Managing the co-evolution of software artifacts

J.M.A.M. Gabriels[1], D.H.R. Holten[2], M.D. Klabbers[1], W.J.P. van Ravensteijn[2], A. Serebrenik[1]
[1]Laboratory for Quality Software, Eindhoven University of Technology, [2]Synerscope B.V.
{j.m.a.m.gabriels, m.d.klabbers, a.serebrenik}@tue.nl, {danny.holten, wiljan.van.ravensteijn}@synerscope.com

Software development projects are virtually always carried out under pressure. Planning and budgets are tight, room for errors is non-existent and the pressure to deliver is high. Natural questions for (test) managers arise, such as: “When have we tested enough?” and “How many tests do we have to redo for this new version?”. The naive answer would be: “when we have convinced ourselves through testing that all requirements are satisfied.”. Unfortunately, attaining maximal confidence with minimal effort is not easy.

In order to convince ourselves that the system does what it is supposed to do, tests are required. Requirements, design and code change during the development of software. As a consequence, tests need to change as well for in the end we want to ensure that all requirements and risks are adequately addressed with tests. For this, tests at different levels of abstraction and for different software artifacts are required and need to be managed.

To relate user requirements, design, code and tests, traceability matrices are often used. Traceability allows to link elements from different software artifacts, like requirements, design components and code components, to each other and to test cases. As a result, traceability can be used to analyze for example how well software artifacts are covered by test cases. Because a requirement leads to design components and eventually to code, tests are needed at each stage. Traceability can tell us how well test cases cover different software artifact elements. This information can be used to uncover mistakes in software artifacts at an early stage and actively manage the development and test efforts. Unfortunately, traceability information is often spread out over multiple artifacts and describes only the current situation.

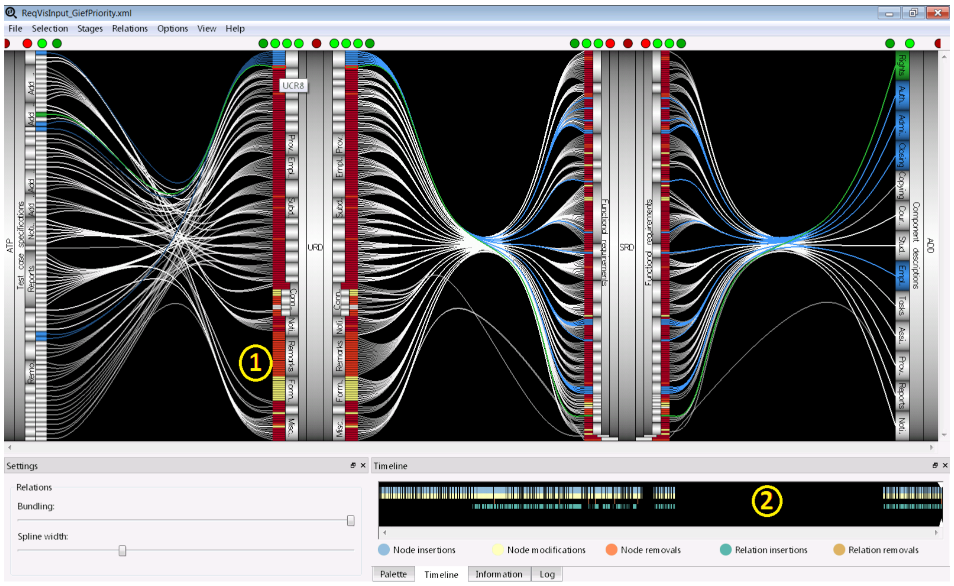

Figure 1: TraceVis tool with traceability information from a student capstone project at the Eindhoven university of Technology. The edge bundeling technique makes it easy to spot deviations. The gap labeled (1) shows some medium and low priority user requirements not covered by acceptance tests. The gap labeled (2), located in the timeline, shows that test cases were added very late in the project. The gap itself relates to implementation activities in which there were no changes to the shown software artifacts.

TraceVis, a visual analytics tool based on the master thesis of Van Ravensteijn[1], combines the traceability information of multiple software artifacts in an interactive way. Furthermore it provides a way of assessing the evolution of traceability between artifacts though a timeline. Figure 1 shows the traceability between four, vertically placed, hierarchical software artifacts: acceptance test plan, user requirements document, software requirements document, and architectural design document. Each line represents a link between elements of two artifacts. Priorities and/or risks can be marked with colors and hierarchies can be collapsed and extended. Furthermore, the edge bundling technique bundles similar relations in the middle, clearly showing deviations. Already at first glance, we can see points of attention in Figure 1: a gap in requirement coverage and a gap in the timeline.

The evolutionary traceability information allows us to see how well tests cover artifacts and whether risks are sufficiently tackled. It gives insight in the balance between tests, priorities, and risks and can support decision making in assigning test effort. Furthermore, it can help in determining which tests need to be redone when a certain component or requirement changes. The insight in the co-evolution of software artifacts and associated tests makes it possible to actively manage test effort from an early stage on.

[1]W.J.P. van Ravensteijn, *Visual traceability across dynamic ordered hierarchies*, 2011, Eindhoven University of Technology