MANAGING TRACEABILITY IN DATA WAREHOUSE DEVELOPMENT PROJECTS

Dr. Vinay Kumar  
Professor VIPS affiliated to Guru Gobind Singh Indraprastha University, Delhi. Author, Ex-Scientist ‘E’ NIC  
vinay5861@gmail.com

Reema Thareja  
Assistant Professor SPM Delhi University, Author of Oxford University Press, India  
reema_thareja@yahoo.com  
*Corresponding Author

Abstract

Traceability is the ability to verify the history, location, or application of an entity by means of a documented record. It is considered to be one of the vital quality attribute of data warehouses and is of great interest to developers. Although traceability is a key success factor of data warehouse development projects but lack of a proper methodology has impeded the application of traceability techniques in practice.

From a long time, traceability had been ignored as an area of research. This is because many developers used to think that it is not an important quality factor and practicing traceability would simply involve cost, time, money and other resources. However, a data warehouse is of little use to its users if the entire history or details of a particular entity is not recorded properly.

The paper therefore presents an overview of data warehouse followed by key terms and definitions of important aspects of traceability. The paper also brings forth the scheme of implementing traceability in data warehouse projects and the workflow of activities that must be performed while applying traceability practices. Last but not the least, the paper discusses certain benefits, limiting factors of implementing traceability as well as the approaches which when adopted can help the data warehouse developers to overcome the hurdles in practicing traceability. As a final note the paper puts light on future challenges where much more research needs to be done.

Keywords Traceability · Data warehouse · Software engineering, requirements engineering

1. Introduction to Data Warehouse

A data warehouse is an informational delivery system that is meant to provide information to its users so that they can make use of this information to make their businesses better. For this purpose, the data warehouse integrates and transforms the organization’s data into information to make it suitable for making strategic decisions. For building a data warehouse we need to extract the current and historical data from the organization’s operational databases and from other external sources; transform this data to resolve any conflicts and finally store it in the data warehouse.

W. H. Inmon, who is considered to be the father of data warehousing has defined the data warehouse as “a subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management's decisions.”

2. Defining Traceability

Traceability which is being used as a data warehouse quality attribute has been defined as the ability to describe and follow the life of software artifacts [1]. While supporting traceability enhances understanding of the project, neglecting this attribute, on the other hand, leads to less maintainable and a defective product due to inconsistencies or omissions.
The traceability parameter of data warehouse quality is constantly gaining importance and has become a hot subject for research. However, in today’s scenario, traceability practices are far from mature, and we are still standing at the beginning of an emerging discipline. Therefore, a lot more research still has to be done. Traceability is a challenging issue especially because researchers in this field usually belong to very different research communities like requirements engineering, modeling, etc and there is only little communication between these communities.

Initially, traceability was taken up as a subject under requirements engineering but it is in fact, a method to manage traces of artifacts other than requirements. When we talk of requirements engineering or software engineering, traces are used for validation and verification of relating artifacts and for software maintenance as it improves project understanding.

**Traces in the system**

When we develop a data warehouse from scratch, we actually end up in creating multiple unique intermediate and final products. This is because data warehousing is an emerging and not a very mature technology. So, the users are not able to specify their exact requirements from the system in the very early stages. In such a case the only input that the data warehouse development team gets is the high-level goal, or product vision. This vision is slowly transformed into requirements.

The requirements thus formed, are used to form the architecture, design test cases, documentation, and code. Each of these products is verified and validated against its specifications and if the product passes its acceptance testing phase, then it is deployed at the end user’s site. These days, the entire process of transforming the goal into the product is done in increments or iterations.

Therefore, in this context, in the IEEE Standard Glossary of Software Engineering Terminology traceability has been defined as “The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor–successor or master–subordinate relationship to one another.”

The entire process of building a data warehouse product involves creation and modification of artifacts either because of new requirements or because of transformation of an artifact into another, more concrete one as a product of a development activity. This leads to *traces* left on the artifacts. Therefore, in this context, IEEE Standard Glossary has defined trace as a relationship between two or more products of the development process.

A trace is also specified as a *non-material indication or evidence showing what has existed or happened*. This definition simply implies that whenever a developer works on an artifact, he leaves traces: There are some software configuration management tools which records which artifact has been processed, how it has been processed, by whom it has been processed and when it was processed.

### 3. Requirements Traceability

When we talk of requirements engineering, the term *traceability* can be defined in two ways.

1. According to Pinheiro [3] traceability is the ability to define, capture, and follow the traces left by requirements on other elements of the software development environment and the traces left by those elements on requirements.
2. Gotel and Finkelstein [28] says traceability is the ability to describe and follow the life of a requirement, in both a forwards and backwards direction.

Since all artifacts are driven by requirements, Stefan Winkler, Jens Von has extended these definitions and has defined the general traceability of all artifacts as the ability to define, describe, capture, and follow traces from and to artifacts throughout the whole software development process [5].
Other useful terms that must be considered while talking of requirements traceability includes: \textit{pre-RS, post-RS, forwards, backwards, horizontal, and vertical traceability}. Refer figure 1.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{traceability_figure.png}
\caption{Terms in Requirements Traceability}
\end{figure}

According to Gotel and Finkelstein [28], \textit{pre-requirements specification (pre-RS) traceability} is concerned with traces occurring during elicitation, discussion, and agreement of requirements until they are included in the requirements specification document. Following the same grounds, \textit{post-RS traceability} is concerned with the stepwise implementation of the requirements in the design and coding phases. It involves documentation of the traces of different manual and automatic transformation steps that are performed to create the system.

Moreover, ANSI/IEEE Std 830–1984 [7] has given the terms \textit{backward traceability} and \textit{forward traceability}. While, backward traceability refers to the ability to follow the traceability links from a specific artifact back to its sources from which it has been derived, forward traceability on the other hand, refers to following the traceability links to the artifacts.

When considering development process, Antoniol et al. [8] has introduced the terms horizontal and vertical traceability to the iteration- and increment-dimensions of the product, respectively.

4. \textbf{Tracking Traces In A Data Warehouse Project}

Learning the definition of a trace is incomplete without knowing what are the syntax and semantics of that trace. In requirements traceability, these are often called traceability schemes. A traceability scheme is usually used to determine for which artifacts and up to which level of detail traces can be recorded. In other words, the traceability scheme defines the constraints needed to guide the recording of traces. Usually, whenever the traceability scheme is represented in a tool, it is technically implemented as a model.

Data warehouse project developers always face the problem of economics before applying traceability practices. This has always inspired the researchers to get insight in what could be achievable in theory, so that they can suggest just the right traceability practices for a particular project.
Had there been no economical issue involved, the best traceability could be achieved by recording any trace to the greatest possible extent. But the economical aspect cannot be ignored and the researchers have to define how much traceability can be achieved without violating the economic constraints.

In order to adopt the most appropriate traceability scheme for the project, the data warehouse development team must record traces that should at least answer the following questions:

- **What information is recorded in the artifact?**
- **Who has created or updated the artifact?**
- **Who are the potential users of the artifact?**
- **What is the source of information (recorded in the artifact)?** whether it is policies, telephone calls, documents, standards, laws, etc
- **How is the information represented?** whether it is documented as formal or informal text, or as graphics, or documented using audio or video recordings
- **When was the artifact created or modified?**
- **Why was the artifact created or updated?**

The answers to these questions are not easy to capture and document. Therefore, traceability scheme must facilitate the process of recording traces and making them persistent.

In addition to the answers to the above a good traceability scheme must record the types of links that must be recorded in order to ensure that the traceability information is correct and complete. These traceability links include:

1. **Dependency** links between two artifacts \(a_1\) and \(a_2\) means that \(a_2\) relies on the existence of \(a_1\). Therefore, any change in \(a_1\) would lead to a change in \(a_2\).
2. **Refinement** links are used in abstraction hierarchies where complex artifacts are broken down into smaller ones.
3. **Evolution** links are used when an artifact \(a_1\) replaces another artifact \(a_2\) (for ex when a new artifact replaces an earlier one)
4. **Satisfiability** links denote that a downstream artifact is developed which compiles with an upstream artifact. For example, a particular code which satisfies one or more requirements.
5. **Overlap** links denote two artifacts describing the common features or aspects of the system. For example, requirements represented in natural language and in formal notation.

**5. Activities supporting traceability**

In order to enable traceability in a data warehouse development project, different activities need to be performed. These activities are used to structure the field of research and provide a basis for a common taxonomy [6]. In this section we will go through these steps and then also analyze the benefits of implementing traceability.

Basically, there are seven activities that must be performed while working with traces: **definition** of traces, **identification** of traces, **planning** for traceability, **recording** traces, **retrieving** them, **using** them, and **maintaining** them (figure 2).

**Definition** of traceability involves the determination of the artifacts for which traces have to be recorded and the relationship that exists between those traces.

**Identification** includes discovering previously unknown traceability relationships between artifacts. This is done by discovering instances of the relationship types determined by the definition activity. This can be done either manually or automatically. While manual identification of relationships is expensive and susceptible to errors [205], automatic identification, on the other hand, does not achieve high levels of precision [9].
Planning for traceability is done simultaneously with the planning phase of data warehouse development project. While planning, the traceability scheme which will be followed and the supporting tools are selected. Additionally, the artifacts which will be created during the project and the detail of the traces which will be recorded for them are identified. Therefore, the outcome of planning is a traceability scheme as well as a list of the supporting tools.

Recording must be performed as a part of data warehouse development activity. The development activity leaves traces on the artifacts which must be stored by populating the tools and data structures. For example, requirements traceability can be performed online (traces are stored automatically by a tool as a by-product of the development activity) or offline (traces are recorded once the development activity is completed). If performed offline, the traces must be recorded as soon as possible because there is always a potential risk of losing some important traceability links.

Retrieval of relevant traceability information which has been recorded previously. The retrieved information is then presented to the user. The technology of retrieving relevant information very much depends on the approach that has been followed to record that information. For example, SQL is used for fetching information from relational databases, the Graph Repository Query Language (GReQL) [10, 11] for graph-based repositories, and ontology query languages such as nRQL [12] for ontologies.

In [13], the author has presented a technique that uses regular expressions to describe patterns of traceable artifacts and their interconnecting relationships that have to be retrieved. In [14], a suite of tools has been mentioned which enable users to retrieve traceability information either by textually specifying the search criteria or by graphically highlighting related artifacts when any one traced artifact is selected in a browser.

Using the traces can be done in different data warehouse development activities. The retrieved data may be used to produce reports, to find relevant information, etc.

Maintaining finally, is an activity that updates and modifies the already existing traceability relationships. Since traceable artifacts are susceptible to constant change during the data warehouse lifetime, traceability relationships have to be updated to reflect the ongoing changes.

Maintaining an artifact differs from a usual recording activity in that a new iteration of definition of traces, identification of traces, planning for traceability, recording traces has to be carried out.
6. Benefits Of Requirements Traceability

When practicing traceability, it may cost some time and money but the time and money invested to track traceability is worth being applied. Some benefits of requirements traceability can be summarized as:

- Helps to prioritize requirements. The requirement that has a higher risk can be given a higher priority.
- If traceability links are properly maintained by the data warehouse development team then in case a new requirement pops up at a later stage then the time, effort and cost that will be required to incorporate that requirement can easily be calculated. Moreover, this will result in better product quality also. [28,15, 16].
- Traceability management ensures the customers that all the requirements have been implemented and the system has no superfluous functionality (the requirement for which was not specified). [16,28,15,17].
- Traceability links helps to discover any kind of inconsistency, incompleteness, and other defects at an early stage of development [15,17].
- Traces can be used to ensure that test cases have been generated for every part of the system. [16,28,17].
- Traces helps in security audits by facilitating the developer to identify the critical elements of the system [28].
- Traces contain a log of events that have occurred during the development process. Using the information stored in the log, one can assess the development process [28].
- Traces enhance the understandability of the system thereby improving the quality of the product delivered. [19, 28, 36].
- When all the artifacts as well as their traces are well maintained and documented, any person can easily understand what, why, when and how is it created? Complete knowledge always helps to take right decisions at the right time thereby delivering a high quality product [16, 15].
- The source of information for a data warehouse can also be legacy systems. When traceability links are maintained, the relationship between the data and functionality of the legacy system and the new data warehouse system can easily be tracked and recorded.
- When we develop data marts from data warehouse or vice versa; or when we write the code to extract data from different operational systems, some code can always be re-used. Traces can therefore, be used to identify re-usable elements and the artifacts that have high cohesion can easily be identified. .

7. Traceability In Practice

Nowadays traceability is being considered as an attribute of data warehouse quality: Rummler et al. [23] has stated that ignoring the traceability aspect has been explicitly criticized in a quality certification audit of a large software development company. This clearly indicates that traceability management is gaining importance in the development phases of the data warehouses.

Many data warehouse development companies are emphasizing on traceability management due to following reasons:

- they are working in domains where regulations demand for traceability, or
- to obtain the quality certification or
- to satisfy the customer’s explicit request to maintain traceability, or
- urge to produce a higher quality product more efficiently.

Ramesh and Jarke [24, 16] categorized the people involved in requirements traceability in two groups:

- low-end users who follow traceability practices as a necessity imposed by regulations, sponsors, or customers. These users find it a cumbersome as well as useless job to record traces. Thus they do not invest
much effort in performing the activities involved in traceability.

- **high-end users** who appreciate the benefits of traceability and prefer to set up tools, technologies, strategies and policies to record and use traces. Such types of users are always keen to follow best practices and end up in building up a set of success stories.

These days, requirements management tools facilitate the users to follow requirements traceability practices. Neumuller and Grunbacher [26] have found that traceability is usually practiced in large companies but even the small ones can also use these practices beneficially.

Kirova et al. [27] has stated the experience of evaluating traceability methodologies, and developing a custom tool which integrates with other tools in the organization. They have observed that reports which needed two days to be formulated could now be created in few seconds.

Despite all goods, a majority of the stakeholders are not familiar with tools to record, manage, and use traces [28]. Gills [29] have found that for each project the users need to create its own trace metamodel almost from scratch. Consequently, more time, effort and resources are needed. This becomes even more frustrating for the low-end users. Therefore, many people neither maintain nor manage traces in a controlled and disciplined way.

8. Limiting Factors

Brooks [32, 14] have stated that it is impossible to capture traces completely. He has found that when traces are maintained manually, people incorporate implicit, social, and tacit knowledge which cannot be formalized, recorded and managed completely. There are problems even with formal modeling techniques as there are always some aspects remain which cannot be captured in explicit traces. Some reasons for this can be given as:

All the traces can not be identified automatically. There has to be some human intervention to identify the traces completely, to understand the semantics of information and to record meaningful traces properly. However, no tool supports this.

Data warehouse is developed using several independent tools. Lack of integration of tools is another point of concern. Recording of traces is not enough. As and when requirements or artifacts change, the traces needs to be updated.

The economical benefits of traceability are not easy to calculate [37]. Investing in traceability is more or less like investing in insurance. Many companies under estimate the benefits of traceability and therefore do not invest sufficient resources in following traceability practices.

Moreover, there is no sound empirical proof which can prove that traceability is beneficial for a project. So, if traceability has to be considered as a critical quality attribute for a data warehouse, then research will have to be done on all of these limitations. The efforts made so far are described in the next section.

9. Overcoming Integration Issues

[5] Integration problems can be resolved either by developing *integrative approaches* which adapt standard tools for a common purpose, or by creating *specialized traceability tools* which provides user interfaces and methods especially to perform traceability tasks.

*Integrative approaches*

In this approach, the traceability links between different artifacts generated by different tools have to be stored and maintained. Usually, data warehouse developers who already have a good hands-on experience with their development tools would not want to switch tools just because some other tool supports traceability.
Overcoming recording issues

Making the entire process of recording traces automatically is not possible. However, a lot of research is being done in this area from last several years and efforts are being made to optimize automatic recording and identification of traces in different phases of data warehouse development. These can be classified as:

Rule-based approaches which identifies traces by applying rules to artifacts. They can further be classified as:

- structural which are based on the structure and existing relations of artifacts. For ex, a type of traceability link can be defined as transitive. That is, if a link is recorded from an artifact A to an artifact B and another link from B to C, then the system will automatically detect a link from A to C.

- linguistic which is based on analysis and processing of natural language texts and application of rules based on their syntax. For example, this can easily be done by searching texts for occurrences of common keywords and establishing traceability links based on these keywords [41]. Moreover, this technique can be complemented with natural language processing (NLP) techniques for parsing texts, to analyze the syntax, and to identify words together with their grammatical attributes.

Information retrieval algorithms which deduce traceability links between artifacts from textual documents using information retrieval (IR) methods [36]. The IR approaches are based on natural language processing techniques and returns a set of matching documents from a set of documents for a given query. This is done by first identifying the keywords from each document in the set of documents and then computing the similarity between the words in the query with that of key words in each document. Finally, a list of candidates is presented to the user who can then identify the relevant candidates for his query.

Overcoming trace maintenance issues

These days, recording of traces is an area of active research. Chen and Chou [37] have suggested to use object-oriented languages for expressing the decomposition of artifacts and traceability links between them. They have proposed a framework to define trigger events and consistency checks for traceability links. This will ensure that whenever an artifact is modified, the change is propagated to all affected artifacts so that all the relationships between the artifacts are consistent and in case of any violation or exception, notification events will be triggered.

10. Future Challenges

As of now, there is no agreement on a thoroughly investigated foundation for traceability. The data warehouse development industry still lacks a universal traceability scheme. Though there are different schemes with each one being suitable for a different domain or process but still the adoption of the suitable traceability scheme and guidelines on how to adapt it to different situations is yet to be defined and evaluated.

The descriptive metrics for a traceability scheme and attributes such as granularity, completeness, correctness, and quality of traces recorded for a project has to be defined precisely.

Several researchers have observed that there is a trade-off between applying resource-extensive, but semantically more accurate manual techniques, and cost-efficient, but inaccurate automatic approaches. So, another challenge is to bridge this gap, and find a good balance between the two ways [5].

In the present scenario a framework to compare the research results is badly needed. Though Hayes and Dekhtyar [38] have given a framework to compare requirements traceability link retrieval algorithms but it has been used only by the publishing authors. One strong reason for this is that the framework has not been designed for the comparison of complete traceability methods or tools in general, but only for evaluating and comparing automatic
approaches to trace detection [5]. So, another challenge lies in developing an extensive comparison scheme which should include aspects such as trace granularity and intended trace usage targeted by the approach.

Recording and maintaining traces can be really helpful only if it could be able to solve practical problems of the real world. In [23, 39], the authors have cited that there is a big gap between the industry and the published solutions from the academic research.

Therefore, a lot of effort has to be put in to make traceability a relevant practical skill, and focus must be laid on a better technology transfer of successful traceability techniques to the industry. [5]

Moreover, even today traceability is not supported by current software development processes [40]. There is a lack of guidance, both with respect to traceability planning and the evaluation of cost and benefit, and with respect to when and how actually carrying out traceability-related tasks during software development. The industry also needs guidance for setting up traceability procedures, structures, and schemes.

Furthermore, the data warehouse development organization deploys numerous tools for managing requirements, modeling data, managing metadata, performing ETL, so on and so forth. If all these tools had standardized interfaces, or had supported a common trace data format, traces would have been much easier to record, manage and maintain. Hence, it has become an active area of research.

Industry data is often present in large number of unstructured documents [41]. The current traceability schemes deals with homogeneous and somewhat structured data. But, when applying traceability scheme for building a data warehouse; such traceability schemes cannot be used at all because the source data for the data warehouse is not only heterogeneous but unstructured as well. Therefore, a traceability scheme for such a real world environment is yet to be defined.

All the more, if the data warehouse development process is distributed to different companies and teams, then a lot of work has to be done to integrate these potentially heterogeneous environments. Till date, distributed traceability and re-usability of traces has been an unexplored area.

Last but not the least, the traceability maintenance scheme must be improved by using a versioning system. This is because as requirements change the software also keeps changing during the development process. Till now, many schemes involved deleting of obsolete links and addition of new links which resulted in loss of traceability links. Therefore, version of traces will help to retain information for future references.

11. Conclusion

Initially, traceability was considered to be a part of requirements engineering but now it is a quality attribute and is used to manage traces of artifacts other than requirements. To be used widely in the industry, traceability costs have to be minimized.

Moreover, a lot of research must be done to make traceability management a mature discipline. The researcher must try to improve the basic understanding of traceability and issue guidelines to help the practitioners in the industry. Focus must also be laid on aspects such as tailoring, customization, usability, and ergonomics.

All in all, traceability research is constantly gaining attention and there have been regular workshops covering traceability. This development will most likely continue in the future and traceability will continue to be a serious research topic.
12. References

5. Stefan Winkler, Jens von Pilgrim In A survey of traceability in requirements engineering and model-driven development. In Springer-Verlag 2009
6. Hannes Schwarz, Jürgen Ebert, Andreas Winter In Graph-based traceability: a comprehensive approach. In In Springer-Verlag 2009
26. Neumuller, C., Grünbacher, P.: Automating software traceability in very small companies: a case study and...