Progress Towards RACE: a 'Soft-Centred' Requirements Definition Method

D.W. Bustard

Department of Computing Science, University of Ulster, Cromore Road, Coleraine, BT52 1SA, Northern Ireland

Abstract

This paper summarises some recent work by the Requirements Definition Group at the University of Ulster and other collaborators on developing a better requirements definition method for computing systems. The method, RACE (Requirements Acquisition and Controlled Evolution), is gradually being shaped through a series of intermediate research studies. In essence, the approach has been to establish requirements for RACE, identify individual techniques that meet those requirements, experiment with the combined use of the techniques and finally build the method. In practice, RACE has been influenced significantly by Checkland and Wilson's Soft Systems Methodology (SSM) and this forms the core of the method. The paper discusses eight main requirements for RACE and describes four of the intermediate studies. Plans for future work are also outlined

Keyword Codes: D.2.1; D.2.2
Keywords: Requirements/Specification; Tools and Techniques

1. INTRODUCTION

Despite an increasing clarification of what 'good practice' means in software engineering [1, 2], and a general willingness by industry to follow that practice, the goal of routinely completing computing projects to specification, on time, and within budget, remains well out of reach [3]. The obvious implication is that some aspects of the software engineering process need to be improved and the best place to start is probably right back at the beginning of the process, in the requirements definition phase, where the greatest potential benefit can be obtained [4, 5]. The purpose of this paper is to describe the rationale for, and basic characteristics of the requirements definition method, RACE, Requirements Acquisition and Controlled Evolution, currently under development at the University of Ulster in collaboration with co-workers in other establishments. The method is being designed from scratch, on the assumption that a thorough revision of current thinking is needed to produce significant benefits. The RACE method is, however, building on known concepts and techniques that are considered valuable, such as the Soft Systems Methodology [6, 7], object-oriented development [8], and risk management [9].

The development of RACE has, as might be expected, involved a specification of its requirements. The specification has been produced as a list of the essential attributes of any acceptable method. For example, these include "it must support evolutionary system development" and "it must support the use of formal description techniques". None of the requirements identified is particularly novel. Consequently for each requirement there is a considerable amount of material in the literature describing how to deal with this aspect of analysis, and highlighting specific techniques that are applicable. There is, however, no existing requirements definition method that comes close to covering all of the requirements specified. The approach to developing RACE has, therefore, involved selecting from appropriate existing support techniques and examining how they might be used in combination.

The work is currently in this experimental phase and the second section of the paper summarises four particular areas of study. This is preceded by a discussion of the eight main requirements of RACE that have been identified.

2. RACE REQUIREMENTS

When considering the desirable attributes of a requirements definition method, or indeed any method, it is perhaps tempting to start by putting down broad general characteristics such 'easy to learn', 'easy to apply', and so on. These are obviously important but unfortunately are at such a high level that they provide little help in clarifying the method. The initial approach taken with RACE has been to set out attributes that are much more specific to requirements definition although still general enough to give substantial flexibility in the design of the method. The broader quality attributes will be considered later when the method is being assembled.

So far, eight main attributes of RACE have been isolated. These are:

1. It must deal with the broad problem situation and not be specific to computing

Computing systems do not exist in isolation and are rarely a goal in themselves. They are developed to support some wider system and are introduced because of a need to make changes to that system. This wider system is the *problem situation* and any useful requirements definition technique must address requirements for the wider system before considering specific computing needs. Indeed, the analysis technique must be sufficiently neutral to allow the conclusion, in some circumstances, that the system change need not include computing facilities.

2. It must support both goal driven and problem-driven approaches to analysis

There are two broad approaches to solving any problem: *goal-driven* and *problem-driven* and it is unsatisfactory for a requirements definition method to support one to the exclusion of the other. In essence, the goal-driven approach means identifying some desirable position and then deciding how best to reach that position; the problem-driven approach means investigating the current position and then deciding how to best to improve it. Both approaches have their uses. For example, if no documented goals exist or if a major change to a system is envisaged, then the goal-driven approach is desirable. Alternatively, if a system is largely satisfactory and only needs minor enhancement then the problem-driven approach is preferred.

3. It must allow for multiple perspectives

Most systems defy precise definition. In general, a system exists for multiple purposes and individuals construct personal viewpoints of the system using various combinations of these purposes. For example, a restaurant may exist to supply sustenance, to provide entertainment, to facilitate business discussions, and so on. The actual nature of the restaurant will depend on the precise emphasis placed on each of these purposes by the proprietor. Allowance must be made for supporting such multiple perspectives in managing requirements. More precisely, this means enabling multiple system perspectives to be identified and explored, and having some way of combining the perspectives in one system in a balanced way. It must be appreciated, however, that there is therefore no notion of identifying the 'right' system, only of composing a system that is effective given the preferences of the client and others in the problem situation. In many circumstances opinions will differ so some means of resolving conflict is also required.

4. It must support collaborative requirements definition

There are several different ways to view the role of an analyst in the requirements definition process. At one extreme the analyst can be seen as a passive gatherer of requirements, responsible for documenting the needs of those in the problem situation, the *problem-owners*, and identifying any conflicts. At the other extreme, the analyst has the role of 'expert problem-solver', responsible for clarifying the problem and developing possible solutions. Neither of these extremes is satisfactory. Certainly the analyst must be a good listener but it is equally important that the analyst contributes to the solution of the problem to take advantage of his or her own expertise. Similarly, although the analyst can help to solve the problem, ownership of the final solution must lie with the problem-owners, making it essential that they be directly involved in problem solving. The conclusion, therefore, is that the process of defining requirements is best treated as a collaborative problem-solving activity with the analyst and problem-owners having similar status. The requirements definition method must therefore support such a collaborative approach.

5. *It must address risk issues*

Requirements definition, like any creative process, is a cycle in which proposals are made, evaluated, and then revised and refined as necessary. Developing an initial proposal for change is largely a positive activity concerned with identifying what a system is required to do. Evaluation is much more negative, tending to focus on ways in which the proposed system or change process may fail. These opportunities for failure are *risks*. The requirements definition method must provide a mechanism for identifying such risks and for managing them appropriately. In some cases the identified risks can be avoided; in others it is necessary to develop contingency plans to deal with the associated failures should they occur.

6. *It must support evolutionary system development*

All useful systems evolve, meaning that they go through a sequence of development changes. This evolution can be forced by need or be part of a planned programme of change. The latter approach allows for the possibility of making the initial change through a sequence of intermediate steps [10, 11]. This has the benefit of simplifying each change step and thereby reducing the risk of failure. Gilb [10] states that each phase should make a change that is of measurable benefit to those in the problem situation. The intention is that once a particular phase is complete the benefit of the change is assessed and the planned programme revised accordingly. The requirements definition method must support such evolutionary development in an appropriate way.

7. *It must address the needs of subsequent system development*

Any requirements definition method must be well integrated with the larger software development process to which it belongs. It is clearly impractical to link up to all, or even a substantial majority of existing development methods but there is a need to demonstrate such integration. The link must consider how the specified requirements will be used in subsequent development. For example, this would include the use of the requirements specification in software design and acceptance testing. The change control mechanism for requirements must also be considered, covering both initial development and system maintenance.

8. *It must support the use of formal description techniques*

The use of formal description techniques in specifying requirements offers several potential benefits including (i) the development of a deeper understanding of both the problem situation and the changes proposed to alleviate that situation; (ii) the avoidance of misunderstandings in subsequent system development based on the requirements; and (iii) having a precise statement of needs against which to evaluate the final system. Traditionally, formal techniques have been more associated with proving the correctness of a final implementation and of supporting a refinement approach to program development in which an initial formal specification is successively transformed into a coded implementation. While this is possible, in principle, the effort involved means that it can only be attempted for critical parts of large systems, or where safety concerns are paramount, or where formal proof is mandated. This paper takes the view that formal descriptions are of benefit in general software development but that their use should be optional. Like the previous requirement, there is then a need to demonstrate how this can be achieved in at least one case.

The next section considers four intermediate studies that are contributing to the definition of RACE by examining the use of techniques that go part of the way towards meeting the stated requirements.

3. INTERMEDIATE STUDIES

The discussion so far has been slightly misleading in implying that the requirements for RACE were developed before identifying relevant support techniques. In practice, the two evolved together through an investigation of existing analysis methods in search of worthwhile ideas. In all of this work, the most significant 'discovery' was Checkland and Wilson's Soft Systems Methodology (SSM) [6, 7], which now forms the core of the RACE method. This section describes four RACE studies, all of which involve SSM. The first examines the value of SSM as the development base for RACE. SSM meets several of the stated requirements but its most attractive characteristic is that it is a 'lean' technique, meaning that all of its individual elements having a clear purpose. The requirements that SSM does not meet can be handled by introducing other techniques. The second study, for example, considers how risk management and SSM can be combined. The third study examines the integration of SSM with traditional computing analysis techniques. This covers both structured analysis [12] and object-oriented development. Finally the fourth study looks at the use of formal description techniques as a means of strengthening the SSM process and assisting with the link to computing analysis techniques.

3.1 The Use of Soft Systems Methodology for Requirements Definition

Soft Systems Methodology (SSM) has emerged as the core of the RACE method. It meets two of the listed requirements in full and one in part:

1. It must deal with the broad problem situation and not be specific to computing

SSM is a general problem-solving technique. It is not specific to any problem area although recently it has been promoted extensively as an aid to the development of information systems [7]. Its areas of application have included public utilities, health, agriculture, industry, law, defence and scientific systems [13, 14]. SSM is a very well established technique. It was first conceived in the early 1970s and then refined by use in real problem situations, mostly in industry. It is now relatively stable. Overall then, SSM is a good match for the requirement for a technique that deals with the broad problem situation; it is particularly attractive because it has proven its value over many years.

2. It must support both goal-driven and problem-driven approaches to analysis

SSM is a strongly goal-driven technique. It focuses on the basic purposes of a system and on the behaviour necessary to achieve those purposes. Its particular strength is in dealing with 'messy' situations where there is uncertainty about the nature of the problem and/or disagreement about the options for its solution. In principle, all problems can be approached in this way and this provides a good counter to the common tendency to underestimate the extent of problems initially. However, although such a general approach can always be used effectively for any new problem situation, thereafter, it should only be used again when a major review is required. Thus, while SSM can provide a goal-driven technique for RACE, a problem-driven technique must also be found and integrated appropriately.

3. It must allow for multiple perspectives

SSM's support for multiple system perspectives is one of its major strengths. The first stage of SSM is to identify as many reasons as possible why a system exists and to express these reasons succinctly as system *root definitions*. Each root definition is expanded into a *conceptual model*, defining the activities necessary for the system to meet its purpose and also indicating relationships among the activities. The conceptual models are combined into a single *consensus model*, which when evaluated and refined subsequently becomes the *required system model* [15]. This latter model reveals the desirable system changes when it is compared with the problem situation. The SSM approach to multiple perspectives is particularly appealing because it concentrates on a small manageable number of identified system purposes, as distinct from considering the viewpoints of individuals in the problem situation, which are likely to differ for everyone involved.

In summary, SSM successfully addresses two and a half of the requirements for RACE. Of equal importance, is the fact that SSM has no features that are irrelevant to, or conflict with, the remaining requirements. Thus, SSM seems an ideal starting point for the development of RACE, particularly so because it also covers the very earliest phases of problem investigation and is well tried in practice.

3.2. Enhancing SSM With Risk Management Techniques

Risk is the possibility of loss or injury and risk management is the process of determining risks and responding to them. Risk management can be divided into two main activities [9]: *risk assessment* and *risk control*. Risk assessment involves (i) identifying the risks concerned; (ii) analysing the threat of those risks in terms of their probability of occurrence and the magnitude of the resulting loss; and (iii) prioritising the risks accordingly. Risk control covers (i) risk management planning, to develop plans for addressing each major risk; (ii) risk resolution, to implement the plan that will either eliminate the threats or keeping them at an acceptable level; and (iii) risk monitoring, to assess the effectiveness of risk resolution and take corrective action as necessary.

Risk is not addressed explicitly by SSM. Certainly SSM includes recommended practices that reduce risk, such as implementing system changes incrementally or having a monitoring and control mechanism within a system, but there is no direct, systematic treatment of risk. The expected benefits of considering risk explicitly are that (i) the process of system change should be smoother; and (ii) the resulting systems should be more robust. Risk management is particularly relevant to SSM because of its goal driven approach to analysis, which tends to identify substantial system changes, carrying relatively high risk.

Risk investigation, being a process of critical evaluation, is best considered after a creative phase of analysis yielding phase products ready for assessment and refinement. On that basis there are four stages of SSM where risk could be addressed:

1. When the problem situation has been described: this would consider risks in the current situation to identify opportunities for improvement and give a base for subsequent evaluation of change proposals.
2. When root definitions of relevant systems have been produced: root definitions contain a list of system constraints. Thus, identifying risks at the root definition stage can help to determine additional system constraints for inclusion in each root definition.
3. When conceptual models have been derived from the root definitions: risk can be considered (i) after each conceptual model has been constructed, to ensure that it has been adequately formed; and (ii) when the consensus model has been developed (by combining the individual conceptual models), to ensure that the resulting model deals adequately with the separate risks and to examine risks resulting from interaction and conflict among the models.
4. When feasible, desirable system changes have been determined: all proposals at this lower level of detail can be examined for risk and refined accordingly.

In practice, examining risk after the first problem analysis phase is not recommended because it requires the analyst to explore the problem situation in more detail than is desirable. The analyst must not be overly exposed to the problem situation, initially, to avoid the practical difficulty of being lured into modelling what exists rather than what should exist. Risks associated with the current system can instead be considered at stage three when conceptual models are being evaluated with respect to the problem situation.

Further details on the proposed risk management additions to SSM, and an example of its use, can be found in [16]. Overall the approach is appealing because it strengthens SSM without interfering with its basic operation.

3.3. Integrating SSM With Computing-Oriented Analysis Techniques

The possibility of using SSM with a computing-oriented analysis technique was proposed as long ago as 1985 [17] and has since been followed up by many related investigations [18, 19]. One significant development has been the recent establishment of a link between SSM and the government standard analysis and design method, SSADM [20]. This work was commissioned by the CCTA in the UK [21]. The link has been facilitated by the fact that there is a strong resemblance between the conceptual models of SSM and dataflow models of SSADM (similar to those found in any Structured Analysis technique): both are concerned with describing system behaviour and both are structured in a similar way. This connection has also been explored as part of the RACE study [15] and a technique for linking conceptual models and dataflow diagrams has been proposed. In summary, the proposal is:

1. Perform a complete SSM analysis, developing full hierarchical conceptual models and validating them to produce required system models.
2. Construct an *interaction table* for each model, identifying the interaction of each activity with other *processes*. A process is either an activity or a *repository* of information. A repository may be an individual, a document, a computer disc, or any other form of information consumer or producer.
3. Construct dataflow diagrams from the interaction tables and conceptual models. The interaction tables provide the technical content for the dataflow diagrams and the conceptual models give layout guidance. This is desirable to help the analyst understand the relationship between models.

This process of converting conceptual models to dataflow models is relatively straightforward and can largely be automated. The link between SSM and object-oriented analysis (OOA) is also being considered [22]. This is much more problematic, however, because SSM produces behaviour oriented conceptual models whereas OOA starts with static data oriented object models. Nevertheless, the ever growing popularity of OOA means that some satisfactory link must be found.

For any link it is always possible to take a *loosely coupled* approach, meaning that the knowledge gained in applying the first technique is use in the application of the second without any specified rules or guidelines. A check is then performed to ensure that the models produced in each case are consistent. Such an approach is unsatisfactory in several ways, including: (i) it is hard to teach; (ii) it is hard to apply consistently; and (iii) change control is difficult. A *tightly coupled* approach, like that suggested for dataflow modelling, is clearly preferable because it allows for some automation of consistency checking and can even assist with the initial development of the computing-oriented models.

The link for OOA is currently being investigated. In effect, this means defining a method of transforming SSM models into object models. A starting point has been to investigate existing OOA techniques to see to what extent they are compatible with SSM concepts [14]. This work has identified four techniques that provide a reasonable fit, namely those due to Jacobsen [23], Reenskaug [24], Booch [25] and Gibson [26]. Further research is being undertaken using a selection of these techniques.

3.4. SSM and formal descriptions

As indicated earlier, SSM conceptual models are imprecise: the meaning of each activity has to be deduced from the term used for its name, and the links between activities are really just indications of relationships, without any accompanying definitions. Precision can be added in the computing-oriented analysis phase but there is benefit in introducing it at the end of the SSM phase. Specifically, it can help to avoid misunderstandings about what exactly is being proposed as a result of the SSM analysis and so help identify problems earlier in the process. Precision in the SSM analysis can also strengthen the link with the subsequent computing-oriented analysis.

Work is currently being done on the particular case of supporting the specification of conceptual models using LOTOS, a process algebra-based formal description technique [27], in conjunction with developing a link to Structured Analysis. This combination is being considered because the underlying notations used by each technique are all based on describing behaviour and so are largely compatible. This will allow attention to be concentrated on the main issue of how best to manage the combined use of such a collection of techniques. Once this relatively homogeneous case has been examined the research will be extended to consider the use of less compatible formal descriptive notations and computing-oriented analysis techniques.

The research started by looking at the broad issue of the role of formal descriptions in the software development process [28]. In this field, much work has been done on the development of formal notations, rather less on the development of formal techniques and very little on the production of formal software development methods. The expression 'formal methods' perhaps needs some clarification. It is often used as a blanket term to cover the related activities of constructing, analysing, transforming, and verifying formal descriptions. The word 'formal' is used here in the sense that all of the activities have a sound mathematical basis. It does not mean, however, that there are step-by-step procedures identifying the precise way to perform the activities, either singly or in combination [29]. In that sense the word 'method' is misleading and 'technique' would give a rather clearer picture of what is intended.

Defining a relationship between SSM conceptual models and LOTOS processes gives a formal semantics to this currently informal notation. It also provides a means of saying more about the relationship between activities. Through LOTOS, the conceptual model activities can be treated a set of communicating processes and the order and effect of communication specified. The meaning of the resulting description can be portrayed as an *action tree* showing all possible behaviours as event sequences [30]. Also these permitted behaviours can be projected on to the conceptual models to provide a simulation that may facilitate communication with the client.

Another advantage of this work is that it gives a good starting point for computing-oriented analysis. The formal description provides a precise specification on which the subsequent development can be based and/or assessed. Also, since dataflow diagrams can be derived from SSM conceptual models in a largely automated way, the formal description provides a means of portraying system behaviour through an simulation of the dataflow representation. Again this may facilitate communication with the client.

4. CONCLUSION

This paper has summarised the current position of some research directed towards the identification of RACE, a better requirements definition method for computing systems. The main desirable attributes of RACE have been identified and explained, followed by a brief outline of four studies concerned with investigating techniques that are likely to form part of the method. SSM has emerged as the core of the method and the studies have all built around it. The influence of SSM has even permeated the language used in this paper (e.g. 'problem situation' and 'problem-owner'). This research is, however, at a relatively early stage. The implications of several of the requirements have still to be explored in detail and there are many remaining inter-connections between techniques to examine. Nevertheless, the results so far have confirmed the initial belief in SSM [32] and its combination with other techniques seems to be giving additional value without having any detrimental effect.

Acknowledgements

This research is being undertaken as part of the RUNES project (Requirements Understanding, Negotiation, Expression and Scrutiny), funded by the University of Ulster. RUNES, is broadly concerned with investigating techniques for improving the requirements definition process, and the development of tools to support that process.

I am very grateful to the many people who have contributed to and influenced this work in various ways, including Raymond Oakes, Des Vincent, Mark Norris, Dave Horncastle, Brian Carey, John Wright, Anthony Alston, Adam Winstanley, Sally McClean, Philip Morrow, Pat Lundy, Des Greer, Trevor Dobbin and Eoin Heslin.

REFERENCES

1. Humphrey W: *Managing the Software Process*, Addison Wesley, 1989
2. DTI: TickIT, *Guide to Software Quality Management System Construction and Certification*, Issue 2.0, Febuary 1992

3. Norris M, Rigby P, and Payne M: *The Healthy Software Project*, Wiley, 1993
4. Boehm BW: *Software Engineering Economics*, Prentice Hall, 1981
5. Davis AM: *Software Requirements, Object, Functions and States*, Prentice Hall, 1993
6. Checkland PB and Scholes J: *Soft Systems Methodology in Action*, Wiley, 1990
7. Wilson B: *Systems: Concepts, Methodologies and Applications*, 2nd Edition, Wiley, 1990
8. Wilkie G: *Object Oriented Software Engineering: The Professional Developer's Guide*, Addison-Wesley, 1993
9. Boehm BW: *Software Risk Management*, IEEE Computer Society Press, 1989
10. Gilb T: *Principles of Software Engineering Management*, Addison-Wesley, 1988
11. Cobb RH and Mills HD: *Engineering Software Under Statistical Quality Control*, IEEE Software, **7** (6), November 1990, pp 44-54
12. Yourdon, E.: *Modern Structured Analysis*, Prentice Hall, 1989
13. Mingers J and Taylor S: *The Use of Soft Systems Methodology in Practice*, Journal of the Operational Research Society, **43**(4), 1992, pp. 321-332
14. Dobbin TJ and Bustard DW: *Combining Soft Systems Methodology and Object-Oriented Analysis: The Search for a Good Fit*, Proceedings of the 2nd Information Systems Methodologies Conference, Edinburgh, August 1994
15. Bustard DW, Oakes R and Heslin E, *Support for the Integrated Use of Conceptual and Dataflow Models in Requirements Specification*, Colloquium on Requirements for Software Intensive Systems, DRA Malvern, May 1993, pp. 37-44
16. Bustard DW and Greer D: *Enhancing Soft Systems Methodology with Risk Management Techniques*, Proceedings of 2nd International Conference on Software Quality Management, Edinburgh, July 1994
17. Stowell F: *Experience with Soft Systems Methodology and Data Analysis*, Information Technology Training, May 1985, pp. 48-50
18. Mingers, J.: *Comparing Conceptual Models and Data Flow Diagrams*, Computer Journal, **31**(4), pp. 376-378, 1988
19. Stowell F (Ed.): *Soft Systems Methodology and Information Systems*, Systemist, **14** (3), August 1992
20. *SSADM Version 4 Reference Manual*, NCC/Blackwell, Oxford, 1990
21. CCTA: *Using Soft Systems Methodology for SSADM Feasibility Study*, 1994
22. Alston AJ, Wright PJ and Bustard DW: *COBRA: A Methodologv for CIS Object-Oriented Business and Requirements Analysis*, Proceedings of the 1st Information Systems Methodologies Conference, Edinburgh, September 1993, pp. 99-112
23. Jacobsen I, Christerson M, Jonsson P and Overgaard F, *Object-Oriented Software Engineering - A Use Case Driven Approach*, Addison-Wesley, 1993
24. Reenskaug T, Andersen EP, Berre AJ, Hurlen A', Landmark A, Lehne OA, Nordhagen E, Ness-Ulseth E, Oftedal G, Skaar AL and Stenslet P, *OORAA: Seamless Support for the Creation and Maintenance of Object-Oriented Systems*, Journal of Object-Oriented Programming, October 1992, pp. 27-41
25. Booch G, *Object-Oriented Analysis and Design with Applications*, Benjamin-Cummings, 1994
26. Gibson E, *Objects - Born and Bred*, Byte, October 1990, pp. 245-254
27. ISO - Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807, 1989.
28. Lundy P and Bustard DW: *Making Formal Methods Work: What's being Done and What Can be Done*, Proceedings of 2nd International Conference on Software Quality Management, Edinburgh, July 1994
29. Woodcock J and Loomes M, *Software Engineering Mathematics*, Pitman, London, 1988
30. Winstanley A and Bustard DW: *EXPOSE: An Animation Tool for Process-Oriented Specifications*, Software Engineering Journal, **6** (6), November 1991, pp. 463-475
31. Bustard DW and Winstanley AC: *Making Changes to Formal Specifications: Requirements and an Example*, IEEE Transactions on Software Engineering, Vol. 20, No. 8, August 1994
32. *SSM in Computing*, Workshop Notes, University of Ulster, April 1994