ISR Institute for Software Research
University of California, Irvine

An End-To-End Software Traceability Tool in an Industrial Context

Hazeline Asuncion
University of California, Irvine
hasuncion@ics.uci.edu

Frédéric François
Wonderware Corp.
Frederic.Francois@wonderware.com

Richard N. Taylor
University of California, Irvine
taylor@ics.uci.edu

October 2006

ISR Technical Report # UCI-ISR-06-16

Institute for Software Research
ICS2 110
University of California, Irvine
Irvine, CA 92697-3455
www.isr.uci.edu

An End-To-End Software Traceability Tool in an Industrial Context

Hazeline Asuncion, Frédéric François, Richard N. Taylor
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425
hasuncion@ics.uci.edu

ISR Technical Report # UCI-ISR-06-16

October 2006

Abstract:

Traceability is a critically important aspect of software development that is often required by various professional standards and government agencies. Yet, current approaches do not adequately address end-toend traceability. Consequently, many industry projects become entangled in process overhead and fail to derive much benefit from current traceability solutions. This paper presents a successful end-to-end software traceability tool developed at Wonderware, a software development company and a business unit of Invensys Systems, Inc. Our process-oriented approach achieves comprehensive traceability and supports the entire software development life cycle by focusing on both requirements traceability and process traceability. This paper offers general traceability guidelines that have emerged from the experience of implementing and deploying this traceability tool within actual company constraints. We discuss encouraging preliminary results and point to the advantages gained in using our approach.

An End-To-End Software Traceability Tool in an Industrial Context

Hazeline Asuncion
Institute of Software Research
UC Irvine
hasuncion@ics.uci.edu

Frédéric François
Wonderware Corporation
Frederic.Francois
@wonderware.com

Richard N. Taylor
Institute of Software Research
UC Irvine
taylor@ics.uci.edu

ISR Technical Report # UCI-ISR-06-16
October 2006

Abstract

Traceability is a critically important aspect of software development that is often required by various professional standards and government agencies. Yet, current approaches do not adequately address end-to-end traceability. Consequently, many industry projects become entangled in process overhead and fail to derive much benefit from current traceability solutions. This paper presents a successful end-to-end software traceability tool developed at Wonderware, a software development company and a business unit of Invensys Systems, Inc. Our process-oriented approach achieves comprehensive traceability and supports the entire software development life cycle by focusing on both requirements traceability and process traceability. This paper offers general traceability guidelines that have emerged from the experience of implementing and deploying this traceability tool within actual company constraints. We discuss encouraging preliminary results and point to the advantages gained in using our approach.

1. Software Traceability

While traceability is recognized as a "critical success factor" in software development [6], the lack of effective software traceability continues to be a perennial problem in industry projects [10]. The sheer number of artifacts produced in a project, the differing levels of formality and specificity between various artifact types, and the complex interrelationships between artifacts [23][24][2] combine to form the heart of the traceability problem. Finding a comprehensive traceability solution yields many benefits. Traceability aids in system comprehension, impact analysis, system debugging, and communication between the development team and stakeholders [21][6][13][17].

While traceability is encouraged or even mandated by various standards and government agencies [22] [14][15], high costs [13][19] make it infeasible for many organizations to incorporate traceability [2]. Even in companies where a specialized traceability tool is adopted, the traceability problem still exists [10], due to the tool's rigidity, narrow focus, and lack of interoperability with other tools. Various techniques have been proposed to reduce overhead and enhance traceability [1][2][7][5][23]. Yet, these approaches fall short of providing a comprehensive approach to traceability that supports the entire software development life cycle. Current techniques usually only attempt to link artifacts within one phase or between two adjacent phases in the life cycle. For instance, although tracing the life of a requirement is essential to the success of a software product, most approaches emphasize performing requirements traceability within one phase of software development: the requirements phase.

We therefore define the concept of end-to-end traceability as an overarching traceability that extends throughout the entire life of a project, from the requirements phase to the test phase. End-to-end traceability weaves artifacts together in a sequential fashion in tandem with the various phases of the life cycle. For instance, end-to-end requirements traceability is satisfied when different phase-specific manifestations of the same requirement are linked together across the life cycle.

We also emphasize process traceability as an important facet of an effective traceability approach. Relationships between artifacts can be intertwined with the underlying software processes. This view can be represented as a graph with nodes representing artifacts and links between the nodes representing the

process. Raising the visibility of actual software processes enables users to accurately compare actual practices to stated company procedures. Not only does process traceability improve the actual software process, but it also captures the rationale behind a specific artifact and fosters system comprehension.

In this paper, we attempt to combine end-to-end requirements traceability and process traceability. We present our insights in designing a software traceability tool at Wonderware, a mid-sized software development company known for producing industrial automation software. We adopt a process-oriented approach to achieve comprehensive traceability that supports the whole development life cycle.

Note that we limit our scope to post-RS traceability [10]. In addition, the paper does not concentrate on configuration management, which is concerned with tracing linkages between different versions of the same artifact. Configuration management is orthogonal and complementary to both end-to-end requirements and process traceability. Finally, our approach mainly applies to tracing text-based artifacts.

The next section provides a brief discussion of the traceability problem at Wonderware. Section 3 presents an overview of the traceability tool we have implemented, the preliminary results after deploying the tool, and an overview of traceability guidelines. We present our guidelines in depth in Sections 4 – 11. In each of these sections, we first present the guideline along with the rationale for including it. We then present our method of implementing the guideline. Each section wraps up by discussing the guideline in the context of related research. Section 12 concludes the paper with a discussion and future work.

2. Traceability Problem at Wonderware

We present a brief background of Wonderware in order to introduce the company's traceability problem. A business unit of Invensys, Wonderware is a mid-sized software sales and development company with distributed development centers across the globe. Wonderware is a leading supplier of industrial automation and information software, with software deployed to approximately a hundred thousand plants worldwide [25]. The company is based in Lake Forest with development centers in the United States, Australia, EMEA, and India. There are about 250 development employees. Projects run in parallel, with 40 projects currently in development.

As a company that deploys its software products to a hundred thousand plants worldwide, the problem of traceability takes center stage. Many of Wonderware's customers are food and pharmaceutical manufacturing companies that have internal regulations or external obligations to adopt products that follow government standards. Consequently, software products that run the plants of Wonderware's customers must pass FDA approval. Not only does Wonderware need to demonstrate traceability to comply with various standards, but a lack of traceability equates to inability to win new customers and new business partners, leading to missed revenues. In addition, existing customers require traceability audits on Wonderware's software development process. Wonderware fits the classification of a high-end traceability user [18] since traceability is viewed as a benefit to the company. Wonderware is actually ahead of most organizations in addressing requirements traceability. They recognize the need to identify problem artifacts and improve their process.

Even though Wonderware has an advanced notion of traceability, it is still difficult for the company to effectively trace requirements and processes. The problems are many, complex, and subtle. We introduce both the requirements traceability problem and the process traceability problem in the upcoming sections.

2.1. Requirements Traceability Problem

The commercial traceability tool that the company originally used is expensive in terms of both licensing costs and labor hours expended to maintain traceability. Yet, despite the high costs, the requirements traceability problem still existed at Wonderware. The most glaring manifestation of the problem is in the inconsistencies between different representations of the same artifact. One particular example of this is in document obsolescence. An artifact X is stored in a database (accessed via the commercial traceability tool) and in multiple documents outside the tool. Each representation achieves a specific purpose. The artifact in the documents enables shared understanding among a project team, while the artifact in the database enables collective organizational knowledge and reporting across multiple projects. However, when artifact X is modified in a document, the same artifact stored in the traceability tool database and in other documents becomes obsolete. This problem is intensified by the thousands of documents and the numerous individual artifacts within each document that Wonderware manages.

Another major problem is the lack of demonstrable end-to-end traceability. Different groups within

Wonderware own different artifact types that are keys in establishing the traceability chain. Not only are the different artifact types owned by different groups, but they are also stored in different tools lacking interoperability between them. These conditions prevent end-to-end traceability from occurring.

Other problems include the inability to manipulate artifacts in bulk, the limited functionality of the commercial traceability tool, the inability to scale, and the lack of effective artifact visualization.

2.2. Process Traceability Problem

At Wonderware, many individuals adopt ad-hoc workarounds to accomplish their development process tasks. These workarounds are not captured in any organizational document. Since actual processes oftentimes only reside in the minds of individuals, the capability of groups to share knowledge with each other is limited and is highly subject to staff turnover.

In addition, obtaining an accurate project status is a time-consuming process. In an environment where projects run in parallel, the immediate retrieval of accurate project status is crucial. Project Managers, Development Managers, and Architects all oversee multiple projects at once. It is sometimes difficult to know the current status of a project since every related individual must be manually solicited for information. Thus, process traceability is needed to alleviate these issues.

3. Implementing the Software Traceability Tool

We designed the software traceability tool to address the existing problems stated in the previous section. The tool was intended to achieve three main goals: 1) enable the maintenance of traceability links between key artifacts; 2) preserve the integrity of documents; 3) support software development life cycle activities. While the tool is not expected to automatically generate traces between artifacts, the tool is intended to ease the establishment and maintenance of accurate traces by providing access to all of the necessary information. To this end, the tool must support the reuse of existing artifacts by providing a search by keyword functionality, by allowing access to all key artifacts that reside in various groups, by identifying traced and untraced artifacts, by assigning a unique ID to new artifacts, and by updating the artifact status (i.e. active, new, retired, etc).

To preserve the integrity of the documents, the tool must provide automated support for cascading changes from one representation of an artifact to another. In addition, a document must be stored in a location that is accessible to all members of the development team in order to eliminate manual document reconciliation.

The tool should provide support for software development life cycle activities by providing a user-specific task list (each item in the list is a link to another screen which aids users to perform the task), document reviews, and document approvals. The task list is simply a guidance mechanism, with no enforcement controls. There is no strict ordering to the tasks.

We used a three-tiered client-server architecture in designing our tool. Figure 1 shows the main components. We used MS SQL databases as our artifact repository. We used MS SharePoint to provide workflow support. Clients access the SharePoint server via client browsers to enter data, perform specific tasks or produce reports. MS InfoPath is also used in generating reports. MS Word contains macros to support document automation. The embedded macros are able to directly access and manipulate the database in order to maintain traceability among artifacts.

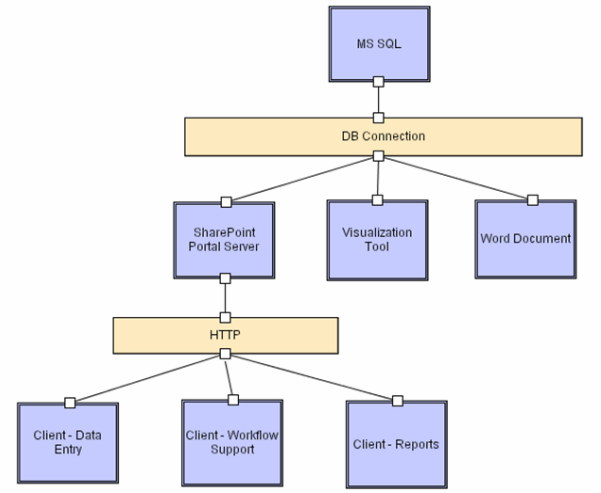

Figure 1: Traceability Tool Design

Here is a usage scenario. In order to support establishing of traces, users may search for existing artifacts by keywords and view reports of traced/untraced artifacts. Process support is provided via user-specific task lists. Each item on the task list links to another window which provides the functionality for the user to accomplish the task. Document integrity is guaranteed via bidirectional

updates. For example, when users create a document, the artifacts in the document are automatically saved to the database. Changes to the artifact may be made via data entry forms provided in the workflow. The next time users open the document, they can automatically retrieve the latest artifact update from the database.

3.1. Preliminary Results

To evaluate the traceability tool, we tested the artifact repository by populating the repository with live data from new Wonderware projects. We also allowed the entire organization to utilize the SharePoint traceability site by giving them read-access. The main test users of this tool are the Architect Group. This group oversees the technical aspects of every software project at Wonderware. We tested the following functionalities: 1) mapping between trace artifacts (Projects-Features, Use Cases, and Functional Requirements); 2) maintaining document integrity; and 3) supporting the software development lifecycle, by managing the artifact list, document creation, and document reviews. Tracing between marketing requirements and Use Cases has not been tested, although the group was able to successfully import a list of requirements into the artifact repository. In addition, the Test Group was able to develop test cases and trace them to the Functional Requirements. The traceability tool has been running successfully for the last six months.

According to feedback from the architects, the traceability tool is a welcome change to the Architect Group since performing their traceability tasks are now much easier than before. These preliminary results suggest that architects are now more efficient since their tasks are now integrated into the workflow and since process overhead has been minimized. Although none of the architects received any training in using the site, they still successfully performed their tasks. Furthermore, document obsolescence and multiple data entry have been largely eliminated.

Due to the success of the traceability tool, the company plans to deploy the tool to the rest of the organization. Wonderware is willing to fully adopt the tool since the tool provides the substantial benefits of traceability at low maintenance costs. In addition, the cost of deployment is low. The traceability tool is a web-based application where users can simply download the latest version of a deployed code or macro.

3.2. Guidelines Overview

The guidelines presented in this paper are not meant to replace existing traceability techniques, but rather to complement them. The purpose is to provide insights in approaching a traceability problem in an industrial setting.

Our aim is to provide an end-to-end requirements traceability approach. Although we do not necessarily cover each development phase, we select key global trace artifacts that help us validate that the stated requirements have been fulfilled globally.

While parts of the traceability tool are still in development, we have identified key lessons which have contributed to the success of our approach within the Wonderware organization. We present the lessons we learned as guidelines. These lessons reinforce some current approaches in traceability while challenging others. In addition, we have formulated new ideas to meet company-specific requirements. The guidelines are ordered from organization-level guidelines to tool-specific guidelines.

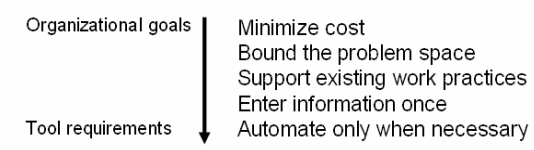

Figure 2: Guidelines ordering

4. Guideline: Minimize Cost

4.1. Rationale

Given that we were developing a traceability tool in a real-world setting with concrete goals and hard deadlines, the main justification for undertaking the traceability project at Wonderware is to minimize cost. There are two main costs: 1) the number of labor hours to train users to adopt a traceability strategy and the number of labor hours to establish and maintain traceability; and 2) the cost of purchasing or developing a traceability tool. We designed our approach with cost-minimization as our primary goal. In retrospect, this guideline proved to be a key idea in designing a practical and feasible approach.

Since Wonderware opted to develop a home-grown traceability tool, it is important to minimize the cost of tool evolution as well. We adopted an approach where the tool can be easily modified to reflect changes in actual processes (See Section 6).

4.2. Implementing Low-Cost Traceability tool

Implementing a low-cost traceability tool is a high level organizational goal that spawned some of the other guidelines that we discuss below. For example, minimizing the number of labor hours in performing traceability tasks can be achieved by supporting existing work practices. We also adopted a "just enough traceability" strategy. How is this strategy operationalized?

First, there must be a benefit derived from each traceability link established. For example, tracing a requirement to a Use Case identifies to the customer that a specific requirement has been implemented. Adding another link from the Use Case to a passed Test Case proves to the customer that the requirement has been tested. In this situation, establishing links to implemented code is unnecessary since there is no added value.

Second, the "just enough traceability" point is achieved when the trace information enables users to accomplish specific tasks. For example, if architects can easily identify which requirements have not been mapped to Use Cases, this provides them a list of remaining requirements they must analyze. In addition, if project managers can easily obtain an accurate status report of a specific project, then an adequate process traceability support has been provided.

To lower the cost of maintaining a traceability tool, Wonderware is phasing out the commercial traceability tool they were using. The company also leveraged existing company-owned tools as a platform for developing the traceability tool: MS SQL, MS SharePoint, etc. Using MS SharePoint was beneficial since client access is web-based, lowering the cost of deployment. MS SharePoint functionality can be extended using WebParts, independent components embedded on a SharePoint webpage.

4.3. Related Research

Minimizing cost implies a cost/benefit analysis in adopting a traceability approach. This is consistent with prioritizing artifacts according to stakeholder values [3] and assigning values to trace links [7]. Minimizing cost also reinforces the "trace for a purpose" strategy in [5].

5. Guideline: Bound the Problem Space

5.1. Rationale

Whereas minimizing cost tackles the traceability problem from an economic standpoint, bounding the problem space tackles the problem from a technical viewpoint. We adopted this guideline from [6]. Since the overall goal is to achieve end-to-end requirements traceability, constraining the types of trace artifacts becomes even more important. Not only is too much traceability unnecessary, but it may cause more harm than good. Excessive traceability increases the chance of inaccurate traces, and a few inaccurate traces can throw into question the validity of all the other traces [8].

The types of requirements artifacts to trace are artifacts specified in the company standard operating procedures to achieve requirements traceability. Not only do these artifacts represent various phases in development spanning the entire lifecycle, they also act as interfaces between group boundaries. Cooperation between various groups is enhanced by conforming to a bounded set of published artifacts. For example, the Architect Group publishes a list of Use Cases (UCs). The Development Team takes this published list of artifacts and produces another set of artifacts, namely code. The Architect Group has complete control over tracing artifacts at a finer level of granularity, as long as they trace to the list of UCs. In the same token, the Development Team may also establish finer granularity traces, but they must trace back to the UCs. Thus, each group is free to implement their own localized trace artifacts, given that they trace to the company stated global trace artifacts. A discussion of how this supports work practices follows in the next section.

In retrospect, the distinction between global and local trace artifacts is an important insight. Identifying a few types of global trace artifacts enables end-to-end traceability at the organization level while localized traces distributes traceability tasks among different groups. Essentially, this distinction allows us to bound the problem space at various levels of granularity.

The types of process artifacts to trace are Product, Project and Features. A Product is a collection of Projects while a Feature is a collection of Use Cases. We limit our scope to tracing artifacts at the Project level, while supporting Project traces to the Product level.

5.2. Identifying Global Trace Artifacts

Setting bounds on the problem space includes the identification of global trace artifacts. We identified

two sets of global trace artifacts: requirements trace artifacts and process trace artifacts. Requirements trace artifacts include Marketing Requirements (MRQ), Use Cases (UCs), Functional Requirements (FRs), and Test Cases (TCs). The relationships between these artifacts are shown in Figure 3. These artifacts are also related to process trace artifacts which include Product, Project, and Feature information (see Figure 5). End-to-end requirements traceability is achieved since artifacts are related from requirements all the way to test cases.

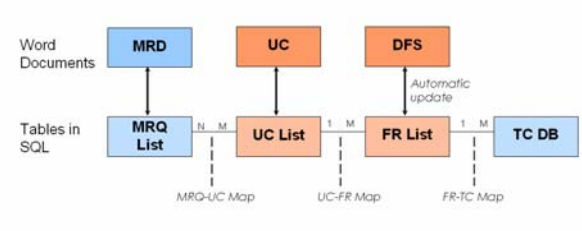

Figure 3: Requirements Trace Artifacts

5.3. Related Research

Bounding the problem space builds on the approach in [6] where a project manager defines trace data types. We added the distinction between global trace artifacts (organization level) and localized trace artifacts (group level).

6. Guideline: Support Existing Work Practices

6.1. Rationale

Since the company is concerned with process traceability, capturing existing practices and formalizing it into a workflow was emphasized by key users from the start of the project. As we previously mentioned, an effective end-to-end traceability strategy requires cooperation between various groups within Wonderware. This is achievable if the approach supports existing work practices. Any traceability tasks that users may have to do must be integrated with their existing tasks to ensure that the task is accomplished. In addition, showing users they directly benefit from performing the traceability task minimizes the distaste for establishing and maintaining traceability [13] and increases the likelihood that the trace information they provide is accurate.

Providing process support also enables work processes to be standardized across organization. This eliminates the need for ad-hoc workarounds and supports organizational knowledge. Standardizing the process especially helps raise the visibility of actual process to remote users or groups in different geographical locations. Thus, a group in Australia can participate in the development processes in Lake Forest.

Another advantage to supporting the current work process is the increased productivity of users. Providing automated support to manual tasks increases the efficiency of users. In addition, streamlining the flow of work as a task list lessens the cognitive load of users.

Furthermore, supporting existing work practices enables process traceability. This is essential to collecting meaningful data for project management.

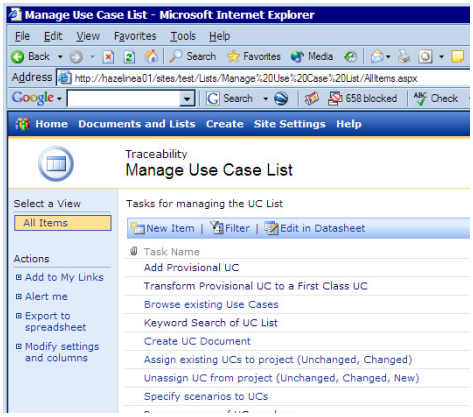

Figure 4: Task List

6.2. Supporting Software Development Life Cycle with a Workflow

Requirements traceability is closely tied to process traceability in Wonderware. The production and consumption of trace artifacts are linked to various key users who follow a specified process. Understanding the process provides insights on how to incorporate traceability tasks into existing tasks such that it does not impose heavy burdens on users, minimizing cost in terms of required man hours. Thus, once the global trace artifacts were determined, we identified key users as well as high level tasks that the workflow will support. We selected to implement the workflow in MS SharePoint since it has built-in process support (e.g. document reviews) and it integrates well with other tools used (e.g. MS Office).

We designed the workflow to cater to various key users. Producers of trace information are marketing group, architects, project managers, and test managers. Consumers of trace information are the executive

group, developers, test engineers, and external auditors.

Once the high level tasks were identified, they were further subdivided into independent lower level tasks. We emphasize "independent" since this minimizes the cost of evolving the workflow to support process changes. Independent subtasks were implemented as separate functions that can be added, modified, or removed independently of each other. We wrote these functions as ASP WebPart components embedded into a MS SharePoint web page. There is no ordering imposed on the tasks. Users are presented with a task list (see Figure 4) which links to another screen that provides the functionality to help them accomplish that specific task.

Figure 5 shows how the different artifacts are traced and how the workflow supports the creation and maintenance of these artifacts.

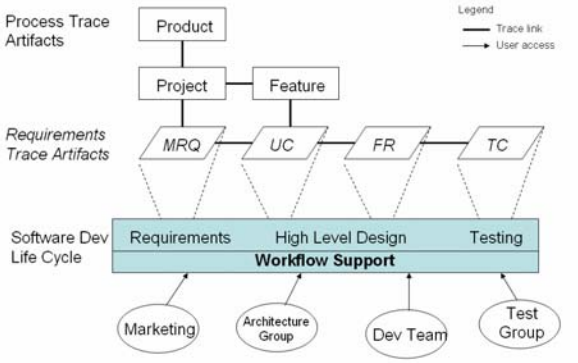

Figure 5: Software Traceability Model

6.3. Related Research

Few approaches take a process-oriented approach to traceability. Process traceability is coupled with requirements traceability in [16]. However, the focus is on the requirements phase. [6] is also a related approach although it focuses more on an organization learning from experience to identify trace artifacts, rather than supporting actual work practices to achieve an end-to-end traceability throughout the software lifecycle. Knowledge-Based Software Assistant (KBSA) provides process support for the software development life cycle [9]. KBSA describes an integrated tool support that includes a Project Management (PM), Work Breakdown Structure (WBS), and a hyper-media tool (IBIS). While it provides traces between a specified set of information (issues, arguments, and positions), it does not provide requirements traceability.

7. Guideline: Enter Information Once

7.1. Rationale

Due to the difficulties encountered in reconciling multiple representations of the same artifact, we adopted this guideline in designing the traceability tool. Entering artifact information in multiple tools not only requires extra labor hours to enter the same information multiple times, but it also adds an additional overhead of ensuring that these artifacts are consistent with each other. Due to heterogeneous tools used in supporting the development process, it becomes impossible to enter the information once since the tools do not interoperate.

Entering information once does not only apply to having an artifact stored in different tools. It also applies to information entered in one phase being carried over to another phase. For instance, in the planning phase, provisional Use Cases are created without a unique identifier. During the design phase the list of provisional Use Cases are revisited, with some being accepted and others rejected. The accepted Use Cases are then assigned a unique identifier. This ensures that artifacts created earlier in the phase do not "slip through the cracks" until the testing phase uncovers the problem.

7.2. Centralizing Artifact Storage

Instead of storing artifacts in multiple tools, we centralized artifact storage in several MS SQL databases: MRQs are stored in a database owned by the Marketing Group, UCs and FRs are stored in a database owned by the Architect Group, and TCs are stored by the Test Group. In addition, process artifacts (Product, Project, Features) are stored in a database owned by the Project Managers. Since the traceability software tool is owned by the Architecture Group, the tool has read only access to the other groups' databases.

Tracing between distributed artifacts is possible as long as each group abides by the unique identification of the published artifacts. That is, a unique ID assigned to published artifact will never change. The relationship between artifacts is specified in Figure 3.

7.3. Related Research

Having redundant data is one of the main problems of traceability [24], causing additional overhead of reconciling data [23]. Another method of enabling entering information once is through the information

integration approach [23]. This supports traceability between heterogeneous artifacts by translating them to a homogeneous representation. Once inside the homogeneous environment, traces can be automatically generated using a set of rules. In the case of Wonderware, this environment can be used to maintain traces, but not in establishing traces. Mapping the key artifacts is based on architects' requirements analysis and is difficult to automate. The next guideline discusses this issue further.

8. Guideline: Automate Only When Necessary

8.1. Rationale

Due to time constraints and the necessity to have the framework of the software traceability tool up and running quickly, we identified which tasks must be automated right away, which tasks can be automated but can wait, and which tasks do not need to be automated at all. In retrospect, this analysis proved useful not only in the short-term, but also in the long-term as far as understanding the limitations of automating traceability tasks.

High priority tasks for automation included maintaining consistency between different representations of the same artifact, bulk manipulation of artifacts, and automatic generation of reports. In order to migrate data to the traceability tool, it was also necessary to automate the extraction of artifacts from legacy documents.

Although automation can greatly alleviate the cost of traceability, there are cases when manually establishing traces is not only acceptable, but necessary. In the case of Wonderware, Architects are in charge of translating requirements (MRQs) to implementable modules (UCs), and later ensuring that the implementation meets the requirements. Thus, the manual mapping between MRQs and UCs is part of the Architects' task of requirements analysis and does not need to be automated.

It is important to know when automated support is appropriate. Automation is necessary for tasks that are tedious and error prone if manually done. This is the case in manually maintaining consistency between various forms of an artifact. For instance, a marketing requirement is represented in two forms: a list form stored as a table and a verbose form stored as a Word document. Maintaining consistency between the same artifacts in different representation qualifies for automation.

Document automation, which involves the use of embedded macros, is also necessary to ensure consistency between an artifact stored in various documents and avoiding document obsolescence. This automation is necessary at Wonderware since there are thousands of documents to manage. Document automation also aids in extracting artifacts from legacy documents that do not conform to the current templates.

8.2. Maintaining Consistency Using Document Automation

Since artifacts may have multiple representations, and manually reconciling between representations is a tedious task, we provide automated support in our tool. One of the ways we achieve this is via document automation with Word macros. Using macros embedded in Word document templates, we can extract artifacts using a set of criteria (i.e. using keywords). The extracted artifacts are checked by a user to ensure that they are valid trace artifacts. Once they are checked, they are saved to the database.

We also use Word macros to implement bidirectional updates between the artifact stored in the database and the artifact represented as a document. Thus, whenever a document is opened, it checks the database for the latest version of the artifact. Once the user closes the Word document, the artifacts are automatically updated in the database. (Note: artifacts may also be manipulated outside the Word document through various forms related to the workflow support. Since these changes are automatically reflected back to the database, it is important to have automated support to update the corresponding artifacts represented as a Word document.)

In addition, since Wonderware has hundreds of legacy documents not associated with a template, we also used macros to extract trace artifacts in order to migrate the documents to the current template. It was doable to extract trace artifacts from Word documents since they were tagged. For example, each Functional Requirement in a Detailed Functional Specification document is labeled as "FRxxx" where "x" is a unique number assigned.

8.3. Related Research

Automating traceability tasks has its limitations [11]. For instance, automatically generating trace links is only as accurate as the user input [7]. Most of traceability approaches deal with establishing traceability links after the fact, and not during the

generation of artifacts to support the software development life cycle [11]. The different types of automated support available include 1) automated generation of traceability links such as in [23] [21] [1]; and 2) traceability link support for automated queries [5] which is concerned with traversing the related traces. Although item (1) is useful in the context of discovering traces for software maintenance, it does not apply to Wonderware where traces are established when artifacts are created. According to the classification in [5], our traceability tool provides semi-automated queries in that a set of traced artifacts are returned. Event-based traceability (EBT) provides automatic notification to traced artifacts that a related artifact has changed, although consistency is not enforced [4]. [20] is a commercial traceability tool that provides limited support for maintaining consistency between an artifact stored as a Word document and an artifact stored within a tool. In contrast to our traceability tool, change updates only flow in one direction, from the Word document to the commercial tool.

9. Discussion and Future Work

Designing a software traceability tool in the context of an actual software development setting presented several subtle and complex challenges. The tool must clearly demonstrate end-to-end requirements traceability, have a high return on investment, and gain organization-wide acceptance. The wide range of potential users, from upper management to test engineers to external auditors, prompted us to take a comprehensive view of traceability. The insights presented in this paper are both adaptations of current approaches in literature and novel ideas that resulted from discussions with key users. A process-oriented approach to requirements traceability not only supports users in accomplishing their tasks, but it also encourages users to adopt the traceability tool. The identification of a few types of global artifacts mitigates the complexity associated with traceability. Differentiating between global and local trace artifacts means that various groups maintain full ownership of their localized trace artifacts while the organization achieves a high level end-to-end requirements traceability. Automation is limited to replacing burdensome tasks associated with traceability, such as maintaining consistency between various representations of an artifact.

Since the approach described has only been evaluated in one software development setting, it is worthwhile to test whether these ideas hold in other contexts. Other aspects of traceability such as tracing non-functional requirements and tracing between different levels of abstraction of an artifact (general to detailed) are open issues. In addition, tracing artifacts through the maintenance phase has not been considered in our approach.

10. Acknowledgements

We would like to thank Jim McIntyre, the Systems Architect at Wonderware, for driving the traceability project and for all the insightful discussions.

11. References

[1] Antoniol, G., Caprile, B., Potrich, A., & Tonella, P., "Designcode traceability recovery: selecting the basic linkage properties", *Sci. of Comp. Programmin*g, 2001, 40, pp. 213–234,

[2] Alexander, I., "Towards Automatic Traceability in Industrial Practice", *Proc. of the First Intl. Workshop on Traceability*, 2002, pp 26-31.

[3] Boehm B., & Huang, L.G., "Value-based software engineering: a case study", *Computer*, 36, 3, pp. 33-41.

[4] Cleland-Huang J., Chang C.K., & Christensen M, "Event-based traceability for managing evolutionary change", *IEEE TSE, 29*, 9, Sept. 2003, pp.796-810.

[5] Cleland-Huang J., Zemont G., & Lukasik W., "A heterogeneous solution for improving the return on investment of requirements traceability", *Proc.. 12th IEEE Intl. Conf. Requirements Engineering*, 2004, pp. 230-239.

[6] Domges R., & Pohl K., "Adapting traceability environments to project specific needs", *CACM, 41*, 12, Dec. 1998, pp.54-62.

[7] Egyed, A., Biffl, S., Heindl, M., & Grunbacher, P., "A Value-Based Approach for Understanding Cost-Benefit Trade-Offs During Automated Software Traceability", *Proc. of the 3rd Intl. workshop on Traceability in emerging forms of software engineering,* Long Beach, CA, 2005, pp.2-7.

[8] Egyed, A., "A Scenario-Driven Approach to Traceability", *Proc. of ICSE 2001 (ICSE 23),* pp.123-32.

[9] Gerken, M.J., Roberts, N.A., & White D.A., "The knowledge-based software assistant: a formal, object oriented software development environment", *Proc. of NAECON 1996*. 2, New York, NY 1996, pp.511-18.

[10] Gotel, O.C.Z., & Finkelstein, C.W., "An analysis of the requirements traceability problem", *Proc. of the First Intl. Conference on Requirements Engineering*, 1994, pp.94-101.

[11] Hayes, J. H., & Dekhtyar, A., "Humans in the Traceability Loop: Can't Live with 'Em, Can't Live Without 'Em", *Proc. of the 3rd Intl. workshop on Traceability in emerging forms of software engineering*, Long Beach, CA, 2005, pp.20-23.

[12] *IEEE Standard Computer Dictionary*, IEEE, New York, NY, Jan 1991.

[13] Jarke, M., "Requirements Tracing", *CACM, 41*, 12, Dec. 1998, pp. 32-36.

[14] Leffingwell, D., & Widrig, D., "The Role of Requirements Traceability in System Development", http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/sep02/TraceabilitySep02.pdf, Viewed Sep 5, 2006.
[15] Wallace, D., Ippolito, L., "A Framework for the Development and Assurance of High Integrity Software", *NIST Special Publication 500-223,* Dec 1994, Taken from http://hissa.nist.gov/publications/sp223/ on Sep 5, 2006.
[16] Pohl, K., *Process-Centered Requirements Engineering*, Advanced Software Development Series, J. Wiley & Sons Ltd., Taunton, England, 1996.
[17] Pohl, K., Brandenburg, M., & Gulich, A., "Integrating Requirement and Architecture Information: A Scenario and Meta-Model Based Approach", *7th Intl. Workshop on Reqts. Engr*, 2001
[18] Ramesh, B. "Factors Influencing Requirements Traceability Practice",*CACM*, *41*, 12, Dec. 1998, p. 37.
[19] Ramesh B., Powers T., Stubbs C., & Edwards M. "Implementing requirements traceability: a case study", *Proc. 2nd IEEE Intl. Symp. on Reqts. Engr*,1995, pp. 89-95.
[20] IBM Rational Requisite Pro, http://www-306.ibm.com/software/awdtools/reqpro/, Viewed Sep5, 2006.
[21] Richardson, J., & Green, J, "Automating traceability for generated software artifacts", *Proc. of the 19th Automated Soft. Engr. Conf. (ASE)*, Linz, Austria, 2004, pp.24-33.
[22] Taken from Rolland, C., & Prakash, N., "From conceptual modeling to requirements engineering", *Annals of Software Engineering*, 10(1-4), 2000, pp. 151-176.
[23] Anderson, K. M., Sherba, S. A., & Lepthien, W. V., "Towards Large-Scale Information Integration", *Proc. of the 24th Intl. Con. on Software Engineering*, May 2002.
[24] Singleton, M. *Automating Code and Documentation Management.* Prentice-Hall, Inc. New Jersey. 1987.
[25] Strothman, J., "Wonderware Pioneer Pitsker Wins ISA Life Achievement Award", *Intech*, August 2006, p. 64.

12. Keywords

Requirements Traceability, Process Traceability, End-To-End Traceability.