test Design Techniques

Test Design Techniques pdf

**Black-box**: req. based, workflow, statistical/markov, eq.class & boundary value, state-based, combinatorial, model based **Gray-box**: interfaces between components, services, systems **White-box**: statement, branch, path: cyclomatic complexity *(Edges-Nodes-2 = independent paths)* **Fault-based**: exploratory, fuzzing, mutation. Data Type Attacks and Web Tests pdf **Regression**: Risk Based Testing, testing firewall (re-test parts influenced by changes)

8 / 13

README.md 4/18/2019

9 / 13

Test Automation Patterns website

Test Automation Design Patterns paper

Test Environment

**Test environment**: test rig

README.md 4/18/2019

**Test infrastructure**: test rig + tools + office network etc.

**Test suite architecture**: test levels

10 / 13

Internal Quality

Negative efects:

Slows development with unplanned activities Rising cost of maintenance, new features, change Rising cost of regression testing, system testing for hotfixes Rising cost of onboarding Complex & risky integration

README.md 4/18/2019

11 / 13

Technical Debt

Lack of internal quality reseults in technical debt.

Measuring and Driving Internal Quality

To measure internal quality

Static code analysis, linters etc. Req. trace

README.md 4/18/2019

On-boarding feedback Visualize with tools, reviews Test gap analysis Automated document analysis

12 / 13

To drive internal quality, you must monetize it:

Test Code & Architecture Quality Management

Test Code Quality at different levels:

Micro code tools

Macro hacky code review

Architecture UML review, some tools for architecture analysis

README.md 4/18/2019

13 / 13