Software Architecture and Techniques

Architecture Documentation

Marcel Baumann, tangly llc

Lecture Content

- *Why Agile Architecture and Design?*
- *Evolution of Software Architecture over the last Decades*
- *What is Agile Architecture?*
- *Agile Approaches with Scrum, XP, LeSS*
- *Refactoring*
- *Errors, Vulnerabilities, Smells in Source Code*
- *Architecture of Components and Subsystems*
- *Verify Functional Features*
- *Validate Quality Attributes of Software Architecture*
- **Architecture Documentation**
- **Architecture Trends I**
- **Architecture Trends II**
- **Domain Driven Design Workshop**
- **Team and Technical Excellence for Architects**

Truths (1/2)

- **Source code** is the architecture
- It is **expensive**, error prone and **cumbersome** to synchronize documentation with source code
- Agile is about **people, interactions**, stories, discussions, not about processes or tools
- ATAM, TOGAF, IEEE-SW standards are **obsolete**
- Hermes, Prince2, PMI are archeology subjects

Truths (2/2)

- **Never** use Microsoft Word – it is proprietary, and cannot be put under version control. You cannot easily search a set of Word documents.
- The more text documentation you have, the more synchronization errors you will have.
- Nobody reads a user manual. You open a user manual when you are desperate.
- Paper is useless.

What does an Architect?

- Understand requirements *and document them*
- Create collaboratively architecture *and document it*
- Advocate and promote architecture *in oral and written form*
- Evaluate architecture *and document the findings*

Why Should You Document?

Good architectural documentation

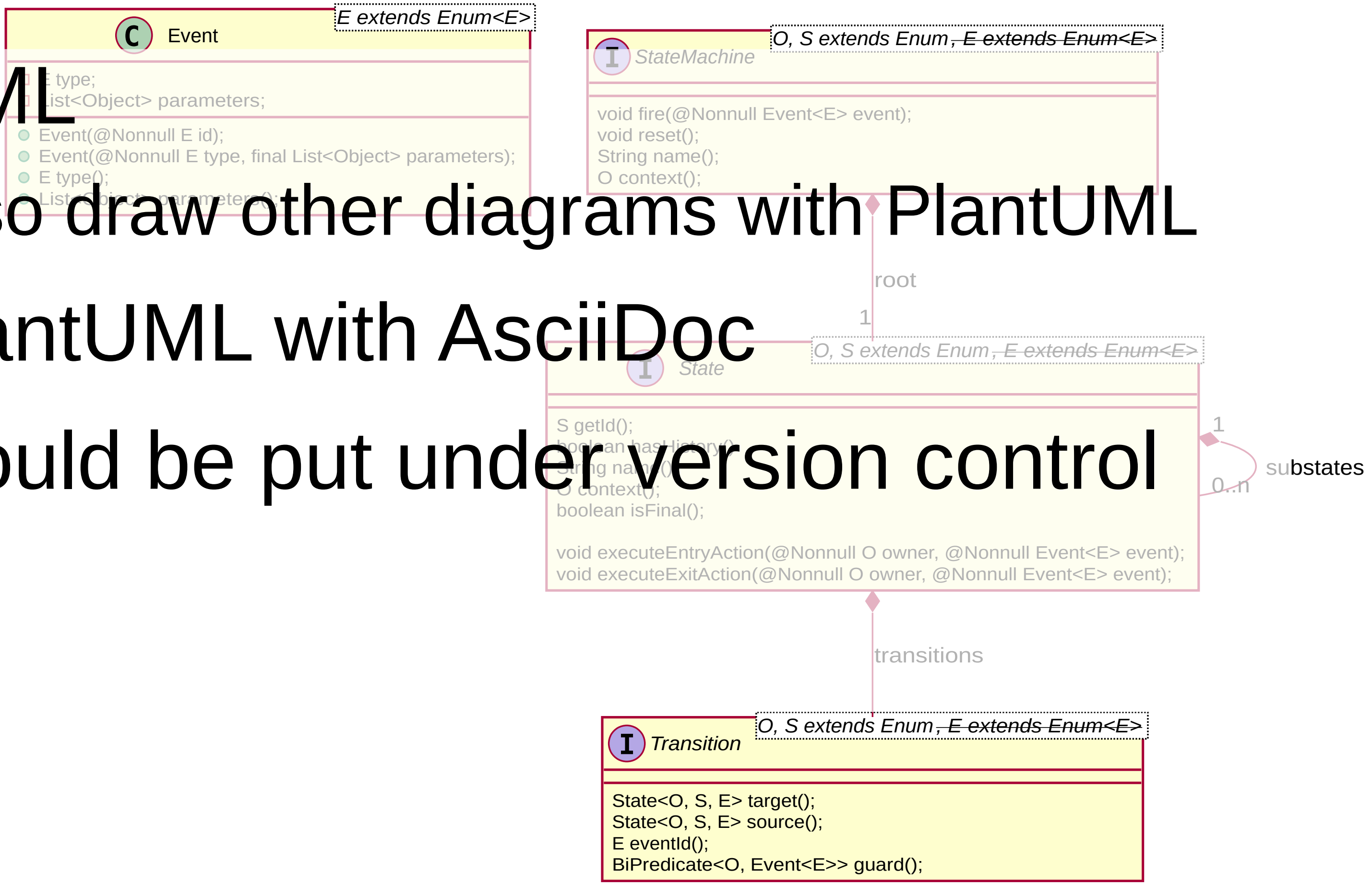
- is communicative and informative to its audience
- relies on **explanation** over notation
- meaningfully constrains the system
- conveys **critical information**
- chooses **simplicity** over sophistication
  - choose established solutions over novel solutions
  - must be a provable solution → code

Domain Driven Models

- Code is documentation
- Small models with explanation
- Event diagrams
- Acceptance test reports
- Traceability between code, acceptance tests and associated requirements

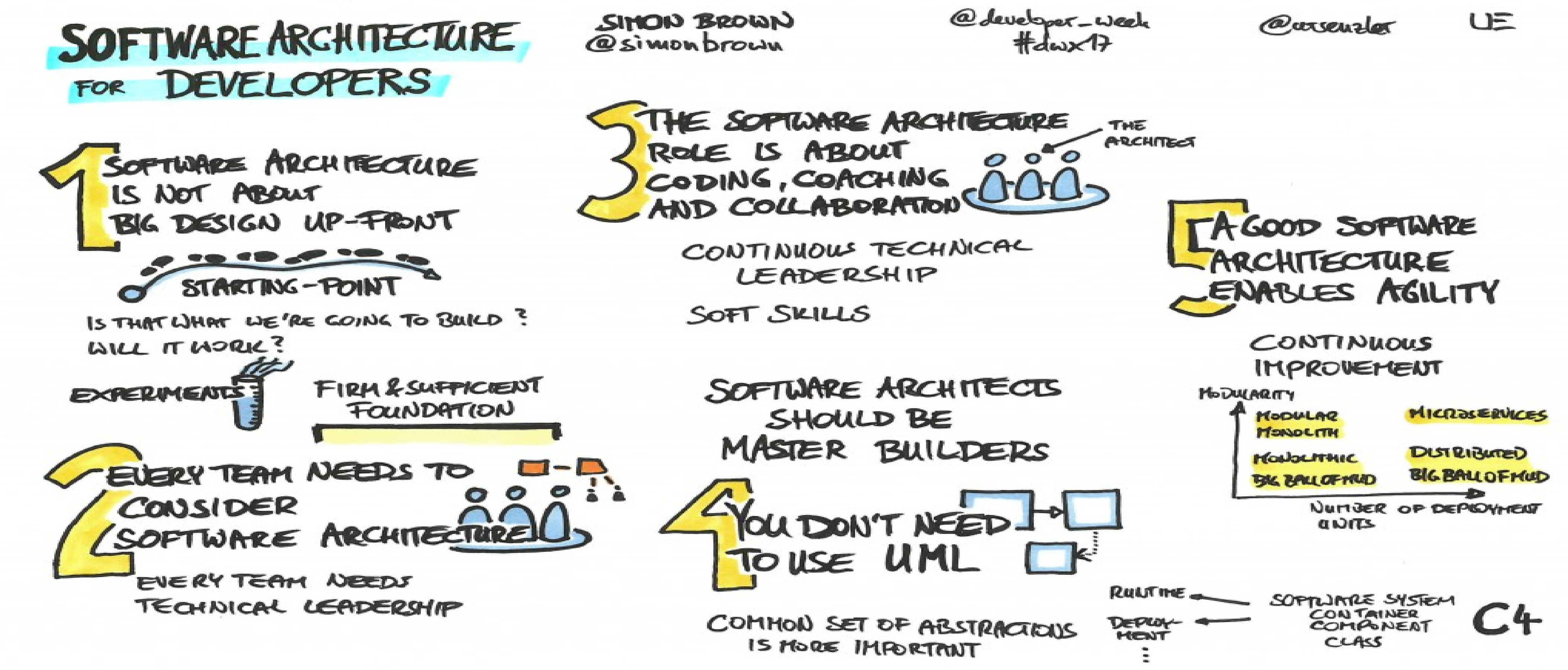
UML for Small Models

- Use PlantUML
  - You can also draw other diagrams with PlantUML
- Integrate PlantUML with AsciiDoc
- Can and should be put under version control

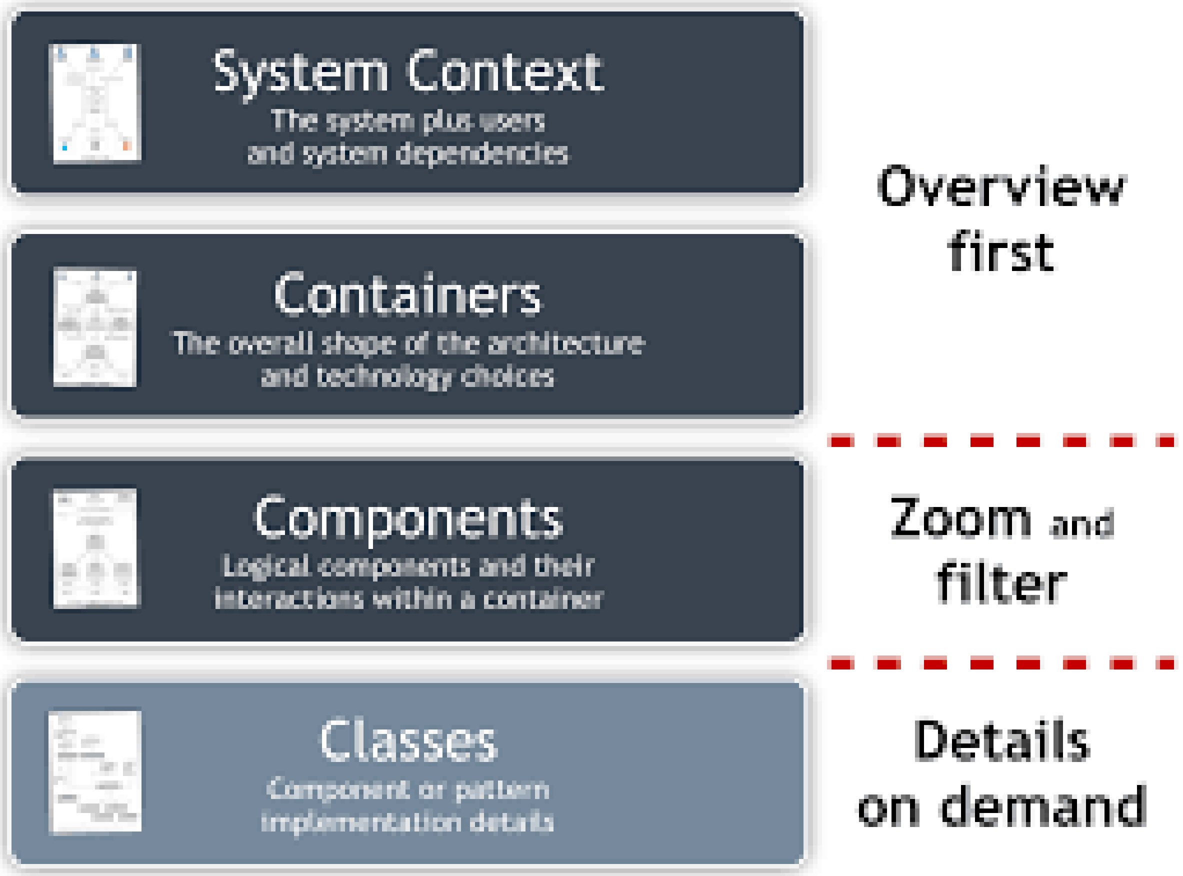


```
Event                          E extends Enum<E>
  E type;
  List<Object> parameters;
  Event(@Nonnull E id);
  Event(@Nonnull E type, final List<Object> parameters);
  E type();
  List<Object> parameters();

StateMachine                   O, S extends Enum, E extends Enum<E>
  void fire(@Nonnull Event<E> event);
  void reset();
  String name();
  O context();

State                          O, S extends Enum, E extends Enum<E>
  S getId();
  boolean hasHistory();
  String name();
  O context();
  boolean isFinal();

  void executeEntryAction(@Nonnull O owner, @Nonnull Event<E> event);
  void executeExitAction(@Nonnull O owner, @Nonnull Event<E> event);

Transition                     O, S extends Enum, E extends Enum<E>
  State<O, S, E> target();
  State<O, S, E> source();
  E eventId();
  BiPredicate<O, Event<E>> guard();
```

C4 Model for System


SOFTWARE ARCHITECTURE FOR DEVELOPERS
SIMON BROWN @simonbrown
@developer_week #dwx17
@ursenzler
UE
1 SOFTWARE ARCHITECTURE IS NOT ABOUT BIG DESIGN UP-FRONT
STARTING-POINT
IS THAT WHAT WE'RE GOING TO BUILD ?
WILL IT WORK?
EXPERIMENTS
FIRM & SUFFICIENT FOUNDATION
2 EVERY TEAM NEEDS TO CONSIDER SOFTWARE ARCHITECTURE
EVERY TEAM NEEDS TECHNICAL LEADERSHIP
3 THE SOFTWARE ARCHITECTURE ROLE IS ABOUT CODING, COACHING AND COLLABORATION
THE ARCHITECT
CONTINUOUS TECHNICAL LEADERSHIP
SOFT SKILLS
SOFTWARE ARCHITECTS SHOULD BE MASTER BUILDERS
4 YOU DON'T NEED TO USE UML
COMMON SET OF ABSTRACTIONS IS MORE IMPORTANT
5 A GOOD SOFTWARE ARCHITECTURE ENABLES AGILITY
CONTINUOUS IMPROVEMENT
MODULARITY
MODULAR MONOLITH
MICROSERVICES
MONOLITHIC BIG BALL OF MUD
DISTRIBUTED BIG BALL OF MUD
NUMBER OF DEPLOYMENT UNITS
RUNTIME
DEPLOYMENT
SOFTWARE SYSTEM
CONTAINER
COMPONENT
CLASS
C4

C4 Model for System

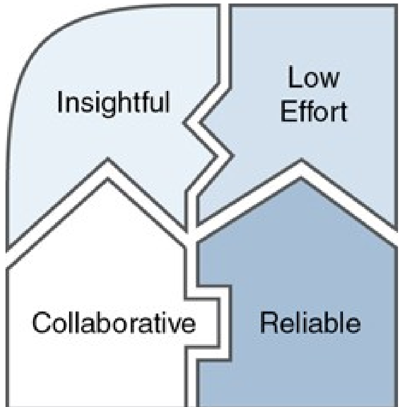

System Context
The system plus users and system dependencies
Containers
The overall shape of the architecture and technology choices
Components
Logical components and their interactions within a container
Classes
Component or pattern implementation details
Overview first
Zoom and filter
Details on demand

Architectural Design Record

- Document decisions with context, rationale and history as ADR
- History is part of the model
- Can and should be put under version control

Rules for Documentation

- Document stable concepts, not speculative ideas
- **Living documentation** is insightful, collaborative, reliable and requires low effort
  - JavaDoc (see also javadoc.io)
- Keep documentation just simple enough, not too simple
- Write the fewest documents with the least overlap
- **Display information publicly**
- *It should be searchable*



AsciiDoc (1/3)

- Write your **short** Software Architecture Document *SAD* in AsciiDoc
- Map your code examples with explanation
- AsciiDoc is text and is under version control
- AsciiDoc has the expression power of DocBook

AsciiDoc (2/3)

- Combine AsciiDoc text, cross-reference and UML diagram
- Find out how to generate documents
- Explore GitHub, GitLab and Bitbucket offerings – static web sites are automatically generated and stored in git -

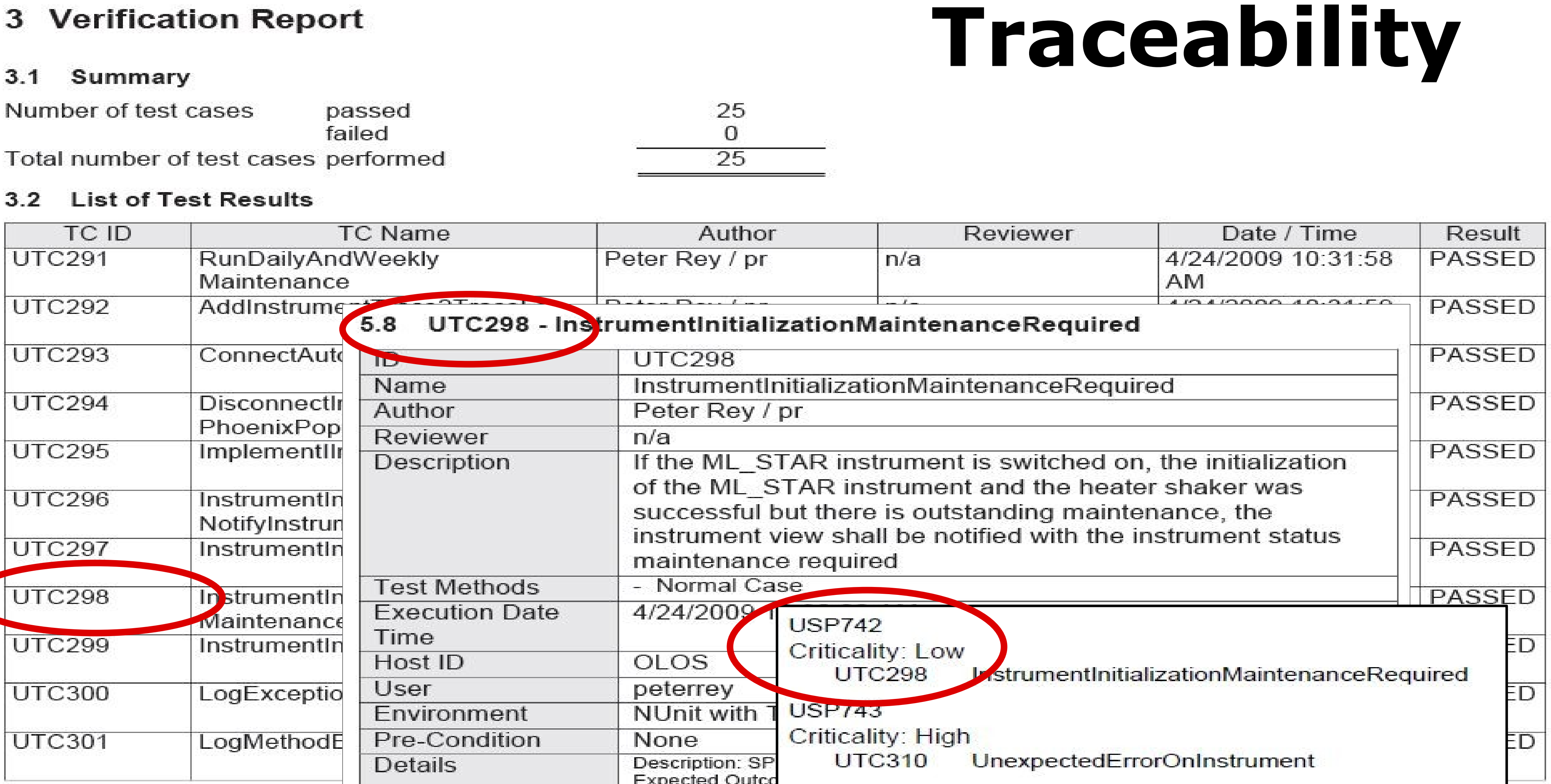
AsciiDoc (3/3)

- Living documentation means you see it in your browser and in your IDE
- Living documentation means you can link to it, or from it
- Living documentation means you can update it in minutes
- *Use **static sites** to publish documentation*

Acceptance Tests

- Each story or requirements shall have **acceptance criteria**
- Acceptance criteria are validated with **acceptance tests**
- Acceptance criteria is an executable specification and always up to date
- Traceability is implicit → *specification by example*

Traceability



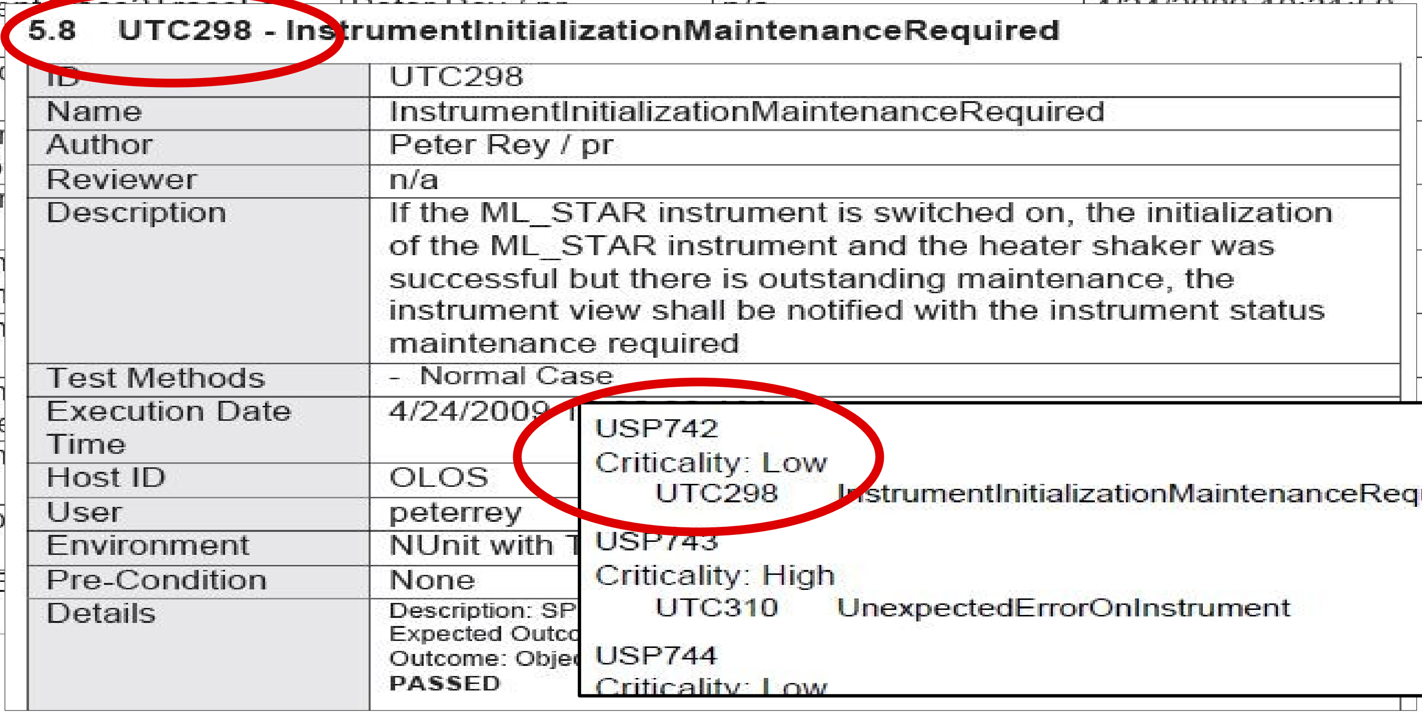
3 Verification Report

3.1 Summary

Number of test cases	passed	25
	failed	0
Total number of test cases	performed	25

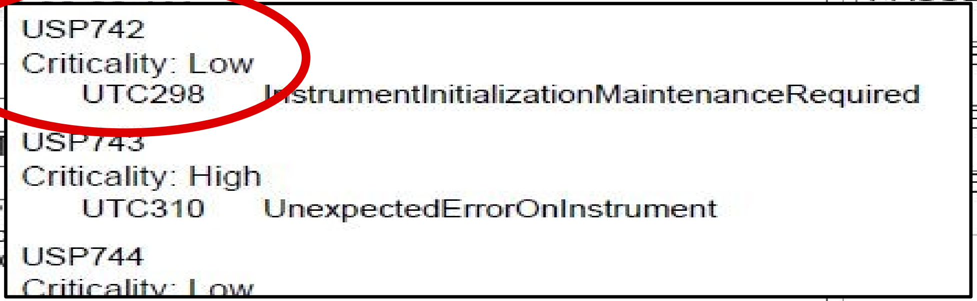
3.2 List of Test Results

TC ID	TC Name	Author	Reviewer	Date / Time	Result
UTC291	RunDailyAndWeekly Maintenance	Peter Rey / pr	n/a	4/24/2009 10:31:58 AM	PASSED
UTC292	AddInstrume	Peter Rey / pr	n/a	[illegible]	PASSED
UTC293	ConnectAuto				PASSED
UTC294	DisconnectIr PhoenixPop				PASSED
UTC295	ImplementIr				PASSED
UTC296	InstrumentIn NotifyInstrun				PASSED
UTC297	InstrumentIn				PASSED
UTC298	InstrumentIn Maintenance				PASSED
UTC299	InstrumentIn				ED
UTC300	LogExceptio				ED
UTC301	LogMethodE				ED



5.8 UTC298 - InstrumentInitializationMaintenanceRequired

ID	UTC298
Name	InstrumentInitializationMaintenanceRequired
Author	Peter Rey / pr
Reviewer	n/a
Description	If the ML_STAR instrument is switched on, the initialization of the ML_STAR instrument and the heater shaker was successful but there is outstanding maintenance, the instrument view shall be notified with the instrument status maintenance required
Test Methods	- Normal Case
Execution Date Time	4/24/2009 1
Host ID	OLOS
User	peterrey
Environment	NUnit with T
Pre-Condition	None
Details	Description: SP Expected Outc Outcome: Objec **PASSED**



USP742
Criticality: Low
UTC298 InstrumentInitializationMaintenanceRequired

USP743
Criticality: High
UTC310 UnexpectedErrorOnInstrument

USP744
Criticality: Low

Fitness Functions

- Automatic tests for non-functional requirements
- Reports provides validation for all non functional requirements
- Traceability is implicit

Source Code

- Source code should be **legible**
- Source code is **never printed**
- History of source code is managed in git
- Tools provides traceability between requirements, validation and associated source code

API Documentation

- Coding and Naming Guidelines
- JavaDoc
- Code Snippets in Java API Documentation (JEP 413)
- Part of a static web site
- Integrated in modern IDE (e.g. IntelliJ IDEA)

Git Documentation

- Git Commit Structured Comment
  - <type> - <description>
  - Type → feat, fix, refactor, chore, docs, build
  - Use “BREAKING CHANGE” in description if semantic change
- Automatic change log
- Git commit contains number of closed PBI

Configuration as Code

- Any aspect of the system shall be handled as source code
- Source code is **always** under version control
- History is always available
- Traceability and audit-ability is implicit

Static Web Sites

- Hugo, Jenkyll
- Docsy plugin for Hugo
- Pages for GitHub, GitLab, Bitbucket
  - Updated through your CI pipeline
- JavaDoc, ADR as part of your static website
- Synchronized with your git repository
- Publish daily

Exercises (1/2)

- Write an ADR – *Architecture Design Record* -
- Create UML diagrams with PlantUML
- Refresh Risk Management - e.g. ALARP Matrix -
- Read DaD Documentation Tips

Exercises (2/2)

- Ideas to Discuss
  - Explore static web site generators and Pages
  - Why is JavaDoc still relevant?
  - Are unit tests part of the documentation?
  - Explore wiki as documentation – *advantages and disadvantages -*
  - How long are developers part of a specific development team?