CHAPTER 9

SOFTWARE ENGINEERING PROCESS

Khaled El Emam
Institute for Information Technology
National Research Council
Building M-50, Montreal Road
Ottawa, Ontario K1A 0R6, Canada
+1 (613) 998 4260
Khaled.el-emam@iit.nrc.ca

Table of Contents
1 Introduction.................................................................................. 1
2 Definition of the Software Engineering Process
   Knowledge Area........................................................................... 1
3 Breakdown of Topics for Software Engineering
   Process and Breakdown Rationale ............................................. 2
4 Key References vs. Topics Mapping ................................. 10
5 Recommended References for Software Process...... 12
Appendix A – List of Further Readings............................... 14

1 INTRODUCTION

The software engineering process Knowledge Area has witnessed dramatic growth over the last decade. This was partly due to a recognition by major acquirers of systems where software is a major component that process issues can have an important impact on the ability of their suppliers to deliver. Therefore, they encouraged a focus on the software engineering process as a way to remedy this. Furthermore, the academic community has recently pursued an active research agenda in developing new tools and techniques to support software engineering processes, and also empirically studying these processes and their improvement. It should also be recognized that many software engineering process issues are closely related to other disciplines, namely those in the management sciences, albeit they have used a different terminology. The industrial adoption of software engineering process technology has also been increasing, as demonstrated by a number of published success stories. Therefore, there is in fact an extensive body of knowledge on the software engineering process.

Keywords
software process, software process improvement, software process modeling, software process measurement, organizational change, software process assessment.

Acronyms
CBA IPI CMM Based Appraisal for Internal Process Improvement
CMM Capability Maturity Model
EF Experience Factory
FP Function Points
G/Q/M Goal/Question/Metric
HRM Human Resources Management
IDEAL Initiating-Diagnosing-Establishing-Acting-Leaning (model)
MIS Management Information Systems
PDCA Plan-Do-Check-Act (cycle)
QIP Quality Improvement Paradigm
ROI Return on Investment
SCE Software Capability Evaluation
SEPG Software Engineering Process Group
SW-CMM Capability Maturity Model for Software

2 DEFINITION OF THE SOFTWARE ENGINEERING
   PROCESS KNOWLEDGE AREA

The software engineering process Knowledge Area (KA) can potentially be examined at two levels. The first level encompasses the technical and managerial activities within the software engineering process that are performed during software acquisition, development, maintenance, and retirement. The second is the meta-level, which is concerned with the definition, implementation,
measurement, management, change and improvement of the software engineering process itself. The latter we will term **software process engineering**.

The first level is covered by the other KA’s of this Guide to the Software Engineering Body of Knowledge. This Knowledge Area is concerned with the second: **software process engineering**.

### 2.1 Scope

This Knowledge Area does not explicitly address the following topics:

- Human resources management (for example, as embodied in the People CMM [30][31])
- Systems engineering processes

While important topics in themselves, they are outside the direct scope of software process engineering. However, where relevant, interfaces (or references to interfaces) to HRM and systems engineering will be addressed.

### 2.2 Currency of Material

The software process engineering discipline is rapidly changing, with new paradigms and new models. The breakdown and references included here are pertinent at the time of writing. An attempt has been made to focus on concepts to shield the knowledge area description from changes in the field, but of course this cannot be 100% successful, and therefore the material here must be evolved over time. A good example is the on-going CMM Integration effort (see [71]), both of which are likely to have a considerable influence on the software process community once widely disseminated, and would therefore have to be accommodated in the knowledge area description.

In addition, where Internet addresses are provided for reference material, these addresses were verified at the time of press. However, there are no guarantees that the documents will still be available on-line at the same location in the future.

### 2.3 Structure of the KA

To structure this KA in a way that is directly related to practice, we have defined a generic process model for software process engineering (see Figure 1). This model identifies the activities that are performed in a process engineering context. The topics are mapped to these activities. The advantage of such a structure is that one can see, in practice, where each of the topics is relevant, and provides an overall rationale for the topics. This generic model is based on the PDCA (plan-do-check-act) cycle (also see [79]).

### 3 Breakdown of Topics for Software Engineering Process and Breakdown Rationale

The following figure shows the breakdown of topics in this knowledge area. Further explanation is provided in the subsequent sections.

**Software Engineering Process Concepts**

- Themes
- Terminology

**Process Infrastructure**

- The Software Engineering Process Group
- The Experience Factory

**Process Measurement**

- Methodology in Process Measurement
- Process Measurement Paradigms
  - Analytic Paradigm
  - Benchmarking Paradigm

**Process Definition**

- Types of Process Definitions
- Life Cycle Framework Models
- Software Life Cycle Process Models
- Notations for Process Definitions
- Process Definition Methods
  - Automation

**Qualitative Process Analysis**

- Process Definition Review
- Root Cause Analysis

**Process Implementation and Change**

- Paradigms for Process Implementation and Change
- Guidelines for Process Implementation and Change
- Evaluating the Outcome of Process Implementation and Change

### 3.1 Software Engineering Process Concepts

#### 3.1.1 Themes

Dowson [35] notes that “All process work is ultimately directed at ‘software process assessment and improvement’”. This means that the objective is to implement new or better processes in actual practices, be they individual, project or organizational practices.
We describe the main topics in the software process engineering (i.e., the meta-level that has been alluded to earlier) area in terms of a cycle of process change, based on the commonly known PDCA cycle. This cycle highlights that individual process engineering topics are part of a larger process to improve practice, and that process evaluation and feedback is an important element of process engineering.

Software process engineering consists of four activities as illustrated in the model in Figure 1. The activities are sequenced in an iterative cycle allowing for continuous feedback and improvement of the software process.

The “Establish Process Infrastructure” activity consists of establishing commitment to process implementation and change (including obtaining management buy-in), and putting in place an appropriate infrastructure (resources and responsibilities) to make it happen.

The activities “Planning of Process Implementation and Change” and “Process Implementation and Change” are the core ones in process engineering, in that they are essential for any long-lasting benefit from process engineering to accrue. In the planning activity the objective is to understand the current business objectives and process needs of the organization, identify its strengths and weaknesses, and make a plan for process implementation and change. In “Process Implementation and Change”, the objective is to execute the plan, deploy new processes (which may involve, for example, the deployment of tools and training of staff), and/or change existing processes.

The fourth activity, “Process Evaluation” is concerned with finding out how well the implementation and change went; whether the expected benefits materialized. This is then used as input for subsequent cycles.

At the centre of the cycle is the “Process Experience Base”. This is intended to capture lessons from past iterations of the cycle (e.g., previous evaluations, process definitions, and plans). Evaluation lessons can be qualitative or quantitative. No assumptions are made about the nature or technology of this “Process Experience Base”, only that it be a persistent storage. It is expected that during subsequent iterations of the cycle, previous experiences will be adapted and reused. It is also important to continuously re-assess the utility of information in the experience base to ensure that obsolete information does not accumulate.

With this cycle as a framework, it is possible to map the topics in this knowledge area to the specific activities where they would be most relevant. This mapping is also shown in Figure 1. The bulleted boxes contain the Knowledge Area topics.

It should be noted that this cycle is not intended to imply that software process engineering is relevant to only large organizations. To the contrary, process-related activities can, and have been, performed successfully by small organizations, teams, and individuals. The way the activities defined in the cycle are performed would be...
different depending on the context. Where it is relevant, we will present examples of approaches for small organizations.

**Process Implementation and Change:** This is concerned with deploying processes for the first time and with changing existing process. This topic focuses on organizational change. It describes the paradigms, infrastructure, and critical success factors necessary for successful process implementation and change. Within the scope of this topic, we also present some conceptual issues about the evaluation of process change.

The main, generally accepted, themes in the software engineering process field have been described by Dowson in [35]. His themes are a subset of the topics that we cover in this KA. Below are Dowson’s themes:

- Process definition: covered in topic 3.4 of this KA breakdown
- Process assessment: covered in topic 3.3 of this KA breakdown
- Process improvement: covered in topics 3.2 and 3.6 of this KA breakdown
- Process support: covered in topic 3.4 of this KA breakdown

We also add one theme in this KA description, namely the qualitative process analysis (covered in topic 3.5).

### 3.1.2 Terminology

There is no single universal source of terminology for the software engineering process field, but good sources that define important terms are [51][96], and the vocabulary (Part 9) in the ISO/IEC TR 15504 documents [81].

#### 3.2 Process Infrastructure

At the initiation of process engineering, it is necessary to have an appropriate infrastructure in place. This includes having the resources (competent staff, tools and funding), as well as the assignment of responsibilities. This is an indication of management commitment to and ownership of the process engineering effort. Various committees may have to be established, such as a steering committee to oversee the process engineering effort.

It is widely recognized that a team separate from the developers/maintainers must be set up and tasked with process analysis, implementation and change [16]. The main reason for this is that the priority of the developers/maintainers is to produce systems or releases, and therefore process engineering activities will not receive as much attention as they deserve or need. This, however, should not mean that the project organization is not involved in the process engineering effort at all. To the contrary, their involvement is essential. Especially in a small organization, outside help (e.g., consultants) may be required to assist in making up a process team.

Two types of infrastructure are have been used in practice: the Experience Factory [8][9] and the Software Engineering Process Group [54]. The IDEAL handbook [100] provides...
a good description of infrastructure for process improvement in general.

3.2.1 The Software Engineering Process Group

The SEPG is intended to be the central focus for process improvement within an organization. The SEPG typically has the following ongoing activities:

- Obtains and maintains the support of all levels of management
- Facilitates software process assessments (see below)
- Works with line managers whose projects are affected by changes in software engineering practice
- Maintains collaborative working relationships with software engineers
- Arranges and supports any training or continuing education related to process implementation and change
- Tracks, monitors, and reports on the status of particular improvement efforts
- Facilitates the creation and maintenance of process definitions
- Maintains a process database
- Provides process consultation to development projects and management
- Participate in integrating software engineering processes with other organizational processes, such as systems engineering

Fowler and Rifkin [54] suggest the establishment of a steering committee consisting of line and supervisory management. This would allow management to guide process implementation and change, align this effort with strategic and business goals of the organization, and also provides them with visibility. Furthermore, technical working groups may be established to focus on specific issues, such as selecting a new design method to setting up a measurement program.

3.2.2 The Experience Factory

The concept of the EF separates the project organization (e.g., the software development organization) from the improvement organization. The project organization focuses on the development and maintenance of applications. The EF is concerned with improvement. Their relationship is depicted in Figure 2.

The EF is intended to institutionalize the collective learning of an organization by developing, updating, and delivering to the project organization experience packages (e.g., guide books, models, and training courses).2 The project organization offers to the experience factory their products, the plans used in their development, and the data gathered during development and operation. Examples of experience packages include:

- resource models and baselines (e.g., local cost models, resource allocation models)
- change and defect baselines and models (e.g., defect prediction models, types of defects expected for the application)
- project models and baselines (e.g., actual vs. expected product size)
- process definitions and models (e.g., process models for Cleanroom, Ada waterfall model)
- method and technique evaluations (e.g., best method for finding interface faults)
- products and product parts (e.g., Ada generics for simulation of satellite orbits)
- quality models (e.g., reliability models, defect slippage models, ease of change models), and
- lessons learned (e.g., risks associated with an Ada development).

Figure 2 The relationship between the Experience Factory and the project organization as implemented at the Software Engineering Laboratory at NASA/GSFC. This diagram is reused here from [10] with permission of the authors.

3.3 Process Measurement

Process measurement, as used here, means that quantitative information about the process is collected, analyzed, and interpreted. Measurement is used to identify the strengths and weaknesses of processes, and to evaluate processes after they have been implemented and/or changed (e.g., evaluate the ROI from implementing a new process).4

---

2 Also referred to as process assets.

3 Baselines can be interpreted as descriptive reports presenting the current status.

4 Process measurement may serve other purposes as well. For example, process measurement is useful for managing a software project. Some of these are covered in the Software Engineering Management and
An important assumption made in most process engineering work is illustrated by the path diagram in Figure 3. Here, we assume that the process has an impact on process outcomes. Process outcomes could be, for example, product quality (faults per KLOC or per FP), maintainability (effort to make a certain type of change), productivity (LOC or FP per person month), time-to-market, the extent of process variation, or customer satisfaction (as measured through a customer survey). This relationship depends on the particular context (e.g., size of the organization, or size of the project).

Not every process will have a positive impact on all outcomes. For example, the introduction of software inspections may reduce testing effort and cost, but may increase interval time if each inspection introduces large delays due to the scheduling of large inspection meetings [131]. Therefore, it is preferred to use multiple process outcome measures that are important for the organization’s business.

In general, we are most concerned about the process outcomes. However, in order to achieve the process outcomes that we desire (e.g., better quality, better maintainability, greater customer satisfaction) we have to implement the appropriate process.

Of course, it is not only process that has an impact on outcomes. Other factors such as the capability of the staff and the tools that are used play an important role. Furthermore, the extent to which the process is institutionalized or implemented (i.e., process fidelity) is important as it may explain why “good” processes do not give the desired outcomes.

One can measure the quality of the software process itself, or the process outcomes. The methodology in Section 3.3.1 is applicable to both. We will focus in Section 3.3.2 on process measurement since the measurement of process outcomes is more general and applicable in other Knowledge Areas.

3.3.1 Methodology in Process Measurement

A number of guides for measurement are available [108][109][126]. All of these describe a goal-oriented process for defining measures. This means that one should start from specific information needs and then identify the measures that will satisfy these needs, rather than start from specific measures and try to use them. A good practical text on establishing and operating a measurement program has been produced by the Software Engineering Laboratory [123]. This also discusses the cost of measurement. Texts that present experiences in implementing measurement in software organizations include [86][105][115]. An emerging international standard that defines a generic measurement process is also available (ISO/IEC CD 15939: Information Technology – Software Measurement Process) [82].

Two important issues in the measurement of software engineering processes are the reliability and validity of measurement. Reliability is concerned with random measurement error. Validity is concerned with the ability of the measure to really measure what we think it is measuring.

Reliability becomes important when there is subjective measurement, for example, when assessors assign scores to a particular process. There are different types of validity that ought to be demonstrated for a software process measure, but the most critical one is predictive validity. This is concerned with the relationship between the process measure and the process outcome. A discussion of both of these and different methods for achieving them can be found in [40][59]. An IEEE Standard describes a methodology for validating metrics (IEEE Standard for a Software Quality Metrics Methodology. IEEE Std 1061-1998) [76].

An overview of existing evidence on reliability of software process assessments can be found in [43][49], and for predictive validity in [44][49][59][88].

3.3.2 Process Measurement Paradigms

Two general paradigms that are useful for characterizing the type of process measurement that can be performed have been described by Card [21]. The distinction made by Card is a useful conceptual one. Although, there may be overlaps in practice.

The first is the analytic paradigm. This is characterized as relying on “quantitative evidence to determine where improvements are needed and whether an improvement initiative has been successful.” The second, the benchmarking paradigm, “depends on identifying an ‘excellent’ organization in a field and documenting its

Figure 3 Path diagram showing the relationship between process and outcomes (results). The context affects the relationship between the process and process outcomes. This means that this process to process outcome relationship depends on the context value.

© IEEE – Trial Version 1.00 – May 2001

9–6
Benchmarking assumes that if a less-proficient organization adopts the practices of the excellent organization, it will also become excellent. Of course, both paradigms can be followed at the same time, since they are based on different types of information.

We use these paradigms as general titles to distinguish between different types of measurement.

3.3.2.1 Analytic Paradigm

The analytic paradigm is exemplified by the Quality Improvement Paradigm (QIP) consisting of a cycle of understanding, assessing, and packaging [124].

Experimental and Observational Studies

- Experimentation involves setting up controlled or quasi experiments in the organization to evaluate processes [101]. Usually, one would compare a new process with the current process to determine whether the former has better process outcomes. Correlational (nonexperimental) studies can also provide useful feedback for identifying process improvements (e.g., for example, see the study described by Agresti [2]).

Process Simulation

- The process simulation approach can be used to predict process outcomes if the current process is changed in a certain way [117]. Initial data about the performance of the current process needs to be collected, however, as a basis for the simulation.

Orthogonal Defect Classification

- Orthogonal Defect Classification is a technique that can be used to link faults found with potential causes. It relies on a mapping between fault types and fault triggers [22][23]. There exists an IEEE Standard on the classification of faults (or anomalies) that may also be useful in this context (IEEE Standard for the Classification of Software Anomalies. IEEE Std 1044-1993) [74].

Statistical Process Control

- Placing the software process under statistical process control, through the use of control charts and their interpretations, is an effective way to identify stability, or otherwise, in the process. One recent book provides a good introduction to SPC in the context of software engineering [53].

The Personal Software Process

- This defines a series of improvements to an individual’s development practices in a specified order [70]. It is ‘bottom-up’ in the sense that it stipulates personal data collection and improvements based on the data interpretations.

3.3.2.2 Benchmarking Paradigm

This paradigm involves measuring the maturity of an organization or the capability of its processes. The benchmarking paradigm is exemplified by the software process assessment work. A general introductory overview of process assessments and their application is provided in [135].

- Process assessment models

   An assessment model captures what are believed to be good practices. The good practices may pertain to technical software engineering activities only, or may also encompass, for example, management, systems engineering, and human resources management activities as well.

   Architectures of assessment models

   There are two general architectures for an assessment model that make different assumptions about the order in which processes must be measured: the continuous and the staged architectures [110]. At this point it is not possible to make a recommendation as to which approach is better than another. They have considerable differences. An organization should evaluate them to see which are most pertinent to their needs and objectives when selecting a model.

   Assessment models

   The most commonly used assessment model in the software community is the SW-CMM [122]. It is also important to recognize that ISO/IEC 15504 is an emerging international standard on software process assessments [42][81]. It defines an exemplar assessment model and conformance requirements on other assessment models. ISO 9001 is also a common assessment model and conformance requirements on other assessment models. ISO 9001 is also a common model that has been applied by software organizations (usually in conjunction with ISO 9000-1) [132]. Other notable examples of assessment models are Trillium [25], Bootstrap [129], and the requirements engineering capability model [128]. There are also maturity models for other software processes available, such as for testing [18][19][20], a measurement maturity model [17], and a maintenance maturity model [36] (although, there have been many more capability and maturity models that have been defined, for example, for design, documentation, and formal methods, to name a few). A maturity model for systems engineering has also been developed, which would be useful where a project or organization is involved in the development and maintenance of systems including software [39]. The applicability of assessment models to small organizations is addressed in [85][120], where assessments models tailored to small organizations are presented.

---

7 These are intended as examples of the analytic paradigm, and reflect what is currently done in practice. Whether a specific organization uses all of these techniques will depend, at least partially, on its maturity.

8 In some instances the term “appraisal” is used instead of assessment, and the term “capability evaluation” is used when the appraisal is for the purpose of contract award.
• Process assessment methods

In order to perform an assessment, a specific assessment method needs to be followed. In addition to producing a quantitative score that characterizes the capability of the process (or maturity of the organization), an important purpose of an assessment is to create a climate for change within the organization [37]. In fact, it has been argued that the latter is the most important purpose of doing an assessment [38].

The most well known method that has a reasonable amount of publicly available documentation is the CBA IPI [37]. This method focuses on assessments for the purpose of process improvement using the SW-CMM. Many other methods are refinements of this for particular contexts. Another well known method using the SW-CMM, but for supplier selection, is the SCE [6]. The activities performed during an assessment, the distribution of effort on these activities, as well as the atmosphere during an assessment is different if it is for the purpose of improvement versus contract award. Requirements on both types of methods that reflect what are believed to be good assessment practices are provided in [81][99].

There have been criticisms of various models and methods following the benchmarking paradigm, for example [12][50][62][87]. Most of these criticisms were concerned with the empirical evidence supporting the use of assessments models and methods. However, since the publication of these articles, there has been an accumulation of systematic evidence supporting the efficacy of process assessments [24][47][48][60][64][65][66][94].

3.4 Process Definition

Software engineering processes are defined for a number of reasons, including: facilitating human understanding and communication, supporting process improvement, supporting process management, providing automated process guidance, and providing automated execution support [29][52][68]. The types of process definitions required will depend, at least partially, on the reason.

It should be noted also that the context of the project and organization will determine the type of process definition that is most important. Important variables to consider include the nature of the work (e.g., maintenance or development), the application domain, the structure of the delivery process (e.g., waterfall, incremental, evolutionary), and the maturity of the organization.

There are different approaches that can be used to define and document the process. Under this topic the approaches that have been presented in the literature are covered, although at this time there is no data on the extent to which these are used in practice.

3.4.1 Types of Process Definitions

Processes can be defined at different levels of abstraction (e.g., generic definitions vs. tailored definitions, descriptive vs. prescriptive vs. proscriptive). The differentiation amongst these has been described in [69][97][111].

Orthogonal to the levels above, there are also types of process definitions. For example, a process definition can be a procedure, a policy, or a standard.

3.4.2 Life Cycle Framework Models

These framework models serve as a high level definition of the phases that occur during development. They are not detailed definitions, but only the high level activities and their interrelationships. The common ones are: the waterfall model, throwaway prototyping model, evolutionary prototyping model, incremental/iterative development, spiral model, reusable software model, and automated software synthesis. (see [11][28][84][111][113]). Comparisons of these models are provided in [28][32], and a method for selection amongst many of them in [3].

3.4.3 Software Life Cycle Process Models

Definitions of life cycle process models tend to be more detailed than framework models. Another difference being that life cycle process models do not attempt to order their processes in time. Therefore, in principle, the life cycle processes can be arranged to fit any of the life cycle frameworks. The two main references in this area are ISO/IEC 12207: Information Technology – Software Life Cycle Processes [80] and ISO/IEC TR 15504: Information Technology – Software Process Assessment [42][81]. Extensive guidance material for the application of the former has been produced by the IEEE (Guide for Information Technology - Software Life Cycle Processes - Life cycle data, IEEE Std 12207.1-1998, and Guide for Information Technology - Software Life Cycle Processes – Implementation. Considerations. IEEE Std 12207.2-1998) [77][78]. The latter defines a two dimensional model with one dimension being processes, and the second a measurement scale to evaluate the capability of the processes. In principle, ISO/IEC 12207 would serve as the process dimension of ISO/IEC 15504.

The IEEE standard on developing life cycle processes also provides a list of processes and activities for development and maintenance (IEEE Standard for Developing Software Life Cycle Processes, IEEE Std 1074-1991) [73], and provides examples of mapping them to life cycle framework models. A standard that focuses on maintenance processes is also available from the IEEE (IEEE Standard for Software Maintenance, IEEE Std 1219-1992) [75].
3.4.4 Notations for Process Definitions

Different elements of a process can be defined, for example, activities, products (artifacts), and resources [68]. Detailed frameworks that structure the types of information required to define processes are described in [4][98].

There are a large number of notations that have been used to define processes. They differ in the types of information defined in the above frameworks that they capture. A text that describes different notations is [125].

Because there is no data on which of these was found to be most useful or easiest to use under which conditions, this Guide covers what seemingly are popular approaches in practice: data flow diagrams [55], in terms of process purpose and outcomes [81], as a list of processes decomposed in constituent activities and tasks defined in natural language [80], Statecharts [89][117] (also see [63] for a comprehensive description of Statecharts), ETVX [116], Actor-Dependency modeling [14][134], SADT notation [102], Petri nets [5], IDEF0 [125], rule-based [7], and System Dynamics [1]. Other process programming languages have been devised, and these are described in [29][52][68].

3.4.5 Process Definition Methods

These methods specify the activities that must be performed in order to develop and maintain a process definition. These may include eliciting information from developers to build a descriptive process definition from scratch, and to tailoring an existing standard or commercial process. Examples of methods that have been applied in practice are [13][14][90][98][102]. In general, there is a strong similarity amongst them in that they tend to follow a traditional software development life cycle.

3.4.6 Automation

Automated tools either support the execution of the process definitions, or they provide guidance to humans performing the defined processes. In cases where process analysis is performed, some tools allow different types of simulations (e.g., discrete event simulation).

There exist tools that support each of the above process definition notations. Furthermore, these tools can execute the process definitions to provide automated support to the actual processes, or to fully automate them in some instances. An overview of process modeling tools can be found in [52], and of process-centered environments in [57][58].

Recent work on the application of the Internet to the provision of real-time process guidance is described in [91].

3.5 Qualitative Process Analysis

The objective of qualitative process analysis is to identify the strengths and weaknesses of the software process. It can be performed as a diagnosis before implementing or changing a process. It could also be performed after a process is implemented or changed to determine whether the change has had the desired effect.

Below we present two techniques for qualitative analysis that have been used in practice. Although it is plausible that new techniques would emerge in the future.

3.5.1 Process Definition Review

Qualitative evaluation means reviewing a process definition (either a descriptive or a prescriptive one, or both), and identifying deficiencies and potential process improvements. Typical examples of this are presented in [5][89]. An easily operational way to analyze a process is to compare it to an existing standard (national, international, or professional body), such as ISO/IEC 12207 [80].

With this approach, one does not collect quantitative data on the process. Or if quantitative data is collected, it plays a supportive role. The individuals performing the analysis of the process definition use their knowledge and capabilities to decide what process changes would potentially lead to desirable process outcomes.

3.5.2 Root Cause Analysis

Another common qualitative technique that is used in practice is a “Root Cause Analysis”. This involves tracing back from detected problems (e.g., faults) to identify the process causes, with the aim of changing the process to avoid the problems in the future. Examples of this for different types of processes are described in [13][27][41][107].

With this approach, one starts from the process outcomes, and traces back along the path in Figure 3 to identify the process causes of the undesirable outcomes. The Orthogonal Defect Classification technique described in Section 3.3.2.1 can be considered a more formalized approach to root cause analysis using quantitative information.

3.6 Process Implementation and Change

This topic describes the situation when processes are deployed for the first time (e.g., introducing an inspection process within a project or a complete methodology, such as Fusion [26] or the Unified Process [83]), and when current processes are changed (e.g., introducing a tool, or optimizing a procedure). In both instances, existing practices have to be modified. If the modifications are extensive, then changes in the organizational culture may be necessary.

3.6.1 Paradigms for Process Implementation and Change

Two general paradigms that have emerged for driving process implementation and change are the Quality Improvement Paradigm (QIP) [124] and the IDEAL model.

---

9 This can also be termed “process evolution”.

© IEEE – Trial Version 1.00 – May 2001
The two paradigms are compared in [124]. A concrete instantiation of the QIP is described in [16].

3.6.2 Guidelines for Process Implementation and Change

Process implementation and change is an instance of organizational change. Most successful organizational change efforts treat the change as a project in its own right, with appropriate plans, monitoring, and review.

Guidelines about process implementation and change within software engineering organizations, including action planning, training, management sponsorship and commitment, and the selection of pilot projects, and that cover both the transition of processes and tools, are given in [33][92][95][104][114][120][127][130][133]. An empirical study evaluating success factors for process change is reported in [46]. Grady describes the process improvement experiences at Hewlett-Packard, with some general guidance on implementing organizational change [61]. The role of change agents in this activity should not be underestimated. Without the enthusiasm, influence, credibility, and persistence of a change agent, organizational change has little chance of succeeding. This is further discussed in [72].

Process implementation and change can also be seen as an instance of consulting (either internal or external). A suggested text, and classic, on consulting is that of Schein [121].

One can also view organizational change from the perspective of technology transfer. The classic text on the stages of technology transfer is that by Rogers [119]. Software engineering articles that discuss technology transfer, and the characteristics of recipients of new technology (which could include process related technologies) are [112][118].

3.6.3 Evaluating the Outcome of Process Implementation and Change

Evaluation of process implementation and change outcomes can be qualitative or quantitative. The topics above on qualitative analysis and measurement are relevant when evaluating implementation and change since they describe the techniques. Below we present some conceptual issues that become important when evaluating the outcome of implementation and change.

There are two ways that one can approach evaluation of process implementation and change. One can evaluate it in terms of changes to the process itself, or in terms of changes to the process outcomes (for example, measuring the Return on Investment from making the change). This issue is concerned with the distinction between cause and effect (as depicted in the path diagram in Figure 3), and is discussed in [16].

Sometimes people have very high expectations about what can be achieved in studies that evaluate the costs and benefits of process implementation and change. A pragmatic look at what can be achieved from such evaluation studies is given in [67].

Overviews of how to evaluate process change, and examples of studies that do so can be found in [44][59][88][92][93][101].

4 KEY REFERENCES VS. TOPICS MAPPING

Below are the matrices linking the topics to key references. In an attempt to limit the number of references and the total number of pages, as requested, some relevant articles are not included in this matrix. The reference list below provides a more comprehensive coverage.

In the cells, where there is a check mark it indicates that the whole reference (or most of it) is relevant. Otherwise, specific chapter numbers are provided in the cell.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineering Process Concepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Themes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Terminology</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Infrastructure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Software Engineering Process Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Experience Factory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Measurement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Methodology in Process Measurement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Measurement Paradigms</td>
<td>Ch. 1, 7</td>
<td>Ch. 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Definition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------</td>
<td>------------</td>
<td>----------------</td>
<td>---------------</td>
<td>----------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Definitions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life Cycle Framework Models</td>
<td>Ch. 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Life Cycle Process Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notations for Process Definitions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Definition Methods</td>
<td>Ch. 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automation</td>
<td>Ch. 2</td>
<td>Ch. 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Qualitative Process Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Definition Review</td>
<td>Ch. 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root Cause Analysis</td>
<td>Ch. 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Implementation and Change</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paradigms for Process Implementation and Change</td>
<td>Ch. 1, 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guidelines for Process Implementation and Change</td>
<td>Ch. 11</td>
<td>Ch. 4</td>
<td>Ch. 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluating the Outcome of Process Implementation and Change</td>
<td>Ch. 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineering Process Concepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Themes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminology</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Infrastructure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Software Engineering Process Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Experience Factory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Measurement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Methodology in Process Measurement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Measurement Paradigms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Definition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Types of Process Definitions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life Cycle Framework Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch. 11</td>
</tr>
<tr>
<td>Software Life Cycle Process Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notations for Process Definitions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Definition Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5 RECOMMENDED REFERENCES FOR SOFTWARE PROCESS

The following are the key references that are recommended for this knowledge area. The mapping to the topics is given in Section 4.


This IEEE edited book provides detailed chapters on the software process assessment and improvement area. It could serve as a general reference for this knowledge area, however, specifically chapters 1, 7, and 11 cover quite a bit of ground in a succinct manner.


This IEEE edited book describes the emerging ISO/IEC 15504 international standard and its rationale. Chapter 3 provides a description of the overall architecture of the standard, which has since then been adopted in other assessment models.


This general software engineering reference has a good chapter, chapter 2, that discusses many issues related to the process modeling area.


This edited book provides a good overview of the process area, and covers modeling as well as assessment and improvement. Chapters 1 and 2 are reviews of modeling techniques and tools, and chapter 4 gives a good overview of the human and organizational issues that arise during process implementation and change.


This IEEE edited book provides a comprehensive perspective on process assessment and improvement efforts in Europe. Chapter 7 is a review of the costs and benefits of process improvement, with many references to prior work. Chapter 16 describes factors that affect the success of process improvement.


This IEEE book provides a comprehensive framework and guidance on software engineering standards. Chapter 13 is the process standards chapter.


This article provides an overview of different types of process definitions and relates them within an organizational context.


This article provides an overview of the main themes in the software process area. Although not recent, most of the issues raised are still valid today.


This article was one of the first attempts to define terminology in the software process area. Most of its terms are commonly used nowadays.


This article provides a pragmatic look at using measurement in the context of process improvement, and discusses most of the issues related to setting up a measurement program.


This is a standard reference on the concepts of the QIP and EF.


This is the standard reference on setting up and running an SEPG.


Chapter 11 of this IEEE volume gives a good overview of contemporary life cycle models.


This chapter provides the most up-to-date review of evidence supporting process assessment and improvement, as well as a historical perspective on some of the early MIS work.
APPENDIX A – LIST OF FURTHER READINGS


