Requirements Eng (1996) 1: 170–189
© 1996 Springer-Verlag London Limited

Requirements Engineering

A Conceptual Framework for Requirements Engineering

Alistair Sutcliffe
Centre for HCI Design,School of Informatics, City University, London, UK

A framework for assessing research and practice in requirements engineering is proposed. The framework is used to survey state of the art research contributions and practice. The framework considers a task activity view of requirements, and elaborates different views of requirements engineering (RE) depending on the starting point of a system development. Another perspective is to analyse RE from different conceptions of products and their properties. RE research is examined within this framework and then placed in the context of how it extends current system development methods and systems analysis techniques.

Keywords: Requirements Engineering; Process models; Products; Review; Framework

1. Introduction

The high cost of errors incurred during requirements analysis and many system failures have been attributed to mistakes in the early phases of system development [1]. Requirements engineering (RE) aims to address this problem but the scope of what this entails is not clear. The foundations of the area were set out in a collection of papers edited by Thayer and Dorfman [2] and special issues of *IEEE Transactions on Software Engineering* [3,4]. These were followed by IEEE symposia and conferences [5,6]. These events have revealed a diversity of research issues and industrial practice which can be loosely associated with defining 'what' to build rather than 'how'. The diversity in subject matter and approaches published in the field demonstrates that RE, like many new disciplines, has to go through a definition stage and somewhat ironically establish its own requirements. Lubars *et al.* [7] report one of the few investigations of RE practice in industry. They note that ambiguity and changing requirements are a constant problem and that developers prefer organisational to technological solutions for RE problems. More recently, field studies of system development practice [8] have indicated that changing requirements, lack of trained manpower and inadequate methods are responsible for system failures. However, there is still a pressing need for further data on industrial RE problems. RE research has tended to be dominated by large customer-driven systems, typically in the defence sector. In these contexts requirements are complex and often driven by a super designer's vision, whereas market-driven requirements arise out of a more creative brainstorming approach [7]. So far the research agenda has not addressed how context may change the nature of requirements problems.

A necessary precursor to studying requirements problems is to create a framework for describing the RE process to scope further investigation. In this paper we propose a framework to analyse current research and describe the dimensions of RE. The motivation is two-fold: first to give an overview of different pathways in which requirements are discovered, analysed and recorded; and secondly to review RE research in the perspective of structured system development methods.

Although the RE process usually starts with top-down decomposition to create the goal hierarchies, many systems start with problems rather than intentions to create a new system. Furthermore, user requirements may also be promoted by examples of other successful systems. Clearly RE has several starting points. Pathways are proposed as coarse-grained process models of RE from different viewpoints. RE research effort has not been evenly distributed. Much research has been concentrated on large-scale military

Correspondence and offprint requests to: Alistair Sutcliffe, Centre for HCI Design, School of Information, City University, Northampton Square, London EC1V 0HB, UK. Email: A.G.Sutcliffe@uk.ac.city systems, which are atypical in many respects because they are large, complex and go through a tender/procurement process. This is very different for information systems requirements. To assess the implications of different types of applications we investigate how different products effect the requirements process. The paper is organised in four parts. First definitions of RE and activities are reviewed. The next part develops the process view of RE, followed by the product view. The paper concludes with a discussion of outstanding research issues.

2. Existing Conceptions of Requirements Engineering

Several definitions of RE have been given, for instance in terms of the outcome: 'a requirements specification should tell a designer everything he needs to know to satisfy all the stakeholders - but nothing more' [9]. Alternatively, the principal RE issue described by Bubenko [10] is 'how to proceed from informal, fuzzy individual statements of requirements to a formal specification that is understood and agreed by all stakeholders'. Dubois *et al.* [11] use the term to refer to the part of the development life cycle consisting of investigation of the needs and requirements for the user community and abstracting from these to formal specifications.

Other reports have revealed a variety of approaches ranging from the formal [12] to informal approaches [13] addressing diverse issues starting with 'needs' but also encompassing domain modelling [14] and design requirements such as safety critical properties. Clearly, a number of issues are involved in the process of deciding what to build in a software system. Lubars *et al.* [7] note that issues such as recording assumptions and decisions, understanding the effect of business changes, and the use of domain models have not been solved. Desiderata of requirements specifications such as completeness, correctness, unambiguous expression, traceability, etc. have been enumerated; likewise taxonomies of non-functional requirements [15], contents of requirements specifications [16] and standards for requirements documentation [17] have been described. Unfortunately categorisation of the desired outcome of RE processes gives little help for those wishing to practise it.

Feather *et al.* [18] draw attention to the importance of informal knowledge which is often incomplete, ambiguous and inconsistent; hence there is a need to preserve expressive freedoms in requirements, i.e. varying degrees of formality. Social factors, such as power and responsibility, influence the expression of requirements [19] while fact gathering has to tackle problems of tacit knowledge and public/private versions of truth which stakeholders may, or may not, wish to communicate [20,21].

The three dimensions of RE proposed by Pohl [22] (see Fig. 1) illustrate three major facets of the problem: namely modelling the future application in a more complete manner, modelling with more formality, and the stakeholder dimension of agreeing requirements. Input to the process starts with coarse-grained and ambiguous statements about the intended system. Users may have different visions of the system or only partial and incomplete ideas about what they want. Input is characteristically informal in its representation, imprecise and personal as requirements are initially held by individuals and frequently conflict with one another. However, the desired output from RE is very different. It should be a complete system specification, within the constraints of available resources, using a formal language, and agreed by all involved.

As the process proceeds a thread traces the emerging requirements specification as it becomes more complete, accurate and shared. On the specification dimension, RE has to guide the discovery, refining and validation of requirements as they become more thoroughly understood and complete. Representation has to support expression of requirement statements and models in natural language, semi-formal graphical notations such as data flow diagrams (DFDs) and entity relationship (ER) diagrams and formal notations. Finally the agreement dimension has to support tradeoffs between requirements and different stakeholders' views, negotiation and coordination of a distributed, cooperative process.

Requirements can be imposed on a system by laws of physics and facts of nature. In other words the system must conform to these facts and process events in a specific order otherwise it will not work. This problem of requirements emanating from implications of events crossing the system boundary has been investigated by Jackson [23] in several refinements starting with the Jackson method which derived entity life histories from explicit consideration of real world events. In more recent papers, the correspondence between real world events and system behaviour has been expressed in terms of optative requirements to which the system must respond [12]. This has led to proposing problem frames for handling event patterns belonging to generic models of applications [14].

The event-oriented view may be considered to be a special case of requirements imposed by the environment; however, the difference between users' requirements and requirements imposed by the external world is probably best seen as a matter of separating concerns in RE. One deals with understanding the user's need, while the other investigates constraints resulting from the physical system environment.

Previous frameworks for RE have considered the subject from an activity-oriented viewpoint. Roman [24] provided the first list of issues in RE and drew attention to the needs for validation and modelling of functional and non-functional requirements. Davies [16] reviewed requirements specification in the light of development methods such as System Analysis and Design Technique (SADT), and investigated the specification of behavioural requirements in particular. Loucopoulos and Karakostas [25] provided a more in-depth view of the RE process by reviewing techniques and approaches to elicitation, modelling and validating of requirements. They also drew on the information systems methodology literature and assessed the suitability of Computer Aided Software Engineering (CASE) technology for RE. A more elaborate taxonomy of issues is used by Zave [26] to classify contribution to RE state of the art in three problem dimensions. These consist of problems of investigating goals, functions and services for software systems; specifying system behaviour; and managing the evaluation of systems and families of systems. Within each category sub-issues are listed such as elaborating vague goals, communicating with users, etc. While this taxonomy is probably the most elaborate to date, its aims were to evaluate research contributions rather than describe the process of RE. In this paper we focus on activity descriptions of RE and propose a framework for analysing those activities for the purpose of (a) planning future research directions, (b) reviewing current contributions to research and (c) understanding the requirements on such activities implied by different products.

3. A Framework for Discovering and Refining Requirements

While we do not intend to give an exhaustive process definition of RE activities as many will depend on local organisational contexts, it is useful to consider a high-level process model for RE. In this light we propose a

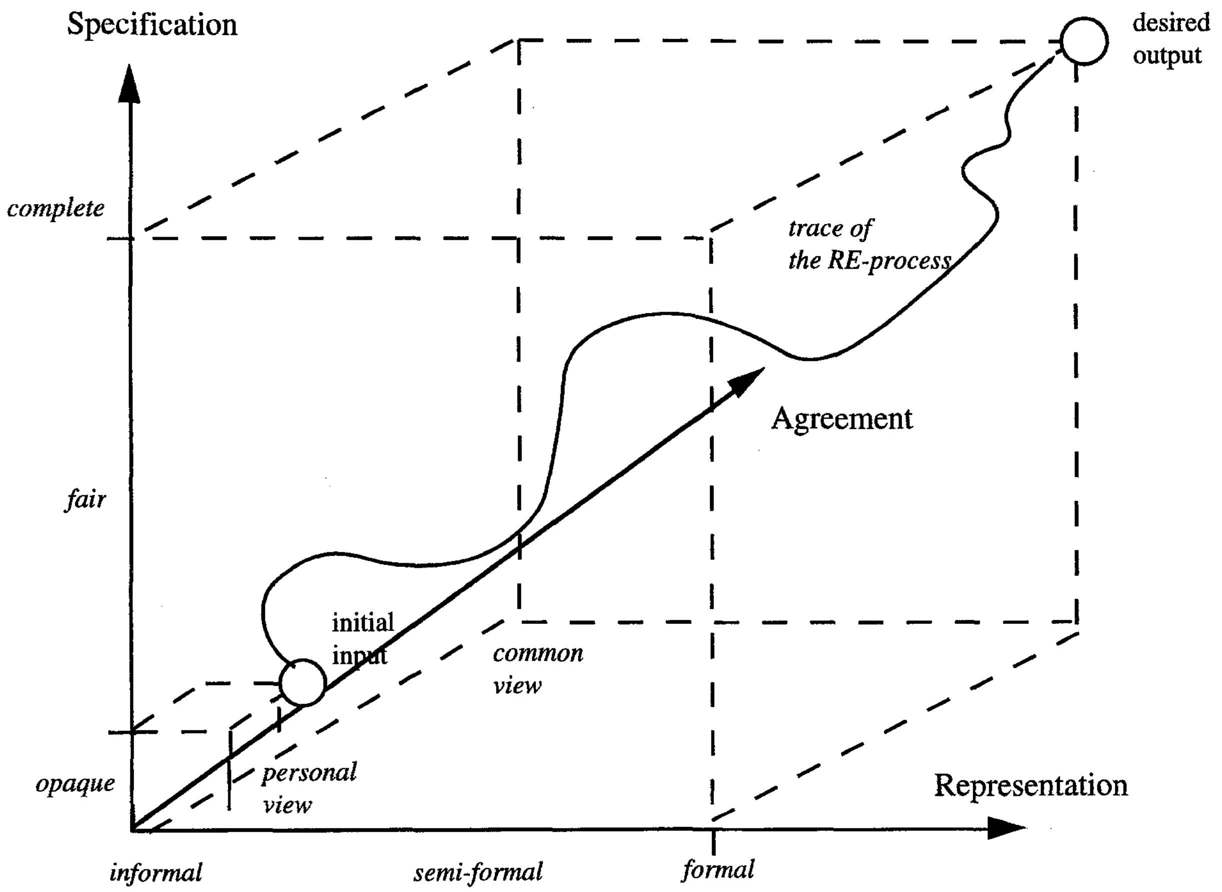

Fig. 1. The three dimensions of requirements engineering [22].

'road map' of RE processes to summarise different approaches which may be taken for requirements analysis. First we describe high-level activities which form components of the RE process model.

3.1. RE Activities

3.1.1. Scoping

Requirements frequently start with a vague statement of intent. The first problem is to establish the boundary of investigation and, inter alia, the scope of the intended system. Unfortunately, this is rarely an easy process as clients often don't know exactly what they want, and knowledge about the intended system is vague. Scoping tends to be an iterative activity as the boundaries become clearer with increasing understanding of the domain shared by all the stakeholders. However, the process is poorly understood. For general scoping, enterprise modelling [27] provides a way of describing the business context to discover requirements in the large (i.e. goals, aims, policies), but little process guidance is offered. KJ brainstorming methods and rapid application development/joint application development (RAD/JAD) [28] workshops are the current state of the art, although these too offer little systematic guidance. More detailed scoping has been researched by Jackson and Zave [12], who propose techniques for establishing the system boundary by examination of the intended system's obligations for responding to real world events, although this does not help bounding investigations which start from general statements of users' intentions.

3.1.2. Fact Gathering

For the most part the techniques for this activity have been borrowed from systems analysis, e.g. interviews, observation, questionnaires, text and document analysis [29]. Techniques from knowledge acquisition such as repertory grids and protocol analysis have been employed, but there have been no systematic investigations into the merits of different fact capture techniques, apart from a preliminary study by Maiden and Rugg [21]. An interesting emergent area is the use of ethnographic and associated 'observational' methods [13,30]; however, these have failed so far to deliver explicit guidance for fact capture or analysis, leading software engineers to propose their own 'quick and dirty' approaches [31].

3.1.3. Analysis

Analysis and modelling generally follow top-down approaches, concentrating on goal decomposition. Potts *et al.* [32,33] provide a means of goal-related analysis which uses scenarios to discover obstacles, or potential problems caused by external agents that the system has to deal with. From obstacles, goals for maintaining, avoiding and repairing situations can be elaborated. Other goal decomposition methods follow a taxonomic approach and attempt to analyse goals in the context of domain models [34]. For problem analysis, soft systems methodology [35] gives a means of informal modelling and an analytic approach to discovering problem-oriented requirements. Rationale-based techniques are also appropriate. These structure analysis in hierarchies of graphs linking goals with potential solutions and supporting arguments; see Conklin and Begerman [36] and MacLean *et al.* [37].

3.1.4. Modelling

This activity consumes the output from analysis, structures facts and represents them in a notation. RE has borrowed techniques for this activity from structured system development methods and conceptual modelling. Informal modelling notations such as data flow diagrams and entity relationship diagrams have been widely used, although the value of hypertext for informal structuring of linguistically expressed requirements has also been recognised [17]. Many formal approaches to modelling have been imported from software engineering [12,38], although the effectiveness of these techniques has yet to be demonstrated in industrial practice. Analysis and modelling are frequently interleaved to elaborate the requirements as understanding of the problem domain increases through the act of representation.

3.1.5. Validation

This is a key activity in RE which, in spite of being extensively researched, is still problematic. Validation implies getting users to understand the implications of a requirements specification and then agree that it accurately reflects their wishes. The current state of the art is walkthrough techniques in which semi-formal specifications such as data flow diagrams are critiqued in a workshop of designers and users. Walkthroughs have the merit of early validation of specifications, whereas prototypes are probably more powerful as users react more strongly to an actual working system. Unfortunately prototypes still incur construction costs and poorly organised use of prototyping can be detrimental [39]. However, prototypes in combination with techniques for gathering and evaluating user feedback can be highly effective [40]. Overall, the process of validation is poorly understood and explanation is an important yet often neglected component.

Some research in explaining complex requirements has demonstrated that a combination of visualisation, examples and simulation is necessary [41,42]. Scenario-based representations and animated simulations help users see the implications of system behaviour and thereby improve validation [43]; furthermore, early prototypes with scenarios are a powerful means of eliciting validation feedback [44]. The inquiry cycle technique [33] of approaches validation by comparing scripts of imagined real world behaviour against the required behaviour in a specification. Validation is still poorly understood and further research is necessary to discover how explanation, representation and users' understanding of system specifications interact.

3.1.6. Trade-off Analysis

Requirements frequently cannot be satisfied by a specification. Non-functional requirements (NFRs) fall into this category whereby the design can accommodate them to some degree although a complete solution is not feasible. Requirements are often held by different stakeholders who may have conflicting views, hence trade-off analysis is an essential activity for comparing, prioritising and deciding between different requirements or design options. Ranked lists or matrix-based techniques using decision tables are useful for this analysis. The modelling techniques proposed by Chung [45] and Yu [46] for mapping relationships and dependencies between goals, tasks, actor and soft goals (alias non-functional requirements) contain some guidance for trade-off analysis. Their method and support tools facilitate tracing influences between goals and NFRs, as well as giving active guidance about potential clashes between different types of NFR (e.g., security may militate against ease of use). This work is a significant advance in handling trade-offs. In spite of this, few tools or methods exist to help the requirements engineer, although house of quality [47] techniques have been imported into RE and some tool support is available [48]. More complex approaches such as multi-criteria decision making [49] do not seem to have been considered in RE.

3.1.7. Negotiation

The social dimension of RE is poorly understood. This activity subsumes many others, e.g. analysis, trade-off, modelling, but the essence lies in discussion, explanation and negotiation of conflicting requirements. The modelling work of Chung [45] contributes to negotiation by creating a shared artefact through which influences and design alternatives can be discussed. This is effected by creating a strategic dependency model to map out relationships between goals, tasks, actors, etc., followed by a strategic rationale model which illustrates potential system solutions for the requirements with arguments for and against them. Unfortunately, these models provide no active guidance for agreeing requirements, although Boehm *et al.* [50] suggest some heuristics for structuring successful negotiation of requirements. Stakeholder analysis methods in cooperative requirements capture [51] help to structure the composition of workshops with different stakeholders and to provide a framework for considering requirements from different viewpoints. Guidance for managing RE meetings, handling negotiation and conflict resolution is hard to find. Social science research on meetings describes roles, desiderata for leadership and managing consensus in groups [52]; however, this research has not been applied in RE.

3.1.8. Assessing Non-functional Requirements

Non-functional requirements become criteria which must be satisfied in a designed solution, so functional requirements should be assessed and prioritised against non-functional requirements using trade-off techniques, e.g. house of quality [47]. Chung [45] described graphical notations for modelling the interrelationships between non-functional requirements (called soft goals), functional requirements, tasks and activities. Dependencies are mapped from the goals which need to be satisfied to activities for carrying them out, and actors responsible for instigating the activities, etc. Rationale-like relationships are added to show the degree with which a requirement is satisfied by an activity (i.e. a system function). Chung proposes template methods for dealing with a limited set of non-functional requirements, such as accuracy and security [45], while Yu [46,58] has extended this work to investigate relationship modelling for information systems and business process design. Such modelling techniques promote a more systematic evaluation of non-functional requirements and trade-offs between design options. When design alternatives have been examined, requirements for the preferred option are elaborated into models of the intended system.

3.1.9. Evaluating Socio-technical Solutions

Satisfying a requirement may not necessitate an automated system, as management action for changing resources, procedures or responsibilities may suffice. Three possible avenues need to be explored:

- management implications for requirements that are not amenable to automation; instead a decision is required about resources, for example 'Hire more staff to improve customer service';

- operational implications require some change to procedures carried out by people;
- opportunity for automated support; a computer system could be introduced to implement a required function, or an existing system could be improved to meet the need.

If automated support is required, the question becomes what sort of computerised support is required to help the users achieve their goals. Stakeholder analysis methods [51] can help evaluate the advantages and disadvantages of proposed solutions for different user groups, e.g. primary users who operate the system, secondary users who receive its output, users who are responsible for the system, etc. Unfortunately, few methods exist for forecasting the potential change to organisations as a consequence of introducing computer systems. Organisational and enterprise models [20,27] can informally map out the problem, while the modelling framework of Chung [45] gives some support for tracing allocations of responsibility for achieving goals to task and agents. Ultimately though, assessing the impact of IT still relies on human judgement.

3.1.10. Specifying Human Computer Cooperation

Users' goals which imply some automated support require further analysis to define the functional requirements for supporting users' work. Judgement about whether a process should be automated or not should be taken in consultation with users. This is a critical activity in RE, frequently ignored in many reports. Task analysis methods may help to elaborate requirements for supporting the user's work; however, few methods have explicitly addressed this problem [44]. Reviews of task modelling can be found in Johnson *et al.* [43], or Sutcliffe [63], while an approach integrated with software engineering is given by Lim and Long [64]. Many task analysis methods are based on goal modelling [63], so synergy with these RE approaches needs to be explored.

This concludes the description of the principal high-level components of the RE process. We now consider how these vary according to the route from which requirements start out.

3.2. A Road Map of Requirements Analysis

This section presents tentative process models of the RE process based on the author's experience and investigation of the literature. The models propose possible pathways for different RE contexts. These are presented to review research approaches to date, expose future research issues and act as hypotheses which can be tested by empirical research into RE processes and task models. Requirements analysis and modelling may be influenced by the context of their initiating conditions, following four main paths.

3.2.1. Policy-driven Requirements

Requirements are initiated by senior managers and company executives as policies, aims, objectives and other high-level statements of intent. This source includes visions of the future, such as the famous statement by President Kennedy to 'send a man to the moon and safely return him to the earth within this decade'. This route, illustrated in Fig. 2, necessitates considerable scoping activity as requirements start with vaguely expressed intentions and users' wish lists. Policy can be analysed within the business context by enterprise models. Arguably there are non-RE activities which are pertinent to policy analysis such as business modelling, value chain analysis [53], competitive advantage theories and more recently business process re-engineering [54]. Business analysis techniques such as business process analysis, concept maps [55], and critical success factors [56] are also applicable at this stage; however, proposing a detailed methodology is beyond the remit of RE. The key problem is to model the business to discover opportunities for developing computer systems to enhance competitive advantage. Although some suggestions can be found in value chain models [53], and case histories of inter-organisational system design [57], this area is poorly understood. The methods and approaches in the business analysis community are still largely a matter of intuition, so regrettably only limited guidance can be gleaned from this source.

Top-down decomposition is the normal approach whereby policy-level intentions are successively decomposed to goals. Relationships are added progressively as the context of the policy is understood in terms of what has to be done to achieve it (goals) and what the implications are for people (actors) and their organisations (organisation unit, objects, etc.). Modelling goals in the context of how they impact on tasks and the organisation is vital not only to elaborate the meaning of informal statements of intent but also to enable assessment of the impact of change [45,58]. Goals have to be refined as linguistic statements of intent until the stage when the desired state of the system can be described, when formalisation may be possible [87]. Hypertext tools can help to represent informal goal hierarchies [17], as can standard conceptual models, e.g. data flow diagrams, but there is little advice or process guidance for goal-related requirements analysis. Chung [45] and Yu [58] provide representations of goals in context models showing dependencies between goals (both functional and non functional), actors and tasks, with some guidelines for goal decomposition and modelling. Modelling is an essential precursor for validation in this route as goals cannot be easily understood without contextual detail about how they may be achieved and their relationship to agents and processes. The policy route converges with other RE pathways for common activities of trade-off analysis and negotiation, both of which are important for goal-oriented RE. Once goals are decomposed to the stage when the desired state of the system can be described, at least informally, the first cut decisions on use of technology can be made. Some goals become functional requirements, while others have implications for management alone (e.g. decisions about resources, organisation and human activity).

3.2.2. Problem-Initiated Requirements

In this case an existing system contains a problem, and the user's need is to specify a solution. Scoping is

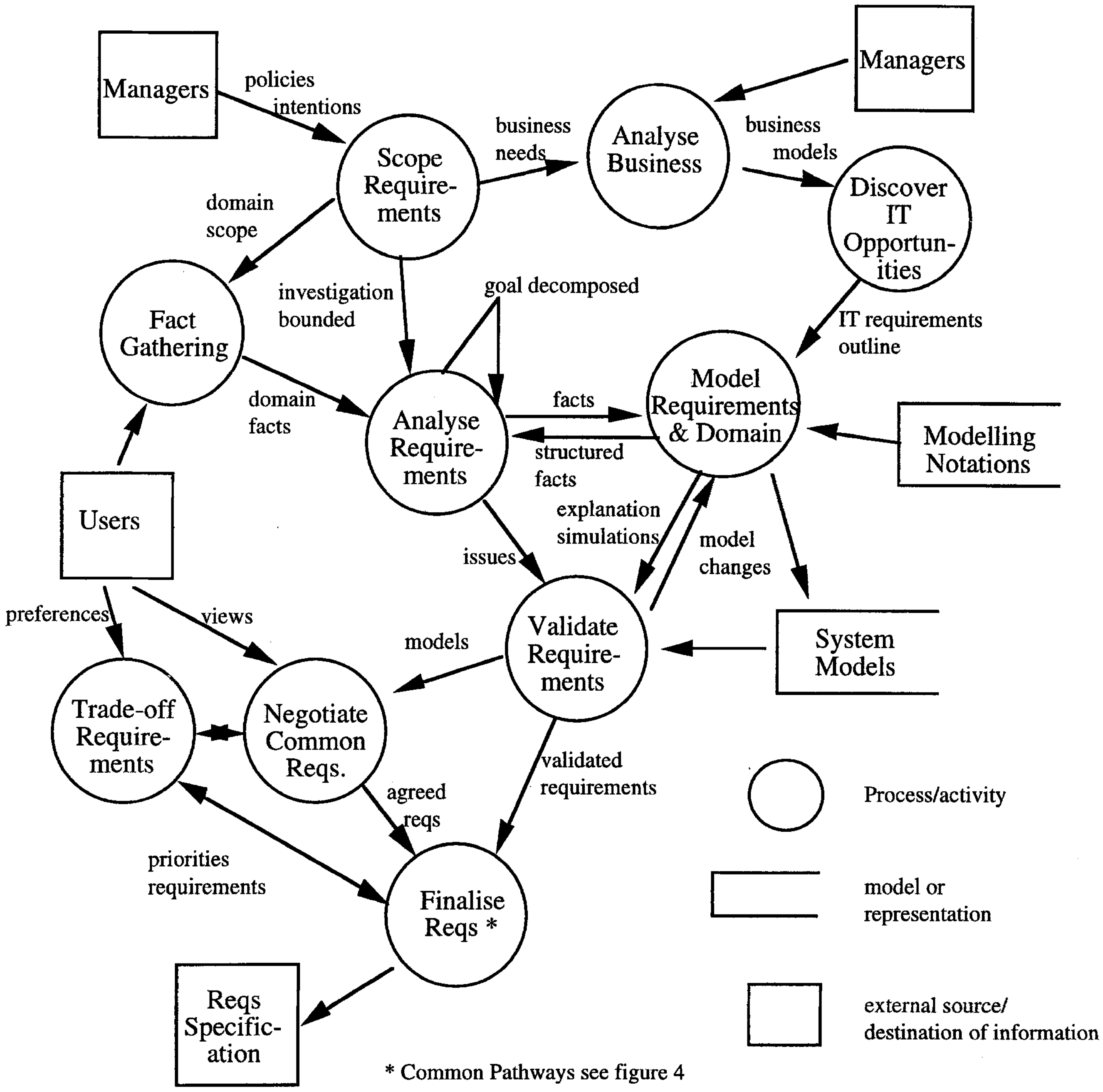

Fig. 2. Policy–goal route for requirements discovery.

necessary but less critical as the observed problem defines the initial scope for investigation. To analyse the problem context various techniques can be applied. Checkland's Soft Systems methodology [35] provides one approach for diagnosis of business problems via informal modelling. More detailed problem analysis in safety critical systems may employ fault/event trees, or Hazops methods [59]. Cause–effect nets and systems dynamics models could also be employed, although these do not appear to have been used within RE. Cybernetic models and their derivatives [60] present another view of feedback to uncover potential problems; however, these, too, do not appear to have been applied to RE.

Conceptual models can help to identify different causes which may be attributable to people (actor relationships), performance problems (object property relationships, e.g. volumes or transactions), organisational mismatches, etc. Flow diagram tracing for process/event dependency is useful for problem analysis, and detailed process analysis is often necessary to establish causes. Requirements are realised as technical solutions, or managerial decisions to remedy problems in the social system. Negotiation and trade-off analysis are less important activities as the need to fix the problem drives priorities. Validation, however, is important to check that the proposed solution will actually cure the original problem. The inquiry cycle [32,33] links problem analysis with goal modelling by positing obstacles in scenarios of use which the required system must deal with. Requirements goals are then proposed to address problems, correct errors, and maintain a desired state. In general, this pathway has received less attention in RE research than the policy route, probably because problem-initiated requirements are seen as modifications rather than implying the need for a completely new requirements specification (see Fig. 3).

3.2.3. Requirements by Example

These requirements happen when stakeholders hear about or see a demonstration of an existing, innovative application. The user's immediate goal is often to acquire new technology, although the fit of technology with their work goals should be investigated. A variant of this pathway is experts recommending a system enhancement to improve the effectiveness of users' work. Scoping in this pathway is oriented to finding the fit between the new technology and existing work practices. Analysis is driven by an existing system which is, in itself, a complex and detailed requirements specification. In this route the properties of the designed product are known quantities; furthermore, several alternative product solutions may exist. Modelling may be necessary to establish the fit between users and the new system, although analysis and modelling may not be as critical as in other RE pathways because a requirements specification could be derived from the product documentation. Organisational and task models may be constructed to create scenarios for assessing how the new system may fit into existing working practices, and for deciding how the system or working practices may have to be changed. Trade-off analysis is important as the requirements engineer needs to establish the goodness of fit between the requirements and the properties of one or more products which satisfy them.

This pathway has received very little attention in RE research. Procurement-based requirements imply analysis of the usage context to assess the advantages of the new product need before a decision to purchase is taken. Decision tables and matrices comparing product properties against requirements are state of the practice. Some tools have been developed, based on matrix-based decision support from the quality function deployment literature [48]. Scenario simulations, mock-ups and Wizard of Oz techniques all provide users with realistic visions of the required system. Prototypes may also be considered as a version of requirements by example in that they provide a limited vision of the eventual system. Such techniques have been effectively employed for a number of years in combination with simplified analysis methods (e.g. rapid applications development [28]) or in a usability evaluation cycle [61], although warnings about the dangers of unguided prototyping have been reported [39].

However, prototypes, simulations and other early delivery mechanisms are not truly requirements by example, but a manifestation of the required system which has been analysed by other techniques. In contrast, the process of analysing requirements from an existing complete product have received little attention. In industry this approach is often followed when purchasing commercial off-the-shelf (COTS) software and application packages. Matrix comparison techniques can establish trade-off judgements; alternatively, conformance checks against a product's features can be used. More research is necessary to provide more systematic guidance, and this need will become pressing with the growth in COTS software.

3.2.4. Requirements Imposed by the External Environment

Initially these may be high-level statements similar to policies, e.g. standards, legislation, and regulations to improve safety, reduce pollution, or more detailed requirements from an external source, e.g. provide audit data for the chief accountant. In this pathway scoping becomes impact analysis to establish whether existing systems will have to be changed to conform with the new requirement. Analysis refines the impact assessment once the subsystems to be changed have been identified. Walkthroughs with models of the actors, activities and objects can facilitate identification

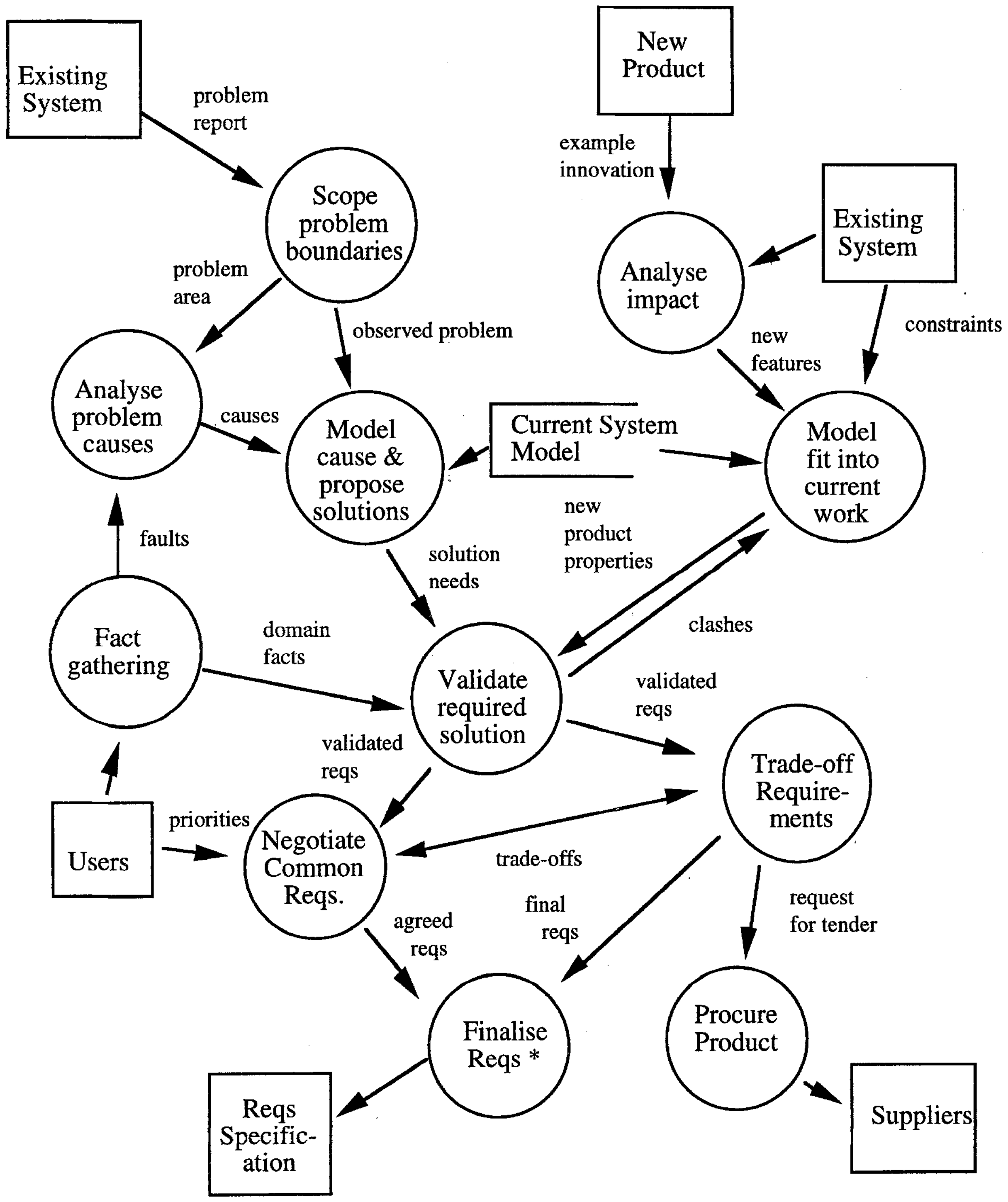

Fig. 3. Process model for requirements discovery: problem and technical innovation routes.

of people, organisational units and tasks that may be effected. Validation is necessary to ensure that the system components to be changed will actually conform with the external requirements. These requirements are often non-functional in nature, e.g. security, usability standards, performance criteria, so trade-off analysis is important for determining how design functions can satisfy the requirement. Some research has been carried out on modelling non-functional requirements in the context of functional goals, tasks and agents, enabling dependencies and options to be inspected [45]. Design rationales are also pertinent for modelling the relationship of non-functional criteria to design options [37]. Overall imposed requirements have not received much attention and most research has concentrated on describing taxonomies of non-functional requirements [15]. At a detailed level, the environment imposes requirements on systems as a consequence of physical laws of nature. An approach to analysing and representing such requirements as formal obligations on the system to process input events has been investigated by Jackson [14]. At a more general level, analysing requirements which are consequent upon events caused by people and legislation is an error-prone process as demonstrated by the failure of complex systems, for instance the London Ambulance Service [62].

3.2.5. Common Activities

All four requirements origins converge on a common pathway, as illustrated in Fig. 4. First, a decision point when the need for change is evaluated. This may lead either to identification of IT opportunities, i.e. new computer systems or changes to an existing one, or managerial decisions about the social system, e.g. resources, organisation and operational procedures. If the IT route is followed the next decision point is whether to develop or procure the new system. The endpoint of requirements analysis may be either a request for tender or a specification for in-house development, or a mixture of both. For in-house development the requirements specification is elaborated with conceptual models drawn from structured methods. For outsourced development the requirements specification may be similar, but with more emphasis placed on non-functional requirements and performance criteria in request for tender documents.

4. RE for Different Target Products

Just as the starting point for RE influences the process model, so does the intended product. Requirements research has tended to assume that the target application will be a bespoke system. This is becoming less common as legacy systems, system evolution and reuse increase their influence. Software is being increasingly developed for configuration, adaptation and reuse as the 'middleware' market increases. Requirements for such software are very different from standard, bespoke applications. The different perspectives of requirements for COTS products versus bespoke developments were explored in an RE 95 workshop [89]. Grudin [90] has also drawn attention to the implications of different product types for requirements analysis, in particular to the problems of accessing users when designing shrink-wrap products. Apart from these investigations RE has not addressed the impact of different product conceptions on the requirements analysis process.

Clearly, different domains have specific impacts on RE, e.g. safety critical applications pose problems not encountered in business applications. Surveying application areas is not feasible given the diversity of computer systems; however, a framework describing the product's relationship to its intended market can facilitate analysing RE issues. The product dimensions are summarised in Fig. 5. Three dimensions characterise the market orientation from narrow targeted applications to wider ranging products, the degree of embeddedness from user services to system automation, and specific products to configurable and reusable components.

Different product conceptions imply different requirements analysis activities. For instance, more horizontal market-oriented products will require market surveys to establish requirements; while reuse libraries need a domain analysis. The product framework provides a starting point for investigating such impacts on requirements process models.

4.1. The Market Dimension

At one end of this dimension, requirements are analysed de novo for a single user and few market considerations constrain the eventual implementation. At the other end, market-led requirements dominate and users are many and harder to identify. Within this dimension there are permutations according to the size and degree of knowledge about the users held by the requirements engineers.

4.1.1. In-house, Bespoke Applications

These imply requirements analysis following the policy route, although the need may also arise from problems in an existing non-automated system. Knowledge of the users and the domain is usually available to the requirements engineers. Little competition exists and the user places the contract for the system development with an in-house supplier. The required system tends to be a vertical market type product, because the system

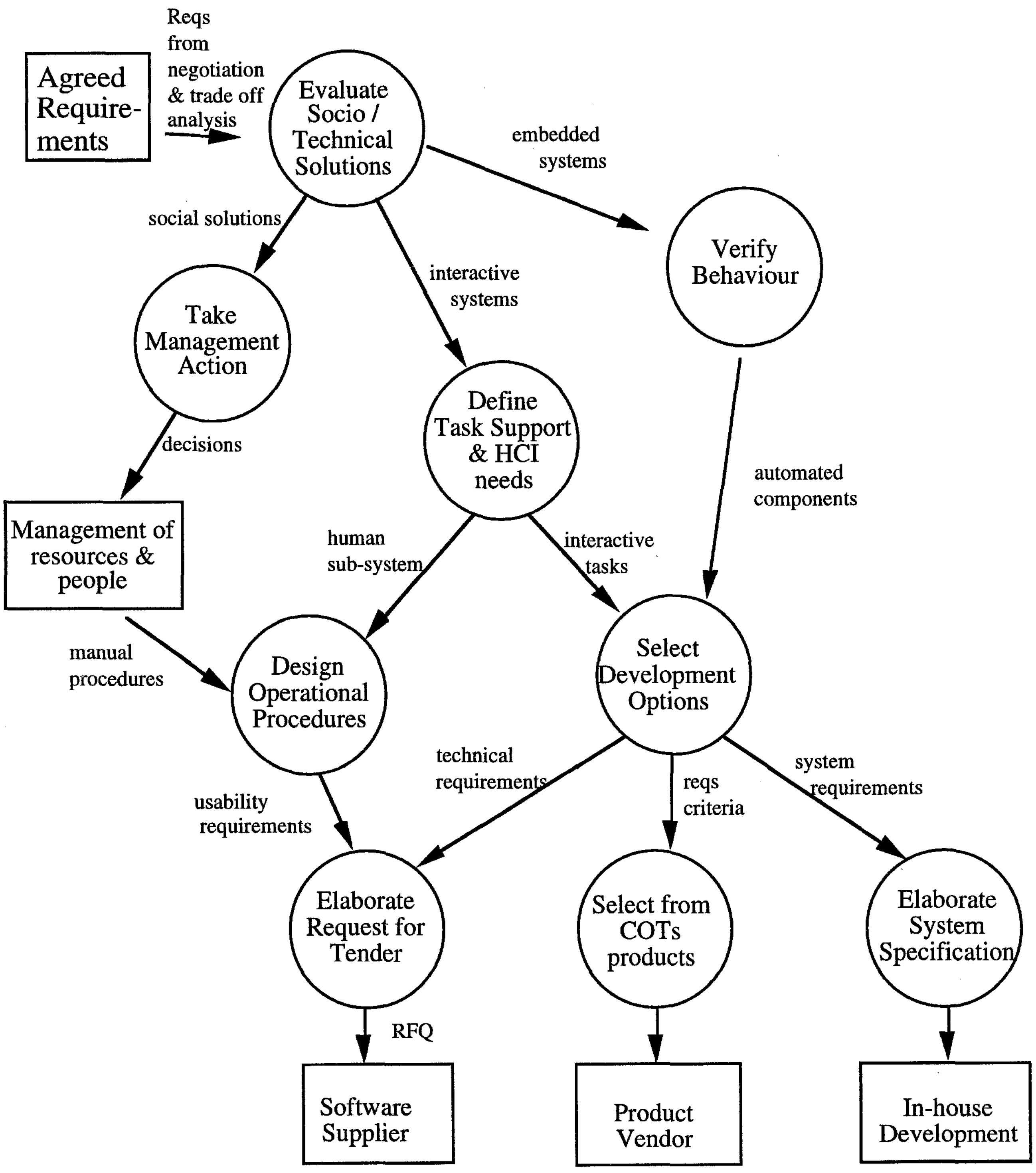

Fig. 4. Common activities and decision points in requirements specification.

has to be tailored to fit into the organisation's business practices. Requirements for major transaction processing or complex technical systems may be detailed and voluminous, although many in-house developments are for smaller information systems. These are being developed more frequently by users themselves with the growth in fourth-generation languages, application generators and rapid prototyping tools. Participatory design and stakeholder analysis methods [28,51] are contributions to bespoke driven developments, although evidence for widespread application of these RE techniques is hard to find.

4.1.2. Contract Applications

These are typically large, expensive systems developed for a specific customer. The principal difference with in-house applications is that the starting point is a 'request for tender' published by the customer. Requirements analysis may be conducted both by the customer before the tender is published and afterwards by the software contractor who will deliver the system. The implications of different contract types, e.g. cost-plus or fixed price, may have interesting consequences for RE. Fixed price contracts limit the freedom of requirements engineers to respond to requirements discovered later in the process; however, such socio-technical implications are rarely considered.

Frequently RE is a collaborative activity between customer and supplier. Knowledge of the user and domain is available to the requirements engineer. In these systems the domain is often complex, but relatively stable. Validation is a key concern because of the mass of technical detail, although few solutions

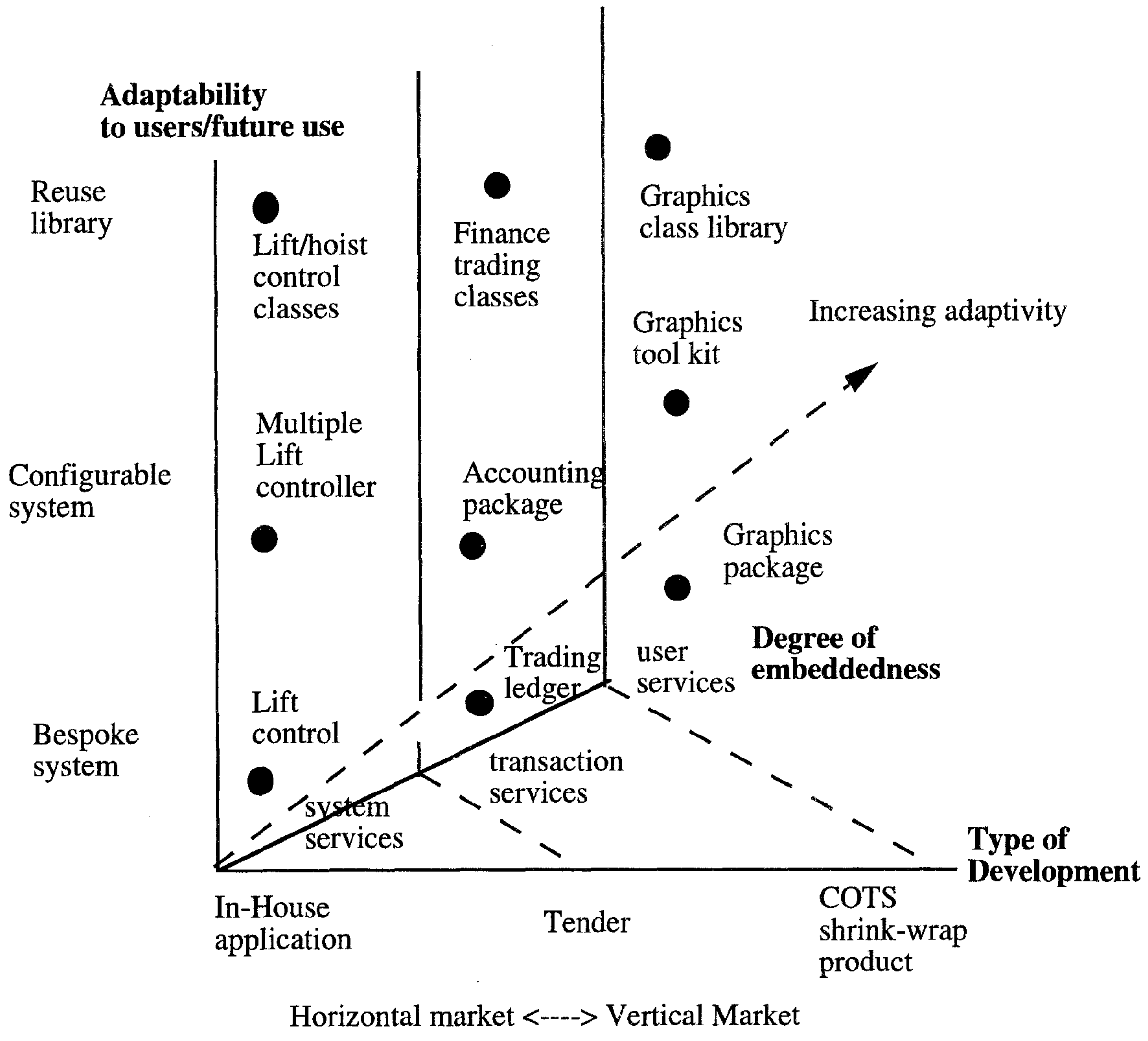

Fig. 5. Dimensions of product conceptions in requirements engineering.

appear to have been found [50]. Management of requirements documentation by traceability tools has been a positive contribution of RE research for these applications [65], and some progress has been made on extracting and summarising requirements semi-automatically from large document libraries [66], but many problems in elaborating and validating requirements for large-scale, technically complex systems are still unsolved.

4.1.3. Commercial Off-the-shelf Products

Requirements are often initiated by the marketing department as an entrepreneur's vision or a request for products in either vertical markets (a specific business area such as manufacturing, banking) or in a horizontal market (a more general product need, such as accounting, word-processing, drawing packages). Requirements discovery is more problematic for shrink-wrap applications as contact between the requirements engineer and prospective users is more difficult. Brainstorming and RAD approaches [28] are frequently used to generate ideas and scope the product. Market surveys can provide further input for scoping and early validation. For horizontal products (e.g. graphics packages) domain knowledge and users may be hard to find; moreover, the context of use may not be known to the requirements engineer. General products have to satisfy a wide range of users in the general public, so scoping and analysis for these products is a challenging and little-understood process. Vertical market products have a more identifiable domain and set of target users, although customisation of off-the shelf software is becoming increasingly necessary, so the problem of requirements analysis with a wide range of, possibly inaccessible, users is also present. Very few reports of research or industrial experience concern shrink-wrap products; for instance, Microsoft's approach appears to be successful but we have no data or understanding about how and why such commercial success is correlated with RE.

4.2. The Specific to Generic Dimension

Most requirements research has focused on a specific application in either in-house or contract type contexts. However, many products aim to be more generic and satisfy a range of users and potential applications.

4.2.1. Configurable Products

These imply some intention by the requirement engineer/developer to produce products which can be customised to different users' needs. The motivation can be either for user interface adaptation so the product delivers appropriate usability according to an individual user profile, or adaptation of functionality to different user groups. The problem is that the range of users and the scope of the domain is more extensive than for bespoke products. This implies analysis not only of a range of required functionality but also of potential user profiles, and the need for configuration tools so users can modify the product's functionality. Little research has been reported in this area, apart from noting configurability as a concern in taxonomies of non-functional requirements [15] and using scenarios as a means of explaining configurability requirements to users [44].

The range and ambition of configurable products depend on the defined market need and the power of the customisation technology. Ultimately this category includes generative systems whereby user requirements are captured in a domain language and the application is generated automatically. However, research prototypes following this approach have a poor track record of commercial success [67]. In a more pragmatic sense fourth-generation languages are a successful generative technology. In this paper we restrict our attention to configurable products where no programming expertise is demanded of the user.

4.2.2 Reuse Libraries

Reuse libraries share many RE problems with configurable products. The main difference is that the user is expected to build the application by composing it from a library of components rather than tailoring an existing application. The RE challenge is to develop 20/20 foresight, i.e. specify wide-ranging requirements for future uses by users that can never be completely known. Users and their domain are rarely accessible to the requirements engineer; moreover, the intended use of software components may be difficult to anticipate.

Little research on requirements for reuse libraries has been reported, although it must be practised by domain analysts [68] who identify requirements for an applications sector (e.g. banking) and build reuse libraries for future applications. Domain analysis methods are not very explicit on requirements capture and validation. The general approach is listing functions and possible 'use cases' (cf. Jacobsen [69]) for future applications before proceeding to design of reusable components, but requirements analysis is still rudimentary. Reuse-driven development may also involve customisation of the component's code. More likely is the need to develop some application-specific 'glue' code to coordinate a set of reused components.

From the requirements viewpoint the new challenges are to discover the range of functionality a library needs to contain, and then define the scope and granularity of reusable components. For instance, a component in a banking library may perform amortisation calculations; however, the compound interest algorithm within it may be reused in a wider range of target applications in finance and elsewhere. Generic requirements models for application classes [70] may provide a future solution, although the utility in reusing generic models has yet to be demonstrated in practice. Some encouragement may be taken from the success of the reusable sector models from the ARIS system [71]. These reusable systems are limited to a particular functional area of business, e.g. logistics, accounting, manufacturing, and are closer to configurable systems than reuse libraries; nevertheless, they do demonstrate the potential for reusable products resulting from a thorough, enterprise-based domain analysis. More generally, the concerns of abstraction, scope and granularity in requirements for reusable components are complex research topics which are only just starting to be addressed.

4.3. The Service Dimension

This dimension characterises how the target product relates to the real world within which it will be embedded. Systems that support users' tasks, such as decision support, command and control, and ubiquitous applications such as word-processing, imply requirements driven from analysis of users' work. Task analysis methods, e.g. knowledge analysis for tasks [72], can help to model users' activity, but deriving requirements from task models is still largely an intuitive exercise. Some guidance is given for defining task support requirements in integrated methods for HCI and software engineering [44,64] but this has not been elaborated for requirements definition or validation. Transactional systems have less user interaction and more automated processing. These systems are driven by business needs to process information, so they may be considered as serving organisations rather than individual users. Requirements for these systems are often expressed as business rules [25], which are elaborated into conceptual models for information systems. Business applications and small to medium-scale information systems may be successfully developed using prototyping approaches combined with cut-down versions of structured analysis and design methods [73]. However, such approaches are not so effective for complex and embedded systems which interface with other mechanisms and automata, e.g. software control systems in car engines, aircraft fly-by-wire systems. Requirements for embedded systems are frequently safety critical and have exacting standards of accuracy, hence more formal approaches to requirements are appropriate. Examples of these approaches are the work of Jackson [12] on concepts of designation and obligations in requirements for correct system behaviour and constraint-based specifications of van Lamsweerde *et al.* [38].

4.4. Impact of Products on RE Activities

Product types focus attention on particular activities in the RE process pathways (see Figs 2 and 3). For instance, interactive products imply more attention to specification of task support requirements and human computer interface design, whereas embedded systems may be amenable to formal verification. Embedded systems frequently have safety critical properties (e.g. brake and engine control systems in cars); since it is important that the behaviour of such systems is rigorously verified against their requirements, more formal approaches to the elicitation of requirements and system obligations are appropriate [12].

In-house systems necessitate participative analysis and modelling activity. However, products which are open to tender imply that negotiation, trade-off analysis and prioritisation are more critical than for in-house RE (see Fig. 6). Indeed, requirements will unfold and be negotiated as the tender documents are drawn up; moreover, understanding, prioritisation, and requirements elaboration will continue once the contract has been awarded. Explanation and communication support, such as simulation by concept demonstrators, are necessary for tendering.

All types of products may have previous automated versions which have to be taken into account when developing a new application. Legacy systems influence RE in a similar manner to requirements by example in that the legacy system imposes a set of requirements, or more effectively constraints, on the new development. Requirements for the new system may be constrained by the need to ensure backward compatibility; for instance, with the user interface look and feel for consistency, or with data formats for database compatibility. When a technical system already exists, it has to be understood in relation to how software applications are being used by people, i.e. a socio-technical system. Requirements have to be understood in the context of existing systems as well as from the perspective of stakeholders' goals for new systems. Reverse engineering methods may help recovery of requirements from designed systems, but experience has shown that it is difficult to extract design intent from code, even when some documentation exists. Analysing requirements in tandem with assessing the impact of legacy systems appears to be an untouched area of research.

COTS products raise different problems for scoping and analysis activities. Users are hard to find, so RE has to rely on surveys and these products necessitate more emphasis on scoping by surveys and market assessment. COTS type products are often driven by a single designer's vision and requirements result from brainstorming [7]. Ideally requirements should be validated with users but commercial sensitivities, e.g. the need for secrecy, can prevent this. Furthermore, it is difficult to get good feedback from questionnaires and market exercises. One approach is to use mock-ups and concept demonstrators, then place these in public places and observe user reactions [40].

Although RE for generic and reusable products shares the same set of activities with conventional applications, some pathway specialisations are necessary. The main pathway for configurable products is assumed to be policy driven, although other initiations

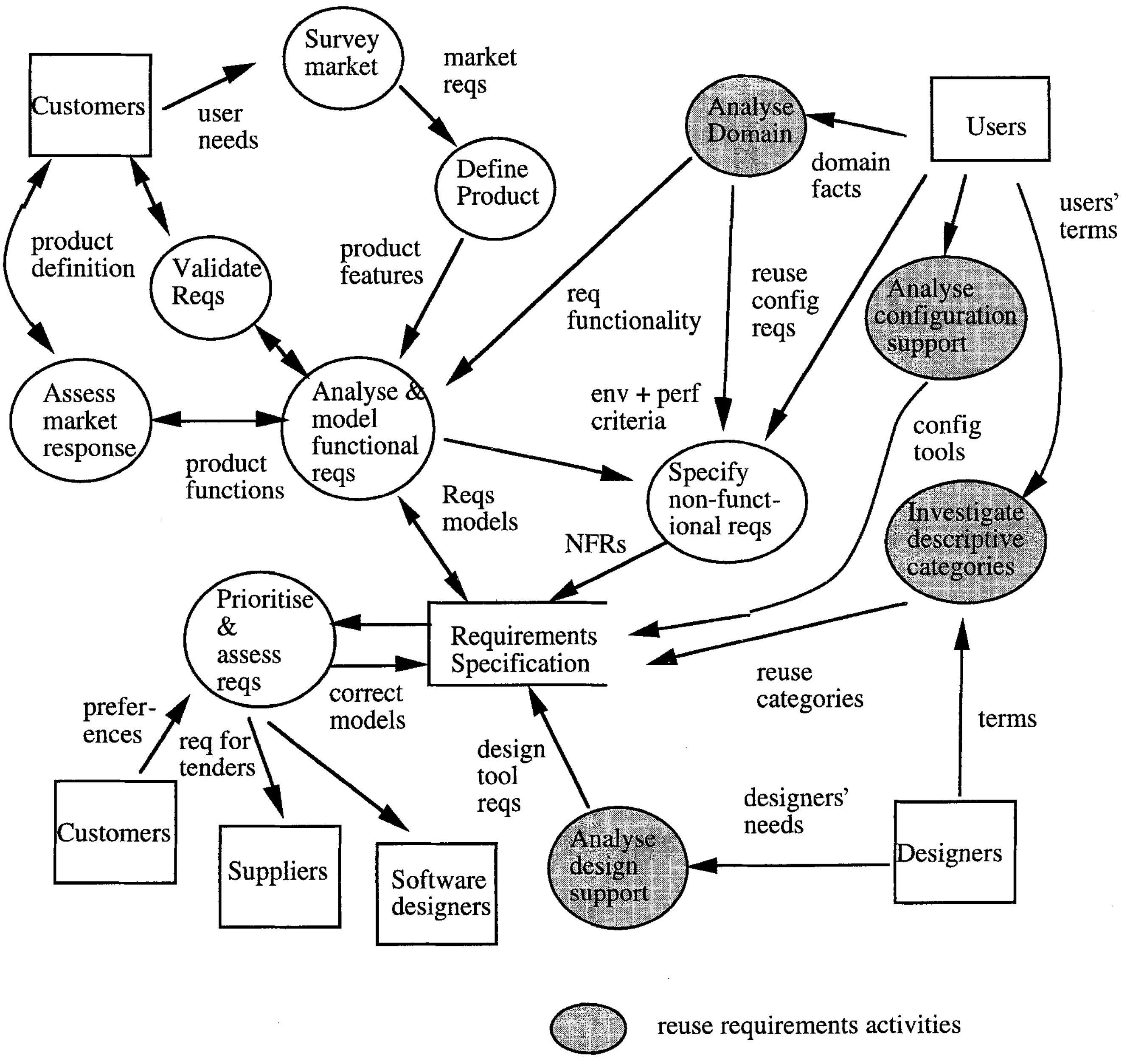

Fig. 6. Requirements engineering activities oriented to off the shelf product development and reusable component libraries.

are conceivable if users discover examples of new features which they want. Requirements analysis follows a similar path to the policy-driven route illustrated in Fig. 2, but the starting point is a market survey or high-level scoping of the product area. Fact gathering, eliciting users' goals and validating requirements can be difficult as users may not be accessible, and even if they are, their future needs may be hard to articulate. A range of functionalities have to be considered, so requirements acquisition may proceed via surveys as well as more standard techniques such as interviews. Validation in a true sense can only be achieved by actual use of the component library, although testing the completeness of a domain analysis may be possible by expert judgement. Validation and trade-off analyses for non-functional requirements may also have to be carried out using indirect techniques such as questionnaires and market surveys.

Analysis for configurable/reusable software has to consider not only the functionality of the product components but also requirements for supporting software which will help users/designers to customise components for new applications. For reuse library products, the pathway for requirements discovery starts with the problem of defining the marketplace for a reuse library. Figure 6 summarises the pathways for reusable and configurable RE, showing additional activities for surveying and analysis of configuration tools. This involves first deciding what the target domain for reuse is and then conducting a domain analysis of existing user activities, automated functionalities, etc. A range of functionalities and services have to be investigated to analyse not only requirements but also all the identifiable components in the domain. Domain analysis methods for this purpose [74] follow systems analysis approaches but are more exhaustive, although there is only rudimentary guidance about what and how much should be analysed.

Designers are a different stakeholder group who need powerful yet easy-to-use customisation and composition tools. Requirements for user guidance and support tools for configuring the target system have to be investigated, with the necessary user/system models, parameter files, and their editing/configuration interfaces. This activity involves modelling users' expectations in terms of how much work they expect to do to tailor a configurable system to their needs. Other concerns may be the need for explanation facilities to help users derive the optimal potential from complex configurable products. Taken to an extreme this approach becomes similar to RE for software development products, i.e. generator tools and programming languages.

Projections of future usage are hard to achieve since requirements may only be determined within the context of eventual use. For instance, larger design architectures may fit the requirements for a vertical market sector (e.g. financial dealing systems), although components within these architectures may have more widespread target applications, e.g. forecasting and amortisation algorithms could be exploited in a variety of financial applications and elsewhere. Sector models and templates for configuration information systems have had some success [71], but these are limited to well-defined vertical markets, e.g. banking, accounting. Design of reusable components at different levels of granularity and abstraction needs further research, since there are currently few guidelines to inform such decisions.

In reuse libraries requirements also concern non-functional criteria such as ease of construction, portability, interoperability, and maintainability. These in turn become further functional requirements for developing configuration support tools, development environments and software harnesses for promoting ease of reuse. Reuse in RE has to solve the dilemma of providing material which has sufficient utility to help a requirements engineer without unnecessarily prescribing the developer's view of a new system. Furthermore, reuse can encourage a 'copy-cat' approach leading to errors, so its use within RE will have to be treated with care [75]. Further research is necessary on methods and tools to help developers assimilate and understand the documents and models produced by others. The other key research issue is scaling up to provide significant libraries of generic components for realistic domains.

5. Discussion

RE has a considerable contribution to make to software development in a broader sense. One of its strengths is an eclectic approach manifest in a range of influences from ethnography, design rationale and scenarios. RE opens the debate about different conceptions of design ranging from the formal, methodical and prototype based. One potential benefit may be the emergence of a meta-theory of design which acknowledges that applications are diverse and that a battery of techniques is necessary to address different requirements needs. This trend has been noted in specification languages and methods with the argument that appropriate techniques need to be matched to different applications. The pathways and product dimensions we propose may render some service in mapping the diversity of issues which needs to be addressed.

While RE has been strong on modelling techniques and addressing issues such as traceability, it has been less active in providing process guidance. Notable exceptions are the inquiry cycle of Potts *et al.* [33], which does give a systematic way of validating requirements from scenarios, while the modelling work of Chung [45] and Yu [58] does embed process guidance for analysis and treating of non-functional requirements. However, there is little guidance about which techniques to apply for eliciting or validating a particular type of requirement. Indeed, there are few surveys of appropriate methods for requirements analysis beyond the traditional offerings of systems analysis, e.g. interviews, observations and analysis of documentation [29]. In a recent survey of RE techniques Maiden and Rugg [21] point out the applicability of knowledge engineering techniques to requirements problems. Ethnomethodology has attracted considerable attention, but most reports are short on prescriptive guidance for carrying out ethnographic studies and somewhat longer on their justification. Ethnomethodology has yet to deliver a significant design impact in spite of several studies [30,31] and no cost–benefit analysis of employing such techniques has been reported. Although ethnography has pointed out the importance of context [13], its role in providing more prescriptive advice for the social dimension of requirements analysis must be subject to some doubt.

A variety of other techniques has emerged for requirements-related activities, for instance, design rationale and scenario analysis. These have produced some encouraging results in industrial trials [42,85]; however, their impact in wider practice has yet to be proven. Informal representation of requirements knowledge in hypertext tools [17] and design rationale graphs is accepted as a necessary means of supporting RE; unfortunately transforming such representations into more formal models is not so clear. Industrial experience indicates that even the informal design rationale (DR) notation may need simplification [85].

Participative and user-centred design techniques have existed for some time [81,82], but these approaches do not seem to have solved RE problems even though they have highlighted a critical success factor, namely involving users early and continuously in the requirements and design process. Prototyping with a methodical approach has been taken up in industry, as exemplified by rapid application development [28]. This approach provides outline guidance and a general framework for approaching requirements analysis including user participation. However, RAD methods are short on precise advice and tend to focus on setting up the environment by workshops for joint development rather than offering detailed techniques to solve problems.

So far no comprehensive method or process model for RE has been proposed. Indeed, as Rolland [86] points out, RE is by its nature ill structured, so a situation-driven approach may be more appropriate. Rolland's process meta-model allows method guidance to be driven from the product context, i.e. appropriate techniques which may be applied to a developing requirements specification are selected according to its current state. More structured guidance has been produced by the F3 project which has provided a road map of the overall RE process that shows some similarities with this paper [27]. However the F3 models do not account for different initiations of requirements or the product context, and focus primarily on the policy/goal modelling.

Goal modelling in a bespoke development context is the conventional approach for RE research. Although some promising new directions are emerging from inquiry-based approaches with scenarios [33], we are not much further forward than top-down functional decomposition techniques advocated by structured systems analysis [76]. Formal approaches to goal modelling [38,87] offer a means of expressing intentions as constraint-based specifications of system behaviour. This approach shows promise of making requirements definition more reliable, but it is dependent on linguistic articulation of detailed requirement as a bridge to formalisation. Furthermore formal methods do not address the need to refine ambiguous natural language expressions by analytic processes and how the detailed understanding necessary for formalisation may be achieved by user–analyst dialogue. The inquiry cycle [32,33] is one way to bridge that gap. Further research is necessary to provide process guidance for goal modelling and integrate it with scenario-based analytic techniques.

There is a growing realisation that goals are better understood in the context of organisational models [20] and some attempts have been made to provide schemas for representing goal-oriented requirements knowledge [45,88]. The enterprise modelling approach of Chung combines informal graphics with formalisation of high-level concepts such as responsibility and dependency to provide automated reasoning about relationships between goals, tasks, and non-functional requirements with an accessible notation. An open question is how far RE research should progress into the business analysis area. Some research refers to business process engineering; for instance Yu [58] has extended the dependency modelling approach of Chung for business process modelling, but there are few serious methodological recommendations. Whether enterprise and organisational modelling can address issues of diagnosing opportunities for computer technology to enhance competitive advantage remains to be seen.

In conclusion, RE research has produced useful results from formal expression of requirements as an extension of system specification and design, to process models for requirements capture and validation. There is considerable potential for RE research to update commercial system development methods, even though reports of application are still rare. Further research is necessary on process guidance and linking the boundaries of informal and formal representation. Finally, the bounds of the subject in terms of requirements analysis for innovative business solutions and requirements for different products have yet to be fully determined.

Acknowledgements

This research is funded as part of the European Union ESPRIT III 6353 NATURE basic research action. The author wishes to thank all his colleagues on the project for input and comments on this paper.

References

1. Bell TE, Thayer TA. Software requirements: are they really a problem? In: Proceedings of the 2nd international conference on software engineering, 1976, pp 61–68
2. Thayer R., Dorfman M. System and software requirements engineering. IEEE Computer Society Press, 1990
3. IEEE-TSE. Special issue on requirements engineering. IEEE Trans Software Eng 1991; 17(3)
4. IEEE-TSE. Special issue on requirements engineering. IEEE Trans Software Eng 1992; 18(6)
5. Finkelstein ACW, Fickas S (eds). Proceedings of IEEE symposium on requirements engineering. IEEE Computer Society Press, 1993
6. Davies AM, Hsai P (eds). Proceedings of the 1st International conference on requirements engineering. IEEE Computer Society Press, 1994
7. Lubars M, Potts C, Richter C. A review of the state of the practice in requirements modelling. In: Fickas S, Finkelstein ACW (eds). Proceedings of RE'93. IEEE Computer Society Press, 1993, pp 2–14
8. El Emam K, Madhavji NH. A field study of requirements engineering practices in information systems development. In: Harrison MD, Zave P (eds). Proceedings of RE '95. IEEE Computer Society Press, 1995, pp 68–80
9. Sommerville I. Software engineering. Addison Wesley, Reading, MA, 1989
10. Bubenko J. Extending the scope of information modelling. SISU, Stockholm, 1993
11. Dubois E, Hagelstein J, Rifaut A. Formal requirements engineering with ERAE. Philips J Res 1989; 43(4): 393-414
12. Jackson M, Zave P. Domain descriptions. In: IEEE symposium on requirements engineering. IEEE Computer Society Press, 1993, pp 56–64
13. Goguen J, Linde C. Techniques for requirements elicitation. In: Proceedings of the 1st international symposium on requirements engineering. IEEE Computer Society Press, 1993, pp 152–164
14. Jackson M. Software requirements and specifications. Addison Wesley, Reading, MA, 1995
15. Keller SE, Kahn LG, Parna RB. Specifying software quality requirements with metrics: tutorial paper. In: Thayer RH, Dorfman M (eds). System and software requirements engineering. IEEE Computer Society Press, 1990, pp 145–163
16. Davies AM. Software requirements: object functions and states. Prentice-Hall, Englewood Cliffs, NJ, 1993
17. Pohl K. Process centered requirements engineering. Wiley, Chichester, 1996
18. Feather MS, Fickas S, Helm R. Composite system design: the good news and the bad news. In: Proceedings of the 6th knowledge-based software engineering conference, Syracuse, NY. IEEE Computer Society Press, 1991, pp 16–25
19. Goguen JA. Social issues in requirements engineering. In: Proceedings of IEEE symposium on requirements engineering. IEEE Computer Society Press, 1993, pp 194–195
20. Harker SDP, Eason KD, Dobson JE. The change and evolution of requirements as a challenge to the practice of software engineering. In: IEEE symposium on requirements engineering, RE '93, San Diego, CA. IEEE Computer Society Press, 1993, pp 266–272
21. Maiden NAM, Rugg G. ACRE: Selecting methods for Requirements Acquisition. Software Engineering Journal 1996; 11(3): 183-192
22. Pohl K. The three dimensions of requirements engineering. In: Proceedings of CAiSE '93, Paris. Springer-Verlag, Berlin, 1993
23. Jackson M. Problems, methods and specialisation. Software Eng J 1994; 9(6): 249-255. Special issue on software engineering in the year 2001
24. Roman G. A taxonomy of current issues in requirements engineering. IEEE Comput 1985; April: 14–22
25. Loucopoulos P, Karakostas V. System requirements engineering. McGraw-Hill, London 1995
26. Zave P. Classification of research efforts in requirements engineering. In: Harrison MD, Zave P (eds). Proceedings of RE '95: second international symposium on requirements engineering. IEEE Computer Society Press, 1995, pp 214–216
27. Kirikova M, Bubenko JA. Enterprise modelling: improving the quality of requirements specifications. Olou, Finland, 1994
28. DSDM-Consortium. Dynamic systems development method. Tesseract, Farnham, UK, 1995
29. Gause D, Weinberg G. Exploring requirements. Dorset House, New York, 1989

30. Luff P, Jirotka M, Heath C, Greatbatch D. Tasks and social interaction: the relevance of naturalistic analyses of conduct for requirements engineering. In: IEEE symposium on requirements engineering. IEEE Computer Society Press, 1993, pp 187–190
31. Hughes J, O'Brien J, Rhodden T, Rouncefield M, Sommerville I. Presenting ethnography in the requirements process. In: Zave P, Harrison MD (eds). Proceedings of RE '95, Second international symposium on requirements engineering. IEEE Computer Society Press, 1995, pp 27–34
32. Potts C, Takahashi K, Anton A. Inquiry based requirements analysis. IEEE Software 1994; March: 21–32
33. Potts C, Takahashi K, Smith J, Ora K. An evaluation of inquiry based requirements analysis for an Internet service. In: Zave P, Harrison MD (eds). Proceedings of RE '95, second international symposium on requirements engineering. IEEE Computer Society Press, 1995, pp 27–34
34. Sutcliffe AG, Maiden NAM. Bridging the requirements gap: policies, goals and domains. In: Proceedings of the 7th international workshop on system specification and design. IEEE Computer Society Press, 1993, pp 52–55
35. Checkland P. Systems thinking, systems practice. Wiley, Chichester, 1981
36. Conklin J, Begeman ML. gIBIS: a hypertext tool for exploratory policy discussion. ACM Trans Office Inform Syst 1988; 6(4): 303–331
37. MacLean A, Young RM, Belotti VME, Moran TP. Questions, options and criteria: elements of design space analysis. In: Carroll JM, Moran TP (eds). Human–computer interaction 1991; 6(3,4): 201-250. Special issue on design rationale
38. van Lamsweerde A, Darimont R, Massonet P. Goal directed elaboration of requirements for a meeting scheduler: problems and lessons learnt. In: Harrison MD, Zave P (eds). Proceedings of RE '95. IEEE Computer Society Press, 1995, pp 194–203
39. Attwood ME, Burns B, Girgensohn A, Lee A, Turner T, Zimmerman B. Prototyping considered dangerous. In: Nordby K, Helmersen PH, Gilmore DJ, Arnesen SA (eds). Proceedings of human computer interaction: INTERACT '95. IFIP/Chapman & Hall, London, 1995, pp 179–184
40. Gould JD. How to design usable systems. In: Bullinger H-J, Shackel B (eds). Proceedings INTERACT '87. North-Holland, Amsterdam, 1987
41. Maiden NAM, Sutcliffe AG. Requirements critiquing using domain abstractions. In: Siddiqi J (ed). Proceedings of the 1st international conference on requirements engineering, IEEE Computer Society Press, 1994, pp 184–193
42. Carroll JM, Alpert SR, Karat J, Van Deusen M, Rosson MB. Raison d'être: capturing design history and rationale in multimedia narratives. In: Adelson B, Dumais S, Olson J (eds). Proceedings of CHI '94: human factors in computing systems. ACM Press, 1994, pp 192–197
43. Johnson WL, Feather MS, Harris DR. Representation and presentation of requirements knowledge. IEEE Trans Software Eng 1992; 18(10): pp 853–869
44. Sutcliffe AG. Requirements rationales: integrating approaches to requirements analysis. In: Olson GM, Schuon S (eds). Proceedings of Designing Interactive Systems, DIS '95. ACM Press, 1995, pp 33–42
45. Chung L. Representing and using non-functional requirements: a process-oriented approach. Department of Computer Science, University of Toronto, 1993
46. Yu ESK. Modelling organisations for information systems requirements engineering. In: Finkelstein ACW (ed). Proceedings of IEEE symposium on requirements engineering, RE '93, San Diego, CA. IEEE Press, 1993, pp 34–41
47. Hauser J, Clausing D. The house of quality. Harvard Business Rev 1988; 5: 63–73
48. Jacobs S, Kethers S. Improving communication and decision making within quality function deployment. In: Proceedings of the 1st international conference on concurrent engineering, research and application, Pittsburgh (Also in NATURE Report Series 94–13), 1994
49. Fenton NE. The role of measurement in software safety assessment. In: Proceedings of CSR/ENCRESS conference, Bruges, September 1995. Springer-Verlag, Berlin, 1995
50. Boehm B, Bose P, Horowitz E, Lee MJ. Software requirements as negotiated win conditions. In: Proceedings of IEEE conference on requirements engineering. IEEE Computer Society Press, 1994, pp 74–83
51. Macaulay L. Requirements capture as a cooperative activity. In: IEEE symposium on requirements engineering. IEEE Computer Society Press, 1993, pp 174–181
52. Bowers J, Viller S, Rhodden T. Human factors in requirements engineering. Department of Computer Science, University of Lancaster, 1994
53. Porter ME. Competitive strategy. Free Press, New York, 1980
54. Davenport T. Process innovation: re-engineering work through information technology. Harvard Business School Press, Boston, 1993
55. Eden C. Cognitive mapping. Eur Operational Res, 1988; 36: 1–13
56. Rockart JF, Short JE. The networked organisation and the management of interdependence. In: Scott-Morton M (ed). The corporation in the 1990s: information technology and organisational transformation. Oxford University Press 1991, pp 189-219
57. Holland CP. Cooperative supply chain management: the impact of inter-organisation information systems. J Strat Inform Syst 1995; 4(2): 117–133
58. Yu ESK. Modelling strategic relationships for process reengineering. Department of Computer Science, University of Toronto, 1994
59. Swain AD, Weston LM. An approach to the diagnosis and misdiagnosis of abnormal conditions in post accident sequences in complex man machine systems. In: Goodstein L, Andersen H, Olson S (eds). Tasks, errors and mental models. Taylor & Francis, London, 1988
60. Beer S. The brain of the firm. Wiley, Chichester, 1981
61. Nielsen J. Usability engineering. Academic Press, Boston, 1993
62. HMSO. Report of the Inquiry into the London Ambulance Service. HMSO, London, 1993
63. Sutcliffe AG. Human computer interface design, 2nd edn. Macmillan, London, 1995
64. Lim KY, Long JL. The MUSE method for usability engineering. Cambridge University Press, Cambridge, UK, 1994

65. Ramesh B, Dhar V. Supporting systems development by capturing deliberations during requirements engineering. IEEE Trans Software Eng 1992; 18(6): 498–510
66. Goldin L, Berry D. Abstfinder, a prototype abstraction finder for natural language text for use in requirements elicitation: design, methodology and evaluation. In: Proceedings of IEEE conference on requirements engineering. IEEE Computer Society Press, 1994, pp 84–93
67. Neighbors J. An assessment of reuse technology after ten years. In: Frakes WB, (ed). Proceedings of the 3rd international conference on software reuse: advances in software reusability. IEEE Computer Society Press, 1994, pp 6–13
68. Prieto-Diaz R. Implementing faceted classification for software reuse. Commun ACM 1991; 34(5): 88–97
69. Jacobsen L. Object oriented development in an industrial environment. In: Proceedings of OOPSLA '87. ACM Press, 1987, pp 183–191
70. Sutcliffe AG, Maiden NAM. Domain modelling for reuse. In: Frakes WB (ed). Proceedings of the 3rd international conference on software reusability. IEEE Computer Society Press, 1994, pp 169–173
71. Scheer AW. Enterprise-wide data modelling. Springer-Verlag, Berlin, 1994
72. Johnson P. Human computer interaction. McGraw-Hill, London, 1995
73. Crinnion J. Evolutionary systems development: a practical guide to the use of prototyping within a structured systems methodology. Pitman, London, 1991
74. Prieto-Diaz R. Domain analysis: an introduction. ACM SIGSOFT Software Eng Notes 1990; 15(2): pp 47–54
75. Sutcliffe AG, Maiden NAM. How specification reuse can support requirements analysis. In: Hall P (ed). Proceedings of Software Engineering '90. Cambridge University Press, Cambridge, UK, 1990, pp 489–509
76. De Marco T. Structured systems analysis and specification. Prentice-Hall, Englewood Cliffs, NJ, 1978
77. Jackson MJ. Systems development. Prentice-Hall, Englewood Cliffs, NJ, 1983
78. Coad P, Yourdon EE. Object oriented analysis. Yourdon Press, New York, 1991
79. Rumbaugh J. Object oriented modelling and design. Prentice-Hall, Englewood Cliffs, NJ, 1991
80. Yourdon EE. Modern structured analysis. Prentice-Hall, Englewood Cliffs, NJ, 1989
81. Mumford E. Designing participatively. Manchester Business School Publications, Manchester, 1983
82. Eason KD. Information technology and organisational change, Taylor & Francis, London, 1988
83. Gough PA, Fodemski FT, Higgins SA, S.J. R. Scenarios: an industrial case study and hypermedia enhancements. In: Harrison M, Zave P (eds). Proceedings of RE '95. IEEE Computer Society Press, 1995, pp 10–17
84. Pfleeger S, Hatton L. How do formal methods affect code quality? Centre for Software Reliability, City University, London, 1994
85. Bellotti V, Buckingham-Schum S, Maclean A, Hammond N. Multidisciplinary modelling in HCI design in theory and practice. In: Proceedings of CHI '95. ACM Press, 1995, pp 146-–153
86. Rolland C. Modelling the evolution of artifacts. In: IEEE Conference on requirements engineering. IEEE Computer Society Press, 1994, pp 216–219
87. Dardenne A, van Lamsweerde A, Fickas S. Goal directed requirements acquisition. Sci Comput Program 1993; 20: 3–50
88. Mylopoulos J, Chung L, Nixon B. Representing and using non functional requirements: a process-oriented approach. IEEE Trans Software Eng 1992; 18(6): 483–497
89. Potts C. Invented requirements and imagined customers: requirements for off-the-shelf software. Briefing for working group 2. In: Proceedings of RE '95. IEEE Computer Society Press, 1995, pp 128–130
90. Grudin J. Systematic sources of suboptimal interface design in large product development organisations. Human Comput Interact 1991; 6: 147–196
91. Gotel OCZ, Finkelstein ACW. An analysis of the requirements traceability problem. In: First international conference on requirements engineering. IEEE Computer Society Press, 1994, pp 94–101
92. Finkelstein ACW, Kramer J, Nuseibeh B. Viewpoints: a framework for integrating multiple perspectives in system development. Int J Software Eng Knowledge Eng, 1992; 2(1): 31–57