A MANY-TO-MANY (M:N) RELATIONAL MODEL TO IMPROVE REQUIREMENTS TRACEABILITY

Craig L. Williams
Associate Professor of Management
School of Management and Business
St. Edwards University
3001 South Congress Avenue
Austin, TX 78704-6489
Phone: (512) 233-1441
E-mail: craiglw@stedwards.edu

ABSTRACT

Gotel and Finkelstein have defined requirements traceability as the ability to describe and follow the life of a requirement in both the forward and backward direction (i.e. from its origin, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases). Research has shown that inadequate traceability is an important contributing factor to project failures and budget overruns.

One of the many challenges of requirement traceability is poor tool support. The best known method for tracing requirements to their outcomes is a Traceability Matrix: for each requirement, the matrix identifies the outcome. The main problem with this approach is that it assumes that for each requirement there is a single outcome, and for each outcome, a single requirement. Using relational database design models we will show haw it is possible to map a requirement to multiple outcomes and an outcome to multiple requirements.
Keywords: Project requirements, Requirements traceability, Traceability matrix, relational Database.

Introduction

A requirement defines a condition or capability needed to solve a problem or achieve an objective that must be met by a system to satisfy a contract, standard, or specification. Requirements describe the features, functions, capabilities, and characteristics of the end product, service, or result of the project. For example, a business-to-business (B2B) may require a drop down menu as a feature of the web application. The web application may also require an advanced search feature as part of the application.

Requirement errors are the largest class of errors typically found in a project. Hooks and Farry (2001) estimate the requirement errors account for 41 – 56 percent of errors discovered. Reducing requirement errors can be the single most effective action developers can take to

improve project outcomes. Identifying omitted requirements and finding errors during the requirements stage, as opposed to later stages of the life cycle, provides great leverage and cost savings.

In addition, requirements-related activities can reduce rework substantially. The cost of rework is typically 45 percent of projects. [Dion 1993, McConnell 1996].

REQUIREMENT TRACEABILITY

Gotel and Finkelstein (1994) have defined requirements traceability as the ability to describe and follow the life of a requirement in both the forward and backward direction (i.e. from its origin, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases). Forward traceability looks at tracing the requirements sources to their resulting product requirement(s) to ensure completeness of the product specification and tracing each unique requirement forward into the design that implements that requirement. Backward traceability looks at tracing each work product back to its associated requirement and tracing each requirement back to its source(s) (Westfall 2006). This is illustrated in the diagram below.

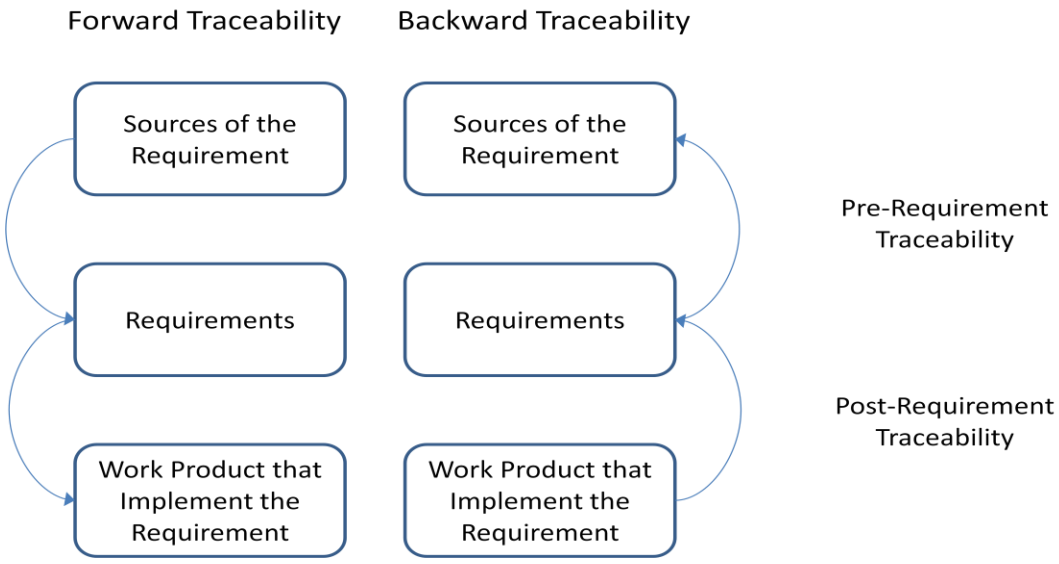

Figure 1. Bidirectional (Forward and Backward Traceability. (Westfall 2006)

In addition, Gotel and Finkestein (1994) also introduced two fundamental types of requirement traceability. They define pre-requirement specification traceability as those areas that are concerned with all aspects of a requirement's life prior to its inclusion into the requirement specification. Post-requirement specification traceability is defined as those areas that are concerned with all aspects of a requirement's life after to its inclusion into the requirement specification. This is illustrated in Figure 2.

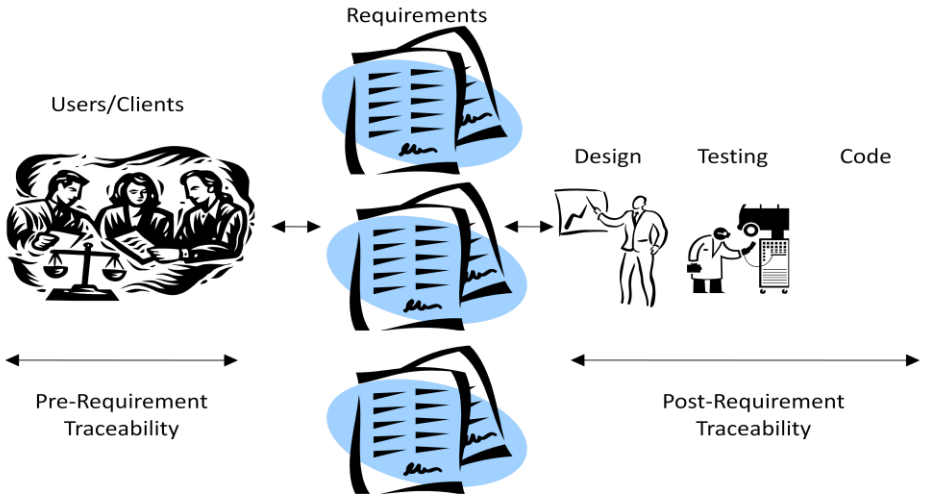

Figure 2: Pre-requirement and post-requirement traceability.

Requirements traceability is recognized as a concern in an increasing number of standards and guidelines for systems and software requirements engineering (Dorfman and Thaylor 1990). Research has shown that inadequate traceability is an important contributing factor to project failures and budget overruns. Requirements traceability has been demonstrated to provide many benefits to organizations that make proper use of traceability techniques (Kannenberg 2009). Also, important benefits from traceability can be realized in the areas of: project management, process visibility, verification and validation and maintenance (Palmer, 1997)

Requirement traceability makes project management easier by simplifying project estimates. By following traceability links, a project manager can quickly see how many artifacts will be affected by a proposed change and can also make an informed decision about the costs and risks associated with that change. Project managers can also utilize traceability to assist in measuring project progress.

Traceability offers improved process visibility to both customers and project engineers. Traceability can be used to improve customer satisfaction and provide project engineers with access to contextual information that can be used to assist them in determining where the requirement came from, its importance, its implementation, and how it was tested.

One of the most significant benefits provided by traceability can be realized during the validation and verification stages of a project. Traceability offers project engineers the ability to assess system functionality on a requirement by requirement basis, from its origin through the testing of each requirement. If properly implemented, traceability can be used to prove that a system complies with all its requirements and that they have been implemented correctly.

Traceability is also a valuable tool during the maintenance phase of a project. The requirement that were defined at the start of a project can often change even after the project has been completed. It is important to be able to assess the potential impact of these changes and to determine what requirements need to be updated in order to fulfill a change request.

CHALLENGES OF REQUIREMENT TRACEABILITY

In spite of the benefits that traceability offers, its practice faces many challenges. These challenges can be identified under the area of;

- Cost
- Maintaining traceability through change
- Different viewpoints on traceability
- Organizational problems and politics
- Poor tool support.

Cost

A major challenge facing the implementation of traceability is simply the cost involved. As the system grows in size and complexity, requirement traces become complex and expensive (Heindl and Biffl, 2005). One method of dealing with the high cost of traceability is to practice value-based requirement tracing. Value-based requirement tracing prioritizes all of the requirements in the system, with the amount of time and effort expended on tracing each requirement depending on the priority of that requirement (Heindl and Biffl, 2005). Alternatively, the high cost of traceability can be viewed with the attitude that the costs incurred will save much greater costs later on in the development process due to the benefits that traceability offers.

Managing Change

Managing traceability through system changes can be another significant challenge. Dealing with change and its impact on traceability is a difficult prospect. Studies have shown that change can be expected throughout the lifecycle of nearly every project (Wiegers 2003, Boehm 2003). Strong discipline in maintaining the accuracy of traceability is uncommon. A lot of manual time and effort is still required to update the traceability data. Focusing on the long-term benefits rather that the short-term costs are is the best way an organization can maintain a healthy attitude toward the costs of maintaining traceability information.

Different viewpoints on traceability

A factor contributing to poor support for traceability are the differing viewpoints regarding traceability. Some stakeholders require traceability but provide little guidance as to why and how it should be performed. Sponsors and senior management view traceability as something that needs to be implemented to merely comply with standards (Ramesh 1998). The best way to deal with the differing viewpoints on traceability is to create an organizational policy on traceability.

Organizational Problems

Organizational problems provide a significant challenge to the implementation of traceability. Many organizations view traceability as a mandate from sponsors or a tool for compliance (Ramesh 1998). Politics and lack of training poses additional challenges. Individuals feel that traceability data will be used against them in performance reviews or as a threat to their job security (Jarke 1998). Another problem is that other organizations do not train their employees regarding the importance of traceability.

Poor Tool Support

Poor tool support is perhaps the biggest challenge to the implementation of traceability (Kranneberg 2009). Traceability tool penetration throughout the software engineering industry is surprising low, at about 50% throughout the industry (Gills 2005, Lempla and Miller 2006). Traceability information can be captured manually through utilizing techniques such as traceability matrices. Traceability information can be captured manually through the use of techniques such as traceability matrices. A traceability matrix can be defined as "a table that illustrates logical links between individual functional requirements and other system artifacts" (Wiegers 2003). See Table 1 for an example of a traceability matrix.

Manual traceability methods are not suitable for the needs of the software engineering industry. The number of traceability links that need to be captured grows exponentially with the size and complexity of the software project (Cleland-Huang et al 2003). Manual traceability methods are also vulnerable to changes in the system as well as being prone to errors that are not easy to catch. Because of these disadvantages, manual traceability methods are not suitable for anything other than the smallest projects. However, Gotel and Finkelstein (1994) found that manual traceability methods were preferred due to the shortcomings in available traceability tools.

Systems Requirement	Software Requirement	Design Element	Code Module	Test Case
005-00150-80#00505	005-00150-80#00112	Airspeed Calculation	Calculate_airspeed()	Tc_103.doc
005-00150-80#00506	005-00150-80#00234	Airspeed Display	Display_airspeed()	Tc_125.doc

Table 1: Traceability Matrix (Kranneberg 2009)

One of the main problems associated with traceability matrixes is that they assume that for each requirement there is a single origin and for each origin there is single requirement and for each requirement there is a single outcome, and for each outcome, a single requirement. Traceability matrices can be established using a variety of tools including requirements management software, databases, or spreadsheets. Using relational database design models we will show how it is possible to map a requirement to multiple outcomes and an outcome to multiple requirements.

RELATIONAL DATABASE DESIGN

Entity-Relationship (ER) models and relational database design identifies three different types of relationships between data entities. An entity is a person, place, thing, or anything you want to collect data on. The three different types of relationships are: One-to-One Relationship, One-to-Many Relationship, and Many-to-Many Relationship. In a one-to-one relationship, a single instance of one entity is related to a single instance of another entity. Examples are available in most textbooks on database development. For example, a single client may desire a geographical

search function on their web application. To relate the two entities, the primary key of one entity is added to the other entity as a foreign key, or vice versa. This is illustrated in Figure 3.

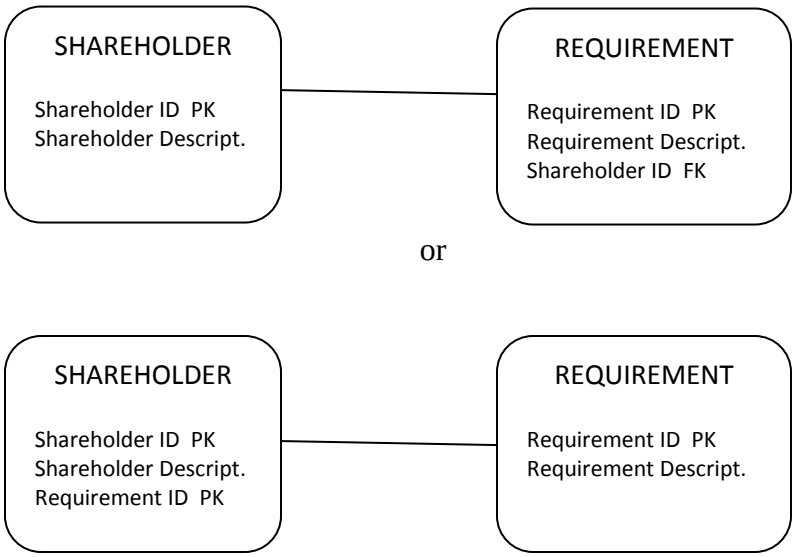

Figure 3. Illustration of a one-to-one relationship

In a one-to-many relationship, a single instance of one entity is related to many instances of another entity. For example, a single client may desire several design features, such as font size, font color, back ground color. To relate the two entities, the primary key of the one entity is added to the many entity as a foreign key. This is illustrated in Figure 4.

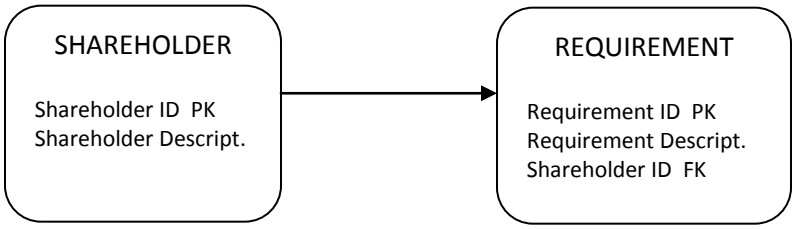

Figure 4. Illustration of a one-to-many relationship

In a many-to-many relationship, an instance of one entity is related to many instances of second a second entity and an instance of the second entity is related to many instances of the first entity. For example, a client may want several design features, such as font size, font color, background color and the font size and color may be required by several clients. To relate the two entities, an intersection entity is created and the primary key of the two entities are added to the intersection entity as foreign keys. This is illustrated in Figure 5.

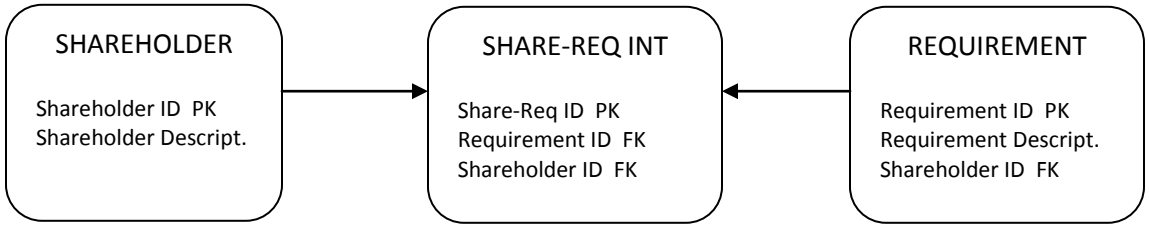

Figure 5. Illustration of a many-to-many relationship

Consider the case of a gas pump and pumping gas. The stakeholders are the customer, the gas station or convenience store, and the credit card company. The requirements are the ways in which the transaction can be terminated.

When the stakeholder CUSTOMER wishes to terminate a transaction, he or she can do so in one of two ways. First, the customer may terminate the transaction by pressing the CANCEL button on the gas pump. The second way a customer may terminate a transaction is to return the gas nozzle at the end of the transaction. Both termination methods can be related to multiple segments of computer code and testing plans.

When the stakeholder STORE wishes to terminate a transaction, they can set an amount limit at the terminal inside the store and the transaction will be terminated when the dollar amount is reached. This termination method can also be related to multiple segments of computer code and testing plans.

The CREDIT CARD COMPANY stakeholder can terminate a transaction in one of three methods. The transaction can be terminated when an amount limit, set by the credit card company, is reached. The transaction may also be terminated when the credit card is refused or if the customer enters an incorrect PIN number. Again, these termination methods can be related to multiple segments of computer code and testing plans.

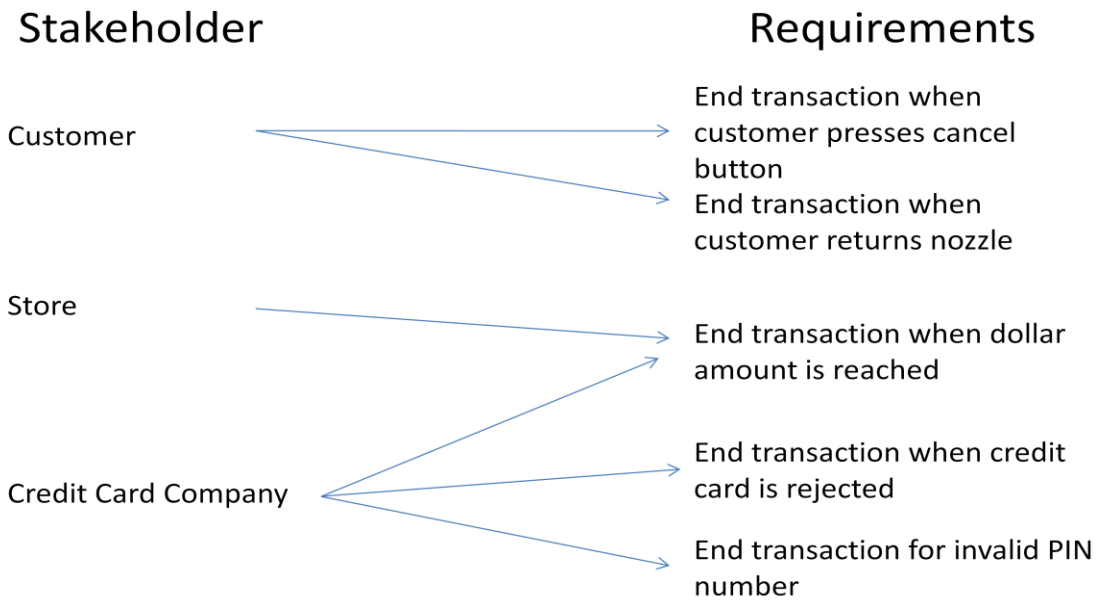

Figure 6. Stakeholder and Requirement Relationships

COMMENTS AND CONCLUSIONS

As mentioned earlier in this paper, traceability matrices are appropriate for anything other than the smallest projects. With the development of relational database design models and the ease of use of database development tools, a requirement traceability database is a more appropriate tool for capturing project requirements.

REFERENCES

Boehm, B. Value Based Software Engineering. *ACM SIGSOFT Software Engineering Notes*, 2 (2003).

Cleland-Huang, J., Chang, C. and Christensen, M. Event-Based Traceability for Managing Evolutionary Change. *IEEE Transactions on Software Engineering* 9 (2003): 796-810.

Dion, R. Process Improvement and the Corporate Balance Sheet. IEEE Software, October 1993, pp 28-35.

Dorfman, M. and Thayer, R., Standards, Guidelines, and Examples on Systems and Software Requirements Engineering, *IEEE Computer Society Press Tutorial*, (1990).

Gills, M. Software Testing and Traceability. University of Latvia. 2005 www3.acadlib.lv/greydoc/Gilla_disertacija/MGills_ang.doc.

Gotel, O. and A. Finkelstein, An Analysis of the Requirements Traceability Problem. Proc. of the First International Conference on Requirements Engineering. Colorado Springs, CO, (1994) 94-101.

Heindl, M. and Biffl, S. A Case Study on Value-Based Requirements Tracing. *Proceedings of the Tenth European Software Engineering Conference.* Lisbon, Portugal, (2005) 60-69.

Hooks, I.F., and K.A. Farry. Customer-Centered Products: Creating Successful Products through Smart Requirements Management, New York, NY, AMACOM, (2001).

Jarke, M. Requirements Tracing. *Communications of the ACM* 12 (1998) 32-36.

Kannenberg, A. and H. Saidian, Why Software Requirements Traceability Remains a Challenge, The Journal of Defense Software Engineering, July/August 2009.

Kroenke, D.M., Database Processing: Fundamentals, Design, and Implementation, Prentice-Hall, Upper Saddle River, NJ, (1998).

Larson, E. and R. Larson. Requirements Management: Part 1: Requirements Planning, Watermark Learning Publications, Minneapolis, MN (2009).

Lempia, D. and Miller, S. Requirements Engineering Management. Proc of the National Software and Complex Electronic Hardware Standardization Conference. Atlanta, GA (2006.).

McConnell, S. Rapid Development: Taming Wild Software Schedules. Redmond, WA, Microsoft Press, (1996).

Palmer, J.D., Traceability, *Software Requirements Engineering*. Richard H, Thayer and Merlin Dorfman, eds. New York: IEEE Computer Society Press, (1997).

Ramesh, B. Factors Influencing Requirements Traceability Practice. *Communications of the ACM* 12 (1998): 37-44.

Westfall, L., Bidirectional Requirements Traceability, (2006) www.compaid.com/caiinternet/ezine/westfall-bidirectional.pdf.

Wiegers, K. Software Requirements. 2nd ed. Redmond, WA: Microsoft Press, (2003).

Young, R. Project Requirements: A Guide to Best Practices, Management Concepts, Vienna, VA (2006).