An End-To-End Industrial Software Traceability Tool

Hazeline U. Asuncion
Institute for Software Research
University of California, Irvine
Irvine, California 92697
+1-949-824-2703
hasuncion@ics.uci.edu

Frédéric François
Wonderware Corporation
26561 Rancho Parkway South
Lake Forest, California 92630
+1-949-303-8388
frederic@uxdesigner.com

Richard N. Taylor
Institute for Software Research
University of California, Irvine
Irvine, California 92697
+1-949-824-6429
taylor@ics.uci.edu

ABSTRACT
Traceability is an important aspect of software development that is often required by various professional standards and government agencies. Yet current industrial approaches do not typically address end-to-end traceability. Moreover, 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. This process-oriented approach achieves comprehensive traceability and supports the entire software development life cycle by focusing on both requirements traceability and process traceability. We offer new perspectives in analyzing the problem as well as general traceability guidelines. These guidelines have emerged from the experience of implementing and deploying the traceability tool within actual company constraints. We discuss encouraging results and point to the advantages gained in using our approach.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management - *Life cycle;* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *Documentation.*

General Terms
Documentation, Management.

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

1. SOFTWARE TRACEABILITY
While traceability is recognized as a "critical success factor" in software development [13], the lack of effective software traceability continues to be a perennial problem in industry projects [17, 28]. 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 [4, 5, 27] form the heart of the traceability problem.

ESEC-FSE'07, September 3–7, 2007, Cavtat near Dubrovnik, Croatia.

Compounding these technical problems are social problems which include internal politics, lack of communication between groups, and unclear understanding of artifact ownership [17, 19]. 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 [13, 19, 23, 26].

While traceability is encouraged or even mandated by various standards and government agencies [20, 24, 30], high costs [19, 24] make it infeasible for many organizations to incorporate traceability into their practices [4]. Even in companies where a specialized traceability tool is adopted, the traceability problem often still exists [17], due to tool rigidity, narrow focus, and lack of interoperability with other tools. Various techniques have been proposed to reduce overhead and enhance traceability [4-6, 11, 15]. Yet these approaches fall short of providing a comprehensive approach to traceability that supports the entire software development life cycle (SDLC).

We therefore define the concept of end-to-end traceability as an overarching traceability that extends throughout the entire life of a development project, from the requirements phase to the test phase. End-to-end traceability weaves artifacts together in tandem with the various phases of the life cycle. For instance, end-to-end requirements traceability is achieved 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. Raising the visibility of actual software processes enables users to compare actual practices to stated company procedures. It thus offers the potential for improving the actual software process, as well as capturing the rationale behind a specific artifact, fostering 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-requirements specification traceability [17], the tracing of requirements to downstream artifacts. 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 an analysis of the traceability problem at Wonderware. Section 3 discusses related work. Section 4 presents an overview of the traceability tool, the traceability model, the design of the tool, the implementation status, and the preliminary results after initial deployment. Section 5 covers an overview of the traceability guidelines. We present our guidelines in-depth in Sections 6 – 10. In each of these sections, we present the rationale for including the guideline, our implementation, and related research. Section 11 concludes the paper with a discussion and future work.

2. TRACEABILITY PROBLEM AT WONDERWARE

We present a brief background of Wonderware [1] 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 [29]. The company is based in Lake Forest, California 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 their customers must pass FDA approval. Not only does the company 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. The company fits the classification of a high-end traceability user [25] since traceability is viewed as a benefit to the company. They are 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. We introduce both the requirements traceability problem and the process traceability problem in the upcoming subsections. We then characterize the problem into three key perspectives which are addressed by the guidelines.

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 is 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. (We define a project team to consist of all individuals responsible for the delivery of software. It includes an Architect, a Project Manager, a Development Manager, Developers, a Test Manager, and Testers.). 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 crucial 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 hinder end-to-end traceability.

Other problems include the inaccessibility of artifacts, the inability to manipulate artifacts in bulk, the limited functionality of the commercial traceability tool, the inability to scale, and the lack of effective trace 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 reside only 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 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.

2.3 Problem Analysis

In order to achieve end-to-end software traceability, we believe the traceability problem must be tackled from three key perspectives: economic, technical, and social (see Figure 1). We hypothesize that these are inherently interwoven and must be addressed simultaneously in order to have an effective traceability solution in organizations similar to Wonderware.

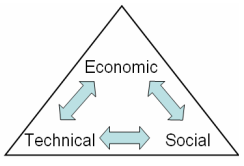

Figure 1: Key Traceability Perspectives

The economic perspective is mainly concerned with the cost of adopting a traceability strategy and the expected benefits. The cost can either be in terms of the cost of purchasing or developing a traceability tool or the labor hours in training users and performing traceability tasks. The economic perspective also considers whether adopting a strategy increases the profitability of the software product. Since traceability is known to incur high overhead costs [19, 24], the economic perspective is the main determinant for adopting or rejecting a strategy, especially in an industrial software development setting such as Wonderware. Thus, the economic perspective takes a more prominent role among the three. If the benefits outweigh the costs, then an organization is likely to adopt a traceability solution.

However, simply considering the economic perspective is not sufficient. Indeed, the technical perspective also plays a key role. This perspective is concerned with defining the types of artifacts to trace and their trace relationships. It also addresses the heterogeneity of the artifacts as well as the heterogeneity of the tools used in creating and viewing the artifacts.

The social perspective is the final test for whether a traceability approach is achievable. The social perspective is concerned with understanding the interaction of various groups in the organization and their expectations of the traceability tool. An organization-wide adoption of the tool critically depends on the social perspective.

3. RELATED WORK

Although the use of traceability in practice has been studied in different contexts, from small software companies to government funded projects [4, 8, 21, 24], none focuses on achieving end-to-end software traceability across various groups in an organization. The case study on a US DoD project [24] conducted more than a decade ago is the closest match to our work since it takes a comprehensive view of traceability and targets a wide-range of users: Upper Management, Project Managers, Engineers, Testers, and Auditors. Since the study was conducted while a CASE traceability tool was being introduced, it does not report on whether the expected benefits were actually realized.

Other studies [8, 21] conducted in smaller software development contexts (e.g. four to five engineers [8]), report successes in implementing traceability across selected artifacts. The main users (i.e. producers of trace information) are developers. Limited success in implementing traceability techniques is reported in [4].

Interestingly, the study that closely matches our work [24] documents difficulties that reflect the key traceability perspectives we identified: high costs of adopting a traceability solution (related to the economic perspective), lack of interoperability between the traceability tool and existing tools (technical perspective), and different user expectations of the traceability tool (social perspective).

4. THE TRACEABILITY TOOL

We designed the software traceability tool to address the problems stated in Section 2. The tool was intended to achieve three main goals: 1) minimize overhead in trace definition and maintenance; 2) preserve document integrity; 3) support SDLC activities.

The tool must minimize overhead in performing traceability tasks. To this end, even though the tool takes a manual approach in trace definition (i.e. users select artifacts to map), overhead is minimized by providing users all the necessary information to accurately establish trace links. For instance, the ability to search by keyword through all existing artifacts across projects and the ability to access artifacts that reside in various groups enables users to find possible artifacts to trace. Not only does this support artifact reuse, it also significantly limits the number of artifacts to examine. Overhead is further minimized by intertwining trace maintenance with trace utilization. Users are not required to perform the tedious task of checking trace links to prevent link deterioration. Instead, trace links are updated as a side effect to trace utilization. For example, the accuracy of the Use Case to Functional Requirement trace link is checked by the Test Group while they create new Test Cases. Intuitive data entry forms enable users to correct traces and update artifacts status (i.e. active, new, retired, etc). Thus, trace links are kept accurate and updated as they are used.

To preserve the integrity of the documents, the tool must provide automated support for cascading changes from one representation of an artifact to another. Document integrity is guaranteed via bidirectional updates between documents and the artifact repository. 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. In addition, documents must also be accessible to all members of the project team to eliminate manual document reconciliation.

The tool should support SDLC activities by providing a prescriptive workflow catering to groups of users. The workflow is presented as a hierarchical task list with hyperlinks to components that automate (e.g. bidirectional document update) or semi-automate (e.g. artifact query) tedious tasks. There are no enforcement controls and no strict ordering to the tasks.

4.1 End-To-End Software Traceability Model

Figure 2 shows our end-to-end software traceability model. The boxes at the top represent the global trace artifacts and solid lines

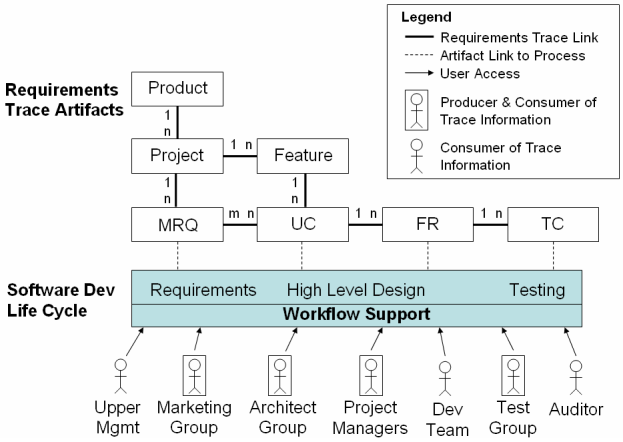

Figure 2: End-To-End Software Traceability Model

represent their requirements trace links. Most of the direct artifact links are one to many (e.g. Product to Projects mapping). The exception is the many to many mapping between the Marketing Requirements and Use Cases. We identified the following global artifacts types to trace: Marketing Requirements (MRQs), Use Cases (UCs), Functional Requirements (FRs), and Test Cases (TCs). MRQs are organized by Projects and UCs are organized by Features. Several Projects roll up to one Product. Each of the artifact types (MRQ, UC, FR, TC) links to a phase in the software development life cycle (indicated by dotted lines). (Note that a UC is subdivided into several Scenarios and each Scenario links to several TCs. For brevity, Scenarios are not included in the figure.)

The users of the system, shown at the bottom of the diagram, are distinguished by whether they produce trace information or not. Producers, i.e. groups responsible for entering trace information, are the Marketing Group, Architect Group, Project Managers, and Test Group. All users are consumers of trace information.

The software development life cycle is supported by the workflow. As users perform their tasks, they access the globally traced artifacts via the workflow. This allows for close proximity between tasks and related artifacts. Users may also directly access the artifacts outside the workflow.

4.2 Traceability Tool Design

We used a three-tiered client-server architecture in designing our tool. Figure 3 shows the main components. The traceability tool has a web front-end to increase the accessibility of artifacts among all members of the project team, including teams working remotely. Here, users can enter data, perform specific tasks or generate reports. We chose MS SQL database as our artifact repository and MS SharePoint to provide workflow support and to store all project files. These components are selected because the company already owns these tools and they are interoperable with existing tools (e.g. MS Office) that the company uses. MS Word macros are used to support document automation. The embedded macros directly access and manipulate the database in order to maintain consistency among artifacts.

We designed our trace links as follows. The trace relationships between artifacts are stored in the repository, separate from documents. These trace links only represent satisfaction relationships [28] and do not hold additional semantic information. Users map the artifacts through a form that contains

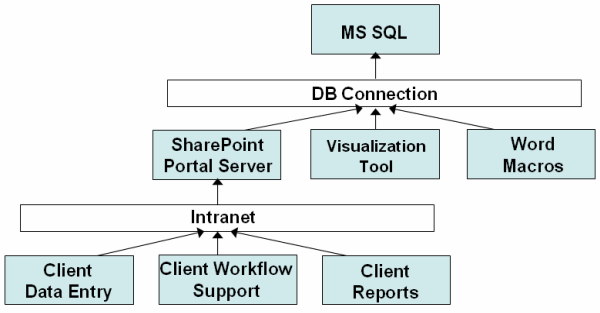

Figure 3: Traceability Tool Design

a Map button, an Unmap button, a keyword search function, and lists of artifacts by project. Auto-generated trace reports enable users to identify traced and untraced artifacts. All artifacts are related to a project, and all corresponding documents are stored in a project site in MS SharePoint.

We addressed the process traceability problem in the following ways. First, we studied current work practices and codified them in a workflow, minimizing the necessity for ad-hoc workarounds. Secondly, we facilitated obtaining accurate project status. The existence of key documents in a project site and the mapping between artifacts implicitly indicates the status of a project. For instance, the presence of an approved Marketing Requirements Document but the lack of UC mapping indicates that the next task is requirements analysis by the Architect. In addition, the workflow supports document reviews so that a Project Manager knows the status of a document (e.g. reviewed, approved) at any given time without manually soliciting information from various members of the team. We also designed the main screen of the traceability tool to contain a list of all active projects with their status.

Some of the usage scenarios we envision include the following. Architects establish traces between UCs and FRs as part of their high level design task. Project Managers track project status by viewing the status of artifacts in a project site. Developers implementing UCs refer to the traced MRQs or FRs to gain a better understanding of how their components fit into the overall system. Upper management views the project status of all current projects by checking the current phase of the development life cycle. Auditors may check whether a Marketing Requirement has been adequately tested by navigating through the trace links from MRQs to UCs to FRs to TCs.

4.3 Implementation Status

The basic components of the tool have been implemented. Each artifact type is stored in an MS SQL table, with their trace relationships stored as a database schema. SharePoint has been configured to store all project related files (e.g. documents, spreadsheets). Word Macros have been embedded on the templates of high level design documents to provide bidirectional updates.

The paper design of the workflow has been completed. Workflow support that helps Project Managers to reuse artifacts from old projects has been deployed for trial usage. This early workflow implementation has both the task and the functionality integrated into one screen. The supporting forms for data entry and task automation are currently being implemented. Some reports have also been implemented, but have not been deployed. The Visualization component has not been implemented. Additional functionalities are incrementally released to the users.

All current project artifacts have been successfully migrated from the commercial tool to the new traceability tool. At this point, the commercial tool only holds artifacts of legacy projects.

4.4 Preliminary Results

To evaluate the traceability tool, we tested the artifact repository by populating the repository with live data from all current Wonderware projects. We also allowed the entire organization to

utilize the traceability tool by giving them read-access. The main test users of this configuration have been the Architect Group. This group oversees the technical aspects of every software project at Wonderware. They analyze requirements from the Marketing Group, perform high level design, and ensure that the delivered software satisfies the requirements. 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 life cycle, 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 and Scenarios within Use Cases. The traceability system has been running successfully for the last year.

According to feedback from the architects, our new traceability tool is a welcome addition to the Architect Group since performing their traceability tasks are now much easier than before. In comparison to the commercial traceability tool, the amount of time spent in performing traceability tasks has been reduced by a factor of two. Currently, the tool aids Architects in their high level design tasks. Creating and managing Use Case and Detailed Functional Specification documents are now less time-consuming since all the artifacts are easily accessible through the workflow. The tool facilitates reuse of existing artifacts, the extraction of new artifacts from documents, and the establishment and maintenance of trace links.

These preliminary results suggest that architects are now more efficient since their traceability tasks are now integrated with their SDLC tasks and since process overhead has been minimized. Although none of the architects received any training in using the initial release of the traceability tool, they still successfully performed their tasks. Furthermore, document obsolescence, manual document reconciliation, and multiple data entry have been largely eliminated. All members of a project team are now able to access all key artifacts. Finally, the interoperability problem no longer exists.

Due to the success of the traceability tool, the company is currently deploying the tool to the rest of the organization. Training is underway for Architects, Project Managers, and Developers to teach users the newly released functionality. Wonderware is willing to fully adopt the tool since it provides substantial benefits of traceability at low maintenance costs. For instance, the decrease in the required support staff allowed tool administrators, who used to solely manage the commercial traceability tool, to assist teams with project specific tasks. Consequently, this enables the company to improve their product quality by allocating more resources to the delivery of their products. In addition, the cost of deployment is low. The latest functionality of the traceability system is immediately accessible to all members of the organization, including remote users, through the web user-interface.

5. TRACEABILITY GUIDELINES

The success observed at Wonderware to date supports our hypothesis that the three key traceability perspectives must be tackled simultaneously. The guidelines we used in designing our traceability tool directly address each of these perspectives. The first guideline, 'minimizing cost', addresses the economic perspective. 'Bound the problem space', 'enter information once', and 'automate only when necessary' are guidelines that tackle the technical perspective. Finally, 'support existing work practices' addresses the social perspective.

Even though our approach has only been evaluated in one software development setting, the general applicability of these guidelines is evidenced by similar findings of other researchers in different settings. For instance, the guideline 'minimize cost' was the topic of a panel discussion titled "Just Enough Requirements Traceability" in a recent conference [12]. The diverse panel, comprising researchers and practitioners from corporate and agile software development settings, are all concerned with minimizing costs when adopting a traceability strategy [2]. Similarly, one of the major concerns in adopting a comprehensive traceability strategy in a government funded project was the high costs that significantly exceeded their existing documentation costs [24].

The guideline 'bound the problem space' addresses the definition and selection of trace artifacts and has been adapted from [13]. This has also been applied in [24].

A case study in a very small software development setting yielded parallel lessons to the guidelines 'support existing work practices' and 'automate only when necessary' [21]. In that setting, the traceability tool is mainly used to support the developers in the implementation phase. 'Support existing work practices' has also been found applicable in [8, 24].

The guideline 'enter information once' also has general applicability. According to a survey of nine software projects, one of the problems with traceability is the dual data entry users must perform [7].

6. GUIDELINE: MINIMIZE COST

6.1 Rationale

Given that we were developing a traceability tool in a real-world setting with concrete goals and hard deadlines, we were forced to deal with cost as a key concern. There are three main costs: 1) the number of labor hours to train users in applying a traceability strategy; 2) the number of labor hours to establish and maintain traceability; and 3) the cost of purchasing or developing a traceability tool. We designed our approach with cost-minimization as a primary goal. In retrospect, this guideline proved to be a key force 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.

6.2 Implementation

Implementing a low-cost traceability tool is a high level organizational goal that spawned some of the other guidelines we discuss below. For instance, minimizing the number of labor hours in performing traceability tasks can be achieved by supporting existing work practices. We minimize labor hours in training users by designing the front-end with usability in mind. This is accomplished by talking to key users and understanding

their expectations of the system. We also adopted a "just enough traceability" strategy to minimize labor hours in performing traceability tasks.

How is this strategy operationalized? First, there must be a benefit derived from each trace link established. For example, tracing an MRQ to a UC identifies to the customer that a specific requirement has been implemented. Adding another link from the UC to a passed TC proves to the customer that the requirement has been tested. In this case, establishing links to implemented code is unnecessary since there is no added value to the customer.

Second, the "just enough traceability" point is achieved when the trace information enables users to accomplish specific tasks. For example, it is necessary to establish a trace link from MRQs to UCs. It aids architects in their requirements analysis task since it is easy to identify the remaining of MRQs to analyze (i.e. unmapped MRQs). However, it is not necessary to establish trace links from MRQs to Scenarios within UCs since there are no tasks aided by this mapping. In addition, if Project Managers can easily obtain an accurate status report of a specific project, then adequate process traceability support has been provided.

Minimizing cost was also targeted in the development and maintenance of the traceability tool. Due to the high cost of maintaining the commercial traceability tool, Wonderware is phasing out its usage. To minimize development costs, company-owned tools were selected as a platform for developing the traceability tool. To minimize the cost of deployment, users can access and download the latest version of a document template, Word macro, or component (in the form of WebParts) from the web front-end. Section 8.2 provides details on how we achieved minimizing the cost of evolving the traceability tool.

6.3 Related Research

Minimizing cost implies a cost/benefit analysis in adopting a traceability approach. This is consistent with prioritizing artifacts according to stakeholder values [9] and assigning values to trace links [15]. Minimizing cost also reinforces the "trace for a purpose" strategy in [11]. A COMPSAC 2006 panel discussed low-cost traceability approaches such as automated traceability using information retrieval techniques and lean traceability using developers' knowledge of the system [12].

7. GUIDELINE: BOUND THE PROBLEM SPACE

7.1 Rationale

Whereas minimizing cost addresses the economic aspect of the traceability problem, bounding the problem space addresses the technical aspect. We adopted this guideline from [13]. 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, 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 others [14].

We constrained the types of artifacts to trace and we limited the traceability relationship to the category of satisfaction [28] (i.e. trace links that show that the Marketing Requirements have been satisfied). We did not include evolution relationships since we were only concerned with the most current trace links between the current versions of the artifacts.

In general, it may be difficult to select which artifacts to trace, especially in settings where an organization has not explicitly identified key artifacts. [13] states that this determination is based on a project manager's experience and can vary from project to project. When selecting artifacts to trace, one should consider external organizational goals as well as internal benefits derived from tracing each artifact.

7.2 Implementation

Setting bounds on the problem space includes the identification of global trace artifacts. Since the tool takes an organizational view of traceability, it handles tracing between the global trace artifacts. Tracing at the group level is left to the individual groups. The types of requirements artifacts to trace are artifacts specified in the company standard operating procedures to achieve requirements traceability.

The global trace artifacts we selected enable us to demonstrate end-to-end requirements traceability: MRQs, UCs, FRs, and TCs. The relationships between these artifacts are shown in Figure 2. We limit our scope to tracing artifacts at the Project level, while supporting Project traces to the Product level. End-to-end requirements traceability is achievable since artifacts are related from Marketing Requirements all the way to Test Cases.

Not only do these artifacts represent various phases in development spanning the entire life cycle, but 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 UCs. The Development Team takes this published list of artifacts and produces another set of artifacts, namely code. 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. Limiting the types of global trace artifacts enables efficient end-to-end traceability at the organizational level while localized traces distributes traceability tasks among different groups. Essentially, this distinction bounds the problem space at different levels of granularity.

7.3 Related Research

Bounding the problem space builds on the approach in [13] where a project manager defines trace data types. We added the distinction between global trace artifacts (organization level) and localized trace artifacts (group level).

8. GUIDELINE: SUPPORT EXISTING WORK PRACTICES

8.1 Rationale

Since the company is concerned with process traceability, capturing existing practices and formalizing them 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 that they directly benefit from performing the traceability task minimizes their disinclination towards establishing and maintaining traceability [7, 19] and increases the likelihood that the trace information they provide is accurate.

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

Another advantage to supporting the current work process is the increased productivity of users. First, the close proximity between the task at hand and the relevant artifacts eliminates the time users spend in searching for artifacts. Second, providing automated support to manual tasks increases the efficiency of users. Third, streamlining the flow of work as a task list lessens the cognitive load of users. In the case of the Architect Group, the prescriptive workflow is particularly helpful in recalling traceability tasks since these are performed only at specific phases in the software development life cycle.

Furthermore, supporting existing work practices enables process traceability. One can identify areas were work practices diverge from stated company procedures. Once detected, either the work practices are adjusted or company procedures are modified to close the gaps. Thus, Project Managers are able to coordinate tasks between the team and collect meaningful data for project management.

8.2 Implementation

Requirements traceability is closely tied to process traceability at 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 so that they do not impose heavy burdens on users, which minimizes cost in terms of required labor hours. Once the global trace artifacts were determined, we identified key users as well as high level tasks that the workflow will support. We chose to implement the workflow in MS SharePoint since it has built-in process support (e.g. document reviews).

We designed the workflow to cater to various key users (e.g. producers of trace information). We support these users by enabling them to continue using their tools and to retain full ownership of their artifacts (i.e. other groups only have read access to their artifacts). We also ensured that the artifacts are accessible to all consumers of trace information.

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 web page. We did not impose a strict ordering on the tasks.

The traceability tool provides custom task lists to the producers of trace information. Thus, a task list for a Project Manager is different from a task list for an Architect. A high level task hyperlinks to lower level tasks which in turn hyperlink to the functions that help users accomplish their tasks.

8.3 Related Research

Few approaches take a process-oriented approach to traceability. Process traceability is coupled with requirements traceability in [22], but the focus is only the requirements phase. [13] 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 end-to-end traceability throughout the software life cycle. The Knowledge-Based Software Assistant (KBSA) provides process support for the SDLC [16]. KBSA has integrated tool support that includes Project Management (PM), a Work Breakdown Structure (WBS), and a hypermedia tool (IBIS). While KBSA provides traces between a specified set of information (issues, arguments, and positions), it does not provide requirements traceability.

9. GUIDELINE: ENTER INFORMATION ONCE

9.1 Rationale

Due to prior 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 the overhead of ensuring consistency between artifacts. The heterogeneous tools used in supporting the development process, with no data exchange capability between tools make it impossible to enter information once. This lack of interoperability between tools have been identified as a problem by [5, 13, 24]

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

In general, enabling tool interoperability such that users enter information once is difficult. One approach is to use custom code to enable data exchange. This is suitable in settings where there is a limited number of tools to support. Another approach is to translate data into a common format that can be consumed by the heterogeneous tools. However, this approach runs the risk of losing information in the translation. At Wonderware, we opted to use both custom code and shared repositories to address this guideline.

9.2 Implementation

We implemented this guideline by taking a "distributed centralization" approach to artifact storage. Instead of storing the same artifact types in multiple tools, we centralized the storage of an artifact type to one MS SQL database. However, in order for groups to retain ownership of their artifacts type, it was necessary to use multiple distributed databases across the organization. Thus, 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. Since the traceability software tool is owned by the Architect Group, the tool has read 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. The unique ID assigned to a published artifact will never change.

We also ensure that the tools are interoperable and that artifacts can be seamlessly accessed using different tools. Data exchange between tools is done via the shared artifact storage (i.e. MS SQL databases). Custom code (e.g. macros, scripts) is used to make the tools aware of the database schema. Consequently, different tools are able to accurately render and edit the globally traced artifacts. Thus, a new artifact can be entered via a Word document, modified through a web data entry form, and viewed in a table-format report. We also designed the traceability tool with extensibility in mind. Any new tool may interoperate with the traceability system as long as it can establish database connection with MS SQL and it can be customized.

9.3 Related Research

Having redundant data is one of the main problems of traceability [27], causing additional overhead of reconciling data [5]. Another method of enabling entering information once is through the information integration approach [5], which 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, the approach in [5] can be used for maintaining traces but not for establishing traces because mapping the key artifacts is based on users' semantic knowledge of the artifacts and is difficult to automate. The next guideline discusses this issue further.

10. GUIDELINE: AUTOMATE ONLY WHEN NECESSARY

10.1 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 maintenance of 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 that do not conform to the current templates.

Automation is necessary for tasks that are tedious and error prone if manually done. This is the case with manually maintaining consistency between various representations of an artifact (e.g. between a list form and a verbose form). Document automation, which involves the use of embedded macros, is also necessary to avoid document obsolescence.

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, traces are established when artifacts are created, and not after the fact. For example, the manual mapping between MRQs and UCs is part of the Architects' task of requirements analysis and does not need to be automated. We mitigate the time spent in manual mapping by providing automated support to the surrounding tasks.

Manual mapping runs the risk of establishing inaccurate trace links, but so does automated trace link generation (i.e. problems with precision and recall [18]). Since our approach is integrated with the development tasks and is performed by the creators of artifacts, trace link creation is accurate and adequately efficient. Scalability may be an issue, although this can be addressed by constraining the number of artifacts to trace.

10.2 Implementation

We provide the following automated support to alleviate the burden of performing traceability tasks: document automation, automated assignment of unique IDs to new artifacts, automated report generation, and artifact search across projects. Document automation enables artifact extraction from new documents, bidirectional updates between documents and the artifact repository, and the migration of legacy documents that do not conform to current document templates.

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. 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 contained in a document (see Figure 4). The macros in the documents ensure consistency with the artifacts stored in the repository. Thus, the user activated macros automatically retrieve and update the artifacts in the database. 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 in the documents. Figure 5 shows that the Marketing Requirements Document (MRD), Use Case Document (UCD), and Detailed Functional Specification (DFS), are automatically updated with the artifacts in the repository, namely the Marketing Requirements (MRQ) List, Use Cases (UC) List, and Functional

Requirements (FR) List. At Wonderware, Test Cases (TC) do not have a corresponding 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. Extracting trace artifacts from Word documents was feasible 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.

The artifact search function enables users to search for artifacts across projects and across groups by keywords, IDs, the creator of the artifact, or any other information related to the artifact. Artifact IDs are automatically assigned when an artifact is added to the repository. Furthermore, custom trace reports enable users to view traced or untraced artifacts.

10.3 Related Research

Automating traceability tasks has its limitations [18]. For instance, automatically generating trace links is only as accurate as the user input [15]. Most of traceability approaches deal with establishing traceability links after the fact, and not during the generation of artifacts to support the SDLC [18]. The different types of automated support available include 1) automated generation of traceability links such as in [5, 6, 26]; and 2) traceability link support for automated queries [11] 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 [11], our traceability tool provides semi-automated queries in that a set of traced artifacts are returned in the search. Event-based traceability (EBT) provides automatic notification to traced artifacts that a related artifact has changed, although consistency is not enforced [10]. [3] 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.

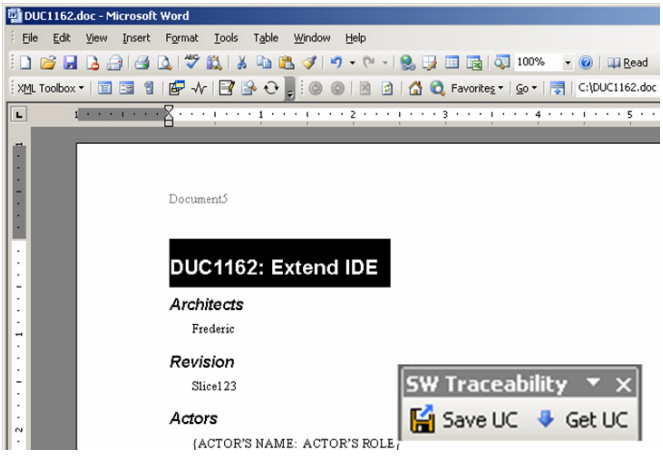

Figure 4: Document Automation

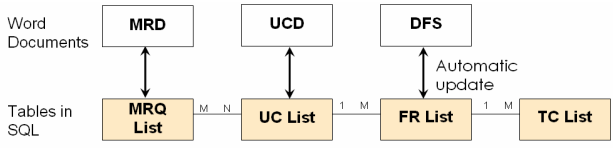

Figure 5: Maintain Consistency via Bidirectional Updates

11. 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 wide range of potential users, from Upper Management to Architects to External Auditors, prompted us to take a comprehensive view of traceability. The tool must have a high return on investment, clearly demonstrate end-to-end requirements traceability, and gain organization-wide acceptance. These requirements reflect the three key traceability perspectives we identified. The guidelines we used in designing our tool tackle these perspectives simultaneously. Minimizing cost ensures that the traceability approach is feasible (economic perspective). A process-oriented approach to requirements traceability not only supports users in accomplishing their tasks but also encourages users to adopt the traceability tool (social perspective). The identification of a few types of global artifacts mitigates the complexity associated with traceability (technical perspective). 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 (social and technical perspectives). Automation replaces burdensome tasks associated with traceability, such as maintaining consistency between various artifact representations (technical perspective).

Our experience both confirms and challenges conventional notions about traceability. We confirm that high cost imposed by a traceability approach is a key inhibitor to traceability [19]. We also confirm that the selection of key trace artifacts aids in a successful traceability solution [13]. Our experience challenges the notion that manual trace links are to be avoided [15, 21]. Even though users manually establish trace links, we minimize costs by automating or semi-automating the surrounding tasks related to traceability.

Although our approach has only been evaluated in one software development setting, the generality of the guidelines is suggested by similar findings 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.

12. ACKNOWLEDGMENTS

We would like to thank Jim McIntyre, the Systems Architect at Wonderware, for driving the traceability project and for all the insightful discussions. We thank Arthur Asuncion for high level discussions. We also thank the anonymous reviewers for their feedback. Effort partially funded by the National Science Foundation under grant number IIS 0205724.

13. REFERENCES

[1] *Wonderware*. <http://www.wonderware.com/>.

[2] *Final Program - COMPSAC 2006*. <http://conferences.computer.org/compsac/2006/pdf/final-prog-compsac2006-9-15.pdf>.

[3] *IBM Rational Requisite Pro*. <http://www-306.ibm.com/software/awdtools/reqpro/>.

[4] Alexander, I. Towards Automatic Traceability in Industrial Practice. In *Proc. of the 1st Int'l Workshop on Traceability.* p. 26-31, 2002.

[5] Anderson, K.M., Sherba, S.A., et al. Towards Large-Scale Information Integration. In *Proc. of the 24th ICSE.* Orlando, Florida, May, 2002.

[6] Antoniol, G., Caprile, B., et al. Design-code Traceability Recovery: Selecting the Basic Linkage Properties. *Elsevier. Science of Comp Prog.* 40(2-3), p. 213-34, Jul, 2001.

[7] Arkley, P. and Riddle, S. Overcoming the Traceability Benefit Problem. In *Proc. of the 13th IEEE Int'l Conf on Reqts Engr.* Paris, France, Aug 29 - Sep 2, 2005.

[8] Arkley, P. and Riddle, S. Tailoring Traceability Information to Business Needs. In *Proc. of the 14th IEEE Int'l Conf on Reqts Engr.* Minneapolis, St. Paul, MN, Sep 11-15, 2006.

[9] Boehm, B. and Huang, L.G. Value-based Software Engineering: A Case Study. *Computer.* 36(3),p. 33-41,2003.

[10] Cleland-Huang, J., Chang, C.K., et al. Event-based Traceability for Managing Evolutionary Change. *IEEE TSE.* 29(9), p. 796-810, Sep, 2003.

[11] Cleland-Huang, J., Zemont, G., et al. A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability. In *Proc. of the 12th IEEE Int'l Reqts Engr Conf.* p230-239, Kyoto, Japan, Sep 6-11, 2004.

[12] Cleland-Huang, J. Just Enough Requirements Traceability. In *Proc. of the 30th Annual Int'l Comp Soft and Applications Conf.* Chicago, IL, Sep 17-21, 2006.

[13] Domges, R. and Pohl, K. Adapting Traceability Environments to Project Specific Needs. *CACM.* 41(12), p. 54-62, 1998.

[14] Egyed, A. A Scenario-driven Approach to Traceability. In *Proc. of the 23^{e} ICSE* p. 123-132, Toronto, Ontario, Canada, May 12-19, 2001.

[15] Egyed, A., Biffl, S., et al. A Value-based Approach for Understanding Cost-benefit Trade-offs During Automated Software Traceability. In *3rd Int'l Workshop on Traceability in Emerging Forms of Soft Engr*. p. 2-7, ACM Press: Long Beach, CA, 2005.

[16] Gerken, M., J., Roberts, N., A., et al. The Knowledge-based Software Assistant: A Formal, Object Oriented Software Development Environment. In *Proc. of the NAECON.* 2, p. 511-18, Dayton, OH, May 20-23, 1996.

[17] Gotel, O. and Finkelstein, C. An Analysis of the Requirements Traceability Problem. In *Proc. of the 1st IEEE Int'l Conf on Reqts Engr.* p. 94-101, Los Alamitos, CA, 1994.

[18] Hayes, J.H. and Dekhtyar, A. Humans in the Traceability Loop: Can't Live with 'Em, Can't Live Without 'Em. In *Proc. of the 3rd Int'l Workshop on Traceability in Emerging Forms of Soft Engr.* p.20-23, Long Beach, CA, Nov 8, 2005.

[19] Jarke, M. Requirements Tracing. *CACM.* 41(12), p. 32-36, Dec, 1998.

[20] Leffingwell, D. and Widrig, D. *The Role of Requirements Traceability in System Development*. <http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/sep02/TraceabilitySep02.pdf>, 2002.

[21] Neumuller, C. and Grunbacher, P. Automating Software Traceability in Very Small Companies - a Case Study and Lessons Learned. In *Proc. of the 21st IEEE Int'l Conf on Automated Soft Engr.* Tokyo, Japan, Sep 18-22, 2006.

[22] Pohl, K. *Process-Centered Requirements Engineering*. Advanced Software Development Series. 342 pgs., John Wiley & Sons, Inc.: Taunton, England, 1996.

[23] Pohl, K., Brandenburg, M., et al. Integrating Requirement and Architecture Information: A Scenario and Meta-Model Based Approach. In *7th Intl. Workshop on Reqts Engr: Foundation for Soft Quality*, 2001.

[24] Ramesh, B., Powers, T., et al. Implementing Requirements Traceability: A Case Study. In *Proc. of the 1995 Intl. Symp on Reqts Engr.* p. 89-95, York, UK, Mar 27-29 1995.

[25] Ramesh, B. Factors Influencing Requirements Traceability Practice. *CACM.* 41(12), p. 37-44, 1998.

[26] Richardson, J. and Green, J. Automating Traceability for Generated Software Artifacts. In *Proc. of the 19th Int'l. Conf on ASE.* p. 24-33, Linz, Austria, Sep 20-24, 2004.

[27] Singleton, M.E. *Automating Code and Documentation Management*. Prentice-Hall, Inc.: New Jersey, 1987.

[28] Spanoudakis, G. and Zisman, A. *Software Traceability: A Roadmap* Advances in Soft Engr and Knowledge Engr. Chang, S.K. ed. 3, World Scientific Publishing, 2005.

[29] Strothman, J. Wonderware Pioneer Pitsker Wins ISA Life Achievement Award. *Intech.* p. 64, Aug, 2006.

[30] Wallace, D. and Ippolito, L. A Framework for the Development and Assurance of High Integrity Software. US Dept. of Commerce, NIST, 1994.