Why Software Requirements Traceability Remains a Challenge

Andrew Kannenberg
Garmin International

Dr. Hossein Saiedian
The University of Kansas

Why do so many challenges exist in traceability practices today? While many of these challenges can be overcome through organizational policy and procedure changes, quality requirements traceability tool support remains an open problem. After discussing the basics of software requirements traceability, this article shows why neither manual traceability methods nor existing COTS traceability tools are adequate for the current needs of the software engineering industry.

Software products are increasingly being deployed in complex, potentially dangerous products such as weapons systems, aircraft, and medical devices. Software products are critical because failure in these areas could result in loss of life, significant environmental damage, and major financial loss. This might lead one to believe that care would be taken to implement these software products using proven, reproducible methods. Unfortunately, this is not always the case.

In 1994, a Standish Group study [1] found that 53 percent of software projects failed outright and another 31 percent were challenged by extreme budget overruns. Since that time, many responses to the high rate of software project failures have been proposed. Examples include the SEI's CMMI®, the ISO's 9001:2000 for software development, and the IEEE's J-STD-016.

One feature that these software development standards have in common is that they all impose requirements traceability practices on the software development process. Requirements traceability can be defined as "the ability to describe and follow the life of a requirement, in both a forward and backward direction" [2]. This concept is shown in Figure 1.

Although many facets of a software project can be traced, the focus of this article is on requirements traceability; therefore, the term *traceability* is used to refer to requirements traceability throughout. See Figure 2, which provides an alternative view to Figure 1.

Research has shown that inadequate traceability is an important contributing factor to software project failures and budget overruns [3]. As a response, there has been an outpouring of research and literature on the subject of traceability, and many organizations have been striving to improve their traceability practices. These efforts have not been in vain. In 2006, The Standish Group updated their 1994 study [4], showing that only 19 percent of software projects failed outright with another 46 percent challenged by budget overruns. The improvement since 1994 is clearly shown in Table 1 (see page 16); however, room for growth remains.

Although the importance of traceability seems to be well-accepted in the software engineering industry, research suggests that many organizations still do not understand the principles of traceability and are struggling with implementing traceability practices in the software development life cycle [5]. Perhaps the biggest need is for a better understanding of why traceability is important and the challenges facing its implementation. This article attempts to address this need by studying the factors that make traceability important and discusses the challenges facing traceability practices in industry.

> *"Through traceability, each project engineer has access to contextual information that can assist them in determining where a requirement came from, its importance, how it was implemented, and how it was tested."*

The Importance of Traceability

Requirements traceability has been demonstrated to provide many benefits to organizations that make proper use of traceability techniques. This is why traceability is an important component of many standards for software development, such as the CMMI and ISO 9001:2000. Important benefits from traceability can be realized in the following areas: project management, process visibility, verification and validation (V&V), and maintenance [6].

Project Management

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 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. As requirements are traced to code and later to test cases, management can estimate the project completion status based on how many requirements have been traced to artifacts created later in the development cycle. This information can be used to estimate the schedule for a project during development and can be used to assess risk.

Process Visibility

Traceability offers improved process visibility to both project engineers and customers. Through traceability, each project engineer has access to contextual information that can assist them in determining where a requirement came from, its importance, how it was implemented, and how it was tested. Traceability can also be viewed as a customer satisfaction issue. If a project is audited or in the case of a lawsuit, traceability can be used to prove that particular requirements were implemented and tested. The availability of this information also increases customer confidence and satisfaction because it reassures customers that they will receive the product that they requested.

Verification and Validation

The most significant benefits provided by traceability can be realized during the V&V stages of a software project. Traceability offers the ability to assess system functionality on a per-requirement basis, from the

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

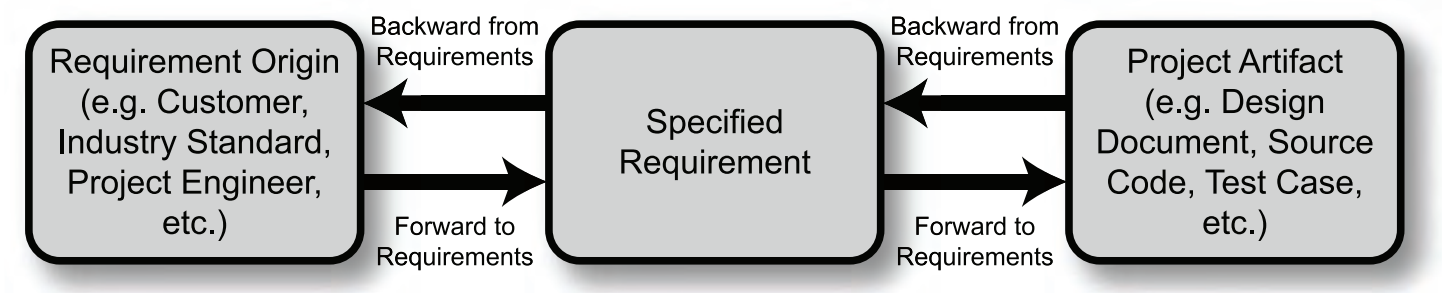


Figure 1: *A View of Software Requirements Traceability*

origin through the testing of each requirement. Properly implemented, traceability can be used to prove that a system complies with its requirements and that they have been implemented correctly. If a requirement can be traced forward to a design artifact, it validates that the requirement has been designed into the system. Likewise, if a requirement can be traced forward to the code, it validates that the requirement was implemented. Similarly, if a requirement can be traced to a test case, it demonstrates that the requirement has been verified through testing. Without traceability, it is impossible to demonstrate that a system has been fully verified and validated.

Maintenance

Traceability is also a valuable tool during the maintenance phase of a software project for many of the same reasons that it is valuable for project management. Initially defined requirements for a software project often change even after the project is completed, and it is important to be able to assess the potential impact of these changes. Traceability makes it easy to determine what requirements, design, code, and test cases need to be updated to fulfill a change request made during the software project's maintenance phase. This allows for estimates of the time and cost required to make a change.

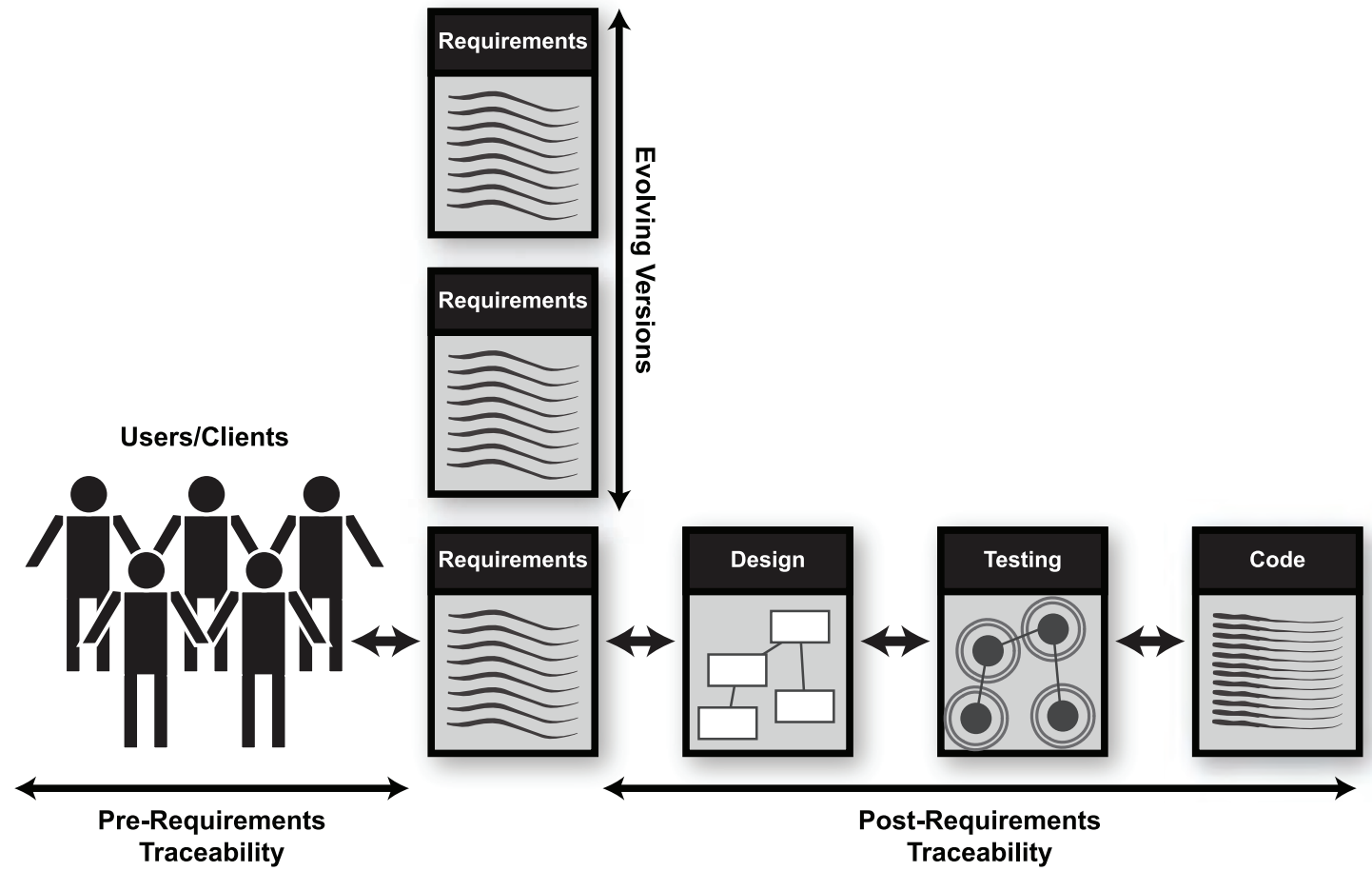
Challenges in Requirement Traceability

In spite of the benefits that traceability offers to the software engineering industry, its practice faces many challenges. These challenges can be identified under the areas of cost in terms of time and effort, the difficulty of maintaining traceability through change, different viewpoints on traceability held by various project stakeholders, organizational problems and politics, and poor tool support.

Cost

One major challenge facing the implementation of traceability is simply the costs involved. As a system grows in size and complexity, capturing the requirement traces quickly becomes complex and expensive [7]. Because of this, the budget for a project implementing traceability must be greater than that of a project without it. However, a project implement-

Figure 2: *An Alternative View of Software Requirements Traceability*



Year	Failed Projects	Challenged Projects	Successful Projects
1994	53%	31%	16%
2006	19%	46%	35%

Table 1: *Comparison of the Standish Group's 1994 and 2006 Results*

ing traceability is far less likely to incur major budget overruns because traceability can detect project problems early in the development process when they are easier and cheaper to correct.

One method of dealing with the high cost of traceability is to practice value-based requirement tracing instead of full 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 [7]. This can save a significant amount of effort by focusing traceability activities on the most important requirements. However, value-based tracing requires a clear understanding of the importance of each requirement in the system; it may not be an option if full tracing is a requirement of the customer or the development process standards used for the project.

Alternatively, the high costs of traceability can be approached with the attitude that the costs incurred will save much greater costs further along in the development process due to the benefits that traceability offers to software projects. This method does not solve the problem of the high costs of traceability implementation, but it promotes a healthy attitude towards managing costs for the entire duration of a project instead of merely looking at the short term.

Managing Change

Maintaining traceability through changes to the system is another significant challenge. Studies have shown that change can be expected throughout the life cycle of nearly every software project [8, 9]. Whenever such changes occur, it is necessary to update the traceability data to reflect these changes. This requires discipline on the part of those making the change to update the traceability data, which can be costly in terms of time and effort when the changes are extensive. Unfortunately, strong discipline in maintaining the accuracy of traceability is uncommon, leading to a practice of disregarding traceability information in many organizations [10].

Dealing with change and its impact on traceability is a difficult prospect. Some COTS tools offer assistance with identifying the impact of change on existing traceability data; however, a lot of manual time and effort is still required to update the traceability data [11]. Alternatively, training can help users understand the importance of discipline in maintaining traceability data when changes occur. Focusing on the long-term benefits of traceability instead of the short-term costs can help an organization sustain a healthy attitude toward the costs of maintaining traceability data amidst change.

"If an organization has clear traceability policies in place and provides training on how to comply with these policies, it is likely that traceability will be implemented in a thorough manner consistent with policy."

Different Stakeholder Viewpoints

A contributing factor to poor support for traceability may be the fact that many different viewpoints regarding traceability exist, even among different project stakeholders. These different viewpoints exist primarily because current software engineering standards typically require traceability to be implemented but provide little guidance as to why and how it should be performed [5].

Project sponsors and upper management often view traceability as something that needs to be implemented merely to comply with standards [12]. This leads to a desire to spend as little time as possible on traceability because the benefits outside of standards compliance are not well understood. This viewpoint will likely conflict with that of project engineers familiar with the importance of traceability who will want to ensure that the traceability performed is complete and correct. Perhaps the best way to deal with the problem of different stakeholder viewpoints on traceability is to create an organizational policy on traceability to apply uniformly to all projects. Because the standards requiring traceability are vague, organizations have a lot of leeway in getting their own procedures in place for implementing traceability. This can reduce the amount of confusion about traceability and leads to more consistent viewpoints among the stakeholders involved.

Organizational Problems

Organizational problems also provide a significant challenge to the implementation of traceability. Many organizations view traceability as a mandate from sponsors or a tool for standard compliance [12]. Typically, these organizations do not have a commitment to comprehensive traceability practices. This leads to an ad-hoc practice of traceability, where traceability data is created and maintained haphazardly.

Lack of training poses another challenge [2]. Many organizations do not train their employees regarding the importance of traceability and traceability is not emphasized in undergraduate education. This can lead to resentment on the part of those tasked with creating and maintaining traceability information. They may view the added workload as impacting their productivity due to a staff's insufficient understanding of why traceability is important.

Politics can also play a role. Individuals may be concerned that traceability data will be used against them in performance reviews or as a threat to their job security [13]. This issue can arise because the individual who captures a piece of traceability

Table 2: *Example Traceability Matrix*

System Requirement	Software Requirement	Design Element	Code Module	Test Case
005-00150-80#00505	005-00150-85#00112	Airspeed Calculation	calculate_airspeed()	tc_103.doc
005-00150-80#00506	005-00150-85#00234	Airspeed Display	display_airspeed()	tc_125.doc

information is usually not the one who makes use of it later. Those involved with creating and maintaining traceability data may feel that they are helping others to look good while reducing their own productivity.

The easiest way to correct organizational problems related to traceability is through the use of policy and training. If an organization has clear traceability policies in place and provides training on how to comply with these policies, it is likely that traceability will be implemented in a thorough manner consistent with policy [12].

Poor Tool Support

Poor tool support is perhaps the biggest challenge to the implementation of traceability. Even though the International Council on Systems Engineering (INCOSE) has a survey (see [14]) that lists 31 different tools claiming to provide full traceability support, traceability tool penetration throughout the software engineering industry is surprisingly low. Multiple studies have found the level of commercial traceability tool adoption to be around 50 percent throughout industry [15, 16]. The majority of the remaining companies utilize manual methods (such as manually created traceability matrices for implementing traceability), and a small percentage develop their own in-house traceability tools.

Problems With Manual Traceability Methods

Traceability information can be captured manually through utilizing 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" [8]. Since traceability matrices are in tabular form, they are typically created using a spreadsheet or a word processing application's table function and are independent of the artifacts from which they've captured traceability information. An example traceability matrix is shown in Table 2.

Unfortunately, manual traceability methods are not suitable for the needs of the software engineering industry. In [17], the authors found that the number of traceability links that need to be captured grows exponentially with the size and complexity of the software system. This means that manually capturing traceability data for large software projects requires an extreme amount of time and effort.

Manual traceability methods are also vulnerable to changes in the system. If changes occur to any elements captured in the traceability data, the affected portions of the traceability data must be updated manually. This requires discipline and a significant amount of time and effort spent on link-checking throughout the traceability data. Because of this, it is easy for manually created traceability data to become out-of-sync with the current set of requirements, design, code, and test cases.

Manual traceability methods are also prone to errors that are not easy to catch. Errors can arise from simple typographic mistakes, from inadvertently overlooking a portion of the traceability data (such as an individual requirement), or from carelessness by the individual capturing the data. Because traceability artifacts for large projects are often hundreds or even thousands of pages in length, such errors are difficult to detect when depending on manual methods for error checking.

" ... the number of traceability links that need to be captured grows exponentially with the size and complexity of the software system. This means that manually capturing traceability data for large software projects requires an extreme amount of time and effort."

Because of these disadvantages, manual traceability methods are not suitable for anything other than small software projects. Ralph R. Young stated: "in my judgment, an automated requirements tool is required for any project except tiny ones" [18]. Similarly, Balasubramaniam Ramesh (in [12]) found that traceability is error-prone, time-consuming, and impossible to maintain without the use of automated tools. Then why would nearly 50 percent of software companies use manual traceability methods? Is it because they are all developing tiny projects? This is highly unlikely. In 1994, Orlena Gotel and Anthony Finkelstein [2] found that manual traceability methods were preferred in the industry due to shortcomings in available traceability tools. It is apparent that this problem still exists today because manual traceability methods are still preferred by a significant percentage of software organizations.

Problems With COTS Traceability Tools

Regrettably, currently existing COTS traceability tools are not adequate for the needs of the software engineering industry. Studies have shown that existing commercial traceability tools provide only simplistic support for traceability [5]. Surprisingly, the tools that are available do not fully automate the entire traceability process; instead, they require users to manually update many aspects of the traceability data. This has led some researchers to conclude that poor tool support is the root cause for the lack of traceability implementation [19].

COTS tools are typically marketed as complete requirements management packages, meaning that traceability is only one added feature. The traceability features usually only work if the project methodology is based around the tool itself. Unless the project is developed from the ground up using a particular tool, the tool is unable to provide much benefit without significant rework. Support for heterogeneous computing environments is also lacking.

Although most tools support the identification of impacted artifacts when changes occur, they typically do not provide assistance with updating the traceability links or ensuring that the links and affected artifacts are updated in a timely manner [17]. This means that even when tools are used, the traceability information is not always maintained, nor can it always be trusted to be up-to-date and accurate. This problem is exacerbated by the fact that tools typically only allow primitive actions to be taken (in regards to traceability).

Another issue with tools is that they often suffer problems with poor integration and inflexibility. This has led at least one researcher to conclude that existing traceability tools have been developed mostly for research purposes, and that many projects are still waiting for tools that do not require a particular development or testing methodology [15].

Cost is another major disadvantage. Although the licensing fees vary per tool,

Coming Events

August 24-28
13th International Software Product Line Conference
San Francisco, CA
www.sei.cmu.edu/activities/splc2009

August 31-September 4
17th IEEE International Requirements Engineering Conference
Atlanta, GA
www.re09.org

September 8-10
2009 Unique Identification Forum
Orlando, FL
www.uidforum.com

September 21-24
4th Annual Team Software Process Symposium
New Orleans, LA
www.sei.cmu.edu/tsp/symposium

October 4-9
ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems
Denver, CO
www.modelsconference.org

October 18-21
MILCOM 2009
Boston, MA
www.milcom.org

October 19-23
International Conference on Software Process Improvement 2009
Washington, D.C.
www.icspi.com

2010
22nd Annual Systems and Software Technology Conference

www.sstc-online.org

Coming Events: **Please submit coming events that are of interest to our readers at least 90 days before registration. E-mail announcements to: nicole.kentta@hill.af.mil.**

the price tends to be thousands of dollars up front, per license, in addition to yearly maintenance fees. Because of this, the cost of using COTS tools is often prohibitive, even for fairly small teams. Such tools are also decoupled from the development environment, meaning that important traceability information—such as code modules that implement requirements—may not be available [20]. For this reason, Ramesh concluded that COTS tools have "very limited utility in capturing dynamic traceability information" [12].

Few solutions are available for the problem of poor tool support for traceability. Many organizations shun COTS tools altogether due to their high cost and inflexibility, instead making use of manual methods such as traceability matrices. Another approach—common among organizations concerned with high-quality traceability information—is to develop elaborate in-house tools and utilities to implement traceability [5]. Unfortunately, this approach is not always feasible since many organizations do not have the manpower or the knowledge necessary to develop such tools. Therefore, poor tool support for traceability currently remains an open problem.

Conclusions

This article has presented an introduction to the benefits offered by traceability and the challenges faced by the practice of traceability in software projects today. Traceability offers benefits to organizations in the areas of project management, process visibility, V&V, and maintenance. Traceability needs to be hardcoded into a process to be replicated iteratively on each and every project. Unfortunately, many organizations struggle to understand the importance of traceability, meaning that these benefits often go unrealized.

In spite of the benefits offered by traceability, its implementation still faces many challenges, especially in the areas of cost, change management, organizational problems, and poor tool support. The lack of quality COTS traceability tools is a significant challenge facing the implementation of traceability in the software engineering industry today. These challenges lead many organizations to implement only as much traceability as is required by their customers.

The challenges faced by traceability are not new. Many of these challenges can be mitigated by process and organi-

zational changes by groups interested in improving their traceability practices. Poor tool support for traceability remains an exception; this is an area that is still an open problem. Existing tools are costly and provide only partial traceability support. This means that implementing traceability is often tedious, requiring a large amount of manual effort.

The lack of quality tools for implementing traceability is not a technically insurmountable problem. The solution simply involves creating cost-effective traceability tools that improve upon the design and feature set of currently available tools. Such tools would serve to greatly improve the practice of traceability in the software engineering industry.◆

References

1. The Standish Group. The Chaos Report. 1994 <www.ibv.liu.se/content/1/c6/04/12/28/The%20CHAOS%20Report.pdf>.
2. Gotel, Orlena, and Anthony Finkelstein. An Analysis of the Requirements Traceability Problem. Proc. of the First International Conference on Requirements Engineering. Colorado Springs, 1994: 94-101.
3. Dömges, Ralf, and Klaus Pohl. "Adapting Traceability Environments to Project Specific Needs." Communications of the ACM 41.12 (2008): 55-62.
4. The Standish Group. The Chaos Report. 2006.
5. Ramesh, Balasubramaniam, and Matthias Jarke. "Toward Reference Models for Requirements Traceability." IEEE Transactions on Software Engineering 27.1 (2001): 58-93.
6. Palmer, J.D. "Traceability." Software Requirements Engineering. Richard H. Thayer and Merlin Dorfman, eds. New York: IEEE Computer Society Press, 1997.
7. Heindl, Matthias, and Stefan Biffl. A Case Study on Value-Based Requirements Tracing. Proc. of the 10th European Software Engineering Conference. Lisbon, Portugal, 2005: 60-69.
8. Wiegers, Karl. Software Requirements. 2nd ed. Redmond, WA: Microsoft Press, 2003.
9. Boehm, Barry. "Value Based Software Engineering." ACM SIGSOFT Software Engineering Notes 28.2 (2003).
10. Clarke, Siobhán, et al. Subject-Oriented Design: Towards Improved Alignment of Requirements, Design, and Code. Proc. of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. Dallas, TX: 325-329.
11. Cleland-Huang, Jane, Carl K. Chang, and Yujia Ge. Supporting Event Based Traceability Through High-Level Recognition of Change Events. Proc. of the 26th Annual International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment. Oxford, England, 2002: 595-602.
12. Ramesh, Balasubramaniam. "Factors Influencing Requirements Traceability Practice." Communications of the ACM 41.12 (1998): 37-44.
13. Jarke, Matthias. "Requirements Tracing." Communications of the ACM 41.12 (1998): 32-36.
14. INCOSE. "INCOSE Requirements Management Tools Survey." 2008 <www.paper-review.com/tools/rms/read.php>.
15. Gills, Martins. "Software Testing and Traceability." University of Latvia. 2005 <http://www3.acadlib.lv/greydoc/Gilla_disertacija/MGills_ang.doc>.
16. Lempia, David L., and Steven P. Miller. Requirements Engineering Management. Proc. of the National Software and Complex Electronic Hardware Standardization Conference. Atlanta, 2006.
17. Cleland-Huang, Jane, Carl K. Chang, and Mark J. Christensen. "Event-Based Traceability for Managing Evolutionary Change." IEEE Transactions on Software Engineering 29.9 (2003): 796-810.
18. Young, Ralph R. "Twelve Requirement Basics for Project Success." CROSSTALK Dec. 2006.
19. Spanoudakis, George, et al. "Rule-Based Generation of Requirements Traceability Relations." Journal of Systems and Software 72.2 (2004): 105-127.
20. Naslavsky, Leila, et al. Using Scenarios to Support Traceability. Proc. of the Third International Workshop on Traceability in Emerging Forms of Software Engineering. Long Beach, CA, 2005: 25-30.

About the Authors

Andrew Kannenberg is a software engineer at Garmin International in Olathe, Kansas, and is currently working on his doctorate in computer engineering at the University of Kansas. He received a bachelor's degree in computer science from the South Dakota School of Mines and Technology and his master's degree in computer science from the University of Kansas.

Garmin International, Inc.
1200 E 151st ST
Olathe, KS 66062
Phone: (913) 397-8200
E-mail: andy.kannenberg@gmail.com

Hossein Saiedian, Ph.D., is currently a professor of software engineering at the University of Kansas. Saiedian's primary area of research is software engineering and in particular technical and managerial models for quality software development. His past research has been supported by the National Science Foundation as well as regional organizations. He has more than 100 publications on a variety of topics in software engineering and computer science and is a senior member of the IEEE. Saiedian received his doctorate in computing and information sciences from Kansas State University.

The University of Kansas
Electrical Engineering and Computer Science
University of Kansas
12600 Quivira RD
Overland Park, KS 66213
Phone: (913) 897-8515
Fax: (913) 897-8682
E-mail: saiedian@ku.edu