Software Engineering Journal, May 1996.

ACRE: Selecting Methods For Requirements Acquisition

N.A.M. Maiden & G. Rugg[1]

Centre for Human-Computer Interface Design
City University, London UK.

Synopsis

A framework is presented which assists requirements engineers to choose methods for requirements acquisition. Practitioners are often unaware of the range of methods available. Even when practitioners are aware, most do not foresee the need to use several methods to acquire complete and accurate requirements. One reason for this is the lack of guidelines for method selection. The ACRE framework sets out to overcome these limitations. Method selection is achieved using questions driven from a set of facets which define the strengths and weaknesses of each method. The framework is presented as guidelines for requirements engineering practitioners. It has undergone some evaluation through its presentation to highly-experienced requirements engineers. Some results from this evaluation have been incorporated into the version of ACRE presented in the paper.

1: The ACRE Framework

Requirements engineers face a major problem when choosing methods for acquiring requirements of software-intensive systems. The problem is not that there is a lack of methods, since a wide range exists from the ethnographic (e.g. Sommerville et al. 1993, Goguen & Linde 1993) to the constructivist (Shaw & Gaines, in press). Rather, no guidance is available to choose methods, to plan a systematic, well-grounded acquisition programme, or even to sequence methods. Indeed, many requirements engineers are unaware of the methods which are

[1] Current Affilitation: School of Computing Science, Middlesex University, London

available. This paper describes a framework for such a purpose. It is called ACRE (ACquisition of REquirements).

Requirements engineers are familiar with methods such as observation, interviewing and using documentation. However, each is insufficient in isolation to capture complete requirements. Consider an example of a real software system to plan vehicle loading which one of the authors observed. A large system specification had been developed from existing documentation. However, several minutes of direct observation revealed to a team of software engineers that a process which they had assumed to be completely regular could, in fact, speed up and become dangerous. A new requirement to warn the user when this happened was complicated by further observed phenomena such as extreme glare, loud noises and extreme vibration. This paper argues that more than one acquisition method is needed to capture the full range of complex requirements for most complex software-intensive systems. One assumption is that effective requirements acquisition is not simply a by-product of using existing structured methods and notations (e.g. Cutts 1987). Rather the ACRE framework provides guidelines for selecting from a broad range of different methods with different features from different backgrounds.

To complement familiar methods, some researchers posit use of ethnographic methods (e.g. Sommerville et al. 1993, Goguen & Linde 1993). Such methods have received considerable recent interest in requirements engineering (e.g. Luff et al. 1993). Knowledge engineering has also provided useful methods, such as card sorts and laddering (Gammack 1987, Rugg et al. 1992, Rugg & McGeorge, in press). Indeed, requirements engineering and knowledge engineering share many concerns. Knowledge engineers now build larger knowledge-based systems using structured methods (Schreiber 1994) and integrate these systems with conventional systems (Brameur 1990). In requirements engineering, there been a focus on knowledge representation languages (e.g. Greenspan et al. 1986) and attempts to bring knowledge acquisition methods into requirements engineering (Maiden & Rugg 1994). This paper brings acquisition methods from software engineering, knowledge engineering and the social sciences together into a single framework. Similar work in the knowledge engineering area (e.g. Corbridge et al. 1992) has been more on method comparison rather usefulness. In this sense

ACRE also differs from previous requirements engineering classifications (e.g. Bickerton & Siddiqi 1993) in its focus on the acquisition process.

ACRE is also more than a simple collection of methods. Its core guidelines for method selection are derived from theories of cognition (e.g. Anderson 1990) and social interaction (e.g. Kelly 1955), and empirical data which supports these theories. The theories provide a rigorous, well-founded starting point for method selection. However, evaluation of these guidelines is also needed. An initial evaluation is reported in the paper.

The remainder of this paper presents the framework. The next section defines requirements acquisition and, hence, ACRE's boundaries. This is followed by presentation of the framework itself, in the form of guidelines for method selection. A decision support system to enable validation of ACRE is outlined. The paper also reports feedback from experienced requirements engineers which has resulted in changes to ACRE's guidelines. The paper ends with future extensions to ACRE.

2: A Definition of Requirements Acquisition

ACRE was bounded using a definition which separates acquisition from other requirements engineering activities. ACRE provides methods for acquiring requirements from stakeholders, rather than for mining requirements out of documents (e.g. Goldin & Berry 1994). This is not intended to diminish the importance of documentation. Indeed, knowledge acquired through communication will often be interpreted using existing documents or documented rationale, for political or other reasons. Each acquisition method in ACRE aims to improve communication between stakeholders. One of the stakeholders must be a requirements engineer who has been trained in use of that method. However, ACRE also recognises that requirements acquisition, negotiation and agreement are often interleaved, so although it provides methods for acquisition, improvements in requirements negotiation and agreement are unplanned but welcome. Specific methods and techniques which aim to improve negotiation are discussed elsewhere (e.g. Boehm et al. 1994).

ACRE proposes methods to acquire both requirements for the software system and knowledge about that system's domain and environment. The environment is described by diverse phenomena in the problem domain such as behaviour, events, structure and states. This version of ACRE also does not include participative design (e.g. Clement & Van den Besselar 1993) which necessitates wider organisational and cultural changes. Rather, it proposes methods which can be used by practitioners with little training within the existing organisational and cultural context.

3: The Framework

The framework offers 12 acquisition methods. Figure 1 describes each method, preconditions for its use and perceived strengths and weaknesses. Useful references for would-be users are also listed. The framework does not claim that these are the sole methods for requirements acquisition, but rather a representative sample of the types available. Most of these methods are comparable in their objectives, duration and manpower needed. RAD and ethnographic methods, though, are larger techniques which have been included for completeness, although this makes systematic method comparison more difficult. However, our goal is practical guidance, so the tables presented in this paper are intended to provide guidelines with practical use in mind.

FIGURE 1: SEE APPENDIX

Requirements for software-intensive systems are complex and varied. This is reflected in the framework which suggests different methods to acquire different knowledge and requirements. Research has identified six facets which inform method selection:

- *purpose of requirements*: requirements can be acquired for different purposes, such as specification of bespoke systems, selection of software packages and to provide a legal contract for requirements procurement. Different methods assist each;
- *knowledge types*: requirements modelling languages (e.g. Greenspan et al. 1986) include semantic primitives such as events, states and agents. Different methods acquire different types of knowledge;

- *internal filtering of knowledge*: it is often the case that stakeholders are unaware of their own knowledge and its boundaries. Problems can include poor recall and communication of incomplete or incorrect knowledge. Methods are offered to overcome these limitations;
- *observable phenomena*: some knowledge cannot be communicated by stakeholders but only learned by observing a system and its environment;
- *acquisition context:* method choice also depends on the context of its use. Acquisition does not occur in a vacuum. Complex organisational, political, financial and temporal pressures influence acquisition. Method selection in ACRE recognises this;
- *method interdependencies*: an acquisition programme will include a sequence of methods which influences method choice.

In most cases, ACRE has been designed to aid method selection for one acquisition session. One session is defined as an uninterrupted communication between stakeholders who do not change throughout the session. Several methods can be selected for one session. However, some methods do not involve planned sessions. Observation and ethnographic methods tend to be studies of the workplace which have a long duration, and their selection is intended for such. Furthermore, the sixth facet, method interdependencies, constrains method choice for one session by taking into account methods used in previous sessions and planned for future sessions in a programme.

The next section describes each of the facets, beginning with the purpose of requirements.

3.1: The Purpose Of Requirements

Requirements for software-intensive systems are acquired for three basic, sometimes overlapping purposes: (i) to provide a specification for design and implementation of a bespoke system (e.g. Gotel & Finkelstein 1995); (ii) to enable selection of commercial off-the-shelf software packages (e.g. Finkelstein et al. 1996), and; (iii) to provide a legal contract for system procurement (e.g. Boocher 1990). Whereas acquisition for bespoke systems is wide-ranging and can benefit from using all methods, requirements for better package selection and procurement can be acquired through effective method selection.

Package selection: purchasing a software package often involves selection (e.g. Finkelstein et al. 1996). In such cases, requirements for the new system should act as selection criteria, so each requirement should be defined as a separate criterion. Selection can be aided if these requirements/criteria are ranked by importance. Furthermore, if candidate packages are known, the acquisition session can capture the fit between each package and requirement, thus providing a basis for selection. Guidance for choosing methods is shown in Table 1. The most effective methods when candidate packages are known are card sorting and repertory grid analysis. Unlike most methods, card sorts explicitly encourage respondents to give criteria which discriminate between things such as software packages (e.g. Gammack 1987). Attributes in repertory grids have a similar role (e.g. Shaw & Gaines, in press). Rapid prototyping, in the loosest sense, is also effective for package selection. This is because software packages provide full working prototypes which can be exploited to improve acquisition and validation of requirements. On the other hand, if software packages are not available, sorting cards which have descriptions of these packages on them has been shown to be effective when the respondent is familiar with the packages and/or their main functions (Maiden & Rugg 1994). Another powerful method in such circumstances is laddering, which has also been shown to acquire criteria which can act as requirements for package selection (Maiden & Rugg 1994). Laddering uses probes to acquire requirements and selection criteria in an explicit, semi-structured form. Other methods such as interviewing can also acquire such knowledge, but only laddering makes definition of criteria an explicit goal.

TABLE 1: SEE APPENDIX

Requirements procurement: legal contracts must often be expressed in natural language. Indeed, the use of this language is very precise and open to misinterpretation by requirements engineers. On the other hand tables, figures and sketches are not part of the legal document. Therefore methods producing natural language output are preferred if the method is producing requirements which will be part of the contract. However, output from methods ranges from natural language, via restricted language, to closed sets. Furthermore, observational techniques, and those with natural language as output, often require coding which

is error- and bias-prone and time-consuming. Table 1 also shows the form of output from each method and methods to use for requirements procurement. Interviewing, scenario analysis and RAD all provide requirements in text form which is suitable for a legal contract, whereas the output from other methods often needs further processing into a suitable form. However, in the future, it will be important to stress the need for multi-form legal documents. Some current method notations (e.g. Jacobson et al. 1992, Cutts 1987) stress communication without considering the contractual role of the specification. Requirement notations must provide affordances for both roles, with notational extensions to handle intentional redundancies in specifications.

3.2: The Type of Knowledge

The type of domain knowledge to acquire is another factor influencing method choice. ACRE follows standard software engineering practices (e.g. Cutts 1987, Rumbaugh 1991) and divides knowledge into three broad types (Ashworth & Goodland 1990): behaviour, process and data. Most methods are effective for acquiring behaviour and process knowledge since it can often be observed (e.g. Goguen & Linde 1993), verbalised (e.g. Ericcson & Simon 1984), and communicated during interviews (e.g. Cordingsley 1989), see Table 2. However, acquiring data is more problematic as most stakeholders are more aware of their own actions than of the information around them. ACRE suggests that card sorting and laddering are often the most effective methods for acquiring data. Both, unlike other methods, explicitly acquire categorial and hierarchical knowledge about domains which is useful for defining classes, entities and attributes (e.g. Chi et al. 1982, Corbridge et al. 1992). RAD is also encouraged due to its reliance on structured notations such as entity-relationship modelling.

TABLE 2: SEE APPENDIX

3.3: The Internal Filtering of Knowledge

Interviewing is a traditional method for acquiring requirements. However, it assumes that the stakeholder has conscious, accurate access to all relevant knowledge. In fact though, this is often not the case. Much knowledge is not accessible to conscious introspection, and of the knowledge which is, much may be missed (Diaper 1989). ACRE suggests methods to overcome specific

limitations, and argues that several methods will often be needed to acquire complete requirements.

The first factor to consider is whether the method must acquire knowledge about the existing domain, requirements for the new system, or both. This distinction is highlighted in Jackson (1995). Different methods are better for acquiring knowledge about each. ACRE suggests that scenario analysis, prototyping and RAD are more effective for acquiring requirements for new systems, see Table 3. There are several reasons for this. Scenarios and prototypes (e.g. Regnell et al. 1995, Acosta et al. 1994) are both simulations of the required system and its interaction with the environment, and hence provide more effective cues for recall of knowledge (e.g. Baddeley 1982) and generating new requirements (e.g. Graham 1994). RAD's structured analysis notations also provide models with a similar purpose. However, the poor predictive power of people's introspections means that such methods should be used with care (Wicker 1969). No one acquisition method can provide a complete solution to such a complex problem. When considering existing domain knowledge, the framework makes a broad distinction between non-tacit, semi-tacit and tacit knowledge. Non-tacit knowledge is accessible to accurate introspection using interviewing under reasonable circumstances. Semi-tacit knowledge is accessible, but only with appropriate methods. Tacit knowledge is not accessible to any form of valid introspection. Attempts to elicit such knowledge will only acquire reconstructed accounts, which are unlikely to be valid except by accident. Most requirements acquisition sessions will encounter tacit, semi-tacit and non-tacit knowledge. However, the requirements engineer needs to recognise where such knowledge does or might exist, and select methods accordingly.

Non-tacit knowledge: recall of non-tacit knowledge is subject to numerous biases. At a trivial level, shortcomings include more likelihood of remembering items early or late in a sequence rather than in the middle of it. A more serious example is that even domain experts may only selectively recall examples which fit with their preconceived schemata, and forget examples which do not (Kahneman et al. 1982). ACRE proposes using most verbal methods for acquiring non-tacit knowledge, see Table 3. People, if they are able and willing to recall and communicate knowledge, will do so primarily using verbal communication,

although other methods, such as modelling, can support such communication. However, stakeholders tend also often deliberately withhold knowledge for political or personal reasons. In such cases, more direct methods such as observation can be more effective, see section 3.4.

TABLE 3: SEE APPENDIX

Semi-tacit knowledge: the main factors for semi-tacit knowledge are recognition, taken-for-granted knowledge and working memory processes:

- recognition of non-tacit knowledge differs from recall in the access to knowledge (e.g. Baddeley 1982). Recall often accesses only a fraction of the knowledge which can be recognised, as shown in work by Tulving & Osler (1968) and their contemporaries. For example, most people are able to recall only a few function names of a software package which they routinely use, but are able to recognise all of them if shown a list. Table 3 highlights methods which provide good cues for knowledge recognition in the form of rich semantic models. Rapid prototyping, scenario analysis and RAD are the most effective methods because prototypes, scenarios and, to a lesser extent conceptual models, can provide suitable cues for recalling and recognising knowledge. For example, prototypes can often remind stakeholders of low-level functions which are often otherwise overlooked. On the other hand, observation and methods which use verbal communication are less effective because such methods do not cue recognition;
- taken-for-granted (TFG) knowledge refers to the filtering process in communication (Grice 1975) which leads to the omission of information which can be taken for granted (e.g. one would not say: "My computer, which is a machine...."). Stakeholders often treat knowledge as TFG when it involves everyday things, forgetting that it may be unknown to outsiders such as requirements engineers. TFG knowledge is best acquired using methods which do not rely on communication such as observation, protocol analysis and ethnographic methods, see Table 3. Indeed, this is one of the main arguments for ethnographic methods (Goguen & Linde 1993). Other methods which do not depend on verbal communication (scenario analysis, rapid prototyping and

RAD) can also acquire TFG knowledge by exposing omissions, errors and inconsistencies in scenarios, prototypes and semantic models;

- working memory is a limited-capacity, short-duration memory for information used during a task which is forgotten within a few seconds unless there is some reason to commit it to long term memory. It can provide important information about processes and their rationale. Table 3 shows that only protocol analysis, which combines observation with verbalisation of thought processes, can capture knowledge in working memory (see Ericcson & Simon 1984 for details).

Tacit knowledge: tacit knowledge subsumes compiled and implicit knowledge. There is considerable debate in the literature about these, but the broad trends are well agreed. Compiled knowledge is knowledge which was once non-tacit, but which has since been practised so often that it has become habitualised and speeded up to the point where accurate introspection is no longer possible. It is well-understood using theories of cognition such as ACT* (Anderson 1990). An everyday example is using a direct manipulation interface, which involves considerable thought for a novice user, but which an experienced user can do without even being aware of the steps involved. Implicit knowledge is knowledge which has been acquired without the individual ever having accurate conscious awareness of what was being learnt and how (Seger 1994). Much of the knowledge usually termed "gut feel" is of this sort. Again, such knowledge is not often available through verbal channels of communication. ACRE promotes use of observational methods (observation, protocol analysis, ethnographic methods) for acquiring both compiled and implicit knowledge (see Table 3) because the respondent does not need to be aware of such knowledge. It is also worth noting that requirements engineers themselves are also liable to fail to communicate tacit knowledge about, for example, notations, methods and concepts because they assume that other stakeholders already know such knowledge.

3.4: Observable Phenomena

Not all knowledge and requirements can be acquired from stakeholders. Methods are also needed to acquire observable phenomena such as domain objects and processes. Domain objects are things in the domain such as tangible things like artefacts (e.g. Bright et al. 1995) and abstract entities like social networks (e.g. Yu

1993). Processes may be very different from those reported by stakeholders. In most professions there is a difference between the "front" (i.e. public consumption) version of how things are done and the "back" (i.e. behind the scenes) reality (Goffman 1959). It is essential to understand the back version. Individual processes are the "how" and "what" of what the individual does. It is often the case that users tackle tasks using modalities of which they are unaware. Observation and ethnographic studies are the obvious methods for acquiring observable phenomena (see Table 4), for obvious reasons.

TABLE 4: SEE APPENDIX

3.5: Acquisition Context

Acquisition does not occur in isolation but as part of a larger requirements engineering process. This process determines the context in which acquisition takes place, and imposes constraints on method use and hence selection. The most important constraints on method use are temporal. Each acquisition session requires time to prepare, undertake and analyse. The most desirable method should be quick to set up, run and obtain requirements and knowledge from. Other constraints are linked to the manpower available. Skilled requirements engineers are in short supply (Curtis et al. 1988) and other stakeholders do not want to spend a long time away from their workplace. Therefore, the most desirable method should need only one requirements engineer and one other stakeholder. Some methods also need technological support. Most desirable methods should not require much hardware and other tools such as video cameras (e.g. Bright et al. 1995). In some cases an acquisition session cannot be organised, thus limiting the choice of method which is applicable. Furthermore, introduction of new software systems alters the organisation and power structures within an organisation (e.g. Yu 1993), with the consequence that stakeholders can feel threatened and insecure. Their first point of contact with this threat is often during requirements acquisition. Therefore, although this framework cannot hope to resolve all social problems which arise, method selection should be tailored to avoid alienating stakeholders. The most desirable method should not antagonise or bore stakeholders, or be too intrusive. ACRE defines guiding limits on use of each method, see Table 5.

TABLE 5: SEE APPENDIX

Reports of method use indicate that most are quick to prepare and use in a single session (e.g. Cordingsley 1989, Ericsson & Simon 1984, Rugg & McGeorge 1992, Corbridge et al. 1992, Takeda et al. 1993). Requirements for most large and complex software systems cannot be acquired in one session (e.g. Gough et al. 1995), so session duration is often determined by contextual factors such as manpower available. However, some methods need longer preparation. Building even a small set of scenarios takes considerable time and effort (Gough et al. 1995). Small rapid prototypes need time to plan, implement and test before use, although this time is often longer (e.g. Acosta et al. 1994). RAD needs at least one week preparation and follow up (Andrews 1991). Ethnographic methods need even longer (Sommerville et al. 1993). Some methods also require considerable transcription which can take substantial amounts of time. RAD needs a trained facilitator and minimum of 6 stakeholders (Andrews 1991). At least 2 stakeholders are needed for social interaction which can be observed using ethnographic methods. Furthermore, ethnographic methods should include trained sociologists to work alongside requirements engineers (Sommerville et al. 1992). RAD and ethnographic methods can run the greater risk of alienating stakeholders (DSDM 1995). RAD needs stakeholders to commit precious time to a long meeting or series of meetings. Ethnographic methods often 'intrude' into stakeholders' workplaces for extended lengths of time, which is also a limitation with observation (e.g. Sommerville et al. 1993).

Of course, method selection within managerial, organisational and political constraints is a complex decision, and ACRE does not attempt to provide complete guidance. However, it does define some limitations and highlights issues to be considered.

3.6: Method Interdependencies

The final factor influencing method choice is the other methods used in an acquisition programme. ACRE uses a set of acquisition programme plans which represent typical sequences of methods used in practice. Methods earlier in the sequence fulfil preconditions of subsequent methods. One example of a simple plan is unstructured interviews followed by structured interviews, in which the

unstructured interview identifies entities such as broad problems, requirements and domain features which can be explored in a more systematic manner with structured interviews. Another is laddering followed by card sorts. Again, laddering acquires critical domain entities which provide the contents of a card set for sorting. Other plans are observation or ethnographic methods followed by verbal methods, and interviews acquiring knowledge to generate scenarios. However, the current set of plans is incomplete. It is our aim to embed them into the ACRE decision support system outlined in the next section.

3.7: Practical Guidance

ACRE aims to provide practical guidance for requirements acquisition. Each of the facets can be interpreted as one or more questions to ask about an acquisition session. Answers to these questions inform method choice. A reader wishing to use ACRE should:

1) Determine acquisition needs: use all facets to determine the needs for each acquisition session. The facets provide a useful checklist for planning sessions:

- what is the purpose (bespoke, package selection, procurement, or a combination);
- what knowledge can be acquired from stakeholders and observed in the domain;
- what types of knowledge can be observed;
- what types of knowledge can be provided by stakeholders and what problems are likely when attempting to acquire this knowledge;
- which stakeholders to observe or communicate with;
- what are the practical constraints on each session (time, manpower etc.).

Effective requirements acquisition cannot be planned without first understanding the domain and knowing both stakeholders and high-level system requirements. Therefore, ACRE promotes iterative acquisition. It proposes an initial period of information gathering using the checklist. Several such periods might be needed as requirements and stakeholders change. Future versions of ACRE are anticipated to include more explicit heuristics to guide this information gathering.

2) Choose acquisition method(s): an effective acquisition programme must be defined, however this, in turn, depends on the methods chosen. Again, more specific guidelines are needed:

i) impose constraints defined by the acquisition context to determine the possible range, scope and number of acquisition sessions;
ii) prioritise acquisition sessions according to: (a) the importance of knowledge and requirements to be acquired and: (b) problems might arise during acquisition;
iii) choose methods for planned sessions which acquire the most critical requirements and knowledge. Use the look-up tables in this paper to select methods with the best-fit. It is unlikely that one best fit method will emerge, so compromises and payoffs are inevitable;
iv) choose methods for other sessions;
v) check the acquisition programme against constraints imposed by method interdependencies, and revise where needed.

Choosing acquisition methods is complex, so it can be more effective to derive a first-pass plan without considering the interdependencies between methods. This makes application of ACRE more simple. One of the findings from the ACRE questionnaire was that use of different methods to detect conflicts between requirements owned by different stakeholders can promote considerable acquisition in its own right.

The ACRE decision support system: the steps above provide basic guidance for readers of this paper. However, improved guidance is being planned in the form of an automated decision support system which provides advice for requirements engineers. It is important to emphasise that requirements engineers can redefine or override advice given by the system at any time. ACRE's framework is being operationalised as a set of production rules which are being implemented in an expert system. Such a system is needed for at least three reasons. First, method selection is time-consuming. Allowing the tool to plan questions allows the requirements engineer to design each acquisition session. Second, the framework also provides guidance for planning an acquisition programme. The complex interdependencies between methods demand more complex questioning and

reasoning. Indeed, programme plans must be stored, retrieved and adapted, which necessitates some tooling. Third, selecting the most appropriate methods is difficult. Advice offered by the tool can aid selection. A further option under consideration is extending the ACRE system to incorporate interactive planning facilities for acquisition programmes.

3.8: Framework Validation

The framework has been derived from an extensive review of the literature and the experiences of the two authors. However, empirical validation is needed. Therefore, the first version of ACRE was distributed, along with a questionnaire for feedback, to highly-experienced requirements engineers, most of whom had over 20 year's experience in requirements engineering, and some of whom were also authors of textbooks and methods for requirements engineering. Use of such a questionnaire was both quick and cost-effective for capturing large amounts of data about ACRE. The questionnaire was divided into questions about: (i) the requirements engineers themselves; (ii) current requirements engineering practice; (iii) ACRE's guidelines. The questionnaires are still being gathered, however enough data has already been gathered to provide some feedback and conclusions about ACRE.

Current practice: each requirements engineer was familiar with different methods in ACRE but not one of them claimed to be familiar with more than half of them. All but one of the requirements engineers did not suggest additional methods for inclusion in ACRE, the exception being the inclusion of De Bono's work on thinking and decision-making.

ACRE: all requirements engineers were asked to rate ACRE as essential, very useful, useful, quite useful or not at all useful. All requirements engineers rated ACRE as "essential" in both its present form and as an automated decision support system. One argued that ACRE "is a valuable piece of work because it demystifies requirements gathering methods and hence makes them accessible to those who need them" while another stated that "acquisition is a black art at all stages of the development process and it need not be so opaque if proper guidance and training were available". Specific recommendations have led to changes in the tables in this paper. Earlier versions of ACRE made specific recommendations

about time needed to prepare, use and report results from methods. One requirements engineer claimed that these times were inaccurate due to wide variations in external factors, hence more general guidelines are now provided. Other guidelines were refined, for example Table 1 now shows that methods such as brainstorming can be used in conjunction with, for example, structured analysis models. One requirements engineer uses different methods in parallel to detect possible conflicts which is, he argues, an effective mechanism for further acquisition.

4: Conclusions and Future Work

ACRE, as presented in this paper, serves at least three purposes. First, it identifies critical, but often overlooked factors to consider when selecting acquisition methods. Second, it raises awareness of diverse acquisition methods available, and the need to use more than one method to capture complete requirements. Third, it provides guidelines for method choice based on the critical factors for selection. As such, we hope that ACRE encourages further work on methods for acquisition in the requirements engineering community. However, in this respect, it is important to stress how ACRE has been developed. First, it is not an ad hoc collection of methods. Rather it is a collection of techniques which individually may have theoretical and empirical bases derived from an extensive knowledge of both the literature and current practice. Second, ACRE needs to be evaluated in practice. Between-method evaluation is difficult because of the variation and complexities of requirements engineering tasks. Therefore feedback about ACRE's rules and guidelines can only be qualitative rather than quantitative, as exemplified by the author's use of questionnaires.

ACRE has been developed to fill a gap in methodological guidance for requirements engineers. This view is supported by empirical data gathered so far about ACRE. It contrasts with previous reported research which has tended to push one method. More holistic approaches are needed which recognise the complexities of requirements for software-intensive systems, and hence the range of acquisition methods needed. Most facets in the ACRE framework were met by at least one method, although more methods and tools are needed to capture observable phenomena. Multi-media tools such as AMORE (Wood et al. 1994) are one solution.

Future work is four-fold. First, further questionnaires will be used to gather feedback from experienced requirements engineers. Indeed, this work is now ongoing. The aim here is to elaborate the ACRE framework even further before real-world evaluation takes place. Second, ACRE's set of method plans must be strengthened. Third, interleaving acquisition with other activities such as modelling and validation will be explored through coarse-grain process modelling. A fourth task is to implement the ACRE decision support system to enable more efficient use of the framework. We hope to report all of this work soon.

Acknowledgements

The authors wish to thank their colleagues for ideas and encouragement which have led to ACRE. They also wish to thank everyone who returned comments on ACRE, with special thanks to Geoff Mullery and Suzanne Robertson for their insightful advice. Last but not least our thanks also go to anonymous reviewers of this paper throughout its earlier versions.

References

- ACOSTA R.D., BURNS C.L., RZEPKA W.E. & SIDORAN J. L., 1994, 'A Case Study of Applying Rapid Prototyping Techniques in the Requirements Engineering Environment', Proceedings 1st International Conference on Requirements Engineering, IEEE Computer Society Press, 66-73.
- ANDERSON J.R., 1990, 'The Adaptive Character of Thought', Hillsdale NJ, Erlbaum.
- ANDREWS D.C., 1991, 'JAD: A Crucial Dimension for Rapid Applications Development', Journal of Systems Management, March 1991, 23-31.
- ASHWORTH C. & GOODLAND M., 1990, 'SSADM: A Practical Approach', McGraw-Hill.
- BADDELEY A., 1982, 'Your Memory: A User's Guide', Multimedia Publications 1982.
- BICKERTON M.J. & SIDDIQI J., 1993, 'The Classification of Requirements Engineering Methods', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press 182-185.

- BOAR B.H., 1984, 'Applications Prototyping', Wiley-Interscience.
- BOEHM B., BOSE P., HOROWITZ E. & LEE M-J., 1994, 'Software Requirements as Negotiated Win Conditions', Proceedings of IEEE Conference on Requirements Engineering, IEEE Computer Society Press, 74-83.
- BOOCHER H.R., 1990, 'MANPRINT: An Approach to Systems Integration', New York, Van Norstand Reinhold.
- BRAMEUR M.A., 1990, 'Building Expert Systems', John Wiley.
- BRIGHT B.P., MAIDEN N.A.M. & SUTCLIFFE A.G., 1995, 'Requirements Engineering: Defining the Needs for Collaborative Assistance', Technical Report, Centre for HCI Design, City University.
- CHI M.T.H., GLASER R. & REES E., 1982, 'Expertise in Problem Solving', Advances in the Psychology of Human Intelligence, ed. R. Sternberg, Lawrence Erlbaum Associates, 7-75.
- CLEMENT A. & VAN DEN BESSELAR P., 1993, 'A Retrospective Look at PD Projects', Communications of the ACM 36(4), 29-39.
- CORBRIDGE C., RUGG G., MAJOR N.P., SHADBOLT N.R. & BURTON A.M., 1994, 'Laddering: Technique and Tool Use in Knowledge Acquisition', Knowledge Acquisition 6, 315-341.
- CORDINGSLEY E., 1989, 'Knowledge Elicitation Techniques for Knowledge-Based Systems', in 'Knowledge Elicitation: Principles, Techniques and Applications', D. Diaper (ed), Ellis Horwood, 89-175.
- CRINNION J., 1991, 'Evolutionary Systems Development: a Practical Guide to the Use of Prototyping within a Structured Systems Methodology', Pitman.
- CURTIS B., KRASNER H. & ISCOE N., 1988, 'A Field Study of the Software Design Process for Large Systems', Communications of the ACM 31(11), 1268-1287.
- CUTTS G., 1987, 'SSADM - Structured Systems Analysis and Design Methodology', Paradigm Publishing.
- DIAPER D., 1989, Knowledge Elicitation: Principles, Techniques and Applications', Chichester: E Horwood, New York.
- DSDM, 1995, 'Dynamic Systems Development Method Manual, Version 2'.
- ERICSSON K.A. & SIMON H.A., 1984, 'Protocol Analysis', MIT Press.
- FINKELSTEIN A.C.E., RYAN M. & SPANOUDAKIS G., 1996, 'Software Package Requirements and Procurement', Proceedings 8th International Workshop on System Specification and Design, IEEE Computer Society Press.

- GAMMACK J.G., 1987, 'Different Techniques and Different Aspects of Declarative Knowledge', in A.L. Kidd (ed) 'Knowledge Acquisition for Expert Systems': A Practical Handbook', New York: Plenum Press, 137-163.
- GOFFMAN E., 1959, 'The Presentation of Self in Everyday Life', New York; Doubleday.
- GOGUEN J.A. & LINDE C., 1993, 'Techniques for Requirements Elicitation', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press 152-164.
- GOLDIN L. & BERRY D., 1994, 'AbstFinder, A Prototype Abstraction Finder for Natural Language Text for Use in Requirements Elicitation: Design, Methodology and Evaluation', Proceedings Ist International Conference on Requirements Engineering, IEEE Computer Society Press, 84-93.
- GOTEL O.C.Z. & FINKELSTEIN A.C.W., 1995, 'Contribution Structures', Proceedings 2nd International Symposium on Requirements Engineering, IEEE Computer Society Press, 100-107.
- GOUGH P.A., FODEMSKI F.T., HIGGINS S.A. & RAY S.J., 1995, 'Scenarios - An Industrial Case Study and Hypermedia Enhancements', Proceedings 2nd International Symposium on Requirements Engineering, IEEE Computer Society Press, 10-17.
- GRAHAM I., 1994, 'Object-Oriented Methods', Addison-Wesley.
- GREENSPAN S., BORGIDA A. & MYLOPOULOS J., 1986, 'A Requirements Modeling Language and its Logic', Information Systems 11(1), 9-23.
- GRICE H.P., 1975, 'Logic and Conversation', in Cole, P. & Morgan, J.L. (eds.) Syntax and Semantics 3, New York: Academic Press.
- JACKSON M., 1995, 'Software Requirements and Specifications', ACM Press/Addison-Wesley.
- JACOBSON I., CHRISTERSON M., JONSSON P. & OVERGAARD G., 1992, 'Object-Oriented Software Engineering', Addison-Wesley.
- KAHNEMAN D., SLOVIC P. & TVERSKY A., 1982, 'Judgement under Uncertainty: Heuristics and Biases', Cambridge University Press.
- KELLY G., 1955, 'The Psychology of Personal Constructs', Norton & Company Inc.
- LUFF P., JIROTKA M., HEATH C. & GREATBATCH D., 1993, 'Tasks and Social Interaction: the Relevance of Naturalistic Analyses of Conduct for

Requirements Engineering', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press 187-190.

- MAIDEN N.A.M., & RUGG G., 1994, 'Knowledge Acquisition Techniques for Requirements Engineering', Proceedings Workshop on Requirements Elicitation for System Specification, Keele UK, 12-14 July.
- REGNELL B., KIMBLER K. & WESSLEN A., 1995, 'Improving the Use Case Driven Approach to Requirements Engineering', Proceedings 2nd International Syymposium on Requirements Engineering, IEEE Computer Society Press, 40-47.
- RUGG G., CORBRIDGE C., MAJOR N.P., BURTON A.M. & SHADBOLT N.R., 1992, 'A Comparison of Sorting Techniques in Knowledge Elicitation', Knowledge Acquisition 4(3), 279-291.
- RUGG G. & MCGEORGE P., in press, 'Laddering', to appear in Expert Systems.
- RUMBAUGH J., BLAHA M. & PREMERLANI W., 1991, 'Object-Oriented Modelling and Design', Engleswood Cliffs, NJ: Prentice-Hall.
- SCHRIEBER A. Th., 1994, 'Applying KADS to the Office Assignment Domain', International Journal of Human-Computer Studies 40, 349-377.
- SEGER C.A., 1994, 'Implicit Learning', Psychological Bulletin 115(2) 163-196.
- SHAW M.L. & GAINES B.R., in press, 'Requirements Acquisition', to appear in Software Engineering Journal.
- SOMMERVILLE I., RODDEN T., SAWYER P., BENTLEY R. & TWIDALE M., 1992, 'Sociologists can be Surprisingly Useful in Interactive Systems Design', Proceedings People & Computers VII (HCI'92), eds A. Monk, D. Diaper & M.D. Harrison, Cambridge University Press.
- SOMMERVILLE I., RODDEN T., SAWYER P., BENTLEY R. & TWIDALE M., 1993, 'Integrating Ethnography into the Requirements Engineering Process', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press 165-173.
- TAKEDA N., SHIOMI A., KAWAI K. & OHIMA H., 1993, 'Requirement Analysis by the KJ Editor', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press 98-101.
- TULVING E. & OSLER S., 1968, 'Effectiveness of Retrieval Cues in Memory For Words', Journal of Experimental Psychology 77, 593-601.
- TURBAN E., 1993, 'Decision Support and Expert Systems', MacMillan NY.

- WICKER A.W., 1969, 'Attitude Versus Actions: The Relationship of Verbal and Overt Behavioral Responses to Attitude Objects', Journal of Social Issues 25(4), 41-78
- WOOD D.P., CHRISTEL M.G., STEVENS S.M. 1994, 'A Multimedia Approach To Requirements Capture And Modelling', Proceedings of 1st International Conference On Requirements Engineering, IEEE Computer Society Press, 53-56.
- YU E.S.K., 1993, 'Modelling Organisations for Information Systems Requirements Engineering', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press, 34-41.

FIGURES WITH CAPTIONS

Observation
Description: the requirements engineer observes actual practices in the domain.
Preconditions: few, although gaining access to work places can be problematic.
Strengths: simple to do and does not require much training and preparation.
Weaknesses: can capture large amounts of irrelevant data.

Unstructured Interviews (Cordingsley 1989, Turban 1993)
Description: requirements engineer asks stakeholder about the topic in question without a prepared list of questions.
Preconditions: none
Strengths: simple to do. It is good for eliciting stakeholder's agenda of what is relevant. Different variations are available.
Weaknesses: easy to spend too much time on side-issues. The captured information may be hard to analyse.

Structured Interviews (Cordingsley 1989, Turban 1993)
Description: requirements engineer asks stakeholder list of prepared questions.
Preconditions: relevant questions have to be identified first.
Strengths: systematic, constant across stakeholders. Again, different variations are available (Cordingsley 1989).
Weaknesses: impose a framework on users which may miss important issues.

Protocol Analysis (Ericsson & Simon 1984)
Description: someone undertakes a task and speaks out loud during it.
Preconditions: suitable task needs to be identified.
Strengths: provides access to working memory processes. It also provides large amounts of data about specific tasks.
Weaknesses: protocols may be difficult to understand, because of their "chain of consciousness" nature. Analysing protocol transcripts is tedious and time-consuming.

Figure 1(a) - some of the methods included in the ACRE framework.

Card Sorting (Gammack 1987, Rugg et al. 1992)
Description: stakeholder is asked to sort into groups a set of cards each of which has the name of some domain entity written or depicted on it; user then says what the criterion was for the sorting, and what the groups were.
Preconditions: suitable entities need to be identified, with suitable semantic spread across domain.
Strengths: knowledge is represented in standardised format. Amenable to automation. No need for transcripts.
Weaknesses: knowledge elicited is "flat" object-attribute-value knowledge. No knowledge of classes, structures, procedures.

Laddering (Rugg & McGeorge 1992, Corbridge et al. 1992))
Description: requirements engineer uses small set of probes to acquire structure and content of stakeholder's knowledge.
Preconditions: deals with hierarchical knowledge, including polyhierarchies (e.g. goal trees, is-a taxonomies).
Strengths: knowledge is represented in standardised format. Can elicit structural knowledge. Suitable for automation.
Weaknesses: Assumes hierarchically-arranged knowledge.

Repertory Grids (Shaw & Gaines, in press)
Description: stakeholder is asked for attributes applicable to a set of entities and values for cells in entity:attribute matrix.
Preconditions: suitable entities need to be identified, with suitable semantic spread across domain.
Strengths: knowledge is represented in standardised format. Amenable to automation. Supports statistical analysis.
Weaknesses: knowledge elicited is "flat" object-attribute-value knowledge, with no knowledge of classes, structures, procedures, etc. Problems representing some types of attribute (nominal values, ranges of values, etc).

Brainstorming (Takeda et al. 1993)
Description: requirements engineer asks group of stakeholders to generate as many ideas as possible, with emphasis on generation rather than on evaluation.
Preconditions: suitable group of stakeholders.
Strengths: good for eliciting high-level domain entities and questioning assumptions which might otherwise have constrained approaches considered. Tool support available for some varieties.
Weaknesses: susceptible to group processes; unsystematic in "classic" form, though some varieties overcome this.

Figure 1(b) - some more methods included in the ACRE framework.

Rapid Prototyping (Boar 1984, Crinnion 1991)
Description: stakeholder is asked to comment on a prototype physical working model of the desired system.
Preconditions: a prototyping environment is needed. The prototype must then be generated.
Strengths: good for catching taken-for-granted issues, anomalous state of knowledge issues, etc..
Weaknesses: take time and effort to build. Evolutionary prototyping can have inherent dangers for system design if misused.

Scenario Analysis (Jacobson et al. 1992, Gough et al. 1995)
Description: a scenario is a description of a sequence of actions and events for a specific case of some generic task which the system is intended to accomplish. Scenarios include use-cases (Jacobsen 1992, Rumbaugh 1994).
Preconditions: a set of relevant scenarios must be generated. This can be difficult and time-consuming.
Strengths: integration with structured methods such as OOSE (Jacobsen 1992). Positive feedback in trials (Gough et al. 1995).
Weaknesses: Gough et al. (1995) reported that the effort expended during generation was a demotivator towards end of use.

Rapid Application Development (RAD) Workshops (Andrews 1991)
Description: focus is a workshop in which 8-20 individuals make decisions through the concensus building leadership of a trained, unbiased facilitator who is not a stakeholder in the desired system. Offered in different formats (e.g. IE method).
Preconditions: workshops require up to three weeks of planning. Important post-workshop activities also exist.
Strengths: improves quality and speed of system design. Integrated with current structured methods and CASE tools.
Weaknesses: workshops tie up stakeholders often for several days at a time. Only effective for smaller systems.

Ethnographic Methods (Luff et al. 1993, Sommerville et al. 1993, Sommerville et al. 1992)
Description: requirements engineer spends extended periods of time observing users in their normal workplace.
Preconditions: requirements engineers need considerable training in the use of ethnographic methods.
Strengths: good for identifying taken-for-granted issues and "back" versions of domain.
Weaknesses: require considerable time to undertake. Liable to attribution errors, etc. May fail to detect infrequent events. Possible ethical problems in reporting "back" information given confidentially.

Figure 1(c) - some more methods included in the ACRE framework.

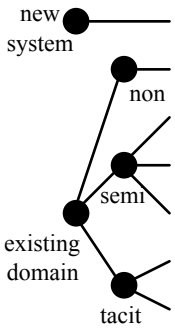
Requirements Purposes	observation	u. interviews	s. interviews	protocols	card sorting	laddering	rep. grid analysis	brainstorming	rapid prototyping	scenario analysis	RAD	ethnographic
Packages known	x	-	--	x	==	--	==	-	==	-	x	x
Packages unknown	x	-	--	x	--	==	--	-	-	-	x	x
Requirements procurement	-	==	==	-	-	--	-	--	-	==	==	-
Method output	NL CD SA	NL	NL	NL	OV	OV	SM	OV SA	NL	NL	NL SA	NL CD SA

Table 1 - fit between requirements purpose and method. √√ indicates very good fit, √ indicates good fit, - indicates weak fit and x indicates poor fit. The same scheme is used in all other tables. The table also shows the standard form of output from using each method: NL is natural language, CD is coded data, OV is object-

attribute-value triplets, SM is set/hierarchy maps, and SA is structured analysis notations.

Knowledge type	observation	u. interviews	s. interviews	protocols	card sorting	laddering	rep. grid analysis	brainstorming	rapid prototyping	scenario analysis	RAD	ethnographic
Behaviour	==	--	--	--	-	--	--	--	--	==	==	==
Process	--	--	--	--	-	--	--	--	--	==	==	==
Data	-	-	-	-	==	==	--	-	--	--	==	--

Table 2 - the effectiveness of methods for acquiring different types of knowledge.



Internal representation of knowledge	observation	u. interviews	s. interviews	protocols	card sorting	laddering	rep. grid analysis	brainstorming	rapid prototyping	scenario analysis	RAD	ethnographic
Future system knowledge	x	--	--	x	-	--	-	--	==	==	==	x
Non-tacit knowledge	-	==	==	==	==	==	==	--	-	==	==	-
Recognised knowledge	x	x	x	-	--	-	-	--	==	==	==	x
TFG knowledge	==	-	-	--	-	-	-	-	--	--	--	==
Working memory knowledge	x	x	x	==	x	x	x	x	x	x	x	x
Compiled knowledge	==	-	-	==	-	-	-	-	--	--	--	==
Implicit knowledge	==	-	-	==	-	-	-	-	--	--	--	==

Table 3 - the effectiveness of methods for acquiring knowledge with different internal representations.

	observation	u. interviews	s. interviews	protocols	card sorting	laddering	rep. grid analysis	brainstorming	rapid prototyping	scenario analysis	RAD	ethnographic
Observable phenomena	==	x	x	-	x	x	x	x	x	x	-	==

Table 4 - the effectiveness of methods for acquiring observable phenomena.

Constraints	observation	u. interviews	s. interviews	protocols	card sorting	laddering	rep. grid analysis	brainstorming	rapid prototyping	scenario analysis	RAD	ethnographic
Meeting is needed	x	--	--	--	--	--	--	--	--	--	--	x
Time to prepare session	==	==	-	--	--	==	-	==	-	-	--	==
Time for acquisition session	--	==	==	==	==	==	==	==	==	==	x	x
Time to obtain requirements	-	--	--	-	--	--	--	--	==	--	x	x
Number of requirements engineers	1	1	1	1	1	1	1	1	1	1	1	1
Number of stakeholders	1	1	1	1	1	1	1	1	1	1	6	2
Friendliness to stakeholders	-	==	--	-	==	==	==	--	--	--	x	x
No technological overheads	--	==	==	-	==	==	==	==	x	==	x	--

Table 5 - minimum conditions for method use. Manpower constraints indicate the minimum number of people needed to use the method.