CHAPTER 8

SOFTWARE ENGINEERING MANAGEMENT

Stephen G. MacDonell and Andrew R. Gray
University of Otago,
Dunedin, New Zealand
+64 3 479 8135 (phone) +64 3 479 8311 (fax)
stevemac@infoscience.otago.ac.nz

Table of Contents

1 INTRODUCTION

This is the current draft (version 0.9) of the knowledge area description for *Software Engineering Management*. The primary goals of this document are to:

1) define the *Software Engineering Management* knowledge area,

2) present a breakdown of the knowledge area in an hierarchical topic framework,

3) provide a list of references that addresses the topics in the breakdown,

4) provide a topic-reference matrix,

5) provide a list of further readings and supplementary references that also address topics in this knowledge area.

2 DEFINITION OF THE SOFTWARE ENGINEERING MANAGEMENT KNOWLEDGE AREA

Before defining the *Software Engineering Management* knowledge area it is first necessary to set out the scope or context in which it is placed. As an organizational process, it is also important that its relationship to other related standards and to other knowledge areas is clear.

2.1 Scope and definition

The scope of this knowledge area follows the general focus of the Guide; that is, "emphasis… is placed upon the construction of useful software artifacts" (page i, Preface to the Guide to the SWEBOK). As a result we are principally concerned with issues related to software *development* – the *acquisition* of software solutions receives less attention here.

Software engineering is characterized in this Guide according to the IEEE definition: "(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software." *Management* generally incorporates the following activities: planning, coordinating, measuring, monitoring, controlling and reporting. Combining these two definitions leads us to an understanding of *Software Engineering Management*: the application of management activities – planning, coordinating, measuring, monitoring, controlling and reporting – to ensure that the development of software is systematic, disciplined and measured.

The *Software Engineering Management* knowledge area therefore addresses the management of software development and the measurement and modeling of software development. Whilst measurement is an important aspect of all Guide to the SWEBOK knowledge areas, it is here that the topic is most focused, particularly with regard to issues involved in goal-driven measurement selection, model development and testing for the purposes of software engineering management.

2.2 The management of software engineering

Whilst 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 particular

to software products and the software engineering process that complicate effective management – just a few of them are as follows:

- the perception of clients is such that there is a lack of appreciation for the complexity inherent in software engineering, particularly in relation to the impact of changing requirements
- related to the point just made, it is almost inevitable that the software engineering process itself will generate the need for new or changed client requirements
- as a result, software is often built in an iterative process rather than a concrete sequence of closed tasks
- software engineering necessarily incorporates aspects of creativity and discipline – maintaining an appropriate balance between the two is often difficult
- unlike many other disciplines, we are largely lacking an underlying theory (e.g. engineering is founded on the principles of physics and mathematics)
- software engineers create intangible products that cannot easily be tested in the same sense that a physical product can
- the degree of novelty and complexity of the software we are asked to build is extremely high, in that most (if not all) of the common and simple products have already been built
- we are faced with an extremely rapid rate of change in the underlying technology.

2.3 Relationship to general management and project management

With respect to software engineering, management activities occur at two levels. Aspects of general organizational management are relevant in terms of their impact on software engineering. For instance, planning at the strategic, tactical and operational level, 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. Perhaps more pertinent to this knowledge area 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) [PMI, 1996], which itself includes the following project management knowledge areas: integration, scope, time, cost, quality, human resource, communications, risk, and procurement. Clearly all of these topics have direct relevance to this knowledge area. Rather than attempt to duplicate the content of the Guide to the PMBOK here, which would be both impossible and inappropriate, we instead provide a cross-reference table at the end of this document so that the relationship between the two is evident.

2.4 Relationship to other Guide to the SWEBOK knowledge areas and standards

Not unexpectedly this knowledge area is closely related to others in the Guide to the SWEBOK, and reading the following knowledge area documents in conjunction with this one would be particularly useful. Material that is covered in these separate documents is not duplicated here.

*Software Configuration Management***,** as this deals with the administration, monitoring and control of collections of [software] components.

*Software Engineering Process***,** where these process activities must be managed.

*Software Quality***,** as quality is constantly a goal of management and is an aim of many activities that must be managed.

In order to provide a broader context in which these knowledge areas can be considered it is useful to map them to the IEEE/EIA Standard for Information Technology (ISO/IEC 12207) – *Software life cycle processes*. This sees the four management-oriented knowledge areas principally aligned to '6. Supporting Life Cycle Processes' and to '7. Organizational Life Cycle Processes' as follows:

Guide to the SWEBOK	*ISO/IEC 12207*
Chapter 7 Software Configuration Management	6.2 Configuration Management
Chapter 8 Software Engineering Management	7.1 Management
Chapter 9 Software Engineering Process	7.3 Improvement
Chapter 11 Software Quality	6.3 Quality Assurance 6.6 Joint Review 6.4 Verification 6.7 Audit 6.5 Validation

Chapters 2 through 6 of the Guide to the SWEBOK represent the phases of the software development process (and map to sections 5.3 Development and 5.5 Maintenance of ISO/IEC 12207). Clearly each process must be managed – issues of particular relevance to each process are dealt with in the associated knowledge area. Our focus is on the relevant aspects of enterprise, process and project management as they apply to software engineering rather than to individual development processes.

2.5 Management and measurement

As alluded to above, the *Software Engineering Management* knowledge area consists of both the management process and measurement sub-areas. Whilst these two topics are often regarded as being separate, and

indeed they do possess many mutually unique aspects, their close relationship has led to their combined treatment here as part of the Guide to the SWEBOK. Unfortunately the public perception of the software industry is that it delivers products late, over budget, with 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*. Effective management requires a combination of both numbers and stories.

The following working definitions are adopted here:

> *Management process* refers to the activities that are undertaken in order to ensure that the software development process is performed in a manner consistent with the organization's policies, goals, and standards.
>
> *Measurement* (a.k.a. Metrics) refers to the assignment of values and labels to aspects of software development (products, processes, and resources as defined by [Fenton and Pfleeger, 1997]) and the models that are derived from them whether these models are developed using statistical, expert knowledge, or other techniques.

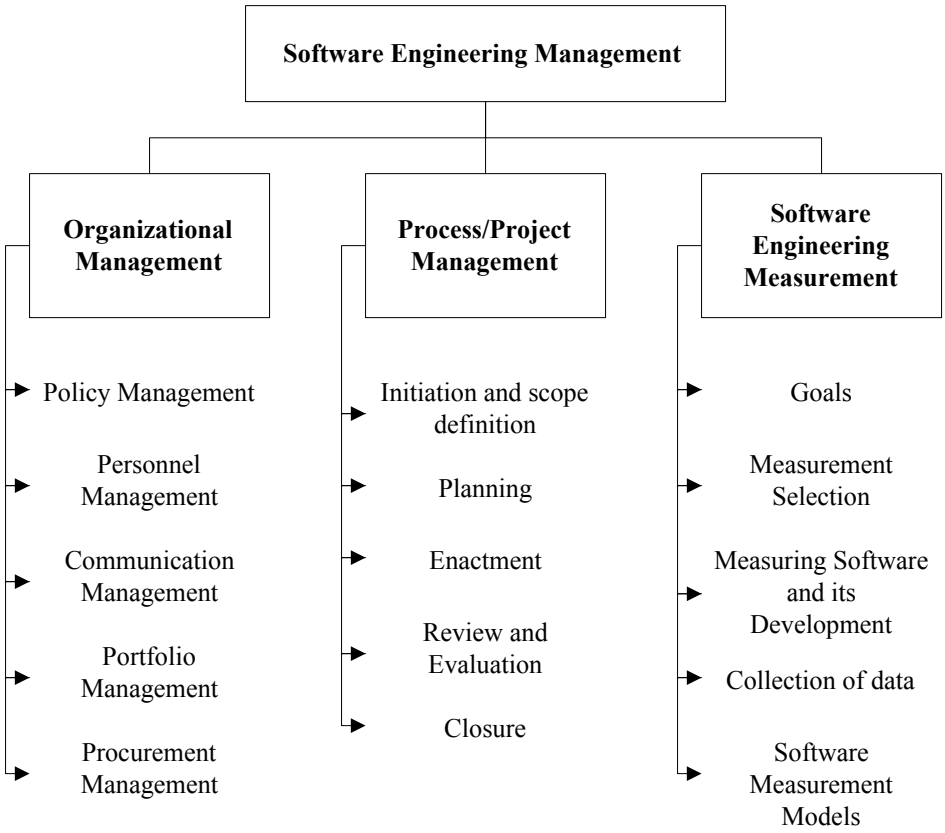
The management process sub-area makes extensive use of the measurement sub-area. This exchange between the two sub-areas occurs continuously throughout the software development processes.

3 BREAKDOWN OF TOPICS FOR SOFTWARE ENGINEERING MANAGEMENT

As the *Software Engineering Management* knowledge area is viewed here as an organizational process that incorporates the notion of process and project management, we have created a breakdown that is both topic-based and life cycle-based. There are three major topic areas: organizational management, which deals with high-level management activities that have a relevant but somewhat indirect impact on software engineering; process/project management, which deals with generally accepted software engineering management activities; and software engineering measurement, which deals with the effective development and implementation of measurement programs in software engineering organizations. Within each main topic area relevant sub-topics are listed, and described where necessary. In particular, further explanation is provided in the process/project management and software engineering measurement topic areas where distinct issues relating to software engineering management warrant more detailed attention.

A. Organizational management

1. Policy management – organizational policies and standards provide the framework in which software engineering is undertaken. As such, they operationalize overall organizational strategies and have an indirect influence on the software engineering process and its management. It is important that those charged with the management of software engineering both understand and influence the development, dissemination, deployment and enforcement of policies and standards. [Pfle: c2; Reif: c2; Somm: c30; Thay: c2,c4]
   1. Means of policy development
   2. Policy dissemination and enforcement
   3. Development and deployment of standards
2. Personnel management – policies and procedures used at the organizational level to recruit, select, motivate and reward personnel also affect the management of software engineering teams and individuals. It is acknowledged that in order to recruit and retain high-quality personnel in the software engineering industry it is vital that training, motivation, career development and the like are given adequate attention. [F&P: c11; Pfle: c3; Press: c3; Reif: c7,c8; Somm: c28; Thay: c7,c8]
   1. Hiring and retention
   2. Training and motivation
   3. Mentoring for career development
3. Communication management – even if project-based communication is effective, an organization is unlikely to survive long-term without clear policies and procedures that are applicable in the wider context. An awareness of communication channels (formal and informal), conventions in terms of terminology, form and style, mechanisms for feedback and the impact of organizational structures on communication, has an indirect but important influence on communication within the software engineering process. [Press: c3; Somm: c28; Thay: c1,c3]
   1. Communication channels and media
   2. Meeting procedures



3. Written presentations

4. Oral presentations

5. Negotiation

4. Portfolio management – organizations that deal with multiple clients and/or multiple projects are often faced with the need to prioritize their effort in terms of the projects they undertake. It is important that those involved in software engineering management both contribute to and are guided by the organizational management of project portfolios, where portfolios are constructed in light of the advantages and disadvantages of undertaking individual projects using a variety of cost/benefit and similar analysis methods. [Press: c10]

1. Strategy development and coordination

2. General investment management techniques

3. Project selection

4. Portfolio construction (risk minimization and value maximization)

5. Procurement management – in cases where an organization outsources (part of) their operation to an external agency this process must be managed effectively in order to ensure a successful outcome. As it is not uncommon for organizations to purchase some or all of their software engineering activity in such a way, organizational policies and procedures should exist to facilitate effective provider-consumer relationships. [Press: c5; Reif: c15; Somm: c2]

1. Procurement planning and selection

2. Supplier contract management

B. Process/project management (largely following 7.1 ISO/IEC 12207 Management Process)

1. Initiation and scope definition – the focus of this set of activities is on the effective determination of process and/or project requirements via various elicitation methods and the assessment of the process/project's feasibility from a variety of standpoints. Once feasibility has been established, the remaining task within this process is the specification of requirements review and modification procedures (see also Chapter 2 of the Guide to the SWEBOK).

1. Determination and negotiation of requirements – methods of requirements engineering, elicitation (e.g. observation), analysis (e.g. data modelling, use case modelling), specification, and validation (e.g. prototyping) must be selected and applied in cognizance of various stakeholder perspectives. This leads to the determination of process/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. [D&T: c4; Pfle: c4; Press: c5,c11,c12; Somm: c4-11]

2. Feasibility analysis (technical, operational, financial, social/political) – the software engineering manager 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 process/project can be successfully completed in a timely and cost-effective manner (using, for example, a requirement-capability matrix). This often requires some 'ball-park' estimation of effort and cost based on appropriate methods (e.g. expert-informed analogy techniques). [Press: c10]

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 (e.g. 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 pre-determined points as the process occurs (e.g. at design reviews, acceptance tests). If changes are accepted then some form of traceability analysis and risk analysis (see below) 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 process/project, as the scope and requirements should form the basis for evaluation of success. [Somm: c4]

2. Planning – the iterative planning process is informed by the scope and requirements and the establishment of feasibility. At this point, software processes are evaluated and the most appropriate (given the nature of the process/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 an 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 process/project is discussed among and accepted by all relevant stakeholders. Comprehensive quality management processes are determined as part of the planning process in the form of procedures and responsibilities for quality assurance, verification and validation (see also Chapter 11 of the Guide to the SWEBOK). 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.

1. Process planning – selection of the appropriate software process (e.g. spiral, cleanroom) and the specification and deployment of appropriate process standards are undertaken in the light of the particular scope and requirements of the process/project. Relevant methods and tools are also selected. [D&T: c5,c11; Pfle: c2; Press: c2; Reif: c1,c2,c4; Somm: c1; Thay: c3]

2. Project planning – appropriate methods and tools are used to decompose the project into tasks, with associated inputs, outputs and completion conditions (e.g. work breakdown structure). [D&T: c10; Pfle: c3; Press: c3,c5; Reif: c3,c4; Somm: c3; Thay: c4,c6]

3. Determine deliverables – the product(s) of each task (e.g. high level architectural design, inspection report) are specified and characterized. [Pfle: c3; Press: c3,c7; Somm: c3; Thay: c4]

4. Effort, schedule and cost estimation – based on the breakdown of tasks, inputs and outputs, the expected effort range required for each is determined using a calibrated estimation model based on historical size-effort data where available and relevant (e.g. analogy-based estimation, function point analysis); task dependencies are established and potential bottlenecks are identified using suitable methods (e.g. critical path analysis); bottlenecks are resolved where possible and the expected schedule of tasks with projected start times, durations and end times is produced (e.g. PERT chart); resource requirements (people, tools) are translated into cost estimates. [D&T: c10; F&P: c12; Pfle: c3; Press: c5,c7; Reif: c4,c5; Somm: c3,c29; Thay: c5]

5. Resource allocation – 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 e.g. productivity of individuals/teams, team dynamics, organizational and team structures. [Pfle: c3; Press: c5; Reif: c7,c8; Somm: c3; Thay: c6,c7]

6. 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 (e.g. 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. [D&T: c10; Pfle: c3; Press: c6; Reif: c11; Thay: c4]

7. Quality management – quality is defined in terms of pertinent attributes of the specific process/project and any associated product(s), perhaps in both quantitative and qualitative terms. (These quality attributes will have been determined in the specification of detailed requirements.) Thresholds for adherence to quality are set for each attribute 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 (e.g. reviews and inspections) (see also Chapter 11 of the Guide to the SWEBOK). [D&T: c7,c9; Press: c8; Reif: c10; Somm: c30,c31; Thay: c9,c10]

8. Plan management – 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 is systematically directed, monitored, reviewed, reported, and, where appropriate, revised. Plans associated with other management-oriented support processes (e.g. documentation, configuration management and problem resolution) also need to be managed in the same manner. [Somm: c3; Thay: c4]

3. 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 over-riding expectation that such adherence will lead to the successful satisfaction of stakeholder requirements and achievement of the process/project objectives. Fundamental to enactment are the ongoing management activities of measuring, monitoring, controlling and reporting.

    1. Implementation of plans – the process is initiated and the process/project activities are undertaken according to the schedule. In the process, resources are utilized (e.g. personnel effort, funding) and deliverables are produced (e.g. architectural design documents, test cases). [Pfle: c3; Somm: c3]

    2. Implementation of measurement process – the measurement process is enacted alongside the software development process/project, ensuring that relevant and useful data is collected (see also section C of this knowledge area breakdown). [F&P: c13,c14; Press: c4; Reif: c9,c10,c12; Thay: c3,c10]

    3. Monitor process – adherence to the various plans is systematically assessed continually and at pre-determined intervals. Outputs and completion conditions for each task are analyzed, deliverables are evaluated in terms of their required characteristics (e.g. via joint reviews, test audits), effort expenditure, schedule adherence and costs to date are investigated, resource usage is examined, the process/project risk profile is revisited, and adherence to quality requirements is evaluated. Measurement data is 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 is performed (e.g. defect density analysis). Risk exposure and leverage are recalculated and decisions trees, simulations and so on are re-run 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. [D&T: c7,c9,c10; Press: c7; Reif: c9,c10; Somm: c31; Thay: c3;c9]

    4. Control process – 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 process/project. This may take the form of corrective action (e.g. re-testing certain components), it may involve the incorporation of contingencies so that similar occurrences are avoided (e.g. the decision to use prototyping to assist in requirements validation), and/or it may entail the revision of the various plans and other project documents (e.g. requirements specification) to accommodate the unexpected outcomes and their flow-on implications. In some instances it may lead to abandonment of the process/project. In all cases, change control and configuration management procedures are adhered to (see Chapter 7 of the Guide to the SWEBOK), 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 section C of this knowledge area breakdown). [D&T: c10; Press: c9; Reif: c9,c10; Thay: c3,c9]

5. Reporting – at specified and agreed periods, adherence to the plans is reported, both within the organization (e.g. to the project portfolio steering committee) and to external stakeholders (e.g. clients, users). Reports of this nature should focus on overall adherence as opposed to the detailed reporting required frequently within the process/project team. [Reif: c9,c10; Thay: c3,c10]

4. Review and evaluation – at critical points in the process/project overall progress towards achievement of the stated objectives and satisfaction of stakeholder requirements is 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.

   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 is formally and periodically assessed. This occurs at the achievement of major process/project milestones (e.g. confirmation of software design architecture, software integration joint review). Variances from expectations are identified and appropriate action is taken. As in the Control process activity above, in all cases change control and configuration management procedures are adhered to (see Chapter 7 of the Guide to the SWEBOK), 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 section C of this knowledge area breakdown). [Reif: c9,c10; Thay: c3,c10]

   2. Reviewing and evaluating performance – periodic performance reviews for process/project personnel provide insights as to the likelihood of adherence to plans as well as possible areas of difficulty (e.g. 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 process/project context (see also the other SWEBOK chapters). Where appropriate, changes are made and managed. [D&T: c7; Pfle: c7,c8; Press: c8; Reif: c9,c10; Thay: c3,c10]

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

   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 process/project have been achieved. These processes generally involve all stakeholders and result in the documentation of client acceptance and any remaining known problem reports. [D&T: c7; Reif: c9,c10; Thay: c3,c10]

   2. Closure activities – after closure has been confirmed, archival of process/project materials takes place in line with stakeholder-agreed methods, location and duration. The organization's measurement database is updated with final process/project data and post-project analyses are undertaken. A process/project *post mortem* is undertaken so that issues, problems and opportunities encountered during the process (particularly via Review and Evaluation) are analyzed, lessons are drawn from the process, and are fed into organizational learning and improvement endeavors (see also Chapter 9 of the Guide to the SWEBOK). [Pfle: c11; Somm: c31]

      (Software then moves into operation, maintenance and, perhaps eventually, retirement. Whilst these tasks also need to be managed they are not explicitly addressed here – software maintenance as a set of activities is addressed in Chapter 6 of the Guide to the SWEBOK, and the other topics (software operation and retirement) are outside the scope of the Guide.)

C. Software engineering measurement

1. Determining the goals of a measurement program – the *ad hoc* approach to software engineering measurement that characterized early efforts – that is, measuring everything possible – often failed to provide genuine insights in terms of organizational improvement, or worse, it led to spurious outcomes that did not generalize to other cases. Each measurement endeavor should be guided by organizational objectives and driven by an over-riding goal that has organizational improvement at its foundation. In this way, measurement effort expenditure should ultimately result in some sort of cost-effective gain to the organization, based on justified prioritization of efforts. An emerging international standard, ISO/IEC FCD 15939, describes a generic process that defines the activities and tasks necessary to implement a software measurement process and includes as well a measurement information model. [ISO/IEC, 2000]

1. Organizational objectives – organizational strategies inform software engineering management in terms of identifying the broad issues and objectives that hold principal relevance at the organizational level (e.g. being first-to-market with new products). [F&P: c3,c13; Press: c4]
2. Software process improvement goals – organizational objectives are translated into specific software-related goals that, if achieved, can assist the organization in attaining its objectives (e.g. optimizing software development with a view to shortening the product life-cycle whilst maintaining process and product quality). [F&P: c3,c13; Pfle: c12; Press: c4; Reif: c2; Somm: c31]

2. Measurement selection – development of an effective measurement process is informed by the organizational objectives and software process improvement goals as specified. This provides the necessary context for more specific and detailed measurement selection. Some understanding of the validity, accuracy and reliability of the selected measures is also crucial in terms of assessing the value of the measurement program and the confidence that can be placed in the results generated from it.
    1. Goal-driven measurement selection – once software process improvement goals are set, we are then in a position to utilize a decomposition process in order to ask questions of direct relevance and interest, leading finally to the selection of useful and relevant measures (e.g. the Goal/Question/Metric approach incorporates just such a decomposition process). In relation to shortening the product life-cycle we may adopt a measurement goal of maximizing software development productivity. In turn, we might ask questions such as: how much effort is expended on rework? what is the range of developer productivity rates? is developer productivity in line with changes in developer experience? All require quite different measures in order to provide the answers needed to achieve the over-riding goals. [F&P: c1,c3,c13,c14; Reif: c12; Thay: c10]
    2. Measurement validity – an awareness of issues relating to measurement validity and reliability is essential if the measurement program is to provide effective and bounded results. In particular, an appreciation of measurement scales and the implications of each scale type in relation to the subsequent selection of data analysis methods is especially important. [F&P: c2; Pfle: c11]

3. Measuring software and its development – whilst the application of measurement to software engineering can be complex, particularly in terms of modeling and analysis methods (see below), there are several aspects of software engineering measurement that are fundamental and that underlie much of the more advanced measurement and analysis processes. Furthermore, achievement of process and product improvement efforts can only be assessed if a set of baseline measures has been established. Software engineering management therefore includes, as a minimum, the measurement of product size, product structure, resource utilization and product and process quality.
    1. Size measurement – software product size is most often assessed by measures of length (e.g. lines of source code in a module, pages in a requirements specification document) or functionality (e.g. function points in a specification or design, COCOMO evaluation of a system design). The standard for functional size measurement methods is [ISO/IEC 1998] and additional supporting standards are under development. A number of specific methods, suitable for different purposes, are available. [F&P: c7; Press: c4,c18,c23; Reif: c12; Somm: c30].
    2. Structure measurement – a diverse range of measures of software product structure may be applied to both high- and low-level design and code artifacts to reflect control-flow (e.g. the cyclomatic number, code knots), data-flow (e.g. measures of slicing), nesting (e.g. nesting polynomial measure, the BAND measure), control structures (e.g. the vector measure, the NPATH measure), and modular structure and interaction (e.g. information flow, tree-based measures, coupling and cohesion). [F&P: c8; Press: c18,c23]
    3. Resource measurement – whilst some effort can be made to assess the utilization of tools and hardware, the primary resource that needs to be managed in software engineering is personnel. As a result the main measures of interest are those related to productivity of individuals and of teams (e.g. using a measure of function points produced per unit of person-effort) and their associated levels of experience in software engineering in general and perhaps in particular technologies. [F&P: c3,c11; Somm: c29]
    4. Quality measurement – as a multi-dimensional attribute, quality measurement is less straightforward to define than those above. Furthermore, some of the dimensions of quality (e.g. usability, maintainability, and value to the client) are likely to require measurement in qualitative rather than quantitative form. A more detailed discussion of software quality

assessment is provided in Chapter 11 of the Guide to the SWEBOK. [F&P: c9,c10; Press: c4; Reif: c12; Somm: c30]

4. Collection of data – when developing a measurement process it is important to ensure that the *optimal* set of measures is chosen. By optimal it is not just meant that the measures are those that necessarily provide the greatest (predictive) power for the desired purpose. It is also important that the cost of data collection is minimized or at least balanced against the benefits to be gained from the outputs of the program. The possibility of reusing measures collected for other purposes is also considered as part of the collection process. The data collected is also useful from the perspective of enabling appropriate models to be developed for analysis, classification and prediction.
    1. Survey techniques and form design – data collection forms and questionnaires are pilot tested before they are used on actual processes/projects. Forms are logically laid out, require minimum completion, and make use of default values where possible. Assistance for form and survey completion is made available. [F&P: c4,c5]
    2. Automated and manual data collection – all data collection has associated costs, both direct (in terms of people employed and software purchased) and indirect (in the costs of interruptions and delays as measurement data are analyzed). For this reason, the measurement process is treated as an *investment* in the development process, with justification for expenditure and quantification of the resulting benefits. Procedures relating to data collection detail the point at which the data is available, the way in which it is collected, the personnel responsible for collection, and the cost associated with collection. Where possible, unobtrusive automated data collection is preferred. This information is important in ensuring that that program is actually feasible. The potential exists for a measurement process to be created, only to find that some of the data cannot physically be collected, or not in sufficient quantities. [F&P: c5; Press: c4; Somm: c30]
5. Software measurement models – as the data is collected and the measurement database is populated we become able to build models using both data and knowledge. These models exist for the purposes of analysis, classification and prediction. Such models need to be evaluated to ensure that their levels of accuracy are sufficient and that their limitations are known and understood. The refinement of models, which takes place both during and after projects are completed, is another important activity. The implementation of measurement models is more management-oriented since the use of such models has an influential effect on personnel behavior.
    1. Model building, calibration and evaluation – the goal-driven approach to measurement informs the model building process to the extent that models are constructed to answer relevant questions and achieve software improvement goals. This process is also influenced by the implied limitations of particular measurement scales in relation to the choice of analysis method. The models are calibrated (by using particularly relevant observations e.g. recent projects, projects using similar technology) and their effectiveness is evaluated (e.g. by testing their performance on holdout samples). [F&P: c4,c6,c13; Pfle: c3,c11,c12; Somm: c29]
    2. Implementation, interpretation and refinement of models – the calibrated models are applied to the process/project (see Process/project enactment), their outcomes are interpreted and evaluated in the context of the process/project, and the models are then refined where appropriate. [F&P: c6; Pfle: c3,c11,c12; Press: c4; Somm: c29]

4 BREAKDOWN RATIONALE

The following subsections each describe how the proposed draft of the knowledge area description meets the criteria given in the project guidelines.

One or two breakdowns with identical topics

A single breakdown of topics is shown.

Soundness and reasonableness

The primary references and secondary sources were examined quite thoroughly in order to list all main topics. The division of the management process into life-cycle based topics seems both plausible and useful in terms of educational presentation.

Generally acceptable

In our view the material in this knowledge area description meets the criterion of being generally acceptable in terms of being "applicable to most projects, most of the time" and having "widespread consensus about their value and usefulness" [PMI, 1996]. These topics are those that receive the greatest coverage in both the original texts and additional materials suggested here.

Similarly, the Industrial Advisory Board definition of "study material of a software engineering licensing exam that a graduate would pass after completing four years of work experience" appears to be met. However, in this case the specific responsibilities of the graduate will obviously influence in what areas they have the opportunity to gain experience. Project management is often a more senior position and as such, graduates with four years of practice

may not have had significant experience in managing, at least large-scale, projects.

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

Compatible with various schools of thought within software engineering

Excluding debate on measurement theoretic issues there is little intense debate in the measurement field. There is nothing that appears to be controversial in the management process sub-area.

Compatible with breakdown in industry, literatures, and standards

The breakdown is in line with others proposed, and is particularly aligned with the IEEE/EIA Standard for Information Technology (ISO/IEC 12207) – *Software life cycle processes* and the Guide to the Project Management Body of Knowledge.

Depth and node density

The suggested guidelines have been met here.

Meaningful topic names

Key terms on software measures and measurement methods have been defined in [ISO/IEC 2000] 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". We recognize that this could make less obvious the connection between this work and many papers and books (including [Fenton and Pfleeger, 1997]).

Brevity of topic descriptions

Although they have been expanded significantly between the last draft and this, the descriptions remain adequately brief and to the point.

Specific reference material

Additional reference material for more specialized topics not covered adequately in the primary reference material has been added.

Proposed reference material (publicly available)

All material is publicly available.

Maximum number of core reference materials is 15

We have adhered to this limit.

Preference to IEEE or ACM copyrighted material

This is evident in the selection of reference material, especially the collections of papers.

5 MATRIX OF TOPICS VS. REFERENCE MATERIAL

The level of granularity used in Table 1 is a mixture of second and third level topics, depending on the specificity of the topic in question.

Topic	D&T	F&P	Pfle	Press	Reif	Somm	Thay
A. Organizational Management							
Policy management			Ch. 2		Ch. 2	Ch. 30	Ch. 2,4
Personnel management		Ch. 11	Ch. 3	Ch. 3	Ch. 7,8	Ch. 28	Ch. 7,8
Communication management				Ch. 3		Ch. 28	Ch. 1,3
Portfolio management				Ch. 10			
Procurement management				Ch. 5	Ch. 15	Ch. 2	
B. Process/project Management							
Initiation and scope definition							
Determination and negotiation of requirements	Ch. 4		Ch. 4	Ch. 5,11,12		Ch. 4-11	
Feasibility analysis				Ch. 10			
Review/revision of requirements						Ch. 4	
Planning							
Process planning	Ch. 5,11		Ch. 2	Ch. 2	Ch. 1,2,4	Ch. 1	Ch. 3
Project planning	Ch. 10		Ch. 3	Ch. 3,5	Ch. 3,4	Ch. 3	Ch. 4,6
Determine deliverables			Ch. 3	Ch. 3,7		Ch. 3	Ch. 4
Effort, schedule and cost estimation	Ch. 10	Ch. 12	Ch. 3	Ch. 5,7	Ch. 4,5	Ch. 3,29	Ch. 5
Resource allocation			Ch. 3	Ch. 5	Ch. 7,8	Ch. 3	Ch. 6,7
Risk management	Ch. 10		Ch. 3	Ch. 6	Ch. 11		Ch. 4
Quality management	Ch. 7,9			Ch. 8	Ch. 10	Ch. 30,31	Ch. 9,10
Plan management						Ch. 3	Ch. 4

Topic	D&T	F&P	Pfle	Press	Reif	Somm	Thay
Enactment							
Implementation of plans			Ch. 3			Ch. 3	
Implementation of measurement process		Ch. 13,14		Ch. 4	Ch. 9,10,12		Ch. 3,10
Monitor process	Ch. 7,9,10			Ch. 7	Ch. 9,10	Ch. 31	Ch. 3,9
Control process	Ch. 10			Ch. 9	Ch. 9,10		Ch. 3,9
Reporting					Ch. 9,10		Ch. 3,10
Review and evaluation							
Determining satisfaction of requirements					Ch. 9,10		Ch. 3,10
Reviewing and evaluating performance	Ch. 7		Ch. 7,8	Ch. 8	Ch. 9,10		Ch. 3,10
Closure							
Determining closure	Ch. 7				Ch. 9,10		Ch. 3,10
Closure activities			Ch. 11			Ch. 31	
C. Software Engineering Measurement							
Determining the goals of a measurement program							
Organizational objectives		Ch. 3,13		Ch. 4			
Software process improvement goals		Ch. 3,13	Ch. 12	Ch. 4	Ch. 2	Ch. 31	
Measurement selection							
Goal-driven measurement selection		Ch. 1,3,13, 14			Ch. 12		Ch. 10
Measurement validity		Ch. 2	Ch. 11				
Measuring software and its development							
Size measurement		Ch. 7		Ch. 4,18,23	Ch. 12	Ch. 30	
Structure measurement		Ch. 8		Ch. 18,23			
Resource measurement		Ch. 3,11				Ch. 29	
Quality measurement		Ch. 9,10		Ch. 4	Ch. 12	Ch. 30	
Collection of data							
Survey techniques and form design		Ch. 4,5					
Automated and manual data collection		Ch. 5		Ch. 4		Ch. 30	
Software measurement models							
Model building, calibration and evaluation		Ch. 4,6,13	Ch. 3,11,12			Ch. 29	
Implementation, interpretation and refinement of models		Ch. 6	Ch. 3,11,12	Ch. 4		Ch. 29	

Table 1: Topics and their references

6 RECOMMENDED REFERENCES FOR SOFTWARE ENGINEERING MANAGEMENT

The Topic-Reference matrix shown above requires the following references to be included in the Guide to the SWEBOK.

1) [D&T: Dorfman and Thayer, 1997] Merlin Dorfman and Richard H. Thayer (eds.). 1997. *Software engineering.* IEEE Computer Society. [Chapters 4, 5, 7, 9-11]

2) [F&P: Fenton and Pfleeger, 1997] Norman E. Fenton and Shari Lawrence Pfleeger. 1997. *Software metrics: a rigorous and practical approach.* PWS Publishing Company. [Chapters 1-14]

3) [Pfle: Pfleeger, 1998] Shari Lawrence Pfleeger. 1998. *Software engineering: theory and practice.* Prentice Hall. [Chapters 2-4, 7, 8, 11, 12]

4) [Press: Pressman, 1997] Roger S. Pressman. 1997. *Software engineering: a practitioner's approach. (Fourth edition)* McGraw-Hill. [Chapters 2-12, 18, 23]

5) [Reif: Reifer, 1997] Donald J. Reifer (ed.). 1997. *Software management*, 5th edition. IEEE Computer Society. [Chapters 1-5, 7-12, 15]

6) [Somm: Sommerville, 1996] Ian Sommerville. 1996. *Software engineering.* Addison-Wesley. [Chapters 1-11, 28-31]

7) [Thay: Thayer, 1997] Richard H. Thayer (ed.). 1997. *Software engineering project management.* IEEE Computer Society. [Chapters 1-10]

APPENDIX A – LIST OF FURTHER READINGS

The following readings are useful sources of information for this knowledge area.

Process/Project Management:

Adler, T.R., Leonard, J.G. and Nordgren, R.K. Improving risk management: moving from risk elimination to risk avoidance. *Information and Software Technology* 41: 29-34 (1999).

Baines, R. Across disciplines: risk, design, method, process, and tools. *IEEE Soft.* (July/Aug): 61-64 (1998)

Binder, R.V. Can a manufacturing quality model work for software? *IEEE Soft.* (September/October): 101-102,105 (1997).

Boehm, B.W. and DeMarco, T. Software risk management (Guest editors' introduction). *IEEE Soft.* (May/June): 17-19 (1997).

Carr, M.J. Risk management may not be for everyone. *IEEE Soft.* (May/June): 21,24 (1997).

Charette, R.N. Large-scale project management is risk management. *IEEE Soft.* (July): 110-117 (1996).

Charette, R.N., Adams, K.M. and White, M.B. Managing risk in software maintenance. *IEEE Soft.* (May/June): 43-50 (1997).

Collier, B., DeMarco, T. and Fearey, P. A defined process for project postmortem review. *IEEE Soft.* (July): 65-72 (1996).

Conrow, E.H. and Shishido, P.S. Implementing risk management on software intensive projects. *IEEE Soft.* (May/June): 83-89 (1997).

DeMarco, T. and Lister, T. *Peopleware: productive projects and teams*. Dorset House Publishing, 1987.

DeMarco, T. and Miller, A. Managing large software projects. *IEEE Soft.* (July): 24-27 (1996).

Favaro, J. and Pfleeger, S.L. Making software development investment decisions. *ACM SIGSoft Software Engineering Notes* 23(5): 69-74 (1998).

Fayad, M.E and Cline, M. Managing object-oriented software development. *Computer* (Sept): 26-31 (1996)

Fleming, R. A fresh perspective on old problems. *IEEE Soft.* (January/February): 106-113 (1999).

Garvey, P.R., Phair, D.J. and Wilson, J.A. An information architecture for risk assessment and management. *IEEE Soft.* (May/June): 25-34 (1997).

Gemmer, A. Risk management: moving beyond process. *Computer* (May): 33-43 (1997).

Glass, R.L. The ups and downs of programmer stress. *Communications of the ACM* 40(4): 17-19 (1997).

Glass, R.L. Short-term and long-term remedies for runaway projects. *Comm. ACM* 41(7): 13-15 (1998).

Glass, R.L. How not to prepare for a consulting assignment, and other ugly consultancy truths. *Communications of the ACM* 41(12): 11-13 (1998).

Henry, S.M. and Stevens, K.T. Using Belbin's leadership role to improve team effectiveness: an empirical investigation. *Journal of Systems and Software* 44: 241-250 (1999).

Hohmann, L. Coaching the rookie manager. *IEEE Soft.* (January/February): 16-19 (1999).

Hsia, P. Making software development visible. *IEEE Soft.* (March): 23-26 (1996).

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

Jackman, M. Homeopathic remedies for team toxicity. *IEEE Soft.* (July/August): 43-45 (1998).

Kansala, K. Integrating risk assessment with cost estimation. *IEEE Soft.* (May/June): 61-67 (1997).

Karlsson, J. and Ryan, K. A cost-value aproach for prioritizing requirements. *IEEE Soft.* (September/October): 87-74 (1997).

Karolak, D.W. *Software engineering risk management.* IEEE Computer Society, 1996.

Keil, M., Cule, P.E., Lyytinen, K. and Schmict, R.C. A framework for identifying software project risks. *Communications of the ACM* 41(11): 76-83 (1998).

Kitchenham, B. and Linkman, S. Estimates, uncertainty, and risk. *IEEE Soft.* (May/June): 69-74 (1997).

Leung, H.K.N. A risk index for software producers. *Software Maintenance: Research and Practice* 8: 281-294 (1996).

Lister, T. Risk management is project management for adults. *IEEE Soft.* (May/June): 20,22 (1997).

Mackey, K. Why bad things happen to good projects. *IEEE Soft.* (May): 27-32 (1996).

Mackey, K. Beyond Dilbert: creating cultures that work. *IEEE Soft.* (January-February): 48-49 (1998).

Madachy, R.J. Heuristic risk assessment using cost factors. *IEEE Soft.* (May/June): 51-59 (1997).

Martin, C. The need for software risk management tools. *Application Development Trends.* p.20,22.

McConell, S.C. *Rapid Development: Taming Wild Software Schedules.* Microsoft Press, 1996.

McConell, S.C. *Software Project Survival Guide.* Microsoft Press, 1997.

Moynihan, T. How experienced project managers assess risk. *IEEE Soft.* (May/June): 35-41 (1997).

Nesi, P. Managing OO projects better. *IEEE Soft.* (July/August): 50-60 (1998).

Nolan, A.J. Learning from success. *IEEE Soft.* (January/February): 97-105 (1999).

Parris, K.V.C. Implementing accountability. *IEEE Soft.* (July): 83-93 (1996).

Putnam, L.H. and Myers, W. *Industrial Strength Software: Effective Management Using Measurement.* Los Alamitos CA, IEEE Computer Society Press (1997) 309p.

Rodrigues, A.G. and Williams, T.M. System dynamics in software project management: towards the development of a formal integrated framework. *European Journal of Information Systems* 6: 51-66 (1997).

Ropponen, J. and Lyytinen, K. Can software risk management improve system development: an exploratory study. *European Journal of Information Systems* 6: 41-50 (1997).

Schmidt, C., Dart, P., Johnston, L., Sterling, L. and Thorne, P. Disincentives for communicating risk: a risk paradox. *Information and Software Technology* 41: 403-411 (1999).

Slaughter, S.A., Harter, D.E. and Krishnan, M.S. Evaluating the cost of software quality. *Communications of the ACM* 41(8): 67-73 (1998).

van Scoy, R.L. Software development risk: opportunity, not problem. CMU/SEI-92-TR-30, Software Engineering Institute, Carnegie Mellon University, 1992.

van Solingen, R., Berghout, E. and van Latum, F. Interrupts: just a minute never is. *IEEE Soft.* (September/October): 97-103 (1998).

Whitten, N. *Managing Software Development Projects: Formulas for Success.* Wiley, 1995.

Williams, R.C., Walker, J.A. and Dorofee, A.J. Putting risk management into practice. *IEEE Soft.* (May/June): 75-82 (1997).

Software Engineering Measurement:

Briand, L.C., Morasca, S. and Basili, V.R. Property-based software engineering measurement. *IEEE Transactions on Software Engineering* 22(1): 68-86 (1996).

Briand, L., El Emam, K. and Morasca, S. On the application of measurement theory in software engineering. *Empirical Software Engineering* 1: 61-88 (1996).

Briand, L.C., Morasca, S. and Basili, V.R. Response to: Comments on "Property-based software engineering measurement: refining the addivity properties". *IEEE Transactions on Software Engineering* 23(3): 196-197 (1997).

Brooks, F.P., Jr. No silver bullet: essence and accidents of software engineering. *Computer* (Apr.): 10-19 (1987).

Davis, A.M. Predictions and farewells. *IEEE Soft.* (July/August): 6-9 (1998).

Fenton, N.E. and Pfleeger, S.L. *Software Metrics: A Rigorous and Practical Approach.* London, International Thomson Computer Press (1997) 638p.

Fuggetta, A., Lavazza, L., Morasca, S., Cinti, S., Oldano, G. and Orazi, E. Applying GQM in an industrial software factory. *ACM Transactions on Software Engineering and Methodology* 7(4): 411-448 (1998).

Glass, R.L. The realities of software technology payoffs. *Communications of the ACM* 42(2): 74-79 (1999).

Grable, R., Jernigan, J., Pogue, C. and Divis, D. Metrics for small projects: experiences at the SED. *IEEE Soft.* (March/April): 21-29 (1999).

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

Hall, T. and Fenton, N. Implementing effective software metrics programs. *IEEE Soft.* (Mar/Apr): 55-64 (1997).

Kautz, K. Making sense of measurement for small organizations. *IEEE Soft.* (March/April): 14-20 (1999).

Kernighan, B. and Pike, R. Finding performance improvements. *IEEE Soft.* (March/April): 61-65 (1999).

McConnell, S. Software engineering principles. *IEEE Soft.* (March/April): 6-8 (1999).

Offen, R.J. and Jeffery, R. Establishing software measurement programs. *IEEE Soft.* (Mar/Apr): 45-53 (1997).

Pfleeger, S.L. Assessing measurement (Guest editor's introduction). *IEEE Soft.* (Mar/Apr): 25-26 (1997).

Pfleeger, S.L., Jeffery, R., Curtis, B. and Kitchenham, B. Status report on software measurement. *IEEE Soft.* (March/April): 33-43 (1997).

Robillard, P.N. The role of knowledge in software development. *Comm. of the ACM* 42(1): 87-92 (1999).

van Latum, F., van Solingen, R., Oivo, M., Hoisl, B., Rombach, D. and Ruhe, G. Adopting GQM-based measurement in an industrial environment. *IEEE Soft.* (January-February): 78-86 (1998).

Zelkowitz, M.V. and Wallace, D.R. Experimental models for validating technology. *Computer* (May): 23-31 (1998).

APPENDIX B – REFERENCES USED TO WRITE AND JUSTIFY THE DESCRIPTION

[IEEE/EIA, 1998] IEEE/EIA. 1998. Standard for Information Technology (ISO/IEC 12207) – *Software life cycle processes*. Institute of Electrical and Electronics Engineers/Electronic Industries Association Engineering Department.

[ISO93] ISO 1993. *International Vocabulary of Basic and General Terms in Metrology*, International Organization for Standardization.

[ISO/IEC, 1998] ISO/IEC 1998. 14143-1 *Software engineering - Software measurement - Functional size measurement - Definition of concepts,* International Organization for Standardization/International Electrotechnical Commission.

[ISO/IEC, 1999] ISO/IEC. 1999. Draft Technical Report (DTR) 16326 – *Software engineering – guide for the application of ISO/IEC 12207 to project management*. International Organization for Standardization/International Electrotechnical Commission.

[ISO/IEC, 2000] ISO/IEC Committee Draft (CD) 15939: Information technology - Software Measurement Process, International Organization for Standardization/International Electrotechnical Commission.

[Moore, 1998] James W. Moore. 1998. *Software engineering standards: a user's road map*. IEEE Computer Society.

[PMI, 1996] Project Management Institute Standards Committee. 1996. *A guide to the project management body of knowledge (PMBOK)*. Project Management Institute.

APPENDIX C – TABLE OF CORRESPONDENCE WITH PMBOK

PMBOK Knowledge Areas	PMBOK Knowledge Area Processes	7.1 ISO/IEC 12207 Management Process Activities				
		7.1.1 Initiation and Scope Definition	7.1.2 Planning	7.1.3 Enactment	7.1.4 Review and Evaluation	7.1.5 Closure
4. Project Integration Management	4.1 Project Plan Development	X	X			
	4.2 Project Plan Execution			X	X	
	4.3 Overall Change Control			X	X	
5. Project Scope Management	5.1 Initiation	X		X		
	5.2 Scope Planning	X	X			
	5.3 Scope Definition	X	X			
	5.4 Scope Verification	X			X	X
	5.5 Scope Change Control	X	X	X	X	
6. Project Time Management	6.1 Activity Definition	X	X			
	6.2 Activity Sequencing		X			
	6.3 Activity Duration Estimating		X	X	X	
	6.4 Schedule Development		X			
	6.5 Schedule Control			X	X	
7. Project Cost Management	7.1 Resources Planning	X	X			
	7.2 Cost Estimating	X	X	X		
	7.3 Cost Budgeting		X			
	7.4 Cost Control			X	X	
8. Project Quality Management	8.1 Quality Planning	X	X			
	8.2 Quality Assurance			X	X	
	8.3 Quality Control			X	X	
9. Project Human Resource Management	9.1 Organizational Planning	X	X		X	
	9.2 Staff Acquisition	X		X		
	9.3 Team Development	X		X		
10. Project Communications Management	10.1 Communications Planning	X	X			
	10.2 Information Distribution			X		
	10.3 Performance Reporting			X	X	
	10.4 Administrative Closure			X		X
11. Project Risk Management	11.1 Risk Identification	X		X		
	11.2 Risk Quantification	X		X		
	11.3 Risk Response Development		X	X	X	
	11.4 Risk Response Control	X	X	X	X	
12. Project Procurement Management	12.1 Procurement Planning	X	X			
	12.2 Solicitation Planning	X	X			
	12.3 Solicitation	X		X		
	12.4 Source Selection	X		X	X	
	12.5 Contract Administration			X	X	
	12.6 Contract Close-out		X			X

Table 2: Correspondence between PMBOK knowledge areas and ISO/IEC 12207 management process activities (taken from ISO/IEC Draft Technical Report (DTR) 16326)