

Latent Semantic Indexing as a Technique for Discovering Traceability Links

Hans-Peter Krüger and P.S. Kritzinger
Data Network Architectures Group
Department of Computer Science
University of Cape Town
Email: {hkruger, psk}@cs.uct.ac.za

Abstract — Software traceability of an initial stakeholder's requirement among their corresponding artifacts (i.e. from the plain stakeholders request, to the requirement, its specification, design and code representation) can help to save costs and raise product quality. Unfortunately existing traceability approaches require a great amount of human effort and intervention. In this work we are trying to improve the situation by applying an information retrieval method Latent Semantic Indexing (LSI) to discover traceability links among software artifacts automatically.

Index Terms – Latent Semantic Indexing, Information Retrieval, Software Traceability, Software Engineering

I.BACKGROUND

It's an illusion to think that the requirements gathering process can be completely finished before the design phase of a software project. Stakeholder's requirements can heavily change during the development process, new requirements are discovered and need to be added or old ones have to be removed (i.e. because of a lately detected lack of consistency).

Software traceability can be of great value in keeping track of the changing requirements, as reflected in the corresponding artifacts and their dependencies throughout the Software Development Lifecycle (SDLC). In general, software traceability can be defined as the ability to trace a requirement, in both a forward and backwards direction through the artifacts which constitute the different stages of a SDLC [1].

Although, high costs can arise due to overlooking a changed requirement in one or other artifact, such as the class diagram used for code generation, the influence on the overall product quality, software traceability has not gained as much attention as it deserves in the research world and industry. This problem seems to be directly connected to an inadequacy of tools which offer at its best, semi-manual software traceability. Establishing traceability links among the requirements and artifacts is primarily the duty of the designers and developers and requires continual effort.

Our belief is that the automated discovery of traceability links is an information retrieval (IR) problem. This implies that methods that are widely used for solving IR problems could also be applied to recover traceability links among artifacts in the SDLC.

II.LATENT SEMANTIC INDEXING

One of the most successful methods is Latent Semantic Indexing (LSI) [2]. LSI is an algebraic IR approach that is based on a vector space model [3] and allows for natural language text processing. It overcomes the fundamental problem of *synonymy* and *polysemy* that plagues many existing text search approaches that try to match words (terms) of queries with words of documents. There are usually many ways to express a given concept in natural language, so the literal terms in a query may not match those of a relevant document. LSI tries to reveal the underlying or latent semantics among documents that is partially obscured by variability in word choice.

III.APPROACH AND METHODOLOGY

Almost every stage during a Software Development Lifecycle (SDLC) produces artifacts (documents) as a result of an activity performed in the stage. Many artifacts like requirements, design or test specifications are generally natural language based documents with a lack of formalism and a high variety of word choice often caused through the influence of different authors. But even artifacts like UML diagrams or source code that have a higher degree of formalism are using natural language as a base.

LSI has been designed to solve the problem of *synonyms* and *polysems* in natural text documents and to perform better search results than other approaches. The broad goal in this work is therefore to investigate how reliable LSI can be in retrieving traceability links automatically among artifacts produced in the SDLC stages. The artifacts which are created along the way in the development of a software product include most, if not all, of the following.

1. A textual product description or vision document which describes the product features.
2. The requirements specification document derived for the above. Usually this will be in textual form as well.
3. Design documents and diagrams using, typically UML. Recently these documents are translated to XMI for transportability between various tools. In such a XMI file various keywords can be retained as text.
4. The code itself, where best practice require the programmer to use variable and object names in the documents he creates which again relate to the existing keywords.
5. During testing, certain acceptance requirements documents are usually defined in textual format.

The interesting question is to what extent an information retrieval approach like LSI is capable of recovering

traceability links when we take all the above mentioned SDLC artifacts into consideration and how already existing approaches like [4][5] can be improved.

IV. ANTICIPATED OUTCOMES

In order to demonstrate the appropriateness of LSI for recovering traceability links we shall develop demonstration software (LSITrace) with the following objectives in mind:

1. **The facility to import various SDLC artifacts such as mentioned above.**
2. **Indexing of artifacts in order to create a document corpus.**
3. **Performing search queries on that document corpus:**

Search queries can be distinguished in two different forms:

- **Search queries with existing artifacts:**
In this case the search query is already an artefact in the document corpus. This is in particular important when we want to know how artifacts are dependent upon one another. In terms of requirement artifacts that may answer the questions of what artifacts in the SDLC stages are presenting a certain requirement.
- **Search queries with non-existing artifacts:**
In this case the user will choose an existing artifact, change it or create a new one and afterwards perform a search with it. In terms of requirement artifacts that may answer the question of what artifacts in the SDLC stages are dependent upon a certain requirement change.

In order to restrict the number of results the user is also able to choose an artifact similarity measure as well as the degree of dimensionality reduction that is performed through LSI. This allows the user to control the statistical measures of Recall and Precision [3] that are well known in information retrieval.

4. **Visualization of traceability links:**

Once a search has been performed the similar/dependent artifacts will be visualized as a directed graph. Representing artifacts as vertices and the traceability links as edges is probably the most obvious and logical representation. The degree of similarity will be displayed along the edges of the graph. Through clicking on the vertices the user is able to see the content of the artifacts.

To determine the reliability of LSI as a traceability discovery approach we plan to apply it to at least two software projects in a case study. Creating traceability links manually for these case studies, comparing and statistically analyzing them to the results gained through our automatic approach LSITrace is likely to require major effort which may prevent us from achieving what we intend to do.

V. CONCLUSION

The results of the case studies presented in [4][5] have shown that LSI is apparently not able to replace manual software traceability completely. But it is in our belief that the research work in this thesis can reveal a few new ideas that could improve the reliability of the received traceability links. On the other side we don't expect LSI to be a standalone approach that replaces human effort and intervention in the traceability discovery process. But we do think that LSI can be very useful as an additional tool that can help to narrow down the number of trace choices significantly and to provide the software engineer with a preselection of artifacts that has to be considered when creating or updating traceability links manually.

REFERENCES

- [1] Gotel, O. C. Z. and Finkelstein A. C. W., An Analysis of the Requirements Traceability Problem, Technical Report TR-93-41, Department of Computing, Imperial College of Science Technology and Medicine, London, UK, August 1993.
- [2] Deerwester S., Dumais S.T., Furnas G. W, Landauer T.K., and Harshman R., Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science, 41, 1990
- [3] Salton G. and McGill, M., Introduction to Modern Information Retrieval, McGraw-Hill, 1983
- [4] A. Marcus and J. I. Malecic, Recovering Documentation to-Source-Code Traceability Links using Latent Semantic Indexing", Proc of 25th International Conference on Software Engineering, Portland, Oregon, USA, 2003
- [5] Andrea De Lucia, Fausto Fasano, Rocco Oliveto, Genoveffa Tortora, Enhancing an Artefact Management System with Traceability Recovery Features, Proceedings of the 20th IEEE International Conference on Software Maintenance, Chicago, USA, 2004

Biography: Hans-Peter Krüger holds a Diplom from the University of Applied Sciences Cologne (Germany) and is currently an MSc student in the Computer Science Department at the University of Cape Town and a member the Data Network Architectures (DNA) group. He enjoys programming, watching football and playing sports. Professor Kritzinger is research leader of the DNA group and is his supervisor even though he doesn't enjoy watching football.