The Future of Requirements Management Tools

Anthony Finkelstein & Wolfgang Emmerich[1]

Abstract

In this paper we look at what industry can expect by way of developments in requirements management tools in the short, medium and long-term future.

Introduction

This paper attempts to identify prospects for existing requirements management tools. The paper is based on considerable experience of building and using requirements management tools, advising organizations on the selection of tools, and knowledge of ongoing research work in the area. The work of the principal author in this area was originally inspired by Prof. Roland Traunmüller who drew his attention to the gap in our understanding of the processes involved in the early stages of software development. Prof. Traumüller has played a key role in bridging between the information systems community where the problems of requirements are properly appreciated and the software engineering community where the tools and methods exist which have a prospect of addressing them.

Requirements management is the systems engineering activity principally concerned with finding, organizing, documenting and tracking requirements for software systems. Its focus is maintaining traceability, defined as the "ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origin, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)" [4].

Where are we?

For any realistically sized systems engineering project, requirements management is a clerically intensive task. It entails being able: to relate many different documents; to obtain a synoptic view of these document relations; to retrieve information from within those documents; to create special document views; to handle changes made across the set of documents in a consistent manner; to accommodate diverse document structuring requirements and document types. In order to understand the scale of requirements management it should be kept in mind that a typical medium-sized systems development project may yield some 2500 distinct statements of requirements, each of which may in turn result in a variety of development documents and associated rationale.

It is to meet the demands of this task that specialist requirements management tools have been developed. They are now among the most dynamic areas of the software development tool market. This paper is not intended to act as a survey of these tools, new tools are being added each day and a survey would have little long-term value, anybody interested in this can make a useful start by looking at the work done by the International Council on Systems Engineering (INCOSE). For orientation however the better known examples of such tools are DOORS, icCONCEPT RTM, RDD-100 and RequisitePro. Information about these is accessible from the INCOSE Survey [6]. It

[1]University College London, Dept. of Computer Science, Gower St., London WC1E 6BT.

is important to understand that the specialist requirements management tools, which are the focus of this paper, are not the only way of maintaining traceability in a systems development project and that some tool support for traceability is embedded within other tools used in the system development process. For a full discussion of this see [4].

Most requirements management tools on the market perform essentially the same core functions. They allow the system developer to import large documents from a variety of standard word processing formats. These documents can be split up into separately managed document elements. The document elements are subject to a rigorous change and version control regime. Relations can be established between document elements and attributes can be associated with the document elements and often the relations. A variety of document views can be generated using both attributes and relations, generally specific traceability views such as traceability matrices. Document templates can be set up and used to create new composite documents. Scripting or query languages provide support for the retrieval of information and the development of project specific views. Simple checks to ensure structural integrity of documents may be performed.

The architecture of the tools have much in common but there are also clear differences. They are generally based on a document repository, which may either be hosted on top of an industry standard database (relational or object-oriented) or a specifically crafted file store. The trade-offs between the different strategies (tailoring and extensibility versus speed and cost) are hotly debated and constitute one of the major distinguishing factors in selling the tools. Most tools provide some simple control for multi-party editing of documents, the granularity of this control is dependent upon the underlying repository. At the front end, the tools generally appear similar to standard document processors. From a user interface standpoint, they provide a number of tools to support work with large hierarchical documents including the ability to work seamlessly in different document views.

Requirements management tools are generic. That is they need to be configured to support specific requirements engineering and system development processes. Configuration is supported by the creation of document templates, schemes of attribute and relation types, and document views. Most tools are sold with some canned processes such as those in the established IEEE and DoD standards. Many of the vendors sell additional consultancy services which help organizations to set up the tools for their specific processes. This constitutes a large part of the vendors offering.

Short-term future

Over the last three years we have seen a significant expansion and a large number of new entrants into the requirements management tool market. It is difficult to predict whether the market is stable as yet, though the clear emergence of a smaller number of major players seems to indicate that it may be. In this position it seems relatively clear that the short term will be spent attempting to mimic the best features of the competitors. Particularly strong features of competing products, that is ones which have an effect on their capacity to win "beauty" contests among major system development organizations are receiving attention. Illustrative examples of such strong features are the intuitive user interface of DOORS and the ability to strip complex documents and perform extraction from pre-existing documents for which icCONCEPT RTM is noted. This may be judged to be a good thing by purchasers. It is however a precursor to the sort of "feature bloat" which some well known software packages display.

As most of the products have been under rapid development there is obviously a concern to ensure the robustness of the tools. Being able to handle large sets of document is also a concern. It has proven easy, even for those with considerable experience, to systematically underestimate the sheer scale of the requirements management process for complex systems. Even some of the most well established tools are reported to either be unreliable or have poor performance when presented with very large sets of documents.

It is also possible to detect a subtle shift in focus among vendors. Very many of the large systems engineering organizations, for whom the requirements management problem was clear and unavoidable, have already committed to tools. While it is probably unfair to suggest that the market in this area is saturated it is certainly an increasingly difficult one to penetrate. By contrast there is a vast swathe of small to medium-sized organizations who are only just waking up to requirements management.

It is these organizations to whom vendors are now turning their attention. However, the problems that these organizations face, with respect to requirements management, are different to those of large organizations. A generic tool is of much less use to them, they commonly do not have a well articulated or documented process, and they look to the tool to provide this. In fact this may be the principal benefit of the tool. The process that is supplied must be suitable for a wide variety of settings and must be sufficiently lightweight that it is acceptable to the target group of organizations. In general therefore vendors are paying attention not just to tool features but to the whole "package" which includes processes.

This fits well with another area of development. It should have been clear from the discussion above that most of the core functionality of requirements management tools is dedicated to general document management activities. These activities take place throughout large organizations, not just in systems engineering. Indeed it is reasonable to argue that traceability itself is a general property required to support most document-intensive business processes. The deployment of requirements management tools in other settings is thus receiving attention. Because of the difficulties of obtaining a toe-hold in other business areas, vendors are looking towards areas immediately peripheral to systems engineering, quality management, the archetypal document-intensive business process, is a case in point.

Another area of short-term interest is tool integration. This integration takes a number of forms for requirements management tools: the ability to import documents, and perhaps to retain some structural and formatting information; the ability to establish traceability links between items maintained in the repositories of other tools within the system development process; the ability to maintain the integrity of those links as information on both sides of the link change; the ability to embed items from other tools into requirements documents; integration with a process wide configuration management infrastructure. Integration with specific tools is an important factor in purchase decisions and there is thus continuing work in all these areas with respect to important tools, integration with Rational Rose [12] is a good example. Of course the ease of integration is dependent on the extent to which the repositories at either end of the integration are open. This is a commercially fraught area and the emergence of strategic business partnerships between tool vendors across the development process is interesting to observe.

Medium-term future

Surprisingly the medium-term horizon for developments in requirements management tools is easier to analyse. The dominant issue is distribution. Most of the existing tools are centralized, that is they rely on a repository which may in a few cases be capable of physical distribution but is logically centralized. Though the tools are, for the most part, well suited to small work groups within a single organization they are relatively poorly adapted to highly distributed heterogeneous collaboration. As the dominant trend in systems engineering is towards global organizations, partnerships and inter-organizational working, distribution requires serious attention.

The obvious difficulty with distribution is that it requires major re-architecting of the tools. The direction in which many are looking are the emerging middleware standards such as CORBA and the like. CORBA provides the basic facilities for distributed communication between heterogeneous tools. The OMG is now working towards standardizing formats that are used to exchange model components between distributed and heterogeneous tools [11].

Hand-in-hand with distribution is the issue of integration with the Web. Many of the existing products have implemented what might be termed trivial integration. Documents can be imported from HTML and published in it. Document links can be realized as Web hypertext links. Achieving this level of integration requires a minimal extension of tool capabilities and has the benefit that documents become viewable on organizational intranet servers.

With the Web rapidly emerging as the predominant document handling medium for large organizations requirements engineering tools will require a deeper integration with Web capabilities. This will include placing a large part of the front-end of such tools within a browser environment. The obvious limitation in this regard is the static, text-oriented nature of HTML. Developments in Web technologies [9], notably Extensible Markup Language (XML) and related developments such as the Open Software Description Format (OSD) are strongly suggestive of convergence between the capabilities of the Web and requirements management tools and may be exploitable in interesting ways.

Multimedia, notably sound and video, is now cheap and practical. The use of multimedia within software engineering is clearly becoming more than a theoretical possibility. Obvious examples are videos of meetings, recordings of key discussions with stakeholders, video comments by way of rationale from developers. Existing compound document technology, such as Microsoft's COM [2], makes placing multimedia objects within documents straightforward. Actually integrating multimedia in this setting requires more attention and problems such as indexing, retrieval and relating multimedia document elements remain to be fully resolved. From the wider point of view it is unclear exactly what the best way to exploit this technological capability might be, this is an area in which requirements management tool vendors might be expected to take a lead.

Long-term future

We identify four major areas for long-term development. One of these areas is clearly fore-shadowed by research, the other three are gaps which need to be filled.

It might seem strange after some of the preceding discussion to describe existing requirements management tools as process-free. By this I do not mean generic but rather that process is captured entirely statically in the form of documents and document templates. Post-hoc it is possible to reconstruct a process by looking at the requirements flow-down. Though, by careful construction of the documents and the set of document relations, it is possible to achieve some control of the process execution, this is necessarily limited. Simple questions like "what should I do next" cannot be answered from within the requirements management tool.

The obvious answer to this problem is to integrate requirements management tools with a workflow or process engine. Despite this being an obvious answer it is not very straightforward to achieve. The construction of requirements documents is a highly complex skilled task involving a large amount of context shifting and of what is termed in the collaborative work community situated action. The rather rigid notions of workflow or process which are well suited to routinized tasks are not suitable for application in this setting.

New, tolerant or flexible concepts of workflow which deliver some process support but also accommodate the ways in which people really construct such documents are required. The predominant mode of such support should be guidance rather than the rather rigid, heavy handed, enforcement which have made process engines unacceptable in many software engineering applications [3].

Anybody observing the state-of-practice in requirements engineering cannot help but be struck by a very sharp divide. On one side of the divide sits the vast bulk of industrial practice in which requirements are stated in natural language, and managed in the style discussed above. On the other side of this divide are the modelling methods, increasingly the object-oriented analysis techniques. Some organisations state their requirements in natural language and proceed directly to design.

Some write their requirements in natural language and alongside this carry out a model-based analysis and then design. Only a relatively few proceed from analysis to design using an unadulterated modelling method.

For long, academics have argued that natural language requirements documents, shot through as they are with ambiguity and inconsistencies that cannot be readily identified, are vestiges of outmoded practice. The persistence of the use of natural language can, they contend, be accounted for by poor training and technology transfer or the inadequacies of current analysis methods (which, of course, they intend to rectify by introducing a new language). The reality belies the story. Most specifications in formal schemes such as Z are accompanied by large bodies of natural language text and are unusable without it.

We argue that the reason that the use of natural language persists in requirements documentation is that it plays a valuable role and furthermore that it is unlikely to be supplanted. This role goes beyond simply providing a means for stakeholders to validate the specifications, though this is in itself very valuable, but is concerned with the essence of the specification task. Requirements refer to the real-world, for the models that result from analysis to be comprehensible it is essential that the correspondences between the components of the model and the real-world phenomena are explicated. Without this the models are airy abstractions.

At the moment we have no very good way of using natural language and modelling together in a synergetic way. At best the two are loosely stuck together rather than being woven into a coherent whole. This is exacerbated by the separation of CASE tool and requirements management tool which will require more than ad-hoc integration to address.

The vision of the future is of what we term "literate modelling" [1]. The basic idea is very similar to Knuth's literate programming [8]. In this vision modelling and natural language documentation are seamlessly bound together and the functionality of CASE tool and of requirements management tool are usable together in a truly integrated manner. The steps that need to be taken to achieve this are largely conceptual if the result is not to be a bloated mega-tool.

Software architecture is now a very active area of research and we have seen the emergence of a number of architectural description languages (ADLs) [10] and schemes for describing architectural styles. Virtually no attention has been paid, to date, to the question of the interplay between requirements and architectures. It is emerging that the selection of a suitable architecture for a system is critically dependent upon the non-functional requirements (performance, load and the like) for that system. Architecting systems and setting out the requirements for systems are much more interdependent than has hitherto been thought. It is clear that this may have implications for the design of both requirements management tools and "architects assistants" [7] but what these implications are clearly requires further research.

Organizations are now increasingly aware of knowledge management. Software development organizations are realizing that their principal asset is their experience, and the knowledge they have gained through that experience. They are taking steps towards establishing corporate knowledge bases which make this asset tangible. Requirements form a key part of this knowledge base, there is thus a need for a shift from managing requirements over the life of a project to managing those requirements as a continuing contribution to general corporate knowledge. What exactly this might entail is rather unclear and depends upon corporate knowledge management and organizational learning becoming more than fashionable tags. Nevertheless, we regard this as an interesting area worthy of attention.

Conclusion

Requirements management tools are not a silver bullet. They are a helpful aid to an organization that is able to understand process and is able to manage commitments in a sensible and consistent manner [5].

We have set out in this paper the prospects for requirements management tools and attempted to distinguish developments that can be expected in the short and medium-term future as well as speculating on the long term agenda. In the short-term, we have identified convergence of tool features, robustness, packaged processes, wider business application and tool integration as key developments. In the medium term we have identified distribution, Web-integration, and multimedia as important prospects. In the long-term we have suggested workflow integration, literate modelling, software architecture and corporate knowledge management as constituting highly significant areas of potential development.

Clearly, it is possible to argue with our list of prospects and particularly with our classification of short, medium and long-term items. What seems less arguable is that requirements management tools are set to be at the heart of future development in tools for systems engineering. Organizations that are not riding the curve of developments in these tools may well lose out significantly.

Acknowledgements

This paper is dedicated to Prof. Roland Traunmüller whose help and encouragement has been of great value and to whom the principal author owes a sincere debt of thanks.

References

[1] J. Arlow, W. Emmerich and J. Quinn. Literate Modelling - Capturing Business Semantics with the UML. In: J. Bezivin and P.-A. Muller (eds) The Unified Modeling Language: <<UML '98>>: Beyond the Notation, Mulhouse, France, Lecture Notes in Computer Science. 1618, pp. 189-199. Springer Verlag. 1999.

[2] D. Box. Essential COM. Addison Wesley, 1998.

[3] W. Emmerich, A. Finkelstein, C. Montangero, Stefano Antonelli, Stephen Armitage and R. Stevens: Managing Standards Compliance. IEEE Transactions on Software Engineering 25(6). 1999.

[4] O. Gotel and A. Finkelstein. An Analysis of the Requirements Traceability Problem. In Proc.of the 1st Int. Conf. on Requirements Engineering, pages 94–101. IEEE Computer Society Press, 1994.

[5] W. Humphrey, D. Kitson, and T. Kasse. The State of Software Engineering Practice: A Preliminary Report. In Proc. of the 11 th Int. Conference on Software Engineering, pages 277–288. IEEE Computer Society Press, 1989.

[6] INCOSE Tools Database Working Group (DBWG). Requirements Management Tools Survey. http://www.incose.org/lib/index.html, 1997.

[7] K. Ng and J. Kramer and J. Magee. A CASE Tool for Software Architecture Design. Automated Software Engineering, 3(3/4):261–284, 1996.

[8] D. Knuth. Literate Programming. The Computer Journal, pages 97–111, 1984.

[9] S. Mace, U. Flohr, R. Dobson, and T. Graham. Weaving a Better Web. Byte, March 1998.

[10] N. Medvidovic and R. N. Taylor. A Framework for Classifying and Comparing Architecture Description Languages. In Software Engineering - ESEC/FSE ’97, 6th European Software Engineering Conference, Zurich, Switzerland, number 1301 in LNCS, pages 60–76. Springer, 1997.

[11] OMG. Stream-based Model Interchange Format - RFP. ftp://ftp.omg.org/pub/docs/ad/97-12-03.pdf, DEC 1997.

[12] Rational Software Corporation. Rational Rose 98. http://www.rational.com/products/rose, 1998.