CHAPTER 8

SOFTWARE ENGINEERING MANAGEMENT

ACRONYM

PMBOK	Guide to the Project Management Body of Knowledge
SQA	Software Quality Assurance

INTRODUCTION

Software Engineering Management can be defined as the application of management activities—planning, coordinating, measuring, monitoring, controlling, and reporting—to ensure that the development and maintenance of software is systematic, disciplined, and quantified (IEEE610.12-90).

The Software Engineering Management KA therefore addresses the management and measurement of software engineering. While measurement is an important aspect of all KAs, it is here that the topic of measurement programs is presented.

While it is true to say that in one sense it should be possible to manage software engineering in the same way as any other (complex) process, there are aspects specific to software products and the software life cycle processes which complicate effective management—just a few of which are as follows:

- The perception of clients is such that there is often a lack of appreciation for the complexity inherent in software engineering, particularly in relation to the impact of changing requirements.
- It is almost inevitable that the software engineering processes themselves will generate the need for new or changed client requirements.
- As a result, software is often built in an iterative process rather than a sequence of closed tasks.
- Software engineering necessarily incorporates aspects of creativity and discipline—maintaining an appropriate balance between the two is often difficult.
- The degree of novelty and complexity of software is often extremely high.
- There is a rapid rate of change in the underlying technology.

With respect to software engineering, management activities occur at three levels: organizational and infrastructure management, project management, and measurement program planning and control. The last two are covered in detail in this KA description. However, this is not to diminish the importance of organizational management issues.

Since the link to the related disciplines—obviously management—is important, it will be described in more detail than in the other KA descriptions. Aspects of organizational management are important in terms of their impact on software engineering—on policy management, for instance: organizational policies and standards provide the framework in which software engineering is undertaken. These policies may need to be influenced by the requirements of effective software development and maintenance, and a number of software engineering-specific policies may need to be established for effective management of software engineering at an organizational level. For example, policies are usually necessary to establish specific organization-wide processes or procedures for such software engineering tasks as designing, implementing, estimating, tracking, and reporting. Such policies are essential to effective long-term software engineering management, by establishing a consistent basis on which to analyze past performance and implement improvements, for example.

Another important aspect of management is personnel management: policies and procedures for hiring, training, and motivating personnel and mentoring for career development are important not only at the project level but also to the longer-term success of an organization. Software engineering personnel may present unique training or personnel management challenges (for example, maintaining currency in a context where the underlying technology undergoes continuous and rapid change). Communication management is also often mentioned as an overlooked but major aspect of the performance of individuals in a field where precise understanding of user needs and of complex requirements and designs is necessary. Finally, portfolio management, which is the capacity to have an overall vision not only of the set of software under development but also of the software already in use in an organization, is necessary. Furthermore, software reuse is a key factor in maintaining and improving productivity and competitiveness. Effective reuse requires a strategic vision that reflects the unique power and requirements of this technique.

In addition to understanding the aspects of management that are uniquely influenced by software, software engineers must have some knowledge of the more general aspects, even in the first four years after graduation that is targeted in the Guide.

Organizational culture and behavior, and functional enterprise management in terms of procurement, supply chain management, marketing, sales, and distribution, all have an influence, albeit indirectly, on an organization's software engineering process.

Relevant to this KA is the notion of project management, as "the construction of useful software artifacts" is normally managed in the form of (perhaps programs of) individual projects. In this regard, we find extensive support in the Guide to the Project Management Body of Knowledge (PMBOK) (PMI00), which itself includes the following project management KAs: project integration management, project scope management, project time management, project cost management, project quality management, project human resource management, and project communications management. Clearly, all these topics have direct relevance to the Software Engineering Management KA. To attempt to duplicate the content of the Guide to the PMBOK here would be both impossible and inappropriate. Instead, we suggest that the reader interested in project management beyond what is specific to software engineering projects consult the PMBOK itself. Project management is also found in the Related Disciplines of Software Engineering chapter.

The Software Engineering Management KA consists of both the software project management process, in its first five subareas, and software engineering measurement in the last subarea. While these two subjects are often regarded as being separate, and indeed they do possess many unique aspects, their close relationship has led to their combined treatment in this KA. Unfortunately, a common perception of the software industry is that it delivers products late, over budget, and of poor quality and uncertain functionality. Measurement-informed management — an assumed principle of any true engineering discipline — can help to turn this perception around. In essence, management without measurement, qualitative and quantitative, suggests a lack of rigor, and measurement without management suggests a lack of purpose or context. In the same way, however, management and measurement without expert knowledge is equally ineffectual, so we must be careful to avoid over-emphasizing the quantitative aspects of Software Engineering Management (SEM). Effective management requires a combination of both numbers and experience.

The following working definitions are adopted here:

- *Management process* refers to the activities that are undertaken in order to ensure that the software engineering processes are performed in a manner consistent with the organization's policies, goals, and standards.
- *Measurement* refers to the assignment of values and labels to aspects of software engineering (products, processes, and resources as defined by [Fen98]) and the models that are derived from them, whether these models are developed using statistical, expert knowledge or other techniques.

The software engineering project management subareas make extensive use of the software engineering measurement subarea.

Not unexpectedly, this KA is closely related to others in the Guide to the SWEBOK, and reading the following KA descriptions in conjunction with this one would be particularly useful.

- *Software Requirements*, where some of the activities to be performed during the Initiation and Scope definition phase of the project are described
- *Software Configuration Management***,** as this deals with the identification, control, status accounting, and audit of the software configuration along with software release management and delivery
- *Software Engineering Process***,** because processes and projects are closely related (this KA also describes process and product measurement)
- *Software Quality***,** as quality is constantly a goal of management and is an aim of many activities that must be managed

BREAKDOWN OF TOPICS FOR SOFTWARE ENGINEERING MANAGEMENT

As the Software Engineering Management KA is viewed here as an organizational process which incorporates the notion of process and project management, we have created a breakdown that is both topic-based and life cycle-based. However, the primary basis for the top-level breakdown is the process of managing a software engineering project. There are six major subareas. The first five subareas largely follow the IEEE/EIA 12207 Management Process. The six subareas are:

- *Initiation and scope definition*, which deals with the decision to initiate a software engineering project
- *Software project planning*, which addresses the activities undertaken to prepare for successful software engineering from a management perspective
- *Software project enactment*, which deals with generally accepted software engineering management activities that occur during software engineering
- *Review and evaluation*, which deal with assurance that the software is satisfactory
- *Closure*, which addresses the post-completion activities of a software engineering project
- *Software engineering measurement*, which deals with the effective development and implementation of measurement programs in software engineering organizations (IEEE12207.0-96)

The breakdown of topics for the Software Engineering Management KA is shown in Figure 1.

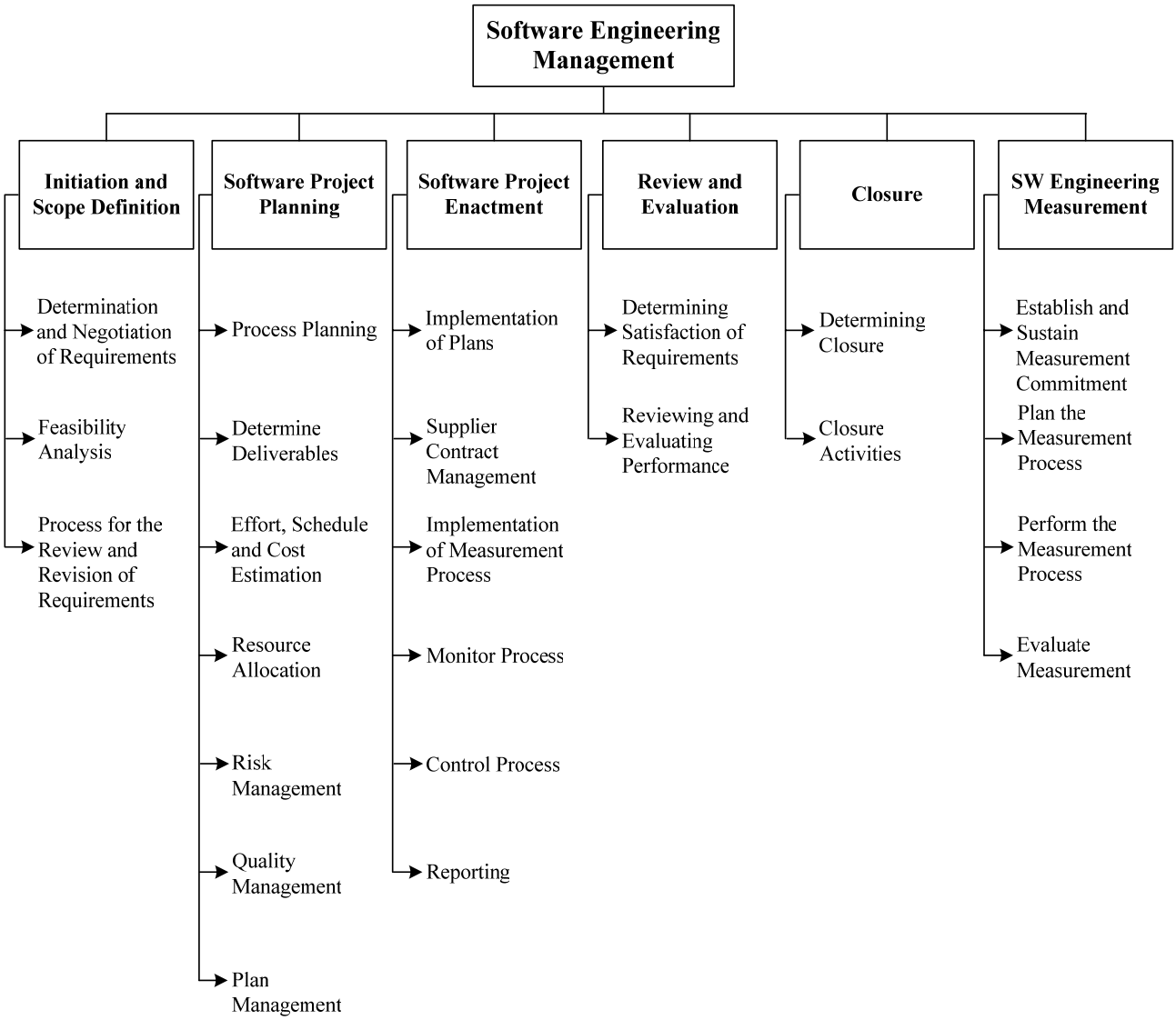

Figure 1 Breakdown of topics for the Software Engineering Management KA

1. Initiation and Scope Definition

The focus of this set of activities is on the effective determination of software requirements via various elicitation methods and the assessment of the project's feasibility from a variety of standpoints. Once feasibility has been established, the remaining task within this process is the specification of requirements validation and change procedures (see also the Software Requirements KA).

1.1. Determination and Negotiation of Requirements

[Dor02: v2c4; Pfl01: c4; Pre04: c7; Som05: c5]

Software requirement methods for requirements elicitation (for example, observation), analysis (for example, data modeling, use-case modeling), specification, and validation (for example, prototyping) must be selected and applied, taking into account the various stakeholder perspectives. This leads to the determination of project scope, objectives, and constraints. This is always an important activity, as it sets the visible boundaries for the set of tasks being undertaken, and is particularly so where the novelty of the undertaking is high. Additional information can be found in the Software Requirements KA.

1.2. Feasibility Analysis (Technical, Operational, Financial, Social/Political)

[Pre04: c6; Som05: c6]

Software engineers must be assured that adequate capability and resources are available in the form of people, expertise, facilities, infrastructure, and support (either internally or externally) to ensure that the project can be successfully completed in a timely and cost-effective manner (using, for example, a requirement-capability matrix). This often requires some "ballpark" estimation of effort and cost based on appropriate methods (for example, expert-informed analogy techniques).

1.3. Process for the Review and Revision of Requirements

Given the inevitability of change, it is vital that agreement among stakeholders is reached at this early point as to the means by which scope and requirements are to be reviewed and revised (for example, via agreed change management procedures). This clearly implies that scope and requirements will not be "set in stone" but can and should be revisited at predetermined points as the process unfolds (for example, at design reviews, management reviews). If changes are accepted, then some form of traceability analysis and risk analysis (see topic 2.5 *Risk Management*) should be used to ascertain the impact of those changes. A managed-change approach should also be useful when it comes time to review the outcome of the project, as the scope and requirements should form the basis for the evaluation of success. [Som05: c6] See also the software configuration control subarea of the Software Configuration Management KA.

2. Software Project Planning

The iterative planning process is informed by the scope and requirements and by the establishment of feasibility. At this point, software life cycle processes are evaluated and the most appropriate (given the nature of the project, its degree of novelty, its functional and technical complexity, its quality requirements, and so on) is selected. Where relevant, the project itself is then planned in the form of a hierarchical decomposition of tasks, the associated deliverables of each task are specified and characterized in terms of quality and other attributes in line with stated requirements, and detailed effort, schedule, and cost estimation is undertaken. Resources are then allocated to tasks so as to optimize personnel productivity (at individual, team, and organizational levels), equipment and materials utilization, and adherence to schedule. Detailed risk management is undertaken and the "risk profile" of the project is discussed among, and accepted by, all relevant stakeholders. Comprehensive software quality management processes are determined as part of the planning process in the form of procedures and responsibilities for software quality assurance, verification and validation, reviews, and audits (see the Software Quality KA). As an iterative process, it is vital that the processes and responsibilities for ongoing plan management, review, and revision are also clearly stated and agreed.

2.1. Process Planning

Selection of the appropriate software life cycle model (for example, spiral, evolutionary prototyping) and the adaptation and deployment of appropriate software life cycle processes are undertaken in light of the particular scope and requirements of the project. Relevant methods and tools are also selected. [Dor02: v1c6,v2c8; Pfl01: c2; Pre04: c2; Rei02: c1,c3,c5; Som05: c3; Tha97: c3] At the project level, appropriate methods and tools are used to decompose the project into tasks, with associated inputs, outputs, and completion conditions (for example, work breakdown structure). [Dor02: v2c7; Pfl01: c3; Pre04: c21; Rei02: c4,c5; Som05: c4; Tha97: c4,c6] This in turn influences decisions on the project's high-level schedule and organization structure.

2.2. Determine Deliverables

The product(s) of each task (for example, architectural design, inspection report) are specified and characterized. [Pfl01: c3; Pre04: c24; Tha97: c4] Opportunities to reuse software components from previous developments or to utilize off-the-shelf software products are evaluated. Use of third parties and procured software are planned and suppliers are selected.

2.3. Effort, Schedule, and Cost Estimation

Based on the breakdown of tasks, inputs, and outputs, the expected effort range required for each task is determined using a calibrated estimation model based on historical size-effort data where available and relevant, or other methods like expert judgment. Task dependencies are established and potential bottlenecks are identified using suitable methods (for example, critical path analysis). Bottlenecks are resolved where possible, and the expected schedule of tasks with projected start times, durations, and end times is produced (for example, PERT chart). Resource requirements (people, tools) are translated into cost estimates. [Dor02: v2c7; Fen98: c12; Pfl01: c3; Pre04: c23, c24; Rei02: c5,c6; Som05: c4,c23; Tha97: c5] This is a highly iterative activity which must be negotiated and revised until consensus is reached among affected stakeholders (primarily engineering and management).

2.4. Resource Allocation

[Pfl01: c3; Pre04: c24; Rei02: c8,c9; Som05: c4;

Tha97: c6,c7]

Equipment, facilities, and people are associated with the scheduled tasks, including the allocation of responsibilities for completion (using, for example, a Gantt chart). This activity is informed and constrained by the availability of resources and their optimal use under these circumstances, as well as by issues relating to personnel (for example, productivity of individuals/teams, team dynamics, organizational and team structures).

2.5. Risk Management

Risk identification and analysis (what can go wrong, how and why, and what are the likely consequences), critical risk assessment (which are the most significant risks in terms of exposure, which can we do something about in terms of leverage), risk mitigation and contingency planning (formulating a strategy to deal with risks and to manage the risk profile) are all undertaken. Risk assessment methods (for example, decision trees and process simulations) should be used in order to highlight and evaluate risks. Project abandonment policies should also be determined at this point in discussion with all other stakeholders. [Dor02: v2c7; Pfl01: c3; Pre04: c25; Rei02: c11; Som05: c4; Tha97: c4] Software-unique aspects of risk, such as software engineers' tendency to add unwanted features or the risks attendant in software's intangible nature, must influence the project's risk management.

2.6. Quality Management

[Dor02: v1c8,v2c3-c5; Pre04: c26; Rei02: c10;

Som05: c24,c25; Tha97: c9,c10]

Quality is defined in terms of pertinent attributes of the specific project and any associated product(s), perhaps in both quantitative and qualitative terms. These quality characteristics will have been determined in the specification of detailed software requirements. See also the Software Requirements KA.

Thresholds for adherence to quality are set for each indicator as appropriate to stakeholder expectations for the software at hand. Procedures relating to ongoing SQA throughout the process and for product (deliverable) verification and validation are also specified at this stage (for example, technical reviews and inspections) (see also the Software Quality KA).

2.7. Plan Management

[Som05: c4; Tha97: c4]

How the project will be managed and how the plan will be managed must also be planned. Reporting, monitoring, and control of the project must fit the selected software engineering process and the realities of the project, and must be reflected in the various artifacts that will be used for managing it. But, in an environment where change is an expectation rather than a shock, it is vital that plans are themselves managed. This requires that adherence to plans be systematically directed, monitored, reviewed, reported, and, where appropriate, revised. Plans associated with other management-oriented support processes (for example, documentation, software configuration management, and problem resolution) also need to be managed in the same manner.

3. Software Project Enactment

The plans are then implemented, and the processes embodied in the plans are enacted. Throughout, there is a focus on adherence to the plans, with an overriding expectation that such adherence will lead to the successful satisfaction of stakeholder requirements and achievement of the project objectives. Fundamental to enactment are the ongoing management activities of measuring, monitoring, controlling, and reporting.

3.1. Implementation of Plans

[Pfl01: c3; Som05: c4]

The project is initiated and the project activities are undertaken according to the schedule. In the process, resources are utilized (for example, personnel effort, funding) and deliverables are produced (for example, architectural design documents, test cases).

3.2. Supplier Contract Management

[Som05:c4]

Prepare and execute agreements with suppliers, monitor supplier performance, and accept supplier products, incorporating them as appropriate.

3.3. Implementation of measurement process

[Fen98: c13,c14; Pre04: c22; Rei02: c10,c12;
Tha97: c3,c10]

The measurement process is enacted alongside the software project, ensuring that relevant and useful data are collected (see also topics 6.2 *Plan the Measurement Process* and 6.3 *Perform the Measurement Process*).

3.4. Monitor Process

[Dor02: v1c8, v2c2-c5,c7; Rei02: c10;

Som05: c25; Tha97: c3;c9]

Adherence to the various plans is assessed continually and at predetermined intervals. Outputs and completion conditions for each task are analyzed. Deliverables are evaluated in terms of their required characteristics (for example, via reviews and audits). Effort expenditure, schedule adherence, and costs to date are investigated, and resource usage is examined. The project risk profile is revisited, and adherence to quality requirements is evaluated.

Measurement data are modeled and analyzed. Variance analysis based on the deviation of actual from expected outcomes and values is undertaken. This may be in the form of cost overruns, schedule slippage, and the like. Outlier identification and analysis of quality and other measurement data are performed (for example, defect density analysis). Risk exposure and leverage are recalculated, and decisions trees, simulations, and so on are rerun in the light of new data. These activities enable problem detection and exception identification based on exceeded thresholds. Outcomes are reported as needed and certainly where acceptable thresholds are surpassed.

3.5. Control Process

[Dor02: v2c7; Rei02: c10; Tha97: c3,c9]

The outcomes of the process monitoring activities provide the basis on which action decisions are taken. Where

appropriate, and where the impact and associated risks are modeled and managed, changes can be made to the project. This may take the form of corrective action (for example, retesting certain components), it may involve the incorporation of contingencies so that similar occurrences are avoided (for example, the decision to use prototyping to assist in software requirements validation), and/or it may entail the revision of the various plans and other project documents (for example, requirements specification) to accommodate the unexpected outcomes and their implications.

In some instances, it may lead to abandonment of the project. In all cases, change control and software configuration management procedures are adhered to (see also the Software Configuration Management KA), decisions are documented and communicated to all relevant parties, plans are revisited and revised where necessary, and relevant data is recorded in the central database (see also topic 6.3 *Perform the Measurement Process*).

3.6. Reporting

[Rei02: c10; Tha97: c3,c10]

At specified and agreed periods, adherence to the plans is reported, both within the organization (for example to the project portfolio steering committee) and to external stakeholders (for example, clients, users). Reports of this nature should focus on overall adherence as opposed to the detailed reporting required frequently within the project team.

4. Review and Evaluation

At critical points in the project, overall progress towards achievement of the stated objectives and satisfaction of stakeholder requirements are evaluated. Similarly, assessments of the effectiveness of the overall process to date, the personnel involved, and the tools and methods employed are also undertaken at particular milestones.

4.1. Determining Satisfaction of Requirements

[Rei02: c10; Tha97: c3,c10]

Since attaining stakeholder (user and customer) satisfaction is one of our principal aims, it is important that progress towards this aim be formally and periodically assessed. This occurs on achievement of major project milestones (for example, confirmation of software design architecture, software integration technical review). Variances from expectations are identified and appropriate action is taken. As in the control process activity above (see topic 3.5 *Control Process*), in all cases change control and software configuration management procedures are adhered to (see the Software Configuration Management KA), decisions are documented and communicated to all relevant parties, plans are revisited and revised where necessary, and relevant data are recorded in the central database (see also topic 6.3 *Perform the Measurement Process*). More information can also be found in the Software Testing KA, in topic 2.2 *Objectives of Testing* and in the Software Quality KA, in topic 2.3 *Reviews and Audits*.

4.2. Reviewing and Evaluating Performance

[Dor02: v1c8,v2c3,c5; Pfl01: c8,c9; Rei02: c10; Tha97: c3,c10]

Periodic performance reviews for project personnel provide insights as to the likelihood of adherence to plans as well as possible areas of difficulty (for example, team member conflicts). The various methods, tools, and techniques employed are evaluated for their effectiveness and appropriateness, and the process itself is systematically and periodically assessed for its relevance, utility, and efficacy in the project context. Where appropriate, changes are made and managed.

5. Closure

The project reaches closure when all the plans and embodied processes have been enacted and completed. At this stage, the criteria for project success are revisited. Once closure is established, archival, post mortem, and process improvement activities are performed.

5.1. Determining Closure

[Dor02: v1c8,v2c3,c5; Rei02: c10; Tha97: c3,c10]

The tasks as specified in the plans are complete, and satisfactory achievement of completion criteria is confirmed. All planned products have been delivered with acceptable characteristics. Requirements are checked off and confirmed as satisfied, and the objectives of the project have been achieved. These processes generally involve all stakeholders and result in the documentation of client acceptance and any remaining known problem reports.

5.2. Closure Activities

[Pfl01: c12; Som05: c4]

After closure has been confirmed, archival of project materials takes place in line with stakeholder-agreed methods, location, and duration. The organization's measurement database is updated with final project data and post-project analyses are undertaken. A project post mortem is undertaken so that issues, problems, and opportunities encountered during the process (particularly via review and evaluation, see subarea 4 *Review and evaluation*) are analyzed, and lessons are drawn from the process and fed into organizational learning and improvement endeavors (see also the Software Engineering Process KA).

6. Software Engineering Measurement

[ISO 15939-02]

The importance of measurement and its role in better management practices is widely acknowledged, and so its importance can only increase in the coming years. Effective measurement has become one of the cornerstones of organizational maturity.

Key terms on software measures and measurement methods have been defined in [ISO15939-02] on the basis of the ISO international vocabulary of metrology [ISO93]. Nevertheless, readers will encounter terminology differences in the literature; for example, the term "metrics" is sometimes used in place of "measures."

This topic follows the international standard ISO/IEC 15939, which describes a process which defines the activities and tasks necessary to implement a software measurement process and includes, as well, a measurement information model.

6.1. Establish and Sustain Measurement Commitment

- Accept requirements for measurement**.** Each measurement endeavor should be guided by organizational objectives and driven by a set of measurement requirements established by the organization and the project. For example, an organizational objective might be "first-to-market with new products." [Fen98: c3,c13; Pre04: c22] This in turn might engender a requirement that factors contributing to this objective be measured so that projects might be managed to meet this objective.
  - Define scope of measurement. The organizational unit to which each measurement requirement is to be applied must be established. This may consist of a functional area, a single project, a single site, or even the whole enterprise. All subsequent measurement tasks related to this requirement should be within the defined scope. In addition, the stakeholders should be identified.
  - Commitment of management and staff to measurement. The commitment must be formally established, communicated, and supported by resources (see next item).
- Commit resources for measurement. The organization's commitment to measurement is an essential factor for success, as evidenced by assignment of resources for implementing the measurement process. Assigning resources includes allocation of responsibility for the various tasks of the measurement process (such as user, analyst, and librarian) and providing adequate funding, training, tools, and support to conduct the process in an enduring fashion.

6.2. Plan the Measurement Process

- Characterize the organizational unit. The organizational unit provides the context for measurement, so it is important to make this context explicit and to articulate the assumptions that it embodies and the constraints that it imposes. Characterization can be in terms of organizational processes, application domains, technology, and organizational interfaces. An organizational process model is also typically an element of the organizational unit characterization [ISO15939-02: 5.2.1].
- Identify information needs. Information needs are based on the goals, constraints, risks, and problems of the organizational unit. They may be derived from business, organizational, regulatory, and/or product objectives. They must be identified and prioritized. Then, a subset to be addressed must be selected and the results documented, communicated, and reviewed by stakeholders [ISO 15939-02: 5.2.2].
- Select measures. Candidate measures must be selected, with clear links to the information needs. Measures must then be selected based on the priorities of the information needs and other criteria such as cost of collection, degree of process disruption during collection, ease of analysis, ease of obtaining accurate, consistent data, and so on [ISO15939-02: 5.2.3 and Appendix C].
- Define data collection, analysis, and reporting procedures. This encompasses collection procedures and schedules, storage, verification, analysis, reporting, and configuration management of data [ISO15939-02: 5.2.4].
- Define criteria for evaluating the information products. Criteria for evaluation are influenced by the technical and business objectives of the organizational unit. Information products include those associated with the product being produced, as well as those associated with the processes being used to manage and measure the project [ISO15939-02: 5.2.5 and Appendices D, E].
- Review, approve, and provide resources for measurement tasks.
  - The measurement plan must be reviewed and approved by the appropriate stakeholders. This includes all data collection procedures, storage, analysis, and reporting procedures; evaluation criteria; schedules; and responsibilities. Criteria for reviewing these artifacts should have been established at the organizational unit level or higher and should be used as the basis for these reviews. Such criteria should take into consideration previous experience, availability of resources, and potential disruptions to projects when changes from current practices are proposed. Approval demonstrates commitment to the measurement process [ISO15939-02: 5.2.6.1 and Appendix F].
  - Resources should be made available for implementing the planned and approved measurement tasks. Resource availability may be staged in cases where changes are to be piloted before widespread deployment. Consideration should be paid to the resources necessary for successful deployment of new procedures or measures [ISO15939-02: 5.2.6.2].
- Acquire and deploy supporting technologies. This includes evaluation of available supporting technologies, selection of the most appropriate

technologies, acquisition of those technologies, and deployment of those technologies [ISO 15939-02: 5.2.7].

6.3. Perform the Measurement Process

- Integrate measurement procedures with relevant processes. The measurement procedures, such as data collection, must be integrated into the processes they are measuring. This may involve changing current processes to accommodate data collection or generation activities. It may also involve analysis of current processes to minimize additional effort and evaluation of the effect on employees to ensure that the measurement procedures will be accepted. Morale issues and other human factors need to be considered. In addition, the measurement procedures must be communicated to those providing the data, training may need to be provided, and support must typically be provided. Data analysis and reporting procedures must typically be integrated into organizational and/or project processes in a similar manner [ISO 15939-02: 5.3.1].
- Collect data. The data must be collected, verified, and stored [ISO 15939-02 :5.3.2].
- Analyze data and develop information products. Data may be aggregated, transformed, or recoded as part of the analysis process, using a degree of rigor appropriate to the nature of the data and the information needs. The results of this analysis are typically indicators such as graphs, numbers, or other indications that must be interpreted, resulting in initial conclusions to be presented to stakeholders. The results and conclusions must be reviewed, using a process defined by the organization (which may be formal or informal). Data providers and measurement users should participate in reviewing the data to ensure that they are meaningful and accurate, and that they can result in reasonable actions [ISO 15939-02: 5.3.3 and Appendix G].
- Communicate results. Information products must be documented and communicated to users and stakeholders [ISO 15939-02: 5.3.4].

6.4. Evaluate Measurement

- Evaluate information products. Evaluate information products against specified evaluation criteria and determine strengths and weaknesses of the information products. This may be performed by an internal process or an external audit and should include feedback from measurement users. Record lessons learned in an appropriate database [ISO 15939-02: 5.4.1 and Appendix D].
- Evaluate the measurement process. Evaluate the measurement process against specified evaluation criteria and determine the strengths and weaknesses of the process. This may be performed by an internal process or an external audit and should include feedback from measurement users. Record lessons learned in an appropriate database [ISO 15939-02: 5.4.1 and Appendix D].
- Identify potential improvements. Such improvements may be changes in the format of indicators, changes in units measured, or reclassification of categories. Determine the costs and benefits of potential improvements and select appropriate improvement actions. Communicate proposed improvements to the measurement process owner and stakeholders for review and approval. Also communicate lack of potential improvements if the analysis fails to identify improvements [ISO 15939-02: 5.4.2].

MATRIX OF TOPICS VS. REFERENCE MATERIAL

	[Dor02]	[ISO15939-02]	[Fen98]	[Pfl01]	[Pre04]	[Rei02]	[Som05]	[Tha97]
1. Initiation and scope definition								
1.1 Determination and negotiation of requirements	v2c4			c4	c7		c5	
1.2 Feasibility analysis					c6		c6	
1.3 Process for the review and revision of requirements							c6	
2. Software Project Planning								
2.1 Process planning	v1c6,v2c7, v2c8			c2,c3	c2,c21	c1,c3,c5	c3,c4	c3,c4,c6
2.2 Determine deliverables				c3	c24			c4
23 Effort, schedule and cost estimation	v2c7		c12	c3	C23,c24	c5,c6	c4,c23	c5
2.4 Resource allocation				c3	c24	c8,c9	c4	c6,c7
2.5 Risk management	v2c7			c3	c25	c11	c4	c4
2.6 Quality management	v1c8,v2c3-c5				c26	c10	c24,c25	c9,c10
2.7 *Plan management*							c4	c4
3. Software Project Enactment								
3.1 Implementation of plans				c3			c4	
3.2 Supplier contract management							c4	
3.3 Implementation of measurement process			c13c,14		c22	c10,c12		c3,c10
3.4 Monitor process	v1c8,v2c2-c5,c7					c10	c25	c3,c9
3.5 Control process	v2c7					c10		c3,c9
3.6 Reporting						c10		c3,c10
4. Review and evaluation								
4.1 Determining satisfaction of requirements						c10		c3,c10
4.2 Reviewing and evaluating performance	v1c8,v2c3, c5			c8,c9		c10		c3,c10
5. Closure								
5.1 Determining closure	v1c8,v2c3, c5					c10		c3,c10
5.2 Closure activities				c12			c4	
6. Software Engineering Measurement		*						
6.1 Establish and sustain measurement commitment			c3,c13		c22			
6.2 Plan the measurement process		c5,C,D,E,F						
6.3 Perform the measurement process		c5,G						
6.4 Evaluate measurement		c5,D						

RECOMMENDED REFERENCES FOR SOFTWARE ENGINEERING MANAGEMENT

[Dor02] M. Dorfman and R.H. Thayer, eds., *Software Engineering*, IEEE Computer Society Press, 2002, Vol. 1, Chap. 6, 8, Vol. 2, Chap. 3, 4, 5, 7, 8.

[Fen98] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous & Practical Approach*, second ed., International Thomson Computer Press, 1998, Chap. 1-14.

[ISO15939-02] ISO/IEC 15939:2002, *Software Engineering — Software Measurement Process*, ISO and IEC, 2002.

[Pfl01] S.L. Pfleeger, *Software Engineering: Theory and Practice*, second ed., Prentice Hall, 2001, Chap. 2-4, 8, 9, 12, 13.

[Pre04] R.S. Pressman, *Software Engineering: A Practitioner's Approach*, sixth ed., McGraw-Hill, 2004, Chap. 2, 6, 7, 22-26.

[Rei02] D.J. Reifer, ed., *Software Management*, IEEE Computer Society Press, 2002, Chap. 1-6, 7-12, 13.

[Som05] I. Sommerville, *Software Engineering*, seventh ed., Addison-Wesley, 2005, Chap. 3-6, 23-25.

[Tha97] R.H. Thayer, ed., *Software Engineering Project Management*, IEEE Computer Society Press, 1997, Chap. 1-10.

APPENDIX A. LIST OF FURTHER READINGS

(Adl99) T.R. Adler, J.G. Leonard, and R.K. Nordgren, "Improving Risk Management: Moving from Risk Elimination to Risk Avoidance," *Information and Software Technology*, vol. 41, 1999, pp. 29-34.

(Bai98) R. Baines, "Across Disciplines: Risk, Design, Method, Process, and Tools," *IEEE Software*, July/August 1998, pp. 61-64.

(Bin97) R.V. Binder, "Can a Manufacturing Quality Model Work for Software?" *IEEE Software*, September/October 1997, pp. 101-102,105.

(Boe97) B.W. Boehm and T. DeMarco, "Software Risk Management," *IEEE Software*, May/June 1997, pp. 17-19.

(Bri96) L.C. Briand, S. Morasca, and V.R. Basili, "Property-Based Software Engineering Measurement," *IEEE Transactions on Software Engineering*, vol. 22, iss. 1, 1996, pp. 68-86.

(Bri96a) L. Briand, K.E. Emam, and S. Morasca, "On the Application of Measurement Theory in Software Engineering," *Empirical Software Engineering*, vol. 1, 1996, pp. 61-88.

(Bri97) L.C. Briand, S. Morasca, and V.R. Basili, "Response to: Comments on 'Property-based Software Engineering Measurement: Refining the Addivity Properties,'" *IEEE Transactions on Software Engineering*, vol. 23, iss. 3, 1997, pp. 196-197.

(Bro87) F.P.J. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer*, Apr. 1987, pp. 10-19.

(Cap96) J. Capers, *Applied Software Measurement: Assuring Productivity and Quality*, second ed., McGraw-Hill, 1996.

(Car97) M.J. Carr, "Risk Management May Not Be For Everyone," *IEEE Software*, May/June 1997, pp. 21-24.

(Cha96) R.N. Charette, "Large-Scale Project Management Is Risk Management," *IEEE Software*, July 1996, pp. 110-117.

(Cha97) R.N. Charette, K.M. Adams, and M.B. White, "Managing Risk in Software Maintenance," *IEEE Software*, May/June 1997, pp. 43-50.

(Col96) B. Collier, T. DeMarco,and P. Fearey, "A Defined Process for Project Postmortem Review," *IEEE Software*, July 1996, pp. 65-72.

(Con97) E.H. Conrow and P.S. Shishido, "Implementing Risk Management on Software Intensive Projects," *IEEE Software*, May/June 1997, pp. 83-89.

(Dav98) A.M. Davis, "Predictions and Farewells," *IEEE Software*, July/August 1998, pp. 6-9.

(Dem87) T. DeMarco and T. Lister, *Peopleware: Productive Projects and Teams*, Dorset House Publishing, 1987.

(Dem96) T. DeMarco and A. Miller, "Managing Large Software Projects," *IEEE Software*, July 1996, pp. 24-27.

(Fav98) J. Favaro and S.L. Pfleeger, "Making Software Development Investment Decisions," *ACM SIGSoft Software Engineering Notes*, vol. 23, iss. 5, 1998, pp. 69-74.

(Fay96) M.E. Fayad and M. Cline, "Managing Object-Oriented Software Development," *Computer*, September 1996, pp. 26-31.

(Fen98) N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous & Practical Approach*, second ed., International Thomson Computer Press, 1998.

(Fle99) R. Fleming, "A Fresh Perspective on Old Problems," *IEEE Software*, January/February 1999, pp. 106-113.

(Fug98) A. Fuggetta et al., "Applying GQM in an Industrial Software Factory," *ACM Transactions on Software Engineering and Methodology*, vol. 7, iss. 4, 1998, pp. 411-448.

(Gar97) P.R. Garvey, D.J. Phair, and J.A. Wilson, "An Information Architecture for Risk Assessment and Management," *IEEE Software*, May/June 1997, pp. 25-34.

(Gem97) A. Gemmer, "Risk Management: Moving beyond Process," *Computer*, May 1997, pp. 33-43.

(Gla97) R.L. Glass, "The Ups and Downs of Programmer Stress," *Communications of the ACM*, vol. 40, iss. 4, 1997, pp. 17-19.

(Gla98) R.L. Glass, "Short-Term and Long-Term Remedies for Runaway Projects," *Communications of the ACM*, vol. 41, iss. 7, 1998, pp. 13-15.

(Gla98a) R.L. Glass, "How Not to Prepare for a Consulting Assignment, and Other Ugly Consultancy Truths," *Communications of the ACM*, vol. 41, iss. 12, 1998, pp. 11-13.

(Gla99) R.L. Glass, "The Realities of Software Technology Payoffs," *Communications of the ACM*, vol. 42, iss. 2, 1999, pp. 74-79.

(Gra99) R. Grable et al., "Metrics for Small Projects: Experiences at the SED," *IEEE Software*, March/April 1999, pp. 21-29.

(Gra87) R.B. Grady and D.L. Caswell, *Software Metrics: Establishing A Company-Wide Program*. Prentice Hall, 1987.

(Hal97) T. Hall and N. Fenton, "Implementing Effective Software Metrics Programs," *IEEE Software*, March/April 1997, pp. 55-64.

(Hen99) S.M. Henry and K.T. Stevens, "Using Belbin's Leadership Role to Improve Team Effectiveness: An Empirical Investigation," *Journal of Systems and Software*, vol. 44, 1999, pp. 241-250.

(Hoh99) L. Hohmann, "Coaching the Rookie Manager," *IEEE Software*, January/February 1999, pp. 16-19.

(Hsi96) P. Hsia, "Making Software Development Visible," *IEEE Software*, March 1996, pp. 23-26.

(Hum97) W.S. Humphrey, *Managing Technical People: Innovation, Teamwork, and the Software Process*: Addison-Wesley, 1997.

(IEEE12207.0-96) IEEE/EIA 12207.0-1996//ISO/IEC12207:1995, *Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology-Software Life Cycle Processes*, IEEE, 1996.

(Jac98) M. Jackman, "Homeopathic Remedies for Team Toxicity," *IEEE Software*, July/August 1998, pp. 43-45.

(Kan97) K. Kansala, "Integrating Risk Assessment with Cost

Estimation," *IEEE Software*, May/June 1997, pp. 61-67.
(Kar97) J. Karlsson and K. Ryan, "A Cost-Value Aproach for Prioritizing Requirements," *IEEE Software*, September/October 1997, pp. 87-74.
(Kar96) D.W. Karolak, *Software Engineering Risk Management*, IEEE Computer Society Press, 1996.
(Kau99) K. Kautz, "Making Sense of Measurement for Small Organizations," *IEEE Software*, March/April 1999, pp. 14-20.
(Kei98) M. Keil et al., "A Framework for Identifying Software Project Risks," *Communications of the ACM*, vol. 41, iss. 11, 1998, pp. 76-83.
(Ker99) B. Kernighan and R. Pike, "Finding Performance Improvements," *IEEE Software*, March/April 1999, pp. 61-65.
(Kit97) B. Kitchenham and S. Linkman, "Estimates, Uncertainty, and Risk," *IEEE Software*, May/June 1997, pp. 69-74.
(Lat98) F. v. Latum et al., "Adopting GQM-Based Measurement in an Industrial Environment," *IEEE Software*, January-February 1998, pp. 78-86.
(Leu96) H.K.N. Leung, "A Risk Index for Software Producers," *Software Maintenance: Research and Practice*, vol. 8, 1996, pp. 281-294.
(Lis97) T. Lister, "Risk Management Is Project Management for Adults," *IEEE Software*, May/June 1997, pp. 20-22.
(Mac96) K. Mackey, "Why Bad Things Happen to Good Projects," *IEEE Software*, May 1996, pp. 27-32.
(Mac98) K. Mackey, "Beyond Dilbert: Creating Cultures that Work," *IEEE Software*, January/February 1998, pp. 48-49.
(Mad97) R.J. Madachy, "Heuristic Risk Assessment Using Cost Factors," *IEEE Software*, May/June 1997, pp. 51-59.
(McC96) S.C. McConnell, *Rapid Development: Taming Wild Software Schedules*, Microsoft Press, 1996.
(McC97) S.C. McConnell, *Software Project Survival Guide*, Microsoft Press, 1997.
(McC99) S.C. McConnell, "Software Engineering Principles," *IEEE Software*, March/April 1999, pp. 6-8.
(Moy97) T. Moynihan, "How Experienced Project Managers Assess Risk," *IEEE Software*, May/June 1997, pp. 35-41.
(Ncs98) P. Ncsi, "Managing OO Projects Better," *IEEE Software*, July/August 1998, pp. 50-60.
(Nol99) A.J. Nolan, "Learning From Success," *IEEE Software*, January/February 1999, pp. 97-105.
(Off97) R.J. Offen and R. Jeffery, "Establishing Software Measurement Programs," *IEEE Software*, March/April 1997, pp. 45-53.
(Par96) K.V.C. Parris, "Implementing Accountability," *IEEE Software*, July/August 1996, pp. 83-93.
(Pfl97) S.L. Pfleeger, "Assessing Measurement (Guest Editor's Introduction)," *IEEE Software*, March/April 1997, pp. 25-26.
(Pfl97a) S.L. Pfleeger et al., "Status Report on Software Measurement," *IEEE Software*, March/April 1997, pp. 33-43.
(Put97) L.H. Putman and W. Myers, *Industrial Strength Software — Effective Management Using Measurement*, IEEE Computer Society Press, 1997.
(Rob99) P.N. Robillard, "The Role of Knowledge in Software Development," *Communications of the ACM*, vol. 42, iss. 1, 1999, pp. 87-92.
(Rod97) A.G. Rodrigues and T.M. Williams, "System Dynamics in Software Project Management: Towards the Development of a Formal Integrated Framework," *European Journal of Information Systems*, vol. 6, 1997, pp. 51-66.
(Rop97) J. Ropponen and K. Lyytinen, "Can Software Risk Management Improve System Development: An Exploratory Study," *European Journal of Information Systems*, vol. 6, 1997, pp. 41-50.
(Sch99) C. Schmidt et al., "Disincentives for Communicating Risk: A Risk Paradox," *Information and Software Technology*, vol. 41, 1999, pp. 403-411.
(Sco92) R.L. v. Scoy, "Software Development Risk: Opportunity, Not Problem," Software Engineering Institute, Carnegie Mellon University CMU/SEI-92-TR-30, 1992.
(Sla98) S.A. Slaughter, D.E. Harter, and M.S. Krishnan, "Evaluating the Cost of Software Quality," *Communications of the ACM*, vol. 41, iss. 8, 1998, pp. 67-73.
(Sol98) R. v. Solingen, R. Berghout, and F. v. Latum, "Interrupts: Just a Minute Never Is," *IEEE Software*, September/October 1998, pp. 97-103.
(Whi95) N. Whitten, *Managing Software Development Projects: Formulas for Success*, Wiley, 1995.
(Wil99) B. Wiley, *Essential System Requirements: A Practical Guide to Event-Driven Methods*, Addison-Wesley, 1999.
(Zel98) M.V. Zelkowitz and D.R. Wallace, "Experimental Models for Validating Technology," *Computer*, vol. 31, iss. 5, 1998, pp. 23-31.

APPENDIX B. LIST OF STANDARDS

(IEEE610.12-90) IEEE Std 610.12-1990 (R2002), *IEEE Standard Glossary of Software Engineering Terminology*, IEEE, 1990.

(IEEE12207.0-96) IEEE/EIA 12207.0-1996//ISO/IEC12207:1995, *Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology-Software Life Cycle Processes*, IEEE, 1996.

(ISO15939-02) ISO/IEC 15939:2002, *Software Engineering-Software Measurement Process*, ISO and IEC, 2002.

(PMI00) Project Management Institute Standards Committee, *A Guide to the Project Management Body of Knowledge (PMBOK)*, Project Management Institute, 2000.