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.

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 that complicate effective management—just a few of which are as follows:

- Clients' perception often lacks appreciation for the complexity inherent in software engineering, particularly regarding 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 as a sequence of closed tasks.
- Software engineering necessarily incorporates aspects of creativity and discipline—maintaining an appropriate balance between the two is often difficult, particularly when choosing between plan-based and agile development processes.
- The degree of software novelty and complexity is often extremely high.
- There is often 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. It is generally agreed upon that software engineering managers should be conversant with the project management and software measurement knowledge described in this KA. They should also possess some target domain knowledge. Even though the first level on organizational and infrastructure management is beyond the scope of this KA, such managers must also be cognizant and, if possible, conversant with this category of knowledge.

Since the link to the related discipline—management—is obviously 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 policies specific to software engineering 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—for example, by establishing a consistent basis on which to analyze past performance and implement improvements.

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 are targeted in the SWEBOK Guide.

Both organizational culture and behavior as well as functional enterprise management—in terms of procurement, supply chain management, marketing, sales, and distribution—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," and the fact that such management is normally done 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)

, which 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, project communications management, project risk management and project procurement management. Clearly, all these topics have direct relevance to the Software Engineering Management KA. To attempt to duplicate the content of PMBOK here would be both impossible and inappropriate. Instead, we suggest that the readers interested in project management, beyond what is specific to software engineering projects, consult the PMBOK itself. Project management is also found in the chapter, "Related Disciplines of Software Engineering."

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 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 [1* , c7, c8] and the models that are derived from them, whether these models are developed using statistical 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 SWEBOK, and reading the following KA descriptions in conjunction with this one would be particularly useful:

- Software Requirements, which describes some of the activities to be performed during the Initiation and Scope definition phase of the project.
- 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 very closely related.
- Software Quality**,** as quality is constantly a goal of management and is an aim of many activities that must be managed.
- Software Engineering Economics, which discusses how to make software-related decisions in a business context.

BREAKDOWN OF TOPICS FOR SOFTWARE ENGINEERING MANAGEMENT

As the Software Engineering Management KA is viewed here as an organizational process that incorporates both the notions 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 seven major subareas. The seven subareas are:

- Initiation and scope definition, which deal 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 a software engineering project.
- 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.
- Software engineering management tools, which addresses the selection and use of tools to manage a software engineering project.

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

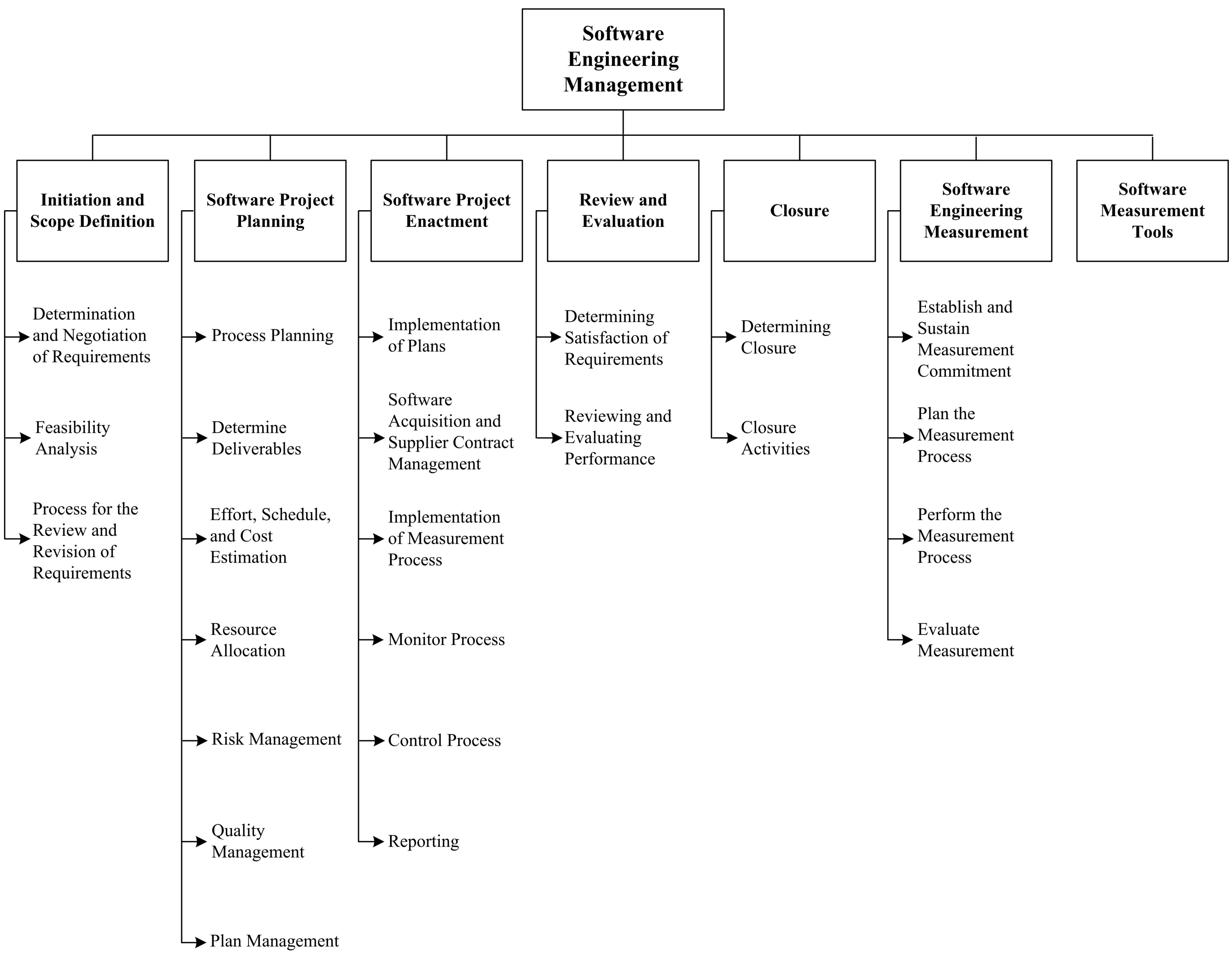

Figure 1: Breakdown of Topics

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

[1* , c3]

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 when the novelty of the undertaking is high. (Additional information can be found in the Software Requirements KA.)

1.2. Feasibility Analysis

[2*, c4]

Software engineers must be assured that adequate technical, operational, financial, and social/political capabilities and resources are available. These include sufficient 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). (See also the Software Requirements KA.) Feasibility analysis 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

[1*, c3]

Given the inevitability of change, it is vital that stakeholders agree on the means by which scope and requirements are to be reviewed and revised (for example, via agreed-upon change management procedures) at this early point. 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 and/or 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. See also the Software Configuration Management KA's Software Configuration Control subarea.

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 upon.

2.1. Process Planning

[1*, c3, c4, c5], [3*, c1]

Selection of the appropriate software life cycle model (for example, waterfall, spiral, or agile) 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. Tools that will be used throughout the project must be planned for and acquired. Tools can include project scheduling tools, software requirements tools, software design tools, software construction tools, software maintenance tools, software configuration management tools, software engineering process tools, software quality tools and others. While many of these tools should be selected based primarily on the technical considerations discussed in other KAs, some of them are more closely related to the management considerations discussed in this chapter.

The project is also decomposed into tasks, with associated inputs, outputs, and completion conditions. This, in turn, influences decisions on the project's high-level schedule and organization structure.

The choice of an appropriate software life cycle model is one of the early decisions that must be made for every project. The choice is usually between the traditional plan-driven (waterfall or spiral) model and the agile model. The plan-driven model views software projects as undertakings that can be successfully accomplished only by using a systematic approach that carefully adheres to specific processes and activities in moving software from requirements to finished code. There is also a concern for completeness of documentation so that thorough verification or validation can notably be accomplished after the fact at appropriate steps in the process. On the other hand, the agile software life cycle model is composed of lightweight processes that employ short iterative cycles, actively involving users to establish and prioritize requirements and relying more on tacit knowledge within a team as opposed to documentation. Decisions on details of processes falling between these two models are difficult and deserve careful analysis of the project's characteristics, availability of resources, and customer needs.

2.2. Determine Deliverables

[1*, c4, c5, c6]

The product(s) of each task (for example, architectural design documents or inspection reports) are specified and characterized. 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

[1*, c6]

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) as well as relevant 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, Project Evaluation and Review Technique (PERT) chart). Resource requirements (people, tools) are translated into cost estimates. This is a highly iterative activity that must be negotiated and revised until consensus is reached among affected stakeholders (primarily engineering and management).

2.4. Resource Allocation

[1*, c5]

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, as well as organizational and team structures).

2.5. Risk Management

[1*, c9] [3*, c5]

Risk identification and analysis (what can go wrong, how and why, and what are the likely consequences), critical risk assessment (for example, which risks are the most significant in terms of exposure and which can we do something about in terms of leverage), as well as 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. 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. In addition, particular attention should notably be paid to the management of security risks.

2.6. Quality Management

[1*, c4] [2*, c24]

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 Software Quality Assurance (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

[1*, c4]

How the project and the project 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; they must also 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

[2*, c2]

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

3.2. Software Acquisition and Supplier Contract Management

[1*, c3, c4]

Software acquisition and supplier contract management is concerned with issues involved in contracting with customers and vendors. It involves selection of appropriate kinds of contracts—such as fixed price, time and materials, cost plus fixed fee or cost plus incentive fee. Contracts typically specify the scope of a project and include clauses such as penalties for breach of contract and intellectual property agreements that specify what the acquirer is paying for and what will be delivered to and owned by the customer. Notably, for software being developed by subcontractors, contracts should clearly indicate quality requirements for acceptance of the software. After the contract has been signed, the implementation of the project in compliance with the terms of the contract must be managed.

3.3. Implementation of Measurement Process

[1*, c7]

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

[1*, c8]

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

[1*, c7, c8]

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, software requirements specification) to accommodate the unexpected outcomes and their implications.

In some instances, it may lead to abandonment of the project. In all cases, software configuration 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 (see also topic 6.3, "*Perform the Measurement Process"*).

3.6. Reporting

[1*, c11]

At specified and agreed-upon periods, adherence to the plan 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 process to date, the personnel involved, and the tools and methods employed are also undertaken at particular milestones.

4.1. Determining Satisfaction of Requirements

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, completion of software design architecture, completion of software integration technical review). Variances from requirements are identified and appropriate action is taken. As in the control process activity above (see topic 3.5, "*Control Process"*), software configuration control and software configuration management procedures are adhered to in all cases (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 (see also topic 6.3, "*Perform the Measurement Process"*). More information can also be found in the Software Testing KA's topic 2.2, "*Objectives of Testing,"* and in the Software Quality KA's topic 2.3, "*Reviews and Audits*."

4.2. Reviewing and Evaluating Performance

[1*, c8, 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 being used by the project is also 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

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

After closure has been confirmed, archival of project materials takes place in line with stakeholder-agreed methods, location, and duration—possibly including destruction of sensitive information and software and the medium on which copies are resident. 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 (particularly via review and evaluation, see subarea 4, "*Review and Evaluation"*) are analyzed, and lessons are drawn from the project and fed into organizational learning and improvement endeavors.

6. Software Engineering Measurement

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.

This subarea follows the IEEE 15939:2008 standard [4], which describes a process that 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

[5*, c1,c2] [1]

- 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." 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) as well as providing adequate funding, training, tools, and support to conduct the process in an enduring fashion.

[1] Please note that these two chapters can be downloaded free of charge from http://www.psmsc.com/PSMBook.asp

6.2. Plan the Measurement Process
[5*, c1,c2]

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, organizational interfaces and organizational structure.

- 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.
- 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. Because internal quality characteristics are often not contained in the contractually binding software requirements, it is important to consider measuring the internal quality of the software to provide an early indicator of potential issues.
- Define data collection, analysis, and reporting procedures. This encompasses collection procedures and schedules, storage, verification, analysis, reporting, and configuration management of data.
- 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.
- 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.
    - 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.
- 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.

6.3. Perform the Measurement Process
[5*, c1,c2]

- 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.
- Collect data. The data must be collected, verified, and stored. 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.
- Communicate results. Information products must be documented and communicated to users and stakeholders.

6.4. Evaluate Measurement

[5*, c1,c2]

- 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.
- 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.
- 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.

7. Software Engineering Management Tools

[1*, c5, c6, c7]

Software engineering management tools can be divided into three categories: project planning and tracking, risk management, and measurement. Project planning and tracking tools are used in project-effort management and cost estimation as well as in project scheduling. Risk management tools are used in identifying, estimating, and monitoring risks. Measurement tools assist in performing the activities related to the software measurement program.

MATRIX OF TOPICS VS. REFERENCE MATERIAL

	Fairley 2009 [1*]	**Sommerville 2010 [2*]**	**Boehm and Turner 2003 [3*]**	**McGarry *et al.* 2001 [5*]**
1. Initiation and scope definition				
1.1 Determination and negotiation of requirements	c3			
1.2 Feasibility analysis		c4		
1.3 Process for the review and revision of requirements	c3			
2. Software Project Planning				
2.1 Process planning	c3, c4, c5		c1	
2.2 Determine deliverables	c4, c5, c6			
23 Effort, schedule and cost estimation	c6			
2.4 Resource allocation	c5			
2.5 Risk management	c9,		c5	
2.6 Quality management	c4	c24		
2.7 Plan management	c4			
3. Software Project Enactment				
3.1 Implementation of plans		c2		
3.2 Software Acquisition and Supplier contract management	c3, c4			
3.3 Implementation of measurement process	c7			
3.4 Monitor process	c8			
3.5 Control process	c7, c8			
3.6 Reporting	c11			
4. Review and evaluation				
4.1 Determining satisfaction of requirements				
4.2 Reviewing and evaluating performance	c8, c10			
5. Closure				
5.1 Determining closure				
5.2 Closure activities				
6. Software Engineering Measurement				
6.1 Establish and sustain measurement commitment				*c1, c2*
6.2 Plan the measurement process				*c1, c2*
6.3 Perform the measurement process				*c1, c2*
6.4 Evaluate measurement				*c1, c2*
7. Software Engineering Management Tools	c5, c6, c7			

Software Engineering Management – Further Readings (to be annotated later) : [4, 6, 7, 8, 9]

[1*] R. E. Fairley, *Managing and Leading Software Projects*. Hoboken, NJ: Wiley-IEEE Computer Society Press, 2009.

[2*] I. Sommerville, *Software Engineering*, 9th ed. New York: Addison-Wesley, 2010.

[3*] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Addison-Wesley, 2003.

[4] IEEE, "IEEE Standard Adoption of ISO/IEC 15939:2007 Systems and Software Engineering Measurement Process," ed: IEEE, 2008.

[5*] J. McGarry*, et al.*, *Practical Software Measurement : Objective Information for Decision Makers*: Addison-Wesley Professional, 2001.

[6] M. Christensen*, et al.*, *Software Engineering, Vol. 1 and Vol. 2*: IEEE Computer Society Press, 2002.

[7] J. McDonald, *Managing the Development of Software Intensive Systems*. Hoboken, NJ: John Wiley and Sons, Inc., 2010.

[8] S. L. Pfleeger, *Software Engineering: Theory and Practice, 2nd*: Prentice Hall, 2001.

[9] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide)*, 4th ed. Newton Square, PA: Project Management Institute, 2008.