

Traceability and Requirement Evolution

Group 3 - RE II - HS14

Agenda

- 1) How our topic fits
- 2) Paper I: Identifying outdated requirements based on source code changes
- 3) Paper II: A tool environment for (semi-) automatic impact analysis based on traces
- 4) Discussion

How our topic fits

- Requirement Traceability relies heavily on tooling and automation
- But how good are the tools?

Paper I: Problem

- Keeping req. specification = manual and expensive task
- Problem: Engineers apply changes in the software directly and leave requirements unchanged



hinders the maintainability of software systems & limits the capacity for evolution (especially when no traces between code and requirement exists)



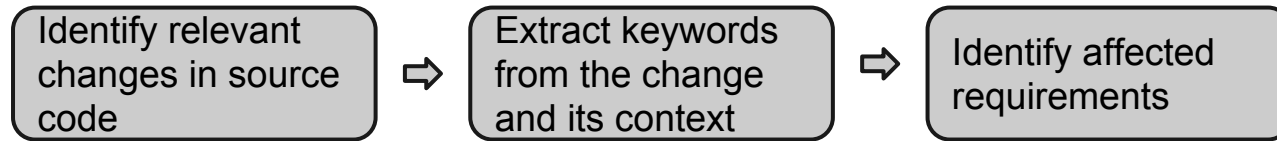
Support the update of requirement specifications when software systems evolve

Heuristics & Observations

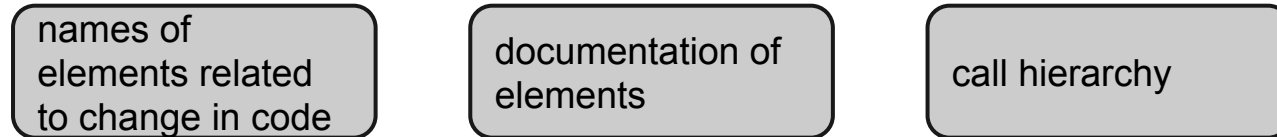
- RQ1: What heuristics can be used to identify source code changes that likely affect the external behaviour of the system?
- changes in system behaviour $\neq$ changes due to refactoring and bug fixes
- Several observations:
 - Changes in method bodies are in most cases related to refactoring / bug fixes
 - Addition of new elements likely affects requirement documentation
 - Changes in methods signature (visibility, return type) are usually related to refactoring→ not considered
 - Changes in private elements can affect the external behaviour of the system

Approach for identifying outdated requirements

- 3 step approach:
 - Identify the changes that are likely to affect the external behaviour (based on the observations and heuristics)



- For each change, a list of keywords is extracted → 3 sources:



- Trace the extracted keywords to the requirements specification in order to identify outdated ones

Results

- 1) Evaluation of effectiveness of the approach for identifying requirements related changes
 - measured by precision and recall
 - approach was able to find most requirements-related changes and excluded majority of irrelevant classes
- 2) Does the approach of change-based tracing give better results than the class-based tracing?

Difficulties



- approach can miss some relevant changes
→ compromise between identifying as many as relevant changes as possible and identifying relevant changes only)
- no generation of links being 100% correct
→ normal traceability problem: tracing is based on pure textual analysis, no consideration of intent behind it
→ false links between documents that are not related but contain similar keywords and misses the links between related documents that do not contain similar keywords
- approach was only tested for one software project (external validity?)

Paper II: Problem / Introduction

- We want to do proper cost estimation of changes (= impact analysis)
- Usually: Intuition of experts
- But: Studies show that they are usually wrong

 We look at the relationships between the document entities

Paper II: QuaTrace

- They want to automatically establish traces
- Built to work upon two other RE tools:
 - *RequisitePro* (text-based requirement management tool)
 - *Rhapsody* (CASE tools)

Paper II: Roles

The tool currently supports 3 conceptual roles:

- **Requirements Engineer**
 - elicits, documents, maintains requirements.
 - establishes traces and keeps them consistent
- **Planner**
 - Understands proposed changes
 - Looks at costs & effort (> performs impact analysis)

Paper II: Roles 2

The tool currently supports 3 conceptual roles:

- Maintainer
 - implements the proposed change

Paper II: Trace Model

The **requirements manager** establishes the **trace model**, which defines:

- tracing guidelines
 - who captures what when at with level of granularity
- templates (eg titles of chapters etc)

Paper II: Trace Model

3 kinds of traces:

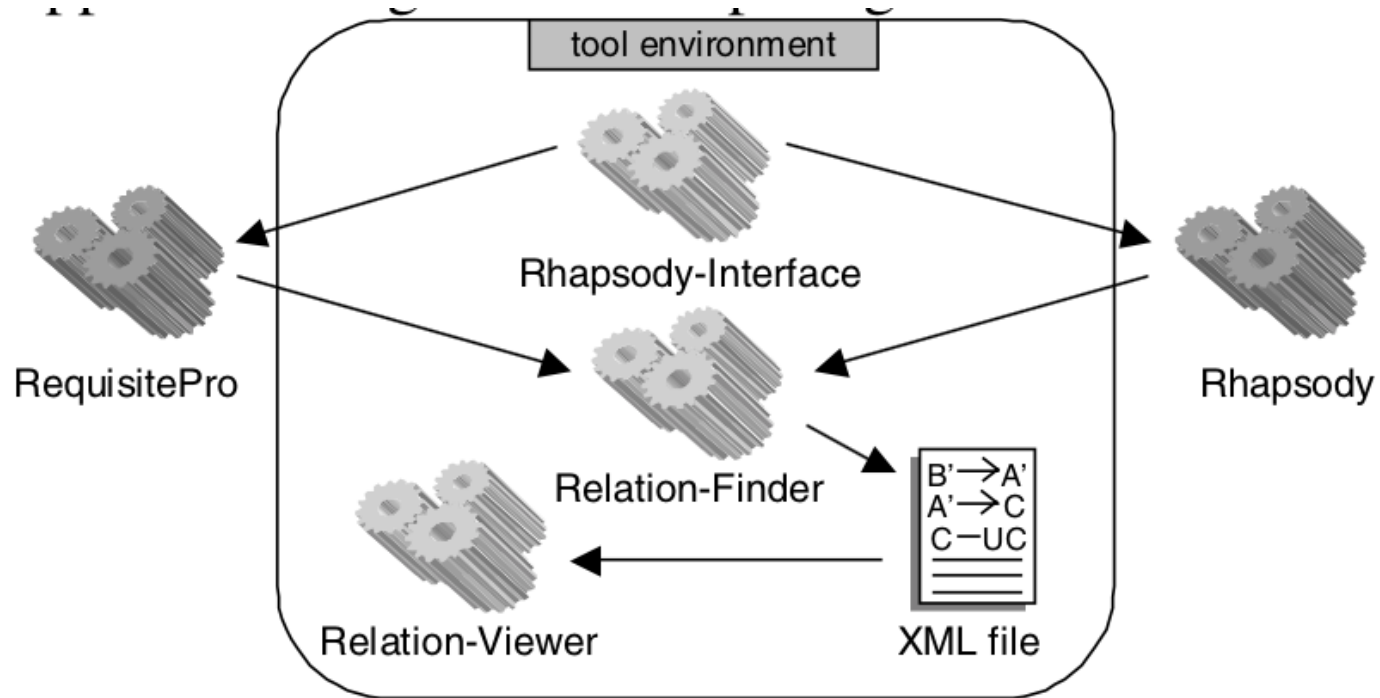
- representation (both documents document the same entity)
 - traced automatically with name tracing
- refinement (different levels of abstraction)
 - traced automatically based on hierarchy

Paper II: Trace Model

3 kinds of traces:

- dependency
 - traced automatically if entity named in document relating to other entity

Paper II: Tool itself



Paper II: Problems

Paper II: Problems

- puts a lot of effort on the requirements manager
- relies on the name tracing to work well / people to adhere to the naming