

Position Paper: Enabling Traceability

Paul Arkley, Paul Mason, Steve Riddle
*School of Computing Science,
University of Newcastle upon Tyne
NE1 7RU
United Kingdom*

paul.arkley@ncl.ac.uk, p.a.j.mason@ncl.ac.uk, steve.riddle@ncl.ac.uk

Abstract

Research into traceability at the University of Newcastle upon Tyne has concentrated on a framework supporting all aspects of the lifecycle, as a vehicle for recording, analysing and tracing development and assessment artefacts. This framework is focussed on the recording of design rationale, over and above the “standard” inter-relationships between product artefacts, and has been developed in an aerospace systems engineering context. Despite technological advances in terms of databases, internet and processing power, many aspects of the “requirements traceability problem” are still current. We look at the reasons for this and suggest some research areas to address some of these aspects.

1 Introduction

In this highly-automated, information-at-your-fingertips society, why is traceability still so difficult? Specifically, why is it so difficult to answer questions about the impact of changes on the rest of the system? We contend that the reason has as much to do with process and human factors as with the issues of heterogeneous tools and distributed teams.

We have studied types of traceability relations and proposed a framework for traceability, centred around a design rationale capture system [1]. This framework consists of a number of *traceability structures* which classify the relationships between development and assessment artefacts according to a number of views including system architecture, argumentation and verification. Since our research is in the context of dependable, often safety-critical, systems we have also concentrated on application-specific views such as safety argumentation [2].

Provision of tool support has been studied in terms of database schemas and modifications to commonly-used requirements management tools [3]. This in

turn has led to consideration of data-exchange issues, since development teams are increasingly distributed and make use of heterogeneous tools.

The overriding focus of our framework is the recording of justifications for design decisions, as without a convincing explanation of *why* a particular decision has been made, it is impossible to gauge the likely impact of changing that decision. However, there are three problems which follow from this focus:

- **Increased Burden.** Overworked engineers do not take kindly to being told to record the reason for every decision they make
- **Accessibility.** It’s not sufficient to document the design justification: this justification must be accessible when you really need to know why a decision was made
- **Freshness.** Out-of-date justification is potentially more dangerous than no justification at all: thus changes to the system will require changes to the traceability information.

We begin by reporting the results of a recent industrial survey of “traceability practices”, drawing out some themes which remain at the heart of the traceability problem. Section 3 then outlines some specific research directions which could be fruitful in addressing these themes, and Section 4 draws conclusions.

2 A Survey of Traceability Practices

The interest in traceability is not new, as evidenced by Gotel’s landmark survey [4]. Our recent survey was conducted over a number of organisational units within a large systems engineering company: their concerns are thus rather wider than *software* traceability. We argue that any study of traceability cannot concern itself with software alone, as it is the re-

relationships with the wider system which often are the most complex.

The survey was more than a sample of current traceability tools and procedures. A major part of the survey took in the views of engineers on the traceability process itself and their opinion of its importance. This was achieved by a series of structured interviews, arranged through a local site contact in each case. Care was taken to provide a neutral, non-judgmental environment in which engineers could speak freely without feeling they were being subjected to a quality audit.

The findings are summarised below.

2.1 Tool issues

Though the projects surveyed were employing many of the leading traceability tools, some issues were raised with respect to their performance:

- **Data entry.** The need to transcribe documents into a tool-specific format is considered a burdensome task, compounded by the need for a detailed understanding of the tool's underlining database structures. As a result, knowledge of the tool is limited to a small number of "gurus", slowing down data input rates and leading to out-of-date traceability information.
- **Granularity.** Typically an "all or nothing" approach leads to users either being swamped by complex diagrams or presented with meaningless one line statements. As a result the traditional paper documentation is most commonly referred to instead.

2.2 Data organisation

All of the projects surveyed had a complex documentation set, with some degree of duplicated information. Keeping this up to date is a slow process, with the consequence that product development is often ahead of the documentation and many developers circumvented the "official" process by keeping their own annotated documents. The level of design detail contained within what should have been design documents often leads to the real requirements being obscured.

2.3 Education and Training

Keeping all engineers trained in the use of the latest tools is a high-cost option, and it is not surprising that in many teams only a small number of developers are trained to use the tools. As a result the "guru" situation alluded to earlier tends to arise.

In addition, short-term project needs lead to traceability being regarded as a bolt-on, burdensome extra task which will benefit the product support teams at the cost of the product development teams.

3 Research Directions

Much of the previous section is neither new nor surprising, though it is a little dispiriting. It is almost ten years since Gotel and Finkelstein identified the crux of the traceability problem as "the inability to locate and access the sources of requirements and [pre-Requirements Specification] work"; they contended that this was a major contributor to the other classic (and still current) issues: out-of-date Requirements Specifications; slow realisation of change; and poor reuse [4]. Their recommendations focussed on the need to be able to facilitate communication between those people responsible for specifying and refining requirements.

In the intervening years much has changed. In particular:

- Automated support for communication is much improved. There are much more powerful, integrated CASE tools such as RTM and DOORS, with pre-configured, extensible information models and a raft of analysis procedures to process the large amount of information held.
- Internet technologies provide the ability to cross-link information artefacts across heterogeneous databases, enabling product development teams to exchange data more easily thanks to standardisation initiatives such as STEP (Standard for the Exchange of Product Data) and technologies such as XML (eXtensible Mark-up Language).

Both of these are factors which might have been expected to aid communication, and to some extent they have. However, we have seen that in practice an even bigger information management problem has been created, and the information models provided by the CASE tools are so complex that in a large project, one engineer is in charge of the tool and all communication goes through him.

- Increasingly systems are being developed concurrently by a consortium of companies. Direct communication is harder and, despite the available technology, engineers do resort to communication via formal paper-based documentation, which effectively places a barrier on traceability. Though

some success has been achieved in the area of *distributed traceability* there are still some barriers to more widespread use.

- The increased pressure to procure software and systems off-the-shelf rather than developing them in-house. This leads to an even greater loss of control, as a level of indirection has been added to the traceability problem.

Communication is still most productive in small, focussed teams. However accessible “legacy” information is made, it is still much easier if the people responsible for specifying and refining requirements are still around. To take a recent high-profile example, the programme to modify Concorde after the Paris accident benefitted from the fact that many of the original engineers were still around, and in some cases could be brought out of retirement.

Suggested future research directions are thus aimed at mitigating the effects of these problems, by taking two approaches: the first aims to enable traceability by harnessing related research and emergent technologies; the second complementary approach looks at getting maximum benefit from the way engineers currently work, without forcing a new process on them. We outline these directions below.

1. **Design Rationale Patterns.** Patterns provide means of representing common elements of structures in a form amenable to reuse. They originated in the 1970s with work to systematically capture expertise in the architecture domain [5], although since then, the underlying principles have been successfully applied to the development of object-oriented software [6], fault tolerant architectures [7] and safety arguments [8]. We envisage applying the same principle to the representation of design rationale; i.e., capturing a library of “tried and trusted” assertions about a target system with respect to intent, assumptions, trade-offs, etc. Potentially, this can both alleviate the “increased burden” on engineers spoken of previously, as well as contributing to development of an organisational memory (and so to further reuse).
2. **Distributed Traceability.** The distributed traceability issue mentioned above has already been the subject of some related research in terms of internet technologies. We propose to extend our research into the use of tool-neutral data exchange and common product information reposi-

tories, in parallel with development to the emergent systems engineering data exchange standard AP-233 [9].

3. **Pro-active repository.** The problem of “freshness” is in part due to the requirement of many traceability tools for the engineer to query the dataset to determine if his area of interest has been affected by any new developments. We intend to look into ways of introducing mechanisms which would allow a tool to pro-actively alert the engineer to possible changes. Though this is not a new area of work, the challenge is to focus the presentation of this information to avoid a burdensome overload.
4. **Informal Procedures.** The survey found that a number of engineers employ an informal directory structure to record traceability information. These informal directory structures are akin to the traditional “kitchen drawer”: a place in every household where useful things are placed before they allocated a permanent storage place. Colleagues were aware that information relating to a particular decision might be found in a kitchen drawer. These directories were popular for obvious reasons: they were flexible, voluntary and easy to use. Any attempt to mandate such an approach would doubtless kill it off: however we would like to study it in order to replicate some of the benefits.

The discussion above considered improvements to the recording of design rationale for current projects. However, the procedure of capturing and structuring information may go awry if the right information is not recorded in the first place. For a project with potentially a 10-15 year life-time, we are unlikely to be able to predict at the start of the project what questions we might want to ask during maintenance or mid-life update phases. It is not a realistic option to record everything.

Much of the information which could be relevant - design justification, trade-off studies and so on - is actually recorded, but not in a structured, explicitly traceable way. The kitchen drawer idea is an example of this, but there are also official documents which are produced and then filed away. Increasingly the piles of documents and spreadsheets which do record this information are stored electronically: and while they are not explicit traceable, there are implicit links between parts of the documents. The problem is to

make them explicit, and to achieve this it is necessary to infer both semantic and structural information.

Technologies such as those employed by Google have made great progress in aiding internet information retrieval. While much of the internet is what could be called “semi-structured”, through the use of standard hypertext markup, a significant amount of material is provided in the form of Adobe PDF (Portable Document Format) and Word documents: one of the strengths of the search engines is their ability to parse and search through these documents as well.

The recording of semantic information is partially achieved through the use of keywords and meta-tags: something more sophisticated is needed if semantic links are to be inferred with any credibility. One candidate technique from the field of computational linguistics is *lexical chaining*: if two documents have a common set of semantically related words contained within them, an inference can be made about the relationship between the documents, and an indexed relationship at the level of words can be produced. Similar approaches have been proposed for program comprehension [10], we are suggesting an application to a wider area, of *product* comprehension.

Several problems need to be overcome: semantic similarity is dependent upon use of the “right” thesaurus - in a particular domain a very specific thesaurus is needed; secondly, there may not be enough precision in the original unstructured documents for a useful result. A more serious problem is that the resulting index of related words may be huge - similar to the standard internet search problem - so a relevance measure could be used to imply a stronger link (for example if two documents have a common author, or related stakeholders).

4 Conclusions and Future Work

In the above discussion we have tried to stress the importance, for a worthwhile, beneficial traceability process, of supporting the way software - and systems - engineers work, rather than imposing new processes on them. Enabling technologies for addressing many of the classic traceability problem exist: in particular for distributed traceability and recovery of semantic information these will include the GRID. This distributed computing infrastructure is particularly suited for storage, analysis and retrieval of large, concurrently-accessed datasets with associated meta-data in real time.

Current and future work is intended to address

these areas. EPSRC project DOTS (Diversity with Off-The-Shelf components) looks at making best use of the (limited) information available about the pedigree of off-the-shelf components, in order to support decisions on how to use them (if at all) in a particular context [11].

Under the umbrella of a long-term research programme, we are building on the results of the survey reported in Section 2. Initial work here consists of developing prototype tool support, complementing the existing toolset, to address research areas 2 and 3 from Section 3. Using a STEP-based, tool-neutral representation of project information we will provide a “pragmatic” demonstrator, which it is hoped will increase the buy-in for traceability and show its relevance to the engineer.

Finally, a “spin-off” project proposal is being developed to address research areas 1 and 3, and to tackle the observed problem of motivating the recording of traceability information (in particular, design rationale). This will focus on the foundation of a library of past project information, expressed as patterns, which can be viewed from different perspectives in order to select the most appropriate design or code for a new project. In selecting from this library a developer will also indicate *why* a particular choice was made, so providing a simple-minded, but still useful, rationale to be built for the project. It is proposed the template format described by Gamma et al. [12] be used as a basis for documenting pattern descriptions. That is, a series of statements organised under predefined headings (motivation, applicability, example applications, etc.) outlining the underlying purpose and guidelines on usage of a particular pattern.

These research programmes proceed pragmatically by maintaining close contact with industrial partners to evaluate the research results. As we have stressed in this paper, technology is only part of the solution for enabling traceability - motivation, in the form of demonstrable benefits to the people charged with recording this information, is also needed. We have outlined some research areas which aim to make the burden of recording lighter, and at the same time provide an incentive for capturing useful traceability information.

Acknowledgements

Work by Paul Mason reported here was conducted while supported by an EPSRC CASE award sponsored by BAE SYSTEMS. The authors would also like to acknowledge the support of the BAE SYSTEMS

References

- [1] M Klein. Capturing design rationale in concurrent engineering teams. *IEEE Computer*, 26(1):39–47, January 1993.
- [2] S Pearson, S Riddle, and A Saeed. Traceability for the development and assessment of safe avionic systems. In *Proc. 8th Int. Symposium International Council on Systems Engineering (INCOSE '98)*, pages 445–452, Vancouver BC, Canada, July 1998.
- [3] S Riddle and A Saeed. Tool support for implementation and analysis of traceability structures. In *Proc. 9th Int. Symposium International Council on Systems Engineering (INCOSE '99)*, Brighton, UK, June 1999.
- [4] OCZ Gotel and ACW Finkelstein. An analysis of the requirements traceability problem. In *Proc. 1st IEEE Int. Conf. on Requirements Engineering*, pages 94–101, Colorado Springs, April 1994.
- [5] C Alexander. *The Timeless Way of Building*. OU Press, 1979.
- [6] P Coad. Object-oriented patterns. *Communications of the ACM*, 35(9):153–159, 1993.
- [7] RW Born. Patterns for safety-critical systems. Master's thesis, Department of Computer Science, University of York, 1998.
- [8] TP Kelly and JA McDermid. Safety case construction and reuse using patterns. In P Daniel, editor, *SafeComp '97: Proc 16th Int. Conf on Computer Safety, Reliability and Security*, pages 55–69, York, UK, 7-10 September 1997.
- [9] JFE Johnson. The SEDRES project: producing a data exchange standard supporting integrated systems engineering. In *Proc. 8th Int. Symposium International Council on Systems Engineering (INCOSE '98)*, 1998.
- [10] J I Maletic and A Marcus. Supporting program comprehension using semantic and structural information. In *Proc. 23rd International Conference on Software Engineering (ICSE 2001)*, pages 103–112, Toronto, Canada, 2001.
- [11] P Popov, S Riddle, A Romanovsky, and L Strigini. On systematic design of protectors for employing OTS items. In *Proc. of the 27th Euromicro conference.*, pages 22–29, Warsaw, Poland, 2001. IEEE CS.
- [12] E Gamma, R Helm, R Johnson, and J Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.