Requirements Process Improvement through the Phased Introduction of Good Practice

Pete Sawyer, Ian Sommerville and Stephen Viller

Cooperative Systems Engineering Group

Technical Report Ref: CSEG/17/1997

http://www.comp.lancs.ac.uk/computing/research/cseg/97_rep.html

Published in "Software Process Improvement and Practice", Vol 3, Issue 1, pp. 19-34, March 1997.

ABSTRACT

Current process improvement and maturity models pay little attention to requirements engineering. Typically, requirements engineering is considered to be a single activity in the overall development process. Even where this is not strictly the case, the requirements activities are not elaborated in sufficient detail to permit the derivation of an improvement plan. This is unfortunate because requirements engineering is increasingly recognised as a problem. Despite the regular improvement of techniques for eliciting, analysing, validating and managing requirements, even otherwise mature organisations repeatedly experience requirements problems. This paper describes a good practice-based approach to requirements engineering process improvement which aims to fill the gap left by existing process improvement methods. This distils practical information from over 60 requirements practices and provides a framework to help organisations identify their problem areas and deploy the practices most appropriate to their needs.

CSEG, Computing Department, Lancaster University, Lancaster, LA1 4YR, UK

Tel: +44-1524-65201 Ext 93799; Fax: +44-1524-593608; E-Mail: julie@comp.lancs.ac.uk

http://www.comp.lancs.ac.uk/computing/research/cseg/

Requirements Process Improvement Through the Phased Introduction of Good Practice

Pete Sawyer, Ian Sommerville and Stephen Viller
Lancaster University

Summary

Current process improvement and maturity models pay little attention to requirements engineering. Typically, requirements engineering is considered to be a single activity in the overall development process. Even where this is not strictly the case, the requirements activities are not elaborated in sufficient detail to permit the derivation of an improvement plan. This is unfortunate because requirements engineering is increasingly recognised as a problem. Despite the regular improvement of techniques for eliciting, analysing, validating and managing requirements, even otherwise mature organisations repeatedly experience requirements problems. This paper describes a good practice-based approach to requirements engineering process improvement which aims to fill the gap left by existing process improvement methods. This distils practical information from over 60 requirements practices and provides a framework to help organisations identify their problem areas and deploy the practices most appropriate to their needs.

Keywords

Requirements engineering, Process improvement, Maturity model

1. Introduction

A recent European survey [ESPITI 96] showed that the organisations consulted consider the principal problem areas in software development to be the requirements specification and the management of customer requirements. There is no doubt that this is equally true in other parts of the world. Improving the processes of discovering, documenting and managing customer requirements is critical for future business success.

Common requirements problems which arise are:

- The requirements don’t reflect the real needs of the customer.
- Requirements are inconsistent and/or incomplete.
- It is expensive to make changes to requirements after they have been agreed.
- There are misunderstandings between customers, those analysing and documenting the requirements, and software engineers developing or maintaining the system.

Because of the persistence of these problems, requirements engineering attracts much interest and investment. However, we and many others have observed a mismatch between the problems experienced by industry and the techniques developed from research in requirements engineering. It is not that the results of the research community are inappropriate, but we believe that their effective deployment is contingent on a number of basic underpinning factors which are often overlooked.

As an example, Davis [Davis 95], in a recent short article, identified requirements tracing as a problem which, although considered "solved" by the research community, still posed practitioners with hard practical problems. Ramesh et. al. [Ramesh 95] further show that although requirements tracing is often mandated by the standards to which developers work, there is seldom a good definition of what is to be traced. Consequently, tracing is sometimes seen simply as a process requirement to be satisfied for accreditation purposes. Engineers, subject to time and cost pressures, and with no clear idea of what the expected benefits are, have little incentive to commit resources to achieving a vague goal.

The work described in this paper was motivated by our conviction that the most effective way to help industrial practitioners was to help them identify how they could make best use of existing good practices. In order to do this it is important to understand their requirements processes since these provide the context within which new practices must be used. Practitioners need to be able to assess their current processes in terms of the requirements problems which they experience and then select practices appropriate to the improvements which they wish to make.

Software Process Improvement (SPI) already offers the means to do this in the later stages of the software development process. In recent years, the SEI's Capability Maturity Model (CMM) for Software [Humphrey 88, Paulk 93] and the ISO 9000 [Johnson 93] quality standard(s) have been enthusiastically embraced throughout the industry. There is evidence that this has led to real improvements in both software product quality and organisations' profitability [Dion 93, Humphrey 91, Herbsleb 97]. However, the experience of our industrial partners in the REAIMS[†] project was that, in their current forms, neither the CMM nor ISO 9000 address requirements processes adequately.

This is a serious shortcoming because :

- The success of any development project is intimately related to the quality of the requirements.
- The requirements process is much less homogeneous and well understood than the software development process as a whole.

Although "life-cycle" standards exist (e.g. ESA Software Engineering Standard PSS-05 [ESA 91, Mazza 94] and the ISO Software Life-Cycle Process standard ISO-12207 [ISO 95, Singh 96]) which provide basic guidance on requirements processes, they provide little help for the identification of problems and the planning of improvements. There is a clear need to extend the principles of SPI to the requirements process:

- To define what the requirements engineering process is
- To describe good practices in requirements engineering
- To provide guidelines for the adoption of these practices based on quantitative and qualitative assessments of the state of organisations' current processes and the likely benefits to be gained from adopting new practices.

This paper is about incremental requirements process improvement through the phased adoption of established good practices. It proposes a process maturity and improvement framework which is designed to help organisations to assess their requirements process and to plan and implement improvements. This framework forms part of the *Requirements*

[†] The Esprit REAIMS (Requirements Engineering Adaptation and IMprovement for Safety and dependability) project has developed a number of technologies for requirements engineering process improvement. These have focussed on the critical systems domain(s) but many of the project results are generically applicable. This is particularly true of the Requirements Engineering Good Practice Guide.

Engineering Good Practice Guide (GPG) [Sommerville 97], developed as part of REAIMS. The GPG is structured as a set of requirements practices which have been classified according to the requirements engineering activities which they address and the level of process maturity which an organisation needs to have attained in order to be able to deploy them effectively.

The rest of this paper is organised as follows: Section 2 reviews how requirements engineering is addressed by current standards and process improvement models. Section 3 discusses the characteristics of the requirements engineering process. Section 4 describes the GPG process improvement model and section 5 concludes the paper.

2. Existing Standards

There are two complementary classes of standard which are relevant to requirements engineering process improvement:

- SPI methods. These are aimed at evaluating organisations' software processes and providing guidance on what can be improved and how. We include the ISO 9000 family of quality standard(s) under this heading. Although not strictly an improvement standard, ISO 9000 has much in common with those parts of improvement models which deal with the attainment of basic levels of process maturity [Paulk 94].

- Life-cycle process standards which are concerned with providing reference models of the software development process. These deal with process models, the activities which comprise a process and their products.

These are discussed in the following sections.

2.1. Process Improvement Standards and Models

Current SPI methods broadly treat requirements engineering as a single activity in the overall development process. Much can be inferred from what they say, but it is seldom made explicit.

ISO 9000 is not a single standard but a series of quality system standards. Those relevant here are ISO 9001 which is generic to designed products, and ISO 9000-3 which interprets 9001 for software. There is no section specific to requirements engineering in either of these. References to requirements occur throughout the standard but occur most frequently under Contract Review and Design Control. These set standards for the contractor's responsibilities for project planning and costing based on the requirements definition, and on the format and properties of a requirements specification document. Little is said about the activities involved in eliciting, analysing and validating the requirements. Ince [Ince 94] shows that what is specified in ISO 9000 can be interpreted to imply a number of more specific requirements management activities. However, the standard offers little direct help to an organisation committed to serious quality improvements in their requirements process.

The SEI's CMM for Software version 1.1 defines standards to be attained by different process areas such as project management, quality management, etc. for the different phases of the development life-cycle. There are 5 maturity levels from level 1 (initial) which is an ad hoc process to level 5 (optimising) where a process is subject to continual, systematic improvement using feedback from projects. The CMM has a *staged* architecture where each process area is uniquely associated with one of the 5 maturity levels. Hence, for an organisation to mature from level n to $n+1$ requires that all the process areas in level n and $n+1$ have been addressed organisation-wide. This reflects the CMM's objective of characterising organisational process maturity. Although it is not without its critics [Bollinger 91], sufficient experience has now been acquired of the CMM's application to software development processes to suggest that it is both workable and economically beneficial [Herbsleb 97].

However, the experience of the industrial organisations with whom we have collaborated is that while this is true for the software development process, the benefits are harder to gain when applied to the requirements process. One problem which they cite is the CMM's vagueness about the composition of the requirements process and what is to be expected of the requirements process attributes. The one aspect of the requirements process treated in detail is requirements management which is identified as a key process area for level 2 (repeatable) processes. A number of practices are given to support this. However, as we show later, requirements management is only one area of the requirements process - albeit a key one.

The planned version 2.0 of the CMM [Paulk 96] is expected to have more to say on requirements engineering. Among its influences is the SPICE project [Konrad 95] which aims to develop an international software process management standard. SPICE is still under definition but its basic structure is becoming clear. One of its central planks is the baseline practices guide (BPG) which defines the development process activities to be undertaken. The BPG is organised into 5 process areas: Customer-supplier, Engineering, Project, Support and Organisation. Of these, the first three include requirements-related activities. In contrast to the CMM, SPICE has a *continuous* architecture where there is a less rigid correspondence between practices, processes and maturity levels. This is intended to encourage a more flexible approach where improvement effort may be focused on the most needy process areas.

Perhaps the major characteristic of all the above standards and models is that they are oriented towards assessing organisations' competence against a set of process criteria using forms, checklists and templates. Inevitably, process assessment is difficult and is based on a snapshot view of the organisation. Because of this, much attention has been paid to devising training regimes and assessment materials with the aim of making the assessment quick but informative. A team of accredited assessors will visit an organisation, collect evidence about their development process over a few days and deliver a rating at the end. In the CMM's case, for example, this rating will be a numerical value corresponding to which of the 5 maturity levels the organisation is judged to have attained. The attainment of such accreditation is becoming increasingly important for software contractors, particularly for government work. In addition, organisations are increasingly keen on performing internal assessments, both as a preparation for official accreditation visits but also simply to help identify their process weaknesses. This reflects growing recognition of the potential for cost savings from process improvement and parallels the enthusiasm for business process reengineering (BPR) and total quality management (TQM) in other business areas.

2.2. Life-Cycle Standards

Standards in this category are designed to provide a reference model for the software life-cycle process(es). They originate from recognition of the wide disparity between organisations and the consequent need to converge on agreed practices between client and contractor(s). This is particularly true where, for example, several subcontractors are involved on large projects. Clearly, there is a need here for a common view of the development process, its activities, their inputs and their products. Unsurprisingly, many life-cycle standards originate from large government organisations. Examples of these are the US DoD's MIL-STD-498 [DoD 94] and the ESA Software Engineering Standard PSS-05 [ESA 91, Mazza 94]. More recently, Software Life-Cycle Process standard ISO-1207 has been proposed as an international standard [ISO 95, Singh 96]. There also exist a number of specialised standards such as the UK MoD's DEF STAN 00-55 [MoD 97] for safety-critical defence software.

Practices covered by these standards include those specific to individual phases of the life-cycle and those which are globally applicable. As an example, PSS-05 characterises these as:

- Product standards. These define the individual phases of process, the activities performed during each phase, their outputs and the milestones to be achieved, and the roles and responsibilities of those involved in each phase.

- Process standards. These define the management procedures which are orthogonal to the process phases and activities. These include configuration management, quality assurance, validation and verification, etc.

All the above standards address requirements engineering to some extent. PSS-05, for example, identifies 6 life-cycle phases of which the first two are: User Requirements Definition (UR) and Software Requirements Definition (SR). The 3rd phase, Architectural Design (AD), also includes activities which are sometimes conducted as part of the analysis of requirements.

In PSS-05, each life-cycle phase is described in terms of specific practices and guidelines for the conduct of the phase. These typically include a short rationale and a brief implementation guide. The standard also offers help on factors such as the cost of applying the practices.

As an example of what is defined for each phase, the UR phase lists 4 activities to be performed (capture of user requirements, determination of operational environment, specification of user requirements, reviews), and lists 6 outputs form the phase (user requirements document, acceptance test plans, project management plan for SR phase, configuration management plan for SR phase, validation and verification plan for SR phase, QA plan for SR phase).

PSS-05 reflects the ESA's application domain; that of systems engineering for space applications. Nevertheless, the standard has been widely used in other application domains, including those of REAIMS's industrial partners.

Inevitably, life-cycle process standards are either designed to be generic (e.g. ISO-1207) or, if applied outside of the author organisation, must be interpreted. They apply across the development life-cycle so requirements processes are not the focus. Similarly, requirements processes are frequently domain, organisation and even customer-specific. Nevertheless, life-cycle standards contain much wisdom and have served as a valuable first reference for organisations interested in establishing good requirements practice.

2.3. Relevance to the Requirements Engineering Good Practice Guide

Although no current SPI methods adequately address requirements issues, there are clear advantages for harmonising requirements engineering process improvement with them. This is particularly true since a clean cut-off between requirements engineering and subsequent development activities never exists. Clearly, therefore, the standards, conventions and techniques used to manage the requirements and development process activities need to be compatible. For example, compatible mechanisms for configuration management should be used for both code and requirements changes. Similarly, the philosophy behind any requirements process improvement model should dovetail with that of the CMM or SPICE.

The REAIMS project has taken the view that guidance on requirements engineering process improvement is timely, and that such guidance can be made compatible with both the ISO 9000 and the CMM as they currently stand. In doing so, it is anticipated that the guidance will also be forwardly compatible with subsequent versions of the CMM and with SPICE.

We have also taken the view that the importance of requirements engineering demands that it be recognised as a complex process in its own right and not simply as a phase of the software life-cycle. Existing life-cycle standards are a valuable source of basic good practices but the GPG aims to provide more focused coverage of how they can be integrated in a requirements process, what their benefits and costs will be and what problems may be encountered. We also show how other practices, for example, those with specialist applications, can be adopted in a way which allows an organisation to plan and evaluate improvements to its requirements process. We have therefore cast our net for good requirements practices wider than existing standards, although these are used to form the foundations of a defined requirements process.

The GPG is therefore a hybrid life-cycle and process improvement guide (it has no pretensions to being a standard). It contains both descriptions of the requirements process and associated activities, practices, etc. and guidelines for assessing and improving an organisation's implementation of the requirements process. These are summarised in the remainder of this paper.

3. The Requirements Process

The objective of the requirements process is to deliver a requirements specification document which defines the system to be developed. However, the process must recognise that the quality of requirements information never attains perfection. The point at which the requirements process terminates is therefore a matter of judgement about when and whether the collected requirements are a sufficiently sound basis for development to commence. In many cases, this point is never reached but development has to commence anyway. Requirements activities continue to be enacted concurrently with down-stream activities as customer requirements change, as design options have to be traced, and as errors and omissions in the requirements specification emerge.

For these reasons, models of the requirements process which reflect its iterative nature and lack of easily defined termination condition have become popular. To work, these must define activities aimed at identifying and resolving requirements defects while coping with those which inevitably emerge at later stages.

For example, Boehm's [Boehm 94] model is based on his spiral model of software development [Boehm 88], augmented to include provision for establishing stakeholders' "win" conditions. This means the provision of steps to facilitate identification and negotiation of requirements trade-offs. These are necessary to ensure that all relevant factors (technical, economic and political) exert the appropriate influence on resolving stakeholders’ conflicting requirements. Hence, these requirements conflicts are more likely be resolved in a way which is satisfactory to each stakeholder - i.e. everyone wins. Potts at al. [Potts 94] have also proposed a cyclical model, called the Inquiry Cycle. This consists of three iteratively repeated activities; expression, discussion and commitment.

Figure 1 illustrates our generic requirements process model which has been strongly influenced by Boehm and Potts' models. Because it is used to underpin our good practice-based process improvement model, however, it has been made somewhat simpler than Boehm's model and adopts more conventional terminology than Pott's model.

It is a spiral in that requirements information emerges from successive iterations, and does so in the context of the requirements information which emerged from previous iterations. Hence, for example, requirements information which emerges in the first iteration may constrain requirements which emerge in later iterations. Conversely, it may also need to be modified in the light of information which emerges later.

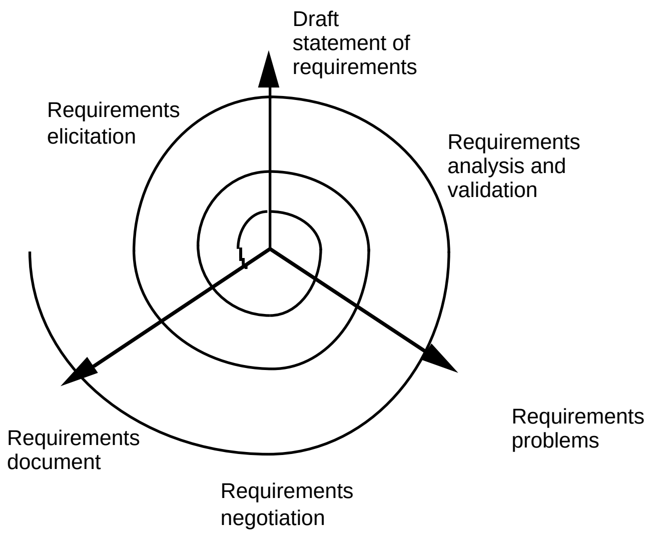

Figure 1 A generic requirements process model

In our model, three activities are repeated each iteration of the spiral. The radial arms of the spiral represent both increasing cost and the generation of information by all three phases. The more iterations of the spiral, the better the quality of the requirements information should be. However, more resources are consumed in doing so.

The three phases of each cycle are:

1. *Requirements elicitation* Given a statement of organisational needs and other inputs, various different sources are consulted to understand the problem and the application domain and to establish their requirements. These requirements may not be complete and may be expressed in a vague and unstructured way. Requirements sources include (but are not restricted to) all stakeholders including customers, domain experts and third parties such as regulatory authorities.

2. *Requirements analysis and validation* The requirements discovered during the elicitation phase are integrated and analysed. Usually, this will result in the identification of missing requirements, inconsistencies and requirements conflicts.

3. *Requirements negotiation* Inconsistencies, conflicts and uncertainties discovered during analysis need to be resolved. The analysts and stakeholders clarify their understanding and consider the problematic "win" conditions. This typically requires negotiation to establish the necessary trade-offs. These trade-offs and the misunderstandings which are revealed may necessitate the elicitation of further requirement information.

An important activity is not explicitly represented in the figure: *requirements management*. Requirements management permeates the whole process and is concerned with coping with the incremental emergence of requirements information and with the inevitable changes to which requirements are subject. This is one of the few requirements activities expressly addressed by the CMM. It is listed as a key process area for level 2 (repeatable) development processes. Clearly, requirements management must be harmonised with subsequent life-cycle practices. Requirements must be traceable forwards into design and backwards to their sources and change control must be applied across all products of the life-cycle.

As implied above, the requirements process as depicted in Figure 1 does not exist in isolation from subsequent software development activities. The requirements spiral will continue to iterate as, for example, requirements prove unfeasible and have to be modified, as the customer requests new features and as cost problems cause priorities to be adjusted. Hence, the three segments of the spiral now represent the emergence of changes and proposed changes; the assessment of the proposed changes to assess their impact; and the formulation and negotiation of other changes consequent on those already proposed.

Although it is possible to build neat models of the requirements process, such as that above, relatively few development organisations have a defined requirements process. Moreover, even relatively "mature" organisations experience problems with their requirements [Hutchings 95]. This cannot be entirely blamed upon the absence of clear guidelines from existing improvement models and quality standards. The reasons are complex but include:

- The volatility of customer requirements. This places a process based on closely coupled activities under considerable strain because the effects of changes are hard to localise. This is particularly true of systems engineering projects where the brunt of changing requirements often has to be borne by the software.

- Most requirements methods support the elicitation and analysis of functional requirements. Consequently, constraints arising from (for example) systems' political and organisational environments are hard to discover and document. Moreover, these are subject to considerable variation, even within the same application domains. For example, a railway signalling system in the UK will have different characteristics from one in France due to the railway systems' different funding and operating procedures.

- Domain expertise may be difficult to acquire. Crucial constraints may be implicit in the domain as "domain phenomena" [Jackson 93]. However, if their significance is opaque to the requirements engineer and if no stakeholder articulates them (perhaps because they assume that they are obvious), they can easily be overlooked. For example, a requirement for error correction may be overlooked if no-one explicitly identifies the presence of electromagnetic radiation in the operational environment.

Clearly, factors such as these make it hard for an organisation to define a sequence of activities guaranteed to result in complete, consistent and validated requirements specification documents for every project. As shown below, established good practices do exist which address these problems but the absence of requirements-specific standards has made it hard for organisations to assess their utility. Life-cycle standards such as ESA PSS-05 are too general to address the requirements process in detail. The aim of the GPG is to fill this gap by providing requirements-specific guidance. Our approach is to categorise good practices according to the aspects they address, the preconditions required to successfully exploit them and the complementary activities from which additional leverage can be gained.

4. Requirements Engineering Process Maturity

Requirements engineering process maturity can be defined as the extent to which an organisation has a requirements engineering process based on good requirements engineering practices. An organisation with a mature requirements engineering process will have this process explicitly defined. It will use appropriate methods and techniques for requirements engineering, will have defined standards for requirements documents, requirements descriptions, etc. The organisation may use automated tools to support process activities. It will have management policies and procedures in place to ensure that the process is followed and may use process measurements to collect information about the process to help assess the value of process changes.

Of course, process maturity level is only one of the factors which affect the quality of the final requirements document. Other important factors are the ability and experience of the people involved in the process, the novelty, difficulty and size of the problem and the time and resources available. Immature organisations can and do produce good quality requirements documents. However, they may not be able to do so consistently or when working to tight deadlines.

Mature organisations, by contrast, should normally produce good quality documents on time and within budget. It doesn’t mean that they won’t ever have requirements engineering problems. Novel systems, particularly, are always likely to be difficult to specify. However, a mature organisation should have processes and procedures in place which give them the best chance of solving unforeseen requirements problems.

4.1. Good Practice Guidelines

The following section (4.2) describes the maturity model used by the GPG to classify organisations' requirements process maturity. The model is based upon an organisation's use of good requirements practices. The GPG describes 66 good requirements practices covering all areas of requirements engineering. These have been derived from existing standards, studies of requirements processes (e.g. [Forsgren 95, El Eman 95a]) and from the practical experience of REAIMS partners and others. The practices vary from basic practices which every organisation should adopt to specialised practices which will only benefit organisations working in demanding application domains.

The practices vary widely in what they are intended to achieve and how they are implemented. Some practices are concerned with simple one-off procedures (e.g. defining a standard document structure), others with the use of techniques, methods or CASE tools (e.g. using scenarios to elicit requirements). Some practices are directly concerned with process change (e.g. the use of multi-disciplinary teams to review requirements).

Potentially, selection of the appropriate practices to use will pose an organisation with a difficult choice. To help manage this problem, the practices are structured as guidelines which offer help on their selection and use. Each guideline describes the key benefit(s) and gives a qualitative assessment of the cost of introducing and applying the practice. Appendix A contains an example guideline from the section on requirements validation.

The guidelines are organised according to the process deliverables and activities to which they apply. These are:

- *The requirements document* How to structure and organise the requirements document in order to effectively communicate requirements to customers, managers and developers.
- *Requirements elicitation* Practices to help discover the requirements from system stakeholders as well as from the application domain, and the system's operational and organisational environments.
- *Requirements analysis and negotiation* Practices to help identify and resolve problems (such as incompatibilities or missing information) with the elicited requirements.
- *Describing requirements* Guidelines for writing requirements so as to maximise readers' understanding.
- *System modelling* Guidelines for the development of the abstract system models which are necessary for the understanding and analysis of the requirements and their implications for the proposed system.

- *Requirements validation* Practices to help establish formal validation procedures concerned with checking for problems such as incompleteness, inconsistency or incompatibility. They are also designed to help ensure that requirements are verifiable and that quality standards are adhered to.
- *Requirements management* Guidelines for aiding the management of requirements information throughout the development life-cycle.

In addition to the deliverables and activities which they address, the practices are classified as basic, intermediate and advanced guidelines as described in Table 1.

Table 1 Guideline classification

Guideline classification	**Description**	**Costs**
Basic	Relatively simple practices concerned principally with standardisation, management and usability issues. They provide the foundation for a repeatable requirements engineering process where the cost, time and resources needed for requirements engineering can be estimated.	Basic practices are usually relatively cheap to introduce and use.
Intermediate	More complex practices which lead to a defined requirements engineering process. These are mostly concerned with the introduction of systematic and structured methods into the RE process.	These usually cost more and take more time to introduce than basic practices.
Advanced	These are intended to support the continuous improvement of an RE process. They include advanced methods which require specialist expertise to apply and guidelines for organisational change.	Costs vary. Some advanced practices, based on advanced technology, are expensive. Others are relatively cheap but may require organisational change.

The GPG currently lists 36 Basic practices, 21 Intermediate practices and 9 advanced practices. The list of good practices is, of course, not exhaustive and we expect the list to expand slowly as the discipline develops. We would not, however, expect the relative proportion of practices in each classification to vary greatly.

4.2. The REAIMS Process Maturity Model

The REAIMS requirements process maturity model is a three-level model. The first two levels are roughly comparable to the first two levels of the CMM. Beyond level 2, it becomes hard to distinguish requirements maturity levels comparable to the CMM *defined*, *managed* and *optimising* levels. Such is the relative immaturity of requirements engineering process analysis that we are doubtful whether many organisations' requirements engineering process could be truly categorised beyond *defined* in the CMM sense. For this reason, the GPG model is not developed beyond level 3. This is illustrated in Figure 2.

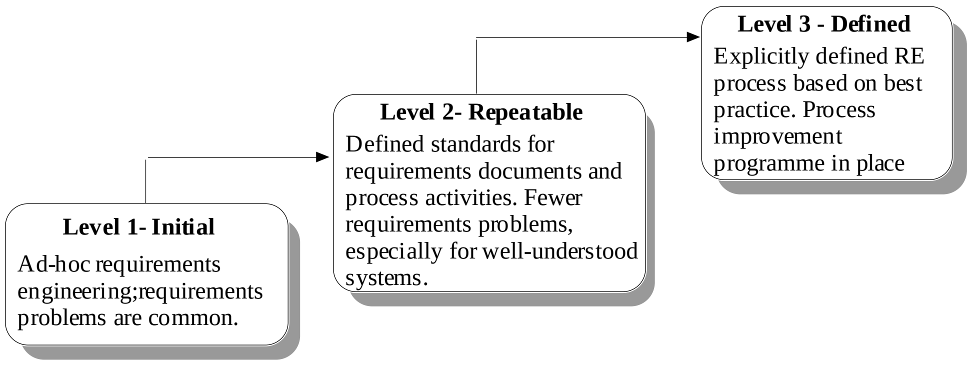
Level 3 - Defined
Explicitly defined RE process based on best practice. Process improvement programme in place
Level 2- Repeatable
Defined standards for requirements documents and process activities. Fewer requirements problems, especially for well-understood systems.
Level 1- Initial
Ad-hoc requirements engineering;requirements problems are common.

Figure 2 Requirements engineering process maturity levels

1. *Level 1 - Initial level* Level 1 organisations do not have a defined requirements engineering process and often suffer from the problems discussed above. They do not use defined methods to support their requirements engineering processes. They often fail to produce good quality requirement documents on time and within budget. They are dependent on the skills and experience of individual engineers for requirements elicitation, analysis and validation.
2. *Level 2 - Repeatable level* Level 2 organisations have defined standards for requirements documents and requirements descriptions and have introduced policies and procedures for requirements management. They may use some advanced tools and techniques in their requirements engineering processes. Their requirements documents are more likely to be of a consistent high quality and to be produced on schedule.
3. *Level 3 - Defined level* Level 3 organisations have a defined requirements engineering process model based on good practices and techniques. They have an active process improvement programme in place and can make objective assessments of the value of new methods and techniques.

These are rough classifications which are principally used as a framework for the requirements practices. In general, level 1 organisations should focus on introducing the basic practices; level 2 organisations should have implemented most of the basic practices and be in a position to implement the intermediate guidelines; level 3 organisations should have implemented almost all basic practices and all appropriate intermediate practices. They may improve their process by introducing the advanced practices.

Some organisations will find it cost-effective to try to increase their level of process maturity; others will find it best to stay at a particular level and to improve their processes within that level. It depends on the type, the complexity and the size of the systems which they produce. The larger and more complex the system, the more benefits are likely to gained by increased process maturity level.

Once an organisation has used the GPG to established its maturity level and identified any particular areas of weakness, it should draw up an improvement plan. The GPG is then used to set realistic goals and identify requirements practices which will help meet those goals. Improvements based on the GPG will normally be incremental and evolutionary. We judge it unlikely that organisations will be able to perform radical redesign of their requirements processes. As a first step, though, the organisation's current requirements process maturity level should be assessed.

4.3. Requirements Engineering Process Maturity Assessment

In order to assess an organisation's requirements engineering process maturity, their existing process must be assessed. This will reveal how well the process is defined and areas of weakness in the process. Detailed process assessment, where a model is constructed showing the sequencing of activities and the individuals or roles within the process who are responsible for its enaction, can be problematic. One reason for this, as shown by empirical studies [Rodden 94, Sommerville 95a], is that different process participants may work to different models. Several techniques have been proposed for discovering the "true" process including ethnographic studies [Goguen 94] and process viewpoints [Sommerville 95b]. Developing an accurate process model is a complex business and most assessment techniques tend to require specialist process assessment input.

Because of this, the GPG suggests an approach where the aim is to rapidly gain an overall view of the extent to which a process is defined. This is achieved by identifying the requirements practices used by assessing them against a checklist of the good practices described in the GPG. Although this does not build a detailed model of an organisation's requirements process, the approach will reveal what practices are in use and the extent to which they are used, while remaining tolerant of differing views of how the process is enacted.

Against each practice in the checklist, one of the following assessments are made:

1. *Standardised* The practice has a documented standard in the organisation and is followed and checked as part of a quality management process.
2. *Normal use* The practice is widely followed in the organisation but is not mandatory.
3. *Used at discretion of project manager* Some project managers may have introduced the practice but it is not universally used.
4. *Never* The practice is never or very rarely applied.

The answers to these questions will vary according to who in the organisation is asked. It is useful, therefore to invest some time in identifying who should be asked. Similarly, there is little point in asking questions for which the answer is already known, especially questions about practices which are obviously never used. For these reasons, the process illustrated in Figure 3 is recommended as an aid to identifying who and what to ask.

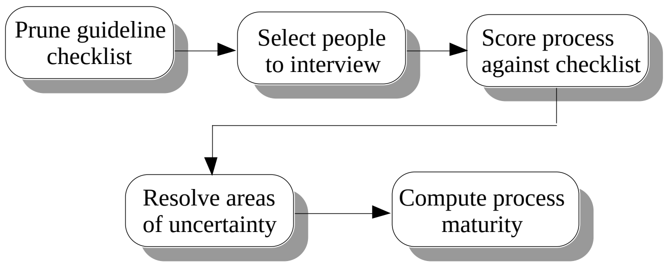
Prune guideline checklist
Select people to interview
Score process against checklist
Resolve areas of uncertainty
Compute process maturity

Figure 3. Process maturity assessment

The activities involved in process maturity assessment are:

1. *Prune guideline checklist* This activity is intended to identify, very quickly, practices which are never used.
2. *Select people to interview* A range of people across the organisation should be identified to interview for the maturity assessment.
3. *Score practices against checklist* This initial scoring should be "quick and dirty", and based on brief guideline descriptions.
4. *Resolve areas of uncertainty* This is likely to be the most time-consuming stage where the guideline descriptions may need to be used to discover if a practice is used.

5. *Compute process maturity* A score is compiled based on the above. 3 points are scored for a standardised guideline, 2 for normal use, 1 for discretionary use and 0 for guidelines which are never used. The higher the score, the fewer weaknesses there are likely to be in the process.

The assessment should reveal:

1. Particular areas of weakness. For example, an important area or activity (e.g. requirements management) may be performed ad-hoc, with few good practices used, or left to individual' discretion. This indicates an area where improvement efforts may be focused in order to achieve substantial improvement.
2. The level of process maturity. Table 2 is a rough indicator of the relationships between assessment scores and maturity levels.

Table 2 Assessment scores and maturity levels

Maturity level	**Assessment score**
Initial	Less than 55 in the basic guidelines. May have implemented some intermediate guidelines.
Repeatable	Above 55 in the basic guidelines but less than 40 in the intermediate and advanced guidelines.
Defined	More than 85 in the basic guidelines and more than 40 in the intermediate and advanced guidelines.

The scheme is based on a combination of empirical data from a small number of organisations, analysis of common practice and what we know about the cost and utility of the individual practices. It is intended to reflect that:

- To be a repeatable process, a good proportion of the basic practices (which are generally concerned with standardisation, management and ease of use) need to be implemented. Scenarios of what this translates to in practical terms might be: just over half the basic practices are standardised; (or more likely perhaps) the "top ten" guidelines have been standardised (see Section 4.4) with a majority of the remaining basic practices in normal or discretionary use.
- To be a defined process, the use of systematic methods suggested by the intermediate practice guidelines are required as process support. An example scenario here might be an organisation where half of the intermediate practices are standardised with approximately the same number of intermediate or advanced practices in normal or discretionary use.

Of course, this assessment scheme is not a precise instrument and is not intended for formal accreditation. An organisation should not react to the results of its application without careful analysis of the results. For example, there may be valid reasons for rejecting some practices and favouring others. We also note that other schemes for assessing requirements processes have been proposed (e.g. [El Eman 95b]), although these often involve measuring such things as numbers of requirements defects which can be both difficult and hard to interpret. The merits of our scheme is that it can give a relatively fast "ball park" indication of an organisation's process maturity in order to, for example, reveal shortfalls in the standardisation of basic practices.

Crucially, the scheme allows for flexibility. We do not believe that a CMM-like staged architecture is appropriate or feasible in the current state of the practice in requirements engineering. Hence, it isn't necessary to implement all of the basic practices to have a repeatable requirements engineering process. Some practices may not be appropriate for an organisation and need never be implemented.

Notice that while basic practices are relatively easily identified as belonging to that classification, it is often a finer line which distinguishes intermediate and advanced practices. The use of advanced guidelines often implies technical sophistication on the part of an organisation but does not necessarily imply a defined or even a repeatable process. For example, an organisation working in a critical system domain may employ advanced practices to address specific problems of that domain. These may result in systems which are highly dependable by, for example, focusing resources on the analysis and specification of critical components. However, other common problems, such as cost-overruns, will still occur unless basic practices are also in place.

To put this into context, background work on REAIMS suggests that the requirements engineering maturity of almost all organisations is at the *Initial* level. While many organisations have implemented some basic practices, few organisations have implemented all of them. Advanced practices have been used successfully in isolated projects but have been universally introduced in no organisations of which we are aware.

4.4. Planning for Requirements Engineering Process Improvement

Once the maturity level is established and any major weaknesses have been identified, planning for improvements can commence. Note that the process assessment should complement rather than replace knowledge of how the requirements process operates in practice. Hence, if there are problems which are already known (e.g. a meeting-dominated process) then these should also influence improvement planning. However, the problems apparent in the operation of a process may be symptoms of a deeper cause. The checklist-based assessment may reveal the underlying cause (e.g. no procedures for agreeing requirements changes).

There are four questions which should be answered when planning requirements engineering process improvements:

1. *What are the problems with the current processes?* These may be identifiable problems such as late delivery of products, budget over-runs, poor quality, products, etc. They may be less tangible problems such as poor staff morale, a reluctance for people to take responsibility or a meeting-dominated process where people spend too much time in meetings. Alternatively, the key problems might be problems of process understanding - no-one actually knows what processes are followed.

2. *What are the improvement goals?* These should normally be related to the identified problems. For example, if there are quality management problems, the goal may be to improve quality management procedures to ISO 9000 certification standard. If there are problems with budget-overruns, the goal may be to reduce the amount of rework which is required in a process. It is important that the goals should be realistic. There is no point in setting unrealisable goals or having unrealistic expectations about the benefits of new techniques or methods. Where there is any doubt, techniques such as Goal-Question-Metric (GQM) [Basili 88] should be used to control the setting and subsequent fulfilment of goals.

3. *How can process improvements be introduced to achieve these goals?* The set of guidelines in the GPG are intended to help make this decision. They suggest specific improvements (some small-scale, others large-scale) which can be applied in a range of different organisations. For example, if the maturity assessment has revealed poor use of good practices for a particular requirements area (e.g. requirements management), this suggests that practices designed to address that area should be a high priority. In such cases, it should be possible to correlate the observed requirements problems with the missing practices. Sometimes, however, the relationship between observed

problems and their root cause can be difficult to establish. If it proves impossible, it may be necessary to perform a detailed process assessment to establish the underlying weaknesses.

4. *How should improvements be controlled and managed?* Procedures should be established to collect feedback on improvements. These may be either quantitative measurements (e.g. Ami [Pulford 95]) of the process or informal comments on the improvements. Action must be taken in response to this feedback to correct any identified problems.

Once the improvement goals have been set, the specific practices to help achieve the goals can be selected. A strategy for their adoption must be established to ensure that their effect is maximised. The order in which improvement guidelines are introduced depends on the process and the need for improvements which have been identified. If starting from a baseline of an unstructured requirements engineering process, the following "top ten guidelines" provide a basic starting point:

- Define a standard document structure
- Make the document easy to change
- Uniquely identify each requirement
- Define policies for requirements management
- Define standard templates for requirements description
- Use language simply, consistently and concisely
- Organise formal requirements reviews
- Define validation checklists
- Use checklists for requirements analysis
- Plan for conflicts and conflict resolution

Notice that these are predominantly concerned with documenting and managing the requirements. As such, they represent relatively inexpensive prerequisites for a repeatable requirements process. As with other aspects of the GPG, this list represents judgements which we have made based on existing practice as represented in standards and our partner organisations' experience. However, they correspond closely to, or are implicit in, several standards. For example, eight of these are included in the UR section of ESA PSS-05.

5. Conclusions

The work outlined here brings together two of the most economically significant developments in software engineering: work on process improvement models and techniques for performing requirements activities. It is designed to help fill the gap left by existing process improvement and maturity methods by providing a process improvement scheme for the front-end activities of the software and systems development processes. It is intended to help disseminate practical information on requirements engineering good practice. It does this by placing in context the wide range of currently available practices, and a process improvement framework. This is designed to be implemented relatively cheaply, to require a minimum of prior process improvement experience and to dovetail with existing and emerging standards. The improvement model has no pretensions to be a fixed accreditation standard. Rather, it aims to provide organisations with a pragmatic route-map for quality improvement and cost saving.

Rather than being primarily aimed at helping organisations embark upon accreditation-driven organisational reengineering (as is supported by, for example, the CMM), our good practice guidelines are intended to help organisations identify and incrementally address specific process problems. Although practices are classified according to maturity level, it is not necessary to implement all practices at each level. The two dimensions of process deliverables/activities and guideline classification (basic, intermediate, advanced), encourage improvement efforts to be focused on particular problem areas. We have been influenced by both the CMM and SPICE. In practical application, however, it is closer in philosophy to that supported by SPICE's continuous architecture. Most importantly, we have paid considerable attention to ease of use and have not assumed any set of organisational practices.

Most of the support focuses on attainment of a repeatable process. Only one maturity level beyond repeatable - a defined process - is recognised. This is because there is strong anecdotal and empirical evidence that requirements process maturity lags well behind that of the subsequent development process. In time, this disparity may erode and the GPG will have to evolve to keep pace with changing practice. At the time of writing, however, we judge it that the attainment of requirements process repeatability continues to evade too many organisations. This is particularly true for organisations who have to adapt to new application domains, domains where technical innovation moves at a rapid pace, and domains subject to stringent time and cost constraints. In other words, to almost every domain.

6. Acknowledgements

We would like to thank all of our partners in the REAIMS project for their support in developing the Good Practice Guide. These are GEC-Alsthom Transport, Adelard, Aerospaciale Avions, Aerospaciale Protection Systems (APSYS), Digilog, TÜVit and the University of Manchester. Particular thanks are due to Stephen Shirlaw (GEC-Alsthom), Herbert Schippers (TÜVit), Robin Bloomfield (Adelard) and Jean-Pierre Heckman (Aerospaciale). We are also grateful to the paper's reviewers for their helpful comments.

7. References

[Basili 88] Basili, V., Rombach, H. 1988. The TAME Project: Towards Improvement-Oriented Software Environments, *IEEE Trans. on Software Engineering*, 14 (6). 758-773.

[Boehm 88] Boehm, B. 1988. A Spiral Model of Software Development and Enhancement, *IEEE Computer*, 21 (5). 61-72.

[Boehm 94] Boehm, B., Bose, P., Horowitz, E., Lee, M-J. 1994. Software Requirements as Negotiated Win Conditions. *Proc. 1st International Conference on Requirements Engineering (ICRE),* Colorado Springs, Co, USA. 74-83.

[Bollinger 91] Bollinger, T., McGowan, C. 1991. A Critical Look at Software Capability Evaluations, *IEEE Sofwtare*, 8 (4). 25-41.

[Davis 95] Davis, A. 1995. Tracing: A Simple Necessity Neglected, *IEEE Software*, 12 (5). 6-7.

[Dion 93] Dion, R. 1993. Process Improvement and the Corporate Balance Sheet, *IEEE Software*, 10 (4). 28-35.

[DoD 94] DoD MIL-STD-498 Software Development and Documentation, 1994.

[El Eman 95a] El Eman, K., Madhavji, N. 1995. A Field Study of Requirements Engineering Practices in Information Systems Development, *Proc. 2nd IEEE International Symposium on Requirements Engineering* (RE95), York, UK. 68-80.

[El Eman 95b] El Eman, K., Madhavji, N. 1995. Measuring the Success of Requirements Engineering Processes, *Proc. 2nd IEEE International Symposium on Requirements Engineering* (RE95), York, UK. 204-211.

[ESA 91] European Space agency Software Engineering Standard ESS-05, Issue 2, 1991.

[ESPITI 96] ESPITI newsletter issue 2. 1996. Software Process Improvement on the Right Road with ESPITI - the ESPITI European Survey Results. http://www.iai.fzk.de/espiti.

[Forsgren 95] Forsgren, P., Rahkonen, T. 1995. Specification of Customer and User Requirements in Industrial Control System Procurement Projects, *Proc. 2nd IEEE International Symposium on Requirements Engineering* (RE95), York, UK. 81-88.

[Goguen 94] Goguen, J. 1994. Requirements Engineering as the Reconciliation of Social and Technical Issues, in *Requirements Engineering*

Social and Technical Issues, M. Jirotka and J. Goguen (Eds), Academic Press. 165-199.

[Hersleb 97] Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., Paulk, M. 1997. Software Quality and the Capability Maturity Model. *Comm. ACM*, 40 (6). 30-40.

[Humphrey 88] Humphrey, W. 1988. Characterizing the Software Process: A Maturity Framework, *IEEE Software*, 5 (2). 73-79.

[Humphrey 91] Humphrey, W., Snyder, T., Willis, R. 1991. Software Process Improvement at Hughes Aircraft, *IEEE Software*, 8 (4). 11-23.

[Hutchings 95] Hutchings, A., Knox, S. 1995. Creating Products Customers Demand, *Communications of the ACM*. 38 (5). 72-80.

[Ince 94] Ince, D.: 1994. ISO 9001 and Software Quality Assurance, McGraw-Hill, 1994.

[ISO 95] ISO/IEC 12207 Software Life Cycle Processes, 1995.

[Jackson 93] Jackson, M., Zave, P. 1993. Domain Descriptions, *Proc. IEEE International Symposium on Requirements Engineering (RE93)*, San Diego, Ca., USA. 56-64.

[Johnson 93] Johnson, P. 1993. ISO 9000 Meeting the New International Standards, McGraw-Hill, 1993.

[Konrad 95] Konrad, M., Paulk, M. 1995. An Overview of SPICE's Model for Process Management, *Proc. Fifth International Conference on Software Quality*, Austin, TX, USA.

[Mazza 94] Mazza, C., Fairclough, J., Melton, B., De Pablo, D., Scheffer, A., Stevens, R. 1994. *Software Engineering Standards*, Prentice Hall.

[MoD 97] MoD Def Stan 00-55 Requirements for Safety Related Software in Defence Equipment. 1997.

[Paulk 93] Paulk, M., Curtis, W., Chrissis, M, Weber, C. 1993. Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-24, Software Engineering Institute, USA.

[Paulk 94] Paulk, M. 1994. A Comparison of ISO 9001 and the Capability Maturity Model for Software, CMU/SEI-94-TR-12, Software Engineering Institute, USA.

[Paulk 96] Paulk, M., Garcia, S., Chrissis, M. 1996. An Architecture for CMM Version 2, *Proc. 8th Software Engineering Process Group Conference*, Atlantic City, NJ, USA.

[Potts 94]	Potts, C., Takahashi, K., Anton, A. 1994. Inquiry-Based Scenario Analysis of System Requirements. *IEEE Software*, 11 (2). 21-32.
[Pulford 95]	Pulford, K., Kuntzmann-Combelles, A., Shirlaw, S. 1995. *A Quatitative Approach to Software Management: The Ami Handbook*, Addison-Wesley.
[Ramesh 95]	Ramesh, B., Powers, T., Stubbs, C., Edwards, M. 1995. Implementing Requirements Traceability: A Case Study, *Proc. 2nd IEEE International Symposium on Requirements Engineering* (RE95), York, UK. 89-95.
[Rodden 94]	Rodden, T. King, V., Hughes, J., Sommerville, I. 1994. Process Modelling and Development in Practice, *Proc. EWSPT'94,* Villard de Lans, France.
[Singh 96]	Singh, R. 1996. International Standard ISO/IEC 12207 Software Life Cycle Processes, Software Process Improvement and Practice, 2 (1). 35-50.
[Sommerville 95a]	Sommerville, I., Rodden, T. 1994. Social and Organisational Influences on Software Process Evolution, in *Trends in Software Processes,* A. Fuggeta and A. Wolf (Eds), Wiley.
[Sommerville 95b]	Sommerville, I., Katonya, J., Sawyer, P., Viller, S. 1995. Process Viewpoints, *Proc. 4th European Workshop on Software Process Technology,* Leiden, Netherlands.
[Sommerville 97]	Sommerville, I., Sawyer, P. 1997. Requirements Engineering A Good Practice Guide, Wiley.

Appendix A. An example guideline

Example Guideline: Define Validation Checklists

You should define a checklist or checklists which helps focus the attention of requirements validators on critical attributes of the requirements document. These checklists should identify what readers should look for when they are validating the system requirements.

Key benefits: Help to focus the validation process

Costs of introduction: Low-moderate

Costs of application: Low

Guideline type: Basic

Benefits

- Checklists add structure to the validation process. It is therefore less likely that readers will forget to check some aspects of the requirements document.
- Checklists help in the introduction of people who are new to requirements validation. The checklist gives them hints what they should be looking for so that they feel more able to participate in the process. This is particularly important for customer management and end-users who may not have requirements validation experience.

Implementation

The use of checklists for requirements analysis has already been discussed in Guideline 5.2, *Use checklists for requirements analysis* and similar checklists may also be used in the validation process. These checklists are oriented towards individual requirements; validation checklists should also be concerned with the quality properties of the requirements document *as a whole* and with the relationships between individual requirements. This can't be checked during requirements analysis as the requirements document is unfinished at that stage.

Questions which might be included in such a checklist should be based on the following general issues:

1. Are the requirements complete, that is, does the checker know of any missing requirements or is there any information missing from individual requirement descriptions?
2. Are the requirements consistent, that is, do the descriptions of different requirements include contradictions?
3. Are the requirements comprehensible, that is, can readers of the document understand what the requirements mean?
4. Are the requirements ambiguous, that is, are there different possible interpretations of the requirements?
5. Is the requirements document structured, that is, are the descriptions of requirements organised so that related requirements are grouped? Would an alternative structure be easier to understand?

6. Are the requirements traceable that is are requirements unambiguously identified, include links to related requirements and to the reasons why these requirements have been included? See Chapter 9 for guidelines on traceability.

7. Does the requirements document as a whole and individual requirements conform to defined standards?

Checklists should be expressed in a fairly general way and should be understandable by people such as end-users who are not system experts. As a general rule, checklists should not be too long. Checklists should not normally have more than ten items. If you have more than this, checkers cannot remember all items and must continually re-consult the checklist.

The danger , of course, is that checklist become too vague and it is impossible to answer the checklist questions in any useful way. You have to find the right balance between generality and detail. Unlike program inspections, low-level checklists concerned with very specific faults are not so good for requirements inspections because of the differences between the requirements for different types of system.

Checklists can be distributed and simply used as a reminder of what people should look for when reading the requirements document. Alternatively, they can be used more systematically where, for each requirement, an indication is given that the checklist item has been considered. This may be done on paper or you can manage this type of checklist completion by using a simple database or a spreadsheet. The checklist items are shown along the horizontal axis and the requirements along the vertical axis. The checker should fill in each cell with an appropriate comment. You shouldn’t force checkers to use an automated approach, as people may read the document in places and at times when they don’t have computer access.

Costs and problems

This is not an expensive guideline to implement if you use a fairly general checklist with questions like those listed above. To introduce the guideline, you need to draw up an initial checklist based on the experience of people who have been involved in requirements validation.

If a checklist is simply used a memory aid, there are no costs involved in applying the guideline. If checkers of the requirements must mark each requirement against the checklists, clearly some additional time is required. This should not be more than one or two minutes per requirement.

In principle, there should be few problems in applying the guideline so long as you have a fairly flexible process which allows people to ignore inappropriate checklist entries. In practice, some requirements analysts may resent the introduction of checklists as they see them as de-skilling the process. You have to emphasise that the checklists are designed to help them and point out that professional judgement must still be used.