IST-1999-55017

Software Process Maturity Model Study

Deliverable A.3

Owner	Michael Grottke
Approvers	Eric David Klaudia Dussa-Zieger
Status	Approved
Date	02/07/01

Contents

1 Introduction

1.1 Project overview

The PETS project is a two-year research and development effort on the part of a consortium of three industrial and one academic bodies, partially funded by the European Commission under the Fifth Framework Agreement (CRAFT project). The objective of PETS is to develop a new, enhanced statistical model which predicts the reliability of SW programs based on test and software maturity results. This new model will be implemented as a prototype which allows small and medium enterprises, especially small software companies, to predict the reliability of their developed and tested software in an easy and efficient manner with a higher accuracy than currently possible.

1.2 This document

Since the mid 1980's, professionals and researchers in software engineering have been aware of the fact that the quality of a software product is not only determined by its complexity, by the skill of the software developers, and by the features of the development environment used (e.g. the application of CASE tools). Rather, the quality of the business processes in a software company has considerable influence on the general capability of satisfying the performance objectives.

In the software reliability model study [18] a unifying framework for software reliability growth models was derived, which includes model elements referring to the quality of the application under test (the expected number of faults in the software) as well as elements depending on characteristics of the testing done (the development of test coverage in time and the testing efficiency). Therefore, it might be beneficial to include information about the maturity of the general software development process and the testing process in these models. This could lead to more stable parameter estimates and better predictions of the number of failures to be experienced in future especially at the early stages of testing when only few failure data are available.

In chapter 2, several models of software process capability are described with special emphasis on the structure of the model components. The influences software process capability is likely to have on software reliability as well as the benefits and problems of linking software maturity information to software reliability models are discussed in chapter 3. In the last section of that chapter, conclusions based on these arguments are stated.

1.3 Related document

The software reliability model study (deliverable A.2) [18] presents several standard software reliability growth models and puts them in a unifying framework. It also discusses possible approaches for including additional information (like data referring to the software process capability) in these models.

2 Software process capability models

In this chapter, several models for determining the capability of the software process(es) are studied. In order to understand how aspects of reality are used to determine a capability or maturity level of a process, it is important to identify the components of the underlying models and how they are related. Therefore, each model is depicted by a diagram providing this kind of information. The notation in the diagram follows the one of entity relationship diagrams and of class diagrams in UML. Boxes represent entity types, i.e. types of real-world objects and constructs that are part of the model. Relationships between the entity types are drawn as lines. The way in which the relationship line connects with the entities determines the degree of the relationship: Each line end shows how many entities of the type at this end are related to one entity of the type at the other end. The line ends used and their meaning are listed in figure 1.

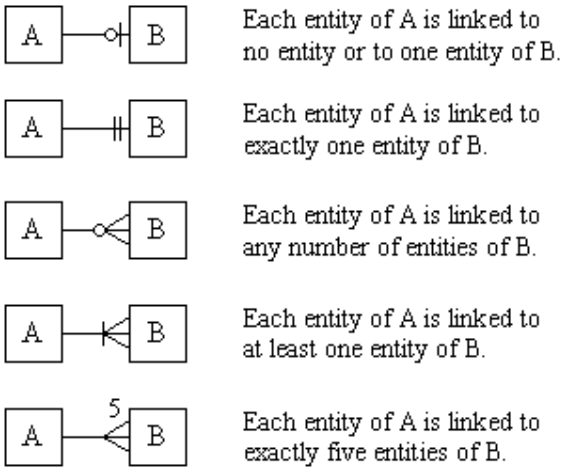

Figure 1: *Line ends and their meaning*

Since a relationship is two-directional, it gives two pieces of information: How many entities of the one type are related to one entity of the other type, and how many entities of the other type are related to one entity of the one type. For example, according to the relationship in figure 2 each entity of A is linked to at least one entity of B, and each entity of B is linked to exactly one entity of A. The degree of the relationship is (1,1) to (1,n).

Figure 2: *Example for a relationship type*

A different kind of relationship is the generalization, which is depicted by a triangle as shown in figure 3. The generalized entity type A, at which one of the corners of the triangle points, consists of the entity types A1, A2 and A3. (Or - changing the viewpoint - A1, A2 and A3 all are special types of A.)

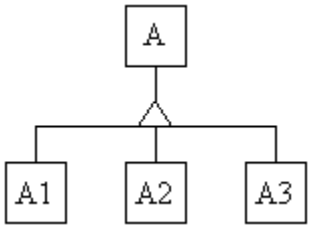

Figure 3: *Example for a generalization relationship*

Including some more information into the diagrams, entity types that may be rated for the determination of the maturity or capability level are shown as ellipses. Those model components which are not required according to the model and therefore do not have to be taken into account during an assessment are drawn with dashed lines.

2.1 Capability Maturity Model for Software (SW-CMM), version 1.1

The Capability Maturity Model for Software (SW-CMM) was developed at the Software Engineering Institute (SEI) of Carnegie Mellon University, Pittsburgh. The version 1.0 was published in 1991, version 1.1 [37, 38, 39] followed in 1993. Meanwhile, concepts of the SW-CMM have been included in the Capability Maturity Models Integration (CMMI) models, which integrate disciplines like systems engineering, software engineering and integrated product and process development (cf. section 2.3). Since SW-CMM 1.1 has served as an input for the newer models and since it is still used in organizations today, its discussion seems to be helpful.[1]

The basic assumption of the SW-CMM is that the *capability* of the software process of an organization - defined as the range of expected results that can be achieved by following a software process [38] - depends on its *maturity*, i.e. the extent to which it is explicitly defined, managed, measured, controlled and effective [38]. It is noteworthy that the SW-CMM does not distinguish between different processes of a software developing organization, for example software development and software maintenance, but it considers related activities to belong to *the one* software process.

This does not mean, however, that the maturity of the entire organization has to be evaluated. The scope of an appraisal can be any portion of an organization, e.g. [35]:

- An entire company
- A selected business unit
- A specific geographic site

[1] In the following the abbreviation SW-CMM always refers to the version 1.1 of the Capability Maturity Model for Software.

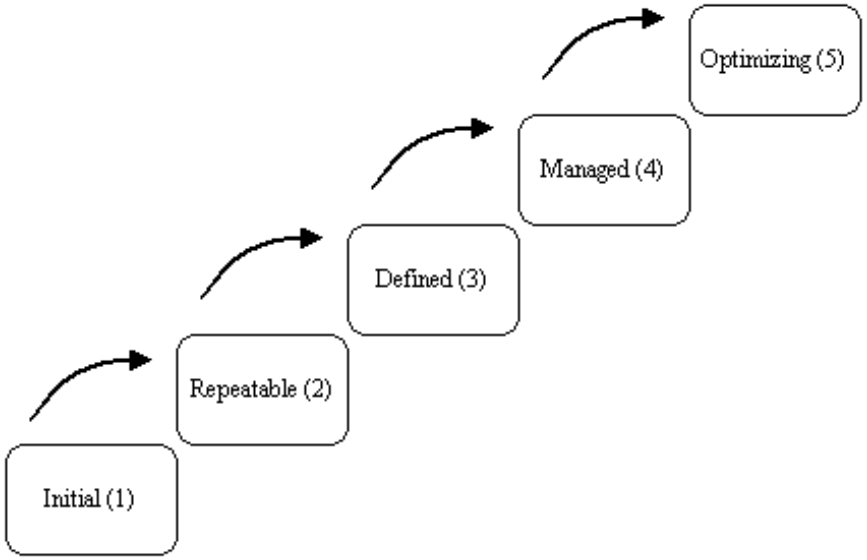

Figure 4: *Levels of Software Maturity according to SW-CMM (cf. [37])*

- Organizational units involved in producing software for a particular product line or application domain
- Organizational units involved in providing a particular type of software service such as maintenance or reengineering of legacy systems
- A specific project
- A team involving multiple companies

The maturity is measured on an ordinal scale. Each of the five defined maturity levels (cf. figure 4) describes goals that have to be satisfied (in addition to the goals on the lower levels) for attaining it. These levels can be characterized as follows [37]:

- Level 1: Initial
  The software development projects follow an *ad hoc* process. There are no defined procedures and techniques for planning and controlling projects. Since the processes are not stable but continuously modified, experience from earlier projects cannot be used for project management. Therefore, the budget needed and the quality of the software product are hardly predictable. Whether a project is successful or not basically depends on the skill and the efforts of the project manager and the team members. If they leave the company, their knowledge is lost.
- Level 2: Repeatable
  At this level basic policies for planning and tracking software projects are established and implemented. The experience with similar projects can be used for realistic predictions of the necessary input and the output of a project. If the environmental

conditions remain stable successes from these earlier projects can be repeated. Moreover, procedures of software quality assurance and the management of subcontracts are introduced.

- Level 3: Defined
  Both software engineering and technical processes are defined and documented. Inconsistencies between the different processes are resolved at the same time. The resulting integrated process is referred to as the organization's standard software process. An organization-wide training program guarantees that the employees have the necessary knowledge about this standard process to fulfil their assigned roles. The organization's standard software process cannot fit and be efficient for all projects with their specific characteristics. Therefore, in each project it is tailored to become the project's defined software process. In order to meet the goals of a project, the different groups commit themselves to the customer's requirements and to their own responsibilities. Verification mechanisms ensure that defects in work products are identified and removed.

- Level 4: Managed
  While an organization with CMM level 3 has defined what has to be done, it does not necessarily know how good these activities are carried out. Therefore, at level 4 quantitative quality goals are set and tracked. The basis for these goals are defined measurements taken across all projects and stored in an organization-wide database. (Statistical) Analysis of this data can help to identify causes of variations in process performance. This understanding facilitates better predictions for individual projects.

- Level 5: Optimizing
  Using the measurements taken since level 4, the organization identifies weaknesses of the process and analyzes the costs and benefits of changes to the process or the introduction of new technologies. Identified improvements are implemented in the entire organization. Furthermore, measures for defect prevention are taken: The common causes of defects are determined, prioritized and removed.

The descriptions of the levels imply that for attaining the respective next level different (sub-)processes have to be established. Due to this it is not possible to determine the maturity level of an individual process. Whether these processes are implemented or not affects the maturity level of *the one* software process of an organization.

In the SW-CMM the (sub-)processes are called *key process areas*. To each of the maturity levels (except the Initial Level 1) several key process areas are attached. Each of these key process areas serves a number of goals.

For example, the key process areas of the Repeatable Level 2 are Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Software Subcontract Management, Software Quality Assurance and Software Configuration Management [37]. The key process area Requirements Management is aimed at the following goals [39]:

1. System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

2. Software plans, products, and activities are kept consistent with the system requirements allocated to software.

According to the CMM Appraisal Framework [35], which describes requirements for conducting (internal) assessments and (external) evaluations of software process capability, a rating algorithm has to follow this structure: At first the goals are rated. A key process area can only be rated satisfied if all of its goals are satisfied. An organization attains a maturity level if all of its key process areas and all of the key process areas of each lower level are either satisfied or not applicable.

How the goals are achieved is not prescribed by the SW-CMM. While this keeps the model flexible enough for applying it to diverse organizations, this also makes it less operational. Whether the procedures established satisfy a goal is always a matter of interpretation.

What could be done to satisfy a goal is described in terms of *key practices*. However, these key practices are not essential, which means that any other solution that achieves the objectives is also acceptable. In each key process area the key practices are structured according to the following five aspects, named *common features* [38]:

- Commitment to Perform:
  It contains activities necessary for establishing and maintaining the process area.

- Ability to Perform:
  It describes preconditions which must exist to implement the process competently, e.g. resources, organizational structures, and training.

- Activities Performed:
  It contains the procedures which have to be implemented for establishing a key process area.

- Measurement and Analysis:
  It describes the activities necessary for determining the status and the effectiveness of the activities performed.

- Verifying Implementation:
  It contains the procedures that ensure that the activities performed are in compliance with the planned process.

For example, the key practices of the Activities Performed common feature of the process Requirements Management are [39]:

1. The software engineering group reviews the allocated requirements before they are incorporated into the software project.

2. The software engineering group uses the allocated requirements as the basis for software plans, work products, and activities.

3. Changes to the allocated requirements are reviewed and incorporated into the software project.

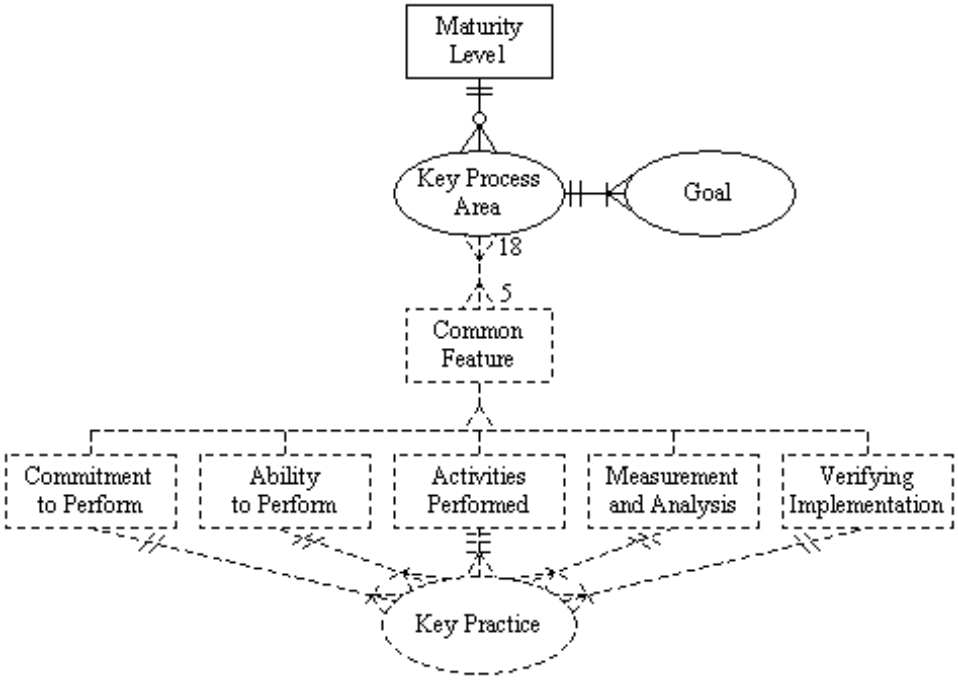

Figure 5: *Elements of the SW-CMM*

The relationships between the components of the SW-CMM are depicted in figure 5.

In 1994, the SEI published a maturity questionnaire to be used as an instrument during a CMM appraisal [41]. For each of the 18 key process areas it contains between six and eight questions asking either whether its goals have been achieved or whether key practices attached to it have been implemented. The questions can be answered with "yes", "no", "does not apply" or "don't know". Unlike an earlier version, the questionnaire does not come with a scoring method for calculating the maturity level based on the answers. The reason for this is that the key practices are suggestions but not orders that have to be unconditionally followed. Therefore, even if several questions about the key practices of a key process area have been answered with "no", this does not necessarily mean that the key process area is not satisfied. This means, that the questionnaire cannot be the sole source of information used in determining the maturity level. However, the questionnaire is especially suited for getting a first impression about the quality of the processes and for focussing on key process areas about which more information has to be collected in the organization. Further sources of information are documents, i.e. directive documents, work products and work records, presentations to a wider audience consisting of members of the organization and interviews with individual members of the organization [35]. For ensuring the validity of the results, judgements on whether the goals of a key process area have been achieved or not should always be based on data from multiple sources.

2.2 Emerging standard ISO/IEC 15504

In 1993, the SPICE (Software Process Improvement and Capability dEtermination) project was established by the seventh subcommittee (SC7) of the Joint Technical Committee 1 (JTC1). The JTC1 had been founded by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) in 1987 with the mandate to create standards in the field of information technology. The domain of the SC7 is software engineering [12]. The objective of the SPICE project is to develop the international standard ISO/IEC 15504 for the assessment and improvement of software processes, which integrates existing approaches like ISO 9000 and SW-CMM. In 1995, the version 1.00 of the document set consisting of nine parts ([20] - [28]) was published as a working draft. It was replaced by a proposed draft technical record with a version showing significant changes to the model after one year; this version has been referred to as version 2.00 [15]. The current version was published in 1998 in form of a type 2 technical report (e.g. [29] - [33]). These kinds of reports can be considered "standards under trial" and are used in order to quickly bring the documents to the marketplace and to make an allowance for the experiences with their application in industry [15]. For the next months, it can be expected that either the international standard ISO/IEC 15504 will be established or that there will be the decision to keep the type 2 technical report status for another three years.

In the following subsections, both the working draft and the technical report version of the emerging standard - referred to as ISO/IEC WD 15504 and ISO/IEC TR 15504, respectively - will be discussed.

2.2.1 ISO/IEC WD 15504

The central objective of the ISO/IEC WD 15504 documents is to describe a process model and the recommended procedures using it for assessing an organization's software processes. The results can be used for taking process improvement actions or for determining the capability of the processes, i.e. for identifying the strengths, weaknesses and risks associated with deploying the processes to meet a particular specified requirement [28].

The evolutionary plateaus, the ordinal categories in the measurement of process capability, are called *capability levels*. In contrast to the SW-CMM, there are six levels - cf. figure 6. In part 2 of ISO/IEC WD 15504 (denoted by ISO/IEC WD 15504-2), these levels are defined as follows [21] (cited verbatim):

- Level 0: Not Performed
  There is general failure to perform the base practices in the process. There are no easily identifiable work products or outputs of the process.

- Level 1: Performed Informally
  Base practices of the process are generally performed. The performance of these base practices may not be rigorously planned and tracked. Performance depends on individual knowledge and effort. Work products of the process testify to the performance. Individuals within the organization recognize that an action should be performed, and there is general agreement that this action is performed as and when required. There are identifiable work products for the process.

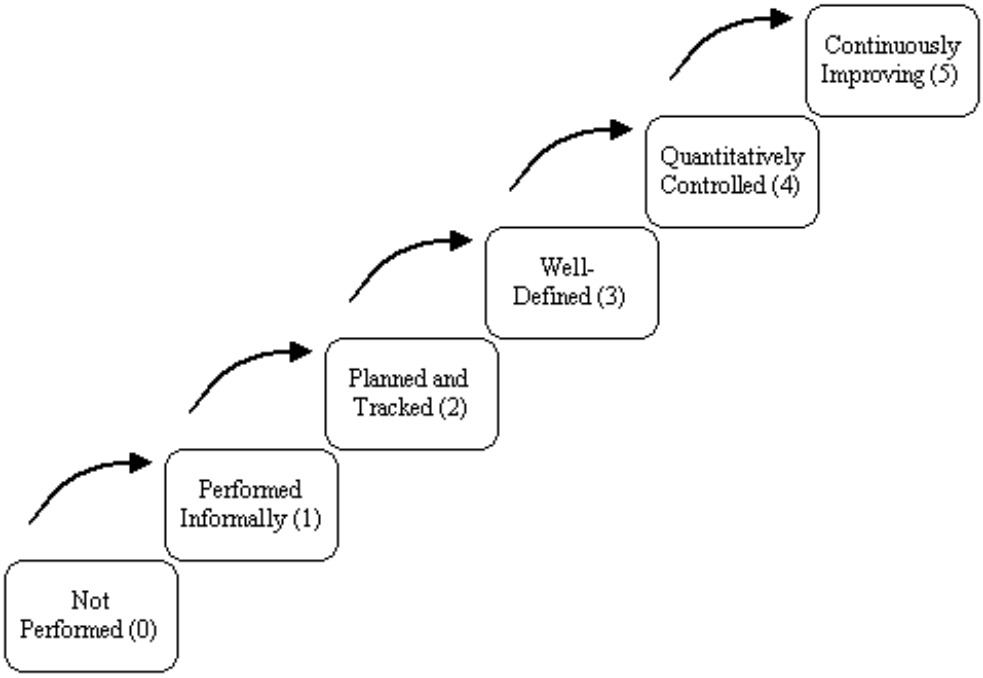

Figure 6: *Levels of the ISO/IEC WD 15504 model*

- Level 2: Planned and Tracked
  Performance of the base practices in the process is planned and tracked. Performance according to specified procedures is verified. Work products conform to specified standards and requirements. The primary distinction from the Performed-Informally Level is that the performance of the process is planned and managed and progressing towards a well-defined process.

- Level 3: Well-Defined
  Base practices are performed according to a well-defined process using approved, tailored versions of standard, documented processes. The primary distinction from the Planned and Tracked Level is that the process of the Well-Defined Level is planned and managed using an organization-wide standard process.

- Level 4: Quantitatively Controlled
  Detailed measures of performance are collected and analyzed. This leads to a quantitative understanding of process capability and an improved ability to predict performance. Performance is objectively managed. The quality of work products is quantitatively known. The primary distinction from the Well-Defined Level is that the defined process is quantitatively understood and controlled.

- Level 5: Continuously Improving
  Quantitative process effectiveness and efficiency goals (targets) for performance are established, based on the business goals of the organization. Continuous process im-

provement against these goals is enabled by quantitative feedback from performing the defined processes and from piloting innovative ideas and technologies. The primary distinction from the Quantitatively Controlled Level is that the defined process and the standard process undergo continuous refinement and improvement, based on a quantitative understanding of the impact of changes to these processes.

A comparison of capability levels 1 to 5 to the SW-CMM maturity levels 1 to 5 shows that the ISO/IEC WD 15504 model assumes the same progression from a performed but chaotic process to an optimizing process. However, there is an important difference: While the SW-CMM rates *the one* software process of an organization, in the ISO/IEC WD 15504 model a capability level is attached to an individual process, like "Determine Software Requirements" or "Establish Project Plan". This means that the processes do not reside on one specific level like the key process areas in the SW-CMM. Rather, the model set-up has two dimensions: According to the descriptions in the process dimension it can be decided whether a process is performed at all in an organization. For the processes performed the capability level equal to or higher than one can be determined in the capability dimension.

The thirty-five processes in ISO/IEC WD 15504-2 [21] are organized in five *process categories*: customer-supplier, engineering, project, support and organization. These process categories and processes can be mapped to the three process categories and seventeen processes of ISO 12207, which describes a software life-cycle model and which is intended for use in a contract situation involving the acquisition and supply of systems containing software [21, Annex E]. Due to the different scope, the ISO/IEC WD 15504 model is more detailed in most areas.

For each process its *purpose* is stated in one paragraph. For example, the purpose of the Develop Software Requirements process is to establish, analyze and refine the software requirements [21]. Furthermore, several *base practices* are attached to each process, i.e. software engineering or management activities that directly address its purpose and contribute to the creation of its output [28]. The base practices of the Develop Software Requirements process, for example, are as follows [21]:

1. Determine software requirements
2. Analyze software requirements
3. Determine operating environment impact
4. Evaluate requirements with customer
5. Update requirements for next iteration

The base practices are considered essential activities of a particular process [28]. Therefore, in an assessment all the base practices of the processes which are in the scope of the assessment have to be evalutated [21]. Only if all of its base practices exist (even if they are *ad hoc*, unpredictable, inconsistent and poorly planned [17]) a process can be rated performed (Capability Level 1 or higher).

The execution of practices typically produces artefacts referred to as *work products* [28]. Assessors may regard the work products they find in an organization as indicators for the

performance of the practices they are associated with. Part 5 of ISO/IEC WD 15504 [24, Annex C] contains lists of the input and output work product types of the base practices defined in part 2. E.g., the input work product types of the Develop Software Requirements process are

- Customer Requirements
- Maintenance Requirements
- Product Needs Assessment
- Customer Request
- Change Request and
- System Design/Architecture,

while its one output work product type is Software Requirements [24].
Furthermore, for each work product so-called *work product characteristics* are provided [24, Annex D], attributes which indicate the adequacy of the implementation of the practice generating the work product [28]. For example, the work product Change Request should have the following characteristics [24, Annex D]:

- Identifies purpose of change
- Identifies request status (new, accepted, rejected)
- Identifies requester contact information
- Impacted system(s)
- Impact to operations of existing system(s) defined
- Impact to associated documentation defined
- Criticality of the request, date needed by

In contrast to the base practices themselves, the work products and the work product characteristics are not essential. Like the key practices in the SW-CMM, their description is intended to give an idea of a possible implementation of a process (area) without prescribing it. If all the base practices of a process exist even though the attached work products do not, the process is performed.

In the capability dimension it is determined *how good* a process is performed, i.e. at which of the capability levels 1 to 5 it resides. For this end twenty-six generic practices are used. They are grouped by eleven common features. In the SW-CMM the common features are the same for all process areas; here they are common to all processes. However, while in the SW-CMM the process areas themselves were attached to maturity levels, in the ISO/IEC WD 15504 model each common feature belongs to a certain capability level [21, chapter 5]:

- Common Feature for Capability Level 1: Performing Base Practices
- Common Features for Capability Level 2:
  - Planning Performance
  - Disciplined Performance
  - Verifying Performance
  - Tracking Performance
- Common Features for Capability Level 3:
  - Defining a Standard Process (Organization-Level Common Feature)
  - Performing the Defined Process
- Common Features for Capability Level 4:
  - Establishing Measurable Quality Goals
  - Objectively Managing Performance
- Common Features for Capability Level 5:
  - Improving Organizational Capability (Organization-Level Common Feature)
  - Improving Process Effectiveness

Just like in the SW-CMM, the common features address aspects in the implementation and institutionalization of a process. In contrast to the latter model, in which a key process area could only be executed or not, here the capability of the process can be at one of the five levels. Consequently, practices like the measurement and analysis of the process performance (one of the common features of the SW-CMM) are assigned to the Capability Level 4 (Quantitatively Controlled). Note that no equivalent to the Activities Performed common feature of the SW-CMM is included in the capability dimension of the ISO/IEC WD 15504 model. These procedures belong to the base practices of the process dimension; the Level 1 common feature Performing Base Practices refers to them in a general way. Since the process-specific practices were moved to the process dimension, the twenty-six practices further specifying the common features are the same for all processes. Thus, they are called *generic practices*. There are between one and six generic practices connected to each common feature. In the ISO/IEC WD 15504 model, the generic practices and common features are considered essential elements - unlike the key practices and the common features in the SW-CMM.

Assessment indicators of process capability are described in part 5 of ISO/IEC WD 15504 [24, Annex A]. To each generic practice several *process management indicators* are attached, which may provide guidance to the assessor on what to probe for in the organization to determine whether the respective generic practice has been adequately implemented. For the generic practice Assign Responsibilities (which belongs to the Capability Level 2 common feature Planning Performance), for example, the following process management indicators are listed [24, Annex A]:

- Job responsibilities correspond to tasks attributes defined in the practices.
- Representative understands the process and tasks they are responsible for (sic!).
- Staff assigned either have the skills or are planned to have the skills needed to perform the task.
- Assigned responsibilities are recorded

Like the work product characteristics the process management indicators are not essential, which means that the generic practices can also be implemented in a different way. These two types of assessment indicators are referred to as *process indicators*. Figure 7 shows the elements of the ISO/IEC WD 15504 model and their relationships.

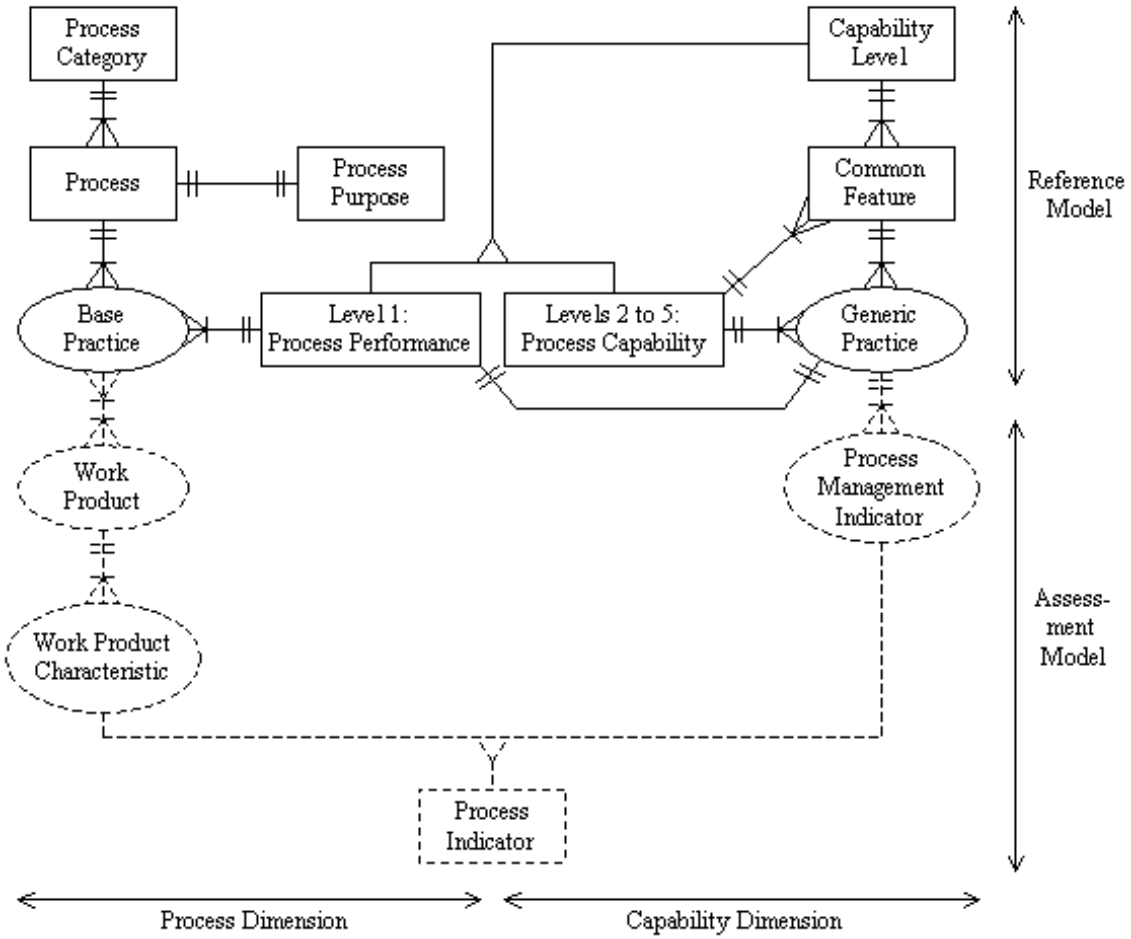

Figure 7: *Elements of the ISO/IEC WD 15504 model*

2.2.2 ISO/IEC TR 15504

Field trials with the ISO/IEC WD 15504 model showed the need for the following changes [17]:

1. A better alignment of the processes with ISO 12207
2. An even less prescriptive process reference model

The first request was taken care of by restructuring the processes. In ISO/IEC TR 15504 the processes are organized an a hierarchy: Four of the twenty-four top-level processes contain sixteen second-level processes. Furthermore, the name of the Project process category was changed into Management process category to better line up with the terminology of ISO 12207.

To make the model less prescriptive, the base practices were moved form part 2 of the document set to part 5, the assessment model. Therefore, they are no longer elements of the reference model. This change also shows in the definitions of the capability levels in ISO/IEC TR 15504 [29]: Even though their names have been altered (cf. figure 8), their descriptions basically remain the same. However, all the references to the base practices have been replaced by allusions to the process purpose. This shows that the process purpose statement, which already existed in the working draft version of the model, has now become a central element. While the base practices of a process propose one possible way for attaining its purpose, their implementation is not mandatory.

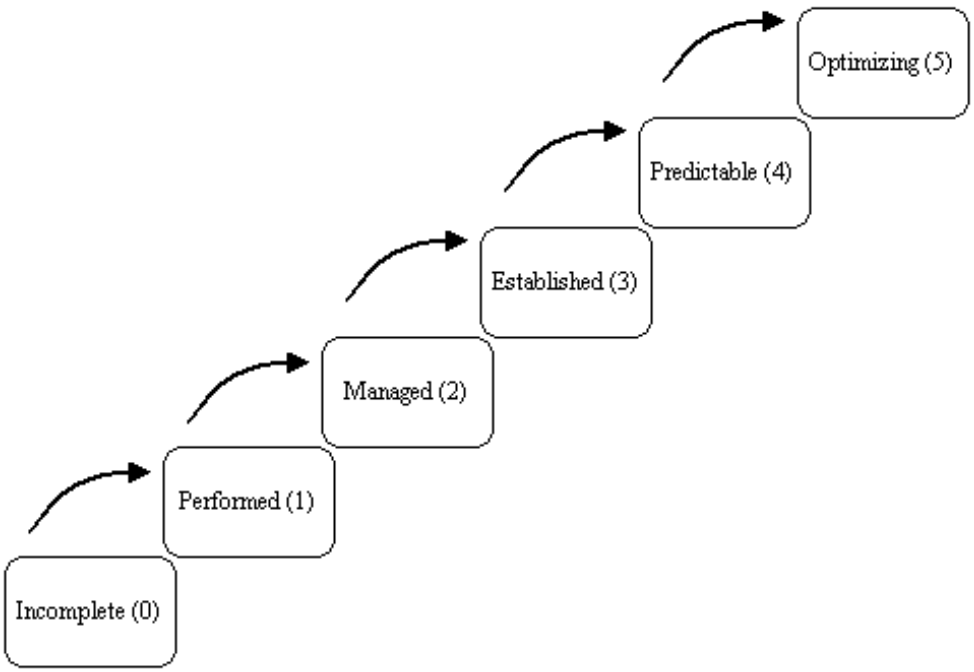

Figure 8: *Levels of the ISO/IEC TR 15504 model*

In the capability dimension similar changes have taken place: Only the nine process attributes, which correspond to the common features in ISO/IEC WD 15504, are essential

elements. The thirty-six *management practices* - formerly called *generic practices* -, which are the means for achieving the capabilities addresses by the process attributes [32], have been moved to the assessment model.

To help assessors determine whether a management practice has been successfully implemented, Annex B of ISO/IEC TR 15504-5 lists indicators for each of the management practices. These indicators, however, are not to be used as checklists, but they must always be adapted to the respective context [32]. There are three types of process capability indicators:

- Practice performance characteristics are concerned with the implementation of the management practice.
- Resource and infrastructure characteristics provide mechanisms for assisting in the management of the management practice.
- Associated processes are processes that may support the implementation of the management practice.

The refined (reference and assessment) model as described in ISO/IEC TR 15504 is depicted in figure 9.

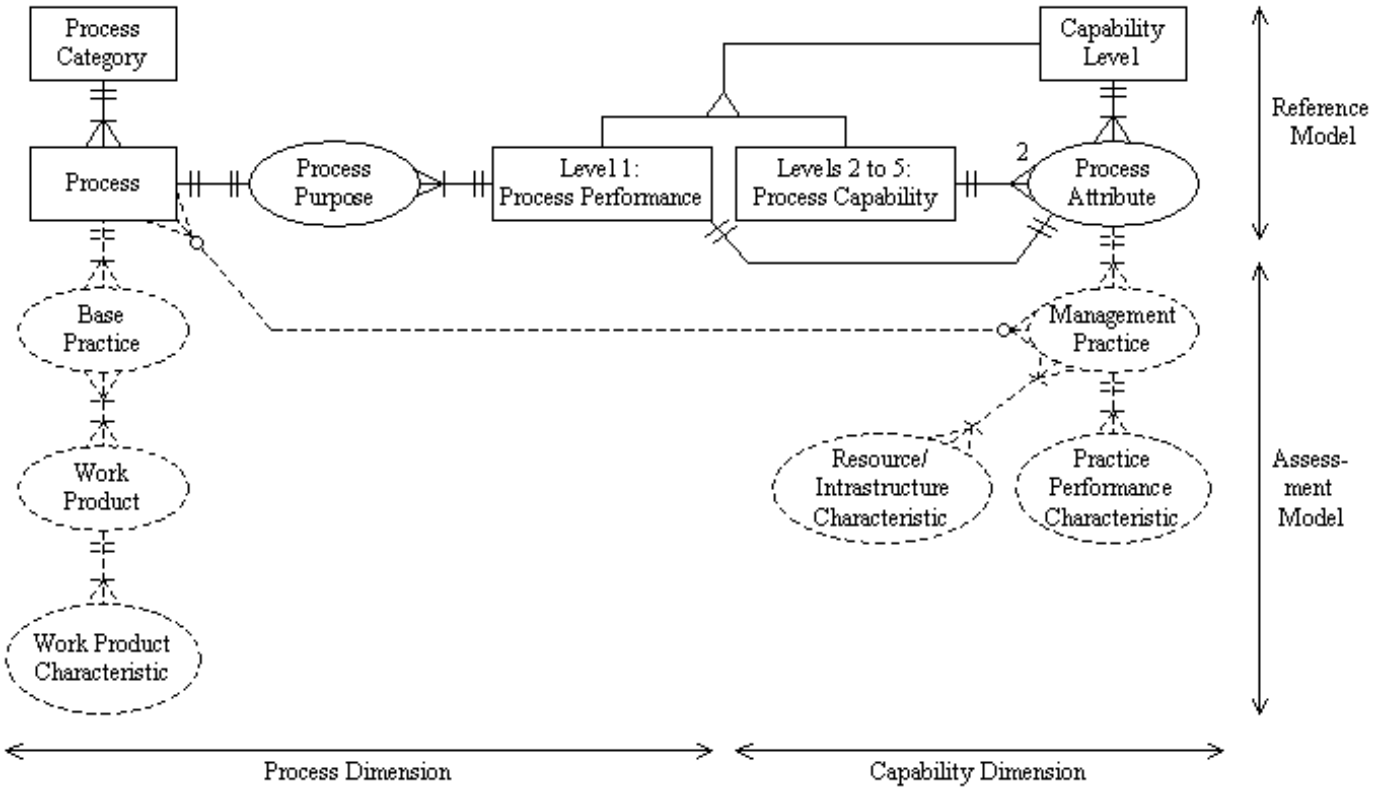

Figure 9: *Elements of the ISO/IEC TR 15504 model*

Note that for an ISO/IEC TR 15504-2 compatible assessment the elements of the assessment model are not obligatory. It is stated that the elements used in the process dimension shall focus on the effective implementation of the processes through their work products and that the elements in the capability dimension shall focus on the process management practices that realize the process attributes [29]. However, they do not have to be defined or structured like in ISO/IEC TR 15504-5. This means that if the process capabilities of several organizations have been rated according to ISO/IEC TR 15504-2, only the rating of the achievement of process purposes and the rating of the process attributes have to follow a common scheme. The way in which these ratings were derived and are backed up may be different.

2.3 Capability Maturity Models Integration (CMMI) models

The SW-CMM had been specifically designed for determining the process maturity of software development companies. Therefore, other (or broader) areas of application, like systems engineering were not addressed. Moreover, the SW-CMM did not deal with some important issues of software engineering projects, e.g. personnel management. Therefore, CMMs for other disciplines were developed, like the Systems Engineering Capability Maturity Model (SE-CMM) [1], the Software Acquisition Capability Maturity Model (SA-CMM) [13] or the People Capability Maturity Model (P-CMM) [14].

The objective of the Capability Maturity Models Integration (CMMI) project is to integrate the various CMMs and to create one consistent framework for them. Furthermore, the resulting models shall be compatible to the ISO/IEC TR 15504 reference model. There will be several CMMI models, each one of them integrating a different set of disciplines. So far, models for Systems Engineering and Software Engineering (CMMI-SE/SW) [6, 7] and for Systems Engineering, Software Engineering and Integrated Product and Process Development (CMMI-SE/SW/IPPD) [8, 9] have been released. There is a working draft version of a model for Systems Engineering, Software Engineering, Integrated Product and Process Development and Acquisition (CMMI-SE/SW/IPPD/A) [10, 11].

Each of the models comes in two representations: While the staged representation [6, 8, 10] permits an easy migration from SW-CMM to CMMI, the continuous representation [7, 9, 11] facilitates an easy comparison to ISO/IEC TR 15504. Both representations are discussed in the following subsections with specific reference to [6] and [7].

2.3.1 Staged representation

The structure of the staged representation is such that the maturity level of an organization's software process can be determined like in SW-CMM. The names of the maturity levels have been adapted to the terminology of ISO/IEC TR 15504 (cf. figure 10), but their descriptions [6] have basically remained unchanged. The key process areas - now called *process areas* - have been restated to better align with the ISO/IEC TR 15504 processes. In the staged representation, each of the twenty-two *processes* is attached to one of the maturity levels 2 to 5. For example, the process areas assigned to maturity level 2 are Requirements Management, Project Planning, Project Monitoring and Control, Supplier Agreement Management,

Measurement and Analysis, Process and Product Quality Assurance and Configuration Management [6] - this resembles closely the list given in section 2.1.

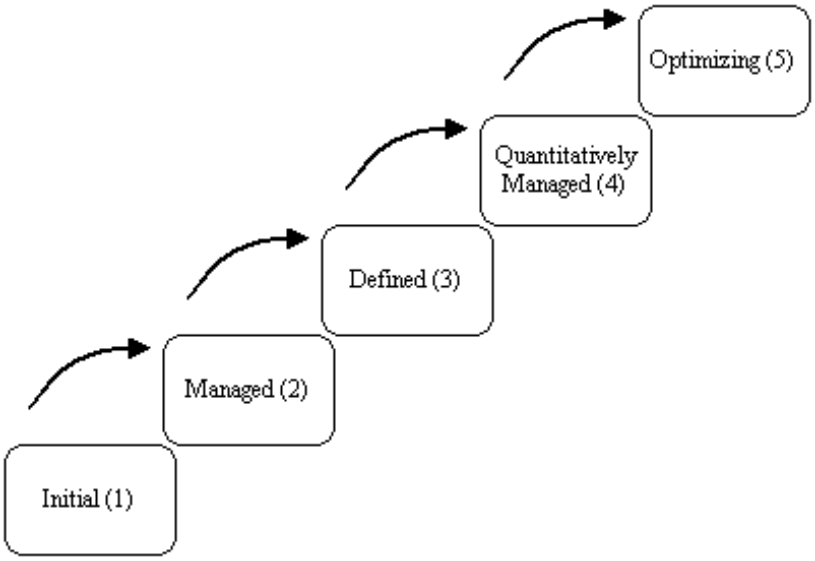

Figure 10: *Maturity levels of the CMMI models with staged representation*

An important feature of the CMMI models is the distinction between *specific goals* and *generic goals* of processes, which mimics the two dimensions in the ISO/IEC TR 15504 model. For each process area there are one or more specific goals describing what procedures have to be implemented for satisfying it. Each specific goal applies to one process only. The *specific practices* mapped to a specific goal are activities which are considered important for achieving the goal.

For example, for the one specific goal of the process area Requirements Management, "Manage Requirements", there are the following suggested specific practices [6]:

- Obtain an Understanding of Requirements
- Obtain a Commitment to Requirements
- Manage Requirements Changes
- Maintain Bi-directional Traceability of Requirements
- Identify Inconsistencies between Project Work and Requirements

For a process area to be rated satisfied not only its specific goals have to be rated satisfied, but also its generic goal. There are only two different generic goals, "Institutionalize a Managed Process" and "Institutionalize a Defined Process", which apply to all process areas attached to maturity level 2 and to all processes attached to maturity levels 3 to 5, respectively. For the generic goal "Institutionalize a Managed Process", there are ten *generic practices*, i.e. activities whose implementation improves the process performance. In addition to two more

generic practices, these ten generic practices also apply to the "Institutionalize a Defined Process" generic goal. Like the key practices in the SW-CMM, the generic practices are organized by *common features*. In the CMMI framework, there are the four common features Commitment to Perform, Ability to Perform, Directing Implementation and Verifying Implementation. Therefore, the generic practices for achieving the generic goal "Institutionalize a Defined Process" can be structured as follows:

- Generic practices of Commitment to Perform:
  Establish an Organizational Policy
- Generic practices of Ability to Perform:
  - Establish a Defined Process
  - Plan the Process
  - Provide Resources
  - Assign Responsibility
  - Train People
- Generic practices of Directing Implementation:
  - Manage Configuration
  - Identify and Involve Relevant Stakeholders
  - Monitor and Control the Process
  - Collect Improvement Information
- Generic practices of Verifying Implementation:
  - Objectively Evaluate Adherence
  - Review Status with Higher-Level Management

The practices "Establish a Defined Process" and "Collect Improvement Information" are the ones that do not apply to the generic goal "Institutionalize a Managed Process".

There is no equivalent to the Activities Performed common feature of the SW-CMM, because these activities are the specific practices assigned to the specific goals of the process area.

In the terminology of the CMMI, the specific and generic goals are *required* model components, i.e. they have to be achieved by the processes of an organization, while the specific and generic practices are only *expected* components, which provide guidance but may be replaced by alternatives as long as these alternatives ensure that the goals are met.

Figure 11 shows the components of the staged representation of the CMMI models and their relationships.

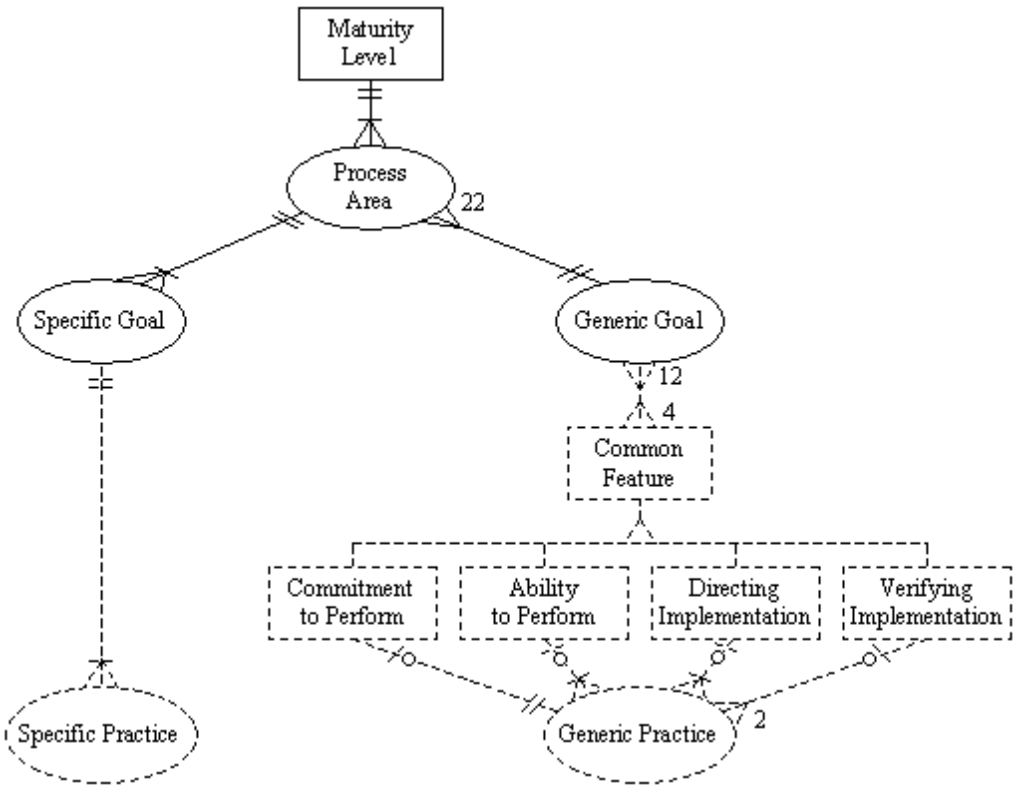

Figure 11: *Elements of the CMMI models with staged representation*

2.3.2 Continuous representation

The continuous representation of the CMMI-SE/SW [7] consists of the same twenty-two *process areas* as the staged representation. However, the process areas are not assigned to maturity levels, but organized in the four *process area categories*

- Process Management Processes,
- Project Management Processes,
- Engineering Processes and
- Support Processes.

Like in the staged representation, for each process area there are *specific goals* to attain with *specific practices* describing possible ways to do that. A notable difference, however, is that each specific practices is connected to a certain capability level. Only for this level and the ones above the specific practice applies. Although the specific practices are only expected model components in the CMMI framework, this means that the specific goals - corresponding to the *purpose* of a process in ISO/IEC TR 15504 - may be augmented for higher capability levels, while they do not change in the ISO/IEC TR 15504 model. Those specific practices that are relevant from capability level 1 on are called *base practices*.

As an example, consider the process area requirements management again. In section 2.3.1 its specific practices, which belong to the one specific goal "Manage Requirements", have already been listed. In the continuous representation [7], the specific practices "Obtain an Understanding of Requirements", "Manage Requirements Changes" and "Identify Inconsistencies between Project Work and Requirements" are considered base practices. The two remaining specific practices only get importance from capability level 2 on. Since in the staged representation the process area "Manage Requirements" resides on maturity level 2, no distinction between the five specific practices has to be made there.

For each of the capability levels there exists one *generic goal*. The names and definitions of the capability levels are very similar to those of ISO/IEC TR 15504 (cf. figure 12) [7]. Most of the generic goals just express that the respective level is to be attained. Only the generic practices provide details about how this could be done.

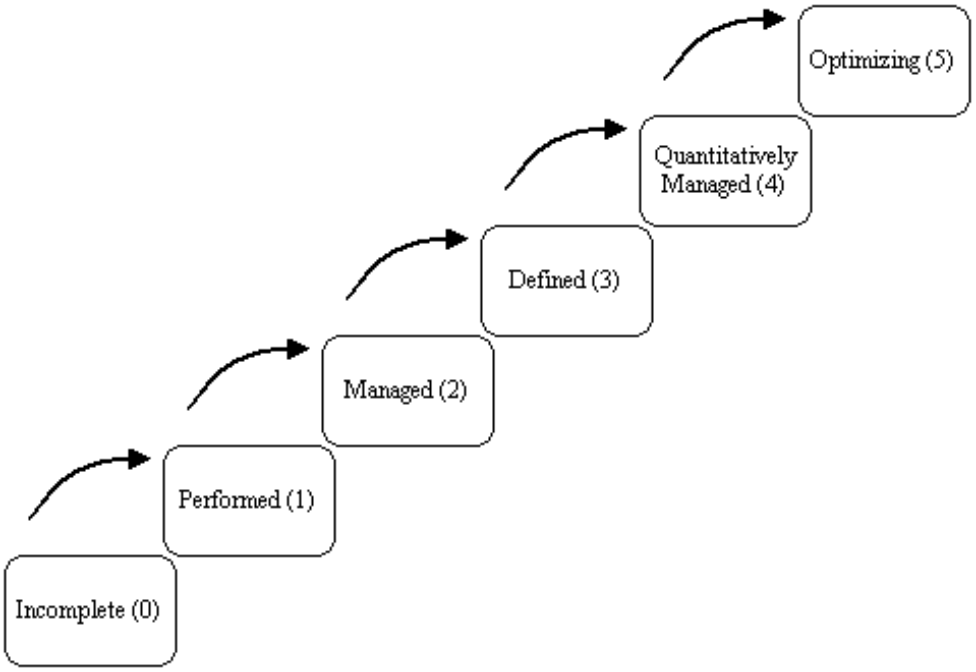

Figure 12: *Capability levels of the CMMI models with continuous representation*

The generic goals and practices are as follows [7]:

- Generic Practices of "Achieve Specific Goals" (Level 1)
  - Identify Work Scope
  - Perform Base Practices
- Generic Practices of "Institutionalize a Managed Process" (Level 2)
  - Establish an Organizational Policy
  - Plan the Process

  - Provide Resources
  - Assign Responsibility
  - Train People
  - Manage Configuration
  - Identify and Involve Relevant Stakeholders
  - Monitor and Control the Process
  - Objectively Evaluate Adherence
  - Review Status with Higher-Level Management
- Generic Practices of "Institutionalize a Defined Process" (Level 3)
  - Establish a Defined Process
  - Collect Improvement Information
- Generic Practices of "Institutionalize a Quantitatively Managed Process" (Level 4)
  - Establish Quality Objectives
  - Stabilize Subprocess Performance
- Generic Practices of "Institutionalize an Optimizing Process" (Level 5)
  - Ensure Continuous Process Improvement
  - Correct Common Cause of Problems

Comparing this structure of the generic goals and generic practices to the one in the staged representation (cf. section 2.3.1) reveals several differences:

First of all, the generic practices of the goal "Institutionalize a Managed Process" are not assigned to the goal "Institutionalize a Defined Process". In the staged representation this is necessary, because to each process area exactly one goal is mapped according to the maturity level the process area resides on. To attain a capability level in the continuous representation, however, all the goals of this level and the lower ones have to be satisfied; therefore, the generic practices do not have to be repeated.

Moreover, the capability level 1 generic goal and its generic practices are not included in the staged representation. The reason for this is that in the staged representation a process area can either be satisfied or not. For satisfying it both its specific and generic goals have to be achieved. Therefore, the instruction to attain the specific goals does not have to be stated again as a generic goal. Of course, in the continuous representation there is some redundancy with respect to the specific and generic goals on capability level 1. This phenomenon is also visible in the ISO/IEC TR 15504 model. It is caused by the fact that on the Performed Level the capability dimension and the process dimension can hardly be distinguished. One could state that the dimensions are not orthogonal. In the CMMI framework the correlation is even higher, because - as discussed above - the number of specific practices of a process may vary with the capability level examined.

Furthermore, in the staged representation process areas attached to the maturity levels 3 to 5 all have the generic goal "Institutionalize a Defined Process". There is no reference to quantitative management on level 4 or to optimization on level 5. This is not necessary, since these procedures are addressed by process areas which are contained in the respective maturity level, e.g. "Quantitative Process Management" in the Quantitatively Managed Level 4. If the continuous representation of CMMI is used, it must not be forgotten that there are dependencies between process areas and capability levels. One process area may only attain the generic goals of a certain capability level if another process area has been implemented [7].

The elements of the continuous CMMI model framework are depicted in figure 13 in the same way as for the other models.

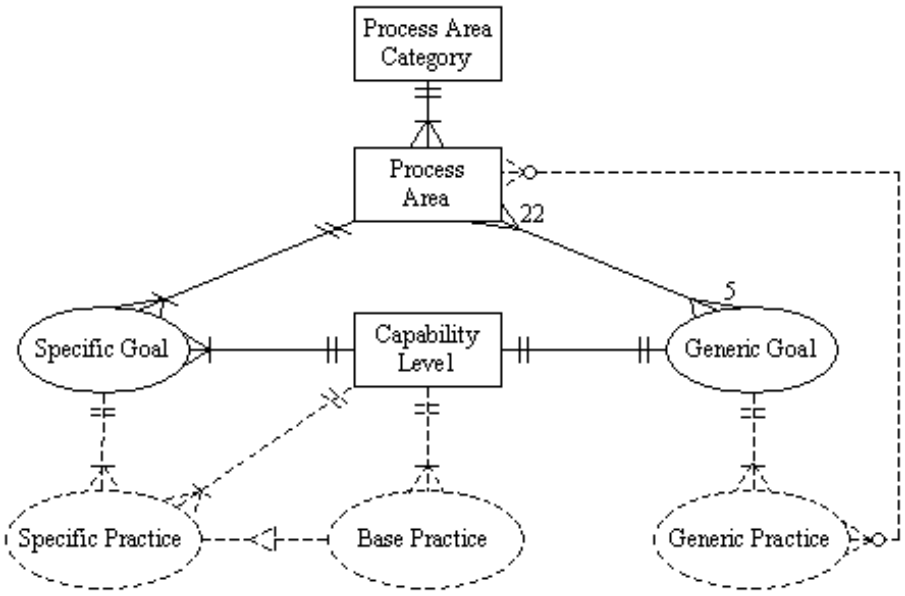

Figure 13: *Elements of the CMMI with continuous representation*

2.4 Testing Maturity Model (TMM)

The Testing Maturity Model (TMM) was developed at the Illinois Institute of Technology in the mid-1990s to remedy the fact that none of the existing software process maturity models comprehensively addressed the testing of software. Like in the SW-CMM, a basic structure of five maturity levels with aspects of the process assigned to each maturity level but the first one was chosen. Since the TMM is mainly concerned with the testing process, these aspects are not described as (key) process areas, but directly as maturity goals of the testing process.

The testing maturity levels (cf. figure 14) can be described as follows [3, 5]:

- Level 1: Initial
  For this level there is no maturity goal. Testing is done ad hoc after the software code

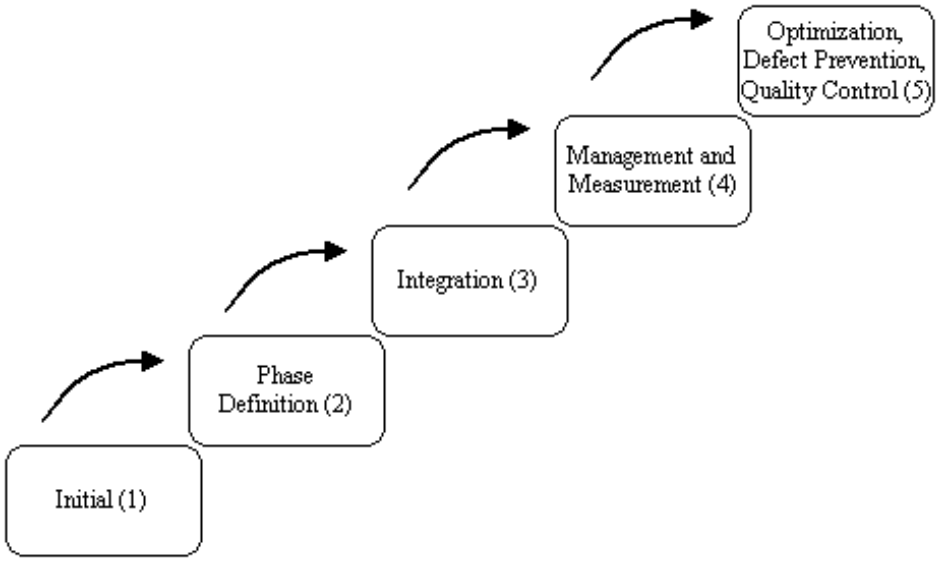

Figure 14: *Testing maturity levels of the TMM*

has been developed. It is not planned, and the process is not defined. In particular, testing is not distinguished from debugging. The staff executing the tests is not trained, and necessary resources are not available.

- Level 2: Phase Definition
  Testing is distinguished from debugging and defined as a separate process which constitutes one phase of the software life-cycle. Furthermore, testing is planned and tracked. This allows the process to be repeated. However, the planning may be done after the code has been developed. Moreover, basic testing techniques and methods are institutionalized. The goals of this maturity level are:

  1. Develop testing and debugging goals
  2. Initiate a test planning process
  3. Institutionalize basic testing techniques and methods

- Level 3: Integration
  A test group is established as an own organizational unit. The testers receive special training. Testing is not only seen as one phase of the software life-cycle, but testing activities are carried out throughout the entire life-cycle. Now testing is planned earlier than it is done at maturity level 2. The testing process is monitored, and its progress and effectiveness are controlled. Attached to this level are the following goals:

  1. Establish a software test organization
  2. Establish a technical testing program
  3. Integrate testing into the software life cycle

4. Control and monitor the testing process

- Level 4: Management and Measurement
  An organization-wide (peer) review program is established to remove faults in the work products with the help of inspections and walk throughs. Moreover, test-related metrics are defined and collected in order to analyze the quality and the effectiveness of the testing process. The goals of this maturity level are as follows:

  1. Establish an organization-wide review program
  2. Establish a test measurement program
  3. Software quality evaluation

- Level 5: Optimization, Defect Prevention, and Quality Control
  The failure data collected is used to identify root causes of faults for taking defect-preventing activities. Statistical testing based on operational profiles is used to estimate the current reliability of the software and to base test-stop decisions on defined reliability goals. Furthermore, the testing process is continuously improved. Attached to this level are the following goals:

  1. Application of process data for defect prevention
  2. Quality control
  3. Test process optimization

For each maturity goal, there are between two and five maturity subgoals supporting it. The subgoals are achieved by activities, tasks and responsibilities. In the context of each maturity goal, the activities, tasks and responsibilities are organized according to three critical views, i.e. three groups playing key roles in the testing process: managers, developers/testers and users/clients. There is some overlap with the organization of key practices by common features in the SW-CMM: the managers' view covers commitment and ability to perform and the developers' and testers' view contain parts of the activities to perform. Activities and tasks that concern user-oriented needs, however, belong to the users' and clients' view.

Figure 15 is a graphical representation of the TMM.

To support the self-assessment of testing maturity, an assessment model (TMM-AM) was developed [2, 3]. The TMM-AM is explicitly not intended for the (external) evaluation or certification of the testing process [3]. It has been constructed to be compliant to the CMM Appraisal Framework (cf. section 2.1). Therefore, the rating procedure is similar: At first the maturity subgoals are rated, then the maturity goals, and afterwards the maturity levels [2].

Like in the CMM Appraisal Framework, a questionnaire is a possible source of input data. A TMM Assessment Questionnaire has been developed and made available in a web-based version [19]. One of its eight parts consists of 161 questions structured according to the maturity goals of maturity levels 2 to 5. The possible answers are "yes", "no", "does not apply" and "don't know" - the same ones as in the CMM Maturity Questionnaire [41]. Another parallel is that no rating scheme is included in this questionnaire. While an early

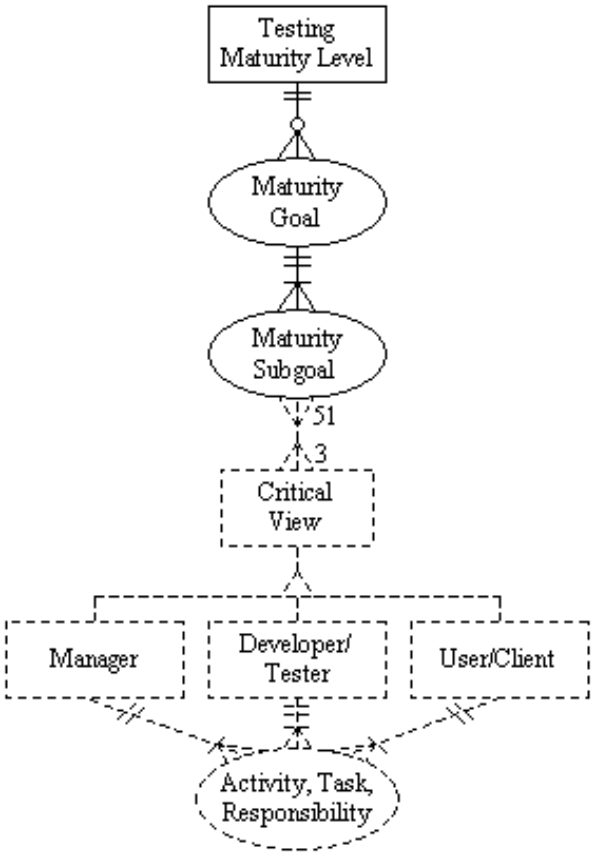

Figure 15: *Elements of the TMM*

description of the TMM-AM merely states that the responses to the questions serve as input to the ranking algorithm that determines the TMM level [2], which might hint at the existence of some simple rating scheme, a subsequent article explicitly notes that the TMM questionnaire is not to be used as the sole source of information for calculating the TMM level [3]. Rather, the responses to the questionnaire may guide the assessor in preparing the interviews with members of the organization. This is the same line of thoughts as in [35].

Olsen and Vinje [36] use a different questionnaire for TMM assessments. Its 47 questions are also structured by the maturity goals and mainly restate (a selection of) the maturity subgoals. The answer categories are “yes”, “yes & no” (partly “yes” and partly “no”) and “no”. Each “yes” is rated with 0 points, each “yes & no” with 2 points and each “no” with 4 points. A TMM level is satisfied as long as less than 25 per cent of the possible points have been attained in the related questions (and if the next lower level has been achieved).

2.5 Test Organization Maturity (TOM) model

Systeme Evolotif Limited developed a simple model for determining the maturity of the testing process and suggesting improvements to it, which they called Test Organization Maturity (TOM) model. The questionnaire [40] for assessing the TOM level also includes

information about the TOM model.

While other process capability models specify a hierarchy of process areas to be implemented or of goals to be achieved as remedies to supposed problems, the TOM model directly addresses symptoms of test organizations. In the questionnaire, twenty symptom descriptions like "Testing has no deliverables of value" are stated. These symptoms have to be scored from 1 to 5 according to the relevance of this symptom for the test organization under consideration. For each symptom, scenarios for the scores 1 (high relevance), 3 (medium relevance) and 5 (low relevance) are concisely exemplified as a guidance for scoring. Furthermore, all symptoms with a score of less than 5 should be prioritized based on the severity of their consequences.

The sum of the scores for the twenty symptoms results in the TOM level. (From a methodical point of view, this sum up of ordinal scores is problematic.) Since all symptoms have to be rated, it is a number between twenty and one hundred. When the questionnaire is repeatedly answered by an organization, this level can be used to track the progress of testing maturity.

Apart from the symptoms, objectives or constraints of the testing organization - e.g. "I want to decrease the cost of testing" - can also be prioritized. Together with the scores and priorities of the symptoms, this information is used for creating a prioritized list of potential testing improvements. This service can be obtained for free by sending the completed questionnaire to Systeme Evolutif. Eighty-three potential testing improvements are contained in their library of the model. However, these improvement suggestions and the algorithm for selecting them based on the answers in the questionnaire have not been published.

The elements of the TOM model are shown in figure 16.

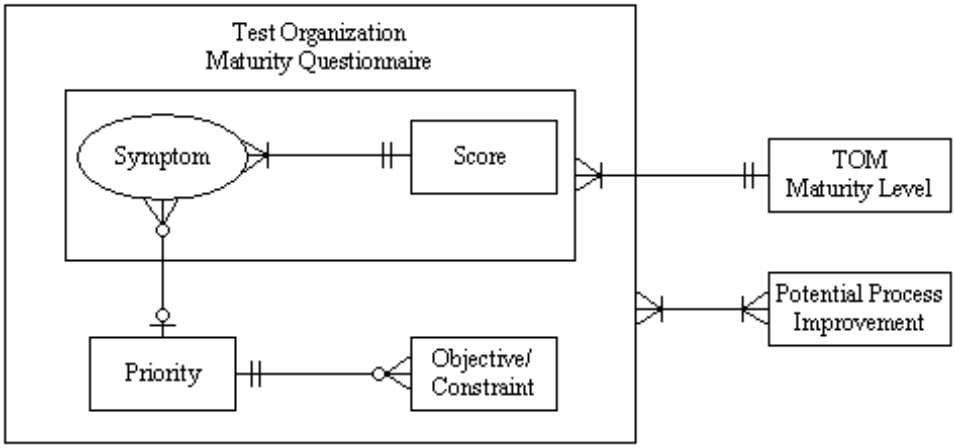

Figure 16: *Elements of the TOM model*

3 Software process capability and software reliability

This chapter is a preliminary discussion about the use of linking software process maturity or capability information to software reliability growth models. At first, the probable effects of software process capability on software reliability growth models are described. After that, the benefits and problems of using the information collected during an assessment of software process capability are discussed. Conclusions drawn from these arguments are stated in the last section of this chapter.

3.1 Influences of software process capability on software reliability

As stated above, the development of software process capability models was a reaction to the experience that the (sole) introduction of new techniques and new tools did not improve the quality of the software products. Achieving processes that are planned, tracked, defined, measured, controlled and continually improving was recognized as an important prerequisite for a constantly (i.e., independently of specific project managers) good performance of software development projects. Therefore, one should expect that a software produced by an organization with high process capability should be better - e.g. in terms of fault density (number of faults divided by lines of code) - than the same software produced by a company whose process is less mature. For the SW-CMM, Paulk et al. assert that as the maturity of an organization increases, the targeted (and actual) results in terms of costs, development time, productivity and quality improve [37]. This influence of software process maturity on software reliability growth models could be taken into account by modelling the fault density in dependence of the maturity level. Together with the lines of code of the new software, the number of faults can then be estimated and used for early prediction of the model parameters. This is exactly the approach of the simple factor multiplicative model by Malaiya and Denton [34] discussed in the Software Reliability Model Study [18].

In addition to this one, Paulk et al. presume two further effects [37]: On the one hand, the targets will be closer to the actual outcomes. Since the software process is better understood, planned and controlled, there is a sounder foundation for making predictions. On the other hand, the variability of the actual results (of similar projects) around their mean value can be expected to decrease. If a standard process is used and identified improvements are implemented across the organization, then the performance of an individual project should not diverge too much from the average. This second phenomenon could be taken into account in a Bayesian software reliability model of the finite-failure category: A higher process maturity does not only lead to a smaller expected value of the prior distribution of the number of inherent faults but also to a lower variance of this prior distribution.

In [18], two elements of software reliability growth models that are mainly related to testing have been identified: the development of test coverage in time (or its "hazard function") and the testing-efficiency, which describes how effective the testers are in producing coverage gains that lead to the occurrence of failures. Test cases which are planned by experienced testers using established techniques should be better both at attaining coverage growth and at detecting faults. In terms of software reliability growth models, this is probably not only a matter of model parameter values. Rather, the functional forms of the two effects change

as testing maturity increases.

Since testing process maturity models like the TMM (at least at the higher levels) are also concerned with activities during the software development (e.g. reviews or unit tests) a better testing maturity may have similar effects on the quality of the software before the system test phase begins - and, therefore, on the number of inherent faults in the software code - even if only some aspects of the software development process are covered. Likewise, the general software process models do not comprehensively address testing, but they do deal with aspects of it, like verification and validation. A higher software process capability may thus also have some influence on the testing-related elements of software reliability growth models.

3.2 Benefits and problems of using maturity information

As discussed in the last subsection, the quality level of the software process(es) of an organization may influence the software reliability growth model to be used and the early estimates or prior distributions of the model parameters. The exogenous variables to be linked to a software reliability growth model could be either data collected in the course of the determination of the capability or maturity level(s) according to one of the existing models described in chapter 2 or some other information unrelated to those models. There are two main benefits of employing the concepts of standard process maturity models:

1. The well-known models SW-CMM, CMMI and ISO/IEC TR 15504 have undergone or are still undergoing intensive trial phases testing the practicability of the models. Furthermore, they all build on generally accepted concepts about what path the improvement of the overall software process or of individual processes should take. By using information which follows the rating framework of one of the models - e.g. dichotome variables indicating whether a process goal/purpose has been satisfied - existing know-how and experience could be utilized.

2. Drawing on data on the required elements of process maturity models would on the one hand ensure that the information is available if an assessment of the organizational unit has already taken place. On the other hand, if data collection is necessary, it could be combined with an official assessment.

However, there also seem to be problems connected to this approach:

1. The resources required prohibit the performance of an entire project assessment with the only goal to improve the predictive power of a software reliability growth model. For an internal SW-CMM assessment of four representative projects with interviews of forty functional area representatives done by an assessment team consisting of eight team members plus assessment team leader, Dunaway and Masters give an estimated effort of 200 person-days for assessment planning, team training, on-site assessment and final report delivery, i.e. excluding the development and implementation of an action plan for improvement [16]. The typical time frame for these activities is four months. If the necessary resources were proportional to the number of projects assessed, then the assessment of one project would require about 50 person-days.

2. While at Aprote home-made software is tested and therefore internal project teams would have to be assessed, at the imbus test-lab software of separate companies is tested. Therefore, if the other organizations have not already performed an assessment, the (external) evaluation of the process maturity would have to be carried out. While at least the SW-CMM, the CMMI framework and the ISO/IEC TR 15504 model are not only directed at guiding process improvement but also at providing a structure for capability determination of a supplier by a potential customer (like the US Department of Defense), it is uncertain whether a company would be willing to endure such an evaluation only for improving software reliability estimation.

3. Experienced software testers report that - especially in larger organizations - the capability of the software process(es) and the quality of the software produced may vary considerably between divisions. Therefore, even if assessment results should be available, they may not be representative for the organizational unit which developed the software to be tested.

4. According to the CMM Appraisal Framework, the results of an assessment or evaluation have to be reported to its sponsor. Any further distribution may only be made at the sponsor's discretion [35]. In part 3 of ISO/IEC TR 15504, in which requirements of the performance of an assessment are stated, one essential type of assessment input are assessment constraints including controls of information resulting from a confidentiality agreement. Furthermore, the assessment results are to be documented and reported to the assessment sponsor [30]. Part 4 providing guidelines to these requirements declares that the mechanisms to record and retain the assessment output should ensure that all confidentiality requirements are met [31]. Therefore, it is not sure whether an organization in which an (internal) assessment has taken place will be willing to disclose any details of its outcomes. For example, only the SW-CMM maturity level of the company may be made public, but not which of the key process areas have been rated satisfied or which goals have been rated achieved. Some low level information is even not meant for disclosure to the sponsor, e.g. the interview statements of specific staff members that lead to the ratings, or related tracking data. They may be destroyed after the appraisal has been finished [35].

5. Even if all information collected during an assessment would be made available, only data concerning the required model elements necessarily have to exist. For example, the key practices in the SW-CMM and the base practices in the ISO/IEC TR 15504 model are suggestions about how to attain the goals or the purpose of a process (area). However, their implementation is not mandatory. Therefore, the sole information whether a practice is established or not does not automatically determine if a goal or purpose is satisfied. The institutionalization of the practices does not have to be rated at all. In the SW-CMM, for example, only the consecutive rating of the goals, of the process areas and of the maturity level has to take place. If the automated integration of assessment data into a software reliability growth model is intended, one has to rely on information required by the capability models. However, this information is already highly aggregated.

6. An easy way to collect information is with the help of formal instruments, like questionnaires. If "open" answers are not possible, then the results can be directly fed into a model without any additional interpretation. Since a questionnaire would guarantee that certain information is available (as long as there is no non-response) in a standard form, developing one for assessing the information considered to be relevant for the reliability of the software produced seems to be an important intermediate goal of the PETS project. As discussed in chapter 2, there exist questionnaires for several of the capability models. However, most of them are explicitly not intended to be used as the sole source of information, but only to get a first impression. For other models, no questionnaire has been developed at all. The reason for this is immanent to the capability models: the required model elements are often not concrete enough for being used as questions to project team members. More specific questions relating to practices implemented, however, can never serve for an "automatic" calculation of the process capability, because they only refer to possible ways of implementing a process (area) and have to be subject of interpretation. Therefore, the determination of process capability based on a questionnaire alone can hardly comply to a process capability model.

   Furthermore, one rule of assessment guidelines is to use data from multiple, independent sources in order to ensure the validity of the information [31, 35].

7. As the descriptions of the maturity and capability levels and the discussion in section 3.1 have shown, only software processes with a high capability lead to a predictable software quality. (It is not without any reason that the maturity level 4 of the SW-CMM is named "Predictable".) For organizations operating at a low process capability, individual effects of the specific project and the team members have considerable influence on the results. Therefore, if information on the particular software (to be) produced, on the programmers involved or on the performance of the project are available, it should be taken into account.

3.3 Conclusions

The arguments in section 3.2 lead to the following (preliminary) conclusions: Complete assessments of the process maturity or capability of software developing organizations are not feasible if their sole objective is to improve the predictive power of software reliability growth models. Even if such assessments should already have taken place, only highly aggregated information can be expected to exist. Furthermore, even this information (except the overall maturity level or the capability levels of individual processes) may be confidential. Therefore, the data to be linked to the new software reliability model should not be derived from an official assessment, but collected separately. Since we are not interested in a certified determination of maturity or capability levels, a questionnaire could be used for collecting the information. The questions should be specific enough to be answered by a software development or testing team member and come with predefined answer categories facilitating an automatic evaluation. The structure of the questionnaire and the areas addressed in it may be based on the process capability models for utilizing the know-how and experience

integrated in these models. Since information about the particular software produced and the course of the development project also seem to have considerable influence on the software quality, questions directed at such matters should be included in the questionnaire.

The development of the questionnaire is one of the next steps to be taken in the PETS project. Intensive participation on the part of the SMEs will be necessary for guaranteeing a successful outcome. Out of their experience, they should contribute ideas concerning the information they deem important. Moreover, they will have to state which information they will in fact be able to collect and supply for statistical evaluations.

References

[1] Bate, R.; Kuhn, D.; Wells, C.: *A Systems Engineering Capability Maturity Model, Version 1.1*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Maturity Model SECMM-95-01 CMU/SEI-95-MM-003, 1995

[2] Burnstein, I.; Homyen, A.; Grom, R.; Carlson, C. R.: *A Model to Assess Testing Process Maturity*, Crosstalk, November 1998,
URL = http://www.stsc.hill.af.mil/crosstalk/1998/nov/burnstein.pdf (site visited 2000-07-17)

[3] Burnstein, I.; Homyen, A.; Suwannasart, T.; Saxena, G.; Grom, R.: *A Testing Maturity Model for Software Test Process Assessment and Improvement*, Software Quality Professional, Vol. 1, Issue 4, 1999,
URL = http://sqp.asq.org/vol1_issue4/sqp_v1i4_burnstein.html (site visited 2001-05-21)

[4] Burnstein, I.; Suwannasart, T.; Carlson, C. R.: *Developing a Testing Maturity Model: Part I*, Crosstalk, August 1996,
URL = http://www.stsc.hill.af.mil/crosstalk/1996/aug/developi.html (site visited 1999-01-18)

[5] Burnstein, I.; Suwannasart, T.; Carlson, C. R.: *Developing a Testing Maturity Model: Part II*, Crosstalk, September 1996,
URL = http://www.stsc.hill.af.mil/crosstalk/1996/sep/developi.html (site visited 1999-01-18)

[6] CMMI Product Development Team: *CMMI ^{SM} for Systems Engineering/Software Engineering, Version 1.02 (CMMI-SE/SW, V1.02) - Staged Representation*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-2000-TR-018 ESC-TR-2000-018, 2000

[7] CMMI Product Development Team: *CMMI ^{SM} for Systems Engineering/Software Engineering, Version 1.02 (CMMI-SE/SW, V1.02) - Continuous Representation*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-2000-TR-019 ESC-TR-2000-019, 2000

[8] CMMI Product Development Team: *CMMI ^{SM} for Systems Engineering/Software Engineering/Integrated Product and Process Development, Version 1.02 - CMMI ^{SM}-SE/SW/IPPD, V1.02 - Staged Representation*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-2000-TR-030 ESC-TR-2000-095, 2000

[9] CMMI Product Development Team: *CMMI ^{SM} for Systems Engineering/Software Engineering/Integrated Product and Process Development, Version 1.02 - CMMI ^{SM}-SE/SW/IPPD, V1.02 - Continuous Representation*, Software Engineering Institute,

Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-2000-TR-031 ESC-TR-2000-096, 2000

[10] CMMI Product Development Team: *CMMI ^{SM} for Systems Engineering/Software Engineering/Integrated Product and Process Development/Acquisition, Version 1.02d DRAFT - CMMI ^{SM}-SE/SW/IPPD/A, V1.02d DRAFT - Staged Representation*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Working Draft, 2000

[11] CMMI Product Development Team: *CMMI ^{SM} for Systems Engineering/Software Engineering/Integrated Product and Process Development/Acquisition, Version 1.02d DRAFT - CMMI ^{SM}-SE/SW/IPPD/A, V1.02d DRAFT - Continuous Representation*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Working Draft, 2000

[12] Coallier, F.; Azuma, M.: *Chapter 1: Introduction to Software Engineering Standards*, in: El Emam, K.; Drouin, J.-N.; Melo, W. (ed.): *SPICE - The Theory and Practice of Software Process Improvement and Capability Determination*, Los Alamitos, Washington, et al., 1998, pp. 1 - 18

[13] Cooper, J.; Fisher, M.; Sherer, S. W. (ed.): *Software Acquisition Capability Maturity Model ® (SA-CMM ®), Version 1.02*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-99-TR-002 ESC-TR-99-002, 1999

[14] Curtis, B.; Hefley, W. E.; Miller, S.: *People Capability Maturity Model ^{SM}*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-95-MM-002, 1995

[15] Drouin, J.-N.; Barker, H.: *Chapter 2: Introduction to SPICE*, in: El Emam, K.; Drouin, J.-N.; Melo, W. (ed.): *SPICE - The Theory and Practice of Software Process Improvement and Capability Determination*, Los Alamitos, Washington, et al., 1998, pp. 19 - 30

[16] Dunaway, D. K.; Masters, S.: *CMM ^{SM} - Based Appraisal for Internal Process Improvement (CPA IPI) - Method Description*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-96-TR-007 ESC-TR-96-007, 1996

[17] Graydon, A. W.; Nevalainen, R.; Drouin, J.-N.: *Chapter 4: The Reference Model*, in: El Emam, K.; Drouin, J.-N.; Melo, W. (ed.): *SPICE - The Theory and Practice of Software Process Improvement and Capability Determination*, Los Alamitos, Washington, et al., 1998, pp. 75 - 99

[18] Grottke, M.: *Software Reliability Model Study*, Deliverable A.2 of project PETS (Prediction of software Error rates based on Test and Software maturity results), IST-1999-55017, 2001

[19] Homyen, A.; Burnstein, I.; Grom, R.: *Questionnaire for the Testing Maturity Model, Version 1.1*, Illinois Institute of Technology, 1998,
URL = http://www.csam.iit.edu/∼tmm/ (site visited 2001-06-20)

[20] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 1: Concepts and introductory guide, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part1100.doc (site visited 2001-06-11)

[21] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 2: A model for process management, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part2100.doc (site visited 2001-06-11)

[22] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 3: Rating process, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part3100.doc (site visited 2001-06-11)

[23] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 4: Guide to conducting assessment, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part4100.doc (site visited 2001-06-11)

[24] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 5: Construction, selection and use of assessment instruments and tools, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part5100.doc (site visited 2001-06-11)

[25] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 6: Qualification and training of assessors, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part6100.doc (site visited 2001-06-11)

[26] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 7: Guide for use in process improvement, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part7100.doc (site visited 2001-06-11)

[27] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 8: Guide for use in determining supplier process capability, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part8100.doc (site visited 2001-06-11)

[28] International Organization for Standardization, International Electrotechnical Commission: *Software process assessment - Part 9: Vocabulary, Version 1.00*, Working Draft, 1995,
URL = http://www.sqi.gu.edu.au/spice/docs/baseline/part9100.doc (site visited 2001-06-11)

[29] International Organization for Standardization, International Electrotechnical Commission: *Information technology - Software process assessment - Part 2: A reference model for processes and process capability*, Technical Report ISO/IEC TR 15504-2, 1998

[30] International Organization for Standardization, International Electrotechnical Commission: *Information technology - Software process assessment - Part 3: Performing an assessment*, Technical Report ISO/IEC TR 15504-3, 1998

[31] International Organization for Standardization, International Electrotechnical Commission: *Information technology - Software process assessment - Part 4: Guide to performing an assessment*, Technical Report ISO/IEC TR 15504-4, 1998

[32] International Organization for Standardization, International Electrotechnical Commission: *Information technology - Software process assessment - Part 5: An assessment model and indicator guidance*, Technical Report ISO/IEC TR 15504-5, 1998

[33] International Organization for Standardization, International Electrotechnical Commission: *Information technology - Software process assessment - Part 9: Vocabulary*, Technical Report ISO/IEC TR 15504-9, 1998

[34] Malaiya, Y. K.; Denton, J.: *What Do the Software Reliability Growth Model Parameters Represent?*, Technical Report CS-97-115, Department of Computer Science, Colorado State University, 1997,
URL = http://www.cs.colostate.edu/~ftppub/TechReports/1997/tr97-115.pdf (site visited 2001-05-31)

[35] Masters, S.; Bothwell, C.: *CMM Appraisal Framework, Version 1.0*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-95-TR-001 ESC-TR-95-001, 1995

[36] Olsen, K.; Vinje, P. S.: *Using the Testing Maturity Model ^{SM} in practical test-planning and post-evaluation*, Proc. 6th European Software Testing, Analysis and Review Conference (EuroSTAR), Munich, 1998, pp. 345 - 359

[37] Paulk, M. C.; Curtis, B.; Chrissis, M. B.; Weber, C. V.: *Capability Maturity Model ^{SM} for Software, Version 1.1*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-93-TR-24 ESC-TR-93-177, 1993

[38] Paulk, M. C.; Curtis, B.; Chrissis, M. B.; Weber, C. V.: *The Capability Maturity Model for Software*, IEEE Software, Vol. 10, No. 4, July 1993, pp. 18-27

[39] Paulk, M. C.; Weber, C. V.; Garcia, S. M.; Chrissis, M. B.; Bush, M.: *Key Practices of the Capability Maturity Model, Version 1.1*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Technical Report CMU/SEI-93-TR-25 ESC-TR-93-178, 1993

[40] Systeme Evolutif Ltd.: *Test Organisation Maturity Questionnaire V2.0*, 2000

[41] Zubrow, D.; Hayes, W.; Siegel, J.; Goldenson, D.: *Maturity Questionnaire*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Special Report CMU/SEI-94-SR-7, 1994