Automating the Management of the Traceability Relation

Hyun Cho
University of Alabama at Birmingham
123D Campbell Hall
1300 University Boulevard
Birmingham, AL, 35294-1170
robusta@uab.edu

ABSTRACT

Software changes are inevitable in the software development lifecycle. The scope and cost of such changes can be estimated through impact analysis. However, impact analysis is often performed only when it is absolutely necessary and the accuracy of impact analysis is not generally satisfactory. This is due to the labor intensive and error prone nature of maintaining traceability relations. This poster introduces an automated approach to traceability management that informs development activities and assists in tool integration.

1. Motivations

Traceability relations assist a developer in following the lifecycle of a requirement in a forward and backward direction [2]. Being able to trace the causal relationships between requirements and other software artifacts offers several benefits, such as: 1) analyzing the scope of the impacted artifacts from the changes, 2) identifying reusable components, and 3) helping to understand the system comprehensively. However, maintaining traceability relations is often a tedious and error prone task that requires intensive labor. Many researchers have proposed several methods [1][5][6] and tools [3][4] that can address this issue, but they are not able to completely automate the process of traceability relation management (TRM). The goal of the research described in this poster is to automate TRM across the entire development phases. This will make traceability relations readily available for impact analysis and reduce the burden of TRM.

2. Activity-Driven TRM

Software development begins by analyzing requirements or reviewing change requests. After new requirements or change requests are analyzed, a chain of development activities are defined for system design, implementation, testing, and release. Thus, activities govern the evolution of the related software artifacts and each activity can be considered a unit of change. As the number of activities is large even in a small project, project management tools or change management (CM) tools are often used to manage the development process. These tools help engineers to manage and control the entire progress of the project. In addition, the activities can be exported from one tool to another tool to manage different aspects of the activities. For instance, activities in MS Project [7], which is widely used to manage projects, are defined to manage the project schedule, cost and resource management. MS Project allows exporting its activities to several tools, including CM tools such as IBM Rational ClearQuest [10] and Change [9]. If the activities are exported to change management, each activity governs the change process through a pre-defined workflow. In addition, activity-driven development offers benefits when change and configuration management processes are integrated and governed by the activity. For example, IBM Rational ClearCase [8], one of the market leading software configuration management (SCM) toolsuites, provides integration with IBM Rational ClearQuest. By integrating the tools, each activity is mapped with a set of configuration items to resolve an issue described in the activity. As a result, all software artifacts that are maintained under SCM tools can be linked to each other through these activities.

In this poster, we present how to automate the generation and management of traceability relations through tool integration. Specifically, the poster describes how TRM activities can govern the evolution of artifacts and can be used to manage the relations. Figure 1 shows the approach for automating the TRM activity.

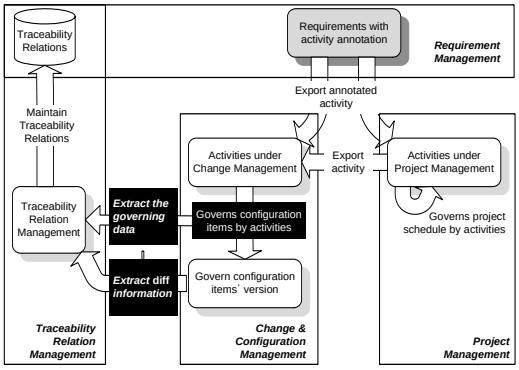


Figure 1. Framework for Automating TRM

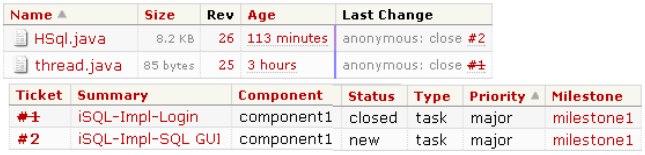
When requirements analysis is performed, each requirement is annotated with activities. Figure 2 shows the examples of annotated activities. Identifier `@Act` is used to distinguish the annotated activities from the requirements. The annotation follows the following syntax:

```
@Act.<development_pahse>(Activity_Name)
```

Seq	Requirement	Activity
1	Interactive SQL	@Act.NewCategory("iSQL")
2	LoginID should be at least 6 characters	@Act.Design("Login") @Act.Impl("Login") @Act.Test("Login")
3	Password should be masked by "*"	
4	SQL statement should be executed interactively in GUI	@Act.Design("SQL GUI") @Act.Impl("SQL GUI") @Act.Test("SQL GUI")

Figure 2. Requirements with Annotated Activity

The annotated activities are exported to either project management tools or CM tools. If activities are exported to project management tools, then the activities can be further refined to handle other constraints such as resources, budgets, and schedule. Then, the refined activities are exported to CM tools. After activities are created in a CM tool, activities are maintained by pre-defined workflow and associated with corresponding artifacts in SCM (provided that the CM tools and SCM tools are integrated properly). Figure 3 shows activities that are associated with corresponding artifacts using Trac [12] and SVN[13]. Trac and SVN are open source tools to manage activity and software configuration, respectively.



Name	Size	Rev	Age	Last Change
HSql.java	8.2 KB	26	113 minutes	anonymous: close #2
thread.java	85 bytes	25	3 hours	anonymous: close ~~#1~~

Ticket	Summary	Component	Status	Type	Priority	Milestone
~~#1~~	iSQL-Impl-Login	component1	closed	task	major	milestone1
#2	iSQL-Impl-SQL GUI	component1	new	task	major	milestone1

Figure 3. Example of CM/SCM Integration

After the needed information is created, the TRM tool retrieves the information to create and manage the relations. For instance, the HSql.java file in Figure 3 will be linked with requirements 2 and 3 in Figure 2. The following are key to this approach:

- ***Controlling the Granularity of a Traceability Relation***: Defining the proper granularity of a traceability relation is still a challenge because each software artifact is produced with different methodologies and represented in different forms. To address this issue, we can consider two techniques: 1) control the granularity of an activity, and 2) employ diff information. If an activity is defined in a fine-grained manner, it may need to change a whole file in the extreme case. Otherwise, more artifacts can be associated with the activities. However, this technique has some problems such as it requires much effort to define and manage activities and a file is still an atomic unit of the change. To handle this issue, we extract diff information between two different versions and associate the information with the traceability relation as a supplement. Figure 4 shows the result of Model Difference in IBM Rational TAU [11] as an example. The orange and pink parts are the differences between version 1 and 2.

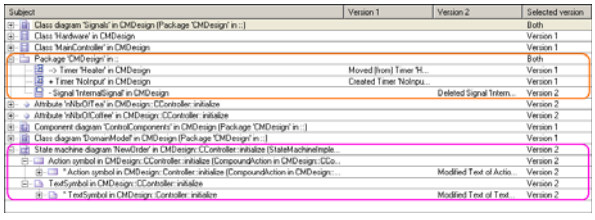


Figure 4. The Result of Model Diff in IBM Rational TAU

- ***Manage the Traceability Relation Evolution***: Changes in software artifacts may invalidate the existing traceability relations or require the creation of new relations. As CM/SCM integration assists with the evolution of the artifacts by activities, a traceability relation can represent the evolution by retrieving CM/SCM integration information.

3. Summary and Future Works

To automate TRM, requirements are annotated with development activities and then these activities are exported to CM or SCM tools. The exported activities function as a unit of software artifact change and are associated with the changed artifacts. Therefore, the traceability relation can be created and updated by retrieving the activities in each management tool and their related artifacts. The proposed approach herein offers several benefits, such as: 1) it can be easily applied to existing development processes with a small amount of investment for developing the TRM tool, 2) it can automate the management of traceability relations and maintain the relation, 3) it can create traceability relations incrementally as the project progresses. However, we need to further study 1) how the automation affects the accuracy of impact analysis and project estimation, 2) what is the best granularity for defining activities and managing the traceability relation, 3) how to measure the improvement of the process and accuracy of estimation from the automation.

ACKNOWLEDGEMENT

This work was supported in part by an NSF CAREER award (0643725).

REFERENCES

[1] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, Recovering traceability links between code and documentation. *IEEE Transactions on Software Engineering*, 28(10):970–983, 2002.

[2] O. Gotel and C. Finkelstein, An analysis of the requirements traceability problem. *International Conference on Requirements Engineering*, pp. 94-101, April 1994, Colorado Springs, CO.

[3] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Transactions on Software Engineering*, 32(1):4–19, 2006.

[4] A.D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Transactions on Software Engineering and Methodology*, 16, 4 (Sep. 2007), 13.

[5] T. Zimmermann, P. Weissgerber, S. Diehl, and A. Zeller. Mining version histories to guide software changes. *IEEE Transactions on Software Engineering*, 31(6):429–445, 2005.

[6] A. Zisman, G. Spanoudakis, E. Perez-Mi˜nana, and P. Kraus. Tracing software requirements artifacts. *International Conference on Software Engineering Research and Practice*, pp. 448–455June 23 - 26, 2003, Las Vegas, NV.

[7] http://office.microsoft.com/en-us/project/default.aspx

[8] http://www-01.ibm.com/software/awdtools/clearcase/

[9] http://www-01.ibm.com/software/awdtools/change/

[10] http://www-01.ibm.com/software/awdtools/clearquest/

[11] http://www-01.ibm.com/software/awdtools/tau/

[12] http://trac.edgewall.org/

[13] http://subversion.tigris.org/