

Tracing Requirements and Source Code

Anna Hanstorp, Fabio Isler,
Adrian Gasparini, David Aggeler, Oliver Zihler

Empirical Study of Delater et al. (2013) - Motivation

- Traceability links between requirements and code are often not created
- No systematic study of using work items to capture links between requirements and code
- Study contributes valuable insights into the tracing of requirements and code

Empirical Study of Delater et al. (2013) - 3-Step-Approach

1. Traceability Information Model

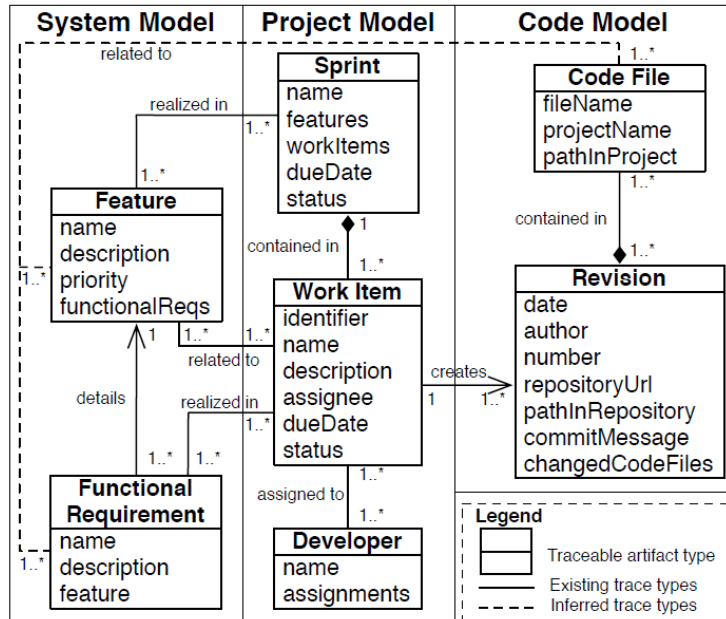
1. Traceability Link Creation Processes

1. UNICASE Trace Client (UTC)

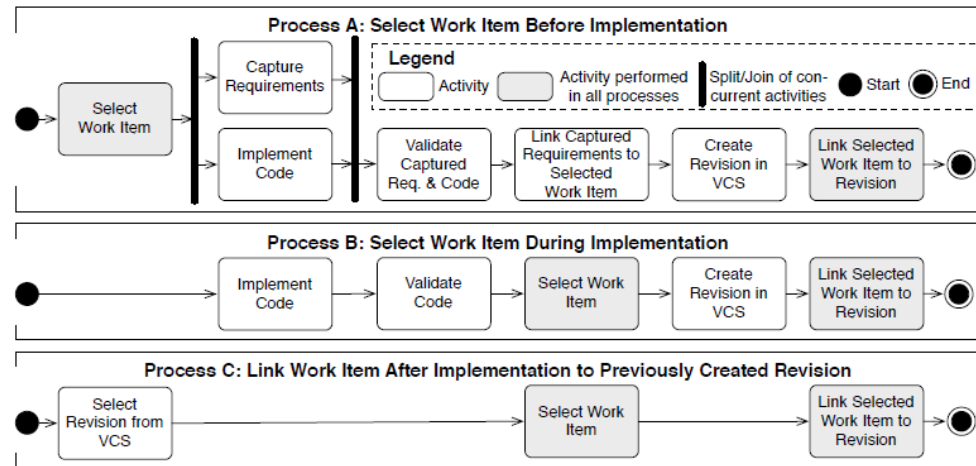
- UNICASE = Eclipse Plugin

Empirical Study of Delater et al. (2013) - Underlying concepts

Traceability Information Model



Traceability Link Creation



Empirical Study of Delater et al. (2013) - Results

- **Feasibility**
 - Precision and Recall of at least 80% without considerable fluctuation from sprint to sprint
- **Practicability**
 - Participants found that the approach was easy and practical to use
- **Conclusion**
 - The approach creates correct traceability links of high quality during development.
 - The approach and tool support was rated easy to use by the subjects

Cerberus: Eaddy et al. (2008) - Motivation

- Concern location (CL) problem: Identify the source code within a program related to a concern
 - Concern = Any consideration that can impact the implementation of a program
 - CL technique: Expert that evaluates a set of heuristics to produce a relevance judgement about the relationship between a program element and a concern
- Goal: Combine three of those experts to automate concern locating

Cerberus: Eaddy et al. (2008) - CL Experts

- Information Retrieval
 - Retrieve similarities between terms used in the concern descriptions and in the program elements
- Execution Tracing
 - Analyze the runtime behavior of a program to determine program elements activated when a concern is exercised
- Prune dependency analysis
 - Prune dependency → pruning the concern implies the element should be removed from the program, or altered

Cerberus: Eaddy et al. (2008) - Experts combined: CERBERUS

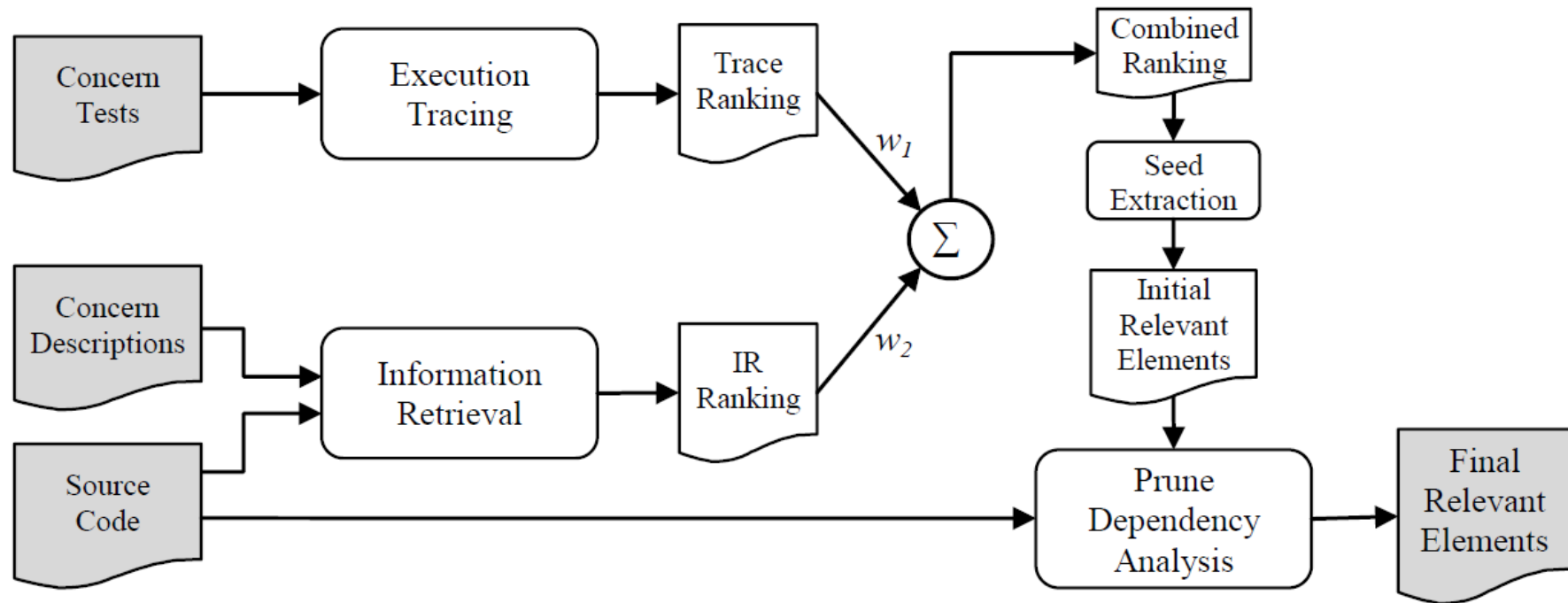


Fig 3. Overview of the Cerberus approach

Cerberus: Eaddy et al. (2008) - Case Study and Results

- Cerberus was tested against other approaches by applying it to existing software (RHINO)
- Cerberus produced more accurate results than all other combination of these methods
- Prune Dependency Analysis boosted the performance of the other techniques massively
 - Recall boost of 155% (IR) and 104% (ET)
 - Only 0.5% loss in precision

Questions?

- Personal experiences with source code traceability?
- Do you think it makes sense to have the “huge” effort of creating source code traces during software development? Does it pay off?

