RETRATOS: Requirement Traceability Tool Support

Gilberto Cysneiros Filho[1], Maria Lencastre[2], Adriana Rodrigues[2], Carla Schuenemann[3]

[1] Universidade Federal Rural de Pernambuco, Recife, Brazil
g.cysneiros@gmail.com
[2]Universidade de Pernambuco, Recife, Brazil
mlpm@ecomp.poli.br, adriana.rodrigues@hotmail.com
[3]Universidade Federal de Pernambuco, Recife, Brazil
carlotcha@gmail.com

Abstract. Software traceability is the ability to relate artefacts created during the development life cycle of software system. Traceability is essential in the software development process and it has been used to support several activities such as impact analysis, software maintenance and evolution, component reuse, verification and validation. Moreover, the importance of traceability in the software development process has been endorsed by several standards for quality management and process improvement such as ISO 9001:2000 and CMMI. Despite the importance of software quality, current support for traceability is inadequate. In this paper, we present a tool that tackle different aspects and issues of the traceability problem. In particular, the tool support a rule based approach to capture traceability relations between software models. The rules can be created to capture traceability relations of different types of software models.

Keywords: Software traceability, rule-based approach, traceability visualization.

1 Introduction

Software traceability has been defined as "the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases)" [1].

Traceability relations can guarantee and improve software quality and can help with several tasks such as the evolution of software systems, reuse of parts of the system, validation that a system meets its requirements, understanding of the rationale for certain design decisions, identification of common aspects of the system, and analysis of implications of changes in the system.

Traceability has been studied for many years and several approaches have been proposed to tackle its different aspects and issues. Pohl [2] states that a traceability approach should provide answers to the following questions:

- What traceability information should be captured?
- How traceability information should be captured?
- How traceability information should be stored?

Sherba adds in [3] that a traceability approach should also to answer the following question:

- How traceability relations are going to be viewed and queried?

Although several approaches have been proposed in the literature, in general they only address one aspect of the traceability problem. This makes more difficult to take advantage of the benefit to use a traceability tool in an industrial setting.

This paper presents a work in progress on a tool that address all the four questions above presented. In particular, the tool supports automatic generation of traceability relations, consistency and completeness checking in models created during the development life cycle of software systems. The tool is based on a rule-based approach that allows capturing traceability relations between different types of models (e.g. BPMN, UML, Java code) created during the development of software systems. Rules can be created to define what and how the traceability information should be captured. A visual editor to create rules and a visualization tool to support different forms of visualization is been developed.

2 Objectives

The goal to develop RETRATOS tool is to extend and tackle some weak points of the traceability approach of one the authors presented in [4,5,6]. The approach was developed to support (i) automatic generation of traceability relations between heterogeneous software models created during the development of multi-agent systems, and (ii) identification of missing elements in these various software models created during the development of multi-agent systems (completeness checking).

The figure 1 presents an overview of our framework. As shown in the figure, initially, the models of our concern represented in their native format are generated using proprietary tools (e.g. TAOME4E, Astah, or any diagram editor tool). These models are translated into XML format by using a Model Translator component based on XML Schemas proposed for the models, whenever the tools used to create the models do not generate them directly in XML.

The XML based models and rules are used as inputs to the Traceability Generator and Consistency Checker component to generate traceability relations between the models and to identify missing elements based on the rules. The engine also uses WordNet to support the identification of synonyms between the names of elements in the models. The WordNet is important component because in general naming conventions can change from high-level models (e.g. i*) to low-level representations (e.g. Java code).

The traceability relations and identified missing elements are represented in an XML document. The use of a separated document to represent the traceability relations and missing elements is important to preserve the original models, to allow the use of these models by other applications and tools, and to allow the generated relations to be used to support the identification of other traceability relations that depend on the existence of previously identified relations.

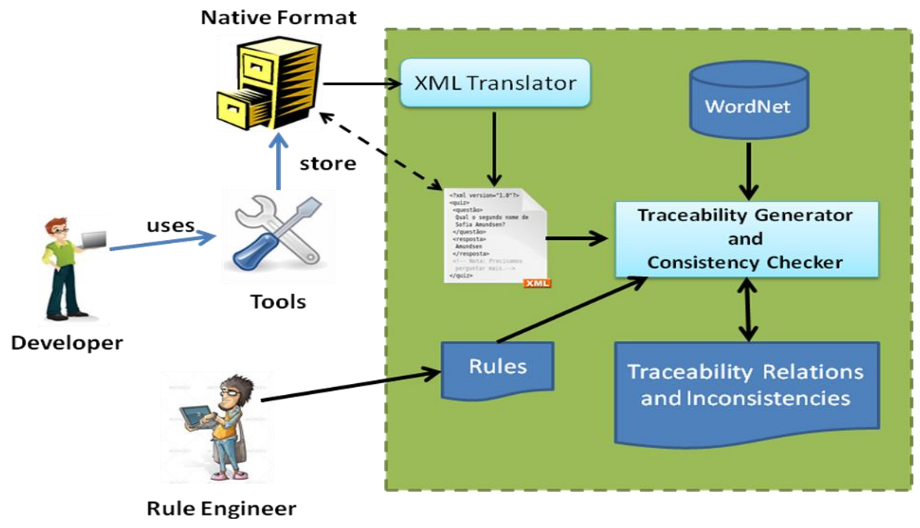

Native Format
XML Translator
WordNet
store
uses
Tools
Developer
Traceability Generator and Consistency Checker
XML
Rules
Traceability Relations and Inconsistencies
Rule Engineer

Figure 1- Cysneiros´s Approach

3 Our Approach

The Cysneiros´s approach only address the problem of identifying traceability relations. The figure 2 shows that two components are added to the original approach: a traceability visualization tool and a rule editor tool.

Our experience has shown that a large number of traceability relations can be generated for the various models. Therefore visualization support is fundamental to: i) allow the user to browse the traceability relations through various types of user interactions; ii) allow the user to add, remove, and modify properties of existing traceability relations and their related artefacts; iii) integrate with tools used to develop, test and maintain the system; iv) capture and maintain browsing history for traceability relations; v) support user querying and filtering of the traceability relations; vi) offer flexible and user customizable view of the traceability data; vii) provide tools to analyse and summarize the data on the traceability process and relations.

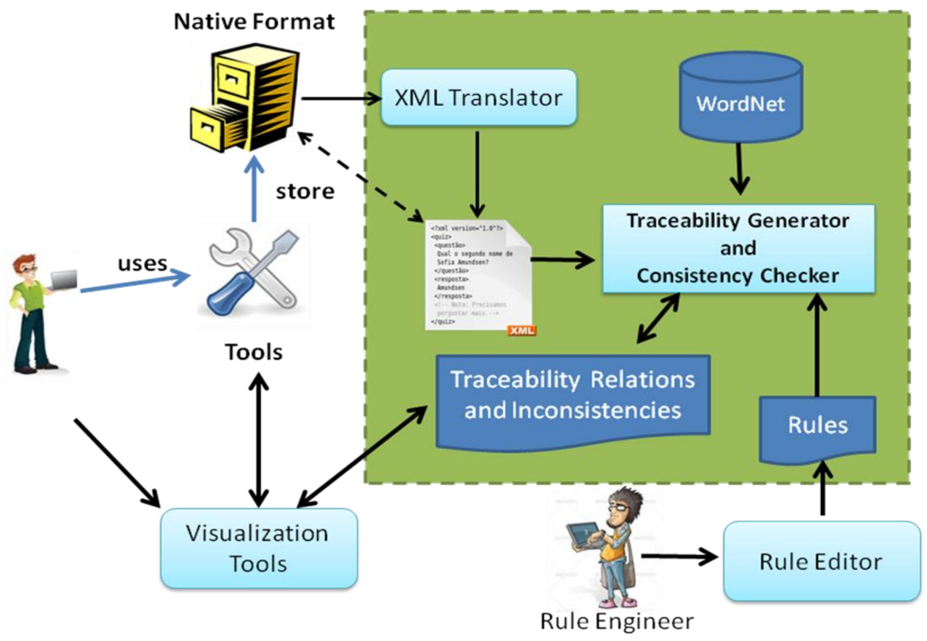


Figure 2 - RETRATOS overview

Currently, the approach support matrix, tree and Sunburst visualization techniques and generation of reports in HTML (see Figure 3).

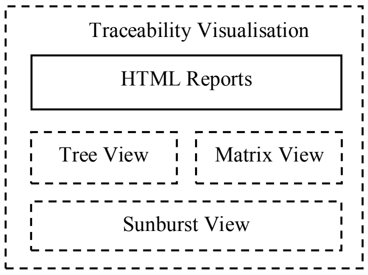
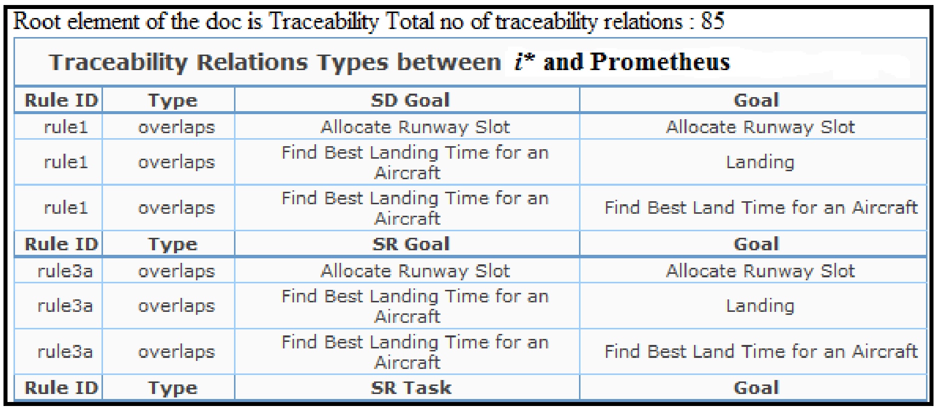


Figure 3 – Visualization Tools

HTML reports (see Figure 4) show the total number of traceability relations identified by the tool and to each relation created shows: (i) id of the rule used (e.g. rule1), (ii) type of the relation (e.g. overlaps), (iii) name of the elements (e.g. Allocate Runway Slot) and (iv) element´s type (e.g. SD goal). A more comprehensive and customizable reporting tool is been built.

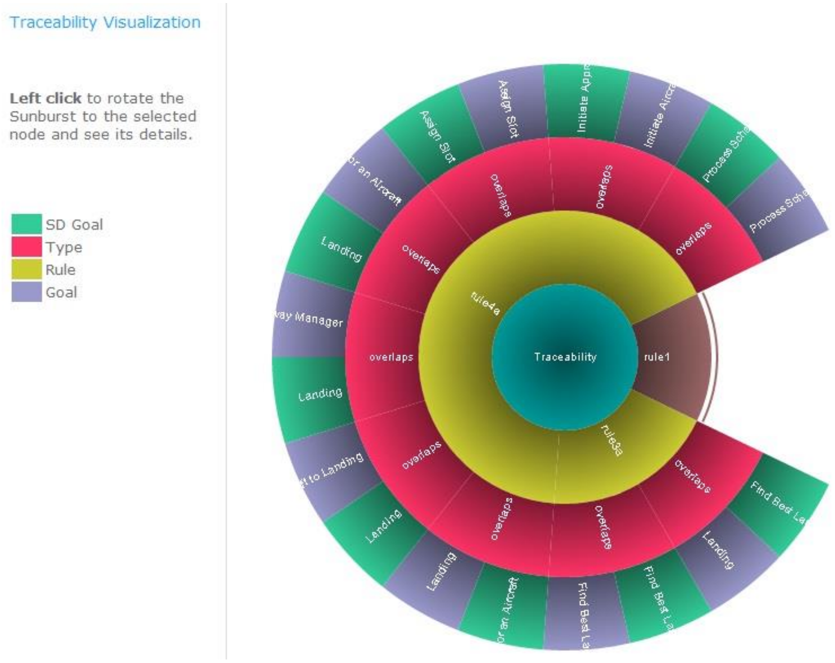

Root element of the doc is Traceability Total no of traceability relations : 85

Traceability Relations Types between *i and Prometheus**

Rule ID	Type	SD Goal	Goal
rule1	overlaps	Allocate Runway Slot	Allocate Runway Slot
rule1	overlaps	Find Best Landing Time for an Aircraft	Landing
rule1	overlaps	Find Best Landing Time for an Aircraft	Find Best Land Time for an Aircraft
Rule ID	**Type**	**SR Goal**	**Goal**
rule3a	overlaps	Allocate Runway Slot	Allocate Runway Slot
rule3a	overlaps	Find Best Landing Time for an Aircraft	Landing
rule3a	overlaps	Find Best Landing Time for an Aircraft	Find Best Land Time for an Aircraft
Rule ID	**Type**	**SR Task**	**Goal**

Figure 4 - HTML Report

Sunburst is a graph (see Figure 5), which nodes are arranged in a radial layout. Nodes are drawn on adjacent rings representing a tree structure. Each child of a node with depth n is represented in the ring n + 1 on the same radian space as its parent(s). Sunburst visualization performs better on large amounts of nodes than traditional tree view representation. Tree view representations grow rapidly in the vertical direction if many branches are expanded while using Sunburst the nodes are distributed uniformly in all directions.


Traceability Visualization

Left click to rotate the Sunburst to the selected node and see its details.

🟩	SD Goal
🟥	Type
🟨	Rule
🟪	Goal

Figure 5 – Sunburst visualization

The Cysneiros´s approach relies on the use of traceability and completeness checking rules specified in an extension of XQuery. The creation of these rules is not a straightforward activity and requires knowledge of XQuery. To address this problem we are developing a visual rule editor that allows to create rules using a drag and drop approach. The editor also provides a graphical interface that allows selecting only the rules that are applicable to a specific software project or domain.

4 Conclusion and Future works

In this paper, we described a traceability tool that extends a rule-based approach to identify automatically traceability links and missing information between artifacts created during the development of software systems. The traceability tool adds support to traceability visualization and visual creation of rules.

One of the challenges to build a traceability tool is to provide support to the vast number of methodologies, platforms, tools, and languages available to develop software. Currently, the tool supports models created to the development of multi agent systems when using i* framework, Prometheus methodology, and JACK code. Our goal is to use and evaluate the approach in the other programming paradigms such as web services and object oriented programming.

As future work, we also intend to extend the tool to other types of visualization techniques and evaluate these techniques in terms of usability.

References

1. Gotel O. and Finkelstein A.: An Analysis of the Requirements Traceability Problem. International Conference on Requirements Engineering. - Colorado, USA: IEEE Computer Society. (1994) 94-101
2. Pohl k. Process-Centered Requirements Engineering. John Wiley & Sons, Inc., New York, NY, USA (1996)
3. Sherba S. and Anderson K.: A Framework for Managing Traceability Relationships between Requirements and Architectures. Second International Workshop from Software Requirements to Architecture (2003)
4. Cysneiros G. and Zisman A.: Traceability and Completeness Checking for Agent-Oriented Systems. 23rd Annual ACM Symposium on Applied Computing (2008)
5. Cysneiros G. and Zisman A.: Traceability for Agent-Oriented Design Models and Code. TEFSE/GCT'07. - Lexington, KY, USA (2007)
6. Cysneiros G. and Zisman A.: Tracing Agent-Oriented Systmes. The Nineteenth International Conference on Software Engineering and Knowledge Engineering (2007)