Defining a Requirements Process Improvement Model

Sarah Beecham, Tracy Hall and Austen Rainer

Technical Report No 379

Department of Computer Science

Faculty of Engineering and Information Sciences

February 2003

Both software organisations and the academic community are aware that the requirements phase of software development is in need of further support. We address this problem by creating a specialised Requirements Capability Maturity Model (R-CMM[1]). The model focuses on the requirements process as defined within the established Software Engineering Institute's (SEI's) software process improvement framework. Our empirical work with software practitioners is a primary motivation for creating this requirements process improvement model. Although all organisations in our study were involved in software process improvement (SPI), they all showed a lack of control over many requirements related activities.

This report describes how the requirements process is decomposed and prioritised in accordance with maturity goals set by the SEI's Software Capability Maturity Model (SW CMM)(Paulk et al., 1995). Definitions are based on an abstraction of the SW CMM's requirements process improvement guidelines and related literature. This new focus will help practitioners to define their requirements process with a view to setting realistic goals for improvements. We outline the processes involved in creating the requirements model. The objective of this report is to give the software industry and academic community visibility into a new method of raising the profile of requirements within software development. This should lead to a better understanding of the requirements process and aid requirements process improvement.

1. Introduction

The software engineering literature is full of references alluding to how difficult it is to get requirements right. There is an emphasis on the importance of the requirements phase of software development, as getting requirements wrong will be costly in terms of lost time, lost revenue, loss of reputation and even survival. Solutions are offered to help practitioners with their technical and organisational problems. Models are created to guide organisations towards optimising their software processes and instruments are designed to measure process strengths. The software industry accepts that they are in need of help and the research community is endeavouring to guide them. With this plethora of information, however, how does the practitioner know where to start to look for help? Perhaps the solutions are out there? Perhaps there is too much information? There is obviously a salutary lesson to be learned from this profusion of ideas and as Humphrey points out,

"Every time software people have faced a new problem, instead of building on prior work, we have invented some new language, tool, or method. This forces us to start over, and it also forces our users to start over" (Humphrey, 2002)

The model we present in this report therefore does not offer 'new' solutions to individual problems nor does it present a 'new' framework. The Requirements CMM (R-CMM) builds on

[1] ®CMM is registered in the U.S. Patent and Trademark Office. Accuracy and interpretation of this document are the responsibility of the University of Hertfordshire, Centre for Empirical Software Process Research. Carnegie Mellon University has not participated in this publication.

prior work to guide practitioners towards improving their processes using a proven and familiar methodology.

Building on prior work, however, is not a straightforward activity. In this report we explain the many processes involved in developing a new model that is based on existing solutions and frameworks. We show how the model links best practices with the maturity characteristics of the SW CMM (Paulk *et al*., 1995). By providing these details, we aim make the requirements process more transparent. The result allows researchers to build on our work and gives practitioners the opportunity to select activities by giving a structure to what appears to be a profusion of unordered practices. The close coupling of the R-CMM with the SW CMM should help strengthen the requirements processes within a general software framework, as

"Requirements serve as living entities that are at the centre of the development activity. .actively managing changing requirements keeps the project under control and helps ensure the reliable, repeatable production of high-quality software products" (Leffingwell and Widrig, 2000).

1.1 Motivation for building a specialised requirements process improvement model

The primary motivation for this work emanates from our empirical research with 12 software development companies (Hall *et al*., 2002; Beecham *et al*., 2003c). Although these companies varied in size and application area, they were all using the SW CMM to guide them in their software process improvement activities. Interviewing three different staff groups (developers, project managers and senior managers) in 45 focus groups sessions revealed a general enthusiasm for this SPI model. A comment from a senior manager shows the wider benefits of implementing a CMM improvement method, "it should help people have a stronger sense of being professionals and working for a first class company and should help towards retaining staff and reducing costs". While a project manager takes a more pragmatic view stating that "[the CMM] helps you to control your destiny (Project Manager view). Comments made during these focus groups kept returning to problems practitioners were experiencing with requirements. For example a project manager states, "I don't believe that we spend enough time up front of the project doing all the work, understanding exactly what we need to do and consequently we learn as we go through and have to keep changing the requirements". And a developer clearly shows his frustration with the lack of control over inevitable changes in requirements, "We get changes in requirements during development which adds extra resource factors onto our jobs but that is not taken into account. It is not factored into our time scales. It is the biggest problem for me at the moment".

1.2 Intended Use

Although R-CMM is intended to be used by practitioners familiar with the SW CMM maturity concept, it should be possible to use the requirements model independently of the SW CMM to assess requirements process capability. The R-CMM is a 'suite of models, that guide practitioners from a high level view of the process, through to a detailed guideline and finally to a process assessment method that helps companies to satisfy particular company goals, see (Beecham *et al*., 2003a).

For further information on how we developed the rule-based requirements maturity framework from the SW CMM please refer to (Beecham *et al*., 2003b).

1.2 Organisation of report

The report is organised as follows.

In section two we give a background to our work. First we provide a rationale for using the SEI's Software Capability Maturity Model (SW CMM) to mould the R-CMM. Then we introduce the specialised R-CMM and explain how we have decomposed the requirements process. In section three we show how CMM Process Level Characteristics are used to focus on requirements.

Section four gives an example of a Level 2 requirements capability model and its associated processes. In section five we discuss where we hope to strengthen current SW CMM characteristics to aid requirements. In Section 6 we summarise and conclude our requirements model presentation.

2. Background

In this section we show how SW CMM characteristics create a framework for defining the requirements process.

2.1 The SW CMM

We have adapted the well-established SW CMM improvement framework (Paulk *et al*., 1995) to represent only the requirements process. This 'R-CMM' is designed to help practitioners strengthen the requirements process within software development. The model uses the SW CMM 5-level maturity characteristics to identify key requirements processes and prioritise their implementation.

There are several reasons for using this framework:

- The CMM is the most applied software process improvement model, e.g. (El Emam and Madhavji, 1995b). "The CMM has become a de facto standard as a basis for software process improvement (SPI). Using the CMM together with the CMM Based Assessment for Internal Process Improvement, thousands of users have made significant improvements in product quality, productivity and cycle time" (Rogoway, 1998).

- The practitioners in our study were all using the CMM as their software process improvement model (Hall *et al*., 2002; Beecham *et al*., 2003c). The CMM is therefore the model on which we base our findings;

- The CMM is designed to be tailored to the specific needs of a company (Paulk *et al*., 1995) and its framework has been adapted both inside and outside the field of Software Engineering. There are reportedly 34 CMMs developed by different groups using different architectures (Reifer, 2000). Examples of model adaptation include, (Hackos, 1997; Christie, 1999; Potter and Sakry, 2001; Ferraiolo, 2002; Neissink *et al*., 2002);

- The CMM is evolving. The SW CMM has been supplemented by other improvement paradigms such as the IDEAL improvement model (McFeeley, 1996), the People Capability Maturity Model (Curtis *et al*., 1995), Personal Software Process (Humphrey, 1997) and Team Software Process (Humphrey, 2000) in (Conradi and Fuggetta, 2002). The latest development is the Capability Maturity Model Integration (CMMI, 2001) that attempts to bring the different improvement models together under one meta-architecture which users can employ to generate combinations of CMMs of interest to them (Reifer, 2000);

- We do not want practitioners to have to start again, but want them to be able to build on previously proven methods (Humphrey, 2002).

The SW CMM contains many requirements related activities, "The CMM moves the organization toward an integrated view wherein technical requirements must be kept consistent with project plans and activities"(Leffingwell and Widrig, 2000). However the current "CMM approach to improvement seems to be 'necessary but not sufficient'. One of the primary contentions is that the CMM does not address many crucial processes or areas of activity" (Rogoway, 1998).

2.3. The Requirements Process

Although the requirements process in the R-CMM covers many requirements activities not included in the SW CMM, it does not model every activity in this phase of software development. For a process to be included in the R-CMM it needs to be at least one of the following:

- a solution to a recurring problem raised by practitioners in our empirical study
- a best practice in the SW CMM
- a recurring theme in the requirements literature

We have prioritised key requirements activities by coupling them to the SW CMM at incremental levels of process maturity. Appendix 1 gives an example of the motivation behind selecting processes at one level of process maturity.

2.4. The R-CMM

The requirements process is decomposed into several activities to allow for individual assessment of activities performed within this front end of software development. Each activity included in the model reflects implicit or explicit needs as expressed by practitioners in our empirical research (Beecham *et al.*, 2002; Hall *et al.*, 2002).

As process definition is recognized as a critical element in software process improvement (Christie, 1999) we include a glossary of terms and acronyms used in the report in Appendix 3. In Appendix 4 we give extended definitions of key requirements processes at a baseline level.

David (2000) highlights the importance of providing clear definitions,

"The notion of model and modelling imply the use of language, which in turn implies the definition of elementary units and terms that explain the relationships between these units. Modelling therefore requires the elaboration of concepts, together with a set of axioms to link such concepts" (David, 2000).

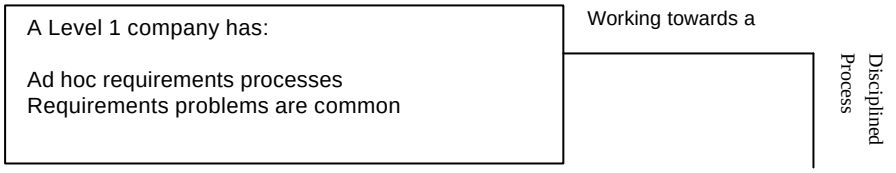
3. CMM Process Level Characteristics with a focus on requirements

This section gives an overview of how 5 levels of process maturity are characterised by the SW CMM. For example the need to assign responsibilities and resources to each activity is included. We have adapted these CMM general practices to relate specifically to the requirements process rather than the whole of software development. We detail how each maturity level relates to the R-CMM 'semantically'. (Please refer to (Beecham *et al.*, 2003b) for details of how the R-CMM relates to the SW CMM 'structurally'.)

We have taken the goals set by the maturity level characteristics in the SW CMM to act as initial goals for companies wanting to improve the requirements process. Although *all* SW CMM Key Process Areas (KPAs) start with 'goals' there is nothing specific about how companies should identify their own goals based on their own personal requirements weaknesses. The R-CMM leads companies to look at their current practices and set realistic goals when planning for requirements engineering process improvement, as recommended by (Davis, 1988; Sawyer *et al.*, 1997; IEEE, 1998).

3.1 R-CMM Level 1:

Level 1 Goal: To raise awareness of the requirement process

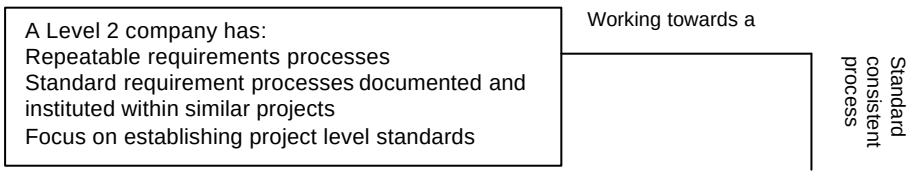


Level 1 organisations are working towards developing a disciplined process and need to raise their awareness of the problems they are having with requirements. Examining the model will help managers gain an insight into their requirements process and encourage them to buy into the idea of software process improvement (Christie, 1999). It is likely that managers at this level will need to prioritise their requirements problems. By definition, this 'ad-hoc' level has no associated 'best practices'. Assessment starts at Level 2, where if an organisation cannot meet the criteria they are encouraged to seek out solutions.

Our empirical research results showed that Level 1 companies are most concerned with vague requirements, traceability and requirements process (technical problems); and resources, training and skills (organisational problems). See Tables 1 and 2 in Appendix 2 for a breakdown of requirements issues raised by CMM Level 1 companies in our empirical study.

3.2 R-CMM Level 2:

Level 2 Goal: To implement a repeatable requirements process



The SW CMM summarises companies at this 'repeatable' Level 2 process maturity as having established basic project management processes to track cost, schedule, and functionality. The necessary process discipline is now in place to repeat earlier successes on projects with similar applications (Paulk *et al.*, 1995) p.125.

Progressing to Level 2 involves organisations in examining their requirement processes in detail. For example, they need to identify their current requirements processes, define them, and monitor how they behave. The requirements model at level 2 can help managers to identify and document their individual requirements processes. It introduces controls over processes that they may not have identified as necessary. Managers begin to gain a general overview and can address issues associated with individual projects. Formal measurement at this level will be project based, include costs and schedules and resource needs. "Only fairly gross metrics (mostly experience-based) that are associated with costs and schedules are likely to be available" (Christie, 1999). This level will provide the basis for further quantitative data capture at higher levels of maturity. As requirements management is a Level 2 Key Process Area in the CMM, the R-CMM reflects this by creating a baseline model of requirements processes. Level 2 therefore establishes most of the requirements key processes that are built on as a company matures. A Level 2 compliant company should have the processes in place listed in section 4, Figure 2, and should be working towards creating a standard and consistent organisation-wide requirements process.

Our empirical research showed that organisations with a level 2 process capability were particularly concerned with complex requirements, requirements growth and undefined processes (technical issues); and staff retention and culture (organisational issues). (See Tables 1 and 2 in Appendix 2 for a breakdown of requirements issues raised by CMM Level 2 companies in our empirical study).

In section 4, we show how we have decomposed this level into 20 key requirements processes. An expert panel is currently validating this candidate list of processes.

3.3 R-CMM Level 3:

Level 3 Goal: To implement a defined requirements process

The SW CMM summarises companies at this 'defined' Level 3 of process maturity as having a

documented, standardised and integrated organisation-wide software process for both management and engineering activities. All projects now use a documented and approved version of the organization's process for developing and maintaining software (adapted from (Paulk *et al.*, 1995) p. 193).

Baseline disciplined activities have been established at level 2. Level 3 co-ordinates the standard requirement processes that are documented and instituted within similar projects. Focus is on processes and creating company-wide, organisational standards and visibility.

Level 3 compliance requires a focus on the organization process, the training program, integrated software management, software product engineering, intergroup co-ordination and peer reviews. Management, have an increasing ability to see and control requirements activities which may previously may have been viewed as black boxes (Christie, 1999). There is an emphasis on recording data, for example keeping a record of task duration and requirements stability. Data recorded from the assessment of requirements activities can be used to understand their behaviour. For example does the introduction of a company-wide method for tracing requirements result in a better requirements process?

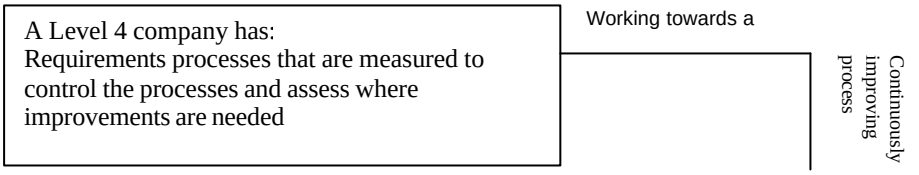
Measurement of Level 3 processes might include data on the approach, deployment and results of processes used to control requirements changes, requirements growth and complex requirements. And Christie advocates that "Level 3 organisations should support a library of processes that can be reused, with appropriate adaptation, in other parts of the organization" (Christie, 1999). This applies to all the requirements processes identified as key to this level of capability.

Our empirical research showed that Level 3 companies are most concerned with user understanding of requirements; internal and external communications. See Tables 1 and 2 in Appendix 2 for a breakdown of requirements issues raised by CMM Level 3 companies in our empirical study.

In Appendix 6, we list 19 candidate processes that are potentially key to the Level 3 capability. Our expert panel validation does not include this level of maturity.

3.4 R-CMM Level 4:

Level 4 Goal: To implement a managed requirements process



The SW CMM summarises companies at this 'managed' Level 4 process maturity as collecting detailed measures of the software process and product quality. Both the software process and products are quantitatively understood and controlled using detailed measurements (Paulk *et al.*, 1995).

As we have very little empirical evidence to guide activities at this level (one level 4 company) we have been guided by the CMM activities. We have adapted these general software activities to relate to the requirements process only.

The problems suggested by our level 4 company were mainly organisational, therefore these activities that are heavily focussed on managing the requirements process quantitatively should address some of their issues raised.

At level 4 organisations can begin to set quantitative quality goals .. managers are also better able to assess the risks of modifying their processes or integrating new process elements. The aim is to operate processes within quantitative performance limits (Christie, 1999).

Examples of SW CMM activities adapted to focus on the requirements process are:

- "The strategy for the data collection and the quantitative analysis to be performed are determined based on the project's defined [requirements] process"; and

- "The measurement data used to control the project's defined [requirements] process quantitatively are collected according to a documented procedure.

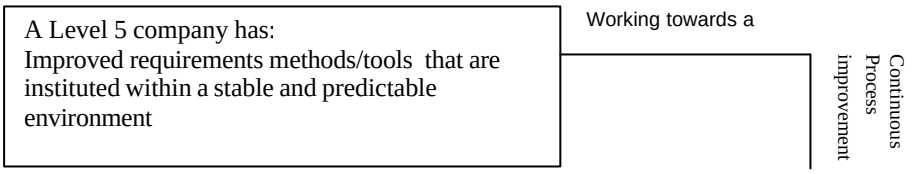
Examples of measurement data include: effectiveness of training; number and severity of defects found in the software requirements (Paulk *et al.*, 1995).

In our empirical research we record a very strong link between level 4 companies and problems with culture. However, with the small size of the sample group for this level (1 company, 3 focus groups) we do not suggest this is necessarily typical of all level 4 companies. See Tables 1 and 2 in Appendix 2 for a breakdown of requirements issues raised by the CMM Level 4 company in our empirical study.

For a list of suggested requirements processes at level 4 for please see Appendix 6. This list is adapted mainly from the CMM as we do not have enough data from our empirical study to influence the direction of recommendations at this level of maturity.

3.5 R-CMM Level 5:

Level 5 Goal: To implement an optimising requirements process



The SW CMM summarises this 'optimizing' level 5 as having continuous process improvement enabled by quantitative feedback from the process and from testing innovative ideas and technologies (Paulk *et al.*, 1995).

Moving up from level 4 to level 5 companies should have "a wealth of metric data, validated models and can use models to track and manage the course of a process" (Christie, 1999). Elements of processes can be confidently modified. New and improved ways of building the software are continually being tried in a controlled manner (Christie, 1999).

We do not have any empirical data to feed into this level of requirements process improvement. We therefore rely mainly on the CMM for best practices and support these through an analysis of the associated requirements literature. For a list of recommended requirements processes at Level 5 maturity please refer to Appendix 6. These processes do not form part of our validation procedure and are purely theoretical.

3.6 The Five Level Goal Focus

The five levels described in sections 3.1-3.5 above show the incremental steps needed to progress to a level 5 requirements capability. We have converted the SW CMM level characteristics to become requirements 'goals'. This goal focussed view is derived from a goal/question/metric paradigm (Basili and Romach, 1988). The R-CMM looks at requirements process goals and the problems (in the form of a question) that practitioners may encounter in trying to achieve these goals. Should the practitioner need help in solving individual process problems, the model defines 'what' the process should do in terms of generic best practices. 'How' these guidelines are implemented is left to the individual organisation based on their interpretation, resources, culture and individual strategies and goals. Indeed, there is a great deal in the literature relating to how management can implement these best practices (Curtis, 2000; Baddoo and Hall, 2002a; Rainer and Hall, 2002).

The new model strikes a balance between helping SPI managers recognise where their problems are, and identifying and prioritising their requirements goals. The model guides the organisation towards aligning process improvements with company goals and strategies rather than focussing on processes in isolation. This goal focus is advocated by (Rifkin, 2001) and (Potter and Sakry, 2001). Fayad (Fayad, 1997) also supports this goal and product approach to process improvement reminding us that "software development organizations exist to develop software rather than processes".

We have made an assumption in our R-CMM, that companies want to either improve their requirements processes or assess their requirements strengths and weaknesses. Using the level 2 characteristics (in 3.2 above) as an example, we set the Level 2 R-CMM goal "to implement a repeatable requirements process".

Using the Basili template, in (Solingen and Berghout, 1999), this Level 2 Requirements goal is defined as follows:

Basili Goal setting Criteria	Applied to Requirements Process improvement
Analyse	The requirements process
For the purpose of	Understanding, controlling and improving the requirements process
With respect to	Creating/developing/complying to a repeatable requirements process as defined by its sub-processes
From the viewpoint of	All requirements process stakeholders
In the context of	The environment in which the requirements process is being used

Figure 1: An example of setting the goal for a level 2 capability using the Basili template

This method of defining goals should help managers to focus on the problem area and helps decision making.

For further details on how the Goal/Question/Metric method is applied in the R-CMM please refer to (Beecham *et al*., 2003b).

4. A Level 2 example of the R-CMM

We have taken the Level 2 SW CMM characteristics to create a more detailed requirements model. This extension of the high level model characteristics serves as an example of how the other levels (3, 4 and 5) might also be extended to provide more detailed guidance to users. The more detailed the model is however, the more prescriptive the model becomes. The requirements model is purposefully descriptive and normative to allow for a more universal application of its improvement concepts. This presentation is consistent with the CMM, that "describes what a process should address rather than how it should be implemented" (Paulk et al, 1995, p.89) In Appendix 7 we give an overview of the tension between descriptive and prescriptive modelling.

(Beecham *et al*., 2003b) gives an explanation of how this Level 2 R-CMM relates to the SW CMM and to other levels in R-CMM.

4.1 Five stages of requirements

Figure 2 shows how the level 2 goal to "implement a repeatable requirements process" is decomposed into 5 requirements related questions. The 5 questions in the R-CMM are used to interrogate recognised stages in the requirements process (Dorfman and Thayer, 1997) and (Pressman, 2001). Each requirements stage has a set of associated processes. The R-CMM interrogates processes in this way to bridge the gap between a traditional, structured 'lifecycle' view of requirements engineering and the more fluid process view of requirements. For definitions of these 5 requirements phases (represented in Q1 – Q5) please see Appendix 5.

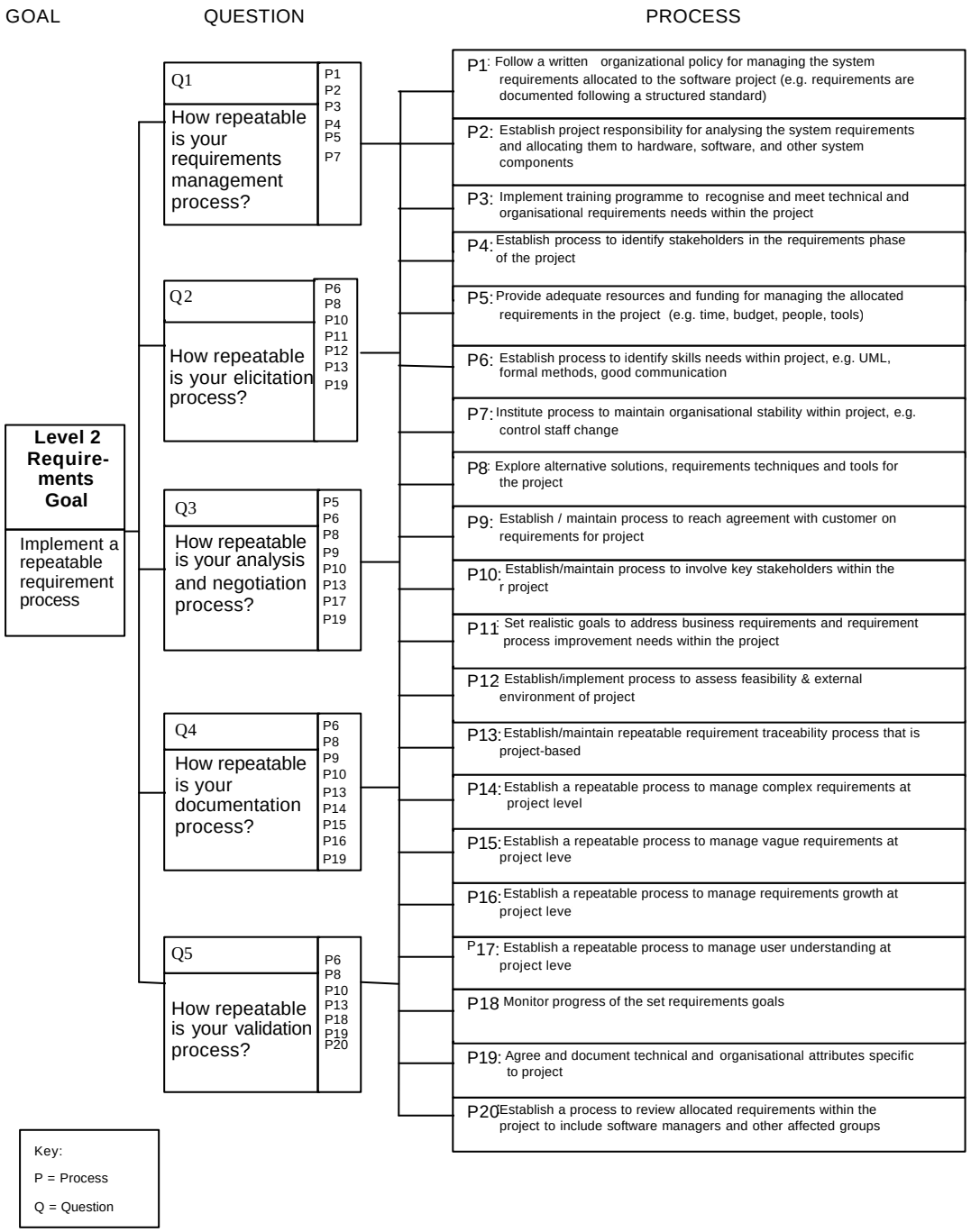


Figure 2: The Requirements CMM: A Level 2 Goal Focussed view of Requirements Processes

The Figure 2 model guides users to examine whether 5 phases of requirements (represented by five questions) are complying with the level 2 maturity goal. To answer this the user must determine whether sets of related processes have been successfully instituted at a repeatable level.

4.2 Requirements Guidelines

Key processes, as in the Figure 2 example (P1 – P20), are at a fairly high level of abstraction. To aid interpretation and correct implementation we plan to provide a more detailed guideline for each process.

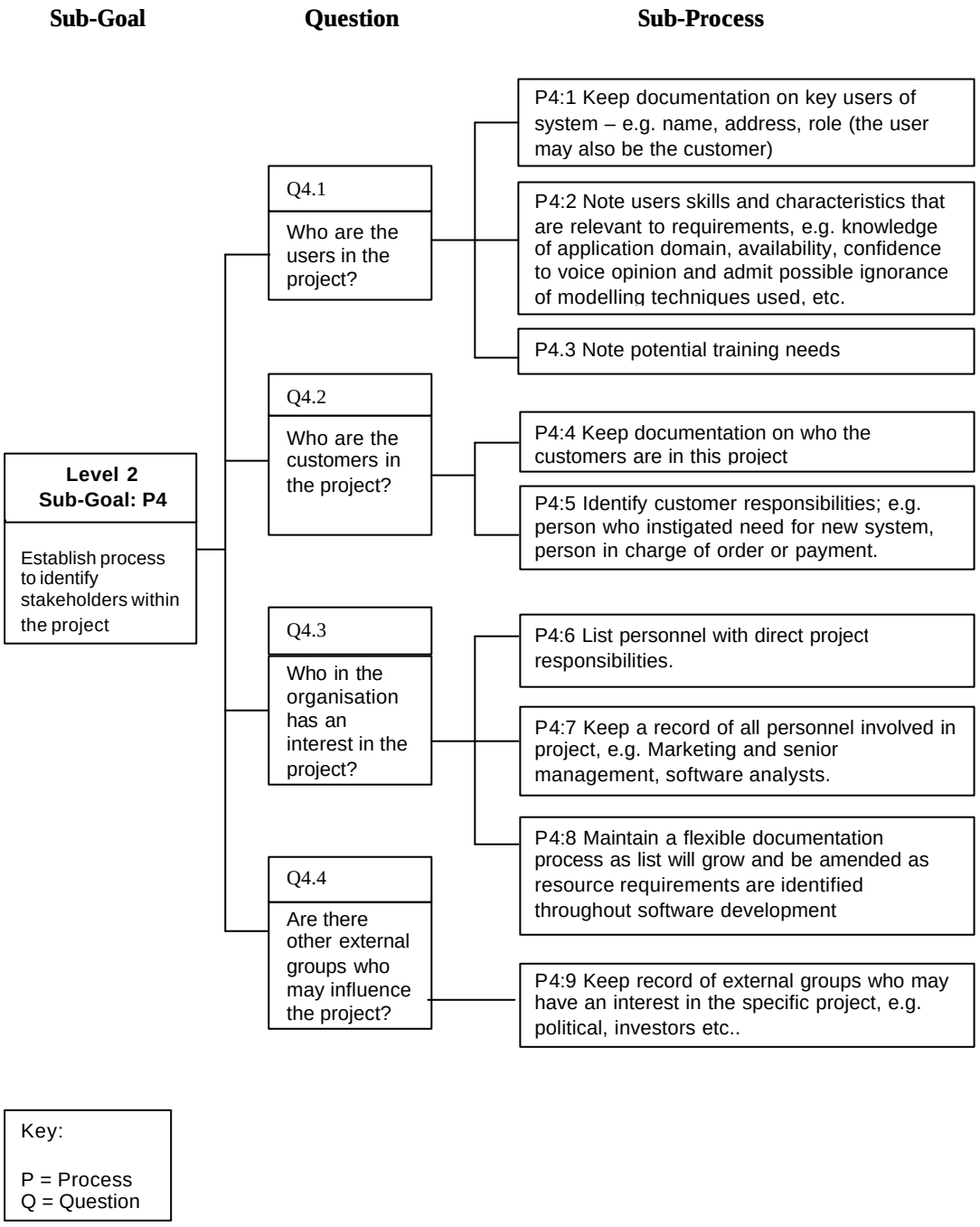


Figure 3: Guideline example of a Level 2 Requirements Process

An example of a level 2 process guideline (P4) with a goal to "establish a process to identify stakeholders within the project" is given in Figure 3. The P4 process from figure 2 becomes the 'goal' for the finer grained guideline model. This retains the GQM improvement focus.

The models presented in figures 2 and 3 represent level 2 maturity. These frameworks will be extended to cover all processes and Levels 3, 4 and 5.

4.3 Assessment

For an example of how each process is assessed to establish whether it is capable of achieving its goal is given in (Beecham *et al*., 2003a).

5. Discussion

Our empirical research highlighted problem areas in software development. This initial work lead to a detailed study of the problems practitioners from 4 levels of capability maturity were experiencing with their requirements (Hall et al, 2002). One of our most significant findings agree with Paulk et al (1995) that, "although software engineers and managers often know their problems in great detail, they may disagree on which improvements are most important". An aim of the requirements model, therefore, is to help organisations agree on a strategy for improvement and achieve a consensus between management and professional staff on what requirement related improvement activities to undertake first. It follows the evolutionary path designed by the CMM that is ordered into stages to create a firm foundation to build on.

A further finding in our empirical study is the need to manage human (or organisational) aspects of requirements together with the technical processes. (Patel, 1999) makes a distinction between physical information and human information and believes that in order make information systems development more sensitive to organisational environments, information needs to be tailorable. The R-CMM addresses both these needs.

In disagreement with some criticisms levelled at the CMM, we believe that the CMM covers most of the requirement process activities required to create a mature capability. However, in its current form the SW CMM addresses all software development that necessitates covering a multitude of activities within its KPAs that do not apply to requirements. It is therefore difficult to isolate and identify all the activities required to meet precise requirement goals. Our empirical study suggests that the SW CMM is not helping practitioners to:

a) identify requirements processes
b) define requirements processes
c) recognise requirements process problems
d) assess and agree requirement improvement priorities
e) relate requirements process problems to requirement improvement goals
f) relate requirement improvement goals to the general CMM guidelines and activities

Our specialised requirements process improvement model isolates the requirements process and assists practitioners to identify and prioritise their problems. It takes the advice given by Paulk et al (1995) and guides practitioners to focus on "a limited set of activities" and "work aggressively to achieve their goals". By so doing, an organisation can "steadily improve its organization-wide software process to enable continuous and lasting gains in software process capability" (Paulk et al, 1995).

6. Conclusion

Requirements remain a major source of problem in software development despite the increase in organisations implementing software process improvement programs. As companies continue to follow the improvement guidelines set out in the SW CMM, we have investigated and evaluated whether it is possible to augment and adapt the CMM to focus on the requirements process.

We give definitions of processes and the motivation for including these processes at one level of maturity. An expert panel is currently assessing the value of each process to the overall requirements phase of software development. We aim to create a model that prioritises the requirements processes both within each level of maturity and between levels of maturity.

A first step to improving and managing requirements is to define and tailor processes to meet the specific needs of the organisation. The SW CMM provides a high level view of requirements that with further detail should prove a more useful tool for both practitioners and researchers in the field of process improvement and requirements engineering. The R-CMM therefore should act as a guide and prompt to help practitioners through the diverse processes involved in requirements engineering. It is intended that the actual processes involved in developing the model, as outlined in this report, will provide a foundation for future development in the area of requirements process improvement.

Acknowledgements:We are sincerely grateful to all the companies and practitioners for their participation in the 'Managing Practitioner impact on Processes and Products' (PPP) project. The PPP project is funded by the UK's Engineering and Physical Science Research Council, under grant number EPSRC GRL91962. We are also grateful to Pete Sawyer for his support in the project.

References

ami. 1992. A Quantitative Approach to Software Management. Univ of the Southbank, London.

Arisholm, E. and Sjoberg, D. I. K. 2000. Towards a framework for empirical assessment of changebility decay. Journal of Systems and Software, (53): 3-14.

Bach, J. 1999. What Software Reality is Really About. IEEE Computer, 32 (12): 148-149.

Baddoo, N. and Hall, T. 2002a. De-motivators for Software Process Improvement: An analysis of practitioners' views. Journal of Systems and Software.

Baddoo, N. and Hall, T. 2002b. Motivators of Software Process Improvement: an analysis of practitioners' views. The Journal of Systems and Software, 62 (2): 85-96.

Basili, V. R. 1995. Measurement Frameworks in Software Quality Assurance and Measurement: A Worldwide Perspective. Fenton, N., Whitty, R. and Iizuka, Y., eds.: International Thomson Computer Press London.

Basili, V. R. and Romach, H. D. 1988. The Tame Project: Towards Improvement-oriented software environments. IEEE Transactions on Software Engineering, 14 (6): 758-773.

Beecham, S., Hall, T. and Rainer, A. 2003a. Assessing the strength of a requirements process. University of Hertfordshire, Hatfield. Technical Report TR 381.

Beecham, S., Hall, T. and Rainer, A. 2003b. Building a Requirements Process Improvement Model. University of Hertfordshire, Hatfield. Technical Report TR 378.

Beecham, S., Hall, T. and Rainer, A. 2003c. Software Process Improvement Problems in 12 Software Companies: An Empirical Analysis. Empirical Software Engineering, 8 (1): 7-42.

Boehm, B. 2001. Using Win-Win for Requirements Negotiation: a mini tutorial. Imperial College, April 2001, London.

Boehm, B. W. 1981. Software Engineering Economics. Englewood Cliffs: Prentice-Hall, Inc.

Brodman, J. G. and Johnson, D. L. 1994. What Small Businesses and Small Organisations Say About the CMM. 16th International Conference on Software Engineering, May 16-21, Sorrento, Italy. IEEE. 331-340.

Christie, A. M. 1999. Simulation in support of CMM-based process improvement. Journal of Systems and Software, (46): 107-112.

CMMI. 2001. Capability Maturity Model ®. Integration (CMMI SM), Version 1.1. Software Engineering Institute, Technical Report CMU/SEI-2002-TR-003. ESC-TR-2002-003.

Conradi, R. and Fuggetta, A. 2002. Improving Software Process Improvement. IEEE Software, 19 (4): 92-99.

Cottengim, D. 2002. Prerequisites for Success: Why Process Improvement Programs Fail. CrossTalk The Journal of Defense Software Engineering,: 26-30.

Cugola, G. and Ghezzi, C. 1998. Software Processes: a Restrospective and a Path to the Future. Software Process - Improvement and Practice, (4): 101-123.

Curtis, B. 2000. The Global Pursuit of Process Maturity. IEEE Software, 17 (4): 76-78.

Curtis, B., Hefley, W. E., Miller, S. and Konrad, M. 1995. People Capability Maturity Model. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Curtis, B., Krasner, H. and Iscoe, N. 1988. A Field Study of the Software Design Process for Large systems. Communications of the ACM, 31 (11): pp 1268-1287.

David, A. 2000. Models implementation: A state of the art. European Journal of operational Research, 134: 459-480.

Davis, A. 1988. A Taxonomy for the Early Stages of the Software Development Life Cycle. Journal of Systems and Software,: 297-311.

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

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., Ta, A. and Theofanos, M. 1993. Identifying and Measuring Quality in a Software Requirements Specification. In Proceedings of the First International Software Metrics Symposium, 141-152.

Dorfman, M. and Thayer, R. H. 1997. Software Engineering. Los Alamitos, CA: IEEE Computer Society Press.

El Emam, K. and Madhavji, N. H. 1995a. A Field Study of Requirements Engineering Practices in Information Systems Development. Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press. 68-80.

El Emam, K. and Madhavji, N. H. 1995b. The Reliability of Measuring Organizational Maturity: John Wiley & Sons.

Fayad, M. E. 1997. Software Development Process: A Necessary Evil. Communications of the ACM, 40 (9): 101-103.

Ferraiolo, K. M. 2002. Considerations for Re-architecting the SEI CMM:
http://www.systemcorp.com/2002/downloads/frames/karen_frame.html. Accessed 16.7.02.

Gause, D. C. and Weinberg, G. M. 1989. Exploring requirements: quality before design. New York, NY: Dorset House Publishing.
Gross, D. and Yu, E. 2001. From Non-Functional Requirements to Design through Patterns. Requirements Engineering, (6): 18-36.
Hackos, J. T. 1997. From Theory to Practice: Using the Information Process Maturity Model as a tool for Strategic Planning. Technical Communication, (44): 369-381.
Hall, T., Beecham, S. and Rainer, A. 2002. Requirements Problems in Twelve Companies: An Empirical Analysis. IEE Proceedings for Software, October, 149: No.5: 153-160.
Hofmann, H. F. and Lehner, F. 2001. Requirements Engineering as a Success Factor in Software Projects. IEEE Software,: 58-66.
Humphrey, W. S. 1997. Introduction to the Personal Software Process. Reading, M.A.: Addison-Wesley.
Humphrey, W. S. 2000. Introduction to the Team Software Process. Reading, M.A.: Addison-Wesley.
Humphrey, W. S. 2002. Three Process Perspectives: Organizations, Teams, and People. Annuls of Software Engineering, 14: 39-72.
IEEE. 1998. IEEE Guide to Software Requirements Specification. IEEE Press, Piscataway, N.J.
IEEE. 1999. IEEE Standards Software Engineering, 1999 Edition. Volume One: Customer and Terminology Standards. The Institute of Electrical and Electronics Engineers, Inc., New York, USA.
Jackson, M. 1995a. Problems and Requirements. Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press. 2-8.
Jackson, M. 1995b. Software Requirements & Specifications: a lexicon of practice, principles and prejudices. Wokingham, England: Addison Wesley Publishing Company.
Kitchenham, B. 1995. Using function points for software cost estimation in Software Quality Assurance and Measurement: A Worldwide Perspective. Fenton, N., Whitty, R. and Iizuka, Y., eds.: International Thomson Computer Press London.
Knethen, A. v., Paech, B., Kiedaisch, F. and Houdek, F. 2002. Systematic Requirements Recycling through Abstraction and Traceability. Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02), Essen, Germany. The Computer Society. 273-281.
Kotonya, G. and Sommerville, I. 1998. Requirements Engineering : processes and techniques. Chichester: John Wiley & Sons.
Lauesen, S. and Vinter, O. 2001. Preventing Requirement Defects: An Experiment in Process Improvement. Requirements Engineering Journal, 6: 37-50.
Leffingwell, D. and Widrig, D. 2000. Managing software requirements: a unified approach. Reading, Massachusetts: Addison-Wesley.
Linberg, K. R. 1999. Software developer perceptions about software project failure: a case study. Journal of Systems and Software, 49: 177-192.
Loucopoulos, P. and Karakostas, V. 1995. System requirements engineering. London, UK: McGraw-Hill Book Company Europe.
Lubars, M., Potts, C. and Richter, C. 1993. A Review of the State of the Practice in Requirements Modeling. In proceedings of the IEEE International Symposium on Requirements Engineering, January 4-6, San Diego, CA. 2-14.
Madhavji, N. H. 1991. The process cycle. Software Engineering Journal, September: 234-242.
Maiden, N. and Gizikis, A. 2001. Where do Requirements Come From? IEEE SOftware, (September/October).
McFeeley, B. 1996. IDEAL: A User's Guide for Software Process Improvement. Software Engineering Instutute, Carnegie Mellon University, Pittsburgh, PE, USA. CMU/SEI-96-HB-001.
Middleton, P. and McCollum, B. 2001. Management of process improvement by prescription. Journal of Systems and Software, (57): 9-19.
Moynihan, T. 2000. Coping with 'requirements-uncertainty': the theories-of-action of experienced IS/software project managers. Journal of Systems and Software, (53): 99-109.
Neissink, F., Clerc, V. and Vliet, H. 2002. The IT Service Capability Maturity Model. Software Engineering Research Centre, Utrecht. http://www.itservicecmm.org
Onvlee, J. 1995. Use of function points for estimation and contracts in Software Quality Assurance and Measurement: A Worldwide Perspective. Fenton, N., Whitty, R. and Iizuka, Y., eds.: International Thomson Computer Press London.
Patel, N. V. 1999. The Spiral of Change Model for Coping with Changing and Ongoing Requirements. Requirements Engineering, 4 (4): 77-84.
Paulk, M. C., Weber, C. V., Curtis, B. and Chrissis, M. B. 1995. The Capability Maturity Model: Guidelines for Improving the Software Process. Reading, Massachusetts: Addison Wesley Longman, Inc.
Pfleeger, S. L. and Rombach, H. D. 1994. Measurement based Process Improvement. IEEE Software, (July): 9-11.
Potter, N. and Sakry, M. 2001. Practical CMM: Software Development. Accessed 9.27.02. http://www.sdmagazine.com/print/documentID=11079

Potts, C. 1997. Requirements Models in Context. Third IEEE International Symposium on Requirements Engineering, January 6-10, Annapolis, Maryland. IEEE Computer Society Press. 102-104.

Pressman, R. 2001. Software Engineering 5th Edition: McGraw Hill.

Rainer, A. and Hall, T. 2002. Key success factors for implementing software process improvement: a maturity-based analysis. Journal of Systems and Software.

Reifer, D. J. 2000. The CMMI: it's formidable. Journal of Systems and Software, 50: 97-98.

Rifkin, S. 2001. Why Software Process Innovations Are Not Adopted. IEEE Software.

Rogoway, P. 1998. How to Reap the Business Benefit from SPI: Adding SPICE while preserving the CMM (Motorola): SPI NEWSPAPER. European Software Process Improvement. SPI and Assessments. Accessed 16.2.03. http://www.iscn.at/select_newspaper/assessments/motorola.html

Rule, G. P. 2001. Using measures to understand requirements. ESCOM.

Sawyer, P., Sommerville, I. and Viller, S. 1997. Requirements Process Improvement Through the Phased Introduction of Good Practice. Software Process Improvement and Practice, 3 (1).

Shere, K. D. 1988. Software Engineering and Management. Englewood Cliffs, N.J.: Prentice Hall, Inc.

Solingen, R. v. and Berghout, E. 1999. The Goal/Question/Metric Method: a practical guide for quality improvement of software development. Maidenhead, UK: McGraw-Hill Publishing Company.

Sommerville, I. and Sawyer, P. 1997. Requirements Engineering A good practice guide. Chichester: John Wiley & Sons Ltd.

StandishGroup. 1995. Chaos Report. http://www.scs.carleton.ca/~beau/PM/Standish-Report.html

Stark, M. 2003. Integrating Theory and Practice: Applying the Quality Improvement Paradigm to Product Line Engineering: NASA, Goddard Flight Center. Accessed 5.3.03. http://isc.gsfc.nasa.gov/Papers/DOC/icse24stark.pdf

Stricker, C. 1995. Evaluating effort prediction systems in Software Quality Assurance and Measurement: A Worldwide Perspective. Fenton, N., Whitty, R. and Izuka, Y., eds.: International Thomson Computer Press London.

Sutcliffe, A. G., Economou, A. and Markis, P. 1999. Tracing Requirements Errors to Problems in the Requirements Engineering Process. Requirements Engineering, (4): 132-151.

Thayer, R. H. and Dorfman, M. 1990. System and Software Requirements Engineering. Los Alamitos, CA: IEEE Computer Society Press.

Watkins, R. and Neal, M. 1994. Why and How of Requirements Tracing. IEEE Software, (July): 104-106.

Weinberg, G. M. 1998. The psychology of computer programming. New York, NY: Dorset House Publishing.

Yourdon, E. 1995. Techniques of Program Structure and Design. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

Yu, E. S. K. and Mylopoulos, J. 1997. Modelling Organizational Issues for Enterprise Integration. International Conference on Enterprise Integration and Modelling Technology, October 28-30, 1997, Turin, Italy.

APPENDICES

Appendix 1: Motivation for including the requirements processes

Appendix 2: Contingency Tables giving Requirements Problems in 12 Software Development Companies by CMM Level

Appendix 3: Acronyms and Glossary of Terms

Appendix 4: Defining Requirements Processes at Level 2 maturity

Appendix 5: Five phases of requirements viewed from a G/Q/M paradigm

Appendix 6: R-CMM: Levels 3, 4 and 5

Appendix 7: Descriptive Modelling

Appendix 1: Motivation for including the requirements processes

Although the Requirements CMM (R-CMM) includes some supplementary requirements activities (not found in the CMM) it does not set out to include *all* requirements activities. Requirements processes are included for the following three reasons:

Primary Source	Requirements Process Activity
1. CMM	P1: Follow a written organizational policy for managing the system requirements allocated to the software project (e.g. requirements are documented following a structured standard)(O)
	P2: Establish project responsibility for analysing the system requirements and allocating them to hardware, software, and other system components (O)
	P5: Provide adequate resources and funding for managing the allocated requirements in the project (e.g. time, budget, people, tools) (O)
	P20: Establish a process to review allocated requirements within the project to include software managers and other affected groups (T)
2. Empirical Research	P3: Implement training programme to recognise and meet technical and organisational requirements needs within the project (O)
	P4: Establish process to identify stakeholders within the requirements phase of the project (O)
	P6: Establish process to identify skills needs within project, e.g. UML, Formal methods (O)
	P7: Institute process to maintain organisational stability within project, e.g. control staff change (O)
	P10: Establish/maintain process to involve key stakeholders in requirements phase of project (O)
	P13: Establish/maintain repeatable requirement traceability process that is project-based (T)
	P14: Establish a repeatable process to manage complex requirements at project level (T)
	P15: Establish a repeatable process to manage vague requirements at project level (T)
	P16: Establish a repeatable process to manage requirements growth at project level (T)
	P17: Establish a repeatable process to manage user understanding at project level (O&T)
	P19: Agree and document technical and organisational attributes specific to project (T)
3. Literature	P8: Explore alternative solutions, requirements techniques and tools for the project (T)
	P9: Establish / maintain process to reach agreement with customer on requirements for project (O)
	P11: Set realistic improvement goals to address problems in the requirements process project (O)
	P12: Establish/implement process to assess feasibility & external environment of project (O)
	P18: Monitor progress of the set requirements goals (O)

Key: **O** = Organisational process **T** = Technical process
Allocated requirements = The agreement with the customer of the requirements for the software project (Paulk *et al*., 1995)

Figure 1: The motivation behind including requirements activities in the R-CMM at Level 2

1. The processes represent general requirements engineering best practice suggested by the CMM (Paulk *et al*., 1995) and reflect implicit practitioner needs.
2. The processes address problems raised by practitioners in our empirical research (Hall *et al.*, 2002; Beecham *et al.*, 2003b) and represent explicit practitioner needs.
3. The processes reflect recommendations in the requirements engineering literature, e.g. (Davis, 1995; El Emam and Madhavji, 1995a; Sawyer *et al*., 1997; IEEE, 1998) and reflect implicit practitioner needs.

A process can belong to more than one category, however categorisation of processes in Figure 1 reflects the 'primary' motivation for including them at a stage 2 capability level.

1. The CMM

The CMM motivated category (1) covers general 'organisational' activities within the requirements process. From our empirical study we conclude that organisational issues are causing practitioners more problems than technical issues, this is also a finding in (Lubars *et al*., 1993). We believe the CMM is therefore placing the right emphasis on 'managing' the requirements process, yet requires enhancement in the areas presented in our empirical research and the literature.

2. Empirical Research

Problems raised in our empirical research are viewed in two categories, 1) Organisational requirements problems and 2) Technical requirements problems.

Organisational Issues:

Training	Recognising and meeting training needs both in technical and organisational areas
Skills management	Ensuring a good spread and appropriate level of expertise is available to prevent over-dependence on few experienced staff. Sharing of best practice
Staff retention	Incorporating recruitment and workforce stability. Recruiting staff of the right level and retaining experienced staff.
User Communication	Communicating with outside users (e.g. company structure should not dictate who should discuss customer requirement needs with the customer – to preclude software designers).
Developer Communication	Communication between staff groups within the Company. E.g. Marketing discussing customer needs and agreements with Software Group, or Requirements Engineers communicating feasibility of design with Software group.
Resources	This relates to time, costs, investment in tools and people. Timescales and estimates given at beginning of project to be managed with allocation of adequate resources (staff time/training/costs of new tools) to include long-term software improvement activities.

Technical issues:

Requirements Growth	functional and non-functional requirements not documented in original specification that resu in changes over time, incorporates changebility decay (Arisholm and Sjoberg, 2000)
Vague requirements	Requirement documentation is incomplete and flawed. Also called requirements uncertaint (Moynihan, 2000)
Requirements Process	A well defined requirements process leads to a flexible system that is quick to respond t change (e.g. links to resources, traceability, and is cohesive).
Poor Use Understanding	User understanding of their own needs is often confused and undetected until too late – a customer will often ask for functions that are not needed and prove difficult to implement.
requirements traceability	**A link or definable relationship between entities** (Watkins and Neal, 1994) **"You can manage what you can't trace"** With low traceability work can be lost and it is often difficult to share work across teams.
Complexity o application	Large-scale projects can span many years and sites: can be highly complex, may need to b highly reliable, safety critical and customized.

For full descriptions please see Hall et al, 2002 and Beecham et al, 2003. For how each of the 12 companies in our empirical study report these problems please see Appendix 2, Tables 1 and 2.

3. Literature

The literature is rich in suggestions for software improvement. Recommendations, although often founded on empirical studies, can conflict with each other. We have been careful not to base the process inclusion solely on best practices taken from the literature. We use the literature to support our own findings and support the best practices within the CMM. "The body of knowledge and supporting licensing programs produced by groups like the IEEE are not supported by qualitative research into the actual practices of successful software organizations" (Bach, 1999). Bach states that the work of software best practice books such as these are often based on 'the passions of people who write the textbooks or run huge stuffy companies'. Bach asks that this knowledge be 'opened up'. (Bach, 1999). We endeavour to do so by combining empirical research with this body of knowledge.

Appendix 2: Contingency Tables giving Requirements Problems in 12 Software Development Companies by CMM Level

Table 1: Requirements Organisational Issues

REQUIREMENTS ORG ISSUES (12 companies, 45 focus groups)	Level 1 6 co's	Level 2 2 co's	Level 3 3 co's	Level 4 1 co.	Total
1. Culture/procedures	4	3	1	10	18
2. Developer communication	31	6	17	1	55
3. Resources	26	3	5	0	34
4. Skills and Responsibilities	29	6	9	2	46
5. Staff retention/ recruitment	21	6	1	1	29
6. Training needs not met	15	3	1	1	20
7. User Communication	13	5	12	0	30
Total	139	32	46	15	232

Table 2: Requirements Technical Issues

REQUIREMENTS TECHNICAL ISSUES (12 companies, 45 focus groups)	Level 1 6 co's	level 2 2 co's	level 3 3 co's	level 4 1 co.	Total
8. Complexity of application	8	8	11	0	27
9. Inadequate requirements traceability	4	0	0	0	4
10. Poor user understanding	2	1	2	0	5
11. Requirements growth	14	7	9	1	31
12. Undefined requirements process	21	6	5	0	32
13. vague initial requirements	24	5	4	0	33
Total	73	27	31	1	132

Appendix 3: Acronyms and Glossary of Terms

Acronym	
CMM	Capability Maturity Model
EPSRC	Engineering and Physical Sciences Research Council
KPA	Key Process Area
PPP Project	'Managing Practitioner impact on Processes and Products' (PPP) Project. A project funded by the UK's Engineering and Physical Science Research Council under grant number EPSRC GRL91962.
RE	Requirements Engineering
RM	Requirements Management
SEI	Software Engineering Institute
SPI	Software Process Improvement

TERM	TYPE	DESCRIPTION
Customer		The person, or persons who pay for the product and usually (but not necessarily) decide the requirements. In the context of this and the IEEE (1998) recommended practice the customer and the supplier may be members of the same organization. Individual, group, organisation that commissions the development of the system (Loucopoulos and Karakostas, 1995)
Engineering	Requirements	Deals with activities which attempt to understand the exact needs of the users of the software intensive system and to translate such needs into precise and unambiguous statements which will subsequently be used in the development of the system (established as a separate field of investigation and practice in mid 1970s) (Loucopoulos and Karakostas, 1995)
Engineering	Software	technical, managerial activities carried out in the production of software (Madhavji, 1991). To include: determination and specification of system and software requirements; analysis and management of risk; software prototyping; design, implementation; verification and validation; software quality control and assurance; integration of components; documentation; management of software configurations and versions, management of data, evolution of software; project management; software evaluation; software contracting; software acquisition; commissioning and decommissioning of software. Or An engineering discipline that applies sound scientific, mathematical, management, and engineering principles to the successful building of large computer programs (software) (Dorfman and Thayer, 1997)
Engineering	Systems	an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem…(International Council on Engineering Systems (INCOSE 1999) in (Leffingwell and Widrig, 2000) p.58
Life cycle	Software	The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. These phases may overlap or be performed iteratively. (IEEE, 1999)
Life cycle	System	The period of time that begins when a system is conceived and ends when the system is no longer available for use (IEEE, 1999)

PPP Project		'Managing Practitioner impact on Processes and Products' (PPP) project. A project funded by the UK's Engineering and Physical Science Research Council under grant number EPSRC GRL91962.
Practitioner		People actively involved in producing software, to include developers, project managers and senior managers.
Practitioner Communication		Communication between staff groups within the Company. E.g. Marketing discussing customer needs and agreements with Software Group, or Requirements Engineers communicating feasibility of design wth Software group.
Process		A collection of activities with entity flows among them (Yu and Mylopoulos, 1997)
Process	Tailoring or Customising	"for any process model to be effective in the specific project in hand, there is a need to customise the model according to the project goals. This may be achieved by characterising various aspects of the project (e.g. resource constraints); setting up project goals; assessing how these goals are supported by the adopted process model, tailoring the process model to suit project goals; using the tailored process model in the project; assessing and fine-tuning the model on an on-going basis. The customisation process would be simplified considerably if process models were organised hierarchically, leading from generic models at the top of the hierarchy to specific models at the bottom."(Madhavji, 1991) (the cmm does this to an extent).
Requirements	Allocated (as in CMM)	The agreement with the customer of the requirements for the software project (Paulk *et al*., 1995)
Requirements		1. A condition or capability needed by a user to solve a problem or achieve an objective 2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. 3. A documented representation of a condition or capability as in (1) and (2) (IEEE, 1999)
Requirements	Market Driven	Requirements are sketchy and informal Use of techniques from manufacturers rather than software engineering Specification is in the form of a marketing presentation No readily identifiable 'customer' developers tend to have less experience in application domain Projects rely on consultants for advice on desirable features Less structured approaches adopted. Task force used in 'brainstorming' sessions. (Loucopoulos and Karakostas, 1995)
Requirements	Analysis	The process of studying user needs to arrive at a definition of system, hardw are, or software requirements. The process of studying and refining system, hardware, or software requirements (IEEE, 1999)
Requirements	Errors	2 classes according to (Davis *et al*., 1993) 1. Knowledge errors: caused by not knowing what the true requirements are 2. Specification errors: caused by not knowing how to adequately specify requirements.
Requirements	Functional	A requirement that specifies a function that a system or sytem component must be able to perform. (IEEE, 1999). What the system should do (Sommerville and Sawyer, 1997) e.g. it should generate membership numbers for each person joining club etc.
Requirements	Growth/ change	functional and non-functional requirements not documented in original specification that result in changes over time, incorporates changeability decay (Arisholm and Sjoberg, 2000)

Requirements	Non-functional	Systems quantities or quality attributes. E.g. safety, security, reliability, usability, maintainability, cost and development time (Gross and Yu, 2001). High level non-functional requirements often decompose into functional requirements (Sommerville and Sawyer, 1997) they are not specifically concerned with the functionality of a system, placing restrictions of the product being developed and the development process.
Requirements	Organisational Issues (PPP Project)	Practitioner communication; Resources; Skills; Staff retention; Training; User communication.
Requirements	Phase	The period of time in the software life cycle during which the requirements for a software product are defined and documented
Requirements	Poor user understanding	User understanding of their own needs is often confused and undetected until too late – a customer will often ask for functions that are not needed and prove difficult to implement.
Requirements	Process definition	Processes used throughout the requirements phase of software development are defined to include all requirements phases, e.g. elicitation, analysis, documentation and validation as well as links to resources, traceability and general requirements management.
Requirements	Technical Issues (In PPP Project)	Requirements Growth/change; Vague/ambiguous requirements; Requirements Process definition; Poor user understanding; Requirements traceability
Requirements	Qualities	Quality attributes in the requirements process (from (Davis *et al.*, 1993)): Achievable; Annotated by Relative Stability; Annotated by Version; Annoted by Relative Importance; At Right Level of Detail; Complete; Concise; Correct; Cross-Referenced; Design Independent; Electronically Stored; Externally Consistent; Executable/Interpretable; Internally consistent; Modifiable; Not Redundant; Organized; Precise; Reusable; Traceable; Traced; Unambiguous; Understandable; Verifiable
Requirements	Review	A process or meeting during which the prequirements for a system hardware item, or software item are presented to project personnel, managers, users, customers, orother interested parties for comment or approval. Types include system requirements review, software requirements review. (IEEE, 1999)
Requirements	Specification	A document that specifies the requirements for a system or component. Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards (IEEE, 1999)
Requirements	Traceability	A link or definable relationship between entities (Watkins and Neal, 1994) that relates primarily to the requirements stage of software development.
Requirements	Vague/ ambiguous	Requirement documentation is incomplete and flawed. Also called requirements uncertainty (Moynihan, 2000)
Requirements Engineering		Is the science and discipline concerned with analysing and documenting requirements, including needs analysis, requirements analysis, and requirements specification. It also provides the appropriate mechanisms to facilitate the analysis, documentation, and verification activities. Requirements engineering can also be defined as a combination of requirements analysis and the documentation of the requirements into a form called requirements specifications (Chapter 1:Thayer, 1990)
Resources		This relates to time, costs, investment in tools and people. Timescales and estimates given at beginning of project to be managed with allocation of adequate resources (staff time/training/costs of new tools) to include long-term software improvement activities.

Skills		Ensuring a good spread and appropriate level of expertise is available to prevent over-dependence on few experienced staff. Sharing of best practice
Software Process Improvement	(SPI)	SPI stands for a collection of paradigms, methodologies, and technologies which help to make electronic and software firms more competitive, productive, defined, and quality oriented. SPI is a well elaborated field with industrial applications and experiences since the end of the eighties across the world. http://www.iscn.at/select_newspaper/select_index.html (Rogoway, 1998)
Software Requirements Review	(SRR)	A review of the requirements specified for one or more software configuration items to evaluate their responsivelness to and interpretation of the system rquirements and to determine whether they form a satisfacoty basis for proceeding into preliminary design of the configuration items.(IEEE, 1999)
Software requirements specification	(SRS)	Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces (IEEE Std 1012-1986 [12]) (IEEE, 1999) A document that describes all the externally observable behaviours and characteristics expected of a software system (Davis *et al*., 1993).
Specification		A document that specifies, in a complete, precise, verifiable manner, the requirements, design, behaviour, or other characteristics of a system or component, and often, the procedures for determining whether these provisions have been satisfied. (IEEE, 1999)
Staff Retention		Incorporates recruitment and workforce stability. Recruiting staff of the right level and retaining experienced staff.
Stakeholders		All practitioners and customers, and users – all people affected by the system with direct or indirect influence on the system requirements (Sommerville and Sawyer, 1997)
Supplier		The person, or persons, who produce a product for a customer. In the context of this and the IEEE (1998) recommended practice, the customer and the supplier may be members of the same organization.
Supplier of System		System developer or service provide who delivers a solution to meet the expected level of functionality and ensure successful integration of the technical system in the organizational setting. (Loucopoulos and Karakostas, 1995)
System		A collection of components organized to accomplish a specific function or set of functions (IEEE, 1999)
System Requirements Engineering		Is the science and discipline concerned with analysing and documenting system requirements. It involves transforming an operational need into a system description, system performance parameters, and a system configuration through the use of an iterative process of analysis, design, trade-off studies and prototyping Chapter 1: (Thayer and Dorfman, 1990)
Training		Training needs both in technical and organisational areas
User		The individual, group or organization that will work with the system itself (Loucopoulos and Karakostas, 1995)
User Communication		Supplier communication with users (e.g. how company structure dictates who discusses customer requirement needs with the customer and user).

Appendix 4: Defining Requirements Processes at Level 2 maturity

" Make the simplest and most distinctly observable phenomena form the soundest basis for description. More complex concepts can be build from them by appropriate use of definition" (Jackson, 1995b).

P1: Follow a written organisational policy for managing the system requirements allocated to the software project

This process is taken directly from the SW CMM: Requirements Management, Key Process Area, Commitment to Perform, Commitment 1 – (Paulk, 1995).
Literature in support of this process includes, e.g. (Sommerville and Sawyer, 1997) p.223; (Cugola and Ghezzi, 1998); (Sawyer *et al*., 1997 4.4); (Pfleeger and Rombach, 1994);(Fayad, 1997); (Christie, 1999)

This process is broken down as follows

- Each <requirements> activity is performed "according to a documented procedure". (CMM Template, Activity 2, Paulk, 1995, p.45)
- The written policy will define processes in requirements activities (CMM SPP Activity 5 (Paulk *et al.*, 1995).
- The written policy will document process goals.
- The written policy will serve to include people who have a central role in performing the activities needed to accomplish the process goals. The definition must reflect and support the need for "co-operation among people" and "must be highly flexible" (Cugola and Ghezzi, 1998) (Sawyer *et al*., 1997 4.4).

Sommerville and Sawyer (Sommerville and Sawyer, 1997) recommend that the management of allocated requirements should include the following policies:

1. a set of objectives for [the requirements management] process and rationale associated with each of these objectives
2. the reports to make the requirements engineering process visible and the activities which are expected to produce these reports as deliverables
3. the standards for requirements documents and requirements descriptions which should be used
4. change management and control policies for requirements
5. requirements review and validation policies
6. relationships between requirements management and other system engineering and project planning activities
7. traceability policies which define what information on dependencies between requirements should be maintained and how this information should be used and managed.
8. Criteria when these policies can be ignored; in these situations, managers use their own judgement on how to implement a requirements change.

Sommerville and Sawyer (1997) place this management activity in their list of basic guidelines for their Level 2 companies in their requirements engineering good practice guide, stating:

> *"Requirements management policies define goals for requirements management, the procedures which should be followed and the standards which should be used. These policies should be explicitly defined as part of your quality management system. … Explicit policies tell people involved in the process what they are expected to do and why it should be done… Projects generally manage their requirements in comparable ways, so with explicit policies, there is less dependence on individual knowledge and expertise.*
>
> *In order to define policies, you must understand your existing processes for requirements management. This is likely to reveal problem areas which may become the focus of process improvements.(Sommerville and Sawyer, 1997) p.223 .*

P2: Establish project responsibility for analysing the system requirements and allocating them to hardware, software, and other system components.

This process is taken directly from the SW CMM (Paulk *et al.*, 1995) Requirements Management (RM) Key Process Area (KPA) Ability 1. The CMM emphasises within each KPA the need to establish responsibility for project tasks, e.g. RM, Ability to Perform 1. "Analysis and allocation of the system requirements is not the responsibility of the software engineering group but is a prerequisite for their work".

This is also a main section in (McFeeley, 1996) p.98, section 3.8 "Finalize Roles and Responsibilities of the Various Infrastructure Entities.

P3: Implement training programme to recognise and meet technical and organisational requirements needs within the project.

"A lack of training.. led to teams that were less familiar with the RE process (Hofmann and Lehner, 2001)

The training programme should provide a platform for explaining why the organization is spending time and effort on a Requirements Process Improvement program. As practitioners' understanding grows so will their support. They must be motivated to join in the effort and assist it. The motivation should address the following points:
• Why change?
• What's wrong with the status quo?
• Why should I care?
• When will I be affected (immediately or sometime in the future)?
For further information on motivating practitioners in software process improvements efforts see (McFeeley, 1996) section 3.6; and (Baddoo and Hall, 2002b).

The ami guide also emphasises the need for 'properly administered' training, stating that "an assessment of the different needs and levels of training has to be made" (ami, 1992).

In his section on the Team Software Process (TSP), Humphrey (Humphrey, 2002) states that "the biggest single problem with the TSP is training. With few exceptions, mangers want the benefits .. but are reluctant to invest in the required training. [Using the improvement method] with untrained or partially trained teams .. have always failed". Humphrey recommends that organizations implement his improvement program properly or not even try it. This could be applied to the CMM too.

P4: Establish process to identify stakeholders within the requirements phase of the project

Stakeholder identification is not explicitly modelled in the Software-CMM, yet it is one of the most critical processes in terms of practitioner feedback and problems cited in the literature.

(Paulk *et al*., 1995) explain how the CMM addresses the customer

> 'The CMM is written from a software perspective. It covers the software process and addresses only those requirements allocated to software. It does not cover the processes of the customer or the system engineering group. It does describe inter-group interfaces that the software engineering group should proactively address, hopefully in a spirit of teamwork and an effective customer-supplier relationship." pp 53-54.

We see from this description of the CMM that inter group processes involving customers (and users) and the system engineering group are implicit rather than explicit.
Our empirical research details developer communication and user communication problems as accounting for 24% and 12% (total 36%) of organisational-based requirements problems (Hall *et al*., 2002). We interpret this as the stakeholder (to include customer and system engineering group) process being poorly defined and implemented.

Stakeholder identification is also central to Sommerville and Sawyer's (1997) Practical Process Improvement Guidelines: "The stakeholders in a system should always be explicitly identified in the requirements document and if appropriate information should be maintained which links specific requirements to the stakeholder who proposed these requirements" P.73.

A survey carried out by Barry Boehm and his team in their 'EasyWinWin' project asked practitioners the question "What are your major concerns with your organization's typical requirements approach? 5 concerns were mentioned, of which "Key stakeholders are excluded" was a major concern (Boehm, 2001). The Standish Group's Chaos report (StandishGroup, 1995) also identified "lack of user input" as contributing to 12.8% of project failure. And, lastly, Dorfman in (Thayer and Dorfman, 1990) states that good requirements include an "agreement among developers, customers, and users on the job to be done and the acceptance criteria for the delivered system" p.4.
Further literature in support of identifying stakeholders include:(Hofmann and Lehner, 2001)

*C*haracterizing the stakeholders may be the most difficult of the 4 areas, as the focus is on people, rather than products or documented processes.However, this is a critical area to the success or
failure of a product line. For example, the Generalized Support Software (GSS) project at GSFC met all its stated goals, yet has fallen into disuse. One of the key reasons for this is that one set of stakeholders, the flight dynamics analysts who would develop mission requirements using core assets, were not sufficiently considered by the GSS development team and their management [4]. **The first issue to be addressed is to identify *all* the stakeholders**. In the case of GSS, the analysts were known to be important, but in principle it is possible to omit a stakeholder (Stark, 2003).

P5: Provide adequate resources and funding for managing the allocated requirements in the project

This process forms a part of the SW CMM that demonstrates an 'ability' to perform the requirements activities: (Paulk et al, 1995), Requirements Management Key process area (Commitment 1; Ability 3).

The requirements process is a microcosm of the software process and as such organisations need to "Launch the [SPI] program by building an understanding and an awareness of the costs and benefits" and "Commit the resources necessary" (McFeeley, 1996).

Not only does the requirements process need resources to perform the activities, part of its activities is to provide "A good basis for resource estimation (cost, personnel quality and skills, equipment and time) Dorfman, in (Thayer and Dorfman, 1990).

P6: Establish process to identify skills needs within the project (for example, the skills required in requirements elicitation)

This requires matching the needs of project to the skills of personnel (Hofmann and Lehner, 2001) It is not a process found in the SW CMM, but is included in the PEOPLE CMM Level 2: Skills (Curtis *et al*., 1995).
There is a general discussion on personnel and the sensitive issue as to how to rate personnel capability and personnel experience in (Boehm, 1981)

(El Emam and Madhavji, 1995a) in their field study have a section on skills sets in their field study. They recommend that appropriately skilled people be assigned to analyst and architect positions, as well as skilled users in the requirements process especially the principal user - project managers should also have a high capability in the requirements engineering phase.

There are ways to directly characterize stakeholders that are currently used in empirical software engineering. One common characterization is a set of ordinal scales measuring experience: total software development experience, experience in a role, or experience with a particular organization. However, there are some more complex questions to answer. One is how does one characterize the interactions between stakeholder roles? Does the current organizational structure support the interactions that are needed to build and exploit a product line?
Another issue is to assess the gap between current stakeholder roles and what would be necessary to succeed in a product line environment. To do this would require substantial understanding of how stakeholders carry out their work product development.
All of these questions are areas where empirical research may provide very practical answers.
(Stark, 2003)

P7: Institute process to maintain stability within project, e.g. cope with changes in staff/ requirements priorities/general priorities in organising the requirements process.

"A disciplined software engineering process helps address many 'accidental' difficulties" S. Faulk, Software requirements: A Tutorial" in (Dorfman and Thayer, 1997) "To achieve a stable project over a long period of time, a manager must encourage the project to function .. with a fresh supply of trainees coming one end and a stream of experienced leaders coming out of the other" … and "A project is not a house of cards which collapses when a single key person is removed .. when management thinks it is, the prophecy becomes self-fulfilling" "If a [practitioner] is indispensable, get rid of him as quickly as possible"!! all quotes from Chapter "Stability through change", in (Weinberg, 1998).
Recognise and anticipate volatile requirements: e.g. mutable requirements; emergent requirements; consequential requirements and compatibility requirements (see (Kotonya and Sommerville, 1998) p.116).
Successful RE teams manage requirements priorities "To specify prioritized requirements, the RE team develops various models together with prototypes" (Hofmann and Lehner, 2001)
McFeeley has a section dedicated to prioritizing activities and developing an improvement agenda (McFeeley, 1996).
"The baselines, particularly the maturity baseline, typically identify issues and provide recommendations based on a much broader consensus than may have been available before.
These issues and recommendations serve to provide some guidance, and often, a prioritization of actions."
(A Level 2 organisation should be in a position to identify where their priorities lie as they must have their baseline maturity processes in place).
Another guide to creating a stable environment is found in (McFeeley, 1996), where McFeeley advocates that organisations "Establish Software Process Improvement Infrastructure"
in order to "build the mechanisms necessary to help the organization institutionalize continuous process improvement. …. A solid, effective infrastructure can sustain a developing [SPI] program until it begins to produce visible results. Unsupported [SPI] programs can become isolated and die out during periods of stress and tension within their organizations…. To effectively manage the SPI program, an infrastructure must be in place or created."

P8: Explore alternative solutions, requirements techniques and tools for the project

"Several methods and languages can be used for specifying the functionality of computer systems. No single language, of those now available, is equally appropriate for all methods, application domains, and aspects of a system. Thus users of formal specification techniques need to

understand the strength and weaknesses of different methods and languages before deciding on which to adopt.
A review of formal methods: Robert Vienneau, in (Dorfman and Thayer, 1997)

"We expect methods to be panaceas – medicines that cure all diseases. This cannot be."(Jackson, 1995a) Classifying problems and relating them to suitable methods is a central theme of Jackson's (1995) book. There is not a one size fits all technique, and in a study of three different projects (Lauesen and Vinter, 2001) conclude "the value of a technique depends on the project".

In a study of management of process improvement by prescription (Middleton and McCollum, 2001) conclude that "the idea of a 'best' method is misleading because of the diverse range of projects and developers". The generic lesson gleaned for their research is that an organization is "probably unwise to use a heavily prescriptive methodology to improve its software development performance" (Middleton and McCollum, 2001).

In the documentation stage it may be necessary to use well defined semantics, such as deterministic finite state machines, Petri nets, decision trees, propositional calculus, predicate calculus to avoid ambiguity… the choice will be driven primarily by expressive power and suitability for the aspect of the system." (Davis *et al*., 1993). However, Davis does admit that replacing natural language with formal notations greatly decreases ambiguity in the SRS but almost always at the expense of understandability (except for decision trees). He therefore suggests augmenting natural language with more formal models.

Other recommendations include:

Requirements should be 'explored' through methods such as: brainstorming, simulation, visualization, storyboard illustrations and scenarios (Maiden and Gizikis, 2001).
Measurement techniques are used to help explore and understand the size of the product and manage project constraints such as duration, time-to-market and productivity, along with customer satisfaction factors e.g. MkII Function Point Analysis (Rule, 2001) – Function point analysis is used to measure productivity of system development and system maintenance, and can also be used for project estimating by converting function points into work-effort (Onvlee, 1995).

In a case study by Kitchenham (Kitchenham, 1995) function points are said to be flawed - don't give accurate predictions of effort, are over-complex as metrics and are unsuitable for cross-company comparisons – signalling that organisations must be cautious about the methods they use and the results they obtain. Yet, in another case study, function point analysis gives slightly better results for effort prediction than using the COCOMO model (Boehm, 1981) and (Stricker, 1995) states that their model (F-PROM) brings better results than either function point analysis or the COCOMO model.

The general message is, understand the technique you are using, acknowledge its strengths and weaknesses and assess whether there may be a better way of achieving your aims.

P9: Establish/maintain process to reach agreement with customer on requirements for project.

This process does not form part of the SW CMM Requirements Management Key Process Area activities, but is included in its definition (Paulk et al, 1995).

Agreement includes "Obtain Approval for [SPI Proposal] and Initial Resources" (McFeeley, 1996) section 1.5.

"…good requirements include .. Agreement among developers, customers, and users on the job to be done and the acceptance criteria for the delivered system." Dorfman in (Thayer and Dorfman, 1990)

P10: Establish/maintain process to involve key stakeholders within the project.

"Involving stakeholders early .. resulted in an increased understanding of the RE process being used" and "Requirements prioritized by stakeholders drive successful RE teams "(Hofmann and Lehner, 2001). There is a need to develop a trust and a shared vision of what the project is trying to achieve; "users are part of the system and therefore it is necessary that their capabilities are explicitly grown with the system…" (Middleton and McCollum, 2001). User contribution should include involvement, expression, participation and commitment (Middleton and McCollum, 2001).
(Cottengim, 2002)
(Paulk *et al.*, 1995) pp 53-54 indicate that there is nothing explicit in CMM. Yet heavily supported in our research (Hall *et al.*, 2002). And the literature, e.g. (Hofmann and Lehner, 2001) (Sommerville and Sawyer, 1997) p.73; (Boehm, 2001);(StandishGroup, 1995) (Thayer and Dorfman, 1990)

"Users should always participate in the requirements engineering process" (El Emam and Madhavji, 1995a)

P11: Set realistic improvement goals to address problems in the requirements process project

The process of setting realistic goals is important for

1) modelling the right level of 'project' improvement goals for the requirements phase to solve recognised problems, and
2) in setting functional and non-functional 'requirements'.

"When there is a perception that the requirements are unrealistic, software developers may become discouraged and not fully commit tot he goals of the project" (Linberg, 1999)

"Determine Key Business Issues Purpose: Unless the SPI program is driven by the current business needs and understood and agreed to by management, it will likely be difficult to sustain the program over the long haul. This is because it will be difficult to clearly demonstrate to senior management that the initiative is achieving real value for the organization in business terms" (McFeeley, 1996).

"For any process model to be effective in the specific project in hand, there is a need to customise the model according to the project goals. This may be achieved by characterising various aspects of the project (e.g. resource constraints); setting up project goals; assessing how these goals are supported by the adopted process model, tailoring the process model to suit project goals; using the tailored process model in the project; assessing and fine-tuning the model on an on-going basis.

"The customisation process would be simplified considerably if process models were organised hierarchically, leading from generic models at the top of the hierarchy to specific models at the bottom."(Madhavji, 1991) (the CMM does this to an extent).

"[Measurement] helps in making intelligent decisions and improving over time. But measurement must be focused, based upon goals and models" (Basili, 1995)

"To improve their software development, organisations need a definition of clear improvement goals, otherwise the improvement activities will turn out to be as chaotic as the development process itself. These improvement goals should support business objectives in the best possible way. For example, it is not recommended to base improvement on a method that prescribes the installation a software configuration management system, while most projects in the organisation fail because of bad requirements management" (Solingen and Berghout, 1999). Setting realistic goals means recognising and prioritising which processes need strengthening.

All identified key stakeholders should be involved in the definition of measurement goals. I.e. project team members involved in requirements, their manager and the improvement team members.

Goals should include
- The purpose (what object and why)
- The perspective (what aspect and who)
- The context characteristics.

P12: Establish/implement a process to assess feasibility & external environment relating to project (Sommerville and Sawyer, 1997)

The CMM states that assessing the feasibility of a project should include risk assessment, e.g. : "Software risks associated with cost, resource, schedule, and technical aspects of the project are tracked." SPP Activity 13, SPTO, Activity 10.

This process includes the need to define system boundaries as in (Sommerville & Sawyer, 1997). (Curtis *et al*., 1988) found that accurate problem domain knowledge is critical to the success of the projects.
Analysts may need to steer the client away from requirements that cannot be met within the budget and schedule constraints P, Coad and E Yourdon, "Object-Oriented Analysis" in (Thayer and Dorfman, 1990).

Patel advocates the use of object oriented technology that can allow both global and local aspects of requirements to be captured i.e. regional (local use cases and commonalities local environments which require analysis) (Patel, 1999)

P13: Establish/maintain repeatable requirement traceability process that is project-based

Establishing and maintaining requirements traceability is a central theme in the CMM, yet it is not explicitly modelled. The traceability activities evident in the CMM include the Configuration Management KPA which is specially focussed on tracking requirements. For example, "Software Configuration Management involves identifying the configuration of the software (i.e. selected software work products and their descriptions) as given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle." CMM section 7.6 Software Configuration Management, a key process area for Level 2.

"Inadequate requirements traceability" was cited as a (albeit minor) problem in our process-based requirements research (Hall *et al*., 2002). A strong requirements traceability process may aid other requirements problems cited such as controlling requirements growth and will assist in requirements re-use, however it is important to use the correct traceability method. For example, requirements recycling is supported by methods that separate vertical, horizontal and evolutionary relationships between entities *(Knethen et al., 2002)*; If you have a legacy system Sutcliffe states that current methods do not address requirements in a legacy system context. He proposes a

model that can cope with the constraints legacy systems place on new requirements and addresses the need to integrate changes resulting from new requirements without introducing errors into acceptable parts of the existing system *(Sutcliffe et al., 1999)*.

"Another important concept in the CMM is traceability. Under the CMM all worthwhile software work products are documented, and the documentation design, code and test cases are traced to the source from which they were derived and to the products of the subsequent engineering activity. Requirements traceability provides a means of analysing impact before a change is made, as well as a way to determine what components are affected when processing a change. " Measurements in the CMM include:

Status of each allocated requirement throughout the lifecycle
Change activity of the allocated requirements
Allocated requirements summarized by category."

(Leffingwell and Widrig, 2000)

Traceability is understood to mean "a link or definable relationship between entities" (Watkins and Neal, 1994), who state that "You can't manage what you can't trace".

The IEEE define traceability as: " (1)The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or mother-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match.
(2) The degree to which each element in a software development product establishes its reason for existing; for example, the degree to which each element in a bubble chart references the requirement that it satisfies. (IEEE std 610.1-1990 in (IEEE, 1999)

"[The successful RE team] maintain a requirements traceability matrix to track a requirement from its origin through its specification to its implementation" (Hofmann and Lehner, 2001).

P14: Establish a repeatable process to manage complex requirements at project level

Large-scale projects can span many years and different sites can be highly complex. They may need to be highly reliable, safety critical and customized. "One of the pitfalls of systems engineering is to think that a system is simple (i.e. not complex) when we have a very good understanding of its (application) features. An example of such a system is a banking system visualised by the users as a set of automatic teller machines (ATMs). The functions of an ATM are extremely well understood; its applications are trivial transactions. From a system viewpoint, however, we have to worry about a system with a large database of sensitive information with hundreds to thousands of users. With this system come problems related to security and data base concurrency. Virtually all real-time systems are complex because of the constraints on both cycle time and memory resources(Shere, 1988).
According to Yourdon, a system is complex if most of the following features apply to the system:

10,000 </= SLOC </= 100,000 (Source lines of Code)
five to twenty programmers over a two to three year period
several subsystems
100 </= number of modules </= 1,000

(Yourdon, 1995)

P, Coad and E Yourdon, "Object-Oriented Analysis" in (Thayer and Dorfman, 1990). Object Oriented Analysis contains four major principles for managing complexity: abstraction, information hiding, inheritance and methods of organization.
Leffingwell recommends that complex systems entail requirements specification for each sub-system, and non-trivial applications, requirements must be captured and recorded in a document database, model or tool (Leffingwell and Widrig, 2000)

Techniques such as functional decomposition and input-output analysis reduce complex systems into manageable subsystems but may not help with complex organizational issues (Yu and Mylopoulos, 1997). The *i** framework may be helpful in identifying enterprise integration solutions for organisations that have complex technical and human organizational environments.(Yu and Mylopoulos, 1997)

P15: Establish a repeatable process to manage vague requirements at project level

The CMM steers companies away from vague requirements with activities such as: "The allocated requirements are reviewed to determine whether they are clearly and properly stated" RM, Activity 1.2

We define vague requirements as requirement documentation that is incomplete and flawed. Also called requirements uncertainty (Moynihan, 2000) (El Emam and Madhavji, 1995a). "The whole purpose of the requirements process is to reduce ambiguity in the development process" (Gause and Weinberg, 1989).

El Emam & Madhavji talk about 'requirements uncertainty' and define it as "the difference between the amount of knowledge that is required and that is available about the problem and solution domains". "The greater the uncertainty the greater the amount of changes to the requirements engineering documentation (El Emam and Madhavji, 1995a).

Davis lists 'unambiguous' requirements specified in the software requirements specification (SRS) on the top of his requirements quality list, and states "an SRS is unambiguous if and only if every requirement stated therein has only one possible interpretation (Davis *et al*., 1993). Davis dedicates a section to unambiguous and complete requirements and suggests ways these may be measured and controlled.

P16: Establish a repeatable process to manage requirements growth/change at project level

Concerns functional and non-functional requirements not documented in original specification that result in changes over time, incorporates changeability decay (Arisholm and Sjoberg, 2000) "Change is inevitable when computer software is built. And change increases the level of confusion among software engineers who are working on a project. Confusion arises when changes are not analysed before they are made, recorded before they are implemented, reported to those who should be aware that they have occurred, or controlled in a manner that will improve quality and reduce error". Software Engineering, R Pressman, p 66 in (Dorfman and Thayer, 1997). "A primary goal of software engineering is to improve the ease with which changes can be accommodated and reduce the amount of effort expended when changes must be made." *Sic*
The CMM covers this extensively, to include:
"Changes to the allocated requirements are reviewed and incorporated into the software project.

1. The impact to existing commitments is assessed, and changes are negotiated as appropriate.

- Changes to commitments made to individuals and groups external to the organization are reviewed with senior management. (Activity 4 Software Project Planning KPA and Activity 3 of Software Project Tracking and Oversight kpa for practices cover commitments made external to the organisation.)
- Changes to commitments within the organization are negotiated with the affected groups. (Software Project Tracking and Oversight KPA for practices covers negotiating changes to commitments.)"

"The CMM recognizes that change is an integral part of software activity in any development project. In place of frozen specifications we instead strive for a stable baseline of requirements

that are well elicited, documented and placed into systems that provide support for managing change. Specifically the CMM requires that as understanding of the software improves, changes to the software work products and activities are proposed, analyzed and incorporated as appropriate. Where changes to requirements are needed, they are approved and incorporated before any work products or activities are changed" (Leffingwell and Widrig, 2000).

"Requirements continue to be in a state of flux…Many forces affect this ever-changing requirements e.g P, Coad and E Yourdon, "Object-Oriented Analysis" in (Thayer and Dorfman, 1990). : customers, competition, regulators, approver, and technology… We have to accept changing requirements as a fact of life, and not condemn them as a product of sloppy thinking" P, Coad and E Yourdon, "Object-Oriented Analysis" in (Thayer and Dorfman, 1990). Patel also advocates the use of object oriented technology in his spiral of change model (Patel, 1999).

P17: Establish a repeatable process to manage user understanding

Comprehension: People do not know what they want. This does not mean that people do not have a general idea of what the software is for. Rather, they do not begin with a precise and detailed understanding of what functions belong in the software, what the output must be for every possible input, how long each operation should take, how one decision will affect another, an so on…. It is a precise and richly detailed understanding of expected behaviour that is needed to create effective designs and develop correct code. (Faulk, S, "Software Requirements: A Tutorial" in (Dorfman and Thayer, 1997).

Laura Scharer, 1981, Pinpointing Requirements in (Thayer and Dorfman, 1990) explains that users have a different goals and approach to requirements than system analysts. She suggests that although users provide the system definition, the systems people are responsible for it, and that if the user understands their own needs definability is positively affected.

Managing uncertainty in requirements was identified as a major concern to practitioners in El Emam's field study (El Emam and Madhavji, 1995a) – recommendations as to how to help solve this problem include recognising the skill levels required in developers and users and assigning the necessary skills to the project.

P18: Monitor progress of the set requirements goals

Goals are a part of every key process activity in the CMM.
Business goals – having 'set' goals, goals need to be monitored. See P11 'set goals' for further references.
Solingen(Solingen and Berghout, 1999) suggests that goals are reviewed:
The goals should be reviewed and approved by a project team before data collection can actually begin. The review session should focus on:
Do project members agree upon the defined goals, questions and metrics?
Do project members identify any missing or unnecessary definitions?

P19: Agree and document technical and organisational attributes specific to project.

The inclusion of this process is primarily motivated by our empirical work (see Beecham et al 2003, and Hall et al 2002). A well defined requirements process leads to a flexible system that is quick to respond to change (e.g. links to resources, traceability, and is cohesive).

"To succeed you must integrate your technical, cognitive, social and organizational processes to suit your project's particular needs and characteristics" (Hofmann and Lehner, 2001). "One of the most common reasons systems fail is because the definition of system requirements is bad" Laura Scharer, Pinpointing Requirements in (Thayer and Dorfman, 1990).

The process and principles of defining and documenting processes are applied to each of the 5 requirements phases. For example, the documentation phase needs to "define a standard document structure; explain how to use the document, include a summary of the requirements; make a business case for the system; define specialised terms; lay out the document for readability; make document easy to change.

One project management method should be used project wide, e.g. waterfall, spiral, rapid and joint application development, eXtreme Programming (Rule,2001). The CMM also recommends that "A software life cycle with predefined stages of manageable size is identified or defined " in Software Project Planning, Activity 5 (Paulk et al, 1997).

Further references in support of this process: (Sommerville and Sawyer, 1997) p.223; (Cugola and Ghezzi, 1998) (Sawyer et al., 1997 4.4); (Pfleeger and Rombach, 1994); (Fayad, 1997); (Christie, 1999).

P20: Establish a process to review allocated requirements within the project to include software managers and other affected groups

This process is taken direction from the SW CMM (Paulk *et al.*, 1995). It is a CMM activity: RM: Activities Performed, Activity 1: The software engineering group reviews the allocated requirements before they are incorporated into the software project.

1. Incomplete and missing allocated requirements are identified
2. The allocated requirements are reviewed to determine whether they are:
   - Feasible
   - Clearly named properly stated
   - Consistent with each other
   - testable

"Successful teams repeatedly validate and verify requirements with multiple stakeholders. They use peer reviews, scenarios, and walk-throughs to improve the specification throughout the software's life cycle."(Hofmann and Lehner, 2001)

"People typically repeat past behaviors, including those that lead to success and those that do not. The organization must ensure that mistakes are not repeated that may have caused similar initiatives to fail in the past". (McFeeley, 1996) section 3.5 "Review Past Improvement Efforts".

According to Davis, a software requirements specification is verifiable if there exist finite, cost effective techniques that can be used to verify that every requirement stated therein is satisfied by the system as built. He states that some requirements are easy to test, whereas others may be difficult to verify – he lists reasons for requirements being difficult and suggests methods for controlling difficult requirements (Davis *et al*., 1993).

"Any engineering process requires feedback and evaluation. Software development is an engineering discipline and measurement is an ideal mechanism for feedback and evaluation.

The measurements and information fed back to developers, managers, customers and the [organisation] help in the understanding and control of the software processes and products and the relationships between them" (Basili, 1995).

Appendix 5: Five phases of requirements viewed from a G/Q/M paradigm

The R-CMM identifies the processes that address the following five phases of requirements:

Q1 How repeatable is your requirements management process?

Definition of the Management phase of requirements:
All requirements engineering phases and activities are planned and controlled (Dorfman and Thayer, 1997), "The CMM summarizes the process area of requirements management"… "the purpose of requirements management is to establish a common understanding between the customer and the software team of the customer's requirements"(Leffingwell and Widrig, 2000)

Q2 How repeatable is your elicitation process?

Definition of the Elicitation phase of requirements:
The system requirements are discovered through consultation with stakeholders, from system documents, domain knowledge and market studies. Other names for this process include 'requirements acquisition' and 'requirements discovery'. Guidelines in (Sommerville and Sawyer, 1997) include:
Assess System Feasibility. Be sensitive to organisational and Political Considerations. Identify and Consult System Stakeholders. Record Requirements Sources. Define the Systems Operation Environment. Use Business Concerns to Drive Requirements Elicitation. Look for Domain Constraints. Record Requirements Rationale. Collect Requirements From Multiple Viewpoints. Prototype Poorly Understood Requirements. Use Scenarios to Elicit Requirements. Define Operational Processes. Reuse Requirements.

Q3 How repeatable is your analysis and negotiation process?

Definition of the Analysis and Negotiation phase of requirements:
Requirements are analysed in detail and different stakeholders negotiate to decide on which requirements are to be accepted. This process is necessary because of inevitable conflicts between the requirements from different sources. This process focusses on incomplete and incompatible requirements (e.g. requirements may be incompatible with the budget available to develop the system). There is usually some flexibility in requirements and negotiation is necessary to decide on the set of agreed requirements for the system. Guidelines given in (Sommerville and Sawyer, 1997) *include: Define System Boundaries. Use Checklists for Requirements Analysis. Provide Software to Support Negotiations. Plan for Conflicts and Conflict Resolution. Prioritise Requirements. Classify Requirements Using a Multi-dimensional Approach. Use Interaction Matrices for Find Conflict and Overlaps. Assess Requirements Risks*

Q4 How repeatable is your documentation process?

Definition of the Documentation phase of requirements:
The agreed requirements are documented at an appropriate level of detail. In general, there needs to be a requirements document which is understood by all system stakeholders. This usually means that the requirements must be documented using natural language and diagrams. More detailed system documentation, such as system models may also be produced. Guidelines in (Sommerville and Sawyer, 1997) include:

(1)Describing Requirements: Define Standard Templates for Describing Requirements. Use Language Simply, Consistently and Concisely. Use Diagrams Appropriately. Supplement Natural Language with Other Descriptions of Requirements. Specify Requirements Quantitatively.

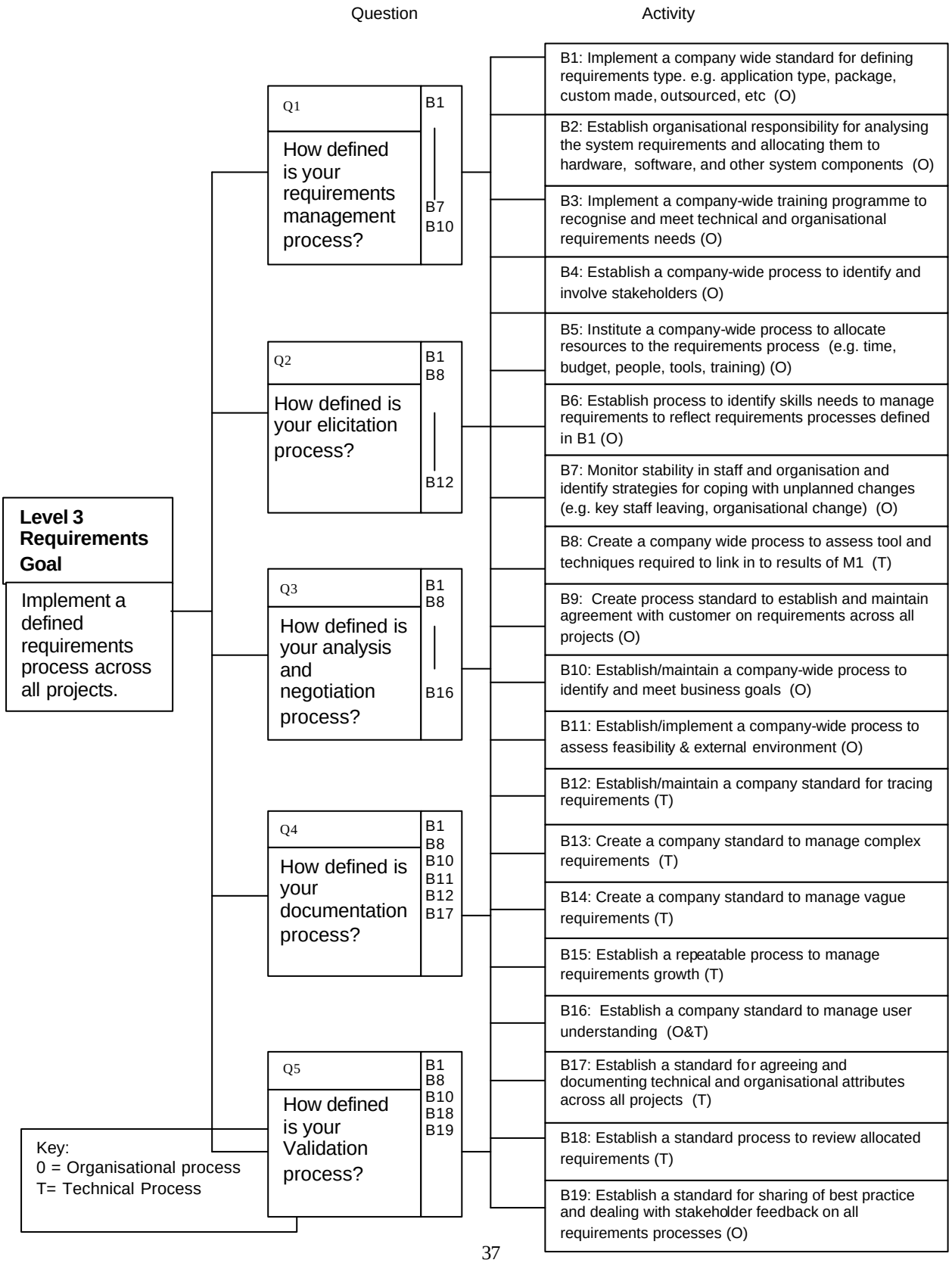
(2) System Modelling: Develop Complementary System Models. Model the System's Environment. Model the System Architecture. Use Structured Methods for System Modelling. Use a Data Dictionary. Document the links between Stakeholder requirements and System Models.

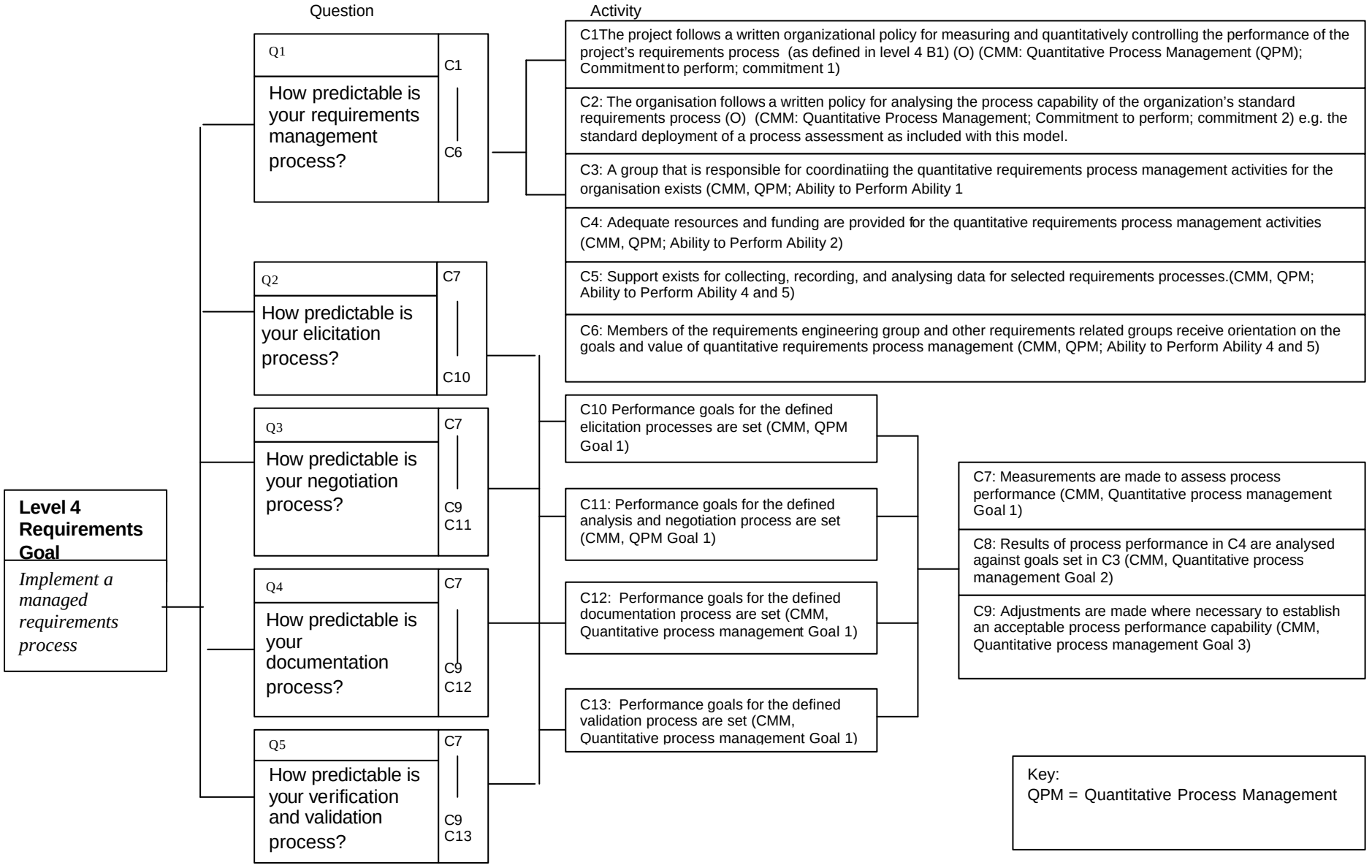
Q5 How repeatable is your validation process?

Definition of the Validation phase of requirements:
A requirements are checked for consistency and completeness. This process is intended to detect problems in the requirement document before it is used as a basis for the system development. Guidelines given in (Sommerville and Sawyer, 1997) include: Check that the requirements document meets your standards; Organize formal requirements Inspections. Use Multi-disciplinary Teams to Review Requirements. Define Validation Checklists. Use Prototyping to Animate Requirements. Write a draft user manual; propose requirements test cases. Paraphrase system models.

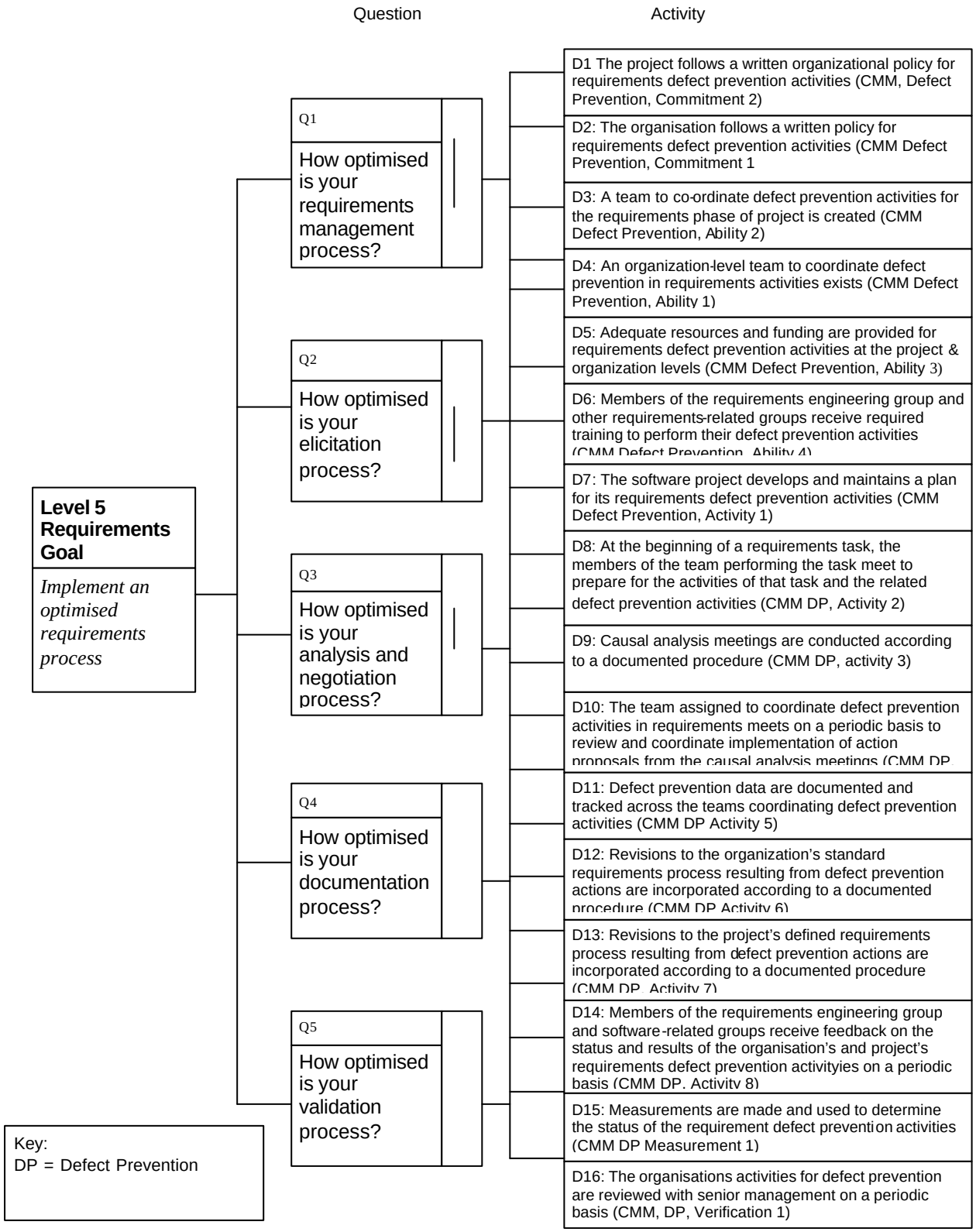
Appendix 6: R-CMM: Levels 3, 4 and 5

LEVEL 3 R-CMM




Question
Activity
Level 4 Requirements Goal
Implement a managed requirements process
Q1
How predictable is your requirements management process?
C1
C6
Q2
How predictable is your elicitation process?
C7
C10
Q3
How predictable is your negotiation process?
C7
C9
C11
Q4
How predictable is your documentation process?
C7
C9
C12
Q5
How predictable is your verification and validation process?
C7
C9
C13
C1The project follows a written organizational policy for measuring and quantitatively controlling the performance of the project's requirements process (as defined in level 4 B1) (O) (CMM: Quantitative Process Management (QPM); Commitment to perform; commitment 1)
C2: The organisation follows a written policy for analysing the process capability of the organization's standard requirements process (O) (CMM: Quantitative Process Management; Commitment to perform; commitment 2) e.g. the standard deployment of a process assessment as included with this model.
C3: A group that is responsible for coordinatiing the quantitative requirements process management activities for the organisation exists (CMM, QPM; Ability to Perform Ability 1
C4: Adequate resources and funding are provided for the quantitative requirements process management activities (CMM, QPM; Ability to Perform Ability 2)
C5: Support exists for collecting, recording, and analysing data for selected requirements processes.(CMM, QPM; Ability to Perform Ability 4 and 5)
C6: Members of the requirements engineering group and other requirements related groups receive orientation on the goals and value of quantitative requirements process management (CMM, QPM; Ability to Perform Ability 4 and 5)
C10 Performance goals for the defined elicitation processes are set (CMM, QPM Goal 1)
C11: Performance goals for the defined analysis and negotiation process are set (CMM, QPM Goal 1)
C12: Performance goals for the defined documentation process are set (CMM, Quantitative process management Goal 1)
C13: Performance goals for the defined validation process are set (CMM, Quantitative process management Goal 1)
C7: Measurements are made to assess process performance (CMM, Quantitative process management Goal 1)
C8: Results of process performance in C4 are analysed against goals set in C3 (CMM, Quantitative process management Goal 2)
C9: Adjustments are made where necessary to establish an acceptable process performance capability (CMM, Quantitative process management Goal 3)
Key:
QPM = Quantitative Process Management

Level 5 R-CMM


Question
Activity
Level 5 Requirements Goal
Implement an optimised requirements process
Q1
How optimised is your requirements management process?
Q2
How optimised is your elicitation process?
Q3
How optimised is your analysis and negotiation process?
Q4
How optimised is your documentation process?
Q5
How optimised is your validation process?
D1 The project follows a written organizational policy for requirements defect prevention activities (CMM, Defect Prevention, Commitment 2)
D2: The organisation follows a written policy for requirements defect prevention activities (CMM Defect Prevention, Commitment 1
D3: A team to co-ordinate defect prevention activities for the requirements phase of project is created (CMM Defect Prevention, Ability 2)
D4: An organization-level team to coordinate defect prevention in requirements activities exists (CMM Defect Prevention, Ability 1)
D5: Adequate resources and funding are provided for requirements defect prevention activities at the project & organization levels (CMM Defect Prevention, Ability 3)
D6: Members of the requirements engineering group and other requirements-related groups receive required training to perform their defect prevention activities (CMM Defect Prevention, Ability 4)
D7: The software project develops and maintains a plan for its requirements defect prevention activities (CMM Defect Prevention, Activity 1)
D8: At the beginning of a requirements task, the members of the team performing the task meet to prepare for the activities of that task and the related defect prevention activities (CMM DP, Activity 2)
D9: Causal analysis meetings are conducted according to a documented procedure (CMM DP, activity 3)
D10: The team assigned to coordinate defect prevention activities in requirements meets on a periodic basis to review and coordinate implementation of action proposals from the causal analysis meetings (CMM DP
D11: Defect prevention data are documented and tracked across the teams coordinating defect prevention activities (CMM DP Activity 5)
D12: Revisions to the organization's standard requirements process resulting from defect prevention actions are incorporated according to a documented procedure (CMM DP Activity 6)
D13: Revisions to the project's defined requirements process resulting from defect prevention actions are incorporated according to a documented procedure (CMM DP, Activity 7)
D14: Members of the requirements engineering group and software-related groups receive feedback on the status and results of the organisation's and project's requirements defect prevention activityies on a periodic basis (CMM DP, Activity 8)
D15: Measurements are made and used to determine the status of the requirement defect prevention activities (CMM DP Measurement 1)
D16: The organisations activities for defect prevention are reviewed with senior management on a periodic basis (CMM, DP, Verification 1)
Key:
DP = Defect Prevention

Appendix 7: Descriptive Modelling

We have not included tried and tested techniques for solving requirements problems in our model, e.g. the work of (Rule, 2001) (use cases and function points), (Sommerville and Sawyer, 1997) (view points) and (Leffingwell and Widrig, 2000) unified approach. These methods enter the prescriptive world of 'how' to solve specific requirements problems and therefore go against the ethos of the model that endeavours to be generic and universally applicable. In agreement with Osterweil 1987 in (Cugola and Ghezzi, 1998) we believe that all organisations are different,

> *"they differ in people, skills, products delivered, commercial and development strategies. Even within the same organization different projects present huge variations … As a consequence, there is no unique, ready-made software development process. The process must be defined based on the problem to be solved."*

This sentiment is echoed by Middleton and McCollum (2001), who point out that:

> *"The idea of 'best' method is misleading because of the diverse range of projects and developers. The generic lesson … is that an organization is probably unwise to use a heavily prescriptive methodology to improve its software development performance."*

Paulk et al, (1995) appreciate these differences and explain that the given practices need to be adapted to be useful:

> *"Intelligence, experience, and knowledge must shape an appropriate interpretation of the CMM in a specific environment".*

To remain consistent with this normative approach, the R CMM does not follow any particular lifecycle model and project management method (such as the waterfall, spiral, prototype, rapid and joint application Development, eXtreme programming). This has been identified as a weakness of the CMM in (Brodman and Johnson, 1994) who state that it favours the waterfall method and does not address prototyping. However, it is likely that whatever method is followed, an initial stage in development will involve deciding what the requirements are of the system.

Balancing abstractionism with context

The descriptive modelling technique used in this study abstracts phenomena from the context of the CMM and empirical data. (Potts, 1997) notes that if a model is based purely on abstraction it will have powerful properties such as the ability to generalize across contexts. However "Abstractionism provides standard methods, yet can also be an over-simplification of the problem domain with an overemphasis on normative cases". Potts adds that there are strengths to including context into the model as, "if the model is context specific it will fit in well with current practice and can be understood by end-users" (sic). But as (Cugola and Ghezzi, 1998) point out, moving away from abstract, normative models towards a context specific model involves following an expected sequence of activities. This limits flexibility and prohibits fast adaptation required in a dynamic marketplace.

Jackson (1995) states that generalisations are weak, "It is a principle of methodology that the power of a method is inversely proportional to its generality…. To be powerful, a method must exploit the problem's features very minutely, because problem features vary widely, we need a repertoire of methods, each suitable for problems of a particular class" (Jackson, 1995b)

Retaining a balance between abstraction and context

In the design stage of our model we endeavour to retain a focus on the context. (Potts, 1997) relates this need directly to the field of requirements engineering practice where "by abstracting away from the context of an

investigation, the designer too easily lapses into modeling only those things that are easy to model." The R-CMM will balance context with abstraction to ensure that the essential properties of process capability are captured. Although the model is an abstraction of key requirements practices, practitioners must take responsibility to ensure that "all requirements, particularly non-functional requirements, have been identified, are described correctly, and are fully detailed" (Middleton and McCollum, 2001)."A starting point for process improvement is to describe the current process as used in software development. The process model inherent in this description is called a 'descriptive' model. Describing a process means making the software process explicit. This involves modelling the actual software process, using an appropriate process modelling methodology." Madhavji (1991) continues "the central part of such a methodology that deals with the design of a process model needs to address the formalisms which may be used to represent process models. Several different formalisms have been proposed to address these needs. – It appears from this definition of descriptive modelling that Madhavji believes that descriptive modelling has some of the formal elements of the prescriptive modelling discussed in the later paper by (Cugola and Ghezzi, 1998). Madhavji continues with a list of different formalisms – those that apply to this report are:

Rules: a formalism for modelling the software process in terms of precondition, activity and postconditions, where the precondition must be true for a particular activity to be executed, and one of the postconditions becomes true after the activity terminates. (The rules are set out in criteria below). This definition of a rule based model suits process modelling particularly well as a process itself contains implicit pre and postconditions: "a process has inputs and outputs …"

Behavioural approach: a formalism for describing the abstractions of software creation and evolution activities, with a focus on the effects which the activities produce, rather than the specific procedures used to produce those effects. (This appears to be goal focussed – one aspect covered in the CMM) The behavioural approach to model building is covered in Figure 2 where goals are listed. Goals are often a starting point to tailoring and prioritising a company's needs (*Ferraiolo*, 2002).

To conclude:

To avoid the limitations inherent in both abstractionism and contextualism, we aim to build a model that is a synthesis of the strength of the two methods of model building. This is in line with the CMM, which is "a descriptive model in the sense that it describes essential attributes that would be expected to characterize an organization at a particular maturity level. It is a normative model …" (Paulk, 1995).