ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, January 2013

A Survey on Emerging trends in Requirement engineering for a Software development Life cycle

Swarnalatha K. S [1], **Dr G N Srinivasan** [2]

Department of Computer Science and Engineering, R.V College of Engineering, Bangalore, Karnataka, India [1]

Department of Information Science and Engineering, R.V College of Engineering, Bangalore, Karnataka, India [2]

ABSTRACT: **Requirement Engineering is an important phase of any software development. 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.**

Keywords: **Requirement Engineering, SDLC, Value Modelling, Gathering, Orientation**

I. 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 categorization 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. "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 [1]" .Put crudely requirements engineering focuses o n improvements to the front-end of the system development life-cycle. Establishing the needs that have given rise to the development process and organizing 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 [2]. The most direct route to improving system quality, therefore to ensure that requirements are accurately determined and that a requirements focus is maintained through the development process. The 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 [2]. As development precedes 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 localize as they spread across components of the system [2].

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 t o control the requirements and systems engineering process; communication and documentation of the results of requirements engineering.

II. CONTEXTS

Precondition for effective requirement 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 disorganized development process. In other words it is important that developers do not run before they can walk! Because organizational distance can dim the "voice of the customer"[3] in the subsequent development process, requirements engineering effort is particularly susceptible to wastage. 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 neither 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. 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 [4]. Requirements engineering research has been no better than any other area of software and systems engineering in this regard.

Issue: 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 t o 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.

Organizational setting

Requirements engineering can take place in many organizational settings. The development process may be: internal to an organization, where the system is being produced by that organization for its own use; bespoke, where a client requests another organization to produce a system specific to its requirements; customization 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 organization develops a product to be placed in a perceived market. Each of these settings confers slightly different responsibilities and in each case suggests different technical priorities. The issue of organizational context and its ramifications for the organization of system development has, been neglected in software engineering. The information systems community has, by contrast, recognized this issue [7] and has attempted to make the assumptions about organizational 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 organizations and contractors who have been most articulate about their difficulties and who have funded software engineering research. There is a growing recognition of the importance of system customization and extension cynics might suggest that this reflects the tougher stance of defence procurement agencies. There is virtually no work o n the support for developing products for markets though this concern is surfacing within general debate, in particular through large telecommunications organizations who, since deregulation, now deliver services into a global competitive marketplace.

Contract and procurement procedures

In many organizational 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. 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. Generally however, contractual and procurement matters are regarded as organizationally specific or otherwise out of the scope of requirements engineering. 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 colored 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 t o 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.

III. GROUNDWORK

Bounding

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 from 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. The issue of bounding is one of the thorniest in requirements engineering. There have been very few attempts to tackle it head on [9]. It appears relatively straightforward in any given case to draw bounds but to give general guidance o n how to make bounding decisions is very difficult. 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

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. 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. The area of risk as it relates to system development is attracting attention and there have been some significant contributions to this literature 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 [11].

IV. ACQUISITIONS

STAKEHOLDER ANALYSIS

The process of stakeholder analysis involves identifying those individuals or roles that 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 organizational relations between them. This analysis serves as a map for subsequent information gathering and a means of interpreting the information provided and its status. There are two threads making up current work o n stakeholder analysis. The first thread arises out of work o n viewpoint-based methods in which viewpoints are tied to client authorities responsible for information provided within those viewpoints. The second thread arises from work on enterprise modelling in which identifying stakeholders is part of the process of modelling the organizational environment in which the system is to be placed. The work in this area needs to be brought together. From the method thread - organized 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

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. There has been a significant body of work o n participation and cooperation in requirements engineering. The most significant of this is the work on participative, joint or facilitated systems development. This work is related to, but distinct from, work on the application of "groupware" to requirements engineering The shortcomings, and hence the research issues, in this area are mostly common to the study of cooperative work in general. 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 t o have real application in a complex task such as requirements engineering.

Information gathering

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, and may be buried in the work practices of individual users, and so on. 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. It is already evident that any realistic domain requires a judicious selection and combination of techniques. How t o make such a selection and combination is however far from clear. There is clearly significant scope for further work in this area.

V. MODELLING

Value modelling

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 document and relates these values. It is unsurprising, given that the literature o n software engineering does not recognize 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. The application of multi-criteria decision making techniques in software development as a whole is an interesting open area. Techniques such as QFD which have been extensively applied to manufacturing development may be transferable.

Modelling goals and required services

The core of requirements engineering, and the primary means by which the needs are rendered in a form that can be used to realize 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. . 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.

Task analysis

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 t o suggest ways in which the interface to the system can be organized to have a better fit with their tasks and the way in which they understand the system and its properties. 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). 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, not been treated as part of a larger systems engineering process. Work on method integration is required and may

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, January 2013

yield substantial economies as information collected in task models is obtainable elsewhere in the requirements engineering process. Further work o n identifying relevant aspects of human task performance that can be derived from task models is also required [10].

Reus e

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 organizing the process of requirements engineering so as to enhance the opportunities for subsequent reuse. Obviously the use of modelling schemes, employing inheritance or the like, which are capable of supporting reuse have attracted attention within requirements engineering [12]. 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 .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.

VI. ANALYSIS

Validation

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 t o 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. Ideally validation should be as tightly tied to and interleaved with requirements production as possible. However organizational factors can intervene to prevent this. In such cases the validation 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

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. 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 realization; and, how to guide exploration and organize feedback. Despite the amount of work in the area these difficulties remain as unresolved research issues.

Verification

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.. Software development orthodoxy sets down that at each stage in software development you should be able t o prove that the specification (however construed) you have developed is secure with respect to the preceding specification. 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 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

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 [15]. Inspection is proven to work; it is simple and widely used throughout industry. It should follow from this that there is widespread research interest. 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 there is however considerable scope for further work in this area.

VII. MEASUREMENT

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. 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. 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.

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. Much of what has been said above for metrics applies to estimation which adds to the challenges of measurement those of predictive models.

VIII. DOCUMENTATION

Information management

The requirements engineering process produces large amounts of richly interrelated technical information and documentation. Some of this is textual, some graphic (drawings and diagrams)[13]. Storage and retrieval and production of high quality, tailored documentation is of considerable practical importance. 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. In the future we can expect to add video and sound records of technical meetings and document annotations.

Recording rationale and argumentation

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 80% of costs are in rework and half the efforts 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. 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

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. 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 visualize navigate these links. In this assessment the issue of "pre-requirements traceability" is highlighted. In particular the problems of linking artefacts produced during requirements engineering t o 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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, January 2013

worthy of further research attention.

Standards and Conformance

Most organizations 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 organized. 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 re-examine this area.

IX. 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.

Acknowledgment

We gratefully acknowledge the support for this paper from the Department of Computer Science and Engineering and also Department of Information Science and Engineering, R.V College of Engineering, Bangalore, India. Our sincere thanks to HOD of Computer Science and Engineering and other professors for their valuable suggestions and support.

References

[1] Benner, K.; Feather, M.; Johnson, W.L. & Zorman,"Utilizing Scenarios in the Software Development Process" Proc. IFIP WG 8.1 Working Conference o n Information Systems Development Process; North-Holland.2012

[2] Blyth, A., Chudge, J., Dobson, J. & Strens, M. (2012); ORDIT: a new methodology to assist in the process of eliciting and modelling organisational requirements; ACM Conference on Organizational Computing Systems 2012; p p 216-223, 2012

[3] Boehm, B.; Bose, P.; Horowitz, E. & Lee, M.J. "Software Requirements as Negotiated Win Conditions" Proc. 1st International Conference on Requirements Engineering; pp 74-83, 2011

[4] Boehm, B.W & Papaccio, P.N." Understanding and Controlling Software Costs " IEEE Transactions on Software Engineering, SE4, 10, pp 1462-77,2011

[5] Brown, P. "QFD: echoing the voice of the Customer" AT&T Technical Journal; March-April; pp18-32,2011

[6] Christel, M.; Wood, D.; Stevens, S. "AMORE: The Advanced Multimedia Organizer for Requirements Elicitation; CMU Technical Report" CMU/EI-93-TR- 12,2011

[7] Conklin, J." Design Rationale and Maintainability" Proc 22nd Hawaii International Conference on System Sciences; II, pp533-539; 2010.

[8] Curtis, B. "Five Paradigms in the Psychology of Programming " MCC Technical Report; STP-132-87, 2010.

[9] Dardenne, A.; Fickas, S. & van Lamsweerde, A. "Goal-directed Requirements Acqusition" Science of Computer Proigramming; 20, pp 3-50,2009;

[10] Davis, A.M. "Software Requirements: analysis and specification"; Prentice Hall Inc.

[11] Dorfman, M. & Thayer, R.H. "Standards, Guidelines and Examples on System and Software Requirements Engineering" IEEE CS Press Tutorial, 2008;

[12] Fagan, M.E. "Design and Code Inspections to Reduce Errors in Program Development" IBM Systems Journal; 15, 3, pp 182-211, 2008;

[13] Feather, M.S. " Language Support for the Specification and Development of Composite Systems" ACM Transactions on Programming Languages and Systems; 9, 2, pp 198-234,2011

[14] Fickas, S. & Helm, R. "Knowledge

Representation and Reasoning in the Design of Composite Systems; IEEE Transactions on Software Engineering; pp 470- 482, 2007;

[15] Fickas, S. & Nagarajan, P. “Being Suspicious:critiquing problem specifications” Proc AAAI 88; 1, pp19-24; 2010.

[16] Finkelstein, A. & Finkelstein, L. 2009; Review of Design Methodology; Proc. IEEE,130 ptA,4, pp213-222,2009

Biography

Swarnalatha K. S is working as an Assistant Professor in the Dept of CSE in R V College of Engineering Bangalore. My area of interest is Software Engineering, Modelling and simulation.

Dr G N Srinivasan is working as a Professor and Associate Dean in the Dept of ISE, R V College of Engineering Bangalore. My area of interest is Software Engineering, Data Engineering, and Image Processing.