Requirements Engineering: a review and research agenda

Anthony Finkelstein

City University, Department of Computer Science, London EC1V 0HB
acwf@cs.city.ac.uk

Abstract

This paper reviews the area of requirements engineering. It outlines the key concerns to which attention should be devoted by both practitioners, who wish to "reengineer" their development processes, and academics, seeking intellectual challenges. It presents an assessment of the state-of-the-art and draws conclusions in the form of a research agenda.

1 Introduction

The purpose of this paper is to give a review of requirements engineering and to present a research agenda based on this review.

The review is not intended to be comprehensive, on the contrary it is based on a particular framework and categorisation of the principal issues, and it relies on a personal assessment of the contributions in each of the key areas. In particular, papers are cited as illustrative examples of work and not as a survey of the literature. For some surveys of the literature reference may be made to Thayer & Dorfman (1990) and to Davis (1990). A useful source of recent references can be found in Greenspan et al. (1994).

In order to orient the paper it is necessary to have a definition of the scope of requirements engineering. I favour that given by Zave (1994). "Requirements engineering is the branch of systems engineering concerned with the real-world goals for, services provided by, and constraints on a large and complex software-intensive system. It is also concerned with the relationship of these factors to precise specifications of system behaviour, and to their evolution over time and across system families."

Put crudely requirements engineering focuses on improvements to the front-end of the system development life-cycle. Establishing the needs that have given rise to the development process and organising this information in a form that will support system conception and implementation. You are asked to note the broad systems engineering remit of requirements engineering.

It is probably unnecessary to set down an extensive motivation for research in requirements engineering. In the final analysis the quality of a system is determined by the extent to which it meets the requirements of the stakeholders. The most direct route to improving system quality is therefore to ensure that requirements are accurately determined and that a requirements focus is maintained through the development process. I prefer this positive view of the importance of requirements engineering to the predominant negative view, which is as follows. Whenever practitioners are questioned about difficulties in system development they stress inadequate requirements engineering as a major cause of problems. Errors or misconceptions identified early in the development process are relatively cheap to eliminate. As development proceeds the cost of error removal escalates rapidly until the system is in the field at which point it is generally prohibitively expensive to correct any errors. Further, as development proceeds errors are more difficult to localise as they spread across components of the system.

The paper is divided into seven areas and into key concerns within each of these areas. The areas reflect the basic structure of the requirements engineering process, they are: the context in which the requirements engineering process takes place; the groundwork necessary for requirements engineering; the acquisition of the "raw" requirements; rendering these requirements useable through modelling and specification; analysis of the requirements; measurement to control the requirements and systems engineering process; communication and documentation of the results of requirements engineering. Where possible I have tried to order the concerns to reflect the progress of a rationalised requirements engineering process. I make no strong claims for this structure except perhaps that it embraces most of the current discussion on requirements engineering without the burden of introducing a novel conceptual framework.

For each of the key concerns the discussion is broken down into three parts: a brief orientation; an assessment of the state-of-the art; and a discussion of research issues.

2 Context

What are the necessary preconditions for effective Requirements Engineering?

Orientation. Before devoting increased effort and resources to requirements engineering it is essential for certain preconditions to be satisfied otherwise it will be dissipated by a generally disorganised development process. In other words it is important that developers do not run before they can walk! Because organisational distance can dim the "voice of the customer" in the subsequent development process, requirements engineering effort is particularly susceptible to wasteage. It should be immediately clear that a defined and documented development process and rigorous project management of costs, schedule & changes are prerequisites for effective requirements engineering. Without these there is no ability to make informed commitments in the development process nor channel for the information produced by requirements engineering.

Assessment. This area has been brought to general attention in the literature on software process maturity (Humphrey, 1988). Perhaps the most important research lesson that this area of work has taught us is that improvements in software development are interlocking. The results of associated studies have firmly indicated to the research community that many of its concerns are beyond the immediate capabilities of industry and that it needs to clearly identify the priorities associated with different improvements and their supporting preconditions (Humphrey et al, 1989). Requirements engineering research has been no better than any other area of software and systems engineering in this regard.

Issues. Much of the work in requirements engineering has been built on the tacit assumption that it is situated in a standard "waterfall" process of system development. In this case there is a clear mechanism for feeding the products of the requirements engineering process through to design and obvious management breakpoints for measurement and control. We have an intuitive understanding of the preconditions for requirements engineering and how to establish them. In "unconventional" processes such as incremental development there is less clarity on the interface between requirements engineering and the overall system development process and how to maintain the link between a design and the emerging requirements. Further work is necessary in this area.

Organisational setting

Orientation. Requirements engineering can take place in many organisational settings. The development process may be: internal to an organisation, where the system is being produced by that organisation for its own use; bespoke, where a client requests another organisation to produce a system specific to its requirements; customisation in which some generic product or framework is tailored to meet a set of requirements set down by an external client; cooperative in which knowledge of the application, the requirements, and the eventual use of the system is distributed among different organisations who are partners in a development process; product oriented in which an organisation develops a product to be placed in a perceived market. Each of these settings confers slightly different responsibilities and in each case suggest different technical priorities.

Assessment. The issue of organisational context and its ramifications for the organisation of system development has, until recently (Jones & Brooks, 1994), been neglected in software engineering. The information systems community has, by contrast, recognised this issue (Yadav, 1983) and has attempted to make the assumptions about organisational context, on which methods and techniques depend, explicit. Broadly the dominant view from within software engineering has been fixed on bespoke development. This is largely because this type of development is characteristic of the defence organisations and contractors who have been most articulate about their difficulties and who have funded software engineering research.

Issues. There is a growing recognition of the importance of system customisation and extension (Lubars et al, 1993) cynics might suggest that this reflects the tougher stance of defence procurement agencies. There is virtually no work on the support for developing products for markets though this concern is surfacing within general debate, in particular through large telecommunications organisations who, since deregulation, now deliver services into a global competitive marketplace.

Contract and procurement procedures

Orientation. In many organisational settings the requirements engineering process is framed by contractual and procurement issues. Statements of requirements assume a different force when embedded in a legally binding contract. The ability to question or pose alternatives to certain requirements may be blocked by the procurement procedures of which system development is only a part. Unless attention is paid to the subtle interactions between contract, procurement and requirements engineering relatively trivial issues can severely distort the development of the system.

Assessment. Most introductory texts on software and system development make mention of the concept of the specification as contract. The contract metaphor has been extensively exploited in the study of specification and of tool support (Lehman, 1985). Generally however, contractual and procurement matters are regarded as organisationally specific or otherwise out of the scope of requirements engineering.

Issues. Requirements engineering does not take place in a vacuum. Let us take as an example a typical competitive tender. Both the tender document and the bids that respond to

it contain products of requirements engineering. These may be coloured by the commercial context and the risks of giving advantage to competitors also engaged in the process. Linked to this is the difficult matter of how to demonstrate the capacity to respond to a set of customer needs without actually doing the design. These are real and significant concerns which should be addressed by research.

Personnel and staffing the requirements engineering process

Orientation. Requirements engineering requires particular skills both on the part of the engineers principally involved in the system development process and on the part of the client representing the requirements within that process. These skills are largely communication skills. On the part of the systems engineer: the ability to listen and to assimilate the vocabulary and problems of the client. On the part of the client representative: to understand their constituents; to have the authority to make commitments and to be held responsible for those commitments. In the final analysis the effectiveness of a requirements engineering process is constrained by the skills of the individuals who are party to it.

Assessment. There is a longstanding recognition in the software and systems engineering literature of the importance of communication skills in all phases of the requirements engineering process (Scharer, 1985). Some more detailed accounts arising from research on participative methods are also available. Unfortunately most of the literature is anecdotal and unsuitable as a basis for selection and training.

Issues. There is substantial empirical evidence to support the importance of skilled individuals to the development process as a whole (Boehm & Papaccio, 1988) and corresponding to this there is a body of work on the psychology, selection and training of programmers (Curtis, 1987). It would be extremely valuable if this work could be extended to the participants in requirements engineering though doing so may pose some methodological difficulties.

3 Groundwork

Bounding

Orientation. The first step in a requirements engineering exercise is to establish the scope and delineate the bounds of the requirements and design space. To set out the broad area of concern and to distinguish it form those aspects of the world which are not of concern or, viewed the other way, to define the space in which as engineers we are free to act. If the bounds are set too narrowly we may be constrained to miss an opportunity to respond to an underlying need. If the bounds are set to widely we may waste time or act outside our competence or authority. Bounding errors are characteristic of novice systems engineers.

Assessment. The issue of bounding is one of the most thorny in requirements engineering. There have been very few attempts to tackle it head on. A notable exception is Jackson & Zave (1993). It appears relatively straightforward in any given case to draw bounds but to give general guidance on how to make bounding decisions is very difficult.

Issues. Further foundational and conceptual work is required in this area. Interesting possibilities include the use of design experience to inform bounding decisions.

Feasibility and risk

Orientation. It is quite clear that there are certain requirements which it is infeasible to respond to. Typically these are cases where the costs of establishing the requirements exceeds the benefits gained in satisfying the needs which underpin them; or, where satisfying the requirements would be, prima facie, illegal, unethical or contrary to the laws of science. It is common sense that feasibility should be determined as early as possible. Alongside this it is important to identify the primary risks to which the system development process is exposed. This involves a basic assessment of the consequences of errors or failures in each part of the development process. Once this has been done it is possible to make a sensible allocation of effort across all aspects of development.

Assessment. Feasibility studies, a feature of almost all industrial system development processes have been largely ignored in the research literature. The exception being where such studies are linked to the development of prototypes (see exploration below). The area of risk as it relates to system development is attracting attention and there have been some significant contributions to this literature (Boehm et al, 1994).

Issues. Establishing feasibility is linked to the same problems of "premature design" as bounding, discussed above. Obviously our ability to establish feasibility and risk can be improved by analysis of previous projects, particularly post-mortem analysis of system failures. Methods for carrying out such analyses and for recording and deploying the resulting knowledge are of interest.

4 Acquisition

Stakeholder analysis

Orientation. The process of stakeholder analysis involves identifying those individuals or roles who should have a voice in the requirements engineering process. These may be clients, users and other beneficiaries, they may also be people involved in subsequent design, implementation, maintenance of the system. Stakeholder analysis involves understanding their responsibilities, capacities and the organisational relations between them. This analysis serves as a map for subsequent information gathering and a means of

interpreting the information provided and its status.

Assessment. There are two threads making up current work on stakeholder analysis. The first thread arises out of work on viewpoint-based methods in which viewpoints are tied to client authorities responsible for information provided within those viewpoints (Kotonya & Sommerville, 1992). The second thread arises from work on enterprise modelling in which identifying stakeholders is part of the process of modelling the organisational environment in which the system is to be placed (Blyth et al, 1993).

Issues. The work in this area needs to be brought together. From the method thread - organised guidance to assist in identifying stakeholders; from the enterprise modelling thread - modelling schemes for capturing the products of analysis. Means for reasoning about, and drawing consequences from, stakeholder analysis can be built on this.

Participation

Orientation. Requirements engineering is a group process involving cooperation. An important part of the requirements engineering task is facilitating collaborative work, consensus building and negotiation between stakeholders.

Assessment. There has been a significant body of work on participation and cooperation in requirements engineering. The most significant of this is the work on participative, joint or facilitated systems development (Macaulay, 1993) This work is related to, but distinct from, work on the application of "groupware" to requirements engineering (for example Kaplan, 1992).

Issues. The shortcomings, and hence the research issues, in this area are mostly common to the study of cooperative work in general (Grudin, 1990). There is not much solid empirical work and what there is, is too narrow to form a basis for effective support, either by tools or methods. The underlying models of cooperation are too impoverished to have real application in a complex task such as requirements engineering.

Information gathering

Orientation. Probably the most difficult task in requirements engineering is information gathering - that is gathering information on the needs and the "domain" or "environment" in which these needs are situated. This information may be set down in large documents, may be held by identifiable experts, may be buried in the work practices of individual users, and so on.

Assessment. For the most part the techniques available in this area have been borrowed from related fields. Requirements engineering has yet to evolve a distinct set of techniques of its own. The use of structured interviews and questionnaires is frequently cited but little analysed. Similarly text and document analysis. Techniques such as repertory grids have been drawn from area of knowledge acquisition. An interesting emergent area is the use of ethnographic and associated "observational" methods. A good starting point is Goguen & Linde (1993).

Issues. It is already evident that any realistic domain requires a judicious selection and combination of techniques. How to make such a selection and combination is however far from clear. There is clearly significant scope for further work in this area.

5 Modelling

Value modelling

Orientation. Decisions and tradeoffs required during design must be built on a systematic appreciation of those attributes (loosely, qualities) which are valued in a system which responds to the originating needs. This means building a model, independent of any subsequent implementation decisions, that documents and relates these values.

Assessment. It is unsurprising, given that the literature on software engineering does not recognise decision between alternatives and tradeoffs at any level above choice of algorithm, that it does not take a value-based view of requirements. Some relevant analysis can however be found in the ideas of multi-perspective development, for example the use of utility analysis by Robinson (1990). The general engineering design literature Finkelstein & Finkelstein (1983) and discussions of quality management provide some useful pointers.

Issues. The application of multi-criteria decision making techniques in software development as a whole is an interesting open area. Techniques such as QFD (Brown, 1991) which have been extensively applied to manufacturing development may be transferable.

Modelling goals and required services

Orientation. The core of requirements engineering, and the primary means by which the needs are rendered in a form that can be used to realise them, is the identification of the goals that a projected system is required to satisfy and the services that it should supply. The goals may, of course have interdependencies or conflicts which must be modelled and where appropriate resolved. In certain circumstances, goals may be interpreted as service provision; however, in identifying these it is necessary to identify the "external" actions the system should perform without constraining precisely how they should be performed.

Assessment. A number of approaches have emerged which explicitly represent goals and build a system model round these goals. Foremost among them are: Feather (1987);

Dardenne et al (1993); Fickas & Helm (1992). Though these approaches differ in specifics the broad outline is the same. I regard them as among the most promising work in the broad area of requirements engineering.

Issues. The approaches all provide means of modelling goals and reasoning about the relations between those goals. They are weak on techniques for actually identifying those goals. If the approaches developed in this area are to be taken up in practice they need to be properly tested in large case studies. Some merging of related methods would be beneficial. The approaches may also have to be shorn of representation schemes which, while important to their development hinder further exploitation.

Domain modelling

Orientation. The projected system will reside in an environment whose structure and behaviour must be understood. To this end a model, or more accurately models, of the environment must be constructed. These models must be in a form that the interactions of the system with the environment can be defined.

Assessment. The term domain modelling has been used in a variety of different senses a common usage is that of modelling an "application domain" within which a class of related systems will be built. This is a reuse issue, discussed below. I am employing a less conventional use of the term, that of modelling the "domain" objects with which a projected system system will interact. Clearly there is an overlap between these usages. The main contributions to this area are the object-oriented analysis methods for example Rumbaugh et al. (1991).

Issues. A practical problem in this area is deciding among the various methods and associated modelling schemes that have been proposed. As yet no clear means of assessing these proposals has emerged. A number of interesting approaches to the identification of domain objects have been suggested, mostly based on the use of events and scenarios (Jacobsen et al, 1992). This seems an important and interesting area for further research.

Task analysis

Orientation. Most systems interact with humans. It is essential therefore to identify these people and understand the tasks that they perform using the system. This model can be used to predict the problems they might encounter and to suggest ways in which the interface to the system can be organised to have a better fit with their tasks and the way in which they understand the system and its properties.

Assessment. In some sense much of the work carried on under the heading of task analysis mirrors other modelling carried out in requirements engineering. The difference is in the modelling focus and in the type of analysis to which these models are subject (to determine user based notions of consistency, complexity, and so on). A reasonable review is given in Johnson (1992).

Issues. Task analysis has largely been of concern to the user interface design community and the techniques which have arisen from it have, with some notable exceptions (Lim et al, 1990), not been treated as part of a larger systems engineering process. Work on method integration is required and may yield substantial economies as information collected in task models is obtainable elsewhere in the requirements engineering process. Further work on identifying relevant aspects of human task performance that can be derived from task models is also required.

Reuse

Orientation. It might be thought from the discussion above that requirements engineering is always done de novo. This is, of course far from the case. There is scope for reusing both the products and process of requirements engineering from previous exercises and for organising the process of requirements engineering so as to enhance the opportunities for subsequent reuse.

Assessment. Obviously the use of modelling schemes, employing inheritance or the like, which are capable of supporting reuse have attracted attention within requirements engineering. The most significant application of these schemes has been in the specification of "families of systems", where rather than setting down the requirements for a single system, the system is seen as an member of a family or class of related systems which share goals, services, domain models, and so on (Garlan, 1989). There is some evidence that for narrow domains with small variants, this approach may yield dividends, at the higher level however (Reubenstein & Waters, 1989) it is unproven. An interesting development in this area is the use of case-based techniques for example Maiden & Sutcliffe (1992).

Issues. I must confess to a scepticism about reuse in requirements engineering. This stems from a basic discomfort about the overall idea of reuse in the context of requirements (as distinct from design) and broader research strategy worries about treating with reuse before we have established practice for use. I feel effort is better spent in the weak link of reuse - giving developers the ability to rapidly assimilate and understand the documents and models produced by others. Given that my scepticism is unlikely to turn the tide of work on reuse the research issues are the construction of significant case bases and generic domain models for realistic domains.

6 Analysis

Validation

Orientation. Assuming that the acquisition and modelling processes are imperfect some validation of the products of the requirements engineering process is necessary. That is, they must be analysed in order to establishing the extent to which they accurately embody the requirements. Where there is a mismatch between the conception of the stakeholders and the requirements as documented this must be ironed out.

Assessment. The bulk of the work on validation has concentrated on exploration and inspection which are discussed separately below. Much of the remainder has concentrated on providing modelling schemes which are, in some sense, easy to validate (graphical languages and so on). Other work relevant to validation includes the use of scenarios (Benner et al, 1992) and specification animation (Finkelstein & Kramer, 1991). Work on tools which allow multiple views and browsing of complex document and model structures are also significant (see information management below). Alternative directions are suggested by work on specification critiquing (Fickas & Nagarajan, 1988) and on annotation schemes for marking errors in specifications (Finkelstein, 1992).

Issues. Ideally validation should be as tightly tied to and interleaved with requirements production as possible. However organisational factors can intervene to prevent this. In such cases the validator may be faced with large amounts of information and no guidance on how to proceed or what questions to ask. Research on methods for providing such guidance and on developing interesting or relevant questions to ask of the products of requirements engineering would be valuable.

Exploration

Orientation. It is well known that when confronted with a system people are able to identify its merits and demerits while unable to set down their requirements on a blank piece of paper. One way round this problem is to build a prototype or devise a system simulation as a vehicle for exploring the requirements.

Assessment. There is a considerable body of work on prototyping and exploratory development. An example of such work immediately relevant to requirements engineering is Luqi (1992). Perhaps the most interesting area of work is at the meeting point of formal specification and exploration both in executable specification and automated reasoning.

Issues. The difficulties of exploration are well documented: what should be included in the prototype or simulation; how much of the prototype or simulation should be carried through to the final realisation; and, how to guide exploration and organise feedback. Despite the amount of work in the area these difficulties remain as unresolved research issues.

Verification

Orientation. Verification seeks to establish that the subsequent products of the development process accurately reflect the requirements as documented (note the distinction between this and validation). It is no use taking great care with the requirements only to be unable to check that they are carried forward through development, for example to the formulation of a testing programme, in a consistent fashion.

Assessment. Software development orthodoxy sets down that at each stage in software development you should be able to prove that the specification (however construed) you have developed is secure with respect to the preceding specification. There is a great deal of research aiming to establish the means to achieve this a good selection can be found in IWSSD (1993).

Issues. Clearly, automated support for formal reasoning and proof requires significant further research. However, for those who are not wholehearted subscribers to the formal or transformational development agendas the issues are less clear. Verification becomes a matter of consistency management (Finkelstein et al, 1994) in which inconsistency is tolerated at certain points in development while at others consistency is checked and enforced. Taking this more permissive view of verification poses research challenges which still have to be resolved.

Inspection

Orientation. To complement more formal analysis, systematic inspection is a proven route to eliminating errors. The purpose of inspection is to remove errors and misconceptions as near source as possible hence reducing costs of rework. The basic approach involves defining exit criteria for each of the major elements of the requirements engineering process and establishing team based review with respect to these criteria. Analysis of the results of such inspections can be used for requirements engineering process improvement.

Assessment. Inspection is proven to work, it is simple and widely used throughout industry. It should follow from this that there is widespread research interest. Strangely this is not the case. The cornerstone of most research and practice in the area of inspection is still Fagan (1976). There have been some refinements (Porter & Votta, 1994) but the basic approach remains essentially unchanged. Inspection as a software engineering activity lends itself nicely to data gathering and empirical research.

Issues. The use of computer support for inspection and automated data gathering are deserving of attention. The development of groupware support for inspection has been

considered (Johnson, 1994) there is however considerable scope for further work in this area.

7 Measurement

Metrics

Orientation. A requirements engineering process is not much different from any other industrial process. It is important that the process be predictable and that schedule commitments are met with reasonable consistency. This means measurement of the products and process of requirements engineering and statistical control applied to process improvement.

Assessment. Without a settled or established requirements engineering process and an agreed set of products derived from that process it is difficult to define appropriate metrics. It should not be surprising therefore that work on metrics has not devoted much attention to requirements engineering. A good review appears in Fenton (1991). Some of the simpler metrics discussed in the literature, for example error detection rates, are broadly applicable across software development and can often provide useful indicators in requirements engineering

Issues. Progress on requirements metrics must lag inevitably research in requirements engineering. While I acknowledge the importance of measurement, a broad range of general purpose metrics in this area may not be achievable in the medium-term future.

Estimation

Orientation. It is the responsibility of requirements engineering to supply preliminary estimates of development cost, effort and schedule. These estimates may be derived from the measurements discussed above and records of development experience. However, this is an area in which current requirements engineering practice is inadequate.

Assessment. Much of what has been said above for metrics applies to estimation. which adds to the challenges of measurement those of predictive models (Boehm & Papaccio, 1988) and data collection.

Issues. Following on from the comments on metrics I would not place research on estimation high on a requirements engineering research agenda. However we need to frame requirements engineering processes and products with a view to estimation. This means among other things building support for comprehensive data collection into methods and tools.

8 Communication & Documentation

Information management

Orientation. The requirements engineering process produces large amounts of richly interrelated technical information and documentation. Some of this is textual, some graphic (drawings and diagrams). Storage and retrieval and production of high quality, tailored documentation is of considerable practical importance

Assessment. There have been substantial improvements in software support for this task based largely on improvements on improved database technology for example James (1994). The broad thrust of research in this area is linked to progress on software engineering repositories and the associated issues of distribution and long transactions.

Issues. In the future we can expect to add video and sound records of technical meetings and document annotations. There has been some preliminary work on this (Christel et al, 1993). The management of this information and its use in a principled manner is of research interest. Information presentation and extraction from large volumes of requirements engineering also presents interesting research issues.

Recording rationale and argumentation

Orientation. In the discussion above we have placed great emphasis on the creation and tracking of models, specifications and associated information - the products of requirements engineering. However in most system development processes more than 70% of costs are in rework and half the effort in these activities are about understanding the system in order to make effective corrections and enhancements. In order to achieve this understanding you need to know what decisions were considered, assumptions made and alternative solutions rejected. This information may be remembered but with time and staff turnover it soon gets lost. It is essential to keep a "process-oriented" record of the rationale and argumentation underpinning the products of requirements engineering.

Assessment. This area has attracted much attention with variants of, and improvements on, the work of Conklin (1989). For a useful discussion see Lee & Lai (1991) and for tool support Kaplan et al. (1992). Recent work of Potts et al. (1994) suggests more focussed application of argumentation support in requirements engineering.

Issues. The use of argumentation support in systems development as a whole has proceeded rapidly without any systematic assessment. Different argumentation schemes have been advanced without a clear understanding of their advantages and drawbacks with respect to existing proposals. This needs to be rectified before the area can advance further.

Traceability

Orientation. Traceability is the ability to follow the "life" of requirements in both a forward and backward direction through the development process. Forward traceability is needed to demonstrate how a requirement is manifested in a system and the intermediate products of system development. Backward traceability is required in order to maintain the integrity of the requirements in the face of subsequent design changes or in the environment in which the system operates.

Assessment. This area has recently seen an upsurge in research interest. The bulk of the work concentrates on the ability to link fragments of text, to visualise navigate these links. A detailed assessment is included in Gotel & Finkelstein, 1994.

Issues. In this assessment the issue of "pre-requirements traceability" is highlighted. In particular the problems of linking artifacts produced during requirements engineering to the groups and individuals involved in their production. Some interesting ideas on the use of truth-maintenance and constraint networks in this context are also emerging and appear worthy of further research attention.

Standards and Conformance

Orientation. Most organisations with mature systems engineering practices require conformance to external standards and codes of practice which set down how requirements should be documented and how the process should be organised.

Assessment. For the most part standards and guidelines in the requirements engineering consist of rather crude checklists and prescribed document layouts (for a good collection of examples see Dorfman & Thayer, 1990).

Issues. Standards constitute minimal good practice thus there will always be a gap between what is suggested by standard bodies and the state-of-the-art. In an area of rapid change such as requirements engineering this is doubly true. However I am inclined to the view that standards in the requirements engineering area have slipped further from what is known in research and advanced practice than is acceptable. This is a challenge to those involved in the standards process, particularly large system procurers, to reexamine this area.

9 Conclusion

I have highlighted a spread of issues that belong on a requirements engineering research agenda and have where possible tried to indicate the priority I believe should be attached to them. However, by presenting these issues area by area I have missed what I regard as the most important problem in requirements engineering research and practice. We lack an adequate understanding of the requirements engineering process as a whole. That is of how the many individual contributions can be assembled into a coherent tool-supported method (using that term loosely). Alongside advance on the issues highlighted above there is an important need for consolidation at both the conceptual and pragmatic levels.

Acknowledgements

I would like to acknowledge my colleagues at Imperial College and at City University for their guidance. I am grateful to the BritishCouncil for travel support.

References

Benner, K.; Feather, M.; Johnson, W.L. & Zorman, L. (1992); Utilizing Scenarios in the Software Development Process; Proc. IFIP WG 8.1 Working Conference on Information Systems Development Process; North-Holland.

Blyth, A., Chudge, J., Dobson, J. & Strens, M. (1993); ORDIT: a new methodology to assist in the process of eliciting and modelling organisational requirements; ACM Conference on Organizational Computing Systems 1993; pp 216-223, ACM Press.

Boehm, B.; Bose, P.; Horowitz, E. & Lee, M.J. (1994); Software Requirements as Negotiated Win Conditions; Proc. 1st International Conference on Requirements Engineering; pp 74-83, IEEE CS Press.

Boehm, B.W & Papaccio, P.N. (1988); Understanding and Controlling Software Costs; IEEE Transactions on Software Engineering, SE4, 10, pp 1462-77.

Brown, P. (1991); QFD: echoing the voice of the customer; AT&T Technical Journal; March-April; pp18-32.

Christel, M.; Wood, D.; Stevens, S. (1993); AMORE: The Advanced Multimedia Organizer for Requirements Elicitation; CMU Technical Report; CMU/EI-93-TR-12.

Conklin, J. (1989); Design Rationale and Maintainability; Proc 22nd Hawaii International Conference on System Sciences; II, pp533-539; IEEE CS Press.

Curtis, B. (1987); Five Paradigms in the Psychology of Programming; MCC Technical Report; STP-132-87.

Dardenne, A.; Fickas, S. & van Lamsweerde, A. (1993); Goal-directed Requirements Acqusition; Science of Computer Proigramming; 20, pp 3-50.

Davis, A.M. (1990); Software Requirements: analysis and specification; Prentice Hall Inc.

Dorfman, M. & Thayer, R.H. (1990); Standards, Guidelines and Examples on System and Software Requirements Engineering; IEEE CS Press Tutorial.

Fagan, M.E. (1976); Design and Code Inspections to Reduce Errors in Program Development; IBM Systems Journal; 15, 3, pp 182-211.

Feather, M.S. (1987); Language Support for the Specification and Development of Composite Systems, ACM

Transactions on Programming Languages and Systems; 9, 2, pp 198-234.

Fenton, N.E. (1991); Software Metrics: a rigorous approach; Chapman & Hall.

Fickas, S. & Helm, R. (1992); Knowledge Representation and Reasoning inthe Design of Composite Systems; IEEE Transactions on Software Engineering; pp 470- 482.

Fickas, S. & Nagarajan, P. (1988); Being Suspicious: critiquing problem specifications; Proc AAAI 88; 1, pp19-24; Morgan Kaufmann Pub.

Finkelstein, A. & Finkelstein, L.(1983); Review of Design Methodology; Proc. IEE,130 ptA,4, pp213-222.

Finkelstein, A. & Kramer, J. (1991); TARA: Tool Assisted Requirements Analysis; Conceptual Modelling, Databases & CASE: an integrated view of information systems development; [Eds] Loucopoulos, P. & Zicari, R.; Addison-Wesley.

Finkelstein, A. (1992); Reviewing and Correcting Specifications; Instructional Science, 21, pp183-198.

Finkelstein, A.; Gabbay, D.; Hunter, A.; Kramer, J.; & Nuseibeh, B. (1994); Inconsistency Handling in Multiperspective Specifications; IEEE Transactions on Software Engineering; 20, 8, pp 569-578.

Garlan, D. (1989); The Role of Formalized Domain-Specific Software Frameworks; Proc. 5th International Software Process Workshop; IEEE CS Press

Goguen, J. & Linde, C. (1993); Techniques for Requirements Elicitation; Proc. IEEE International Symposium on Requirements Engineering; pp 152-164, IEEE CS Press.

Gotel, O. & Finkelstein, A. (1994); An Analysis of the Requirements Traceability Problem; Proc. International Conference on Requirements Engineering; pp 94-101; IEEE CS Press.

Greenspan, S.; Mylopoulos, J. & Borgida, A. (1994); On Formal Requirements Modelling Languages: RML revisited; Proc. 16th International Conference on Software Engineering; pp 135-147, IEEE CS Press

Grudin, J. (1990); Groupware and Cooperative Work: problems and prospects; [In] Laurel, B. [Ed.], The Art of Human-Computer Interface Design; Addison-Wesley.

Humphrey, W.S. (1988); Characterizing the Software Process: a maturity framework; IEEE Software; 5, 2, pp 73-79.

Humphrey. W.S.; Kitson, D.H. & Kasse, T.C. (1989); The State of Software Engineering Practice: a preliminary report; Proc. 11th International Conference on Software Engineering; pp 277-288, IEEE CS Press.

IWSSD (1993); Proc. of the 7th International Workshop on Software Specification and Design; IEEE CS Press.

Jackson, M. & Zave, P. (1993); Domain Descriptions; Proc. IEEE International Symposium on Requirements Engineering; pp 56-64, IEEE CS Press.

Jacobsen, I; Christerson, M.; Jonsson, P. & Overgaard, G. (1992); Object-Oriented Software Engineering: a use-case driven approach; Addison-Wesley.

James, L. (1994); Practical Experience in Automatic Requirements Elicitation: the real issues; Proc. of Workshop on Requirements Elicitation for Software-based Systems (RESS); University of Keele, UK.

Johnson, P. (1992); Human Computer Interaction: psychology, task analysis and software engineering; McGraw-Hill.

Johnson, P. (1994); An Instrumented Approach to Improving Software Quality through Formal Technical Review; Proc. 16th International Conference on Software Engineering; pp113-122, IEEE CS Press.

Jones, M. & Brooks, L. (1994); Addressing Organisational Context in Requirements Analysis Using Cognitive Mapping;; Proc. of Workshop on Requirements Elicitation for Software-based Systems (RESS); University of Keele, UK.

Kaplan, S.M. et al (1992); Supporting Collaborative Software Development with Conversation Builder; Proc. ACM SDE; pp 11-20.

Kotonya, G. & Sommerville, I (1992); Viewpoints for Requirements Definition; Software Engineering Journal; 7, 6, pp 375-387.

Lee, J. & Lai, K-Y. (1991); What's in Design Rationale?; Human-Computer Interaction; 6, 3-4, pp 251-280.

Lehman, M. (1985); Approach to a Disciplined Development Process - The ISTAR Integrated Project Support Environment; Imperial College Dept. of Computing Tech Report 85/19.

Lim, K.Y; Long, J.B. & Silcock, N. (1990); Requirements, Research and Strategy for Integrating Human Factors with Structured Analysis and Design Methods: The Case of the Jackson System Development Method; [In] Contemporary Ergonomics 1990; Proc. Ergonomics Society Conference; Taylor and Francis.

Lubars, M. & Potts, C. & Richter, C. (1993); Developing Initial OOA Models; Proc. 15th International Conference on Software Engineering; IEEE CS Press.

Luqi (1992); Computer-aided Prototyping for a Command-and-Control System Using CAPS; IEEE Software; 9, 1, pp 56-67.

Macaulay, L. (1993); Requirements Capture as a Cooperative Activity; Proc. IEEE International Symposium on Requirements Engineering; pp 174-181, IEEE CS Press.

Maiden, N. & Sutcliffe, A. (1993); Exploiting Reuseable Specifications Through Analogy; Communications of the ACM; 35, 4, pp 55-64.

Porter, A.A. & Votta, L.G. (1994); An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspection; Proc. 16th International Conference on Software Engineering; pp 103-112, IEEE CS Press.

Potts, C.; Takahashi, K.; Anton, A. (1994); Inquiry-Based Scenario Analysis of System Requirements; Georgia Tech Technical Report; GIT-CC-94-14.

Reubenstein, H. & Waters, R. (1989); The Requirements Apprentice: an initial scenario; Proc 5th International Workshop on Software Specification & Design; pp 211-218, IEEE CS Press.

Robinson, W. (1990); Negotiation Behaviour During Requirements Specification; Proc. 12th International Conference on Software Engineering; pp 268-276, IEEE CS Press.

Rumbaugh, J.; Blaha, W.; Premerlani, F. & Lorenson, W. (1991); Object-Oriented Modelling and Design; Prentice-Hall Inc.

Scharer, L. (1985); Pinpointing Requirements; Datamation; April, 1981; pp 139-151.

Thayer, R.H. & Dorfman, M. (1990); System and Software Requirements Engineering; IEEE CS Press Tutorial; IEEE CS Press.

Yadav, S.B. (1983); Determining an Organisations Information Requirements: a state-of-the-art survey; Database; 3-20.

Zave, P. (1994); Call for Papers and Associated Classification Scheme; IEEE International Symposium on Requirements Engineering 1995.