



Requirements Traceability

in Agile Projects

Why not traditional RT?

- ▶ Requirements and solutions will **evolve** in AP.
- ▶ Requirement deployment is a **process** in AP.
- ▶ Traceability is perceived as a **burdensome**.
- ▶ **However**, most traditional AP issues can be solved with traceability.

How AP Dev benefit

Impact
Analysis

Product
Conformance

Process
Compliance

Project
Accountability

Baseline
Reproducibility

Explicit
Context

Organisational
learning

Knowledge
Transfer

Wrap it nicely

Useful

- ▶ Risk reducing
- ▶ Prioritising enabler



Usable

- ▶ Light weight
- ▶ Highly adoptable
- ▶ Robust in a continuous environment

JITT

Just-in-time traceability



Usage

- ▶ Candidate classes
- ▶ Likelihood of linking

Upfront

- ▶ Interfacing with the VCS
- ▶ Parsing code into JITT-readable

Running

- ▶ Only on consult or change

Downsides Depends on user stories text (accuracy, precision, depth). Only for OOL.

Metrics

Rempel and Mäder 2015



Estimate Risk

- ▶ **NRR** Number of Related Req
- ▶ **ADRR** Average Distance to Related Req
- ▶ **RIF** Req Info Flow

Steps

- ▶ **Parse** artefact tracing links
- ▶ Generate traceability **graph**
- ▶ Calculate traceability **metrics**

Analysis

- ▶ Regression Model
- ▶ Prediction of Defects for unseen data

Small (600 artefacts)

- ▶ Volatility of Requirements
- ▶ Hard to justify costs
- ▶ Suggestions (with low granularity)
 - ▶ Requirements (user stories)
 - ▶ Test Cases
 - ▶ Black-box code

Large (scale, distributed)

- ▶ Difficult to rely on communal memory
- ▶ Mix of three techniques:
 - ▶ Small (req -> test cases -> black-box)
 - ▶ JITT
 - ▶ More fine-grained req -> Tagging and Association (code traces)

Safety-critical

- ▶ Compulsory traceability
- ▶ Certification requires bi-directionality:
 - ▶ Req -> design -> code -> test cases
- ▶ Full cycle into traditional traceability matrix
- ▶ Metrics



Questions?