An Analysis of the *Requirements Traceability Problem*

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

Imperial College of Science, Technology and Medicine
Department of Computing
180 Queen's Gate
London SW7 2BZ

Abstract

In this paper, we investigate and discuss the underlying nature of the requirements traceability problem. Our work is based on empirical studies carried out with over a hundred practitioners and an evaluation of current support for requirements traceability. We introduce the distinction between pre-requirements specification traceability and post-requirements specification 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 mainly due to the lack of (or inadequate) pre-RS traceability and explain 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 further research.

1. Introduction

Requirements traceability is recognised as a concern in an increasing number of standards and guidelines for systems and software requirements engineering [Dorfman & Thayer 1990]. This concern is reflected by the variety of systems that have been developed to address requirements traceability issues and by a growing research interest in the area [IEE 1991, Thayer & Dorfman 1990]. Although there have been many recent advances, requirements traceability remains a widely reported problem area by industry. We attribute the persistence of requirements traceability problems to the lack of any thorough problem analysis. Each proposed solution has consequent shortcomings.

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

> **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 periods of on-going refinement and iteration in any of these phases).

We further introduce two fundamental types of requirements traceability:

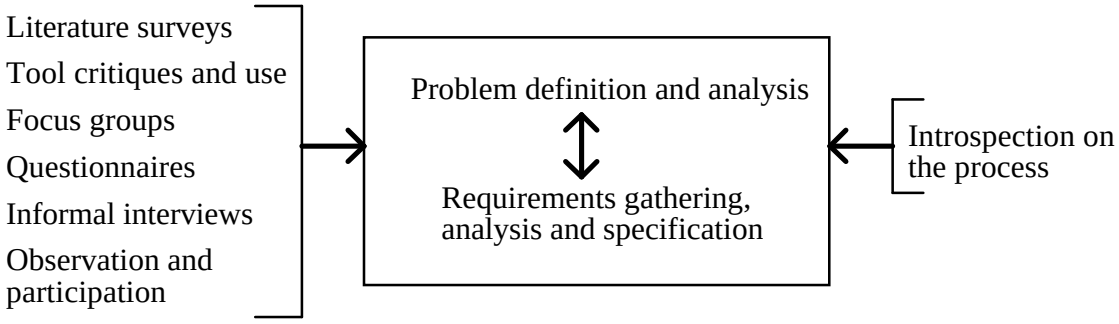
> **Pre-requirements specification (pre-RS) traceability**, which is concerned with those aspects of a requirement's life prior to its inclusion in the RS (requirement production).

Post-requirements specification (post-RS) traceability, which is concerned with those aspects of a requirement's life that result from its inclusion in the RS (requirement deployment).

In this paper, we review the state-of-the-art in support for requirements traceability, analyse the nature of the requirements traceability problem, and make recommendations for addressing this problem. We describe our empirical investigations in Section two and current support is evaluated in Section three. In Section four, we examine the underlying causes of the requirements traceability problem in practice. A framework for addressing these is presented in Section five, in which the distinction is made between pre-RS and post-RS traceability. Section six describes the problems compounding pre-RS traceability improvements in practice and Section seven identifies areas in which some of these problems can be readily addressed. Section eight discusses that aspect of the requirements traceability problem which is the subject of our on-going research.

2. Research method

A combination of data gathering techniques were used to analyse the requirements traceability problem and what is needed to address it. These are outlined in Figure one and described below. To ensure representative coverage, the empirical exercises were carried out over a year period with more than a hundred practitioners in total. Their working areas covered all aspects of development, maintenance and management; their experience ranged from nine months to over thirty years; and their involvement in projects varied (i.e., in the number, type, and size of project).



(Figure 1: Data gathering techniques used.)

2.1. Literature and tool reviews

The literature was surveyed to gather many interpretations as to what requirements traceability is, why it is needed, and what the problems with it are. This was also done to locate research and development in relevant areas. Over a hundred commercial tools and research products were reviewed, to critique the support they offer for requirements traceability, and to identify where subsets of the requirements for pre-RS traceability are already met. This critique was informed by appropriate documentation and marketing brochures, and supplemented by practical demonstrations, hands-on experience, and discussions with practitioners using various tools.

2.2. Focus groups

Five semi-structured focus group sessions were conducted with small groups of practitioners. These were spread across five development sites of a major U.K. company and involved thirty seven practitioners in total. Each session lasted one hour, was audio taped, and later transcribed. The data were used to find out (in their own terms and based on their own experience), what requirements traceability is, what problems it entails and causes, how these problems are overcome (if at all), and suggestions for improvement. The results were also used to direct the design of the questionnaires.

2.3. Questionnaires and follow up interviews

A two-stage questionnaire was used to channel the data gathering. The first stage was short and contained general questions related to practice. This was designed to rapidly gather broad data from a wide population of practitioners, involved in all aspects of systems and software development and maintenance, and to target smaller populations from which more specific data could be gathered. Eighty of these were distributed and sixty nine percent were returned. The second stage was long and detailed. Each of these questionnaires was individually tailored to the primary working areas, job roles, and experiences of practitioners, ensuring a balance between the different facets of development, management and maintenance. Thirty nine of these were distributed and eight five percent were returned. Cross analysis of the responses was possible as the questions were drawn from a reusable pool. These provided a deeper understanding of the problems and issues involved in requirements traceability and identified requirements for addressing them.

The questionnaires were followed up with two informal interview sessions. These were carried out with large groups of the questionnaire respondents and lasted one and a half hours each. They were used to corroborate the questionnaire answers, to probe beyond the answers to appraise their validity, to extract supplementary background information, to encourage spontaneous comments not possible in a questionnaire scenario, and to check the preliminary analysis of the replies. These helped to firm up the problem analysis and requirements specification.

2.4. Observation and participation

Data were also gathered following the observation of (and participation in) a variety of requirements production and development exercises for different projects. For instance, Rapid Application Development workshops were observed and analysed. Here, requirements are dynamically generated, under the guidance of a facilitator, amongst a team of stakeholders. The process and its results are concurrently documented by a scribe. In observing such workshops, comprehensive notes were taken, and any informal documents that were either collectively or individually produced during the process were collected. Our analysis was concerned with comparing these artefacts with the eventual end products of the workshop.

2.5. Summary

Supplementary data gathering techniques were used to combat their individual limitations and to combine their strengths [Goguen & Linde 1993]. A requirements specification for requirements traceability was iteratively produced alongside all these investigations, driving both their direction and focus. Our own experiences throughout this process were continuously reflected upon, to identify requirements for supporting both this activity and its traceability.

3. Current support for requirements traceability

In this section, we review existing support for requirements traceability and summarise the state-of-the-art. The basic techniques employed, and various forms of automated tool support, are described below.

3.1. Basic techniques

A number of techniques have been used for providing requirements traceability, including: cross referencing schemes, based on some form of tagging, numbering, or indexing [Evans 1989]; keyphrase dependencies [Jackson 1991]; templates [Interactive Development Environments 1991]; requirements traceability matrices [Davis 1990]; matrix sequences [Brown 1991]; hypertext [Kaindl 1993]; integration documents [Lefering 1993]; assumption-based truth maintenance networks [Smithers *et al.* 1991]; and constraint networks [Bowen *et al.* 1990]. 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 requirements traceability throughout a project.

Additionally, some form of requirements traceability can often result as a by-product of using certain languages, models and methods for development. This is particularly exemplified by: the Requirements Statement Language [Davis & Vick 1977]; process entity-relationship models [Hamilton & Beeby 1991]; the DesignNet model [Liu & Horowitz 1989]; the Planning and Design Methodology [Mays *et al.* 1985]; formal methods [Cooke & Stone 1991]; object-oriented methods [Henderson-Sellers & Edwards 1990]; and Quality Function Deployment [West 1991]. Here, requirements traceability is dependent on the use of distinct procedures and notations, and the end results will vary according to how rigidly these are adhered to.

3.2. Automated support

Many commercial tools and research products support requirements traceability. This is primarily because they embody either manual or automated forms of the above techniques. As it is not possible to review all of these here, we use the classification scheme below to provide some representative examples and to describe their basic mechanics. Table one clarifies differences in both the type and extent of support offered, and points out the main strengths and weaknesses.

General-purpose tools

General-purpose tools include: hypertext editors; word processors; spreadsheets; database management systems; prototyping tools; etc. These can be hand-configured to allow previously manual and paper-based requirements traceability tasks to be carried out on-line. This generally involves establishing cross references and placing conditions upon their automatic update.

Special-purpose tools

A number of tools support single and well-defined activities related to requirements engineering. Of these, some achieve restricted types of requirements traceability. For example: the KJ-editor assists the organisation of idea formulation, providing traceability between ideas and requirements [Takeda *et al.* 1993]; PORC assists interview transcript analysis, providing traceability between interview transcripts and derived requirements [Langford 1991]; and the T tool assists test case generation, providing traceability between requirements and test cases [Sodhi 1991]. Although there may be a limited degree of explicit control and guidance, support is generally implicit in the use of the tool, which automates any mundane and repetitive tasks needed to provide this requirements traceability.

Workbenches

When a collection of the above types of tool are organised to support a coherent set of activities, less restricted types of requirements traceability can be supported. The degree of support depends on the focal activity of the composite tool. ˝Where requirements traceability and requirements management is the main concern, as in the Automated Requirements Traceability System [Flynn & Dorfman 1990], RTrace (referenced in [Sun Microsystems, Inc. 1990]) and the Requirements and Traceability Management System [Marconi Systems Technology 1992], all tools and activities are configured to ensure requirements traceability. We refer to these as requirements traceability workbenches. Typically centred around a database management system of some form, they comprise dedicated tools for documenting, parsing, editing, interlinking, organising, and managing requirements. They often provide facilities to help assess and carry out any changes made to these requirements.

Other upper-CASE workbenches which focus their support on requirements engineering activities, frequently provide some degree of support. This support can either be explicit and through specific requirements traceability components in the workbench (e.g., a Coupling Module in AGE [Keys 1991]), or implicit from having carried out other activities using the workbench tools (e.g., the Requirements Apprentice [Reubenstein & Waters 1991]). CASE workbenches which accept requirements documents as a starting point, from which to drive design and implementation, commonly provide coarse-grained requirements traceability between requirements and their realisation in subsequent phases.

Environments and beyond

Requirements traceability can potentially be provided throughout a project's life if tools supporting all aspects of development are integrated (e.g., the Virtual Software Factory, referenced in [Sun Microsystems, Inc. 1990]). The basis used for internal integration tends to define how requirements traceability is established: through the use of a common language (e.g., the Input/Output Requirements Language in Technology for the Automated Generation of Systems [Sodhi 1991]); through the use of common structures (e.g., the relations of an Entity-Relation-Attribute Model in Genesis [Ramamoorthy *et al.* 1988]); through the use of a common method (e.g., the Information Engineering Method in the Information Engineering Facility [Texas Instruments 1988]); or through the use of specialised requirements traceability tools or sophisticated repository structures where a number of interlocking tools are combined to support many languages, methods or structures (e.g., Teamwork/RqT [CADRE 1992]). Those with the flexibility to incorporate third-party environments tend to provide requirements traceability support through the use of powerful repositories and underlying database management systems. These are used to relate the products of the individual components (e.g., the Digital CASE Environment [Sodhi 1991]).

Requirements Traceability (RT)	**(A) General-purpose tools**	**(B) Special-purpose tools**	**(C) Workbenches**	**(D) Environments and beyond**
(1) Priority given to RT	Although any general-purpose tool can potentially be configured for RT purposes, RT is not a concern of the basic tool.	Individual tools that support requirements-related activities (i.e., specific analysis techniques), often provide some form of RT as a by-product of use, but RT is not the tool's focus.	RT priority varies dependent on the focal set of activities. Where these are RT and requirements management (RT workbenches), RT is the main concern, else RT is a side concern in those workbenches focusing on other requirements activities.	RT is typically a side concern. The extent of this concern depends on the types of tool contained in the environment (i.e., whether or not there are dedicated tools for RT).

Requirements Traceability (RT)	**(A) General-purpose tools**	**(B) Special-purpose tools**	**(C) Workbenches**	**(D) Environments and beyond**
(2) Support provided for RT	No explicit support is provided. RT must be hand-crafted and the resulting support depends on the effort expended in so doing. The focus can easily become tool configuration rather than RT.	Support is implicit in the framework provided for carrying out the main activity of the tool. Mundane tasks necessary to provide basic RT are typically automated as a result of proper tool use.	In RT workbenches, support is explicit (else as B). These offer: (i) Guidance - providing RT through adherence to the requirements engineering approach and work steps supported, typically top-down and decompositional, and by predefining information to collect and link types to establish. (ii) Assistance - parsing textual documents to identify and tag requirements, establishing (syntactic) links between them, and through a repository which manages simple bookkeeping tasks and enables rudimentary checking. No analytical ability is really provided.	RT is provided as a by-product of co-ordinated tool use and adherence to the development philosophy supported. The extent of support depends on the internal integration strategy and/or repository structure. There is more guidance and assistance if it includes dedicated RT tools, or if RT is an explicit concern of its approach. RT maintenance is supported if the repository can manage large amounts of information and reconfigure in the light of change.
(3) Requirements-related information that can be made traceable	The 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.	The tool 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 it supports. Only a limited scope of requirements-related information can be traced.	The potential to trace a flexible diversity of requirements-related information in those workbenches supporting requirements activities, including multimedia information in some. RT workbenches can impose arbitrary limits on the amount and type of information, and this is often only textual. These can trace information concerning how an RS was produced, but usually only its derivation from a textual baseline, rather than of its exploratory development and refinement, or of the environmental context in which it was produced. Additional information can often be recorded, as informal notes, but is of limited use for RT purposes.	These have the potential to trace all project information, generated in all project phases, related to requirements. The tendency is to focus on that information derived from requirements in the RS in later project phases, so less emphasis on development-related information about individual requirements (i.e., requirements information is often thinly spread). However, these can support the RT of versions, variants, and of user-definable trace items.
(4) Tasks and job roles that RT can assist	These can be tailored so that RT can support any task and job role, though it is problematic to meet different needs simultaneously.	RT is provided to specifically assist the activity the tool supports and the predefined role of the tool user. Their task-specific frameworks constrain the domain of working and are difficult to configure for other purposes.	The RT provided can support a breadth of activities within the concern of the tool's domain (i.e., assist requirements checking, reporting, etc.), so provide dedicated support for specific jobs. They are often configurable to support tasks in additional project phases. The RT provided by RT workbenches tends to support managerial activities rather than the activities of those involved in producing the RS.	RT can assist a wide coverage of lifecycle-wide tasks and roles (e.g., related to maintenance and management, such as impact analysis and progress reporting). Not all are equally well supported and there tends to be more support for activities related to requirements use rather than their production and refinement.
(5) Longevity of RT support	General-purpose tools are typically configured to address immediate needs. RT can degrade with large amounts of information and time, as they are not usually integrated with lifecycle-wide tools, and are poor at handling changes and evolution, unless explicitly prepared for.	As these provide RT at a snapshot in time to support a specific activity, they neglect the requirements for on-going management. Longevity of support depends on both horizontal and vertical integration with other tools.	RT is provided throughout the duration of the activities supported. As they are predominantly forwards-engineering tools, RT can deteriorate with progression to later phases, as it can be difficult to reflect the work here and account for any iteration. Longevity of support depends on vertical integration with other tools.	It is possible to provide RT throughout a project's life, though this tends to start from a relatively static baseline. The tightness and granularity of RT depends on the underlying repository and the degree of internal integration. RT can deteriorate over time, due to iteration problems and poor feedback.

Requirements Traceability (RT)	**(A) General-purpose tools**	**(B) Special-purpose tools**	**(C) Workbenches**	**(D) Environments and beyond**
(6) Support for the traceability of group activities	These promote individualistic working, as they often provide no common or consistent framework for RT. They can encourage immediate and ad hoc solutions. They are typically used, by a single user, to record activities after they have happened.	Most special purpose tools support individualistic working, though some directly support group activities, like the brainstorming of requirements, thus making both the process and its 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 the working practices of requirements engineers. Concurrent work is often difficult to co-ordinate and integrate within the tool, so the potential richness of information can be lost. Participative work is actively supported in some workbenches not focusing on RT, though the subsequent traceability of these activities varies widely.	Multiple users are commonly supported through shareable repositories and techniques to assist the co-ordination and integration of separate activities (e.g., workspaces, views, etc.). This ability often depends on an agreed RS and strict project partitioning, so RT can deteriorate when these are not stable and when overall control is lacking.
(7) Main strengths	(i) Flexibility to provide customised and comprehensive RT to suit individual project and organisational needs. (ii) Often sufficient for the RT required in small and short term projects.	(i) Can provide tight RT sufficient for the immediate needs of particular requirements-related activities. (ii) Those supporting a group activity often provide traceability of this activity.	(i) RT workbenches provide good RT from and back to information which is initially input to the tool, through a breadth of related activities (i.e., fine-grained horizontal RT within requirements phases). (ii) Offer benefits, like facilities for RT checks and clear visibility.	(i) Ability to provide on-going RT (i.e., depth of coverage or vertical RT). (ii) Open environments provide more flexibility in the choice of requirements engineering approach and 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 poor integration and information management potential, preventing fuller and longer RT support.	(i) RT workbenches attempt to be holistic, though none support all activities. Typically, a top-down approach is enforced, classification schemes are predefined, and a (relatively static) baseline is pre-empted, without support for its production and iterative enhancement. As RT depends on correct use, the main concern can become RT rather than RS production. (ii) RT workbenches poorly integrate, so it is difficult to support the RT of early work defining the problem space and to provide on-going RT with later maintenance changes. (iii) The tool dictates, else much manual intervention can be required.	(i) RT is typically coarse-grained and dependent on step-wise development. (ii) The tightness of RT varies, so iteration and later requirements changes can prevent on-going RT (caused by poor backwards RT which cannot account for any manual intervention or work-arounds that occurred). (iii) Increasing flexibility (with those tools open to external integration) is typically counterbalanced by poorer RT.

(Table 1: Tool support for requirements traceability.)

3.3. Summary

It has been noted by others (see [Polack 1990]), that the majority of the tools on the market do not cover requirements traceability, and that even fewer provide support for the particular traceability requirements now enforced by DOD STD-2167A [U.S. Department of Defense 1988a]. Those which do address requirements traceability differ mainly in cosmetics and in the amount of time, effort, and manual intervention they require. The type and extent of support provided depends on the underlying assumptions they embed about requirements traceability and the particular problems and concerns they focus on. The dedicated requirements traceability tools basically employ the same techniques, though they suffer from poor integration and inflexibility. These limitations are reflected by a preference for using general-purpose tools in practice [Lubars *et al.* 1993].

4. Why there is still a requirements traceability problem

Despite growing numbers of specialised tools which support requirements traceability, their use is not widespread, and requirements traceability problems are still cited by practitioners who do use them. Following prolonged investigations with practitioners, we attribute the persistence of requirements traceability problems to a number of fundamental conflicts. These conflicts revolve around little shared agreement concerning: what requirements traceability itself is; what the requirements traceability problem is; what the underlying cause of the requirements traceability problem is; and which additional problems any improvements in requirements traceability should address.

4.1. Lack of common definition

Since the introduction of the term "requirements traceability" by the US Government's Department of Defense, each subsequent attempt at definition, either in the literature or by practitioners, has taken a slightly different form. Definitions are either:

- **Purpose-driven** (i.e., defined in terms of what it should do):

  (i) Requirements traceability is *"the means whereby software producers can 'prove' to their client that: the requirements have been understood; the product will fully comply with the requirements; and the product does not exhibit any unnecessary feature or functionality"* [Wright 1991].

  (ii) *"Requirements traceability is the ability to adhere to the business position, project scope and key requirements that have been signed off"* [Practitioner participating in a focus group].

- **Solution-driven** (i.e., defined in terms of how it should do it):

  (i) *"Traceability refers to the ability of tracing from one entity to another based on given semantic relations"* [Ramamoorthy *et al.* 1986].

  (ii) *"Traceability refers to the ability to cross-reference items in the requirements specification with items in the design specification"* [Roman 1985].

- **Information-driven** (i.e., defined in terms of the information to be made traceable):

  (i) *"Requirements traceability is the ability to link between functions, data, requirements and any text in the statement of requirements that refers to them"* [Practitioner participating in a focus group].

  (ii) The paragraph for requirements traceability must contain *"a mapping of the engineering requirements in this (Software Requirements) Specification to the requirements applicable to this Computer Software Configuration Item in the System/Segment Specification, Prime Item Development Specification, or Configuration Item Development Specification..."* (DI-MCCR-80025A [U.S. Department of Defense 1988b]).

- **Direction-driven** (i.e., defined to emphasise either forwards or backwards direction, or both):

    (i) Traceability is *"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..."* [European Space Agency 1987].

    (ii) Traceability enables *"each requirement to be traced to its origin in other documents and to the software component(s) satisfying the requirement"* [Johnson *et al.* 1991].

Each definition differs in emphasis and delimits the scope of immediate concern. No single definition encompasses all mentioned aspects. This has implications for the development and use of tools to support requirements traceability, especially in large project teams: how can it be coherently and consistently provided if each individual has his or her own understanding of what is meant by requirements traceability?

4.2. Conflicting underlying problems

From our investigations, it was evident that each practitioner had their own understanding as to the main cause of the requirements traceability problem. This finding is reflected in the literature, where the fundamental problem has been attributed to: lack of commitment by all parties [Wright 1991]; coarse granularity of traceable entities [Ramamoorthy *et al.* 1986]; immature integration technology [Brown *et al.* 1992]; knowledge management [Easterbrook 1991]; information complexity [Hoffman 1990]; hidden information [Robinson 1991]; and project longevity [Mays *et al.* 1985].

Similarly, the problems that practitioners believed improved requirements traceability should address were just as diverse. This finding is also reflected in the literature: to track rationale, constraints and relationships used to develop product elements, and to analyse consistency, completeness and process management [Hoffnagle & Beregi 1985]; to verify requirements allocation and flowdown, to assess change impact, and to assist testing [Dorfman & Flynn 1984]; to support the evolvability of requirements specifications [Johnson *et al.* 1991]; to offer some degree of assurance that specifications were written with user requirements in mind and to assist user acceptance testing [Ramamoorthy *et al.* 1984]; to enable safety analysis, audits, and change control [Hamilton & Beeby 1991]; to provide the ability to understand systems from multiple points of view and to assist the pulling together of fragmented information [Easterbrook 1991]; and to permit flexible process modelling [Fischer 1991].

These findings demonstrate that: (a) the phrase "requirements traceability problem" is used to umbrella many underlying problems; and that (b) improvements are expected to yield the solution to additional (and often incompatible) problems. ˝Complicating this further was the observation that each practitioner's perspective of the problems was context-dependent and hence subject to change. This also has implications for providing requirements traceability support: how can this support account for all these problems simultaneously?

4.3. Summary

To date, improvements in requirements traceability have tended to be problem solving rather than problem defining exercises. Techniques have been thrown at the problem without any thorough investigation of what the actual problem is that they should be dealing with. As

illustrated above, this problem is not perceived to be uniform. This is because underlying every situation in which requirements traceability is required, different user, project, task and information requirements come into play, which cumulatively influence the problems experienced. Substantial improvements will be needed in many areas to address the requirements traceability problem.

5. A framework for addressing the requirements traceability problem

It is essential to establish a shared definition of what requirements traceability is, to provide a framework in which we can locate the fundamental cause of the requirements traceability problems that are experienced. Such a definition needs to be general enough to encompass different views of requirements traceability, but specific enough to highlight its significant aspects.

5.1. Defining requirements traceability

The definition which is 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"* [IEEE 1984].

The standard that this definition comes from (ANSI/IEEE Std 830-1984), further recommends: (a) **backward traceability** to previous development stages, which *"depends upon each requirement explicitly referencing its source in previous documents"*; and (b) **forward traceability** to all documents spawned from the software requirements specification, which *"depends upon each requirement in the software requirements specification having a unique name or reference number"*. This definition requires requirements traceability to be established (bidirectionally) between project documents and through the use of a particular scheme. To broaden the scope of this definition, we refer to a definition derived from the word "trace" in the Oxford English Dictionary:

> 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 [Sykes 1978].

Using both these definitions, we define requirements traceability in the following way:

> *"**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 periods of on-going refinement and iteration in any of these phases)."*

5.2. Pre-RS and post-RS traceability

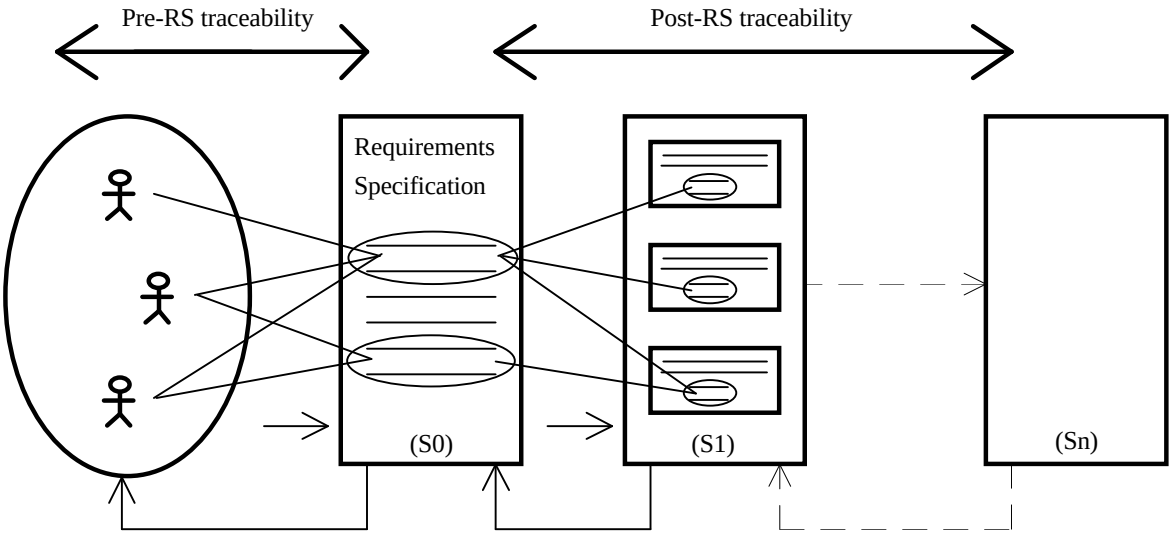
We further suggest that requirements traceability itself can be divided into two basic types. These revolve around the written specification of requirements (the RS), and are what we call *pre-requirements specification (pre-RS) traceability* and *post-requirements specification (post-RS) traceability*. The former is concerned, not only with the ability to record and access the origin of a requirement, but also any additional information which can help describe what has

existed or happened prior to its inclusion in the RS. The latter is concerned with any such information related to a requirement's use.

> *"**Pre-requirements specification (pre-RS) traceability** is concerned with those aspects of a requirement's life prior to its inclusion in the RS (requirement production)."*

> *"**Post-requirements specification (post-RS) traceability** is concerned with those aspects of a requirement's life that result from its inclusion in the RS (requirement deployment)."*

Although forwards and backwards requirements traceability are clearly essential, we emphasise the above separation as our investigations indicate that requirements traceability problems are centred around the current lack of distinction between these two basic types. Comprehensive support can only be provided through an explicit recognition of their differences. Figure two shows the typical setting of requirements traceability to illustrate this distinction. Note the way in which requirements knowledge is distributed and merged in successive representations. Note also the added complication of iteration and change propagation.



(Figure 2: The two basic types of requirements traceability.)

5.3. Support for pre-RS and post-RS traceability

The primary differences between these two types of requirements traceability involves the information they deal with and the problems they can assist [Feather 1991, Mathews & Ryan 1989, Rzepka & Ohno 1985]. The two main phases of a requirement's life impose different requirements on potential support for its traceability. Post-RS traceability depends on the ability to trace requirements from, and back to, a baseline document (the RS), through a succession of documents and products in which they are distributed. When changes are made to this baseline, they need to be re-propagated through this chain of distribution. 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 (often conflicting and overlapping) sources are eventually integrated into a single requirement in the

RS. Any changes in the process need to be re-worked into the RS. When changes are made to the RS, they need to be carried out with reference to this production and refinement process.

Most of the existing support for requirements traceability is directed at providing post-RS traceability. Problems experienced here are an artefact of informal development methods, and can be eliminated by formal development settings which automatically transform an RS into an executable and replay transformations following change [Finkelstein 1991b]. This existing support for post-RS traceability is not directly applicable to providing pre-RS traceability. Pre-RS traceability is needed in recognition that change cannot be adequately handled from the RS alone. Change needs to be both instigated and propagated from its source, to indicate what in the RS and what elsewhere needs changing, so pre-RS traceability needs to make the subtle interrelationships that exist between requirements explicit[1]. The support for post-RS traceability generally treats the RS as a *black-box*, with little to show that the requirements therein are only the end product of a complex process. They further make it difficult to represent this process, because they tend to predefine rigid information categories for recording potentially traceable information, and immediately commit content to syntactic structure. The consequent rigidity of this would make it difficult to account for the dynamic and changing nature of the sources and environment from which requirements are drawn, thus providing little support for the on-going and emergent nature of the work practices involved in producing and refining an RS. Unlike post-RS traceability, it has been argued that the problems of pre-RS traceability will always remain, irrespective of formal treatment [Finkelstein 1991b]. This is beacuse this aspect of a requirement's life is inherently paradigm-independent.

5.4. The need for improved pre-RS traceability

Awareness of the above issue has only recently become apparent [Finkelstein 1991a]. Our empirical findings intensify this concern, as they strongly indicate that the majority of the problems still attributed to poor requirements traceability are in fact due to the lack of (or inadequate) pre-RS traceability, and that techniques are most crucially needed to record and trace information related to the production of an RS. Amongst the other reasons driving practitioner concern for this ability, the most prevalent were to improve quality, and to reduce cost in development and maintenance.

Quality improvements can be attained through its potential to assist: auditing [Chikofsky & Rubenstein 1988]; the handling of changing requirements [Bersoff & Davis 1991]; repeatability [Jarke & Pohl 1992]; and the ability to make logical sense of the requirements, hence avoiding confusion and the production of unacceptable systems [Short 1988]. This is because pre-RS traceability enables previously closed issues, even decisions concerning how to conduct the requirements exercise itself, to be made explicit, possible to re-open, and possible to re-work.

Pre-RS traceability also offers the potential for greater economic leverage, as a significant proportion of development and maintenance cost, time, and effort is presently spent in compensating for invisibility [Devanbu *et al.* 1991]. To make use of an RS, and to maintain it, it is often necessary to reconstruct and rediscover an understanding of how it was produced. This can be a notoriously complex and error-prone endeavour in practice.

[1] Seemingly unrelated requirements in the RS may be strongly interdependent. For example, if an organisational standard that was used to produce some requirements in the RS is changed, the identification of directly and indirectly affected requirements is problematic without pre-RS traceability.

5.5. Summary

The two types of requirements traceability, pre-RS and post-RS, are both important. However, they impose distinct requirements on support. Post-RS traceability is well supported and the remaining problems here are not insurmountable. In contrast, the issues that pre-RS traceability are to deal with are neither well understood, nor comprehensively supported. The means by which post-RS traceability problems can be eliminated will not remove the problems here. Advances in pre-RS traceability are urgently required, as these will be the most instrumental in reducing requirements traceability problems in the long-term, and because these will offer more potential for additional and far-reaching improvements.

6. Problems confronting pre-RS traceability improvements

Having identified insufficient pre-RS traceability as the main contributor to continuing requirements traceability problems, and shown how it is likely to be the only contributor in formal development settings, our problem definition and requirements gathering exercises were re-focused to determine: (a) what improvements in pre-RS traceability would involve; and (b) how these improvements could be realised. These investigations clearly indicated that the main barrier confronting any improvements in pre-RS traceability is the *establish and end-use conflict*. By this, we mean that the two main parties involved (i.e., those who would be in a position to make pre-RS traceability possible and those who would subsequently require it to assist their work), have conflicting problems and needs. A requirements engineer (say), responsible for ensuring pre-RS traceability, is unlikely to require the same things from it as a designer, manager, or someone involved in maintenance[2]. Addressing any one of these concerns often makes it problematic to address the other concerns. Below we list the main problems expressed by those who could potentially make pre-RS traceability possible and the underlying end-user problems that compound their task.

6.1. Problems faced by the providers of pre-RS traceability

- It is perceived as an optional extra by those in a position to resource it, post-RS traceability being given higher priority, so insufficient time, personnel and resources are allocated.

- Pre-RS traceability can rarely be achieved by uncooperating individuals. Such individual efforts are typically ad hoc, localised, and unco-ordinated, especially where there is an imbalance between the extra work involved and the personal benefits gained. It needs to be a combined and full-time responsibility by all involved to succeed. An explicit allocation, awareness, and management of the different roles that practitioners need to assume to achieve three interdependent tasks (i.e., obtaining and documenting required information, organising it, and maintaining it), is typically absent.

- A shared understanding of the diverse requirements for pre-RS traceability, imposed by different stakeholders throughout a project's life, is lacking. There is an obvious tendency to focus on immediate and visible needs, as accounting for the unique and unpredictable nature of end-use is problematic.

[2] The questionnaires pointed out many such conflicts. For instance, those involved with design, implementation, maintenance and managerial aspects of a project, attached a high priority to the ability to trace back to why things are requirements and to requirements process information. Those involved in writing the requirements document, or in work conducted prior to this, attached no such importance to this ability. Amongst their reasons were that: they are aware of such information and do not believe it is of relevance to others; they have too little time or support for providing yet more documentation, so it would distract from their main tasks; and that it is unlikely that all involved would be equally committed in providing for this ability, so they saw little point in their own individual efforts. This implies that those in a position to provide pre-RS traceability have a low motivation to do so, even though other parties involved in development demonstrably required it. Not surprisingly, subsequent questionnaire responses revealed that much of their time was actually spent in explaining exactly this sort of information to others involved in later phases, and that those involved later on spent much of their time actively locating (often unsuccessfully) those individuals who could provide such information.

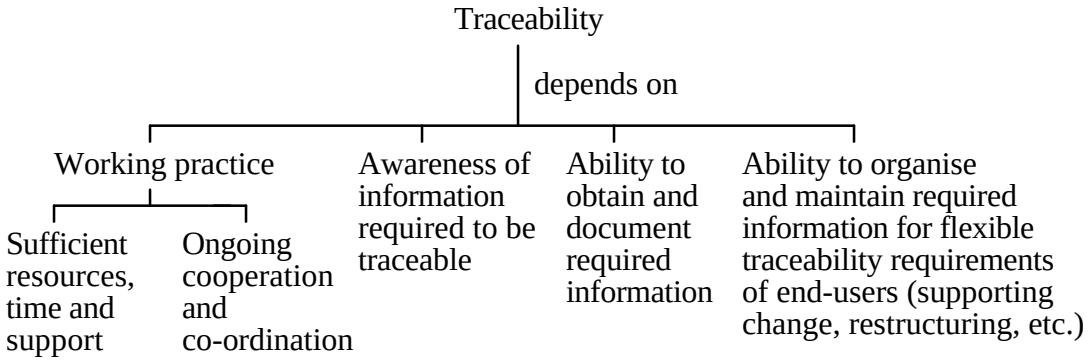
- Concern for pre-RS traceability diminishes, and concern for post-RS traceability increases, after the first snapshot at an RS has been formally signed off. RS production and refinement is a social and on-going activity for which concern must continue throughout a project's life. This is problematic as the exact nature of this activity cannot be fully predefined and because there tends to be poor feedback of later work and requirements changes.

- The information required to be made traceable cannot always be readily obtained and documented (e.g., tacit knowledge). Information which is documented (such as rationale), varies in quality dependent on many factors (like time constraints). Also, a deliverable-driven culture can actively discourage the gathering of certain information.

- The documentation of required information does not imply it will be traceable. A premature commitment to syntactic, rather than to semantic structure, is what typically prohibits this.

- Information that is structured, so that it can be traced in many ways, is no guarantee that it will be up to date and continuously representative. There are problems in accounting for, and in handling, all possible changes. This is mainly due to an immature change culture and the cyclic dependency upon requirements traceability itself.

- Poor feedback regarding best practice, and little dedicated support (be this clerical, procedural, or computer support), perpetuates the same problems.



(Figure 3: Deconstructing the requirements traceability problem for provision.)

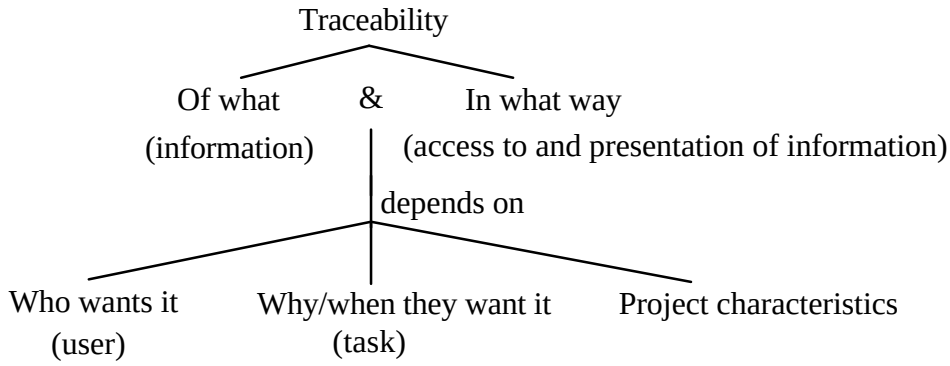
6.2. Problems imposed by the end-users of pre-RS traceability

- A stereotypical end-user cannot be predefined. Requirements for end-use will differ and be inconsistent, even for an individual.

- The potential quantity and heterogeneity of project information required precludes total predefinition. Personal contact is always heavily relied upon, because much of what is required is often undocumented, inaccessible, out of date, or documented in a form not suited to the use for which it is required.

- The way in which end-users require access to (possibly combinations of) information, and how it is most suitably presented for their purposes, cannot be predefined.

- Each end-use situation exhibits unique requirements, precluding predefinition. Problems will always exist if end-users do not have the ability to filter and access the different types of information pertaining to the production of the RS that they require under different circumstances.



(Figure 4: Deconstructing the requirements traceability problem for end-use.)

6.3. Summary

The real challenge in providing support for pre-RS traceability lies in the ability to address the problems confronted by the two main parties, as illustrated in Figures three and four. For the provider of pre-RS traceability, it must be established almost as a by-product of their other work, or be given much higher priority and explicit support. For the end-user, pre-RS traceability must be sensitive to their contextual needs.

To provide an exhaustive account of each requirement in an RS, technological solutions can readily assist with the collection, documentation, and organisation of huge amounts of information. However, the fundamental problems will still reside with the people involved. The end-users need to be able to predefine all their anticipated requirements for pre-RS traceability and make these clear to the providers. The providers need to be able to identify relevant information and document it in a (re)usable form to suit all possible needs. This problem is intensified by the fact that individuals can assume both of these positions. Although we have identified requirements to assist with this dilemma in the course of our investigations, they suggest that technological solutions will not be the complete answer.

7. Solutions to some of the pre-RS traceability problems

Our investigations led to the production of a comprehensive requirements specification[3]. This document stresses what is required to provide and make use of pre-RS traceability, and includes the needs related to stakeholders, performance, operation, tasks (such as change handling and reuse), amongst other aspects. The complexity and diversity of these requirements indicate that it would be premature to offer a solution to the entire pre-RS traceability problem (i.e., one which meets all the requirements we specified), as it is a compound problem in need of improvements in many areas. Having examined a spectrum of products and on-going research, we focus on those subsets of the requirements for which total or partial solutions already exist,

[3] Space prohibits the inclusion of this lengthy document here.

and highlight some of these here[4]. Recommendations for immediate uptake and future research are also made.

7.1. Increasing awareness of useful project information

Although studies have been carried out, to reveal what project information is required by those involved in different phases of development and to inform what needs to be collected [Kuwana & Herbsleb 1993], our investigations indicate that it is not possible to generalise such findings. The appropriate amount and type will remain subject to controversy, and range from deliverables only, through all explicitly generated information, to an unbounded quantity of implicit information suitable for defining the context.

Existing work generally tackles this issue by delimiting the amount, and categorising the type, of information required for assisting focused activities. For example, the Decision Representation Language of the SIBYL system [Lee 1990], and the argumentation scheme of gIBIS [Conklin & Begeman 1988], provide frameworks for representing the structure of decision making and design deliberation. However, such schemes can artificially restrict user input, emphasise the structure as opposed to the content of the information, and are each best suited to particular domains, systems and tasks. Although they may increase the awareness of information needed to support some activities, they do not help promote awareness of all the informational requirements of pre-RS traceability.

Current research into the development of a requirements traceability model is of interest here [Ramesh & Edwards 1993]. The intention of such a model is to increase the awareness of the needs of various stakeholders in the development process, primarily to guide the types of link that should be maintained between different types of project information. However, even with an increased awareness of information required, and of the relation types needed between this information to enable requirements traceability, these categories will not always be shared. Its use, especially by a large number of practitioners or by separate working groups, will be prone to individual subjectivity. Some of these problems could be assisted by establishing dedicated job roles, such as an independent project documentalist to augment and unify individual contributions, and to encourage a more objective view.

7.2. Obtaining and recording diverse project information

There has been much progress in the ability to obtain and record diverse types and detail of project information. For example: the history of the requirements evolution process (REMAP [Ramesh & Dhar 1992]); the design rationale of teams working in real-time (rIBIS [Rein & Ellis 1991]); requirements trade-offs (KAPTUR [Bailin *et al.* 1990]); explanations and justifications (XPLAIN [Neches *et al.* 1985]); a record of collaborative activities (Conversation Builder [Kaplan 1990]); the conversations underlying group work (coordinator software [Marca 1989]); tangible products produced and used, settings in which developed and maintained, and processes carried out (ISHYS [Garg & Scacchi 1989]); distinctive sources in a heterogeneous domain (PROLEXS [Walker *et al.* 1991]); and a rich diversity of multimedia information used in requirements engineering [Palmer & Fields 1992]. Additional advances could be gained from current work directed towards the use of ethnography or ethnomethodology to inform requirements gathering [Jirotka 1991]. Such work would be equally useful for studying and describing information related to the working practices of those involved in requirements production.

[4] This is in recognition that many of the detailed requirements can only be realistically addressed on a project-specific basis, according to immediate and context-specific needs.

Many of the requirements for gathering pre-RS information could be met by amalgamating a variety of the above into an exploratory workbench or requirements pre-processor. How such a tool should be designed and developed needs to be informed by detailed studies of the requirements production process, an understanding of the use and manipulation of requirements throughout a project's life, and through an exploration of appropriate internal and external integration standards. However, with the increased ability to gather extensive project information, the workloads of those involved are likely to increase, and problems in integrating this information are likely to be experienced. More of this information needs to be automatically provided as a by-product of those activities that are considered mainstream. This requires support for more of their activities to be carried out on-line, so increased computer metaphors for individual and group work involved in requirements production.

7.3. Organising and maintaining project information

To support progressive development, maintenance, change, reuse, etc., project information requires flexibility of both content and structure. There is relevant work in many areas for addressing these issues. For example: modularisation techniques and module guides [Parnas *et al.* 1985]; structuring the activities of software engineering for reuse [Freeman 1987]; the use of viewpoints as an organising and structuring principle [Finkelstein *et al.* 1992]; mechanisms to combine viewpoints into single structures (SYNVIEW [Lowe 1985]); logical frameworks for modelling and analysing requirements specifications to support their gradual elaboration [Dubois 1990]; the use of hypertext to provide explicit visibility of structure and to maintain relations (the Document Integration Facility [Garg & Scacchi 1989]); change models and infrastructures for change (PRISM [Madhavji 1992]); the use of abstractions to support rigorous reasoning about change [Ward 1992]; versioning and configuration management [Bersoff & Davis 1991]; and impact analysis and propagation techniques (Mercury [Kaiser *et al.* 1987]). Any necessary extensions in these areas will primarily be to deal with more informal and unstable information. Research in sociology, and in particular on boundary objects, could be used to indicate how information can be structured so that it can be shared between individuals for different purposes [Star 1989].

Automation is not the complete answer for organising and maintaining information, although an advance here would be the provision of guidelines for reconceptualising requirements and requirements-related information as modular viable systems[5]. Additionally, much could be gained from research into the object-oriented representation of multimedia objects, self-monitoring objects, and selective and complete rollback strategies for persistent information stores. Other benefits could result from the introduction of explicit job roles, supported by suitable tools and techniques. These roles could cover the responsibilities of: project librarian, to collect, clean-up, and distribute information; information base manager, to co-ordinate, control, and ensure information is of quality and up to date; and traceability facilitator, to establish and ensure the traceability of all information.

7.4. Flexible access and presentation of project information

Current potential for requirements traceability is predominantly hardwired [Flynn & Dorfman 1990]. This predefines what information can be traced and how this can be presented.

[5] This would involve structuring requirements and requirements-related information so that it is: (1) *modular* (highly cohesive, loosely coupled, with well-defined semantic interfaces), for reuse, and for the increased ability to change content and structure; (2) *generic* (in most abstract form), for adaptability to enable its use in different types of trace, and robustness to maintain continuity of identity; and (3) *parameterised*, for configurability of access and instantiation, and to put information together in alternative and dynamic ways.

Providing the right amount of desired information, at the level of detail necessary for the problem at hand, is not a problem unique to requirements traceability (see [Bocker & Herczeg 1990]). Many developments in information retrieval (particularly using fuzzy logic), artificial intelligence, and human computer interaction, are directly applicable and can address many requirements here ([CACM 1992] provides an example). Recent work, separating the internal representation of requirements information from its (flexible) presentation, is also pertinent [Johnson *et al.* 1992].

Programmable multimedia workstations for end-users are highly recommended. Amongst many benefits, these could enable: the retrieval of multimedia information, through graphical and textual traces; diverse means of visualisation, which could assist impact analysis (i.e., by presenting requirements dynamically, using animation, links which light up, etc.); concurrent (global and local) traces; and alternative engaging methods of interrogation to define requirements traceability requirements. Artificial intelligence or expert system technology could be exploited to provide flexible and user-definable requirements traceability on-the-fly (i.e., to enable traces which dynamically mature to queries and end-use situations).

7.5. Summary

As indicated above, much of what already exists can go a long way towards tackling some of the basic requirements for pre-RS traceability provision, informing what information to obtain, how to do so, how to record it, and how to keep such information up to date and accessible. In addition, we have suggested the desirability of dedicated job roles, the integration of many existing approaches to develop an extensible requirements pre-processor, and recommended a way to reconceptualise requirements for traceability purposes. Work related to the requirements for end-use of pre-RS traceability mainly falls into the areas of information retrieval, artificial intelligence, and human computer interaction. Here, we have further suggested the desirability of programmable workstations and recommended research to address the dynamic and context-sensitive requirements of end-users.

8. A research agenda

The focus of current research, and the recommendations described in section seven, are directed towards providing: (a) comprehensive repositories of project information related to pre-RS work; and (b) elaborate mechanisms to selectively access and present this information. Throwing increasing amounts and types of information at the pre-RS traceability problem, even details of informal activities and techniques to retrieve this, will not completely eliminate it.

Information generated as a by-product of enforced adherence to methods, process models, or guidelines, will vary in reliability, as it is unrealistic to assume that these will always be used as prescribed [Parnas & Clements 1986]. Manually provided information will suffer from subjectivity and incompleteness, not only because it is difficult to be reflexive[6], but notions of relevance differ, classification schemes are rarely shared, and equal commitment to detail is unlikely [Ehn 1988]. The requirements that any individual or group has for pre-RS traceability are situation-specific and not amenable to complete predefinition. There will always be instances when the particular information an individual wants to trace back to will either: not be there; be tailored to a different audience; or not be entirely suited to the purposes at hand.

[6] Our own introspection exercise, conducted throughout our investigations, demonstrated the difficulty in generating and documenting information that was of relevance to (and understandable by) others.

8.1. The need to locate and access pre-RS information sources

It was evident from our investigations that practitioners regularly encounter this described situation in practice. They all resort to the same fall-back strategy when they do. This fall-back strategy involves identifying and talking to those individuals who can fill in the missing details (i.e., those responsible for the information or work in question). A statistically significant finding was the agreement amongst practitioners that the most useful pieces of pre-RS information were: (a) the ultimate source of a requirement (i.e., the individual(s) whose requirement it is); and (b) who was involved in the various activities which led to its inclusion in the RS (i.e., the source(s) of any pre-RS work).

Regardless of major technological advances, this fall-back will always be desirable, and in many cases it will be essential. Even when suitable information is available, practitioners stressed that the ability to augment this with face-to-face communication is paramount[7]. To date, requirements traceability problems have been solely attacked with techniques that aim to supplant human contact with information. Pursuit of this objective disregards a fundamental working practice which we have found to underlie the continued citation of requirements traceability problems.

An implication of this finding is that both *eager* and *lazy* generation of project information is required for pre-RS traceability purposes. By eager, we mean the documentation of requirements-related information whilst actively engaged in aspects of RS production. Such information is often well suited to the immediate and short term needs of those involved and useful as a later reference point. With time, this static snapshot may be less suited to additional needs and is difficult to interrogate if it is the only information recorded. Information generated on need, by those originally responsible (i.e., lazily), can be provided with the benefit of hindsight and can be targeted to suit specific needs. Without reference to information recorded at the time, to regain some context, such information would be increasingly difficult to reproduce over time. Practitioners therefore require access to accurate information which informs them as to the human source(s) of any pre-RS information that is recorded.

8.2. Location and accessibility: the crux of the pre-RS traceability problem

What at first may appear as a straightforward requirement, the location and accessibility of pre-RS information sources, was found to be rather more problematic in practice. The inability to locate and access such sources was *the most commonly cited problem* across all the practitioners in our investigations. Furthermore, this inability was reported to be a direct and major contributor to the following problems also experienced:

- An out of date RS, as an RS evolves poorly (or not at all), when those originally responsible for it are not involved in its evolution.

- Slow realisation of change, as the most time-consuming aspect is often the identification of all those to involve in the change process.

- Deterioration as a result of change, as it is common to incorrectly identify all those that need to be informed of any change.

[7] This finding corroborates a growing awareness that some of the most vexing problems confronting requirements engineering are in fact social, political, and/or cultural in nature, and are not amenable to pure technical solutions [Goguen 1993].

- Inability to re-evaluate and refine existing work in a controlled manner, caused through the inability to re-access the context in which it was originally carried out.

- Unproductive conflict resolution, decision making and negotiation, because tools which are used to support these activities do not address the problem of locating those who should participate in the first place.

- Poor collaboration and co-ordination, as the invisibility of changing work structures and changing responsibilities, makes it difficult to: transfer information amongst appropriate parties; to integrate work; and to assign work to those with relevant knowledge and experience.

- Difficulty in dealing with those individuals who leave a project and with integrating new individuals into a project.

- Little reuse of requirements, and disaster when they are reused, as reuse is mainly practiced successfully when those responsible for their original production are either directly involved or readily accessible.

In some cases, this problem was found to be due to organisational or project politics, which actively prohibited any knowledge of, or access to, the original sources of requirements. It was also not uncommon for access to requirements engineers to be forbidden in later phases. These are political problems that can really only be addressed by re-examining the organisational policies of those projects they are experienced in. Another reason behind this problem was found to be the inability to keep track of the original sources and subsequent traces of participation in an exacting manner. A list of direct contributors to information hidden away in document fields (the common approach), was not felt sufficient. This problem calls out for more appropriate assistance.

In assisting this problem, it is important to note that certain project characteristics were found to actively promote it. In projects consisting of many individuals split into a number of teams, the location and accessing of the sources of pre-RS information was found to be either impossible, time consuming, or unreliable. The characteristics of these projects which led to this problem were: lack of shared or project-wide commitment, no visibility of ownership, and lack of accountability, so much information loss and appeal to the phrase "not invented here"; localised views, making it difficult to pin down the overall state of work or knowledge; little cross-involvement; poor communication and minimal distribution of information amongst teams; and changing notions of ownership and responsibility, due to continually changing work structures. Interestingly, these characteristics were amongst those identified elsewhere in our investigations as high contributors to project failure.

In contrast, projects consisting of few individuals did not find the ability to locate and access the sources of pre-RS information so much of a problem. Where there were no problems, this was attributed to: clear visibility of responsibilities and knowledge areas; clarity of working structures and work relations; individuals who acted as *common threads of involvement*; and a strong sense of team commitment and joint ownership. These characteristics were also identified as high contributors to project success. In facilitating the rapid location of and access to appropriate individuals, there are obvious benefits to be gained by ensuring that those project

characteristics which assist the ability to keep track of all contributors and their contributions are more widely experienced in all projects.

8.3. Related work

There are many traditional project management tools which provide facilities to model organisational charts and work breakdown structures. Some CASE tools now incorporate facilities to model such structures and to provide work-flow analysis (e.g., the ProKit WORKBENCH [Sodhi 1991]). However, these are not ideally suited to the above problem. They typically model formal and static organisational structures, and predefined work plans decomposed from an agreed RS. Such models presuppose rational and deterministic organisations and working practices. Their role is descriptive, prescriptive and/or predictive. They therefore primarily assist project management activities (i.e., to ensure that development goals can be achieved and costed, to schedule activities and resources, and to monitor their progress according to plan), so they tend to be centrally managed and relatively passive. The drift between what is modelled, what actually took place, and what is the case in later project life, can often be dramatic.

RS production and maintenance is an inherently social accomplishment, in which organisational structures and working relations are dynamically and perpetually created and recreated. The notions of ownership and responsibility are subject to continuous change and are therefore transient. The on-going ability to locate appropriate individuals deteriorates as both the volume and complexity of communication paths grow. To make this social process traceable, there is a need to be able to reflect these dynamics and manage this complexity.

Some models do attempt to account for the emergent properties of organisational structure and the dynamics of working practices (e.g., DesignNet [Liu & Horowitz 1989]). Although this model views a project as a hierarchy of tasks, and predefines a plan for carrying out these tasks, it does embed a dynamic view of activity and provide the ability to restructure these plans. There has also been much recent interest in modelling the organisational environment in which systems are to operate, mainly to obtain and clarify organisational requirements. These embed different views of an organisation and focus on specific organisational structures. For example, the intentional structure of an organisation [Yu 1993], or its responsibility structure [Strens & Dobson 1993]. Dynamic models like these would be useful for clarifying aspects of the organisational structure of development projects, though they singularly lack an appreciation of the wider organisational context. Recent research into process modelling is of interest here, as this aims to offer a means for understanding the full environment in which a system is developed (see [Lyytinen 1987, Mi & Scacchi 1990, Jarke *et al.* 1993]).

In a comprehensive analysis of the cause of software errors, recommendations were made for modularising responsibility and promoting informal communication [Lutz 1993]. Our investigations independently consolidate these recommendations, as they make it apparent that requirements traceability problems will continue to be evident where responsibility cannot be accurately located throughout the duration of a project, and where the appropriate individuals cannot be made accessible for informal communication. The ability to do this is compounded by the social nature of pre-RS work. Our current research is directed at providing the means to continuously reflect such information, investigating how to more actively assist the work of those being modelled, and mechanisms for instigating access and informal communication through its use.

8.4. Summary

Advances directed towards recording ever more requirements-related information must be augmented with the ability to rapidly and accurately locate and access those individuals in a position to supplement it. This is of crucial importance for information generated in the early and informal phases of a project (pre-RS), as that which is potentially relevant to assist later phases cannot be accurately or exhaustively predefined. To eliminate one of the most significant underlying causes of the requirements traceability problems that are experienced in practice, the challenge is to provide a continuously up to date picture which can actively promotes this activity. It is exactly the issues of this section that are the subject of our on-going research agenda.

9. Conclusions

We have illustrated the multifaceted nature of the so-called "requirements traceability problem" that many practitioners refer to, and have shown how it can only be tackled through improvements in many areas. In particular, we have highlighted the essential differences between pre-RS and post-RS traceability, and demonstrated why advances in the former are more critical for addressing these problems and how they hold the potential for more far-reaching and long-term opportunity. We have further discussed both the problems confronting, and the requirements for attaining, such advances. We have indicated where some of these requirements are already met and made suggestions for additional progress.

The intrinsic need for the on-going ability to rapidly locate and access those involved, particularly in pre-RS work, has been strongly motivated and empirically confirmed. In order to achieve any order of magnitude improvement with requirements traceability problems, there is a need to re-focus efforts on addressing the issues of pre-RS traceability. Fundamental to this re-orientation, the social infrastructure in which requirements are produced, specified, maintained, and used, needs to be more significantly recognised and explicitly supported.

Acknowledgements

Much of the work reported in this paper was carried out by the principle author whilst at the Centre for Requirements and Foundations, Oxford University, and was supported by a BT University Research Initiative. Both authors wish to acknowledge the comments and assistance of their colleagues and students. In particular, we would like to thank Marina Jirotka, Matthew Bickerton, Joseph Goguen, Daniel Berry, Jeff Kramer and Manny Lehman.

References

Bailin, S. C., Moore, J. M., Bentz, R. and 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, September.

Bersoff, E. H. and Davis, A. M. (1991). Impacts of Life Cycle Models on Software Configuration Management, *Communications of the ACM*, Volume 34, Number 8, August, pp. 104-118.

Bocker, H. D. and Herczeg, J. (1990). What Tracers are Made of, *ECOOP/OOPSLA '90 Proceedings*, October 21-25, pp. 89-99.

Bowen, J., O'Grady, P. and Smith, L. (1990). A Constraint Programming Language for Life-Cycle Engineering, *Artificial Intelligence in Engineering*, Volume 5, Number 4, pp. 206-220.

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

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

CACM. (1992). Information Filtering, *Communications of the ACM*, Volume 35, Number 12, December.

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

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

Conklin, J. and Begeman, M. L. (1988). gIBIS: A Hypertext Tool for Exploratory Policy Discussion, *ACM Transactions on Office Information Systems*, Volume 6, Number 4, October, pp. 303-331.

Cooke, J. and Stone, R. (1991). A Formal Development Framework and its Use to Manage Software Production, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 10/1.

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

Davis, C. G. and Vick, C. R. (1977). The Software Development System, *IEEE Transactions on Software Engineering*, Volume SE-3, Number 1, January, pp. 69-84.

Devanbu, P., Brachman, R. J., Selfridge, P. G. and Ballard, B. W. (1991). LaSSIE: A Knowledge-Based Software Information System, *Communications of the ACM*, Volume 34, Number 5, May, pp. 34-49.

Dorfman, M. and Flynn, R. F. (1984). Arts - An Automated Requirements Traceability System, *The Journal of Systems and Software*, Volume 4, pp. 63-74.

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

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. and Kung, C. H. (Eds.), Elsevier Science Publishers B. V., pp. 79-98.

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

Ehn, P. (1988). *Work-Oriented Design of Computer Artifacts*, Arbetslivscentrum, Stockholm.

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

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

Feather, M. S. (1991). Requirements Engineering: Getting Right from Wrong, in Van Lamsweerde, A. and Fugetta, A. (Eds.), *ESEC '91: 3rd European Software Engineering Conference*, Milan, Italy, October 21-24, Springer-Verlag, pp. 485-488.

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

Finkelstein, A. (1991b). Tracing Back *from* Requirements, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 7/1-7/2.

Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. and Goedicke, M. (1992). ViewPoints: A Framework for Integrating Multiple Perspectives in System Development,

International Journal of Software Engineering and Knowledge Engineering, Volume 2, Number 1, pp. 31-57.

Fischer, W. E. (1991). CASE Seen From Both Sides of the Fence, in Van Lamsweerde, A. and Fugetta, A. (Eds.), *ESEC '91: 3rd European Software Engineering Conference*, Milan, Italy, October 21-24, Springer-Verlag, pp. 509-511.

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

Freeman, P. (1987). A Conceptual Analysis of the DRACO Approach to Constructing Software Systems, *Transactions on Software Engineering*, IEEE, Volume SE-13, Number 7, July, pp. 830-843.

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

Goguen, J. A. (1993). Social Issues in Requirements Engineering, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, January 4-6, pp. 194-195.

Goguen, J. A. and Linde, C. (1993). Techniques for Requirements Elicitation, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, January 4-6, pp. 152-164.

Hamilton, V. L. and Beeby, M. L. (1991). Issues of Traceability in Integrating Tools, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 4/1-4/3.

Henderson-Sellers, B. and Edwards, J. M. (1990). The Object-Oriented Systems Life Cycle, *Communications of the ACM*, Volume 33, Number 9, September, pp. 142-159.

Hoffman, D. (1990). On Criteria for Module Interfaces, *IEEE Transactions on Software Engineering*, Volume 16, Number 5, May, pp. 537-542.

Hoffnagle, G. F. and Beregi, W. E. (1985). Automating the Software Development Process, *IBM Systems Journal*, Volume 24, Number 2, pp. 102-120.

IEE. (1991). *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2.

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

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

Jackson, J. (1991). A Keyphrase Based Traceability Scheme, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 2/1-2/4.

Jarke, M., Bubenko, J., Rolland, C., Sutcliffe, A. and Vassiliou, Y. (1993). Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis, *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, January 4-6, pp. 19-31.

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

Jirotka, M. (1991). *Ethnomethodology and Requirements Engineering*, Technical Report, Centre for Requirements and Foundations, Oxford University Computing Laboratory.

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

Johnson, W. L., Feather, M. S. and Harris, D. R. (1992). Representation and Presentation of Requirements Knowledge, *IEEE Transactions on Software Engineering*, Volume 18, Number 10, October, pp. 853-869.

Kaindl, H. (1993). The Missing Link in Requirements Engineering, *ACM SIGSOFT Software Engineering Notes*, Volume 18, Number 2, pp. 30-39.

Kaiser, G. E., Kaplan, S. M. and Micallef, J. (1987). Multiuser, Distributed Language-Based Environments, *IEEE Software*, November, pp. 58-67.

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

Keys, E. (1991). A Workbench Providing Traceability in Real-Time System Development, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 3/1-3/2.

Kuwana, E. and 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, January 4-6, pp. 273-276.

Langford, D. (1991). *PORC 0.41:Outline Description of Enhancements and Changes*, BT Internal Report, Ipswich, 28 June.

Lee, J. (1990). SIBYL: A Tool for Managing Group Design, *Proceedings of CSCW '90*, pp. 79-92.

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, January 4-6, pp. 82-89.

Liu, L. C. and Horowitz, E. (1989). A Formal Model for Software Project Management, *IEEE Transactions on Software Engineering*, Volume 15, Number 10, November, pp. 1280-1293.

Lowe, D. G. (1985). Co-operative Structuring of Information: The Representation of Reasoning and Debate, *International Journal of Man-Machine Studies*, Volume 23, pp. 97-111.

Lubars, M., Potts, C. and 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, January 4-6, pp. 2-14.

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, January 4-6, pp. 126-133.

Lyytinen, K. (1987). A Taxonomic Perspective of Information Systems Development: Theoretical Constructs and Recommendations, Boland, R. J. and Hirschheim, R. A. (Eds.) (1987). *Critical Issues in Information Systems Research*, John Wiley and Sons, pp. 3-41.

Madhavji, N. H. (1992). Environment Evolution: The Prism Model of Changes, *IEEE Transactions on Software Engineering*, Volume 18, Number 5, May, pp. 380-392.

Marca, D. A. (1989). Specifying Coordinators: Guidelines For Groupware Developers, Proceedings of the Fifth International Workshop on Software Specification and Design, *ACM SIGSOFT Software Engineering Notes*, Volume 14, Number 3, May, pp. 235-237.

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

Mathews, B. and Ryan, K. (1989). Requirements Specification Using Conceptual Graphs, *Third International Workshop on Computer-Aided Software Engineering*, London, July 17-21, pp. 186-193.

Mays, R. G., Orzech, L. S., Ciarfella, W. A. and Phillips, R. W. (1985). PDM: A Requirements Methodology for Software System Enhancements, *IBM Systems Journal*, Volume 24, Number 2, pp. 134-149.

Mi, P. and Scacchi, W. (1990). A Knowledge-Based Environment for Modeling and Simulating Software Engineering Processes, *IEEE Transactions on Knowledge and Data Engineering*, Volume 2, Number 3, September, pp. 283-294.

Neches, R., Swartout, W. R. and Moore, J. D. (1985). Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development, *IEEE Transactions on Software Engineering*, Volume SE-11, Number 11, November, pp. 1337-1351.

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

Parnas, D. L. and Clements, P. C. (1986). A Rational Design Process: How and Why to Fake It, *IEEE Transactions on Software Engineering*, Volume SE-12, Number 2, February, pp. 251-257.

Parnas, D. L., Clements, P. C. and Weiss, D. M. (1985). The Modular Structure of Complex Systems, *IEEE Transactions on Software Engineering*, Volume SE-11, Number 3, March, pp. 259-266.

Polack, A. J. (1990). Practical Applications of CASE Tools on DoD Projects, *ACM SIGSOFT Software Engineering Notes*, Volume 15, Number 1, January, pp. 73-78.

Ramamoorthy, C. V., Garg, V. and Prakash, A. (1986). Programming in the Large, *IEEE Transactions on Software Engineering*, Volume SE-12, Number 7, July, pp. 769-783.

Ramamoorthy, C. V., Garg, V. and Prakash, A. (1988). Support for Reusability in Genesis, *IEEE Transactions on Software Engineering*, SE-14, Number 7, July, pp. 1145-1153.

Ramamoorthy, C. V., Prakash, A., Tsai, W. T. and Usuda, Y. (1984). Software Engineering: Problems and Perspectives, *IEEE Computer*, October, pp. 191-209.

Ramesh, B. and Dhar, V. (1992). Supporting Systems Development by Capturing Deliberations During Requirements Engineering, *IEEE Transactions on Software Engineering*, Volume 18, Number 6, June, pp. 498-510.

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

Rein, G. L. and Ellis, C. A. (1991). rIBIS: A Real-Time Group Hypertext System, *International Journal of Man-Machine Studies*, Volume 34, pp. 349-367.

Reubenstein, H. B. and Waters, R. C. (1991). The Requirements Apprentice: Automated Assistance for Requirements Acquisition, *IEEE Transactions on Software Engineering*, Volume SE-17, Number 3, March, pp. 226-240.

Robinson, D. (1991). CASE Support for Large Systems, in Van Lamsweerde, A. and Fugetta, A. (Eds.), *ESEC '91: 3rd European Software Engineering Conference*, Milan, Italy, October 21-24, Springer-Verlag, pp. 504-508.

Roman, G. C. (1985). A Taxonomy of Current Issues in Requirements Engineering, *COMPUTER*, April, pp. 14-23.

Rzepka, W. and Ohno, Y. (1985). Requirements Engineering Environments: Software Tools for Modeling User Needs, *IEEE Computer*, April, pp. 9-12.

Short, R. M. C. (1988). Learning the First Step in Requirements Specification, *Quaestiones Informaticae*, Volume 6, Number 3, November, pp. 123-128.

Smithers, T., Tang, M. X. and Tomes, N. (1991). The Maintenance of Design History in AI-Based Design, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 8/1-8/3.

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

Star, S. L. (1989). The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving, in Gasser, L. and Huhns, M. (Eds.), *Distributed Artificial Intelligence*, Volume 2, Pitman, pp. 37-54.

Strens, R. and Dobson, J. (1992). *On the Modelling of Responsibilities*, Computing Laboratory, University of Newcastle Upon Tyne, Newcastle, U.K.

Sun Microsystems, Inc. (1990). *Catalyst: A Catalog of International Third-Party Hardware and Software from Sun Microsystems, Inc.*, Summer 1990 Edition

Sykes, J. B. (Ed.) (1978). *The Pocket Oxford Dictionary*, Sixth Edition, Oxford University Press.

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

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

Thayer, R. H. and Dorfman, M. (1990). *System and Software Requirements Engineering*, IEEE Computer Society Press Tutorial.

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

U.S. Department of Defense. (1988b). *Software Requirements Specification*. DI-MCCR-80025A. Washington, D. C., February 29.

Walker, R. F., Oskamp, A., Schrickx, J. A., Van Opdorp, G. J. and Van Den Berg, P. H. (1991). PROLEXS: Creating Law and Order in a Heterogeneous Domain, *International Journal of Man-Machine Studies*, Volume 35, pp. 35-67.

Ward, A. (1992). The Next Generation of Computer Assistance for Software Engineering, *ACM SIGSOFT Software Engineering Notes*, Volume 17, Number 3, pp. 39-42.

West, M. (1991). The Use of Quality Function Deployment in Software Development, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 5/1-5/7.

Wright, S. (1991). Requirements Traceability - What? Why? and How?, *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), Digest Number: 1991/180, December 2, pp. 1/1-1/2.

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