Requirements Traceability

Main Report

Author: Orlena C. Z. Gotel
Date: 18th December 1992

Centre for Requirements and Foundations
Oxford University Computing Laboratory
11, Keble Road
Oxford
OX1 3QD

Telephone: (0865) 272568
Fax: (0865) 273839
E-Mail: Orlena.Gotel@prg.oxford.ac.uk

[The research reported here has been supported by contracts from the BT University Research Initiative in *Requirements Capture and Analysis*.]

Contents

Note to the reader

There is an accompanying set of appendices, the contents of which are listed at the end of this report. They consolidate the work reported here and provide considerably more detail. Due to their size, they are not being distributed as a matter of course, but they can be obtained on request.

Executive summary

This report is the result of an investigation into "requirements traceability" for BT, as part of BT's University Research Initiative with Oxford University.

Traceability has been identified many times (in BT and in other organisations) as a priority problem area in requirements engineering. In response, this contract was placed as part of BT's continuing University Research Initiative with Oxford. The goals were:

- To discover what are the actual, detailed problems encountered.
- To suggest whole or partial solutions.
- To examine the quality of support offered by IEF and RTM support tools (both used within BT) and suggest areas where they could be improved.

Key results

- Traceability problems in BT are not uniform - different people/projects mean different things when they say they have a traceability problem.
- A large number of individual problem areas have been identified which sometimes cause "traceability problems" - including issues of project organisation, clerical support issues, change control and configuring support systems to fit individual project practices.
- A variety of suggestions are given for improvements against the detailed individual problems.
- There are two distinct phases of requirements traceability:
    - Pre-RS: From initial information gathering (interviews, documents, workshops, etc.) to a written requirements specification (RS).
    - Post-RS: From a written requirements specification to design, code, etc.
- Post-RS traceability is widely recognised as an issue, but pre-RS traceability is not; available tool support is predominantly for post-RS traceability.
- Pre-RS traceability has the potential for much greater leverage than post-RS traceability on costs and quality of software produced.
- IEF and RTM both offer some support for post-RS traceability; suggestions are given for how their support could be significantly improved. Some suggestions concern how the existing tools are used within projects; others concern improvements that the suppliers would need to make.

The report is being distributed to BT people responsible for improving requirements capture in BT (particularly members of the RCA Topic Group), and to other BT people who are engaged in requirements capture and are interested in how to improve their practice. Some of the suggested improvements can be put into action quickly - others can be used in discussions with tool suppliers for improved support tools.

1. Introduction

1.1. Terms of reference

On the 1st January 1992, the Centre for Requirements and Foundations at Oxford University Computing Laboratory began a one year project on the subject of "requirements traceability", as part of the BT University Research Initiative in Requirements Capture and Analysis[1]. This report constitutes the final deliverable of that project.

1.2. Intended readership

The intended readers of this report are:

- BT managers and professionals responsible for improving requirements capture in BT.
- Other BT managers and professionals engaged in requirements capture and who are interested in how they could improve their practice.

1.3. Objectives and scope

This report describes requirements to improve the traceability[2] of the information generated and manipulated in producing a requirements specification (RS). It does this by examining:

- Key problems and issues underlying such traceability.
- Requirements to address these underlying problems and issues.

The findings documented here were obtained from a number of investigations[3]. This report further critiques the coverage of the identified requirements by current tools and techniques claiming to support requirements traceability. Particular emphasis is placed upon those within the BT CASE Tools Portfolio, such as Requirements Traceability Manager[4] (*RTM*) and Information Engineering Facility[5] (*IEF*). Solution options and techniques are suggested for those requirements which are insufficiently supported at present. Recommendations are made concerning uptake and further research.

1.4. Motivation

This project was initiated following repeated recognition over the years within BT/D&P of an industry-wide problem with lack of (or inadequate) requirements traceability in systems and software development projects. The existence of this problem, typical consequences of it and what the ideal state of affairs should be, have been extensively documented by both the software and academic communities [1-17].

[1]The original proposal for this project can be found in appendix A.
[2]In so doing, improvements in *modularity*, *accessibility* and *reusability* are also addressed.
[3]The requirements investigations that were carried out are outlined in appendix B.
[4]RTM is a product of Marconi Systems Technology, a division within GEC-Marconi.
[5]IEF is a trademark of Texas Instruments Incorporated.

2. Requirements traceability

2.1. Definition

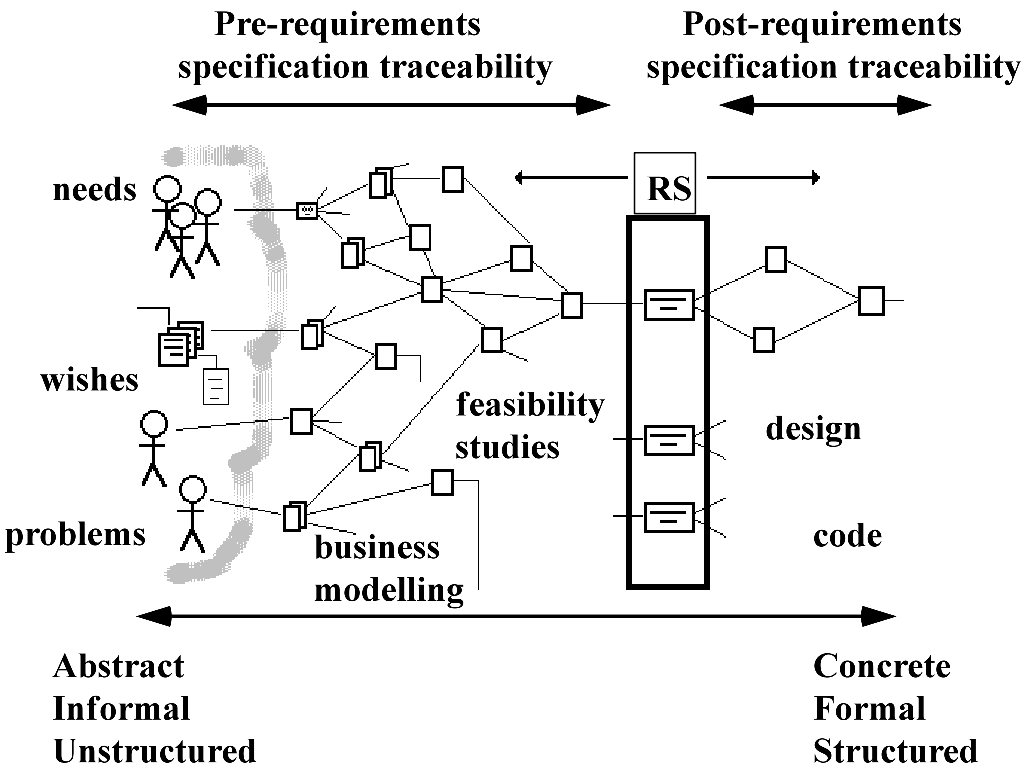
"A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation" [ANSI/IEEE Standard 830-1984].

2.2. Types of requirements traceability

The above definition suggests that requirements traceability itself can be divided into two basic types, revolving around the written statement of requirements (RS). This is illustrated in figure 1. We refer to these as:

- *Pre-requirements specification traceability* (pre-RS traceability).

- *Post-requirements specification traceability* (post-RS traceability).



(Figure 1: The two basic types of requirements traceability.)

2.2.1. Post-RS traceability

- Forwards: Demonstrates how (and which) design and code modules satisfy individual requirements in the RS.

- Backwards: Demonstrates which requirements in the RS the individual design and code modules help satisfy (why these modules exist, where they came from and how they were derived).

2.2.2. Pre-RS traceability

- Forwards: Demonstrates how (and which) requirements in the RS satisfy individual needs.

- Backwards: Demonstrates which needs the individual requirements in the RS help satisfy (why these requirements exist, where they came from and how they were derived). I.e., the source, context and environment from which requirements were drawn, their life-cycles and their interrelationship with other requirements.

2.3. The type of requirements traceability addressed

This project is concerned with improvements in pre-RS traceability for a number of reasons[6]:

- Improvements in post-RS traceability have already been attained. Such traceability can be established using various:

  - *Techniques* (such as: requirements traceability matrices [18-19]; matrix sequences [20]; indexing, numbering and cross referencing schemes [21-22]; keyphrase dependencies [23]; templates [24]; note tagging [25]; assumption-based truth maintenance [26]; and constraint networks [27]).
  - *Methods* (such as: strict top down design [28]; formal transformations [29-30]; object-orientation [31]; Software Requirements Engineering Methodology [32]; and Quality Function Deployment [33]).
  - *Models* (such as: entity-relationship-attribute models [34-36]; process models [37]; and requirements sets [38]).
  - *Languages* (such as: Requirements Statement Language [39]).
  - *Tools* (such as: RTM [40]; IEF [41]; Teamwork/RqT [42]; AGE [43]; Requirements Engineering and Validation System [44]; Software Life-Cycle Support Environment [45]; Statemate [46]; and HUFIT Functionality Matrix [47]).

- Techniques, methods, models, languages and tools to enable post-RS traceability are not directly applicable to pre-RS traceability:

  - They tend to impose premature commitment to structure and detail [48-52].
  - They provide little support for the on-going and emergent nature of the work practices involved in producing an RS [53-56].
  - They have trouble accounting for the dynamic and changing nature of the sources and environment from which requirements are drawn [57-58].
  - They deal with the traceability of markedly different types of information [59-67].
  - The problems they aim to address differ in emphasis [68-73].

- The need for improvements in the ability to trace the information generated in producing an RS is a recent and growing concern [74-83]. Amongst the arguments driving this need, the most prevalent are:

  (i) To **improve quality**, by assisting with auditing and changing requirements[7] [84-89]. This is in recognition that:

[6]These are expanded upon in appendix H.

[7]This has become a particular concern with the increasing popularity of *iterative*, *prototyping* and *evolutionary* approaches to development.

- Change cannot be adequately handled from the RS alone, as the RS is only the end product of a complex process (so is a "black box").
- Change needs to be instigated and propagated from its source to inform what exactly in the RS needs changing.
- Subtle interrelationships that exist between requirements need to be made visible.
- Previously closed issues (even decisions concerning how to conduct the requirements exercise itself) need to be made explicit and possible to re-open.

(ii) To achieve **greater economic leverage**, as a significant proportion of cost, time and effort is currently spent reconstructing and rediscovering an (often erroneous) understanding of what went on in producing a RS, in order to make use of it and to maintain it [90-100].

3. Key problems and issues underlying pre-RS traceability

3.1. Problem analysis

The problem being addressed is lack of (or inadequate) pre-RS traceability. Despite a number of recent advances [101-112], the problem still exists. This is due to the absence of any thorough investigation of what the problem **actually** is (the underlying problems and issues which **cause** pre-RS traceability to be perceived as problematic). Without such an understanding, it is difficult to know: (i) exactly what is meant by improvements; and (ii) how to realise them.

The analysis undertaken in this project used focus and discussion groups, questionnaires (all within BT) and reflection on requirements activities[8].

3.2. Summary

The so-called "traceability problem" is multifaceted. It can be decomposed into (and is hence a consequence of) a number of underlying problems and issues which affect the ability to provide information that can be traced and the ability to actually trace this information. The existence of these multiple problems and issues stem from the different (and continuously changing) viewpoints regarding what traceability is and what it is meant to achieve. Underlying every instance in which traceability is required, different user, task and informational requirements come into play.

These findings suggest that traceability techniques could themselves be quite straight forward. The real difficulty in providing support for pre-RS traceability lies in the ability to address the multitude of underlying problems and issues. Each deserves in-depth research in order to investigate how they could be addressed. There is unlikely to be a single (unique and overarching) solution to the so-called "traceability problem". Instead, there will be the need for different approaches to traceability and different ways for providing and handling the underlying information, dependent on the salient characteristics of individual situations.

3.3. Fundamental problems and issues[9]

- **Lack consensus of definition.**
  There is little agreement regarding what requirements traceability[10] is; individual understanding of what the requirements traceability problem is; and various perspectives as to what requirements traceability should achieve and assist with.

- **Solution-driven emphasis.**
  There is a wide belief that pre-RS traceability is the solution to numerous and multifaceted problems. It cannot solve all these problems; it is not necessarily the optimal or appropriate solution to many. Traceability is a technique, so merely provides the mechanics to filter and access required information[11]. Pre-RS traceability <u>itself</u> is NOT the fundamental problem.

- **The establish and end-use conflict.**
  The two main parties involved (those responsible for making traceability possible and

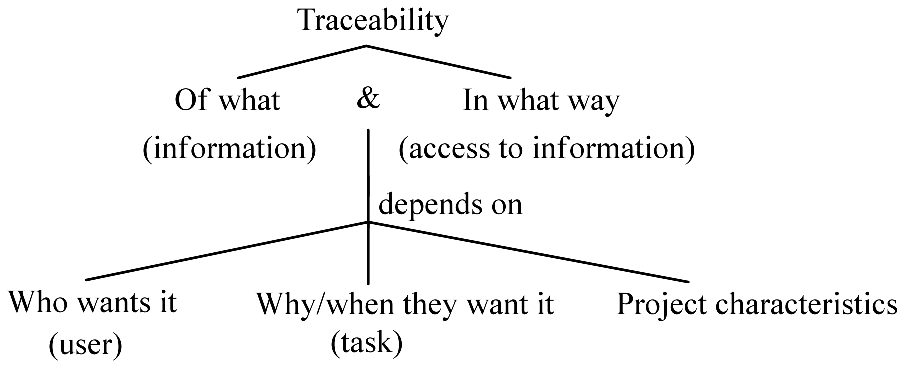
[8]These studies, and their findings, are detailed in appendices C through to E.
[9]Each of the points listed in this chapter are justified and expanded upon in appendix F.
[10]Be this either pre-RS or post-RS traceability.
[11]In the case of pre-RS traceability, the required information is that which pertains to the production of an RS.

those that make subsequent use of this facility) have conflicting problems and needs. Improvements made for one of these parties causes problems for the other. This is highlighted in the situation where an individual takes on both of these roles.

3.4. Problems and issues that underlie its end-use

- **Number and variety of end-users.**
  There is no stereotypical end-user. There are conflicting requirements for end-use amongst the various end-users and even for specific individuals.

- **Unique nature and requirements of each use situation.**
  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 (as illustrated in figure 2).



(Figure 2: Deconstructing the requirements traceability problem for use.)

- **Unavailability of information to trace.**
  Caused by the quantity and heterogeneity of information required to be traced and the inability to pre-define this. Personal contact is relied upon because required information is typically undocumented, inaccessible, out of date, or unsuited to use for which it is required.

- **Rigid and pre-defined information access.**
  Access to information is required in a diversity of ways and to varying extents. All potential combinations cannot be pre-defined. Information available to be traced is typically traceable in standard (non-purposeful) ways and in limited combinations.

3.5. Problems and issues that underlie establishing it

- **Given low priority by those in a position to resource it.**
  Pre-RS traceability is perceived as an optional extra, post-RS traceability being of higher priority, so insufficient allocation of time, personnel and resources.

- **A shared and distributed responsibility by all RS contributors and end-users.**
  Usually, pre-RS traceability is not a single individual's job. There is little awareness, allocation and management of the different roles individuals need to play to achieve three interdependent tasks (obtaining and documenting required information, organising it, and

maintaining it). Individual efforts are typically ad hoc, localised and uncoordinated. There is an imbalance between the extra work involved and the personal benefits gained.

- **Accounting for the unique nature of end-use.**
  There is a lack of understanding of problems and issues underlying and driving end-use (diverse requirements). Tendency to focus on immediate needs and short term end-use.

- **Accounting for the unique nature of RS production.**
  There is a presumption that an RS is produced in a standard manner, with diminishing concern for traceability after the first snap-shot. RS production is a social and on-going activity, dependent on the characteristics of the project concerned, so concern for traceability must continue throughout a project's life.

- **Obtaining and documenting information required to be traceable (content).**
  Required information cannot always be obtained and documented. Much is not readily verbalisable, difficult to linearise, and distracting and costly to obtain. Quality depends on: the characteristics of the project; the characteristics of those individuals in a position to provide the information and those documenting it; the ability to delimit a potentially huge information space and identify relevance; and the amount and length of any delays in documenting information. A deliverable-driven culture actively discourages and precludes capturing certain information.

- **Organising documented information so it is traceable (structure).**
  Documentation of required information does not imply it is traceable. There are problems in: accounting for all the ways in which information is required to be traceable in a project; aligning multiple contributions (of various type, levels of detail and granularity, etc.); and in dealing with different categorisation schemes. Typically, there is a premature commitment to syntactic, rather than to semantic structure.

- **Maintaining information content and structure.**
  Information may be structured so that it is traceable, but this does not ensure it is up to date and representative. There are problems in accounting for all possible kinds of change to information, due to: the lack of change culture (so inadequate support, co-ordination and distribution of updates); characteristics of the project, information and individuals involved; and the cyclic dependency upon traceability itself.

- **Little dedicated clerical, procedural, or computer support.**

- **No transfer of best practice.**

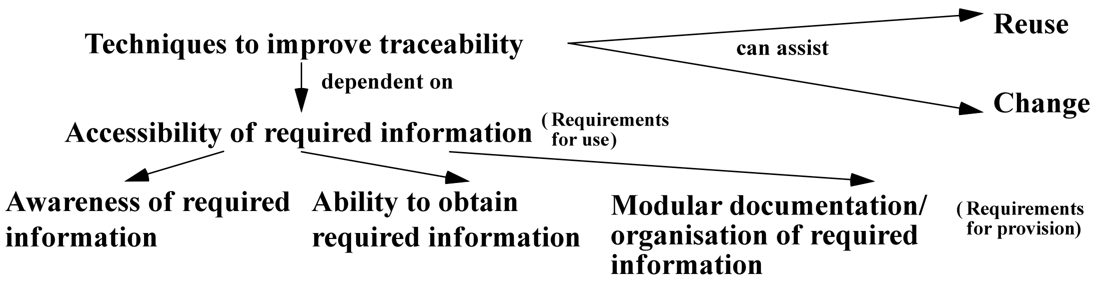
4. Key requirements for pre-RS traceability

4.1. Requirements analysis

Having analysed the problem (as summarised in chapter 3), we are in a more informed position to investigate what is required in order to provide and make use of pre-RS traceability. This chapter outlines the main requirements (suggested from the major problems and issues) at a very high level[12]. Since it has not been possible to investigate how each of the problems and issues underlying pre-RS traceability can be addressed (in an exhaustive manner), these requirements must be read in the light of the information contained in chapter 3.

4.2. Summary

Figure 3 provides a concise summary of the key requirements:


Techniques to improve traceability
can assist
Reuse
Change
dependent on
Accessibility of required information
(Requirements for use)
Awareness of required information
Ability to obtain required information
Modular documentation/ organisation of required information
(Requirements for provision)

(Figure 3: Summary of key requirements for pre-RS traceability.)

4.3. Fundamental requirements

What is required in order to provide and make use of pre-RS traceability:

Problems	Requirements
1. Lack of "buy-in".	1.1 Education in the need for/importance of pre-RS traceability (reduced costs). 1.2 Project time dedicated to providing pre-RS traceability. 1.3 Assistance for routine and mundane aspects (clerical or tool support to reduce effort). 1.4 Up-front commitment from all individuals involved. 1.5 Benefits and incentives for all individuals involved in a project.
2. Not a dedicated job.	2.1 How pre-RS traceability is to be achieved needs to be established early on (joint or individual efforts). 2.2 Explicit organisational structure, work roles and responsibilities. 2.3 Provision for continuity and common threads of involvement (e.g., facilitators, documentation control centres, etc.)

[12]Appendix G contains the full and expanded requirements specification, which pays attention to functional, operational, environmental (etc.) requirements in considerable detail.

Problems	Requirements
3. Uniqueness of each situation in which pre-RS traceability is to be provided.	3.1 Support for flexible working practices (constant re-organisation and re-definition of information). 3.2 Method independence (ability to provide pre-RS traceability not dependent on having followed pre-defined methods). 3.3 Support for co-ordination of collaborative and co-operative aspects of work. 3.4 Support for information sharing and distribution (through documentation). 3.5 Ability to capture details about the conditions of production in documentation. 3.6 Visibility of document ownership.
4. Uniqueness of each situation in which pre-RS traceability is to be used.	4.1 Awareness by all involved of requirements for use (e.g., all end-users' requirements, different types of pre-RS traceability, etc.) 4.2 Support for the requirements imposed by different individual, project and task characteristics. 4.3 Ability to identify and account for the circumstance of use (for obtaining and presenting traces).

4.4. Requirements for use

What is required in order to make use of pre-RS traceability (to trace information):

Problems	Requirements
5. Restricted use (types of possible trace are pre-defined).	5.1 Ability to support all potentialities (types of pre-RS traceability required). 5.2 Flexibility of access to documented information. 5.3 Ability to filter documented information in different ways. 5.4 Ability to dynamically create traces on-the-fly.
6. User is unaware of the possibilities.	6.1 Visibility of documented information that can be traced. 6.2 Visibility of pre-defined traces. 6.3 Visibility of ways in which user-defined traces can be requested and invoked.
7. Problematic to request types of pre-RS traceability required.	7.1 Flexibility in the manner in which pre-RS traceability can be requested and invoked. 7.2 Assistance in setting up views, filters and accessing documented information.
8. Documented information traced is unusable or unsuited to needs.	8.1 Rapid visibility of documented information that is traced, in user-definable manners. 8.2 Ability to represent the same information in multiple ways and concurrently.

4.5. Requirements for provision

(i) What is required to obtain and document information that users want to be able to trace:

Problems	Requirements
9. Large amounts of information produced.	9.1 Awareness of information relevance and priorities (e.g., whether crucial, desirable, superfluous, etc.)

Problems	Requirements
10. Not all information required is made explicit.	10.1 Support to make more information explicit. 10.2 Emphasis on open communication, improved visibility and clarity of detail. 10.3 Production of information as a by-product of working practices (automated or assisted, by clerical or tool support).
11.Unrepresentative information.	11.1 Ability to obtain more objective and detailed information. 11.2 Assistance for immediate production of information whilst 'ready-at-hand' or whilst experiences are still recountable.

(ii) What is required to document information obtained (individual fragments) for traceability purposes:

Problems	Requirements
12. Loss of information or change in emphasis.	12.1 Ability to directly document information obtained in original form (e.g., informal, unstructured, incomplete, different languages and media, etc.) 12.2 Ability to directly document information obtained at the time it is produced.
13. Pre-defined documentation formats designed for specific types of information.	13.1 Flexibility of documentation format (so the ability to provide pre-RS traceability is not dependent on having followed methods and having produced information to fit pre-defined documentation strategies).
14. Internal information content of documents is not accessible.	14.1 Ability to define information content of documents through well-defined document interfaces. 14.2 Ability to manage and make information content accessible through document interface control.
15. Documented information is not designed to be traced.	15.1 Ability to define inter-linkages between documents (for traceability purposes) through document interfaces. 15.2 Ability to trace information using document interface control mechanisms.

(iii) What is required to collectively structure (organise) individual fragments of documented information for traceability purposes (so able to manage it all in the face of continual and on-going change):

Problems	Requirements
16. Individual documents are fragmented (so lack global visibility).	16.1 Central availability (as far as possible) of all documents produced. 16.2 Early and on-going visibility of documents produced and document interfaces. 16.3 Provision of localised and global views of collective documents (ability to see the whole picture).
17. Individual documents are incompatible (of different types, levels of granularity, with overlaps, duplications, inconsistencies, etc.) so hard to structure.	17.1 Support to "polish up" documents into a usable and reusable form (i.e., generic, abstract and traceable) and compartmentalise as modular viable systems (modules/nodes). 17.2 Assistance with integration, alignment, levelling, composition, decomposition, partitioning, merging, etc. of documents. 17.3 Support definition and management of details on a project-specific basis (e.g., whether duplication is to be allowed, coped with, or eliminated, levels of granularity to use, etc.)

Problems	Requirements
18. Uncoordinated structuring (organisation).	18.1 Assistance (i.e., roles, automation, principles, etc.) for handling organisation and re-organisation. 18.2 Visibility of documents, document interfaces and structure(s) being defined.
19. Distributed nature of information across documents, so hard to change information and to manage the effects of change.	19.1 Support the formulation of tightly cohesive modules/nodes from modularised documents (to control duplication, localise detail, etc.) 19.2 Establish well-defined module/node interfaces (defining attributes, such as changeable and non-changeable parts). 19.3 Ability to change or attach new attributes to the interface (so change controlled by changing attributes). 19.4 Support for multiple and flexible module/node types.
20. Rigidity of structure (so information is accessible in set ways, providing single views and unwieldy to change).	20.1 Flexibility to enable information to be rapidly structured and re-structured in multiple ways (to enable different types of traceability, views and filters), through flexible definition of how the individual modules/nodes interact (ability to establish links and link management procedures in the interface). 20.2 Support for loose coupling of modules/nodes. 20.3 Support for multiple and flexible link types (to enable new and alternative structures). 20.4 Ability to define set paths through the information, between individual modules/nodes (primary and secondary links of a syntactic, semantic or pragmatic nature) and also the ability to dynamically define paths (tertiary links of a user-definable nature).
21. Changes too slow and not visible (acting on out of date information).	21.1 Assistance for rapid change (i.e., automatic handling of change details, explicit monitoring for change, rather than searching the entire information, etc.) 21.2 Ability to generate change documentation. 21.3 Immediate distribution of change and change documentation to all involved. 21.4 Regular conformance checks and quality assurance reviews (i.e., information is up to date, required information is present, traceability is possible, etc.)
22. Undesirable/ unidentifiable repercussions of change.	22.1 Ability to trial and explore change (visual demonstrations). 22.2 Ability to identify forward and backward impact. 22.3 Ability to undo. 22.4 Information persistence.

5. Current coverage of pre-RS traceability requirements

5.1. Analysis of the current situation

To assess the extent of coverage for the pre-RS traceability requirements, a spectrum of current tools and techniques were reviewed. This involved looking at both commercial tools and research products, including many not explicitly dealing with requirements support, but offering strength in particular areas of concern[13]. Particular concentration was paid to the strengths and limitations of IEF (a fully integrated software development life-cycle toolset) and RTM (a front end pre-processor for life-cycle-wide requirements traceability and management). These were assessed based upon documentary sources [113-129] and following hands-on experience and discussion with tool users[14]. This chapter outlines some key points[15].

5.2. Summary

Although there are growing numbers of specialised tools to support requirements engineering activities, the majority do not cover requirements traceability, and even fewer provide support for the particular post-RS traceability requirements now enforced by DOD STD-2167A [130]. Those that do, tend to do so by tagging objects in a central repository with an attribute for cross-referencing purposes and tend to be stand-alone tools (so of limited use to the full life-cycle). The different tools differ in cosmetics rather than underlying mechanics for requirements traceability.

No single tool supports all the requirements identified for pre-RS traceability. Evident in the few dedicated requirements traceability tools is a poor understanding of the traceability process and little support for the existing working practices of requirements engineers. However, there is evidence from appendix H that most of the individual pieces are there, so the combination of their strengths is an area which needs to be exploited.

5.3. Coverage by general tools and techniques

5.3.1. Types of traceable information

All but the most general purpose software tools pre-define what type of information is to be made traceable. As tools provide more life-cycle coverage, more attention is paid to post-RS traceability. Single CASE tools support pre-RS activities of a focused nature and establish the traceability of specific information, with limited duration and extent of use. A number of such tools would be required to capture representative pre-RS information, but these tend not to be incorporated in CASE workbenches or environments, as their underlying information bases are typically unsuited to representing such informal information.

5.3.2. Support for establishing traceability

Establishing traceability with general purpose software tools can involve considerable effort, as they provide no explicit support. With single CASE tools or analyst CASE workbenches, it depends on the importance they attach to traceability activities: where traceability is the

[13]Appendix H documents the full literature critique. It provides more detail on all the tools and techniques reviewed, discusses the strengths and limitations of particular categories of tool (partitioned into general purpose software tools, individual CASE tools, CASE workbenches and CASE environments), identifies those tools and techniques which provide good support for particular requirements and steps through the extent of support for each of the requirements listed in chapter 4.

[14]Appendix I provides an in depth review of IEF and RTM and also assesses their coverage of support for the expanded requirements specification of appendix G.

[15]The interested reader is strongly referred to appendices H and I.

main concern, all work steps are centred around ensuring traceability is possible, and assistance is mainly in the form of automated linking between hierarchically decomposed requirements; where it is a peripheral concern, traceability is merely a spin-off from other activities. In CASE environments, traceability tends to be encompassed even more so within other developmental activities, often automatically and basically provided through adherence to the constraints of an underlying development method or language, else provided by a dedicated tool of the above type. Most support is clerical and cosmetic. The best support for traceability appears to be in those tools which explicitly introduce dedicated job roles for it.

5.3.3. Tasks and individuals for which traceability is established

Traceability can be established to suit any purpose and individual with general purpose software tools. With individual CASE tools, it is established to assist the specific task the tool supports (e.g., traceability between transcripts and analyses for interview analysis tools). CASE workbenches tend to focus on traceability to assist requirements engineers in producing an RS, to inform development work and to manage requirements. CASE environments tend to focus on traceability for testing, maintenance and managerial activities (hence the emphasis on post-RS traceability rather than support for, and traceability of, RS production).

5.3.4. On-going provision of traceability with changing requirements

This is unwieldy with general purpose software tools, as traceability is typically set up to address immediate needs, with little consideration of wider concerns. CASE tools are best used to establish traceability for a short duration, since the results do not always filter through to later phases. Horizontally integrated CASE workbenches provide strong on-going support up to the production of the RS, but not always once the RS becomes used to drive subsequent stages. Those CASE workbenches which are vertically integrated, or included within CASE environments, best support on-going and iterative changes to requirements and assist their continued traceability, since they are better able to assess and control impact.

5.3.5. Accounting for unique needs of organisations, projects, individuals and problems

Customisation is best with general purpose software tools and open and integrated CASE environments. With the former, this can involve much effort to configure the tool to do what is required. With the latter, this can require expertise to identify which tools to use, in which order and in which combination. They both focus on tool usage rather than on the underlying activities. Tailorability is more limited within individual CASE tools or dedicated CASE workbenches. The ability to support the traceability of flexible types of information is dependent on the sophistication of the underlying information base. More promise is shown with recent advances in dynamic and distributed information base managers (particularly object-oriented ones) rather than with relatively static table-based and relational ones.

5.3.6. Accounting for group work in producing an RS

Support for traceability is best provided within CASE workbenches and environments with underlying information bases that can be configured to support group activities. Software tools and individual CASE tools primarily support individualistic working, as do the majority of the dedicated CASE workbenches for requirements traceability. However, as group work becomes better supported, the control over traceability tends to become less meticulous.

5.4. Coverage by IEF[16]

5.4.1. Strengths

- Method companion, allowing some customisation in tool use and task sequence. It provides continuity through a guiding meta-model and standardisation (if there is total user commitment).
- Provision of an infrastructure and central encyclopedia facilities to support team working.
- Use of diagrams to represent the end products of analysis. These promote guidelines for managing complexity, check for required information and enhance communication.
- Horizontal, vertical and business-wide traceability, through technique integration, model transformations and repository integration, respectively.
- Support for changing requirements, at a very high level of abstraction, through lack of duplication, information sharing and automated retransformation of models.
- Reasonable forwards traceability to code (the data model goes further in this than the activity model) and some backwards traceability (dependent on tools and techniques used).
- IEF gears established traceability towards project management. It also supports the creation of model families and the reuse of business models across an organisation.

5.4.2. Limitations

- "Cutting and pasting" techniques leads to potentially dangerous drifts from IEM, impoverishing traceability. Administrative overheads are high for maintaining control.
- A steep learning curve (much steeper than RTM) results in individuals only obtaining expertise with particular toolsets, leading to little continuity. This encourages separate working, localised visibility, poor handover between toolsets and little support for iteration or retrospective analysis. Traceability deteriorates with the number of groups working on poorly partitioned models within and across stages.
- The complexity of the tool can encourage off-line working, so used as an after-the-event documentation tool to store the end results, resulting in information loss, the omission of intermediate stages and less representative traceability.
- Does not support all of IEM (particularly in planning and analysis), so lacks traceability back to business plans and much of the information produced in ISP and BAA, resulting in narrow and inward-looking traceability and problems with reuse. Not all of the information produced is used throughout the life-cycle, resulting in backwards traceability gaps (evident in the evolution of the activity model).
- Only information contained in diagrams is traceable (functional as opposed to non-functional requirements). Any information produced is prematurely committed to pre-defined diagrammatic structures. Lack of integration to front end tools, supporting informal planning activities and requirements exploration (etc.), prevents traces of process or their fabrication. There is no ability to interrogate, reassess, clarify or assist change.
- Limited version control and configuration management, only possible at the model level, results in the maintenance of restricted life histories of limited entities (with early information and interconnections lacking) and the inability to pursue parallel lines of work.
- Restrictions on the types of possible change. Change requires full upheavals and are less reliable dependent on the level of model abstraction, the amount of hand crafting carried out, and the amount of prior checking. Trials and basic impact analysis require either the creation of dual models or unrealistic vision (as prone to corrupting the model).
- Lacks an explicit RS: requirements are fragmented in the model, divorced from context.

[16]Each of these points is expanded in appendix I. IEM and its stages (BSP, ISP, BAA, BSD...) are described.

5.5. Coverage by RTM[17]

5.5.1. Strengths

- Inbuilt traceability of the evolution of requirements objects (maintained in a requirements hierarchy) and generic traceability to connect up to later developmental objects.
- Configurability of traceability schemes (i.e., the type of information to make traceable, its tightness, its granularity and its extent).
- Supports engineering requirements from multiple and diverse sources and traceability back to these, to see engineered requirements (and appended queries) in their original context.
- Supports some flexibility of requirement content and RS production. Structure can be progressively imposed on requirements and qualifying details appended when appropriate.
- Promotes the recording of diverse requirements and project information, according to needs, and highly granular traceability amongst requirements objects.
- Good version control enables safety in use and the traceability of past information. This provides the ability to record individual life histories for requirements objects.
- Provides alternative mechanisms to automatically retrieve and display the stored traces.

5.5.2. Limitations

- The process involves many repetitive and mundane activities, introducing potential for corner-cutting and error. It requires RE to be carried out linearly and RS production to be documented as a systematic process of extraction, expansion, focusing and replacement (often resulting in some loss of intermediate flow down). Traceability is therefore of the historical (and not the logical) production of requirements.
- Juggling the disjoint tools further distracts from carrying out RE tasks coherently and encourages batch modes of working. This leads to delays in adding information and the creation of ad hoc links, resulting in the loss of (and less reliable) information and traces.
- No true multi-user support. Complicated by: co-ordination problems, poor complexity management, lack of global or concurrent visibility (reliance on memory) and an uninformative requirements identification scheme. Non-visibility of siblings and peers further impedes the ability to pursue separate lines of progress with a single requirement.
- Focuses on forwards working, with traceability reliant on requirements flow down, though global and local requirements are not discriminated. Inability to use backwards traceability to automatically roll back to previous states in time to support experimentation and iteration. Availability of impact analysis is dependent on system configuration and the software tools used (else it can be rather complex).
- Only traces back to embedded requirements which have flowed down, not to their derivation or original sources they are based upon (as restricted integration to tools not supporting pre-requirements work), so unable to deal with (e.g.) changing assumptions.
- No account is made for the different types of information held in source documents that could assist both engineering and change. Attributes can be used to give some of this information, but are complicated and limited in use (e.g., for supporting diverse types of traceability, enabling dynamic traces, or automatically distributing change details). Informal text is not project-wide, appended to single requirements, so can remain hidden.
- Separate treatment of requirements and generic objects (at differing granularity levels), only enables coarser traceability later on, as problematic to redefine traceability schemes in use.
- Lacks an explicit RS: requirements are distributed across hierarchy leaves, reducing potential for reuse.

[17]Version 1 was examined; some of these points may be addressed in version 2. See appendix I for expansion.

6. Options for improving pre-RS traceability

6.1. Solution options

It would be premature to offer a solution to the entire pre-RS traceability problem. However, suggestions (relating to improvements in education, practice, clerical and computer assistance, or automation) can be made at various levels: (1) at a high level (i.e., for the problem of gathering information to make traceable, storing it, maintaining it and retrieving it); and (2) at a low level (i.e., for the problem of conflicting levels of granularity and diverse types of trace required). We focus on high level suggestions here, as the preceding work indicates that those at a lower level can only be realistically addressed on a project-specific basis. Particular suggestions for improvements to IEF and RTM are also made.

6.2. Summary

The power of existing tools could be combined to meet the majority of the requirements for pre-RS traceability. Such tools could be integrated to form an exploratory workbench, acting as a requirements pre-processor which provides access to tools most appropriate for certain types of problem or method of analysis, to smooth the transition between pre-requirements and post-requirements work and make it traceable. They should be open, so configurable to suit the unique characteristics of development teams and application projects, and able to accommodate in-house tools used for early work. They should also be able to integrate with later development tools to provide on-going support. Standards are strongly recommended for both internal integration (preferably an object-oriented, hypermedia information structure) and external integration (preferably a very low level language protocol, based on generic standards for full requirements traceability).

To ensure that such workbenches are used, they need to support current working practices and add considerable value. Existing techniques (such as hypertext, parameterisation and language processing) could be exploited within such workbench tools, providing new ways to enhance intellectual capabilities, augment skill, help express and manage evolving requirements, and support multiple views of requirements traceability. Studies of the requirements production process, and the subsequent use and manipulation of requirements throughout a project's life, are strongly recommended for this purpose.

6.3. General suggestions

6.3.1. Obtaining information

- Tool support from project inception, to capture more early information, with the ability to: import externally derived information in multiple media (using scanners, video and audio input devices, etc.); and capture informal working and derivation, by encouraging more work to be directly carried out on-line (through the provision of computer metaphors for individual and group activities, such as: electronic white boards, post-its, drawing packages, sketch pads, group discussion tools, voice and e-mail, etc.) Additional information (process and rationale) could be generated as a by-product of use. Automatic playback could establish context and viewpoints. Active monitoring could help ensure the existence of, and assess the completeness of, required information.
- Project documentor, to shadow projects and personnel, providing an independent (and more objective) account of RS production. Such a role would be responsible for filling in any gaps, noticing when clarification is needed and promoting subsequent understanding of the work by those not directly involved.
- Producing requirements within virtual reality would automatically provide a complete and accurate record of everything (all negotiations, justifications, and any assumptions made)!

6.3.2. Storing information

- Centralised and persistent information bases (e.g., use of WORM[18] technology, which adds value through guaranteed and complete roll back. It would currently need to be implemented in software to attain required access speeds and provide selective roll back).
- Object-oriented representation of multimedia objects. This would provide flexibility of content and structure, support progressive structuring, allow storage of contents in original form, enable safe partitioning, and give amenability to hypermedia manipulation.
- Guidance for reconceptualising requirements as modular viable systems: (1) modular (highly cohesive, loosely coupled, with well-defined semantic interfaces), for reuse and ability to change content and structure; (2) generic (most abstract form), for adaptability (to enable use in different types of trace) and robustness (to maintain continuity of identity); and (3) parameterised (using module interconnection languages), for configurability of access and instantiation (to put objects together in dynamic ways).

6.3.3. Maintaining and controlling information

- Dedicated job role, to cover the responsibilities of: project librarian (to collect, polish and distribute information); information base manager (to co-ordinate and control information, ensure it is up to date and of quality); and traceability facilitator (to establish and ensure traceability of all information). Support is required from appropriate tools (e.g., database management systems, hypertext editors, complexity controllers, language processors, etc.)
- Framework for change, providing both active assistance (e.g., to identify the root object of a change and make use of module guides) and more automation (e.g., self-monitoring objects, rechecking, documentation, distribution and notification of change).
- Smoother information transfer, supporting life-cycle phases as integrated and overlapping, and encouraging common threads of involvement. Visible organisational work structures and responsibilities, and an explicit and shareable RS, would help the location of change, integration of additions, and maintenance of traceability amongst separate working groups.

6.3.4. Retrieving information

- Programmable multimedia workstations, to: retrieve multimedia objects; enable graphical and textual traces; provide diverse means of visualisation, to assist impact analysis and present requirements dynamically (using animation, 3-D, links which light up); provide concurrent (global and local) views, to assist comparison and orientation; and provide alternative and more directly engaging methods of interrogation.
- Improved presentation of stored traces, through the use of AI or expert system technology, filtering and presenting them by taking account of characteristics of the use situation and current task needs (i.e., project characteristics, who, what, when, and why...)
- Provision of user-definable traces, to fit new needs and tasks, through the ability to define trace criteria that can be interpreted and used to create them on-the-fly. This would depend on parameterised storage, so defined links could be instantiated from the root of the query and recursively followed, allowing the trace to mature to the query. These could be presented as hypermedia or linearly (by applying suitable linearisation techniques).

[18]Write Once, Read Many (non-erasable) magnetic media.

6.4. Suggestions for IEF

- Support for the analysis and representation of the wider business environment, to provide outward-looking traceability and a viable model which is responsive to external change.
- Support for BSP, to provide access to and make direct use of high level business plans (strategy, priorities and objectives), to help: qualify and assess the quality of ISP plans; justify business area priorities; and respond to changing strategies. Backwards traceability would help ensure separately developed business systems together achieve overall goals.
- More comprehensive coverage of ISP and BAA techniques already partially supported by IEF (providing more than the ability to only represent their end products), to improve conformity of interpretation and to be able to respond to changing assumptions or information that these end products depend on.
- Inclusion of those ISP and BAA techniques currently not supported by IEF (e.g., current systems analysis), so able to make more direct and continued use of their results.
- Integration with additional tools supporting planning and analysis (e.g., financial spreadsheets) and the ability to import information (e.g., interview tapes or transcripts) obtained using alternative methods or techniques (e.g., Soft Systems Methodology). This would centralise all project information, enable more explicit use to be made of it and provide traceability back to the context of derived diagram components, giving the ability to show something of their derivation and to reassess their validity later.
- Tighter traceability between planning and analysis, to provide a better assessment of interrelated business requirements distributed in the model to assist change (horizontal and backwards, as well as forwards impact) and facilitate model (or even sub-model) adoption.
- Exploitation of IEM's modularisation guidelines to create viable object structures (with clearly delineated context and inter-object semantic relations), to assist individual change and retransformation of objects, versioning, additions, early creation and maintenance of object life-cycles and the ability to explore parallel paths.
- Encourage more (informal and exploratory) work to be carried out within IEF, by the provision of flexible access to everyday tools (e.g., electronic notebooks, post-its, work charts, problem solving tools, tools to analyse interviews, etc.), to obtain information related to working practice, enable traceability of context and derivation of end results stored in diagrams, improve communication and assist change.
- Provide the ability to safely explore subsequent stages of IEM, without enforcing completion of the current stage, and the ability to document (and keep track of) information which materialises early on that will only be of value later (so it is not lost).
- Monitoring and recording of work carried out, to maintain a development history and provide the ability to trace backwards to undo transformations, roll back without the need to create explicit model versions and to enable quicker and safer impact analysis.
- Monitoring and recording of additional changes made to transformed models and of checks carried out on them, to provide more representative forwards traceability and quality retransformations which (a) automatically re-invoke the required checking and (b) either explicitly remind of changes or automatically rework them where possible.
- Provide more global visibility (for instance, of working structures and work being undertaken by separate groups, to avoid duplication of effort and assist integration).
- Explicit visibility of requirements, to provide a more globally accessible and sharable baseline than that of a changing model (composed of numerous diagrams which are problematic to jointly and coherently assimilate and determine actual requirements from).
- Smoother handover between IEF stages, to improve traceability, by ensuring certain individuals are involved throughout or there is overlap of involvement across teams and stages.

6.5. Suggestions for RTM

- Integration to diverse front end tools, linking requirements objects in RTM to pre-requirements generic objects related to their production, enabling traceability from problem analysis to requirements analysis and preventing the static treatment of source documents. Added value could be obtained: (1) the ability to construct source documents from within RTM, attach qualifying information, make assumptions explicit and account for external changes; (2) more sophisticated types of traceability, especially for change handling (e.g., to assess: cost of change, effects of changing business strategy, effects of reversing assumptions, impact of change on schedules, etc.); and (3) potential for reuse.
- Fuse the individual tools, to enable smoother working within a single tool, allowing RE work to be done in a more logical, progressive and interactive manner (i.e., extract a requirement and append attributes and links whilst still the current focus and contextually visible), improving the quality of both traceable information and links established.
- Concurrent visibility of all the information necessary for a task, to reduce the reliance on memory and to improve accuracy (e.g., simultaneous visibility of requirements being focused to make direct comparisons possible; visibility of both the original and derived requirements, and the relationships between them, whilst carrying out replacement, etc.)
- Simultaneous extraction from many source documents, as more reflective of RE practice, so possible to locate and handle duplications and relations between information early on.
- Clearer indication of, and accessibility to, project database contents. For instance: an explicit RS; browsing and navigation facilities; more discrimination in the source between new and original requirements; horizontal traceability, to view siblings in a requirements hierarchy; graphical visibility of alternative user-defined (keyword) interconnections; ability to view the whole requirements hierarchy, rather than localised views; direct access to requirements through the requirements hierarchy in a more engaging manner; etc.
- The requirements identifier number should remain internal to RTM and something conveying semantic information about the requirement should be visible to the user.
- Provision of facilities, or integration with tools, to (actively) support routine activities of RE, reduce repetition and support its error-correcting nature (i.e., informal note making, sketching, informal communication channels, etc.) Encourage more work to be done on-line, giving the ability to obtain process information and make it traceable (preventing repetition by others and giving rationale and context for the way work was finally done).
- Provision of local workspaces, as well as global ones, to enable safe trials and exploration of potential "what ifs" with current versions and the ability to directly work with past ones.
- Support the extraction of requirements by multiple individuals (representing concurrent perspectives), to explore interpretations and to attain a consensual view of requirements.
- Mechanisms to hold and discriminate system-wide requirements, information not necessarily associated with individual requirements, future requirements and information for future reference, to enable both system-wide and localised traceability.
- Better facility to discriminate types of information in source documents and directly couple such information with the extracted requirement, making individual requirements more cohesive and better to work from, and potentially more diverse types of early traceability.
- Exploitation of finer grained links between generic objects and requirements attributes, to enable diverse and task-specific post-RS traceability and accurate assessment of impact.
- Modularisation guidance, to assist grouping, and control complexity and impact. Object interfaces, defining interrelated objects, can help provide visibility of interconnections and enable the presentation of more focused and fish-eyed views.
- Thorough use of attributes, to improve impact analysis (e.g., of changing assumptions/plans)
  and improve change management (e.g., automatically monitor for and distribute changes).

7. Acknowledgements

Thanks must go to everyone I came into contact with at BT, as they all contributed in some way (either by participating in focus and discussion groups, by filling in questionnaires, by enabling access to tools, or by showing enthusiasm). Special thanks must go to David Freestone from BT for his continued guidance. I would also like to acknowledge the support from the members of the Oxford University Centre for Requirements and Foundations and its numerous consultants, particularly Marina Jirotka, Matthew Bickerton and Joseph Goguen. Finally, I must especially thank David Michael for all his useful comments, suggestions and essential encouragement over the year.

8. References

[1] Babb, R. G. (1982). Coherent Realization of System Requirements, in Ohno, Y. (Ed.), *Requirements Engineering Environments: Proceedings of the International Symposium on Current Issues of Requirements Engineering Environments*, Kyoto, Japan, September 20-21, North-Holland Publishing Company, pp. 103-105.

[2] Boehm, B. W. (1976). Software Engineering, *IEEE Transactions on Computers*, Volume C-25, Number 12, December, pp. 1226-1241.

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

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

[5] Curtis, B. (1992). The Case for Process, 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. 333-343.

[6] DeGrace, P. and Stahl, L. H. (1990). *Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms*, Yourdon Press.

[7] Dobson, J. E. and McDermid, J. A. (1991). *An Investigation into Modelling and Categorisation of Non-Functional Requirements (Part II: Methodology and Models)*, Universities of York and Newcastle upon Tyne, UK Unclassified, April.

[8] Finkelstein, A. (1991). 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.

[9] Flynn, D. J. (1992). *Information Systems Requirements: Determination and Analysis*, McGraw-Hill.

[10] Gause, D. C. and Weinberg, G. M. (1989). *Exploring Requirements: Quality Before Design*, Dorset House Publishing Company, Inc.

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

[12] Howard, S. G. (1991). Requirements and Traceability Management, *IEE Computing and Control Colloquium on "Tools and Techniques for Maintaining Traceability During Design"*, Digest Number: 1991/180, December 2, pp. 9/1-9/8.

[13] Norris, M., Miller, D., Van Toen, B. and Cordingley, B. (1989). *Review of Current Knowledge for Requirements Capture and Analysis*, RT31 Divisional Memorandum Number: RT31/89/034, BT Research Laboratories, October.

[14] Osborne, W. M. and Chikofsky, E. J. (1990). Fitting Pieces to the Maintenance Puzzle, *IEEE Software*, January, pp. 11-12.

[15] 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.

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

[17] Roman, G. C. (1985). A Taxonomy of Current Issues in Requirements Engineering, *IEEE Computer*, April, pp. 14-23.

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

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

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

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

[22] Lauber, R. J. (1982). Development Support Systems, *IEEE Computer*, May, pp. 36-46.

[23] Jackson, J. (1991). A Keyphrase Based Traceability Scheme, *IEE Computing and Control Colloquium on "Tools and Techniques for Maintaining Traceability During Design"*, Digest Number: 1991/180, December 2, pp. 2/1-2/4.

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

[25] McCausland, C. D. (1991). A Case Study in Traceability, *IEE Computing and Control Colloquium on "Tools and Techniques for Maintaining Traceability During Design"*, Digest Number: 1991/180, December 2, pp. 6/1-6/3.

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

[27] 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.

[28] Early, M. (1986). Relating Software Requirements to Software Design, *ACM SIGSOFT Software Engineering Notes*, Volume 11, Number 3, July, pp. 37-39.

[29] Baxter, I. (1991). Design Maintenance Systems, *IEE Computing and Control Colloquium on "Tools and Techniques for Maintaining Traceability During Design"*, Digest Number: 1991/180, December 2, pp. 11/1.

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

[31] 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.

[32] Alford, M. (1985). SREM at the Age of Eight; The Distributed Computing Design System, *IEEE Computer*, April, pp. 36-46.

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

[34] Penedo, M. H. (1986). Prototyping a Project Master Data Base for Software Engineering Environments, in Henderson, P. (Ed.), *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, Palo Alto, California, December 9-11, pp. 1-11.

[35] 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.

[36] 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.

[37] Hamilton, V. L. and Beeby, M. L. (1991). Issues of Traceability in Integrating Tools, *IEE Computing and Control Colloquium on "Tools and Techniques for Maintaining Traceability During Design"*, Digest Number: 1991/180, December 2, pp. 4/1-4/3.

[38] Stevens. (1987). A Model to Support the Systems Requirements Process, *Workshop on System Requirements Capture*, ESPRIT, October.

[39] 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.

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

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

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

[43] Keys, E. (1991). A Workbench Providing Traceability in Real-Time System Development, *IEE Computing and Control Colloquium on "Tools and Techniques for Maintaining Traceability During Design"*, Digest Number: 1991/180, December 2, pp. 3/1-3/2.

[44] Scheffer, P. A., Stone, A. H. and Rzepka, W. E. (1985). A Case Study of SREM, *IEEE Computer*, April, pp. 47-54.

[45] Strelich, T. (1988). The Software Life Cycle Support Environment (SLCSE): A Computer Based Framework for Developing Software Systems, in Henderson, P. (Ed.), *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, Boston, Massachusetts, November 28-30, pp. 35-44.

[46] i-Logix. (1992). *The STATEMATE Approach to Complex Systems*, Marketing Brochure, i-Logix, Inc.

[47] Catterall, B. J. (1990). The HUFIT Functionality Matrix, 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. 377-381.

[48] Hallmann, M. (1988). An Operational Requirement Description Model for Open Systems, *IEEE Proceedings*, pp. 286-295.

[49] McChesney, I., Glass, D. and Hughes, J. G. (1990). Case Tool Support for Requirements Capture and Analysis, *Microprocessing and Microprogramming*, Volume 30, pp. 281-288.

[50] Nunamaker, J. F., Konsynski, B. R., Ho, T. and Singer, C. (1976). Computer-Aided Analysis and Design of Information Systems, *Communications of the ACM*, Volume 19, Number 12, December, pp. 674-687.

[51] Potts, C. (1991). Expediency and Appropriate Technology: An Agenda for Requirements Engineering Research in the 1990s, in Van Lamsweerde, A. and Fugetta, A. (Eds.), *ESEC '91: 3rd European Software Engineering Conference*, Milan, Italy, October 21-24, Springer-Verlag, pp. 495-496.

[52] Shum, S. (1991). Cognitive Dimensions of Design Rationale, in Diaper, D. and Hammond, N. (Eds.), *People and Computers VI*, Cambridge University Press, pp. 331-344.

[53] 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.

[54] Davies, L. and Nielsen, S. (1992). An Ethnographic Study of Configuration Management and Documentation Practices, 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. 179-192.

[55] Hahn, U., Jarke, M. and Rose, T. (1991). Teamwork Support in a Knowledge-Based Information System Environment, *IEEE Transactions on Software Engineering*, Volume 17, Number 5, May, pp. 467-482.

[56] 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.

[57] Dzida, W. and Valder, W. (1984). Application Domain Modelling by Knowledge Engineering Techniques, in Shackel, B. (Ed.), *HCI Interact '84, Proceedings of the IFIP 1st International Conference on HCI*, Elsevier Science Publishers B. V., North-Holland, pp. 481-488.

[58] Edmonds, E. A., Candy, L., Slatter, P. and Lunn, S. (1989). Issues in the Design of Expert Systems for Business, *Expert Systems for Information Management*, Volume 2, Number 1, pp. 1-22.

[59] Bustard, D. W., Norris, M. and Van Toen, B. (1989). *Capturing and Analysing User Problems: Separating the "Why" from the "What"*, BT Research Laboratories, Internal Report RT31 RP/RCA/PRIDE, December.

[60] Davis, G. B. (1982). Strategies for Information Requirements Determination, *IBM Systems Journal*, Volume 21, Number 1, pp. 4-31.

[61] Finkelstein, A. (1991). Tracing Back *from* Requirements, *IEE Computing and Control Colloquium on "Tools and Techniques for Maintaining Traceability During Design"*, Digest Number: 1991/180, December 2, pp. 7/1-7/2.

[62] Mostow, J. (1985). Toward Better Models of the Design Process, *The AI Magazine*, Spring, pp. 44-57.

[63] Pidgeon, N. F., Turner, B. A. and Blockley, D. I. (1991). The Use of Grounded Theory for Conceptual Analysis in Knowledge Elicitation, *IJMMS*, Volume 35, pp. 151-173.

[64] Rein, G. L. and Ellis, C. A. (1991). rIBIS: A Real-Time Group Hypertext System, *IJMMS*, Volume 34, pp. 349-367.

[65] Robinson, W. N. (1990). Negotiation Behavior During Requirement Specification, *IEEE Proceedings*, pp. 268-276.

[66] Ross, D. T. and Schoman, K. E. (1977). Structured Analysis for Requirements Definition, *IEEE Transactions on Software Engineering*, Volume SE-3, Number 1, January, pp. 6-15.

[67] Rugaber, S., Ornburn, S. B. and Le Blanc, R. J. (1990). Recognizing Design Decisions in Programs, *IEEE Software*, January, pp. 46-54.

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

[69] 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.

[70] MacLean, A., Young, R., Bellotti, V. and Moran, T. (1991). *The AMODEUS Project: Questions, Options and Criteria: Elements of Design Space Analysis*, Amodeus Project Document: JP1, ESPRIT Basic Research Action 3066, April.

[71] MacLean, A., Young, R., Bellotti, V. and Moran, T. (1991). Design Space Analysis: Bridging From Theory To Practice Via Design Rationale, *Proceedings of Esprit Conference*, Brussels, November 25-29.

[72] Ross, D. T. (1977). Reflections on Requirements, *IEEE Transactions on Software Engineering*, Volume SE-3, Number 1, January, pp. 2-5.

[73] Wile, D. S. (1983). Program Developments: Formal Explanations of Implementations, *Communications of the ACM,* Volume 26, Number 11, November, pp. 902-911.

[74] Feather, M. S. (1989). Constructing Specifications by Combining Parallel Elaborations, *IEEE Transactions on Software Engineering*, Volume 15, Number 2, February, pp. 198-208.

[75] Kaplan, S. M., Finkelstein, A., Kaiser, G., Ryan, K. and Schafer, W. (1990). Panel: Interactively Supporting the Software Process, 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. 1047-1049.

[76] Lehman, M. M. (1991). Software Engineering, The Software Process and Their Support, *Software Engineering Journal*, Volume 6, Number 5, September, pp. 243-258.

[77] Potts, C. (1986). *Human and Organizational Factors and Environments for Information System Design*, Research Report DoC 86/6, Imperial College, London University, May.

[78] Reubenstein, H. B. (1990). *Automated Acquisition of Evolving Informal Descriptions*, Ph.D. Thesis, Report Number: AI-TR 1205, M.I.T. Artificial Intelligence Laboratory, June.

[79] 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.

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

[81] Shemer, I. (1987). Systems Analysis: A Systemic Analysis of a Conceptual Model, *Communications of the ACM*, Volume 30, Number 6, June, pp. 506-512.

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

[83] Wing, J. M. (1988). A Study of 12 Specifications of the Library Problem, *IEEE Software*, July, pp. 66-76.

[84] 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.

[85] Brooks, F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering, *IEEE Computer*, April, pp. 10-18.

[86] Carbonell, J. G. (1981). A Computational Model of Analogical Problem Solving, *IJCAI*, Volume 7, p. 147.

[87] Freestone, D. and Van Toen, B. (1991). *Preliminary Report on BT Problem Domains*, RCA-TEC-CENTRE/1225/SURVEY1/Issue 1, BT Research Laboratories, March.

[88] 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.

[89] Mostow, J. (1989). Design by Derivational Analogy: Issues in the Automated Replay of Design Plans, *Artificial Intelligence*, Volume 40, pp. 119-184.

[90] Balzer, R. (1985). A 15 Year Perspective on Automatic Programming, *IEEE Transactions on Software Engineering*, Volume SE-11, Number 11, November, pp. 1257-1268.

[91] Balzer, R., Cheatham, T. E. and Green, C. (1983). Software Technology in the 1990's: Using a New Paradigm, *IEEE Computer*, November, pp. 39-45.

[92] Chikofsky, E. J. and Cross, J. H. (1990). Reverse Engineering and Design Recovery: A Taxonomy, *IEEE Software*, January, pp. 13-17.

[93] Cleveland, L. (1989). A Program Understanding Support Environment, *IBM Systems Journal*, Volume 28, Number 2, pp. 324-344.

[94] Conklin, J. (1989). Design Rationale and Maintainability, *IEEE Proceedings*, pp. 533-539.

[95] 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.

[96] Fickas, S.F. (1985). Automating the Transformational Development of Software, *IEEE Transactions on Software Engineering*, Volume SE-11, Number 11, pp. 1268-1277.

[97] Greenspan, S. J. (1986). On the Role of Domain Knowledge in Knowledge-Based Approaches to Software Development, *ACM SIGSOFT Software Engineering Notes*, Volume 11, Number 4, August, pp. 34-36.

[98] Letovsky, S. and Soloway, E. (1986). Delocalized Plans and Program Comprehension, *IEEE Software*, May, pp. 41-48.

[99] MacLean, A., Bellotti, V. and Young, R. (1990). What Rationale is there in Design?, 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. 207-212.

[100] 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.

[101] Fickas, S. (1987). Automating the Analysis Process: An Example, *Proceedings of the 4th International Workshop on Software Specification and Design*, Monterey, California, pp. 58-67.

[102] Finkelstein, A. and Fuks, H. (1989). Multi-Party Specification, *Proceedings of the 5th International Workshop on Software Specification and Design*, Pittsburgh, Pennsylvania, USA, May 19-20, pp. 185-195.

[103] Finkelstein, A., Kramer, J. and Goedicke, M. (1990). Viewpoint Oriented Software Development, *Proceedings of the 3rd International Workshop on Software Engineering and its Applications*, Toulouse, France, December 3-7, Cigref. EC2 Volume 1, pp. 337-351.

[104] Finkelstein, A., Maibaum, T. and Finkelstein, L. (1990). Engineering-in-the-Large: Software Engineering and Instrumentation, *Proceedings of the UK IT Conference*, pp. 1-7.

[105] Finkelstein, A. and Potts, C. (1986). *Structured Common Sense: The Elicitation and Formalization of System Requirements*, Research Report DoC 86/7, Imperial College, London University, June.

[106] 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.

[107] 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.

[108] Loucopoulos, P. and Layzell, P. J. (1989). Improving Information System Development and Evolution Using a Rule-Based Paradigm, *Software Engineering Journal*, September, pp. 259-267.

[109] 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.

[110] 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.

[111] Nakagawa, A. T. and Futatsugi, K. (1991). Propagating Changes in Algebraic Specifications, *Software Engineering Journal*, November, pp. 476-486.

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

[113] Avison, D. E. and Fitzgerald, G. (1988). *Information Systems Development: Methodologies, Techniques and Tools*, Blackwell Scientific Publications.

[114] Brinkkemper, S., de Lange, M., Looman, R. and van der Steen, F. H. G. C. (1990). On the Derivation of Method Companionship by Meta-Modelling, *ACM SIGSOFT Software Engineering Notes*, Volume 15, Number 1, January, pp. 49-58.

[115] Connor, D. (1985). *Information System Specification and Design Road Map*, Prentice-Hall, Inc.

[116] Hares, J. S. (1992). *Information Engineering for the Advanced Practitioner*, John Wiley and Sons.

[117] Howard, S. G. (1991). Requirements and Traceability Management, Colloquium on *"Tools and Techniques for Maintaining Traceability During Design"*, IEE, December.

[118] Kerr, J. M. (1991). The Information Engineering Paradigm, *Journal of Systems Management*, April, pp. 28-35.

[119] Marconi Systems Technology. (1992). *RTM (the science of Requirements TransforMation) Marketing Brochure*.
[120] Marconi Systems Technology. (1992). *Requirements and Traceability Management Product Overview*.
[121] Marconi Systems Technology. (1992). *Requirements Traceability and Management Manual V1.2.4*, GEC Marconi Ltd., February.
[122] James Martin and Co. (1992). *The Information Engineering Methodology*, Advertising Brochure.
[123] Matthews, G. (1992). Artificial Construct, *Personal Computer Magazine*, March, pp. 220-224.
[124] QED. (1989). *CASE: The Potential and the Pitfalls*, QED Information Sciences, Inc.
[125] Texas Instruments. (1988). *A Guide to Information Engineering Using the IEF: Computer-Aided Planning, Analysis, and Design*, Second Edition.
[126] Texas Instruments. (1990). *IEF Analysis Toolset Guide*, August.
[127] Texas Instruments. (1990). *IEF Technology Overview*.
[128] Vessey, I., Jarvenpaa, S. L. and Tractinsky, N. (1992). Evaluation of Vendor Products: CASE Tools as Methodology Companions, *Communications of the ACM*, Volume 35, Number 4, April, pp. 90-105.
[129] Wilmot, C. D. (1992). Analytical Techniques for Verification, Validation and Testing, *BT Technology Journal*, Volume 10, Number 2, April, pp. 46-53.
[130] U.S. Department of Defense. (1988). *Military Standard: Defense System Software Development*. DOD-STD-2167A. Washington, D. C., February.

9. List of appendices