

Impact Analysis with Traceability

Requirements Engineering II

Raffaele Ciriello Stefan Cortali Beat Kuster Mengia Zollinger

Agenda

Problem Description

Traceability

Involved Roles

Strategies of Impact Analysis

- Automatable Strategies
- Manual Strategies

Constraints of Impact Analysis

Metrics

Tool Support

Problem Description

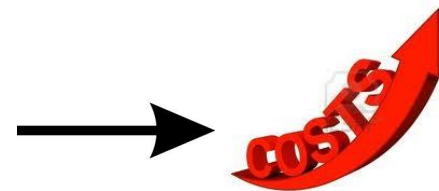
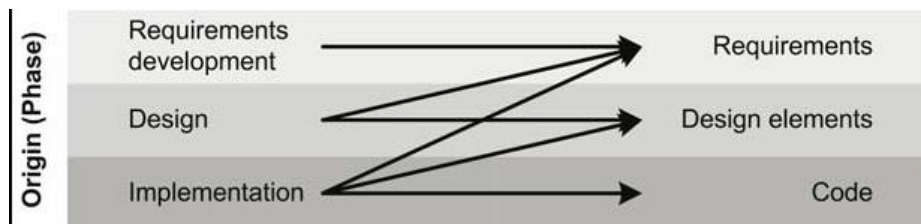
SW not fix

- Changes in technology / market
- New desires arise
- ...



Changes cause side & ripple effects

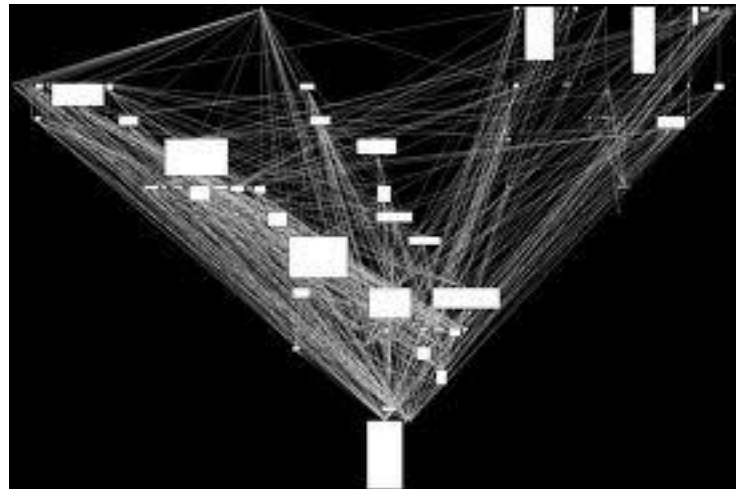
More dependencies in later stages of the project



Traceability

To support proper estimations:

→ trace dependencies between artefacts



Involved Roles

Requirement Engineer:

- Elicit requirements
- Establish traces between requirements



Project Manager:

- Estimates effort needed
- Performs impact analyses using traces



Involved Roles

Maintainer:

- Integrates changes into SW
- Adapt documents
- Discovers further changes needed



Strategies of Impact Analysis

Automatable

Traceability / Dependency Analysis

- Common: Examines relationships among entities
- T: among all types of SLOs
- DA: low-level dependencies extracted from source code
 - More mature
 - Not available until late in project



Slicing Technique

- Understand dependencies using independent slices
- Decomposition & complement slices
- To isolate impact of changes

Strategies of Impact Analysis

Manual

Design Documentation

- Architecture sketches, view-based architecture models, object-oriented UML diagrams, textual descriptions, ...
- Success depends on..
 - Knowledge & skills of analyser
 - Availability, amount & quality of documentation

Interviews

- Most common way today
- Cost-effective



Constraints

- Dependencies to non-functional requirements are hard to identify
 - Use of mentioned strategies is limited if non functional requirements are involved
-
- ➔ Split up and transform into functional requirements
 - ➔ But: not always possible

Metrics

Fasolin & Visaggio

- Inclusiveness (Estimated / Actual impacted)
- Ripple-sensitivity (Indirect / Direct impacted)

Lam & Shankararaman

- Quality deviation (before vs after change)
- Number of defects

Metrics - Example

Predicted vs actual changes:

In total 136 C++ classes

30 classes have been identified in advance

		Predictive Group		
		Unchanged	Changed	
Actual Group	Unchanged	A: 42 (30.9%)	B:0 (0.0%)	A+B: 42 (30.9%)
	Changed	C; 64 (47.1%)	D: 30 (22.1%)	C+D: 94 (69.1%)
		A+C: 106 (77.9%)	B+D: 30 (22.1%)	N: 136 (100.0%)



Many classes were overlooked

Tool Support

Limited tool support exists

- Egyed (dependency extraction)
- ANALYST (assessment in dependency-based models)
- ChAT (calculation of ripple effects)

→ But: either proof-of-concept or limited to certain aspects of impact analysis

Conclusion

- Changes are an important aspect of software development
- Linked artifacts allow easier analysis of dependencies
- Analysis of dependencies within code is supported by tools
- More research must be done in other aspects of dependencies analysis