A Visual Approach to Traceability and Rationale Management in Distributed Collaborative Software Development

T. Hildenbrand and A. Heinzl and M. Geisser and L. Klimpke and T. Acker
Department of Information Systems I
University of Mannheim, 68131 Mannheim, Germany
{thildenb; aheinzl; mgeisser; lklimpke; tacker}@rumms.uni-mannheim.de

Abstract: *Traceability* and *rationale management* are utmost critical, especially in distributed collaborative software development projects, due to a lack of mutual workplace awareness and informal coordination among the participating stakeholders. Therefore, this paper presents the rationale behind and implementation of a conceptional design for a novel approach to traceability and rationale management in distributed settings. In doing so, an innovative solution for extracting, visualizing, and analyzing the relationships between requirements and other key artifacts as well as responsible stakeholders is designed based on business needs within the software industry. Thus, the main objective of this paper is to instantiate the underlying conceptual considerations and methodological guidelines with the help of a comprehensive tool infrastructure particularly adapted to distributed settings.

1 Introduction

Traceability and *rationale management* represent utmost important tasks with respect to the governance of geographically distributed software projects [HGK07, dSHR07]. *Traceability*, in particular, denotes the ability to follow the relations between artifacts, such as requirements, design documents, or source code and responsible stakeholders [GF94]. *Traceability management* (TM) therefore involves activities including the identification, analysis, and maintenance of these relationships [KS98]. *Rationale management*, on the other hand, addresses the documentation and usage of rationale information regarding design and change decisions within software development projects [DP01]. In combination with the concept of *value-based software engineering* (VBSE), denoting that not all software artifacts can be attributed identical (customer) *value* and thus should be represented accordingly [EBHG05], enhancements to software project decision support can be achieved through the application of value-based and integrated end-to-end *traceability and rationale management* (TRM) covering the *entire* software development life cycle. Therefore, especially within spatially and temporally distributed projects, an increase of development efficiency and effectiveness can be achieved by providing (bi-directional) traceability information [dSHR07]. Moreover, an intuitive representation of the data by means of graphical visualization can be especially helpful and effective [dS05]. This quality feature is also required by the *Capability Maturity Model Integration* (CMMI) process standard, for instance, as well as most other important (software) industry standards.

Therefore, the main *goal* of this paper is to document the design and implementation of a novel *tra*ce *vis*ualization approach called "TraVis", which extracts data related to different artifacts, activities, and users from collaborative software development environments in order to support visual analysis and collaborative management of the relations between these entities, while enhancing the platform's functionality by graphical visualization and analytic filtering mechanisms.

The following section 2 will, at first, briefly introduce the design and implementation *methodology* that was chosen for the design of the solution approach. After that, the main functional requirements for the novel solution will be outlined in section 3 before section 4 addresses the implementation platform and basic technologies used for realizing TraVis as part of a larger solution architecture. According to section 2, implementation details regarding the tool and application examples are given in section 5 to provide an initial evaluation in terms of feasibility of the proposed solution. The last section 6 provides a summary of findings and gives an outlook on future work.

2 Methodology

To eventually implement a comprehensive TRM solution, an *object-oriented* design methodology is chosen [Kru04], since trace relation models represent complex interrelated real-world objects such stakeholders and artifacts with different attributes such as descriptions, version numbers, and change rationale (see underlying information model in figure 1). The concurrent requirements management process is mainly driven by common TRM process activities and fields of application which in turn can be regarded as functional requirements or *"use cases"*[1] in the figurative sense guiding the further development process (see also table 1).

Moreover, this object-oriented approach aims at an architecture consisting of independent application *components*, thus, abiding by the main software engineering (SE) best practices formulated by [Kru04], i.e. (1) *iterative* development in combination with different evaluation steps and feedback loops to gain utmost utility of the novel approach [HMPR04], (2) use case-driven and *collaborative requirements management* involving several research groups and other stakeholders, as well as (3) designing a *component*-based solution architecture while separating three independent *system* components, namely (a) data model and persistence layer, (b) collaboration platform to support the overall TRM method, and (c) specialized tool support for traceability capturing, representation, and analysis (TraVis).

Before introducing the actual solution design and implementation, the following section summarizes the major functional requirements that have been gathered by means of several preceding studies.

[1]In this paper, the requirements will not be represented in the form of actual use cases as, for instance, defined by [Kru04] and [BME+07].

3 Functional Requirements and Related Work

In the following paragraphs, requirements for the novel solution elicited in preceding studies are presented in the form of distinct application areas—including a *procedural description* with respect to TRM process steps, actors, requirements *rationale and origin* [Kru04] as well as related work (see also [SZ04] and [Hil08]). Origins and rationale behind these fields of application are based on a broad analysis of literature review data [HRH07] as well as empirical requirements elicitation (cf. analysis results in [dSHR07] as well as [Hil08]).[2]

The main functional areas and requirements regarding TRM in distributed settings derived by means of literature reviews and tool evaluation have been complemented by conducting two observational case studies encoded MBL and VCI[3] (see table 1 and [dSHR07]). Table 1 presents an overview of the three main requirements categories: (1) change management in general (CM), (2) trace *capturing* and maintenance (TCM), and (3) trace representation and analysis. Besides common literature [HRH07, HRG$^+$08], these findings are substantiated by the MBL and VCI studies [dSHR07].

Table 1: Overview of Requirements from Case Studies and Reviews (CM = change management; TCM = trace capturing and maintenance; TRA = trace representation and analysis)

Requirement	**Cat.**	**Source/Reference**
Automatic change notifications	CM	MBL and VCI studies
Workplace awareness support	CM	[DCAC03] and MBL study
Traceability information supply	CM	GSD study, literature review
Collaborative trace capturing	TCM	VCI study, literature review
Collaborative trace maintenance	TCM	VCI study, literature review
Rationale management support	TCM	VCI study, literature review
Alternative visual representations	TRA	VCI study, literature review
Predefined views and filters	TRA	VCI study
Adjacency graph analysis	TRA	VCI study, literature review

Based on these functional requirements, the main functionality supported by the TraVis solution is presented with respect to related work. The requirements elicited here correspond to the main areas of TRM, i.e. (1) *CM and impact analyses* as major use case for utilizing TRM information as well as the process of capturing and analyzing this information. However, in order to support CM in general, advanced methods for trace capturing (TCM) and analysis (TRA) are required as well. In addition to analyzing change impact, traceability information is also critical for (2) *project status reporting* and (3) *overall documentation of traces* by developers and other team members. This functionality can further be subdivided into (a) capturing and generation, (b) storage and representation, (c) analysis and maintenance use cases [Hil08]. In addition, this information is vital to project and

[2]For a more detailed requirements analysis and review of related work see [Hil08].

[3]Real names have been changed by the authors and these acronyms simply represent variable names without any semantic meaning.

program management for (4) *performance monitoring* purposes. Enhanced information supply can additionally assist (5) *post-specification development* tasks that will also be discussed in the following paragraphs.

Change Management and Impact Analysis. *Change management* (CM) tasks turn out to be the prevailing field of application for traceability information—e.g. for change propagation, generating notifications, and facilitating *impact analyses* in particular. Especially in the latter case, traceability information is critical in case of late changes for determining (1) directly and (2) indirectly affected artifacts and thus be able to (3) estimate the resulting *overall* costs of changes proposed in order to decide whether the change can be conducted or not [KS98]. Most often, change impact analysis pertains to changing *requirements* after an initial specification has been defined—e.g. in the form of change requests posted by different stakeholders [Som07]. These include the integration of new requirements as well as deleting and changing existing ones [Poh07].[4] The positive effects of sophisticated traceability information on impact analysis quality and efficiency has also been substantiated empirically (cp. for instance [LS96] and [LS98]). Automatically generated notifications as well as visual representations of dependencies can substantially support impact analyses [EH91].[5]

Project Status Reporting. CM also includes requirements implementation *status tracking and reporting*, i.e. capturing and analyzing the requirements' implementation status in post-specification project phases [Sch02]. In order to be able to determine and analyze the exact status of one particular requirement's implementation, *continuous horizontal traceability* from the *software requirements specification* (SRS) via architectural models, source code, and test cases must be established [Poh07]. This also requires information about contribution structures [Got95], i.e. authorship information [dSDR$^+$04], and underlying rationale pertaining to post-specification artifacts and enables ensuring that all current project activities are based on actual customer demands and thus create customer value. Again, adequate quantitative data and according *visual* representations can be seen as possible approach to supporting this task. Moreover, visualizations not only facilitate inter-developer communication and project management, but also foster customer understanding and thus eventually system *acceptance*.[6]

Overall Project Documentation Support. The overall project documentation subsumes information necessary for both impact analysis and status reporting tasks. Taken together, a project's documentation corresponds to the overall *traceability network* containing both pre- and post-specification information with respect to artifact relations, design and change rationale, as well as contribution structures [KS98]. Project documentation tasks, therefore, pertain to the *entire* TRM process from capturing via storage and representation to

[4]Moreover, traceability information facilitates identifying cause and estimating the impact of bugs within the scope of software *maintenance* and *re-engineering* of legacy systems [Poh07].

[5]Current RM tools, such as DOORS and CaliberRM, do not provide full impact analysis support with respect to a complete analysis of all post-specification artifacts [GHR07].

[6]As it is the case for impact analysis, current RM tool do not allow a continuous post-SRS status tracking.

analysis and maintenance [Som07]. As has been argued before, disposing of relevant traceability information is vital to numerous other TRM-related tasks and can in turn facilitate distributed collaboration on the whole. Especially with respect to distributed development scenarios, current RM and SE solutions cannot provide integrated information and tool support [HRH07, GHR07, HRG$^+$08].

Project Documentation Capturing and Generation. As regards *capturing* traceability network information, in particular, a *collaborative* approach based on one central repository is suggested both in literature and practice (cf. findings in [dSHR07]). Moreover, a common *metamodel*, including artifact entities, traces, and semantics as well as methodological guidelines and policies in the form of a traceability *manual* help coordinating this activity. Automatically *generated* suggestions for candidate links, on the other hand, can provide additional decision support, but are not considered here any further, since so far only research-in-progress solutions exist, which most often still require manual intervention and/or artifact description constraints [Hil08].

Project Documentation Storage and Representation. Based on the assumption of a central *storage* of traceability information, all distributed project stakeholders can be provided with adequate *representations* of relevant extracts retrieved by means of filtering and search techniques. Due to the complexity of large-scale distributed software projects *visualizations* facilitate stakeholder communication as compared to standard list and table representations. Currently, the information necessary for end-to-end TRM can only be provided by collaboration platforms [HRH07, HRG$^+$08].

Project Documentation Analysis and Maintenance. With respect to TRM information *analysis and maintenance* support, filtering and search mechanisms need to be complemented by more advanced *visualization* and analysis methods such as adjacency graphs for more systematic impact analyses. However, current RM tools and collaboration platforms usually provide analysis functionality only based on matrix and linked list representations (cp. above and findings in [GHR07, HRG$^+$08]).

Project Monitoring and Inspection. Within the scope of project monitoring activities, which are mostly conducted by project managers and other high-level stakeholders, the overall development process in terms of *who has done what and when* (process data) needs to be traceable at any given time [DP98]. On this basis, project managers have to be able to assess and report individual and team *performance*, *balance* the overall work load, and maintain reasonable *division of labor* while also considering implementation status reports (see above). To be able to do so, end-to-end traceability information is utilized to understand relations and particularly dependencies between artifacts and the stakeholders involved. Moreover, process data on tasks performed, resources consumed, and other quality measures can be utilized for general project planning and control [DP98].[7]

[7]Current development environments mainly focus on *source code* monitoring [HRG$^+$08]. Other approaches, such as Ariadne [dS05], monitor socio-technical relations, but again only based on *source code* dependencies

Post-Specification Requirements Management. Requirements management tasks also comprise validation, verification, testing, and establishing standards compliance [SZ04]. Contribution structures, for instance, can be utilized to identify and involve relevant stakeholders into *validation* activities. As regards *verification*, refinement, dependency, and satisfiability relations allow for ensuring that all requirements specified have been allocated to ensuing implementation tasks and corresponding artifacts such as models and code. Similarly, traceability relations can be used to check the existence of appropriate test cases for verifying different requirements [SZ04] and to retrieve those.

Other Post-Specification Tasks. Traceability information and management capabilities can also support *other general SE tasks* such as finding the right stakeholders for *communication* and *coordination* purposes, artifact *understanding* and software *reuse* as well as software *maintenance* and thus support SE decisions due to better overview, visualizations, and analysis methods, e.g. for finding relevant information and/or contact persons more quickly. *Group or team awareness* is crucial in distributed settings due to the volatility of communication networks and partially sparse interactions among related stakeholders [HMFG00]. Appropriate visualization of relations between users and/or artifacts (cp. also [dS05]) can thus lead to more purposeful collaboration.

Artifact understanding, informed software reuse, and maintenance can also be accounted to general SE tasks that can benefit from traceability information. Improved traceability supports different stakeholders in *understanding* artifacts and their respective contexts even when not having contributed to their creation [SZ04]. For full artifact comprehension, *rationale* capturing, representation, and analysis capabilities are critical as well [RJ01]. Furthermore, requirements dependencies can support software *reuse* in that similar requirements are identified when the stated requirements are compared with existing requirements for indicating possibly reusable components from different artifact stages. In general, similarities between artifacts on different levels of horizontal abstraction along the software development life cycle can be utilized to manage application frameworks and software product lines [SZ04].

4 Implementation Platform and Technologies

The following sections briefly introduce the underlying information model implemented in the collaboration platform which the TraVis tool is based on. Moreover, other technologies used for the implementation of the TraVis solution approach and relevant implementation details are also outlined.

and code authorship information.

4.1 Traceability Information Model

Based on general trace information models, the underlying platform's inherent information model and its accessibility via the Web service API have to be adapted to TraVis. Compared to the underlying platform information model, some interrelations had to be simplified due to the vendor's practical restrictions. However, all vital elements of the TraVis information model are represented—*tracker items* and all different kinds of *artifacts* (documents, wiki pages, source code, etc.), for instance, can be tracked by distinct *realization states* and *versions* as well as categorized by embracing trackers or containers, respectively. *Rationale* information can be added by means of (wiki) comments to types of associations between tracker items and artifacts.

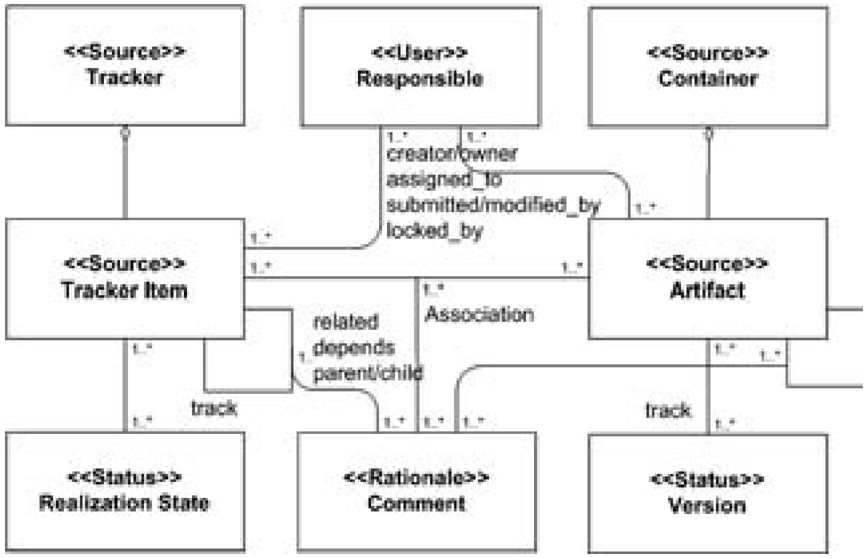

Figure 1: CodeBeamer Information Model

As can be seen in figure 1, the model differentiates between four basic types of *associations*: `depends`, `parent`, `child`, and `related`. Moreover, responsible *users* can take on various roles as they are associated to certain tracker items or artifacts. These include `owner`, `creator`, `assigned_to`, `submitted_by`, `modified_by`, and `locker` (someone who has locked a particular artifact for non-concurrent editing). *Change requests* are modeled as tracker items in a so-called change request tracker and therefore not represented separately in the CodeBeamer model. Furthermore, items in change request and requirements trackers also dispose of wiki-based rationale descriptions directly attached. For further adapting the CodeBeamer information model, specific project templates with predefined tracker structures are created. To be able to *connect* to the collaboration server and *extract* the relevant traceability information, TraVis uses CodeBeamer's Web service API (for a detailed description of the packages used see [HGK08]).

4.2 Implementation Technologies and Details

To be able to connect to the collaboration platform over the Internet and, thus, provide a Web-based user interface, the *Java WebStart*[8] (JWS) technology by Sun Microsystems is chosen. Moreover, JWS allows iterative updates to be able to extend the current prototype's functionality while keeping entailing network traffic low. This is and will be particularly critical for evaluating the overall solution with globally distributed stakeholders [Hil08].

For extracting the traceability information from the *collaboration platform*, the *Hessian*[9] *binary* Web service protocol provided by the underlying CodeBeamer platform is utilized. Besides its most advanced association mechanisms, link semantics, and wiki engine integration, the platform has initially been chosen for the prototypical TraVis implementation due to its fast and flexible Web service application programming interface (API) which has also been extended collaboratively with the vendor in the course of this research.[10]

The CodeBeamer platform provides an integrated *wiki engine* for (a) annotating and commenting on tracker items and other artifacts, e.g. to add *rationale* information, and (b) for creating self-contained wiki documents. Traceability information captured via wiki pages and comments can be in turn *represented* as interlinked wiki content in CodeBeamer's Web frontend and *analyzed* via the Web service API (see also [HGK08]). As for the WebStart application's user interface layer, TraVis uses *Java Universal Network/Graph*[11] (JUNG) framework. The JUNG architecture is designed to support a variety of representations of entities and their relations, such as directed and undirected graphs, multi-modal graphs, graphs with parallel edges, and hypergraphs.

On basis of the different Web-based technology platform just described, the most important details pertaining to the implementation of TraVis are documented in figure 2. However, the focus of this paper is on the TraVis part of the overall solution architecture, i.e. visual representation, analysis, and maintenance functionality, and thus particular the use cases specified in section 3. The overall *solution implementation architecture* underlying this paper also includes an adapted collaboration platform as well as a separate source code management (SCM) repository (see figure 2). Moreover, particular semantic information can be added for more efficient retrieval of certain objects (cp. class descriptions in [Hil08, HGK08] and cf. information model in figure 1).

5 Tool Implementation and Application Scenarios

In this section, the functional areas presented in section 3 are substantiated to demonstrate how the underlying requirements are implemented by the TraVis solution and, thus, the

[8]`http://java.sun.com/products/javawebstart/` (2007-10-16).

[9]`http://hessian.caucho.com/` (2007-10-16).

[10]CodeBeamer has been analyzed and compared to other commercially available collaboration platforms, e.g. in [RGBH07], [HRH07], and [HRG+08].

[11]`http://jung.sourceforge.net/` (2007-10-16).

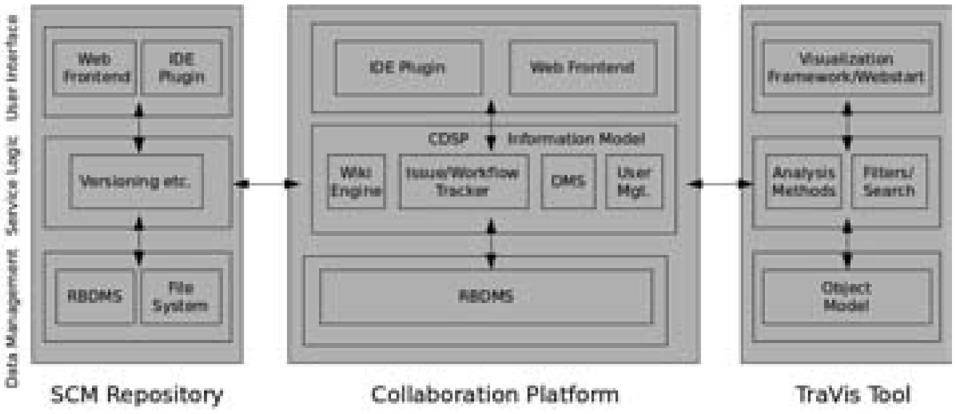

Figure 2: TraVis' Embedding Solution Architecture

feasibility of the approach as well as the applicability of the implementation is *evaluated* in terms of an initial demonstration or proof of concept. In doing so, TRM activities concerning (a) representation and visualization as well as (b) analysis and maintenance can be distinguished.[12]

5.1 Trace Representation and Visualization

TraVis provides different means of displaying the traceability network graph extracted from the development platform. Starting with an empty panel, the application's *user interface* requires the user to either manually select and deselect different element and association types or use various menu options for filtering, searching, and transforming the graph.

Manual Selection. The checkboxes of the TraVis *user interface* allow for a fine-grained configuration of the traceability information to be shown in the center panel. According to the CodeBeamer information model, the following elements are available for visualization: (1) *users* (stakeholders), (2) *issue trackers*, *tracker items*, and *attachments* (3) *documents* and *folders* (as parts of the document management system), (4) *forums* and single *posts*, (5) *wiki pages*, as well as (6) *source files*. When selecting particular project elements, only the resulting combination types of associations are activated while all other relations are shaded and not available. Accordingly, when removing certain information elements, these changes update the active options of manual selection. When checking or un-checking certain association types, these are added or removed, respectively. Checking

[12]The methodology for collaboratively *capturing* traceability information by means of the functionality incorporated in the underlying platform is not in the focus here, for further details see [Hil08]. Moreover, [Hil08] also documents three independent evaluation cycles and provides substantial evidence for the approach's *utility* in distributed scenarios.

Add/Remove all displays or hides all relation types possible. Moreover, *edge labels* can be manually added by means of the respective checkbox in the *Options* menu. Therefore, the checkboxes are the universal user interface for custom analyses concerning all major functional TRM requirements considered in this paper.

Element Filters. In addition to manually removing elements and associations from the graph, TraVis also provides numerous predefined filters for reducing network complexity and facilitating *inspection* tasks. To be able to do so, *inactive* users, other *disconnected* elements with no relations, as well as *closed* tracker items representing finished tasks can be filtered out automatically. Moreover, tracker items can be added to the graph, removed, and highlighted with respect to different tracker *categories*, implementation *phases*, and realization *states*. As has been mentioned earlier, TraVis complementarily analyzes links between wiki pages and thus these can be added or filtered out by checking the *Wiki Links* filter option. In addition to manually selecting and deselecting graph elements, filters are thus also universally applicable to a variety of TRM tasks such as status reporting and general traceability information management.

Predefined Views. Besides selection options and filters, TraVis also disposes of many other possible choices of graph representation. To facilitate overall usability and reduce complexity of the application, TraVis currently provides the following predefined views which can be easily adapted and extended to other use cases by means of TraVis' object-oriented and component-based architecture: (1) An "*Ego View*" with all elements associated to one particular stakeholder, (2) "*Editing (basic)*" and (3) "*Editing (all)*" showing elements that can be linked visually by TraVis, either regarding reduced or full project complexity, (4) "*Task Distribution*" with stakeholders and shared artifacts (see example below), (5) "*Major Artifacts*", (6) "*Tracker Structure*", as well as (7) "*Project Management Analysis*" which display tasks associated with certain stakeholders.

The *task distribution* view reveals who is doing what as well as *collaboration structures* formed by shared artifact relations between stakeholders, which is the basis for further analysis methods such as social network inspections (cp. next section as well as 6). In the case of this particular task distribution view, the implementation consists of four different graph options: (1) the *types of elements* and associations included (here tracker items, users, and their interrelations), (2) value-based *vertex sizing* (instead of uniform sizing, see subsequent section), (3) *mouse mode* (picking), and (4) graph *layout* (cp. also the following section). However, TraVis' architecture allows for adapting and creating views with more or less options very easily, e.g. by simply adding a new radio button in the *Views* menu.

Search Functions. For analyzing complex traceability networks, TraVis implements two complementary types of searches: (1) an integrated *type-ahead* search and (2) a *search menu* for adding and highlighting particular nodes. The former search function can be utilized to spot and find individual artifacts in very dense and complex graphs. In doing so, the type-ahead search already highlights graph items while the user is still typing, i.e. the search process can be gradually concretized, while search results are instantly displayed in the center pane. For this and all other types of searching the traceability network information, the artifacts' titles and major description attributes are indexed and thus searchable. These search results are categorized according to the types of elements found—such as tracker items and documents in this case. By clicking particular elements in the result tree a graph can be built up from scratch or complemented (cp. also filtering mechanisms above). The *Find Artifact (Highlight)* function operates analogously to the type-ahead variant.

Transformations. TraVis also provides different graph transformations such as distortion, rotation and zooming. With respect to graph distortion, different lenses are defined with both hyperbolic and linear magnifying optics. To further reduce the graph's complexity, the different lens modes can be combined with all other options, filters, and views described so far. For zooming and rotating the graph both mouse and keyboard shortcut controls are provided in addition to the respective items in the *Options* menu of TraVis' user interface.

5.2 Trace Analysis and Maintenance

The implementation details described so far focused on visualizing the information retrieved from the platform for universally supporting different software engineering decisions and use cases. On top of that, TraVis also provides tool-supported methods for visual traceability network *analysis* and *maintenance*. Therefore, the following paragraphs explicate the TraVis functionality for exploring and analyzing the graph by means of additional information visualizations as well as visual editing capabilities.

Gradual Graph Exploration. Since the size of real-world traceability graphs are a major concern in TRM, *incremental exploration* techniques are a good solution for analyzing huge graphs originating from one particular element (cf. [HMM00], p. 37). This element can either be specified by a change request or found by the search and add function described above. The context menu option *Show connected vertices* adds all elements connected to a requirement resulting of a search operation, as well as the respective relation types as edge labels (`related`, `depends`, etc.). By applying this method in turn to one of the newly added elements, the graph is gradually explored from its origin. Furthermore, it is possible to show *all* connected vertices, i.e. the complete *adjacency* graph, of one particular start node by means of the corresponding option in the context menu (see figure 3). This type of information visualization therefore identifies artifacts directly and indirectly affected by changes to the focal artifact and thus facilitates *impact analyses* (cf. section 3). Moreover, *status reporting* is supported by enabling to follow the horizontal trace path of one requirement up to the current realization state. Additional artifact information is displayed on mouse-over operations in the form of tooltips. Vice versa, TraVis also allows for *removing* particular nodes.

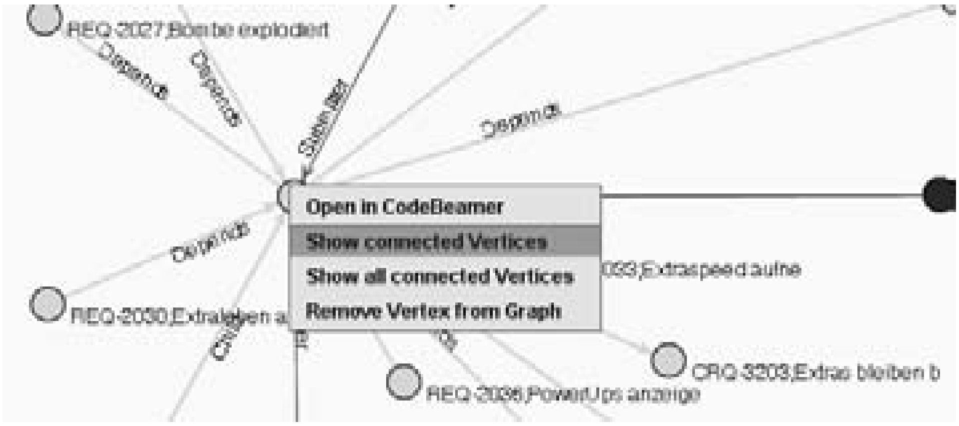

Figure 3: Gradual Graph Exploration Functionality

Trace Network Analysis. Besides manual trace network analysis by means of the check-boxes and options explicated above, TraVis also provides a comprehensive network analysis function activated by choosing the predefined *project management analysis* view. This non-visual "view" analyzes the number of traces among the different types of trackers and displays a detailed dynamically generated list of requirements and relations between tracker items. In doing so, customized trackers in addition to standard requirements, change requests, and bug trackers are considered as well. This function facilitates auditing the overall *project documentation* as well as determining traceability as a quality measure (cp. also data collection and measurement procedures in evaluation section. Again, the *show connected vertices* can also be applied to particular users and thus *monitored* what artifact development activities they are currently involved in. Using the *task distribution* view, on the other hand, also supports project monitoring and *controlling*—e.g. by spotting out project members that do not participate in any collaborative activities. Furthermore, the function *Team Statistics* in the *Project* menu returns a list of all project members' relations to certain project elements such as tracker items and documents, for instance, which also enables project managers to compare and assess the developers' collaboration intensity.[13]

Value-Based Node Sizing. Also mainly for *project management* (monitoring and controlling) purposes, TraVis implements variable node sizing algorithms for indicating customer value based on requirements analysis results such as priorities and other value measures. The customer value assigned to the requirements is then propagated to related and dependent artifacts such as design documents and source code—not including stakeholders. Therefore, the initial heuristic algorithm for calculating the node sizes has been improved by adapting the *PageRank* algorithm as also applied by the *Google*[14] search engine

[13]It has to be noted though, that tools such as TraVis can create a possibly unwanted form of transparency from the developers' perspective and raise data protection concerns that are beyond the scope of this research.

[14]`http://www.google.com/` (2007-10-20).

to the requirements of TraVis' solution architecture [PBMW98]. The PageRank algorithm is based on the assumption that nodes in a network, in this case artifacts and tracker items, are more relevant or *valuable* according to the number of references by incoming relations and their respective values. It has been shown that the values within a network converge after a finite number of iterations [PBMW98]. To be able to do so, TraVis initializes the nodes other than requirements with a value of $\frac{1}{number(nodes)}$ and defines a constant d[15] to accelerate convergence. The *new* value of one particular node is thus calculated as the sum of the related nodes' start values while the node's new *start* value is determined by $(1 - d) + d * new\ value$.[16]

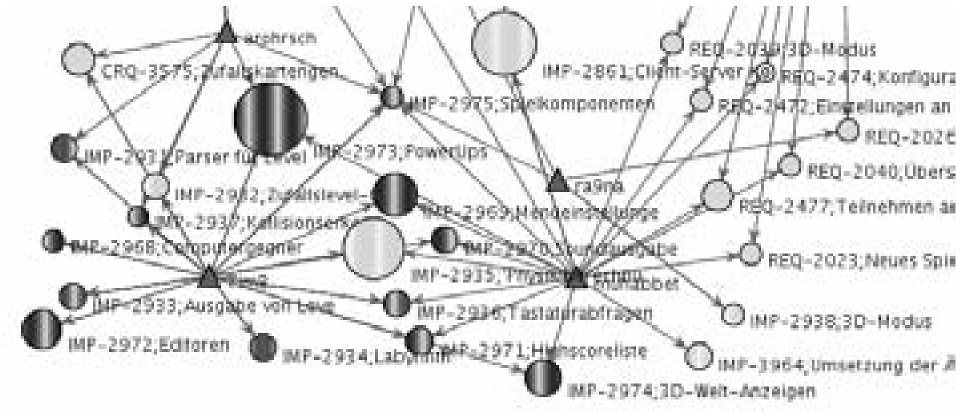

Figure 4: Extract from a Value-Based Task Distribution Graph

Value-based node sizing and the customer values written back to the platform as additional attributes, thus, provide decision support for prioritizing artifact-related activities according to their customer value [EBHG05] and, therefore, iterative as well as agile methods (see figure 4 for an extract of a value-based graph). Furthermore, customer value information also complements and quantifies the overall projects awareness as well as personnel turnover decisions (cf. section 3). Figure 4 also depicts how the different artifact types are encoded in different colors and patterns in order to provide even better visual information and decision support.

Rationale Information Management. Besides customer *value*, the artifact nodes of the traceability network carry a lot of additional information which can be utilized to comprehend justifications behind design as well as change decisions and version history—i.e. *rationale* information. To be able to prepare and provide this artifact context information

[15]Using a constant d is recommended by [PBMW98] and has been calibrated here to a value of 0.85 by means of the data from early evaluations and open source projects on the CodeBeamer-based JavaForge platform: `http://javaforge.com/` (2007-10-20). Currently, the TraVis implementation of PageRank converges after less than ten iterations for most projects. However, to provide some safety buffer, TraVis calculates 15 iterations.

[16]cf. [PBMW98] and [HGK08] for a more detailed description of the algorithm and calculation examples.

in an easily processable manner, the graphs are complemented by a so-called *element information* pane which displays a tree visualization of additional *artifact* or *stakeholder* information depending on the node currently selected in the center pane. While user element information is mainly useful for *monitoring* purposes, artifact rationale information facilitates both *impact analyses* and general *project documentation* and maintenance activities.[17]

Editing and Platform Synchronization. Starting with one of the predefined editing views (cp. section 5.1), for instance, or after manually selecting mouse mode *editing* in combination with any other view or filter, allows for removing and creating new associations between elements of the graph (see example in figure 5). This is conducted by simply dragging and dropping a line from one node to the other. As can be seen in figure 5, an association comment and type can be specified before the edge is added to the graph. Newly created edges as well as deleted ones are first recorded by TraVis' internal object model and later committed as a complete transaction to the platform by clicking on *Submit Changes* in the *Project* menu. Accordingly, changes made directly to the platform via its Web user interface can be synchronized by means of the *Reload Project* function. Visual editing essentially facilitates collaborative capturing and maintenance of traceability information and thus overall *project documentation* (cf. use case description in section 3).

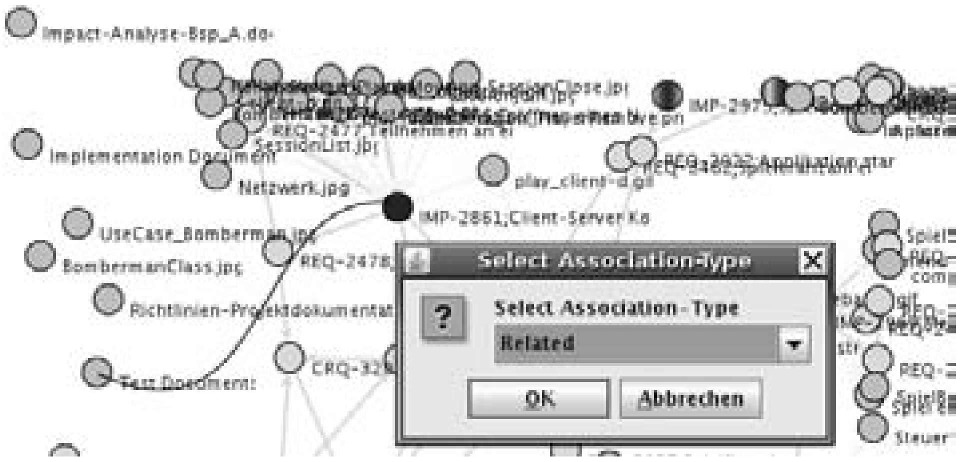

Figure 5: Visual Editing and Maintenance of Traceability Information

6 Summary and Outlook

As has been demonstrated in the preceding sections, the current prototype version implements all major functional requirements specified with respect to TraVis' *visual* rep-

[17] See [DMMP06] for further definitions of rationale management tasks in SE.

resentation, analysis, and maintenance support. It has also been shown that the adapted and customized version of CodeBeamer used for this prototypical implementation of the overall solution architecture covers the complementary *collaborative* capturing and maintenance processes. These TRM activities are particularly important in distributed software projects—mainly for enabling better mutual workplace awareness and informal coordination mechanisms due to the central availability of project knowledge in the form of traceability information.

Due to the fact that software projects can become very complex and spatial distribution of artifacts and actors hampers workplace awareness and process transparency, a lack of traceability can become harmful to project efficiency and documentation effectiveness in particular. Bidirectional end-to-end traceability, as required by CMMI, for instance, is critical, however, both with respect to in-project decision support and later maintenance tasks. Hence, TraVis provides added value particularly to distributed collaborative software development projects.

TRM methods as well as their combination with the VBSE approach can increase the quality of the artifacts developed within the individual project phases and, therefore, also the quality of the final product. For tool-based project support, the application of a software development platform that incorporates communication between all team members and enables information capturing should be considered. Additionally, tracking and managing particular requirements all the way to the new product are also supported [RGBH07].

Thus, within this paper, a novel trace visualization approach and respective tool support is introduced, which is based on such an underlying collaboration platform while enhancing the platform's functionality by graphical *visualization* as well as analytic filtering mechanisms and predefined views. In doing so, various TRM activities within distributed development projects are supported. Hereby, the focus of the tool's conception is mainly the support of cross-cutting project management functionality, requirements and change management in particular. This paper, thus, takes different requirements identified, substantiated, and verified in both, literature and real-world practice into account and implements a Web-enabled solution architecture based on an underlying central collaboration platform integrating various artifact repositories (cp. figure 2).

The latest version of TraVis described in this paper is complemented by enhanced *trace analysis* functionality and *predefined views* specifically designed to improve the tool's performance in certain TRM application respects [HGK07, Hil08]. Furthermore, for the *value-based* calculation of the size of the artifacts [HGK07], an advanced analysis method based on the *PageRank* algorithm is implemented [PBMW98]. Additionally, maintainability and usability of TraVis are further enhanced, for instance through additional filter and search functionality.[18] The main rationale for enhancing the solution with respect to *visual* and value-based analysis support is the need to reduce the complexity of relations in distributed reasonably and thus provide better decision support.

Within future conceptions and implementations of the tool support, functionality for addi-

[18]These enhancements are based on an initial set of two empirical evaluation studies concerning TraVis applicability in distributed settings [Hil08] and eventually aim at increasing the output quality and process efficiency of particular TRM tasks in distributed collaborative software development.

tional analyses like the graphical display of the *social networks* between project members and across multiple projects—both intra- and inter-organizationally—shall be integrated. For comprehensive social network analyses, *authorship* information pertaining to both pre- and post-specification artifacts is required [dSHR07]. Combining and visualizing information on artifact relation and contribution structures in turn allows for deriving social networks or sociograms[19] from socio-technical relations of artifacts and responsible stakeholders which provide a good basis for project and team structure analysis [dSRC$^+$04]. This, in turn, can further facilitate team awareness, communication, and informal coordination [RvdHAA$^+$07]. Personnel turnover situations that are most often noticeable in larger-scale projects, *offshore* outsourcing scenarios in particular, represent one major use case for this kind of information (see section 3 as well as [dSHR07]).

Besides these conceptional enhancements, already integrated functionality will be further evaluated experimentally to gather conclusions for future TRM requirements and development activities [Hil08]. In addition, several case studies and controlled experiments involving both students and partners within the software industry are planned in order to further evolve the solution and elicit remaining deficiencies as well as new requirements.

References

[BME$^+$07] Grady Booch, Robert A. Maksimchuk, Michael W. Engel, Bobbi J. Young, Jim Conallen, and Kelli A. Houston. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, Boston, USA, 3rd edition, 2007.

[DCAC03] Daniela Damian, James Chisan, Polly Allen, and Brian Corrie. Awareness Meets Requirements Management: Awareness Needs in Global Software Development. In *Proceedings of the Workshop on Global Software Development (GSD'03)*, pages 7–11, 2003.

[DMMP06] Allen Henry Dutoit, Raymond McCall, Ivan Mistrik, and Barbara Paech, editors. *Rationale Management in Software Engineering*. Springer, Berlin. Deutschland, 2006.

[DP98] Ralf Dömges and Klaus Pohl. Adapting Traceability Environments to Project-Specific Needs. *Communications of the ACM*, 41(12):54–62, 1998.

[DP01] Allen H. Dutoit and Barbara Paech. Rationale Management in Software Engineering. In Chang SK, editor, *Handbook on Software Engineering and Knowledge Engineering*, volume 1. World Scientific, 2001.

[dS05] Cleidson R. B. de Souza. *On the Relationship between Software Dependencies and Coordination: Field Studies and Tool Support*. PhD thesis, Donald Bren School of Information and Computer Science, University of California, Irvine, USA, 2005. http://www2.ufpa.br/cdesouza/pub/cdesouza-dissertation.pdf.

[dSDR$^+$04] Cleidson de Souza, Paul Dourish, David Redmiles, Stephen Quirk, and Erik Trainer. From Technical Dependencies to Social Dependencies. In *Proceedings of the Workshop on Social Networks at the 2004 International Conference on CSCW*, 2004.

[19]Sociograms are graph-based representation of social relations that a person has. In software projects, these can be derived from shared artifacts and communication structures [dS05].

[dSHR07] Cleidson R. B. de Souza, Tobias Hildenbrand, and David Redmiles. Towards Visualization and Analysis of Traceability Relationships in Distributed and Offshore Software Development Projects. In *Proceedings of the 1st International Conference on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD'07)*. Springer, 2007.

[dSRC$^+$04] Cleidson R. B. de Souza, David Redmiles, Li-Te Cheng, David Millen, and John Patterson. How a Good Software Practice Thwarts Collaboration: The Multiple Roles of APIs in Software Development. In *Proceedings of the 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT'04)*, pages 221–230. ACM Press, 2004.

[EBHG05] Alexander Egyed, Stefan Biffl, Matthias Heindl, and Paul Grünbacher. A value-based approach for understanding cost-benefit trade-offs during automated software traceability. In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 2–7. ACM Press, 2005.

[EH91] Michael Edwards and Steven L. Howell. *A Methodology for Requirements Specification and Traceability for Large Real-Time Complex Systems*. Defense Technical Information Center, Fort Belvoir, USA, 1991.

[GF94] Orlena C. Z. Gotel and Anthony C. W. Finkelstein. An Analysis of the Requirements Traceability Problem. In *Proceedings of the 1st International Conference on Requirements Engineering (RE'94)*, pages 94–101. IEEE Computer Society, 1994.

[GHR07] Michael Geisser, Tobias Hildenbrand, and Norman Riegel. Evaluating the Applicability of Requirements Engineering Tools for Distributed Software Development. *Working Paper des Lehrstuhls für ABWL und Wirtschaftsinformatik der Universität Mannheim*, (2), 2007.

[Got95] Orlena Gotel. *Contribution Structures for Requirements Traceability*. PhD thesis, Department of Computing, Imperial College of Science, Technology, and Medicine, London, UK, London, UK, 1995.

[HGK07] Tobias Hildenbrand, Michael Geisser, and Lars Klimpke. Konzeption und Implementierung eines Werkzeugs für nutzenbasiertes Traceability- und Rationale-Management in verteilten Entwicklungsumgebungen. *Working Paper, Lehrstuhl für ABWL und Wirtschaftsinformatik der Universität Mannheim*, (5), 2007.

[HGK08] Tobias Hildenbrand, Michael Geisser, and Lars Klimpke. TraVis 2 - Ein Werkzeug für verteiltes, nutzenbasiertes Traceability- und Rationale Management. *Working Paper, Lehrstuhl für ABWL und Wirtschaftsinformatik der Universität Mannheim*, 2008.

[Hil08] Tobias Hildenbrand. *Improving Traceability in Distributed Collaborative Software Development—A Design-Science Approach*. Phd thesis, University of Mannheim, Germany, 2008.

[HMFG00] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. Distance, Dependencies, and Delay in a Global Collaboration. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW'00)*, pages 319–328. ACM Press, 2000.

[HMM00] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, 2004.

[HRG$^+$08] Tobias Hildenbrand, Franz Rothlauf, Michael Geisser, Armin Heinzl, and Thomas Kude. Approaches to Collaborative Software Development. In *Proceedings of the 2nd Workshop on Engineering Complex Distributed Systems (ECDS'08)*. IEEE Computer Society, 2008. accepted for publication.

[HRH07] Tobias Hildenbrand, Franz Rothlauf, and Armin Heinzl. Ansätze zur kollaborativen Softwareerstellung. *WIRTSCHAFTSINFORMATIK*, 49(Special Issue):S72–S80, 2007.

[Kru04] Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley, Boston, USA, 3rd edition, 2004.

[KS98] Gerald Kotonya and Ian Sommerville. *Requirements Engineering – Processes and Techniques*. John Wiley & Sons, Chichester, UK, 1998.

[LS96] Mikael Lindvall and Kristian Sandahl. Practical Implications of Traceability. *Software Practice and Experience*, 26(10):1161–1180, 1996.

[LS98] Mikael Lindvall and Kristian Sandahl. Traceability Aspects of Impact Analysis in Object-Oriented Systems. *Journal of Software Maintenance: Research and Practice*, 10(1):37–57, 1998.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford Digital Library Technologies Project*, 1998.

[Poh07] Klaus Pohl. *Requirements Engineering – Grundlagen, Prinzipien, Techniken*. dpunkt.verlag, Heidelberg, Deutschland, 1st edition, 2007.

[RGBH07] Felix Rodriguez, Michael Geisser, Kay Berkling, and Tobias Hildenbrand. Evaluating Collaboration Platforms for Offshore Software Development Scenarios. In *Proceedings of the 1st International Conference on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD'07)*, pages 96–108. Springer, 2007.

[RJ01] Balasubramaniam Ramesh and Matthias Jarke. Towards Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 27(1):58–93, 2001.

[RvdHAA$^+$07] David Redmiles, André van der Hoek, Ban Al-Ani, Tobias Hildenbrand, Stephen Quirk, Anita Sarma, Roberto Silveira Silva Filho, Cleidson de Souza, and Erik Trainer. Continuous Coordination: A New Paradigm to Support Globally Distributed Software Development Projects. *WIRTSCHAFTSINFORMATIK*, 49(Sonderheft):S28–S38, 2007.

[Sch02] Bruno Schienmann. *Kontinuierliches Anforderungsmanagement: Prozesse, Techniken, Werkzeuge*. Addison-Wesley, Boston, USA, 2002.

[Som07] Ian Sommerville. *Software Engineering*. Addison-Wesley, Boston, USA, 8th edition, 2007.

[SZ04] George Spanoudakis and Andrea Zisman. Software Traceability: A Roadmap. In Shi-Kuo Chang, editor, *Handbook of Software Engineering and Knowledge Engineering*, pages 395–428. World Scientific Publishing, River Edge, USA, 2004.