An Analysis of the Requirements Traceability Problem

Orlena C. Z. Gotel & Anthony C. W. Finkelstein

Imperial College of Science, Technology & Medicine
Department of Computing, 180 Queen's Gate
London SW7 2BZ (oczg; acwf@doc.ic.ac.uk)

Abstract

In this paper[1], we investigate and discuss the underlying nature of the requirements traceability problem. Our work is based on empirical studies, involving over 100 practitioners, and an evaluation of current support. We introduce the distinction between pre-requirements specification (pre-RS) traceability and post-requirements specification (post-RS) traceability, to demonstrate why an all-encompassing solution to the problem is unlikely, and to provide a framework through which to understand its multifaceted nature. We report how the majority of the problems attributed to poor requirements traceability are due to inadequate pre-RS traceability and show the fundamental need for improvements here. In the remainder of the paper, we present an analysis of the main barriers confronting such improvements in practice, identify relevant areas in which advances have been (or can be) made, and make recommendations for research.

[**Keywords:** requirements traceability, pre-requirements specification traceability, post-requirements specification traceability, requirements engineering practice, requirements traceability tools.]

1:Introduction

Requirements traceability (RT) is recognised as a concern in an increasing number of standards and guidelines for requirements engineering (RE) [12]. This concern is reflected by the various systems that have been developed and a growing research interest in the area [25]. Despite many advances, RT remains a widely reported problem area by industry. We attribute this to inadequate problem analysis.

Definitions of "requirements traceability" are discussed in detail later, though we provide the following for orientation:

- **Requirements traceability** refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction.
- **Pre-RS traceability** refers to those aspects of a requirement's life prior to inclusion in the RS.
- **Post-RS traceability** refers to those aspects of a requirement's life that result from inclusion in the RS.

In this paper, we analyse the RT problem in detail. We describe our empirical investigations in Section 2, review the current support in Section 3, examine the underlying causes of the problem in Section 4, and present a framework for addressing these in Section 5. Section 6 lists the main problems facing pre-RS traceability improvements, Section 7 identifies how some of these can be tackled, and Section 8 discusses that aspect of the composite problem which is the subject of our on-going research agenda at Imperial College.

[1] [23] is a longer version of this paper. The empirical work and surveys we refer to are fully documented as a BT/Oxford University research report.

2:Research method

Numerous data gathering techniques were used to define and analyse the RT problem. The empirical exercises took place over 1 year and involved more than 100 practitioners. Their work areas covered all aspects of development, maintenance and management; their experience ranged from 9 months to 30 years; and the projects they were involved with varied in number, type, and size. A detailed specification for RT support was produced in parallel. Here, introspection helped to identify requirements to support both this activity and its traceability.

2.1:Literature & tool reviews

The literature was surveyed, and over 100 commercial tools and research products were reviewed, to gather viewpoints regarding: what RT is; why it is needed; what problems it involves; and to locate relevant research and development.

2.2:Focus groups

5 semi-structured sessions were conducted. These involved 37 practitioners spread across 5 sites of a U.K. company. Each session lasted 1 hour, was audio taped, and transcribed. The data were used to: consolidate the above; discover how RT problems are overcome (if at all); get suggestions for improvements; and to inform questionnaire design.

2.3:Questionnaires & follow up interviews

A 2-stage questionnaire was used. Stage 1 was designed to rapidly gather data from a wide population of practitioners and to target those from whom more detail could be gathered. 80 were distributed and 69% returned. Stage 2 was tailored to the primary working areas and experiences of individual practitioners, using a reusable pool of questions. 39 were distributed and 85% returned. These provided a deeper understanding of the problems and preliminary requirements to address them. 2 informal interview sessions were subsequently conducted with the respondents. Each lasted 1.5 hours and were used to corroborate their answers, appraise their validity, extract additional information, and to check preliminary analysis.

2.4:Observation & participation

Data were also gathered following the observation of, and some participation in, a variety of RE exercises. For instance, Rapid Application Development workshops were observed,

where comprehensive notes were taken, and any informal documents produced were collected. Our analysis compared such artifacts with the workshop's end products.

3:Current support for requirements traceability

It has been noted that most tools do not cover RT [46], and that few support the RT requirements enforced by DOD STD-2167A [58]. Our survey further indicates that those which do, employ basically identical techniques. They differ mainly in cosmetics, and in the time, effort, and manual intervention they require to achieve RT. Both the type and extent of support provided depends on the underlying assumptions they embed about RT and the particular problems they focus on. However, they often suffer problems due to poor integration and inflexibility. These shortcomings are reflected by the preferred use of general-purpose tools in practice [38].

3.1:Basic techniques

Numerous techniques have been used for providing RT, including: cross referencing schemes [16]; keyphrase dependencies [28]; templates [27]; RT matrices [9]; matrix sequences [4]; hypertext [32]; integration documents [36]; assumption-based truth maintenance networks [53]; and constraint networks [2]. These differ in the quantity and diversity of information they can trace between, in the number of interconnections they can control between information, and in the extent to which they can maintain RT when faced with on-going changes to requirements.

Additionally, some form of RT can result from using certain languages, models, and methods for development. This is particularly exemplified by: the Requirements Statement Language [10]; process entity-relationship models [24]; the Planning and Design Methodology [42]; formal methods [8]; and Quality Function Deployment [59]. The quality of the resulting RT, however, depends on the rigid adherence to pre-specified procedures and notations for development.

3.2:Automated support

Many commercial tools and research products support RT, primarily because they embody manual or automated forms of the above techniques. We highlight some representative examples below and provide further details in Table 1.

General-purpose tools include: hypertext editors; word processors; spreadsheets; database systems; etc. They can be hand-configured to allow previously manual and paper-based RT tasks to be carried out on-line. This generally involves defining cross references and specifying their update criteria.

Special-purpose tools support dedicated activities related to RE and some achieve restricted RT. For example: the KJ-editor provides traceability between ideas and requirements [56]; and the T tool provides traceability between requirements and test cases [54]. Support is implicit in tool use, by automation of any mundane tasks needed to provide RT, and guidance is limited.

Workbenches contain a collection of the above to support coherent sets of activities. Less restricted RT can be achieved, but the quality depends on the focal workbench activity. Those in which RT and RT management are focal (such as the Automated Requirements Traceability System [21] and the Requirements Traceability and Management System [41]), are what we refer to as RT workbenches. They are typically centred around a database management system, and have tools to document, parse, organise, edit, interlink, change, and manage requirements. Other upper-CASE workbenches which assist RE activities frequently provide some support. This is either from explicit RT components (e.g., a Coupling Module in AGE [34]), or from having carried out other activities using its tools (e.g., the Requirements Apprentice [51]). Those workbenches which accept requirements documents, from which to drive the design and implementation, commonly provide coarse-grained RT between requirements and their realisation.

Environments, which integrate tools for all aspects of development, can enable RT throughout a project's life. The basis for integration defines how RT is established: a common language (e.g., Input/Output Requirements Language in Technology for the Automated Generation of Systems [54]); a common structure (e.g., relations of an Entity-Relation-Attribute Model in Genesis [48]); a common method (e.g., Information Engineering Method in the Information Engineering Facility [57]); or (where the tools are combined to support multiplicity) a specialised RT tool or repository structure (e.g., Teamwork/RqT [5]). Those which incorporate third-party tools use powerful repositories and database management systems to relate their products (e.g., the Digital CASE Environment [54]).

4:Why there is still a traceability problem

To date, techniques have been thrown at the RT problem without any thorough investigation of what this problem is. Despite a growth in specialised tools, and inflated claims of RT functionality from tool vendors, their use is not as widespread in practice as the importance of RT would suggest. RT problems even remain cited where they are used. Following investigations with practitioners, we have found that the RT problem is not perceived to be uniform, and attribute its persistence to diverse definitions and a number of fundamental conflicts.

4.1:Lack of common definition

Definitions of "requirements traceability", either by practitioners or in the literature, were found to be either:

- **Purpose-driven** (defined in terms of what it should do): *"...the ability to adhere to the business position, project scope and key requirements that have been signed off"* [Focus group practitioner].
- **Solution-driven** (defined in terms of how it should do it): *"...the ability of tracing from one entity to another based on given semantic relations"* [47].
- **Information-driven** (emphasising traceable information): *"...the ability to link between functions, data, requirements and any text in the statement of requirements that refers to them"* [Focus group practitioner].
- **Direction-driven** (emphasising traceability direction): *"...the ability to follow a specific item at input of a phase of the software lifecycle to a specific item at the output of that phase"* [15].

Each definition differs in emphasis and delimits scope. No single one covers all concerns. This has implications for the

RT	**(A) General-purpose tools**	**(B) Special-purpose tools**	**(C) Workbenches**	**(D) Environments and beyond**
(1) Priority given to RT	Any general-purpose tool can potentially be configured for RT purposes, though RT is not a concern of the basic tool.	Those that support RE activities (e.g., analysis techniques), often provide some form of RT as a by-product of use, but RT is not focal.	Priority depends on the focal set of activities. Where these are RT and management (in RT workbenches), RT is the main concern. Where other RE activities are focal, RT is a side concern.	Typically a side concern, though the extent of this depends on whether or not there are dedicated tools for RT contained in the composite environment.
(2) Support provided for RT	No explicit support is provided. RT must be hand-crafted and the resulting support provided depends on the initial effort expended in doing this. The focus can easily become configuring the tool to enable RT rather than ensuring RT itself.	Support is implicit in the framework provided for carrying out the main activity of the tool. Mundane and repetitive tasks, which are usually necessary to provide basic RT, are typically automated as a consequence of proper use of the tool.	In RT workbenches, support is explicit (else as B). No real analytical ability is provided, but they offer: (i) Guidance - through adherence to RE approach (typically top-down decomposition), types of information to collect, and link types to establish. (ii) Assistance - by parsing textual documents to tag requirements, establishing (syntactic) links between them, and through a repository which manages any bookkeeping and rudimentary checking.	Provided as a by-product of coordinated tool use and adherence to RE philosophy. Extent of support depends on the internal integration strategy and/or repository structure. More guidance and assistance if it has dedicated RT tools, or if it is a main concern. RT maintenance is supported if the repository can manage quantities of information and reconfigure after change.
(3) Requirements-related information that can be made traceable	Ability to trace any information which can be input to the tool (be this textual, graphical, etc.), so potentially able to trace all requirements-related information if sufficient effort and foresight are exercised.	Predefines the amount and type of information that can be input and made traceable. This is typically restricted to that information necessary to carry out the activity the tool supports. Only a limited scope of requirements-related information can be traced.	Potential to trace a diversity of requirements-related information. RT workbenches often impose arbitrary limits on the amount and type of information. They trace how an RS was produced, but usually only its derivation from a textual baseline, not its exploratory development, refinement, and context of production. Additional information (e.g., informal notes) can often be recorded, but is of limited use for RT.	Potential to trace information related to requirements in all project phases. Tendency to focus on information derived from requirements in the RS in later phases, so less emphasis on production-related information about individual requirements. Often support the RT of versions, variants, and user-defined items.
(4) Tasks and job roles that RT can assist	Offer complete tailorability. RT can be provided to support any task and job role, though it is problematic to meet different needs simultaneously without any RT infrastructure in place.	RT is provided to assist the activity the tool supports, so the role of the tool user is predefined. Their task-specific frameworks constrain the domain of working, making them difficult to use for other purposes.	Support for a breadth of activities within the concern of the tool's domain (e.g., able to assist requirements checking, etc.). Supports specific jobs, but often configurable to support tasks for other project phases. RT workbenches tend to support managerial activities rather than the activities of RS producers.	RT can assist lifecycle-wide tasks and roles (e.g., those related to maintenance and management, such as impact analysis and progress reporting, etc.). More support for activities related to the use of requirements rather than production and refinement.
(5) Longevity of RT support	Configured for immediate needs. RT can degrade with quantities of information and time, as not usually integrated with lifecycle-wide tools, and poor at handling changes and evolution.	Provide RT at a snapshot in time to support a specific activity, so neglect requirements for on-going management. Longevity of support depends on both horizontal and vertical integration with other tools.	RT is provided for the duration of the activities supported. Predominantly forwards-engineering tools, so RT can deteriorate with progression to later phases, as it can be difficult to reflect any work here and account for iteration. Longevity of support depends on vertical integration with other tools.	Can provide RT for a project's life, though tends to start from a static baseline. RT tightness and granularity depends on the underlying repository and degree of internal integration. RT can deteriorate due to iteration problems and poor feedback.
(6) Support for the traceability of group activities	Promote individualistic working, as provide no common or consistent framework for RT, so encourage immediate and ad hoc solutions. Typically used, by a single user, to record activities after they have happened.	Most support individualistic working. Those which directly support group activities (like the brainstorming of requirements amongst stakeholders), increasingly tend to make both the process and its end results traceable.	RT workbenches tend to be used as after-the-event documentation tools by single users, as they can be difficult to adapt to working practices. Concurrent work is often difficult to coordinate, so the richness of information can be lost. Participative work is actively supported in some (generally not RT workbenches), but traceability of this work varies.	Multiple users are supported through shareable repositories and techniques to assist activity coordination and integration (e.g., workspaces). Often depends on an agreed RS and strict project partitioning, so RT can deteriorate when requirements are unstable and overall control is lacking.
(7) Main strengths	(i) Flexibility to provide customised RT to suit individual project and organisational needs. (ii) Often sufficient for the RT of small and short term projects.	(i) Can provide tight RT for the immediate needs of particular requirements-related activities. (ii) Those supporting group activity, often provide traceability of it.	(i) RT workbenches provide good RT from and back to information which is initially input to the tool, and through a breadth of related activities (i.e., fine-grained horizontal RT within requirements phases). (ii) Added value (e.g., RT checks, visibility, etc.).	(i) Ability to provide on-going RT (i.e., depth of coverage or vertical RT). (ii) Open environments (and meta-CASE), provide more flexibility in the choice of RE approach and in the RT of this.
(8) Main weaknesses	(i) Requires much work to initially configure, can involve mundane and repetitive activities for use, and often provides little more than an electronic version of paper-based RT. (ii) Poor control and integration, so no guarantee as to the usefulness, usability, and longevity of the RT provided.	(i) Only provides restricted forms of RT between limited types and amounts of requirements-related information, so has limited life and use. (ii) Typically suffer from a limited potential for integration and poor information management, as mainly stand-alone, preventing fuller and longer RT support.	(i) RT workbenches attempt to be holistic, but none support all activities. Typically enforce: a top-down approach; classification schemes; and pre-empt a relatively static baseline (without support for its development). As RT depends on correct use, the main concern can be RT rather than RS production. (ii) RT workbenches integrate poorly, so difficult to support the RT of early problem definition work, or to provide on-going RT with later changes (much manual intervention can be required to do so).	(i) RT is typically coarse-grained and dependent on step-wise development. As the tightness of RT varies, iteration and later requirements changes can prevent on-going RT (due to poor backwards RT, which rarely accounts for the occurrence of manual intervention or work-arounds). (ii) Increasing flexibility (with those tools open to external integration), is typically counterbalanced by poorer RT.

Table 1: Tool support for requirements traceability

development and use of tools to support RT: how can RT be coherently and consistently provided if each individual has his or her own understanding as to what RT is?

4.2:Conflicting underlying problems

Each practitioner also had his or her own understanding as to the main cause of the RT problem. This finding is reflected in the literature, where it has been attributed to: coarse granularity of traceable entities [47]; immature integration technology [3]; hidden information [52]; and project longevity [42]. The problems that improved RT were expected to address were just as diverse, a finding also reflected in the literature: to support RS evolvability [30]; to enable safety analysis, audits, and change control [24]; to understand systems from multiple viewpoints [14]; and to permit flexible process modelling [20].

These findings demonstrate that: (a) the phrase "RT problem" is commonly used to umbrella many problems; and that (b) RT improvements are expected to yield the solution to further (and even ambitious or conflicting) problems. Complicating this was the observation that, underlying every situation in which RT is required in practice, different user, project, task, and informational requirements come into play. These cumulatively influence the problems experienced, which has further implications for any potential support: how can RT account for all these problems simultaneously?

5:A framework for addressing the problem

To provide a framework in which to locate and address the fundamental cause of RT problems, we first need to establish some shared and working definitions.

5.1:Defining requirements traceability

The definition most commonly cited in the literature is:

- *"A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future*

development or enhancement documentation" (ANSI/IEEE Standard 830-1984) [26].

This definition specifically recommends *backward traceability* to all previous documents and *forward traceability* to all spawned documents. An alternative definition, derived from the word "trace" in the Oxford English Dictionary, is:

- The ability to *"delineate"* and *"mark out" "perceptible signs of what has existed or happened"* in the lifetime of a requirement to enable one to *"pursue one's way along"* this record.

Together, these suggest the following definition for RT:

- *"**Requirements traceability** refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)."*

5.2:Pre-RS & post-RS traceability

Our investigations further suggest that RT is of 2 basic types:

- *"**Pre-RS traceability**, which is concerned with those aspects of a requirement's life prior to its inclusion in the RS (requirement production)."*
- *"**Post-RS traceability**, which is concerned with those aspects of a requirement's life that result from its inclusion in the RS (requirement deployment)."*

Figure 1 shows the typical setting of RT to illustrate these definitions. Note how requirements knowledge is distributed and merged in successive representations; note also the added complication of iteration and change propagation.

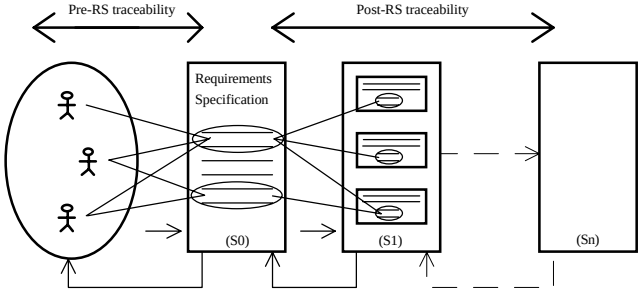

Figure 1: Two basic types of requirements traceability

Forwards and backwards RT are clearly essential. However, we emphasise the pre-RS and post-RS separation, because RT problems in practice were found to centre around a current lack of distinction here. Although both these types of RT are needed, it is crucial to understand their subtle differences, as each type imposes its own distinct requirements on potential support.

The main differences involve the information they deal with and the problems they can assist. Post-RS traceability depends on the ability to trace requirements from, and back to, a baseline (the RS), through a succession of artifacts in which they are distributed. Changes to the baseline need to be re-propagated through this chain. Pre-RS traceability depends on the ability to trace requirements from, and back to, their originating statement(s), through the process of requirements production and refinement, in which statements from diverse sources are eventually integrated into a single requirement in the RS. Changes in the process need to be re-worked into the RS. Changes to the RS need to be carried out with reference to this process, so they can be instigated and propagated from their source. This requires visibility of the subtle interrelationships that exist between requirements early on.

5.3:Support for pre-RS & post-RS traceability

Existing support mainly provides post-RS traceability. Any problems here are an artifact of informal development methods. These can be eliminated by formal development settings, which automatically transform an RS into an executable, and replay transformations following change [18]. In contrast, the issues that pre-RS traceability are to deal with are neither well understood nor fully supported. Post-RS traceability support is not suitable. This generally treats an RS as a *black-box*, with little to show that the requirements are in fact the end product of a complex and on-going process. Rigid commitment to categories for recording information also make it difficult to represent this process and to account for the dynamic nature of the sources and environment from which requirements are drawn. It has been argued that pre-RS traceability problems will remain, irrespective of formal treatment, as this aspect of a requirement's life is inherently paradigm-independent [18].

5.4:The need for improved pre-RS traceability

Only recently have these issues been acknowledged [17]. Our empirical findings intensify this concern: most of the problems attributed to poor RT were found to be due to the lack of (or inadequate) pre-RS traceability. Practitioners require techniques to record and trace information related to RS production and revision. Pre-RS traceability was also required to:

- Yield improvements in quality, as previously closed issues (even decisions about how to conduct the RE exercise itself), could be made explicit, possible to re-open, and possible to re-work (so assisting auditing [6], repeatability [29], etc.).
- Provide more economic leverage, as to use and maintain an RS in practice, it is often necessary to reconstruct an understanding of how it was produced (to compensate for invisibility [11]), which is currently error-prone and costly.

6:Problems confronting pre-RS traceability

Having identified insufficient pre-RS traceability as the main contributor to continuing RT problems, and shown how it is likely to be the only contributor in formal development settings, our investigations were re-focused to determine: what improvements in pre-RS traceability would involve; and how these could be realised. These indicated that the main barrier is due to an *establish and end-use conflict*. By this, we mean that the 2 main parties involved (i.e., those in a position to make it possible and those who require it to assist their work), have conflicting problems and needs (as shown in Figures 2 and 3).

6.1:Problems faced by the providers

- Perceived as an optional extra (and of low priority), so the allocation of time, staff, and resources is often insufficient.
- No allocation and management of the different roles that practitioners need to assume to: obtain and document the required information; organise it; and maintain it.

- Imbalance between the work involved and benefits gained.
- Individual efforts are ad hoc and localised, whereas a combined and full-time responsibility by all is really needed.
- No agreement on the end-user requirements, resulting in a tendency to focus only on their immediate and visible needs.
- Concern for pre-RS traceability lessens, and concern for post-RS traceability increases, after the RS has been formally signed off. Concern must continue, but this is problematic as the activities are unpredictable, change cultures are immature, and it depends upon RT being present to do so.
- Information (e.g., tacit knowledge), cannot always be obtained, and the quality of that which is varies. Deliverable-driven cultures can discourage gathering certain information.
- The documentation of required information is no guarantee of its traceability. That which is structured, so it is traceable in many ways, provides no guarantee it will be up to date.
- Poor feedback regarding best practice, and little dedicated support (be this clerical, procedural, or computer support), perpetuates the same problems and restricts advances.

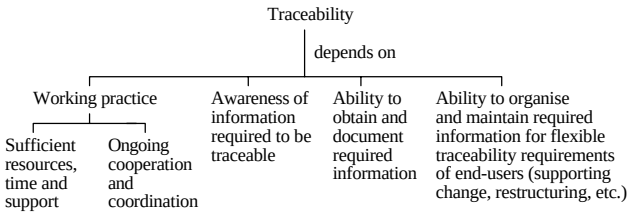

Figure 2: Deconstructing the RT problem for provision

6.2:Problems imposed by the end-users

- A stereotypical end-user cannot be predefined. Their requirements will differ and often be inconsistent.
- The quantity, heterogeneity, and depth of detail of the potential information required, precludes predefinition.
- Inability to predefine how any access to information, and its subsequent presentation, will be required.
- Reliance on personal contact, as there is always something that is out of date, undocumented, inaccessible, or unusable.
- Each end-use context exhibits unique requirements, so problems will exist if end-users do not have the ability to filter and access the information pertaining to RS production that they require under different circumstances.

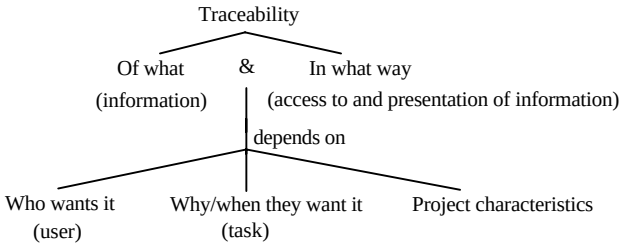

Figure 3: Deconstructing the RT problem for end-use

6.3:Addressing both parties

The challenge lies in satisfying both parties. For end-users, pre-RS traceability must be sensitive to contextual needs, but they cannot predefine their anticipated requirements for it. The providers must identify and document relevant information, in a (re)usable form (either as a by-product of other work or through more explicit support), but they cannot forsee and address all possible needs. Problems intensify when the same individuals assume both positions. The social nature of the activities involved suggests that technology alone will not provide a complete solution for pre-RS traceability.

7:Solutions to pre-RS traceability problems

An RS was produced to specify what is required to provide and make use of pre-RS traceability. The complexity of these requirements indicate that it would be premature to offer a comprehensive solution. It is a compound problem in need of improvements in many diverse areas. Here, we focus on those basic requirements for which some solutions already exist, and make recommendations for additional research.

7.1:Increasing awareness of information

Studies have revealed what project information is required by those involved in the different phases of development [35]. However, our investigations show that it is not possible to generalise, as both the amount and type required will remain subject to dispute. This issue is generally tackled by pre-specifying the types and structure of information required to assist focused activities, like the gIBIS argumentation scheme for design deliberation [7], but such schemes do not consider the wider informational requirements of all potential RE activities. RT models (as described in [50]), specifically aim to increase awareness of the various stakeholders' needs (primarily to inform the link types to maintain between different information), but use of such models will always be subjective. Problems like these could be assisted through the introduction of dedicated job roles (e.g., independent project documentalist, to augment and unify contributions, encourage objectivity, etc.).

7.2:Obtaining & recording information

Much progress has been made in the ability to obtain and record diverse types of RE information. For example: the history of requirements evolution (REMAP [49]); requirements trade-offs (KAPTUR [1]); explanations and justifications (XPLAIN [44]); a record of collaborative activities (Conversation Builder [33]); and multimedia information [45]. For comprehensive coverage, such tools could be amalgamated in an exploratory workbench (or requirements pre-processor), using suitable integration standards. With additional computer metaphors, so that more RE activities can be carried out on-line, more of this information could be produced as a by-product of main activities. Advances here can be informed through the use of ethnography, or ethnomethodology, to study and describe the details of requirements production, use, and manipulation.

7.3:Organising & maintaining information

To support iterative development, information requires flexibility of content and structure. Relevant work includes the use of: viewpoints as a structuring principle [19]; logical frameworks for modelling and analysing an RS to support gradual elaboration [13]; hypertext to provide explicit visibility of structure and maintain relations [22]; and change models [40]. More research is needed to deal with informal and

unstable information. Much could also be gained from: guidelines to reconceptualise requirements as modular viable systems; the object-oriented representation of self-monitoring multimedia objects; various rollback strategies for persistent repositories; and the creation of explicit job roles to cover the responsibilities of: (a) project librarian (to collect, clean-up, and distribute information); (b) repository manager (to coordinate, control, and maintain information integrity); and (c) RT facilitator (to provide and ensure continual RT).

7.4:Access & presentation of information

RT is predominantly hardwired, predefining what can be traced, and its presentation [21]. Developments in areas such as information retrieval, artificial intelligence, and human computer interaction, are often pertinent. Focused research, like that separating the representation of requirements from flexible presentation, offers potential [31]. Programmable multimedia workstations for end-users would also enable: graphical and textual traces; sophisticated visualisation, to assist activities like impact analysis (i.e., presenting requirements dynamically, using animation, links which light up, etc.); concurrent (global and local) traces; and engaging methods of interrogation. To account for the context of end-use, research is needed to provide flexible RT, where traces can dynamically mature to queries.

8:A research agenda

The current research and recommendations concentrate on throwing increasing amounts and types of information at the pre-RS traceability problem. When such information is generated through adherence to methods, models, or guidelines, it will vary in reliability, as these are rarely used as intended. Any manually-provided information will suffer from subjectivity and incompleteness, as it is difficult to be reflexive, notions of relevance differ, classification schemes are rarely shared, and equal commitment to detail is unlikely. Furthermore, there will always be occasions when the information required will either: not be there; be tailored to a different audience; or not be entirely suited to the purposes at hand.

8.1:Location & access of pre-RS sources

In our investigations, we found that practitioners regularly encountered the above situation. When they do, their fall-back strategy involves identifying and talking to those who can assist. A statistically significant finding was the agreement that the most useful pieces of pre-RS information were: (a) the ultimate source of a requirement; and (b) those involved in the activities which led to its inclusion and refinement in the RS. RT problems (to date), have been solely attacked with techniques that aim to supplant human contact with information. However, even when suitable information is available, the ability to augment this with face-to-face communication was found to be desirable, often essential, and even a fundamental working practice. It is the inability to do just this which we found to underlie many of the continued RT problems.

This finding implies that both *eager* and *lazy* generation of project information is required. By eager, we mean whilst engaged in aspects of RS production. Such information is well suited to the immediate needs of those involved and useful as a later reference point. With time, this static snapshot is less suited to additional needs, and difficult to interrogate. Information generated on need (i.e., lazily and by those originally responsible), can be provided with hindsight, and targeted to specific needs. Both are essential. Without reference to information recorded at the time, to regain context, it would become increasingly difficult to reproduce the required details.

8.2:Location & accessibility: the crux of the problem

Surprisingly, the inability to locate and access the sources of requirements and pre-RS work was *the most commonly cited problem* across all the practitioners in our investigations. This problem was also reported to be a major contributor to others:

- An out of date RS, as an RS evolves poorly when those originally responsible are not involved in its evolution, or where it is impossible to regain the original context.
- Slow realisation (and deterioration as a result) of change, as the most time-consuming and erroneous part is often the identification of those to involve and inform.
- Unproductive conflict resolution, decision making, and negotiation, as most tools supporting these activities do not help to identify or locate the essential participants.
- Poor collaboration, as the invisibility of changing work structures and responsibilities makes it difficult to: transfer information amongst parties; integrate work; and assign work to those with relevant knowledge and experience.
- Difficulty in dealing with the consequences when individuals leave a project and with the integration of new individuals.
- Poor reuse of requirements, as reuse is mainly successful when those initially responsible for their production are either directly involved or readily accessible.

This problem was often reported to be due to politics, which prohibited any knowledge of, or access to, the original sources or requirements engineers. This can only be addressed by re-examining the policies of affected projects. The other reason behind this problem was reported to be the difficulty in keeping track of the original sources and subsequent traces of participation. The common approach, listing contributors to information in document fields, was felt insufficient. This cause of the problem can be tackled with suitable assistance.

Certain project characteristics were found to promote the occurrence of this problem. In projects consisting of individuals split into a number of teams, the location and access of sources was found to be either impossible, time consuming, or unreliable. This was due to: a lack of shared or project-wide commitment; information loss; inability to assess the overall state of work or knowledge; little cross-involvement; poor communication; minimal distribution of information; and changing notions of ownership, accountability, responsibility, and working structure (characteristics amongst those identified elsewhere as contributors to project failure). Characteristics that reduced its occurrence were found in projects consisting of few individuals, due to: a clear visibility of responsibilities and knowledge areas; clarity of working structures; team commitment and ownership; and individuals who acted as *common threads of involvement* (also contributors to success).

8.3:Related work

Many project management tools provide facilities to model organisational charts, role structures, work breakdown

structures, work-flow, etc. Although these are often incorporated in CASE tools (e.g., the ProKit WORKBENCH [54]), they are not well suited to the location and access problem. They tend to be descriptive, prescriptive, or predictive, so used to model formal, static structures and predefined work plans. The drift between what is modelled, what took place, and what is the case in later project life, can be substantial. RS production and maintenance is a social accomplishment in which such structures are continuously created and recreated. Notions like ownership and responsibility are often only transient. The ability to locate relevant individuals therefore deteriorates as the volume and complexity of communication paths grow over time. Models which can reflect these dynamics and manage this complexity are critically needed.

Some recent models (e.g., DesignNet [37]), make initial attempts here, typically through the ability to restructure plans, and so forth. Models of the organisational environment in which a system is intended to operate are also relevant. These each tend to embed different views of an organisation. They focus on specific structures, such as the intentional structure [60] or the responsibility structure [55], so singularly lack an appreciation of the wider organisational context. Collective and more dynamic variants of these could help clarify the organisational structure of development projects. Process modelling research is of further interest here, as these models promise to provide a fuller understanding of the complete environment in which a system is developed (see [43]).

Following a comprehensive analysis of software errors, recommendations were made for modularising responsibility and promoting communication [39]. Our studies independently consolidate and particularise this: RT problems will persist when accurate responsibility cannot be located and these individuals cannot be accessed for the informal communication often necessary to deal with them. The remedy is to provide a continuously up to date picture which promotes and instigates these activities. Our current research is directed at exactly this.

9:Conclusions

We have illustrated the multifaceted nature of the so-called "requirements traceability problem" that many practitioners claim to experience. We have shown why little real progress has been made here, and how this can only be achieved if based on a thorough understanding of the actual problem. We have distinguished between pre-RS and post-RS traceability, demonstrated how advances in the former are needed and offer most opportunity, and made suggestions for progress here.

In conclusion, to achieve any order of magnitude improvement with the RT problem, there is a need to re-focus research efforts on pre-RS traceability. Of particular concern is the intrinsic need for the on-going ability to rapidly locate and access those involved in specifying and refining requirements, to facilitate their informal communication. Continuous and explicit modelling of the social infrastructure in which requirements are produced, specified, maintained, and used (reflecting all changes), is fundamental to this re-orientation.

Acknowledgements

Much of this work was carried out by the principle author, whilst at Oxford University, supported by a BT University Research Initiative. This author would also like to thank the States of Jersey for continued financial support. Both authors acknowledge the comments and assistance of colleagues, students, and anonymous referees. In particular: David Michael; Marina Jirotka; Matthew Bickerton; Joseph Goguen; Daniel Berry; Jeff Kramer; and Manny Lehman.

References

[1] Bailin, S.C., Moore, J.M., Bentz, R. & Bewtra, M. (1990). KAPTUR: Knowledge Acquisition for Preservation of Tradeoffs and Underlying Rationales, *Proceedings of the Fifth Conference on Knowledge-Based Software Assistant*, Liverpool NY, Sept.

[2] Bowen, J., O'Grady, P. & Smith, L. (1990). A Constraint Programming Language for Life-Cycle Engineering, *Artificial Intelligence in Engineering*, Vol. 5, No. 4, pp. 206-220.

[3] Brown, A.W., Earl, A.N. & McDermid, J.A. (1992). *Software Engineering Environments: Automated Support for Software Engineering*, McGraw-Hill.

[4] Brown, P.G. (1991). QFD: Echoing the Voice of the Customer, *AT&T Technical Journal*, March/April, pp. 21-31.

[5] CADRE. (1992). *Teamwork/RqT*, Marketing Brochure, CADRE Technologies, Inc.

[6] Chikofsky, E.J. & Rubenstein, B.L. (1988). CASE: Reliability Engineering for Information Systems, *IEEE Software*, March, pp. 11-16.

[7] Conklin, J. & Begeman, M.L. (1988). gIBIS: A Hypertext Tool for Exploratory Policy Discussion, *ACM Transactions on Office Information Systems*, Vol. 6, No. 4, pp. 303-331.

[8] Cooke, J. & Stone, R. (1991). A Formal Development Framework and its Use to Manage Software Production, in [25], pp. 10/1.

[9] Davis, A.M. (1990). *Software Requirements: Analysis and Specification*, Prentice-Hall, Inc.

[10] Davis, C.G. & Vick, C.R. (1977). The Software Development System, *IEEE Transactions on Software Engineering*, Vol. 3, No. 1, pp. 69-84.

[11] Devanbu, P., Brachman, R.J., Selfridge, P.G. & Ballard, B.W. (1991). LaSSIE: A Knowledge-Based Software Information System, *Communications of the ACM*, Vol. 34, No. 5, pp. 34-49.

[12] Dorfman, M. & Thayer, R.H. (1990). *Standards, Guidelines, and Examples on System and Software Requirements Engineering*, IEEE Computer Society Press Tutorial.

[13] Dubois, E. (1990). Logical Support for Reasoning About the Specification and the Elaboration of Requirements, *Artificial Intelligence in Databases and Information Systems*, Meersman, R.A., Shi, Z. & Kung, C.H. (Eds.), Elsevier Science Publishers B.V., pp. 79-98.

[14] Easterbrook, S. (1991). *Elicitation of Requirements from Multiple Perspectives*, Ph.D Thesis, Department of Computing, Imperial College of Science, Technology & Medicine, London University, June.

[15] European Space Agency. (1987). *ESA Software Engineering Standards*, ESA PSS-05-0, Issue 1, Jan., ESA Publications Division.

[16] Evans, M.W. (1989). *The Software Factory*, John Wiley and Sons.

[17] Finkelstein, A. (1991). A (Neat) Alphabet of Requirements Engineering Issues, in Van Lamsweerde, A. & Fugetta, A. (Eds.), *ESEC '91: 3rd European Software Engineering Conference*, Milan, Italy, Oct. 21-24, Springer-Verlag, pp. 489-491.

[18] Finkelstein, A. (1991). Tracing Back *from* Requirements, in [25], pp. 7/1-7/2.

[19] Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. & Goedicke, M. (1992). ViewPoints: A Framework for Integrating Multiple Perspectives in System Development, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 2, No. 1, pp. 31-57.

[20] Fischer, W.E. (1991). CASE Seen From Both Sides of the Fence, in Van Lamsweerde, A. & Fugetta, A. (Eds.), *ESEC '91*, Milan, Italy, Oct. 21-24, Springer-Verlag, pp. 509-511.

[21] Flynn, R.F. & Dorfman, M. (1990). The Automated Requirements Traceability System (ARTS): An Experience of Eight Years, in *System and Software Requirements Engineering*, Thayer, R.H. & Dorfman, M. (Eds.), IEEE Computer Society Press Tutorial, pp. 423-438.

[22] Garg, P.K. & Scacchi, W. (1989). ISHYS: Designing and Intelligent Software Hypertext System, *IEEE Expert*, Fall '89, pp. 52-63.

[23] Gotel, O.C.Z. & Finkelstein, A.C.W. (1993). *An Analysis of the Requirements Traceability Problem*, Technical Report TR-93-41, Department of Computing, Imperial College.

[24] Hamilton, V.L. & Beeby, M.L. (1991). Issues of Traceability in Integrating Tools, in [25], pp. 4/1-4/3.

[25] IEE. (1991). *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1, Digest No.: 1991/180.

[26] IEEE. (1984). *IEEE Guide to Software Requirements Specifications*, ANSI/IEEE Standard 830-1984.

[27] Interactive Development Environments. (1991). *Software Through Pictures: Products and Services Overview*, IDE, Inc.

[28] Jackson, J. (1991). A Keyphrase Based Traceability Scheme, in [25], pp. 2/1-2/4.

[29] Jarke, M. & Pohl, K. (1992). Information Systems Quality and Quality Information Systems, in Kendall, K.E., Lyytinen, K. & DeGross, J.I. (Eds.), *The Impact of Computer Supported Technologies on Information Systems Development*, Elsevier Science Publishers B.V., pp. 345-375.

[30] Johnson, W.L., Feather, M.S. & Harris, D.R. (1991). Integrating Domain Knowledge, Requirements, and Specifications, *Journal of Systems Integration*, Vol. 1, pp. 283-320.

[31] Johnson, W.L., Feather, M.S. & Harris, D.R. (1992). Representation and Presentation of Requirements Knowledge, *IEEE Transactions on Software Engineering*, Vol.18, No.10, pp. 853-869.

[32] Kaindl, H. (1993). The Missing Link in Requirements Engineering, *ACM SIGSOFT Software Engineering Notes*, Vol. 18, No. 2, pp. 30-39.

[33] Kaplan, S.M. (1990). Conversation Builder: An Open Architecture for Collaborative Work, in Diaper, D., Gilmore, D., Cockton, G. & Shackel, B. (Eds.), *HCI Interact '90, Proceedings of the IFIP TC 13 3rd International Conference on HCI*, Cambridge, UK, Aug. 27-31, Elsevier Science Publishers B.V., North-Holland, pp. 917-922.

[34] Keys, E. (1991). A Workbench Providing Traceability in Real-Time System Development, in [25], pp. 3/1-3/2.

[35] Kuwana, E. & Herbsleb, J.D. (1993). Representing Knowledge in Requirements Engineering: An Empirical Study of What Software Engineers Need to Know, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, Jan. 4-6, pp. 273-276.

[36] Lefering, M. (1993). An Incremental Integration Tool Between Requirements Engineering and Programming in the Large, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, Jan. 4-6, pp. 82-89.

[37] Liu, L.C. & Horowitz, E. (1989). A Formal Model for Software Project Management, *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, pp. 1280-1293.

[38] Lubars, M., Potts, C. & Richter, C. (1993). A Review of the State of the Practice in Requirements Modeling, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, Jan. 4-6, pp. 2-14.

[39] Lutz, R.R. (1993). Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, Jan. 4-6, pp. 126-133.

[40] Madhavji, N.H. (1992). Environment Evolution: The Prism Model of Changes, *IEEE Transactions on Software Engineering*, Vol. 18, No. 5, pp. 380-392.

[41] Marconi Systems Technology. (1992). *Requirements Traceability and Management Manual V1.2.4*, GEC Marconi Ltd.

[42] Mays, R.G., Orzech, L.S., Ciarfella, W.A. & Phillips, R.W. (1985). PDM: A Requirements Methodology for Software System Enhancements, *IBM Systems Journal*, Vol. 24, No. 2, pp. 134-149.

[43] Mi, P. & Scacchi, W. (1990). A Knowledge-Based Environment for Modeling and Simulating Software Engineering Processes, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2, No. 3, pp. 283-294.

[44] Neches, R., Swartout, W.R. & Moore, J.D. (1985). Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development, *IEEE Transactions on Software Engineering*, Vol. 11, No. 11, pp. 1337-1351.

[45] Palmer, J.D. & Fields, N.A. (1992). An Integrated Environment for Requirements Engineering, *IEEE Software*, May, pp. 80-85.

[46] Polack, A.J. (1990). Practical Applications of CASE Tools on DoD Projects, *ACM SIGSOFT Software Engineering Notes*, Vol. 15, No. 1, pp. 73-78.

[47] Ramamoorthy, C.V., Garg, V. & Prakash, A. (1986). Programming in the Large, *IEEE Transactions on Software Engineering*, Vol. 12, No. 7, pp. 769-783.

[48] Ramamoorthy, C.V., Garg, V. & Prakash, A. (1988). Support for Reusability in Genesis, *IEEE Transactions on Software Engineering*, Vol. 14, No. 7, pp. 1145-1153.

[49] Ramesh, B. & Dhar, V. (1992). Supporting Systems Development by Capturing Deliberations During Requirements Engineering, *IEEE Transactions on Software Engineering*, Vol. 18, No. 6, pp. 498-510.

[50] Ramesh, B. & Edwards, M. (1993). Issues in the Development of a Requirements Traceability Model, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, Jan. 4-6, pp. 256-259.

[51] Reubenstein, H.B. & Waters, R.C. (1991). The Requirements Apprentice: Automated Assistance for Requirements Acquisition, *IEEE Transactions on Software Engineering*, Vol. 17, No. 3, pp. 226-240.

[52] Robinson, D. (1991). CASE Support for Large Systems, in Van Lamsweerde, A. & Fugetta, A. (Eds.), *ESEC '91*, Milan, Italy, Oct. 21-24, Springer-Verlag, pp. 504-508.

[53] Smithers, T., Tang, M.X. & Tomes, N. (1991). The Maintenance of Design History in AI-Based Design, in [25], pp. 8/1-8/3.

[54] Sodhi, J. (1991). *Software Engineering: Methods, Management, and CASE Tools*, McGraw-Hill.

[55] Strens, R. & Dobson, J. (1992). *On the Modelling of Responsibilities*, Computing Laboratory, University of Newcastle.

[56] Takeda, N., Shiomi, A., Kawai, K. & Ohiwa, H. (1993). Requirements Analysis by the KJ Editor, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, Jan. 4-6, pp. 98-101.

[57] Texas Instruments. (1988). *A Guide to Information Engineering Using the IEF: Computer-Aided Planning, Analysis, and Design*.

[58] U.S. Department of Defense. (1988). *Military Standard: Defense System Software Development*. DOD-STD-2167A. Washington, D. C., Feb. 29.

[59] West, M. (1991). The Use of Quality Function Deployment in Software Development, in [25], pp. 5/1-5/7.

[60] Yu, E.S.K. (1993). Modelling Organizations for Information Systems Requirements Engineering, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, Jan. 4-6, pp. 34-41.