ELSEVIER

Information & Management 39 (2001) 101–107

INFORMATION & MANAGEMENT

www.elsevier.com/locate/dsw

Requirements acquisition for rapid applications development

Malcolm Eva*

Information Systems and Law, School of Accountancy, Information Systems and Law, University College Northampton, Northampton, UK

Received 15 November 1999; received in revised form 5 July 2000; accepted 7 February 2001

Abstract

There is a growing trend to develop Information Systems (IS) using rapid development tools and methods rather than using traditional analysis and specification methods. Some writers have warned of a return to an undisciplined style of systems development. This paper uses the discipline of 'requirements engineering' to validate the use of Rapid Application Development (RAD) techniques in identifying and responding to users' requirements for IS. It argues that traditional systems analysis depended on identifying explicit requirements that could be verbalized; many new IS, however, depend on identifying different forms of knowledge and requirements. Techniques employed by RAD are eminently suited to eliciting these requirements. The paper is founded in part on two empirical studies, which are viewed through a theoretical filter based on a framework from 'requirements engineering'.

Keywords: RAD; JAD; Prototyping; 'Requirements engineering'; ACRE; Tacit knowledge

1. Introduction

A series of surveys conducted in the latter half of the 1990s indicated that many companies are using various rapid development approaches for the development of their Information Systems (e.g. [24,25] in the US, and [8] in the UK).

The surveys indicated that a growing proportion of organisations used approaches that were called severally: rapid application development, rapid prototyping, prototyping, or evolutionary development. These terms were the organisations' own, and were not explained in the responses.

The findings raised two questions:

1. Are the terms synonymous?
2. Do they conceal a tendency to bypass the analysis and requirements process and move straight to coding?

A follow up study identified two models of rapid development: one claims to be disciplined, the other is characterized by its practitioners as JDI (Just Do It). What differentiates them is the attention given to requirements analysis and user participation. If the disciplined version of rapid development can be shown to be consistent with principles of 'requirements engineering' (RE), then it could claim to be a rigorous approach.

This paper focuses on one RE model, the acquisitions of requirements (ACRE) framework [14] in order to examine the validity of rapid application development as a disciplined development approach.

* Present address: 153 Ryeland Road, Northampton, NN5 6XJ, UK. Tel.: +44-0-1604-755948.
E-mail address: malcolm_eva@onetel.net.uk (M. Eva).

0378-7206/01/$ – see front matter
PII: S0378-7206(01)00082-9

2. 'Requirements engineering'

The phases of RE have been expressed as elicitation, expression and validation (e.g. [13,20]). Both the term 'requirements engineering' and its emphasis on correctness and completeness (implied in the term validation) suggest a positivistic view of a requirement as a discrete, identifiable entity. If this were so, then the systems analyst's job would be straightforward and sloppiness would be the probable reason for the inability of new systems to meet users' needs.

Traditional systems analysis and structured methodologies have used a set of techniques to elicit requirements; these include interviewing, questionnaire surveys and document searching. The analyst captured requirements by means of an instrument such as a problems and requirements list, which was completed during the interviews and other fact-finding activities. These proved a reliable means of identifying current activities.

This approach presupposes a tidy and unambiguous definition of requirements, which incorporates solutions to problems with the operations of the current system. It does not seem to be the case in practice. Goguen [9], for example takes a more epistemological view, and considers that a requirement emerges from the social interaction between analyst and user rather than emerging as a discrete entity. Rather than eliciting a requirements set from a single user, the analyst is seeking broader perspectives of business activities, and the information needed to support them.

These perspectives may be seen from a number of different viewpoints, and so it may now be necessary to explore the Weltanschaaungen of various stakeholders in the system. It is also recognised that early acknowledgement of a plurality of viewpoints is a valuable way of identifying unexpected requirements, and that it provides a basis for prioritisation and negotiation [4,22].

3. Acquisition of requirements (ACRE)

The ACRE framework was first published in 1996 [14]. It identifies and describes certain requirement elicitation techniques and maps them onto specific development circumstances.

ACRE is best described as a framework rather than a methodology. It lays down a basis for analysis, but does not identify stages, steps or sets of procedures that must be followed. It is concerned purely with requirements acquisition activities, not the other phases of the RE process. The framework includes 12 acquisition methods:

Observation	Brainstorming
Unstructured interviews	Rapid prototyping
Structured interviews	Scenario analysis
Protocol analysis	RAD workshops
Card sorting	Ethnographic methods
Laddering	Repertory grids

These techniques are described and discussed in [2].

This is not an exhaustive list of course, and neither do the authors claim it to be. Other methods, such as document searching or special record keeping, as practised in traditional systems analysis, can certainly find a place in ACRE, as can task analysis, CSF analysis or Wizard of Oz prototyping [15].

3.1. Knowledge types

Drawing on work in the field of knowledge acquisition for expert systems, ACRE recognises the following internal representations of knowledge:

1. *Future systems knowledge* is, by its nature, imperfect and incomplete. Much information will be readily available from the baseline requirements, but there will also be hidden data to bring out at a finer grained level that must wait to be identified through a process of discovery and discussion. In such cases, two techniques often employed in RAD may be valuable: scenario analysis and prototyping.
2. *Tacit knowledge* poses a particular challenge, as it is simply not susceptible to verbal explication in the same way as recognised knowledge. It is a term that comes from the work of Polanyi [17], who defined it as a piece of knowledge or a skill that a person possesses, but cannot make explicit. One form is that required to make expert judgements.

   A systems analyst required to specify a task based on tacit knowledge is faced with a

significant problem: talking through the task or answering structured questions are inappropriate ways to describe it. Another acquisition method must be found to draw out tacit knowledge. Observation is one way; ethnographic studies allow this opportunity but, if there are time pressures, this may be impractical. However, the workshop context enables a number of possible methods to be tried, such as scenario analysis and task analysis. Kyng emphasises the value of presenting the user with a prototype as soon as possible when attempting to design tasks with tacit knowledge. Techniques borrowed from Personal Construct Theory, such as Card-sorting or Laddering can help to uncover tacit, non-factual elements such as user attitudes or mind-sets [12].

An example of such knowledge was found in the work of knowledge engineers attempting to specify and build a knowledge-based diagnostic system for a brewery in the East Midlands, UK. One part of this project focused on the actual brewing process, overseen by a Head Brewer whose skill ensured a consistently high quality of product. His impending retirement was one trigger for the project. He was unable to explain how he achieved high quality, but was confident in his ability to oversee the production process better than anyone else. After he was shadowed for a period, and his every decision analysed, it was finally discovered, to his chagrin, that his special skill lay in recognising that a motor mixing the ingredients changed its tone when the consistency was at the right level for a new ingredient to be added. Neither the Head Brewer nor any one else realised at a conscious level that the motor had changed its pitch. Once his particular skill had been identified, the automation of that part of the process could be specified. The lessons about acquisition and construction of knowledge learned from that project can be found in [6].

3. Semi-tacit knowledge, especially that taken-for-granted, involves recognising what question has not been asked, because, from the information holder's view, it is too obvious to be worth mentioning. An example is given in the questioning methodology [19]. The authors describe a requirements document, drawn up by software engineers, for computerised support for loading bulk carriers of iron ore. This identified a number of key assumptions that, had they been left unchallenged, would have resulted in a loss of shipping, with accompanying loss of life. The teams responsible for the loading possessed knowledge to refute the assumptions, but neither party were aware of the mismatch until a different approach to requirements elicitation was established. Taken-for-granted knowledge was reported in 1975 by Grice [11], but little work seems to have been undertaken since to quantify its impact on the development process. Interviewing or description alone are unlikely to be fruitful; scenario analysis and task analysis may uncover the missing knowledge, but the expert/user may still not deem it worth mentioning. Prototyping is a more reliable method of identifying missing facts, in the second or third iterations final omissions or misunderstandings are likely to be identified.
4. Non-tacit, or explicit, knowledge is easier to draw out of a subject; traditional structured interviews, record searching and observation are likely to provide the information needed for a requirements document, but in a time-consuming manner.

4. Background to rapid application development research

Rapid application development (RAD) is not new. In its original guise as 'rapid iterative production prototyping', developed by the Du Pont organisation, or as an aspect of Information Engineering, RAD was intended to be a disciplined and effective approach [16]. With the growth of prototyping and development tools, especially the spread of visual programming surveys show that this approach has gained in popularity in the UK and US.

The variety of terms used by respondents in the surveys raised questions as to what was described, especially with the appearance of JDI among the responses.

JDI was cited as a favoured approach by a number of systems developers in small to medium enterprises (SME) during an informal survey conducted among practitioner developers. The inference of JDI, consistent with the responses to Eva and Guilford's survey is that analysis and design of the problem situation are superfluous, and the time is better spent in the

production of data files and the code to maintain them. The significance lies not in the fact that the companies concerned omitted the use of any named analysis methodology (in common with 31% of respondents to the survey), but that the *activity* of analysis was held to be a barrier to timely production of the system.

To clarify the issue, respondents from seven organisations were interviewed in a follow-up study, to establish any commonality in the terms. The interviewees comprised IS managers or senior analysts for the development side, and representatives from the end-user side. All interviews took place in medium to large organisations.

One company was an engineering firm, one a domestic manufacturer, one a major insurance company, one a local council; two were major retailers and the seventh was an American manufacturing company preparing to expand into the European market.

The engineering firm's use of prototyping proved to be JDI based, and so it was dropped from further study. The other companies all showed disciplined rigor in their procedures; they had also adopted RAD (evolutionary prototyping, prototyping) as an organisational strategy in their new systems development and accordingly had published in-house procedures and standards. As a result of interviews, the following umbrella definition was proposed:

> A method of developing information systems that involves: a development team incorporating end-user clients and IS specialists, the use of rapid development software tools, and the staged delivery of a working system by means of iterative prototyping of solutions to business requirements. A RAD project will also involve the use of high level workshops such as Joint Applications Development (JAD) and Joint Requirements Planning (JRP) and the use of time-boxing for each delivery in the project [7].

The elements in this working definition, especially the membership of end-users in the design team and the use of high-level workshops, are those that differentiate RAD from JDI.

There has been a suspicion among some more sceptical parts of the practitioner community that RAD and JDI are synonymous e.g. [3].

While RAD is being taken up with increasing enthusiasm by companies, as shown by surveys and interviews, there are also concerns in the community about its efficacy in terms of meeting requirements, especially non-functional ones. There have been warnings that the need to get to market can over-ride the need for disciplined, accurate requirements specification, quality control and thorough testing [23].

Concern over these issues has led to the implementation of a de facto standard in the UK: dynamic systems development method (DSDM) [5]. This makes explicit attempts to address any reservations by building iterations of testing cycles and validation of business requirements into its structure.

Notwithstanding, the work of DSDM and the growing adoption of RAD approaches, there is still scepticism as to its efficacy, especially in meeting requirements. If RAD is to be regarded as a valid, disciplined approach to developing systems, rather than as a shortcut, it must meet certain criteria: it must significantly reduce production time [1]; it must deliver systems of quality; it should deliver systems that are maintainable and it must be able to capture and meet the user's requirements. Empirical evidence gives conflicting views on the first [10]; there is more work to be done to give more precise data on its success in shortening delivery time.

The IS developer interviewees identified maintainability as the major problem to be overcome, though the systems delivered were of acceptable quality. This suggests that RAD projects met their objectives.

Evidence from such a small sample may be regarded as anecdotal, however, and may give only a limited view as to RAD's validity for requirements acquisition. DSDM, acknowledges the importance of negotiation between stakeholders when identifying project objectives, and the iterative cycles allow regular revisiting of the requirements for re-appraisal. After three such iterations, what surety do the sponsors have that the final requirements bear a close resemblance to the original needs? DSDM includes a validation against original business objectives as a part of its testing cycle, but this is not a prescriptive method; it is quite feasible that, with time pressures, testing may stop at the technical acceptance phase of the development.

To validate RAD's rigor as a means of capturing requirements, it is necessary to identify what methods of elicitation are commonly proposed for RAD projects, and see how well they map onto the appropriate knowledge representations.

All companies interviewed for the study into RAD practice agreed with the importance of specific techniques. While iterative prototyping was generally felt to be the driver for both validation of requirements and production of working systems, a workshop was the principal method for eliciting functional requirements in the first instance.

While the workshops were the forum, the actual techniques varied. Interviews, both structured and unstructured, did not feature in any of the studies. The techniques included brainstorming, task analysis, scenario analysis and critical success factor analysis. The levels of granularity of the requirements extracted by these methods were markedly different: brainstorming provided high level requirements only, while critical success factor analysis, task analysis and scenario analysis yielded precise requirements. Prototyping, an activity sometimes carried out between meetings, provided very fine-grained requirements, and it was the medium for identifying what had been omitted and overlooked, particularly the taken-for-granted requirements.

This is justified in that both scenarios and prototypes are simulations, or projections, of the new system and its transactions with the environment. Wherever the developer's projection is at odds with the user's expectations, a new requirement is recorded. The authors of ACRE use the term RAD to mean workshop, rather than the entire repertory of development activities. In this paper the term RAD is used in the broader sense already discussed.

A discordant note, however, was struck in a study by Purvis and Sambamurthy [18]; they found that designers and end-users were not convinced of the effectiveness of JAD workshops as a design forum, and reported that the interaction between the two groups was less smooth than expected. They presented a number of possible reasons for this, such as the intransigence of some designers who are more used to traditional methods. Another reason is that many designers are not very experienced in facilitating such workshops; DSDM states that it is not the designer's role to be the facilitator, a disinterested third party with facilitating skills should help manage the conflicts and negotiations that are likely to arise (Table 1).

This result is close to the prediction made by the authors of ACRE, but they were more cautious in their verdicts on scenario analysis and prototyping in the domain of taken-for-granted knowledge, rating them as merely ''fit''.

Table 1
The mapping of the techniques employed by RAD with the types of knowledge described in the ACRE framework is shown[a]

Elicitation method	Tacit	Semi-tacit	Non-tacit	TFG
Prototyping	YY	Y	YY	YY
JRP	N	N	YY	N
JAD	Y	Y	Y	Y
Task analysis	YY	YY	YY	YY
Scenario analysis	YY	YY	YY	YY
CSF	N	Y	Y	N

[a] While the axes are based on an example in ACRE, the entries are drawn from the interviewees' responses. YY: strong fit; Y: fit; N: no fit, or weak fit.

It may be argued that JAD and JRP, as the fora for employing other techniques, do not themselves belong on this table. Their inclusion is justified by the nature of the workshops, both being dedicated to identifying and refining requirements instead of the more traditional requirements elicitation.

Traditional texts on systems analysis (e.g. [21]) enumerate a different repertoire of techniques for fact-finding: interviews, questionnaires, observation, record-searching, special purpose records and sampling. For the automation of data processing tasks, these generally proved satisfactory; for providing IS support for organisations that are already computerised, and where tacit and semi-tacit knowledge is involved, they are less helpful. Where a Greenfield system is in development, they are less appropriate (Table 2).

This repertoire is clearly very strong for explicit requirements, and simple automation of existing known tasks. The entry for observation suggests that it will uncover both types of knowledge. In fact,

Table 2
The corresponding mapping of the traditional fact-finding techniques onto the different knowledge types is shown

Elicitation method	Tacit	Semi-tacit	Non-tacit	TFG
1-1 Interviews	N	N	YY	Y
Questionnaire	N	N	Y	N
Observation	YY	Y	Y	YY
Record searches	N	Y	Y	YY
Sampling	N	N	Y	N
Special purpose records	N	Y	Y	Y

according to Polanyi the situation is not so simple; the observation will make the observer/analyst aware that knowledge exists to be captured, but that is not sufficient to impart the knowledge — tacit knowledge requires an indwelling. In such a case, observation could be supplemented by a technique such as laddering or repertory grids, which would help clarify the rationalising process of the knowledge holder. None of the interviewees involved in the study were aware of either such technique, which is why they are not included in the table. For future systems knowledge, most of the traditional techniques do not apply.

5. Implications

The question to address must be: What impact should these ideas have on business? The early generation of computer implementation, reducing staff by automating manual procedures, is a matter of history. In 2001, there are few businesses larger than 20 people in size where major parts of day-to-day administration, if not the core business, is not carried out by computer. These systems were built from requirements based largely on non-tacit and semi-tacit knowledge.

Later generations involving e.g., development of Management Information Systems, or Intranets have a more problematic set of requirements. If one-to-one interviews cannot easily elicit the types of requirements and types of knowledge involved, the analysts and designers need to be aware of, and trained in, the broader forms of requirements acquisition designed to produce a congruent specification.

The increase in the use of RAD developments, driven largely by the growing demands for rapid business change, also makes more urgent the dissemination of the requirements tools needed to ensure rigor of analysis with speedy time to market.

6. Future work

While there is some empirical support for the position maintained in this paper, there is need for a larger scale study of how RAD is applied and how the requirements elicitation toolbox is applied. This might entail a combination of a set of surveys to form an idea of the scale of any problem, with ethnographic studies to form a detailed picture of how well the two disciplines interact in practice.

There is also scope to explore further Grice's descriptions of taken-for-granted knowledge, and to discover how much of a problem in delivering congruent systems these oversights present.

7. Conclusions

The different knowledge representations with which the AI and 'requirements engineering' communities work illustrate the difficulty that requirements analysis can encounter. Traditional systems analysis, which focused on automating data processing, looked mostly at non-tacit or semi-tacit knowledge, and in these cases a fixed set of requirements were identifiable. The positivistic view implicit in 'requirements engineering' is appropriate for this. However, if a system is to provide information support rather than simply to automate, or is to enable business re-engineering, or aid competitive advantage, the requirements may start as more coarse-grained and later need a heuristic repertoire of elicitation and validation techniques. These may well involve a level of negotiation between stakeholders.

Many RAD projects fall into this category where the knowledge that RAD has to address is more susceptible to heuristic, iterative approaches than to traditional systems analysis techniques. The JRP/JAD workshop helps to pull out tacit and semi-tacit knowledge factors behind certain of the requirements as a necessary pre-requisite for prototyping. This aspect of RAD separates it from JDI: the Weltanschauung adopted and the otherwise hidden requirements are already established before the development begins.

This conclusion would seem to validate rapid application development as a rigorous approach to systems development.

Acknowledgements

I would like to thank Gordon Rugg of University College Northampton for his assistance and advice on this paper, and Professor Sibley of Information and Management for his helpful suggestions and diligent editing.

Thanks are also due to Bryan Bennett, Laurence Brooks, Guy Fitzgerald, Ita Richardson, and Richard Vidgen for their helpful comments on an earlier version.

References

[1] D. Card, Is Timing Really Everything? IEEE Software, 1995, pp. 19–22.
[2] E. Cordingley, Knowledge elicitation techniques for knowledge-based systems, in: D. Diaper (Ed.), Advances in the Psychology of Human Intelligence, Ellis Horwood, Chichester, UK, 1989, pp. 89–175.
[3] J. Daniels, Why RAD is BAD, Presentation at Meeting of RESG, Imperial College, London, July 1996.
[4] P. Darke, G. Shanks, User viewpoint modeling: understanding and representing user viewpoints during requirements definition, Information Systems Journal 7, 213–239.
[5] DSDM Manual, DSDM Consortium, Farnham, 1995.
[6] M.J. Elliot, et al., 1995, Constructive Knowledge Engineering, Knowledge-Based Systems, Vol. 8, No. 5.
[7] M. Eva, 1997, Strategic rapid development, in: J. Vorisek (Ed.), Proceedings of the 5th Conference of Czech Systems Integration Society, Prague, Journal Pour (1997) 47–58.
[8] M. Eva, S Guilford, Committed to a radical solution, in: B. Fitzgerald, N. Jayaratna (Eds.), Proceedings of the 4th BCS ISM Conference, Cork, 1996, pp. 87–96.
[9] J. Goguen, Formality and informality in requirements engineering techniques for requirements elicitation, in: Proceedings of the IEEE International Conference, IEEE CS Press, California, 1996.
[10] V. Gordon, J. Bieman, Rapid Prototyping: Lessons Learned, IEEE Software, 1995, pp. 85–95.
[11] H.P. Grice, Logic and conversation, in: P. Cole, J. Morgan (Eds.), Syntax and Semantics, Vol. 3, Academic Press, New York, 1975.
[12] G. Kelly, The Psychology of Personal Constructs, Norton and Co. Inc., 1955.
[13] P. Loucopoulos, Karakostas, Application Requirements Engineering, McGraw-Hill, Maidenhead, 1995.
[14] N.A.M. Maiden, G. Rugg, ACRE: selecting methods for requirements acquisition, Software Engineering Journal (1996) 183–192.
[15] M. McGuire, Respect User Requirements Framework Handbook, 1997.
[16] J. Martin, Rapid Application Development, Macmillan, New York, 1981.
[17] M. Polanyi, Tacit Knowledge in Managerial Success, The University of Chicago Press, Chicago, 1966.
[18] R. Purvis, V. Sambamurthy, An examination of designer and user perceptions of JAD and the traditional IS design methodology, Information and Management 32 (1997) 124–134.
[19] G. Rugg, P. McMaster, Questioning methodology, Working Paper No. 99/03, Faculty of Management and Business, University College Northampton, Northampton, 1999.
[20] J. Siddiqi, M. Shekaran, Requirements Engineering, The Emerging Wisdom, IEEE Software, 1996, pp. 15–19.
[21] S. Skidmore, Introducing Systems Analysis, NCC/Blackwell, Manchester, 1994, pp. 74–85.
[22] R. Vidgen, Stakeholder analysis, soft systems and eliciting requirements, Information Systems Journal 7, 1997, pp. 27–46.
[23] R. Vidgen, C. Spedding, T. Wood-Harper, The limitations of software quality: toward a model of information systems quality assurance, in: B. FitzGerald, N. Jayaratna (Eds.), Proceedings of the 4th BCS ISM Conference, Cork, 1996, pp. 415–426.
[24] J. Wetherbe, N. Vitalari, A. Milner, Key trends in systems development in Europe and North America, Journal of Global Information Management 2 (1994).
[25] J. Wynekoop, N. Russo, Systems development methodologies: unanswered questions, Journal of Information Technology 10 (2) (1995).

Malcolm Eva is an independent consultant, specialising in 'requirements engineering' and systems analysis. He was previously a senior lecturer in Information Systems at University College Northampton, UK, where he carried out the research that led to this paper. His other research interests are concerned with the cultural impact of IS and IS development approaches. His industrial experience includes many years in the Ministry of Defence and British Telecom (BT) as a systems analyst and consultant.