Chapter 2 Software Process - Standards, Assessments and Improvement

(1) **Chapter Editor**: Wolfgang Emmerich (WE)

(2) **Participants:** AF, CM, JCD

(3) Software engineering and software process improvement standards are gaining more and more attention. Why is this? First, standards are recognised by the software industry as a way for transferring good practice into industrial use. Agencies procuring software are focussing on standards as they want to be sure that a certain level of quality, associated with the standard, has been followed during the development of the software and hence has improved the quality of the software itself. Moreover, standards are used as the basis against which organisations and/or software products are certified. Finally, if two organisations that enter a cooperation for the development of a software product follow the same standards, the cooperation will be considerably simplified.

(4) In the last few years there have been a number of standardisation initiatives from which software development organisations benefit. Three different directions can be distinguished:-

a) *Definition of standard processes.* These focus on the key points to be addressed to provide an effective and well defined quality manual. ISO 9000 [ISO91], the ESA process standard PSS-05 [MFM+94], and ISO 12207 [ISO95] are some examples.

b) *Definition of assessment method.* These provide guidelines to evaluate the maturity of the process carried out by an organisation. The Software Engineering Institute's Capability Maturity Model (CMM) [Hum89], the European Bootstrap Method [KSK+94], and ISO 15504 [ISO97b] are examples of these. They are all based on some model of maturity. These models identify different maturity levels, and an assessment method, which collocates an organisation in one of the maturity levels.

c) *Definition of methods supporting the improvement* of an existing process. For instance, the Quality Improvement Paradigm [Bas93] is based on the idea that process improvement can be accomplished only if the organisation is able to learn from previous experiences. During the execution of a project some suitable measures are performed, and the collected data are analysed and packaged for future uses. Some assessment methods also provide improvement guidelines.

2.1 Standard processes

2.1.1 ISO 9000-3

(5) ISO 9000 [ISO91] supplies a set of standards for quality management of production activities. Organisations are supposed to set up a *quality system* in order to supervise all phases of a production and delivery process. Some of the activities carried out by the quality system are:-

a) auditing the projects to ensure that quality controls are respected,

b) improving the quality system itself, and

c) providing input to the development group, such as new notations, procedures, and standards; producing reports to the high-level management.

(6) The details of the quality system are contained in a *quality manual*. It contains standards for the quality and the development activities.

(7) When a new project is planned, the project manager identifies the quality issues relevant for the project and extracts them. All procedures and standards need to ensure that the defined quality levels are extracted from the quality manual and that a *quality plan* is written that identifies the means for quality control.

(8) ISO 9000 has been specialised for software production because it has been recognised as different from general purpose production processes. The result of this customisation is ISO 9000-3 [ISO91]. The basic ideas are the following:-

a) Quality control should be performed during all phases of software production, procurement and maintenance.

b) The purchaser should strictly cooperate with the software product supplier.

c) The supplier should define its quality system and ensure its entire organisation understands and implements the system.

(9) ISO 9000-3 does not impose a specific life-cycle model. It also does not provide specific methods for the evaluation of quality ensurance capabilities of organisation. It, therefore, can be coupled with more specific approaches, such as Boehm's Spiral model [Boe88].

(10) ISO 9000-3 can be used in contractual situations, when the purchaser and the supplier establish that some quality elements will be part of the suppliers quality system, and the supplier commits to follow the quality principles defined in the standard. Moreover, it can be exploited in non-contractual situations, when the supplier voluntarily applies the quality standard in order to be competitive and to ensure a good quality of its products.

2.1.2 PSS-05

(11) The European Space Agency (ESA) adopted PSS-05 [MFM+94] as a software engineering standard that is binding for all software that is either procured or developed in-house by ESA. The standard takes two different perspectives; it contains standards, guidelines and recommendations concerning the software to be defined, implemented, operated and maintained; it also determines procedures that are to be used to manage a software project.

(12) The standard is based on the concept of practices. Practices in PSS-05 are either *mandatory*, *recommended* or *guiding*. A mandatory practice must be followed without exception in all software projects. Recommended practices may or may not be followed. However, a justification has to provided if a recommended practice is abandoned. Guideline practices are less crucial; no justification is needed if they are not followed.

(13) PSS-05 does not prescribe a particular lifecycle model. However, any lifecycle model adopted for a project must be defined in a software project management plan and must include the following mandatory phases:-

a) definition of user requirements,
b) definition of software requirements,
c) definition of architectural design,
d) detailed design and production of code,
e) transfer of the software to operations, and
f) operations and maintenance.

(14) Practices related to products detail the products that are developed in these stages and how that development should be performed. The degree of abstraction of these product practices varies to a large extend. Some practices are very high-level, an example of which is: ‘‘a*ll known user requirements shall be included in the user requirements document´´*. Others are fairly concrete, such as ‘‘*for incremental delivery, each user requirement shall include a measure of priority so that the developer can decide the production schedule´´*.

(15) Practices related to management procedures are of concern throughout the different phases identified earlier. The purpose of these is to ensure that projects are managed in such a way that the product is built within budget, according to schedule and with the required quality. The standard recommends the mandatory definition of plans for:-

a) software project management,
b) software configuration management,
c) software verification and validation, and
d) software quality assurance.

(16) To date, the use of PSS-05 is not confined to ESA and its contractors. It is being used in the automobile industry and for procurements in various European defense projects. Currently, PSS-05 is being used as one basis for the ISO 15288, a new system engineering standard [ISO97a].

2.1.3 ISO-12207

(17) ISO 9000 is a standard for quality management and improvement, but it provides little concrete guidance as to how software engineering processes should be performed. The ISO standard 12207 “software life cycle processes” [ISO95] is more concrete in that it identifies mandatory processes, tasks and activities for software life cycles. Unlike PSS-05, which has been explicitly defined for one particular domain, ISO-12207 is intended to be applicable to software development in a broad range of application domains and a variety of different software systems.

(18) Any ISO standard contains a normative and an informative component. The normative component in ISO-12207 determines mandatory practices that ought to be followed for a particular development effort to be compliant to the standard. The informative component of the standard identifies rationales for the practices required by the standard and explains their application.

(19) ISO-12207 covers the entire lifecycle from “conceptualization of ideas through retirement”. It considers the software life cycle from different levels of abstraction. At the highest level, it identifies a number of *processes*. Three different categories of processes are identified:

a) *primary life cycle* processes are conducted by prime parties, i.e. those that initiate or perform the development, operation or maintenance,

b) *supporting life cycle* processes, for instance configuration management, support primary processes in order to contribute to the success and quality of the project, and

c) *organizational life cycle* processes that are employed by an organisation to establish and implement the underlying structure of the life cycle, such as management and process improvement.

(20) Processes are decomposed into activities. The acquisition process, for instance, is decomposed into activities for initiation, request for tender, contract preparation, supplier monitoring, and acceptance and completion. Activities are further decomposed into tasks. The request for tender preparation activity, for instance encompasses tasks determining system requirements, scope of the system, instructions for bidders, list of software products, terms and conditions, control of subconstracts and technical constraints.

(21) As the standard is meant to be applicable in many different domains it covers a variety of processes, activities and tasks. In order to define customisations of the standard to a particular domain, organisation or individual project, the normative part of the standard includes a ’tailoring process’. It defines how the standard is to be adapted and indicates those processes, activities and tasks that might be omitted. Moreover, it allows processes, activities and tasks to be added, provided that they are specified in a way compliant to the standard. The ability to tailor the standard allows its application in a number of different settings, such as waterfall, evolutionary, incremental and spiral process models.

(22) Hence, the overall process model defined by ISO-12207 is decomposed in a functional way into a three level hierarchy consisting of processes, activities and tasks. A weakness of the process definition is that it only implicitly identifies the products that are to be produced

during processes, activities and tasks. Moreover, coordination and dependencies between tasks are only implicitly defined. Finally, tasks, such as systems requirements, that are considered as atomic in the decomposition are still rather coarse-grained.

2.2 Assessment Methods

2.2.1 The Capability Maturity Model

(23) In the late 80s, the Software Engineering Institute has started to work on software process assessment. The work is motivated by the observation that the first step for consolidating and improving processes is to assess them [Hum89]. The SEI supplies two different programs [PWCC95]

a) *Software-process Assessment Program*. It is directed to those organisations that want to evaluate their process in order to improve it.

b) *Software Capability Evaluation Program*. It can be used by customers (in particular, US government agencies) to assess the processes and maturity levels of their contractors.

(24) These two programs share, to a great extent, the same assessment method. In the first case, the result of the assessment program is a document that provides the organisation with some suggestions on how to conduct process improvement. In the second case, a grade ranging on an ordinal scale from 1 to 5 is calculated. This grade quantifies the maturity level of the organisation. In general, in the first case the assessment is self-performed by the organisation, possibly under the assistance of the SEI, while in the second case, the assessment is performed by an external independent team from the government or the customer.

(25) In order to assess the maturity of the organisation and to identify the issues to be addressed for improving the process, the SEI defined a maturity model, called the *Capability Maturity Model* (CMM). This model defines five levels of maturity for the software industries. It then identifies a set of characteristics organisations at each level are considered to have. Moreover, a set of goals is defined that organisations should pursue for reaching the next level.

a) The lowest level is the *initial* level. Success of organisations at this maturity level depends on the skills and individual efforts of developers rather than on properly defined and managed processes.

b) At the second level processes are *repeatable*. At this level of maturity, organisations establish project management policies and procedures to carry out a project. A quality assurance function controls that the policies and the procedures are being practised. This discipline ensures repeatability of earlier success on similar projects.

c) At the third level (the *defined* level) a standard software process is defined. It defines project management and software engineering processes, and it is tailored to each project. An organisation adopting and tailoring, for instance PSS-05 or ISO 12207, would be at this level.

d) At the fourth level (the *managed* level) the process and product quality is measured, predictable and quantifiable. By using these measures, the managers can identify the causes of exceptional events and can correct the situation.

e) At the fifth level (the *optimizing* level) the process is continously improved on the basis of quantitative feedback from earlier instantiations of the process. Process improvement is obtained by introducing new methods and new technologies, and it is planned like the ordinary management activities.

(26) The Software-process Assessment Program starts with training the assessment team. After that, the assessment team selects some representative project of the organisation to be assessed. The members of the selected projects will complete the SEI questionnaire, and will be interviewed by the assessment team. The team uses the questionaire and the interview to prepare a report, which identifies weaknesses of the organisation. The solution and guidelines for its implementation are traced. To obtain a good result from the assessment, high-level management needs to endorse the assessment. In this way, people participating in the assessment will be encouraged by the fact that their suggestions will be taken into account and will influence the actual improvement of the development process.

(27) The main drawback of the Software Capability Evaluation program is that it tends to oversimplify an organisation by constraining it into a five level classification. The classification itself is based on common sense, but does not have a scientific foundation. The algorithm used to evaluate the scores is based on the idea that an organisation, which holds some characteristics of an higher level, cannot profit from these characteristics if it does not have all the characteristics of the lower levels. However, even in its simplicity, the CMM constitutes the most interesting attempt to analyse software processes that has been developed so far.

2.2.2 Bootstrap

(28) The European Union funded Bootstrap project started in 1989. Its mission was to fertilize the grounds for introducing modern Software Technology into industry [KSK+94].

(29) The project was devoted to defining a framework for assessing European industries in different sectors, such as banking, insurance and administration, and for promoting process improvement. The basic idea (that is also found in the SEI approach) is that technological innovation is not effective if it is not coupled with a careful definition of the methods used for building solutions, and if it is not carried out within a well organised process. Basically, Bootstrap is an improvement of the SEI approach to process assessment and improvement, which takes some ideas from the ISO 9000-3 standard into account. In particular, Bootstrap supplies:

a) A detailed hierarchy of process quality attributes, based on the ISO 9000-3 guidelines on quality management.

b) An enhanced version of the SEI questionnaire.

c) A method, refined from the CMM, for assessing the maturity level of an organisation.

(30) While the SEI questionnaire offers only yes/no answers, the Bootstrap approach provides four different choices: absent/weak, basic/present, significant/fair, and extensive/com-

plete.[1] Moreover, the Bootstrap questionnaire is based on the defined hierarchy of quality attributes. The Bootstrap method for obtaining the score of an organisation is more flexible than the SEI method. It is based on the same five maturity levels, but its goal is not to compute a unique score for the organization, instead, it allows to evaluate a level of maturity for each quality attribute. This way, organisations can identify the weaknesses of their process and can concentrate in fixing them.

(31) The Bootstrap assessment method can be used also to evaluate if an organization is ready to obtain the ISO 9000 certification. There is not a specific maturity level that is a good point to be certified. ISO 9000, in fact, requires that organizations use some statistical techniques that are for level 4. On the other hand, it does not require the use of specific methodologies, nor specifies how efficient they have to be. Organisations only have to prove that they have some methodologies and use them. For these reasons, an organisation that is between levels two and three could obtain the certification.

2.2.3 SPICE

(32) SPICE (Software Process Improvement and Capability dEtermination) [Rou95] is a project granted by the International Committee on Software Engineering Standards ISO/IEC JTC 1/SC7. Its goal is to build an international standard for software process assessment, covering development, acquisition, management, customer support and quality, and also human concerns and technology transfer. It is based on knowledge acquired using existing assessment methods, like CMM, Bootstrap, ISO 9000-3. The result of the project is the new ISO 15504 [ISO97b] standard. It comprises a set of documents that will guide the :-

a) high level definition of goals and fundamental activities that characterise good software engineering, graded according to levels of capability,

b) training of the assessors, e.g .by establishing the procedures for assessors' qualification,

c) process assessment and improvement phases,

d) determination of the capability of an organisation, based on the results of the assessment,

e) understanding of business risks considering the development of a new software product or service, and

f) generation of target profiles and maturity models for process improvement.

2.2.4 Summary

(33) The approaches we have discussed here are used for assessing the capabilities and maturity of individual engineers or organisations. The assessments are intended to provide feedback as to how to improve processes. Kellner et. al. suggest in [KBO96] that process improvement processes are cyclic in itself and that they have their foundation in the Shewhart cycle [She39], further developed by Deming [Dem94].

1. This idea has been partially taken into account in the new version of the SEI questionnaire, in which answers like: "I do not know", and "the question is not applicable" are allowed.

(34) The SEI has developed an organisational reference model for cyclic software process improvement initiatives [McF96]. This model consists of five phases: Initiating, Diagnosing, Establishing, Acting and Leveraging and is referred to as the IDEAL model. The IDEAL model can express the assessment orientated methods we have discussed in this subsection as well as the improvement methods that we discuss in the next section.

2.3 Improvement methods

2.3.1 Quality Improvement Paradigm

(35) The Quality Improvement Paradigm [BCM+92, Bas93] has been proposed by the SEL (Software Engineering Laboratory) of The University of Maryland to perform process improvement. The basic idea is that improvement is a continuous process, that is composed of the following steps:-

a) characterisation of the project and its environment ,

b) planninga set of goals and the appropriate process models, methods, and tools for achieving these goals,

c) execution of the process according to the defined goals, development of the product , and collection and analysis of data for feedback purposes.

d) analysis and packaging of data collected at the end of the project for use in future projects.

(36) Two tools are used for performing the steps described above: the Goal/Question/Metric (GQM), and the Experience Factory Organization. GQM is during the planning phase. It facilitates the definition of goals and suitable metrics for each of them. These will guide the execution of the process. The Experience Factory Organization is an organizational structure that supports the packaging activities.

(37) The quality improvement paradigm teaches how to set-up a continuous improvement process, taking into account previous experiences. It is based on the assumption that the organisation is able to define a process model, is confident with tools and procedures for collecting metrics, manages a repository of data from previous project, and so on. Therefore, the quality improvement paradigm can be profitably used by mature organisations, which are already aware of their process, in order to become more sensitive to the lessons learnt on the way.

2.3.2 The Personal Software Process

(38) The Capability Maturity Model evaluates and rates the maturity of software processes at an organisational level. However, an important factor, which determines the productivity and quality of products, is the maturity and capability of the individual developer. Motivated by the success of the CMM, the SEI has recently started to work on the Personal Software Process (PSP) [Hum95,Hum96,Hum97]. The PSP is aimed at guiding individual engineers to improve their productivity and quality of the overall processes they contribute to. Engineers are introduced to the PSP by means of developing ten small exercise programs.

(39) The PSP defines four levels of maturity and identifies the steps needed for reaching the next higher maturity level:-

a) PSP0 - *Personal Measurement*

b) PSP1 - *Personal Planning*

c) PSP2 - *Personal Quality*

d) PSP3 - *Cyclic Process*

(40) Many considerations of the CMM also apply to the PSP. In order to improve the maturity of individual developers, their performance has to be assessed in the first place. During PSP0, engineers are taught how to measure their development time and the defects they have injected and removed. Moreover, in PSP0 engineers are introduced to coding standards, size measurements and a form for proposing improvements to their personal process.

(41) At level PSP1, engineers learn techniques for estimating size and development time on the basis of data they have gathered during PSP0. Moreover, they learn how to plan tasks and schedules.

(42) PSP2 introduces the management of defects. At lower levels, engineers will have gathered defect data. In PSP2, engineers use this data to construct checklists in order to identify those defects they are likely to make. Engineers then use the checklist to consciously review their designs and their code against them. Engineers, therefore, learn that an individual focus on quality is important. By using these checklists, their skills will improve.

(43) In order to scale the approach from the small examples used in the exercises to real projects, the PSP has to be applied in a cyclic manner. A constant monitoring of injected and removed defects and constant reviews of the defect checklist is supposed to lead to improvements of personal quality.

(44) The PSP has been applied on a number of undergraduate, graduate and industrial courses. The results below have been obtained by 104 engineers, half of which from industrial software organisations. The overall average in productivity improvement (measured in lines of code per hour) is reported in [Hum96] to be about 20%. However, a notable variance was observed in these productivity improvements, in particular between undergraduates and more experienced engineers. While some undergraduates improved their productivity by as much as 420%, some mature engineers experienced performance degradations due to the additional overheads of estimations and reporting introduced through the PSP.

(45) Humphrey reports that quality, measured in terms of the number of defects injected, improved irregardless of the experience levels of engineers. On average, the total numbers of defects during his PSP trials decreased from 140 defects per KLOC in the first programming exercise to 49 defects/KLOC. Defects found during compiling decreased from 80 to 12 defects/KLOC. The decrease rates, were higher for students without industrial experience, but even for engineers with 20 years of experience quality improved.

(46) To summarise, the PSP is an interesting approach for improving quality and productivity of students and young engineers. This improvement is achieved through teaching the explicit focus on the process that is used for developing software. The SEI has provided imperical

evidence that the PSP improves both productivity and quality, which justifies teaching the PSP to students in both academia and industry.

2.3.3 Total Quality Management

(47) Total Quality Management is not a specific and pre-defined methodology for process improvement. It is a paradigm that guides organisations in focusing on quality [Fei91]. Its tenets are that quality is not only related to the product, but also to the organisation, to the production process that is carried out. Moreover, it argues that quality achievement is obtained if all the people involved in an organisation work to obtain it: quality is excellence in all the aspects of an organisation. The management of quality causes a continuous and never-ending process of improvement. Process improvement can be carried out by performing big improvement steps followed by long periods in which no change is done, or by performing many little steps. The first approach is known as the Western approach. American managers, in fact, tend to achieve dramatic jumps by introducing new technologies, and employing big amounts of money in research and development. The latter approach is the Japanese approach, or *kaizen*. Japanese people try to achieve little daily improvements, by using common sense and simple technology.

(48) TQM can be pursued using both approaches. However, a powerful approach can be obtained combining these two. So, big jumps are done through innovations, and between two consecutive jumps incremental changes are performed.

2.4 Standards and Software Process Technology

(49) Software engineering standards, such as ISO-12207 and the ESA PSS-05 standard are rather general. They are written in a way that they can be customised towards the need of a particular organisation or project. ISO-12207 even explictly defines the process with which it is supposed to be customised. This refinement and customisation of a standard can well be considered as a process modelling activity and we believe that the process modelling principles, methods and techniques discussed in the following chapters of this book are applicable to the problem of refining standards.

(50) Software engineering standards only provides normative references. These are not directly applicable in a software engineering process. In order to guide software engineers in following standards they have to, at least partly, be automated. This automation should help engineers to comply to the standards, check whether the documents are structured in a way prescribed by the standards and enable the cooperation between different engineers in a way as it is prescribed by the standard. The architecture of an environment needed for such an automation will be very similar to those discussed for process-centred environments later in this book.

(51) The standards we have looked at are all defined in natural language. Often the standard is defined following a rigid structure and practices to be followed are explicitly highlighted. However, as one would expect from a natrual language text, the standards mentioned in this chapter are all rather informal, ambigous and inherently inconsistent. We believe that any attempt to automate the handling of standards in software engineering environments demands the formalisation of these standards. Again, the process modelling languages dis-

cussed in this book, and also the previous Promoter book, are highly applicable to this formalisation problem.