Modelling the Contribution Structure Underlying Requirements

Orlena C. Z. Gotel & Anthony C.W. Finkelstein

Imperial College of Science, Technology & Medicine
Department of Computing
180 Queen's Gate
London SW7 2BZ
(oczg; acwf@doc.ic.ac.uk)

Abstract

Improvements in requirements traceability (RT) are essential for developing better quality software systems. Based upon empirical investigations with practitioners, and an extensive survey of techniques and tool support for requirements engineering (RE), we explain why conventional RT can only support limited improvements in quality, and suggest extensions required to provide further improvements. These extensions revolve around the ability to continuously model, and hence trace, those who have contributed in the production and refinement of the requirements specification (RS). In this paper, we propose an approach to support such modelling, outline preliminary details, and discuss how it provides the foundations for developing quality software.

1 Introduction

Quality-oriented approaches to software development, and their increasing support by computer-aided software engineering (CASE) tools, have become the focus of considerable attention [15]. Although they have led to marked improvements in software quality (as reported in: [1]; [22]; and [23]), this has rarely been to the extent anticipated (see: [7]; [25]; and [26] which discuss some reasons for this). In this paper, we argue that RT is the primary *quality-enabling* technique, and that more of these expected improvements would be realised if a comprehensive approach to RT was adopted. The extensions we propose for doing this have been motivated by a detailed analysis of the RT problem [12].

The structure of this paper is as follows: In Section 2, we illustrate the relationship between RT and software quality. In Section 3, we show the restricted impact that conventional notions of RT can have on quality, and explain why extensions to this notion are needed to provide a firmer foundation upon which to achieve and assess

quality. In Section 4, we describe how advances can be made by modelling the contributors to any information generated during the production and refinement of requirements, particularly to locate the personnel to participate in quality assurance activities. In Section 5, we outline an approach we propose for the on-going modelling and update of these contributors and their contributions. We discuss the implications of this approach and the future directions of our research in Section 6.

2 Relation Between Requirements Traceability & Software Quality

> *"**Requirements traceability** refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, 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"* [12].
>
> **Software quality** (in software engineering) is: *"1. The totality of features and characteristics of a software product that bears on its ability to satisfy given needs, for example, to conform to specifications. 2. The degree to which software possesses a desired combination of attributes. 3. The degree to which a customer or user perceives that software meets his or her composite expectations. 4. The composite characteristics of software that determine the degree to which the software in use will meet the expectations of the customer"* [2].

The prevailing approach to quality-oriented development involves 2 basic steps: (1) specify the requirements for the proposed system (where the formality of their specification determines the extent to which step 2 can be automated); and (2) use these requirements as a reference point from which to drive, control, and evaluate the development process.

The definition of "software quality" above, indicates that quality is assured if: (a) software meets its users needs (*user requirements*); or (b) software conforms to rigid quality attributes (*quality requirements*). Such quality assurance is supported by many approaches. For example, through: development methods, like structured and top-down decomposition [27]; dedicated techniques, like quality-function deployment (QFD) [4]; development paradigms, like the Quality Improvement Paradigm (QIP) [20]; or through separate quality assurance groups, or quality engineers, as exemplified in the software factory concept [9].

Common to all these approaches is their dependence upon the ability to maintain traceability between requirements in the RS and any subsequent artifacts in which they are distributed. This has been defined in [12] as *post-requirements*

specification (post-RS) traceability (and is illustrated in Figure 1). Techniques for post-RS traceability can help ensure that quality requirements are considered in (and permeate through) all phases of development, and can also be used to check the extent to which these have been met at each phase. In this way, software quality is directly influenced by the techniques and tools used for RT.

3 Why Post-RS Traceability Only Supports Limited Improvements

Although there are numerous techniques and tools which provide dedicated support for post-RS traceability (summarised in [12]), software still frequently fails to attain the levels of quality anticipated for it. This is because these techniques and tools only deal with those phases of a requirement's life which result from its specification in the RS. They typically depend upon the pre-specification of a (relatively static) requirements baseline before they can operate, and lack support for its initial production. Therefore, the assumption they embed is that the requirements in the RS are easy to obtain, accurate, and stable, an assumption that is echoed where "quality" is defined in terms of *meeting the specification*.

These assumptions concerning the nature of requirements are often misguided. User requirements are notoriously difficult to obtain with accuracy, are frequently unstable, and become redefined over time. Furthermore, user satisfaction tends to be a collective and subjective matter for which reliable and all-encompassing measures cannot be articulated. Quality requirements are commonly imported from external standards or policy documents, such as [14], where they are pre-specified along with metrics for promoting *best practice* and measuring compliance [17]. Due to the sheer number of potential quality attributes (see [3] for a representative list), these need to be adopted and tailored on a project-specific basis [5]. Furthermore, these definitions of quality change, the relevance of metrics to quality change, and quality requirements and metrics are often actively constructed within different phases of development. The very nature of these requirements indicates that, although post-RS traceability can promote concern for quality during development, and help assess subsequent conformance to an RS, it cannot guarantee quality. This is even the case where requirements are formally specified, as it lacks the means to probe beyond what is specified, and hence account for all phases of a requirement's life.

Although possible extensions have been suggested (e.g., obtain more complete documentation about the problems being addressed and the wider organisational contexts [11]), these generally do not address the fact that when requirements change (be these quality attributes, metrics, policies, or standards), these changes need to be instigated from their *initial* source, and *re-propagated* through the pre-RS phases, if they are to be handled effectively. We argue that additional improvements in quality could be obtained if quality-oriented approaches to software development included comprehensive support for the production and refinement of requirements. This

would support, and control, the impact of changing quality attributes and measures. This clearly depends upon the ability to maintain traceability between dispersed statements or documents and the requirements into which they have been integrated in the RS. This has been defined in [12] as *pre-requirements specification (pre-RS) traceability* (and is illustrated in Figure 1).

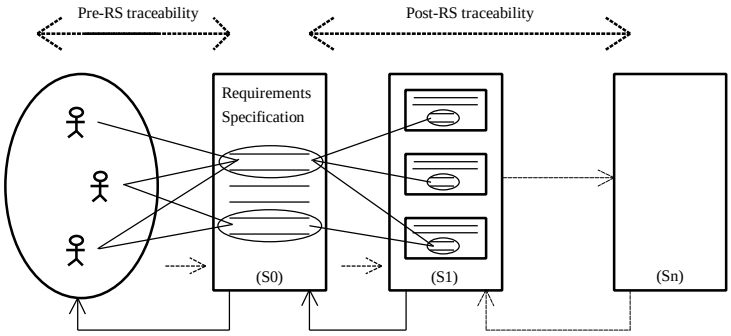

Figure 1: Pre-RS and post-RS traceability

4 How Pre-RS Traceability Can Improve Quality Further

In a critique of quality assurance activities in software engineering, Loka reports how most of these activities tend to be carried out towards the end of a project, after the original development team has moved on [18]. Accordingly, any defects that are identified are inadequately addressed: out of context; by other parties; and by directly *weeding* them out of the end product itself [10]. This leads to ever-deteriorating quality. Pre-RS traceability enables a *quality culture* to be built in from project conception, and for quality to be an on-going concern, by supporting: (a) analysis of the RE process, to identify the cause of any defects (a documentation problem); and (b) identification of those in a position to assess quality or address any defects (an organisational problem). This is essential because a positive correlation has been repeatedly stressed between the earlier that support is injected into the development process and the quality of the software that can be delivered (e.g., in [21]).

To date, most of the support in the pre-RS area focuses on: (a) increasing the amount, and improving the structure, of information that is documented about the RE process; and (b) providing powerful and selective retrieval mechanisms for accessing this information. However extensive or sophisticated these become, our empirical investigations indicated that there will always be situations in practice where this is not sufficient. For exactly this reason, one of the basic working

practices of the practitioners in our studies was found to be identifying the (human) source of requirements and requirements-related information, to enable informal face-to-face communication with appropriate participants. In so doing, problems are currently experienced because such information is either: not available; outdated; or refers to those who wrote the documentation, rather than those who formulated the content therein. This is because the predominant way to attach details of personnel involvement is through a document field, labelled "author", or through the use of annotation mechanisms. This is the reason why the end products of RE lose details of those who originally generated the information and those who were involved in all phases of its refinement.

These findings indicate that more information needs to be provided about the social setting of the RE process, and in particular, that all the information that is produced needs to be augmented with details of personnel contributions. Such details must be updated to reflect the evolving and changing nature of these contributions if they are to be of continued use. In this way, pre-RS traceability provides the ability to trace those involved in producing and refining requirements, which can account for the fact that quality is socially defined, socially evaluated, and only accepted within a social setting. Although requirements may be well defined, and even imported from agreed standards, what is specified will often mean different things to each individual involved: *"probably the most common error made in quality assurance is the assumption that there is a common understanding of what 'high quality' software actually means. No such common understanding exists. Situations arise where different software engineers strive, in a mutually antagonistic way, to ensure that particular, but different, product attributes are achieved"* [24]. Access to those defining (or importing) the requirements being used to drive quality development, is often the only way to ensure a shared interpretation for so doing. Also, because many quality attributes and measures are subjective, tools and formalisms can rarely assess quality independent from those specifying the criteria. This need to consider the social dimension in evaluation exercises has been the subject of a number of papers (e.g., [13]).

5 Proposed Approach for Modelling Contributors & Contributions

Our proposal for improving software quality revolves around augmenting any information produced in the RE phases with details of its contributors. There are some basic requirements for modelling this: (1) the model must be updateable, to reflect the evolving structure of contributions, and be open to alternative presentations; and (2) its basis must offer some guidance for model development and refinement, and enable some capacity for reasoning with and about the contribution structure, plus any subsequent changes to it. The broad approach we propose is illustrated in Figure 2.

5.1 Modelling Artifacts (Contributions)

We are only concerned with the tangible artifacts that are produced and exchanged in the RE process, such as: formal documents; memos; faxes; videos of meetings; spreadsheets; etc. Process details, such as the conversations amongst participants, are not essential for investigating a suitable basis for the modelling, and could be incorporated at a later date. We assume 2 basic types of artifact: (1) *primitive* (i.e., dependent on no other artifacts); and (2) *compound* (i.e., based on, or referencing, other artifacts). These artifacts are represented in an on-line form and are placed under the control of a database management system. This deals with structural and content-based traceability, provides interrogation and query facilities, and maintains the following structures: (a) a *main* (or primary) structure, defined by development-based relations between whole artifacts; and (b) many *subsidiary* (or secondary) structures, defined by content-based relations between artifacts and/or various size components therein. We make these assumptions because there are rudimentary techniques and tools which can do this.

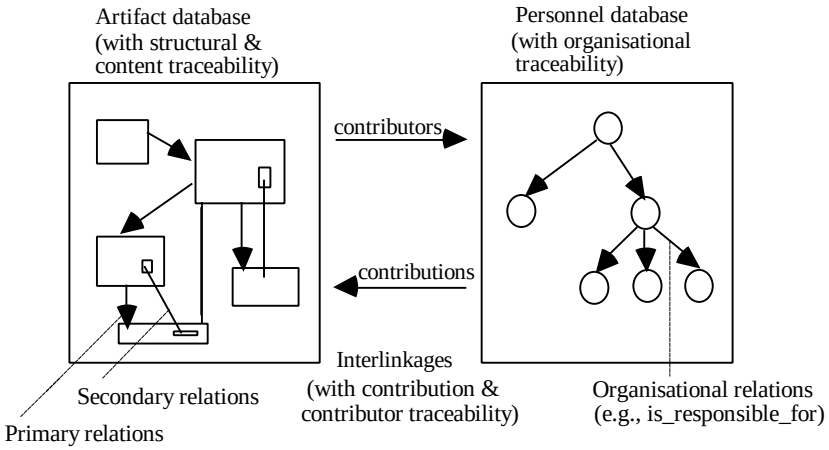

Figure 2: Outline of the approach

5.2 Modelling Personnel (Contributors)

An organisational database will be used to hold various details (e.g., positions and aliases) about the personnel who have directly and indirectly participated (e.g., developers and customers). It may be used to model diverse organisational information (such as: power structures; delegation structures; responsibility structures; group structures; etc.), so it can define and deal with whatever types of relation that can exist between personnel. We assume 2 basic types of participant:

(1) *individual* (i.e., non-decomposable); and (2) *group* (i.e., decomposable into further groups and/or individuals).

5.3 A Scheme for Modelling the Interrelation Between Personnel & Artifacts

The details of the contributors will be manually defined, once in an on-line form, so in a post-hoc manner. Contributions can be of any size, nested, overlapping, embedded, etc. The problem of capturing such details, in an on-going and automated manner during the process itself, can be investigated once a suitable scheme has been developed.

The crux of the approach lies in providing a suitable scheme with which to represent the link between personnel and artifacts. Merely defining this structurally, with no semantics other than *contributes_to* or *contributed_by*, is too coarse to meet the basic requirements we have listed. We suggest that a useful semantics is the *nature of the contribution* (i.e., the role of the individual/group with respect to the artifact/component). A preliminary taxonomy would involve the following types of contribution: originates/produces; documents; adopts (uses without change); adapts (uses with change); and authorises/stabilises (signs off). Such a scheme provides an effective way of dealing with complexity, as it superimposes different layers upon the artifact to deal with the different types of contribution, which can be selectively used for different types of focused reasoning (as depicted in Figure 3). Each of these types has different implications for the recomputation of the contribution structure following change. We are investigating what these implications are, whilst examining further attributes which have an effect on change (e.g., the strength of the contribution). To provide a logical basis for this scheme, we are examining how these types of contribution can be redefined in terms of *commitments* (adopting the definition given in [8]). In this way, we should be able to reason about changes to the contribution structure, and provide rules for its recomputation. In summary, what we are proposing is: to link contributors and contributions, by *marking-up* an artifact with its contributor structure, using an underlying scheme that is theoretically based upon "commitments".

5.4 Extending the Scheme to Keep the Model Up to Date

To be able to accurately locate which personnel can perform quality assurance, or address quality defects, throughout a project's life, there is a requirement to support the evolution of this contribution structure. When changes are made to existing artifacts, the artifact-based traceability links can update the content, but in so doing, the underlying contribution structure is often redefined. The scheme proposed above can be used to support the recomputation of contribution details and update this structure: it can extract the developmental traceability chain behind the information being changed (back to the originating artifact); and it can extract the associated contribution structure at each step. In addition, based on the nature of these

contributions, it can build up a priority structure for who to contact and inform about change.

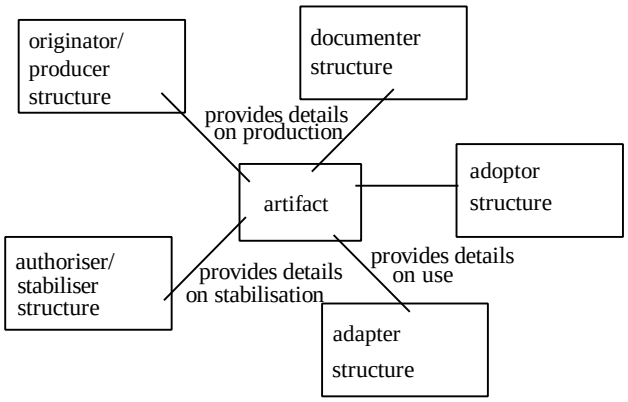

Figure 3: Different social structures an artifact is simultaneously connected to (based on the nature of the contribution)

However, this process is complicated by the fact that the nature of the change, and the nature of the contributions underlying an artifact, have implications on how this recomputation should be done. We are investigating these issues. Also, the type of the content traceability links that exist between these artifacts, will also impact this process. There is a need to define the semantics of these relations, so they can be taken into account. Here, we propose the use of a scheme based on *communicative functions* (i.e., the function of a second artifact with respect to a first), such as: copies; uses; refines; elaborates; etc. We are currently developing a taxonomy, and specifying the impact of each function on the contribution structure (e.g., an identity function, or direct copy, would update the adopter structure, but not effect the originator structure).

6 Research Directions, Issues & Implications

Where attempts are made in practice, to keep track of *ownership*, this usually suffers from: coarse-grain labelling of contributions; no distinction between group and individual (or direct and indirect) contributions; no differentiation between contribution type (this is normally only the documenter/author, as opposed to the inspiration); inability to account for the fact that any subsequent revisions or versions may update or impose new contribution structures; and the inability to make any *intelligent* use of this information. We are unaware of other research explicitly directed at this problem, though research in *process modelling* aims to provide various details about the relations between people, people and their activities, etc.

(e.g., [16] and [19]). However, efforts to increase the amount of development process knowledge tends to touch upon all concerns rather thinly, whereas we have argued the need to focus in detail on the contributor and contribution knowledge. They do not provide an overall and up to date view of the contribution structure, or enable this information to be used for reasoning about quality issues.

We have only given a preliminary sketch of our proposed approach. We believe it will offer the basis for significant improvements in quality. It provides the means to rapidly identify those individuals to involve and inform in situations of change, decision making, quality assessment, conflict resolution, etc. By storing preferred communication protocols with personnel details, and through integration with suitable communication tools (such as those described in [6]), it offers the extra potential to automatically instigate any required communication. The possibilities it opens for project management and education will also influence quality. For example, it provides the apparatus to: compare predefined and actual organisational structures; infer dynamic details about power relations and alliances; integrate new personnel and deal with the consequences of those that leave; etc.

Our current research involves developing a suitable theoretical basis for the artifact, personnel, and contribution traceability referred to above. We are also developing a generic mark-up language, to overlay artifacts with their contributor details. This language will provide the rigorous basis for carrying out the flexible types of reasoning and update we have mentioned. Also, we intend to actively support and guide the revision and correction of requirements based on this approach. We further intend to embed the approach in tool support for RE, for refinement and evaluation purposes, and (where appropriate) in standard document preparation systems.

7 Conclusions

The quality of a software system must be actively built into the development process from the onset of a project. Support must therefore be provided for the continuous definition, redefinition, assessment, and re-assessment of quality throughout a project's life. We have argued how more can be done in the pre-RS phases to establish a firmer foundation for these activities, and in particular because pre-RS traceability makes the work preceding the RS open to interrogation. This enables the concept of "quality" (and similarly its satisfaction criteria) to evolve with: (a) the evolving environment in which "quality" is defined and measured; and (b) the evolving development process through which it takes effect. We have further argued how the traceability of the personnel who have contributed to the requirements in the RS supports a fundamental working practice which is often the only way to explain and assess quality criteria. Therefore, modelling this social structure provides a firmer foundation upon which quality software can be built and measured. We are actively examining a suitable basis for modelling and reasoning with the contributors

and contributions in RE. We also intend to develop suitable mechanisms to support and evaluate its use in practice.

References

| Aaen, I., Siltanen, A., Sorensen, C. and Tahvanainen, V.-P. (1992). "A Tale of Two Countries: Case Experiences and Expectations", in Kendall, K. E., Lyytinen, K. and DeGross, J. I. (eds.), *The Impact of Computer Supported Technologies on Information Systems Development*, IFIP Transactions, North-Holland, pp. 61-93.

| ANSI/IEEE Standard 729-1983, *IEEE Standard Glossary of Software Engineering Terminology*, Institute of Electrical and Electronic Engineers, Inc., N. Y.

| Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., Macleod, G. and Merrit, M. (1978). *Characteristics of Software Quality*, TRW Series on Software Technology, North-Holland.

| Brown, P. G. (1991). "QFD: Echoing the Voice of the Customer", *AT&T Technical Journal*, March/April, pp. 18-32.

| Buckley, F. J. and Poston, R. (1984). "Software Quality Assurance", *IEEE Transactions on Software Engineering*, Volume 10, Number 1, January, pp. 36-41.

| Cockburn, A. and Greenberg, S. (1993). "Making Contact: Getting the Group Communicating with Groupware", *Proceedings of a Conference on Organizational Computing Systems*, Milpitas, California, November 1-4, pp. 31-41.

| Curtis, B. (1992). "The CASE for Process", in Kendall, K. E., Lyytinen, K. and DeGross, J. I. (eds.), *The Impact of Computer Supported Technologies on Information Systems Development*, IFIP Transactions, North-Holland, pp. 333-343.

| Finkelstein, A. and Fuks, H. (1988). "Multi-Party Specification", *Proceedings of the Fifth International Workshop on Software Specification and Design*, (IEEE CS Pr*ess and* ACM SIGSOFT Notes), pp. 185-195.

| Fisher, A. S. (1991). *CASE: Using Software Development Tools*, John Wiley & Sons, Second Edition.

0] Flatten, P. O. (1992). "Requirements for a Life-Cycle Methodology for the 1990s", in Cotterman, W. W. and Senn, J. A. (eds.), *Challenges and Strategies for Research in Systems Development*, John Wiley & Sons, Chapter 13, pp. 221-234.

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

2] Gotel, O. C. Z. and Finkelstein, A. C. W. (1993). "An Analysis of the Requirements Traceability Problem", *Proceedings of IEEE International Conference on Requirements Engineering*, Colorado Springs, Colorado, 18-22 April 1994.

3] Hirschheim, R. and Smithson, S. (1987). "Information Systems Evaluation: Myth and Reality", in Galliers, R. (ed.), *Information Analysis: Selected Readings*, Addison-Wesley, Chapter 20, pp. 367-380.

4] ISO-9000-3. (1991). *Quality Management and Quality Assurance Standards - Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software*, International Organization for Standardization, Geneva, Switzerland.

5] Jarke, M. and Pohl, K. (1992). "Information Systems Quality and Quality Information Systems", in Kendall, K. E., Lyytinen, K. and DeGross, J. I. (eds.), *The Impact of Computer Supported Technologies on Information Systems Development*, IFIP Transactions, North-Holland, pp. 345-375.

6] Jarke, M., Bubenko, J., Rolland, C., Sutcliffe, A. and Vassiliou, Y. (1992). "Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis", *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, California, January 4-6, 1993.

7] Keller, S. E., Kahn, L. G. and Panara, R. B. (1990). "Specifying Software Quality Requirements with Metrics", in Thayer, R. H. and Dorfman, M. (eds.), *System and Software Requirements Engineering*, IEEE Computer Society Press Tutorial, pp. 145-163.

8] Loka, R. R. (1992). "Software Engineering - Quality Assurance", *ACM SIGSOFT Software Engineering Notes*, Volume 17, Number 3, July, pp. 34-38.

9] Mi, P. and Scacchi, W. (1992). "Process Integration in CASE Environments", *IEEE Software*, March, pp. 45-53.

0] Oivo, M. and Basili, V. R. (1992). "Representing Software Engineering Models: The TAME Goal Oriented Approach", *IEEE Transactions on Software Engineering*, Volume 18, Number 10, October, pp. 886-898.

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

2] Polack, A. J. (1990). "Practical Applications of CASE Tools on DoD Projects", *ACM SIGSOFT Software Engineering Notes*, Volume 15, Number 1, January, pp. 73-78.

3] QED. (1989). *CASE: The Potential and the Pitfalls*, QED Information Sciences, Inc.

4] Sommerville, I. (1989). *Software Engineering*, Addison-Wesley, Third Edition.

5] Sumner, M. (1992). "The Impact of Computer-Assisted Software Engineering on Systems Development", in Kendall, K. E., Lyytinen, K. and DeGross, J. I. (eds.), *The Impact of Computer Supported Technologies on Information Systems Development*, IFIP Transactions, North-Holland, pp. 43-60.

6] Wynekoop, J. L., Senn, J. A. and Conger, S. A. (1992). "The Implementation of Case Tools: An Innovative Diffusion Approach", in Kendall, K. E., Lyytinen, K. and DeGross, J. I. (eds.), *The Impact of Computer Supported Technologies on Information Systems Development*, IFIP Transactions, North-Holland, pp. 25-41.

7] Yeh, R. T. and Ng, P. A. (1990). "Software Requirements--A Management Perspective", in Thayer, R. H. and Dorfman, M. (eds.), *System and Software Requirements Engineering*, IEEE Computer Society Press Tutorial, pp. 450-461.