Preface

"*As projects get more complicated, managers stop learning from their experience.* It is important to understand how that happens and how to change it….Fallible estimates: In software development, initial estimates for a project shape the trajectory of decisions that a manager makes over its life. For example, estimates of the productivity of the team members influence decisions about the size of the team, which in turn affect the team's actual output. *The trouble is that initial estimates usually turn out to be wrong.*" (Sengupta, 2008)

This book aims directly to increase the awareness among managers and practitioners that estimation is as important as the work to be done in software and systems development. You can manage what you can measure!

Readers will find in this book a collection of lessons learned from the worldwide "metrics community," which we have documented and enhanced with our own experiences in the field of software measurement and estimating. Our goal is to support our readers to harvest the benefits of estimating and improve their software development processes. We present the 5 ISO/IEC-acknowledged Functional Sizing Methods with variants, experiences, counting rules, and case studies – and most importantly, illustrate through practical examples how to use functional size measurement to produce realistic estimates. *The book is written in a practical manner, especially for the busy practitioner community. It is aimed to be used as a manual and an assistant for everyday work.*

Estimation can be a win–lose job: it has to be done professionally to enable transparency, efficiency, and control of IT projects.

Software project estimation is the first step to determine how successful projects, processes, and product goals will develop and also how they will be measured and how their goals will be reached.

The thesis presented in this book is that software project estimation can be done in a highly professional manner and that it can be done accurately. *The authors also point out that the process of estimation and the required time for it must be planned "a priori"!*

The first step for the success of a software project is to ensure that it is started in a professional manner. This requires a planning period supported by

a highly professional estimation process to ensure a solid foundation for project planning. Accurate estimates require quantitative measurements, ideally tool-based, to reduce measurement variations. *Furthermore, estimating does not gain the respect it deserves when it is done using only paper and pencil support – when the software engineers who provide the input data work professionally with the newest technologies.*

The application of an estimation method as well as the use of estimation tools and benchmarking data are nowadays "sine qua non" conditions for best practices in software engineering. In the unanimous opinion of estimation experts, this is the worldwide "state of the art."

Software project managers must also monitor and actualize their estimates during the project. *Estimates and project measures provide key risk indicators* and hence are excellent for the tracking of the progress of a software project and the monitoring of its success – that is, *they can provide valuable early warning signals if set up properly!* Fire fighting can be exciting, but does not help in fire prevention nor in the avoidance of significant costs and delays. A proper estimation process presents an opportunity for people tired of fire fighting to correctly plan and manage their software projects.

Estimation is an activity of the right brain: (the right brain being known for emotions and imagination, and ideas about the future and the unknown). *Estimation can also be performed with the left brain* (where logic and experience, and ideas about the past and known reside).

History of This Book

This book has a history as long as it took to implement a software measurement and metrics program in the IT department of an international insurance company in Germany. The initial text was published as the diploma thesis of Axel Fabry, when he was a student of Manfred Bundeschuh, working in a practicum project to plan and initiate the estimation program for the IT department. Fabry's thesis reported lessons learned about the trip wires involved with the implementation of estimating and laid the foundation for the first edition of this book. Regrettably, Mr. Fabry could not support the second edition, and its actualizations were done by Manfred Bundschuh.

The translation of the second edition into English was triggered by the German publisher Springer, who asked for an English translation and update, which has become the version you now possess. This led to the involvement and beneficial improvement, enhancement and actualization of the book done by the American author Carol Dekkers.

Why did Springer ask for an English translation and further updating of the successful German book on software estimation?

Initially, the demand emanated from colleagues at the European metrics organizations in Spain and Italy, and later from others who heard about the benefits gained by introducing metrics in Germany over the past years.

Secondly, the ISBSG collection of figures published as *The Benchmark* and other products are featured prominently in this book. These form a treasure trove of data that are of interest in the whole of the English-speaking metrics community.

Thirdly, this book presents an orderly overview of software -estimation, -metrics, -measurement, -measurement standards, and -benchmarking, with all related facets, augmented by many practical experiences of both of the authors. The book is aimed for beginners as well as experienced colleagues working with software -estimation, -measurement, and -metrics.

Last but not least, themes like productivity measurement, estimation tools, software reuse and redevelopment, and estimation in the maintenance process as well as in Object-Oriented-, Data Warehouse-, or Web- environments are dealt with in this book.

The Books' Content

This book delivers a framework for novices who are getting started in software project estimation, and also offers to the practitioner practical information for transfer into the profession. The text is derived from years of experience by the authors in software development and project management, and supported by a national and international networking in European and worldwide metrics- and standards- organizations.

Chapter 1 provides an entrance portal into the theme and introduces the first concepts. Chapter 2 lays the fundamental concepts, and together with Chapter 3 presents an overview for the reader desiring quick access to the information. The remaining chapters present topics on estimation in more detail, progressing in several steps:

- Estimation prerequisites and implementation, together with methods of estimation
- Estimation of maintenance effort
- Software measurement and metrics fundamentals, and product and process metrics
- Measurement communities and resources for measurement and benchmarking
- The IFPUG Function Point Method and the other four ISO/IEC-acknowledged Functional Size Measurement Methods
- Function point related measurement variants, experiences, counting rules, and case studies

- Measurement and metrics in object-oriented environments and data warehouse environments, and in software reuse and redevelopment
- A chapter about tools and their methods
- An appendix with examples and checklists.

Each chapter ends with a management summary for the reader who wants a quick synopsis or a list of important internet addresses for further reading.

Acknowledgements

An ambitious book translation and actualization project like this one (about 700 pages), which was performed as a hobby besides family, profession, lectureship of the German author at the University of Applied Sciences and his commitments in the German metrics organization DASMA over a 10-year timeframe and the commitments of the American author to her consulting business, international speaking engagements, and ongoing leadership endeavors, undoubtedly produces some loss by friction. The more important are the positive direct and indirect contributors each author would like to acknowledge and thank:

- Manfred's family in Germany (who had to live some time with his conscious absence and the partial occupation of the living room as author)
- Carol's two grown children, Corinne and Alex, who continue to be a source of inspiration and support in her international business
- *Good* managers who backed Manfred during the introduction of the metrics program at the international insurance corporation
- Ambitious colleagues at the same organization who were willing to widen their horizon and did not fear the additional effort to embrace estimation with Manfred during the management of their projects; colleagues who maintain an active network in professional and metrics organizations and who generously shared their own experiences. These include especially Luigi Buglione (GUFPI-ISMA), Ton Dekkers (NESMA), Peter Fagg (UKSMA), Peter Hill (ISBSG), Capers Jones, Roberto Meli (GUFPI-ISMA), Jolijn Onvlee (NESMA), Tony Rollo (UKSMA), Luca Santillo (GUFPI-ISMA), Charles Symons (UKSMA), Frank Vogelezang (NESMA), as well as Professors Alain Abran, Reiner Dumke, and Eberhard Rudolph. A special thanks also to Pekka Forselius, president of ISBSG and senior advisor to the Finnish Software Measurement Association (FiSMA), who provided essential contributions including the FiSMA Functional Size Measurement Method, experiences with ISBSG, and expertise about the practical implementation of estimating and software measurement
- Numerous students who committed their practical term and thesis to the introduction of a metrics program in their organizations. These students managed great effort and complex investigations of large amounts of data, some of them being rewarded with a DASMA students thesis award

- Note that the experiences herein are provided from the personal experiences of both authors. Our belief in the importance of professional estimating and benchmarking is passionate and results in opinions that are sometimes articulated in a very pronounced manner to foster an awareness that proven methods should be professionally used
- One final note: during the writing of this book we have used information and communications technology (ICT), information technology (IT), software intensive systems, and similar terms in an approachable and some-times interchangeable manner. For readers desiring a more formal treatise on the use of these and other industry terms, the reader is referred to the foreword and appendix of the American author's April 2008 book: Program Management Toolkit for software and systems development, co-authored with Pekka Forselius et al. by Talentum (ISBN: 978-952-14-1338-4).

Manfred Bundschuh
Carol Dekkers
Bergisch Gladbach and Tampa
Spring 2008

2 Estimation Fundamentals

This chapter introduces an estimation framework to enable the reader to position estimation in project management, project control, and quality assurance. The reader will also become acquainted with the characteristic parameters of estimation.

Objectives of organizations to survive in the market today can all be derived from quality, productivity, and predictability. Quality pertains to the effectiveness of the processes (doing the right processes) and the product (building the right product). Productivity and predictability both pertain to the efficiency of the processes used to develop the product. *Hence, estimation will be an essential part of project management* and must be regarded in the complete context mentioned earlier. *Project management without estimation* (*often justified because it seems to be too time consuming*) *is like driving a car without planning to refuel along the way.* Typically, project management falls into two main types:

1. Strategic project management
2. Operative project management.

Strategic project management organizes the overall life cycle development of all IT projects of an organization. Synonymously it is called program management or project portfolio management. Conversely, operative project management concentrates on a single project level. The major components of both kinds of project management include the following from the Project Management Institute Project Management Body of Knowledge (PMBOK® Guide):

1. *Project Initiation.*
2. *Project planning* (*including Project Estimation*) *provides the basis for the main tools of project control.* As a project progresses, it tends to deviate from plans. To avoid this entropy and to stay on a goal-oriented direction it is necessary to have a detailed plan.
3. *Project execution.*
4. *Project Control.*
5. *Project Closing provides the basis on which project actual hours and other project lessons should be recorded for historical purposes and use on future*

projects. Note that the PMBOK does not explicitly prescribe the data collection at the end of a project, rather it specifies that the project have a formal end (the closing). The authors advocate the northernSCOPE(TM) concepts (www.fisma.fi/in-english/scopemanagement) that organizational learning (via the collection of project actuals at the close of the project) is an important corporate best practice.

Estimation is the foundation of viability assessment of IT projects. The tools of estimation include e.g., cost benefit analysis, Functional Size Measurement, assessment of non-functional requirements (quality requirements), and a myriad of diverse estimation methods.

The distinction between operative and strategic project management must also be made for its subtasks. Hence, there exists *operative and strategic project control* as well as *operative and strategic estimation.*

2.1 Estimation in a Project Controlling Environment

The traditional tasks associated with project control are as follows:

1. Planning (determination of metrics)
2. Information gathering
3. Control
4. Steering.

Exactly these are the core functions of operative project control within operative project management.

Its task is to deliver to the project management the necessary information about project progress:

- At the right time
- Condensed (in summary form)
- Problems and how they can be adequately addressed.

Hence, it has to perform the following tasks:

1. Definition of the effort targets for IT project subtasks based on sound and professional estimating (planning task)
2. Continuous measurement of actual effort for the subtasks of the IT project (information gathering)
3. Continuous comparison of actual effort versus planned effort during project progress (control)
4. Analysis of causes for eventual deviations and recommendations for the actualization of the project plans (steering).

These tasks belong in the context of the cybernetic control circuit (see also "The Cybernetic Estimation Control Circuit" part of this chapter) of project management. Estimation gets its strategic or long-term character through the capability to provide experiences of the past for the improvement of future estimations. This is part of organizational development whereby lessons of the past are used to master the future (*feed forward*). In strategic estimation, this is accomplished by documentation of the estimate and analysis of this documentation for the development of IT metrics and benchmarking. Estimation is the sound foundation of planning and thus also the foundation of project control.

In reference to the many surveys that showed evidence that only a marginal number of IT projects that were started were actually finished on time and within budget one has to conclude: "*Anyone who does not perform the project management task of estimation could be considered as acting grossly negligent!* "

The same premise holds for the project management task of *documentation* (see also the chapter "The Estimation Challenges, Documentation").

In particular, the measurement of project size as a basis for estimation additionally delivers the benefit of providing an objective requirements review for the IT project.

Documentation (also of estimates) is important to be able to quantify and understand the system to be developed. Only with this prerequisite is it possible to extract basic experiences that can be integrated into the project management manual. This is an important prerequisite for organizational learning. If the functional size measurement fails because documentation is not available (i.e., either not existing, not actual, or indecipherable) or there is a lack of know-how on the IT project, then it can be concluded that the requirements analysis is not yet complete. *Alternatively, it is an important early warning sign that shows that the IT project has lost its bearing so early in its lifecycle.*

2.1.1 Adjusting the Estimate to Take into Account Project Environment Factors

A number of factors from the environment of IT projects have an enormous influence on the actual effort and hence must be considered by the project leader as input to the estimate. These factors must be taken into account at the level of project tasks where they can be used to adjust and enable the development of sound estimates. Some of these factors are as follows:

- The development environment and platform such as PC, mainframe, Client/ Server, Expert System,...)
- The development language (Assembler, Cobol, C++, Program Generator, Java, SQL,…)

- The run-time environment (DB2, CICS, IMS, Data Warehouse, Internet, Intranet,…)
- The project classification (new development, enhancement, maintenance, strategic IT project,…)
- The project class (large system, interactive database application, standard software, system software, query system, cash system, online/batch proportions of applications,…)
- Complexity of the IT project (data-, code-, and functional complexity, number and type of interfaces,…)
- Regulations for quality and security standards (four eye principle, test concept, software engineering process model,…)
- Restrictions by law, technique, or organization
- Project novelty (First use of new methods, processes, tools, software, languages, platforms, …)
- Support of the IT project by managers, users, union, …
- Large number of interfaces or new customers (literature: +25%)
- Project duration (literature: more than 6 months +15%, more than 12 months + 30%, more than 18 months + 50%)
- Clarity of responsibilities in the IT project
- Open-plan office (literature: +25% to +30%)
- Experience of the project leader in estimation
- Skill of project team (experts, beginners, mix)
- Team size (in each project phase)
- Availability and time restriction of people, especially of crucial experts
- Business/industry type (military, banking, avionics, government, …).

Table 2.1. Factors influencing software engineering estimation

Technology	Product	Development process	Resources
Technical development platform	Functionality	Process organization	Hardware availability
Hardware (and software)	Quality	Software engineering process model	Software availability
Software	Complexity	Methods	Staff availability
Technical standards	Documentation	Project duration	Staff quality
Tools	Restrictions by law	Interfaces	Costs (budget)
Technical requirements	Project classification	Goals	Organizational restrictions
Technical run-time environment	Project class	Organizational development environment	Project calendar

Table 2.1 shows a structured overview of some of such influential factors but cannot compete with the nearly 100–200 such parameters administered in commercially available estimation tools. Only the use of such tools guarantees that

the estimator does not lose the overview when regarding a larger number of parameters for estimation.

Several estimation methods consider some of these factors of influence. The Function Point Method, e.g., uses 14 General System Characteristics (GSC); COCOMO II uses 22 factors, and FiSMA ND21 uses 21 factors for new product development. *Of these factors, the project objectives (goals for quality, scope, schedule, cost) have the most influence on project effort as well as on project success.*

2.1.2 Project Goals and the Devils Square of Project Management

Generally an IT project is characterized by unique conditions requiring special organizational measures (project management, management of crises, risk management) caused by its complexity. It has normally the following characteristics:

- There exists a clearly formulated and reachable goal.
- There exist time, financial, personnel, and/or other constraints as well as a high degree of innovation.
- The project has a clear demarcation to other tasks and projects and has a start date as well as a delivery deadline.

An IT project is a temporary set of activities with the goal to develop and install a software system. *The objectives of an IT project must be absolute and clearly defined, and the achievement of its targets must be measurable.* This is the main success criteria of an IT project. The goals can be differentiated into primary and secondary goals. *Primary goals* are as follows:

1. Quality
2. Size (Quantity)
3. Duration (Time)
4. Costs.

Possible *secondary goals* may be the following:

- A 25% staff reduction in the order management department
- Reduction of the maximum handling time of a customer claim to 24 h.

The primary goals unavoidably compete with each other for the resources of an IT project. Hence, every additional consumption of one resource leads to reduction in the availability of other resources. This effect is known as *the devils square of project management* (see Fig. 2.1). The example in Fig. 2.1 shows how size is reduced in order to gain more quality and reduce costs.

The devils square also highlights that estimation is the basis for a sound planning of quality, functional size, costs, and dates. How can you plan when you

do not know the necessary effort? The problem of the project leaders in this context is that management expects them always to minimize costs and time while maximizing size and quality – an impossible task!

2.1.3 Estimation and Quality

The quality of a software product is measured by the degree to which it meets or exceeds the user requirements. The measurement of the functional size for estimation thus becomes of extraordinary significance.

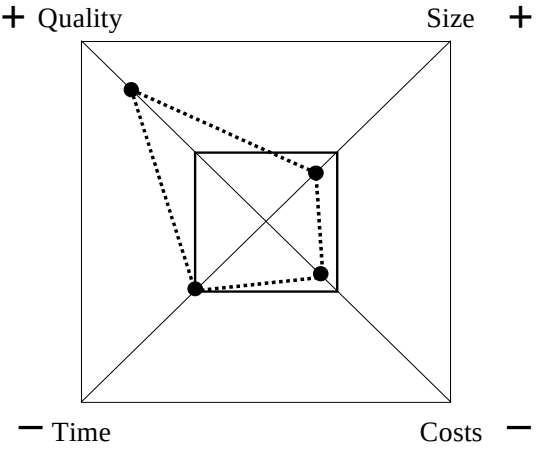

Fig. 2.1. The devils square of project management

The increasing acceptance of IT even in private life leads to increasing demands of high-quality software. This increased *quality consciousness* makes quality one of the most important goals of software development. The PMI (Project Management Institute) identifies quality as the center of the triple constraints triangle consisting of the following as the governing project constraints:

1. Scope (functionality)
2. Cost (budget)
3. Time (duration).

If one of the triple constraints or the quality requirements changes, the other constraints are affected. This directly affects project estimating because software development projects are always limited by budget (cost), time to market (duration), quality, and/or scope (functionality).

For example, once an estimate is made based on a given project scope, quality, budget, and duration, if the scope is increased – then it will affect the other components. Sometimes this is referred to as project tradeoffs because if the

project scope changes and there is limited time and cost allocated for the project, the product quality will suffer. Similarly if the quality demands for a project increase after a project estimate is made, then the functionality (scope) must decrease in order to finish the project within the same timeline and cost structure.

There are a number of relevant and proven measures, methods, and techniques for software and software development quality improvement.

Today, quality is no coincidence, but rather it can and must be planned exactly into a product. Today, good quality is built into a product rather than poor quality detected out.

Quality management in IT projects consists of the following tasks:

- Quality planning
- Quality execution
- Quality control (measurement and tracking)
- Quality assurance.

The first two tasks are performed systematically by so-called *constructive quality assurance measures*, which secure quality a priori. Constructive quality assurance measures include the systematic use of methods, development tools, or standardized processes. Quality control is performed by *analytical quality assurance measures* in order to measure adherence to quality requirements or deviations thereof, and if necessary, to correct any gaps or detected defects.

The focus of these tasks centers on constructive quality assurance measures since *prevention is better than defect correction*, or, using a metaphor*: fire prevention is better than fire fighting.*

This premise is accompanied by the requirement to define quality goals for the software development process, which in turn must meet and exceed the quality goals of the software to be developed. Quality attainment is then measured by comparison of the goals for product quality and the actual quality features of the developed software. In IT projects, as part of the requirements, the quality attributes are defined at the start of the IT project, and become part of the input variable set to the estimation equation. This is a direct link to estimation. The ISO/IEC 9126 External Quality Attributes (see Fig. 2.2) identify the major aspects of product quality for the software to be developed, and each major area such as functionality is further subdivided in the ISO/IEC standard into individual quality characteristics.

2.1.4 ISO/IEC 9126 Quality Attributes and IFPUG GSC

The ISO/IEC 9126 Quality Attributes partially overlap with the 14 GSC of the IFPUG Function Point Method, which are used to adjust/modify the Functional

Size Measurement of the software to arrive at the adjusted Function Points for use in estimating. It is therefore obvious that an automatic interface should be created to avoid double work for the project leaders. A large organization developed the following Excel chart, which automatically calculates the quality

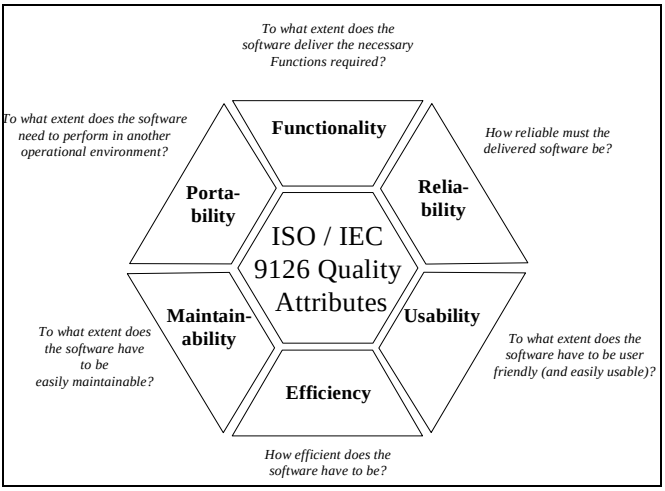

Fig. 2.2. The ISO/IEC 9126 quality attributes

Value: High 5 3 / Average 4 - 3 / 2 / Low 2 - 1 / 1 / None 0 / 0 IFPUG Q-		Quality Characteristic											
Ratings High 9 - 7 / Average 6 - 4 / Low 3 - 1 / None 0		Adaptability	Usability	Efficiency	Functionality	Maintainability	Correctness	Portability	Stability	Security	Interoperability	Reusability	Reliability
IFPUG GSC's	Value	3	2	2	3	2	2	3	2	3	2	1	2
Data communication	5	1			3			1				1	
Distributed Data Processing	4	2									9		
Performance	4			3		2							
Heavily Used Configuration	5			3						1			
Transaction Rate	0			3		2			1				2
Online Data Entry	5	1	1		3				2				2
End-User Efficiency	4		6										
Online Update	5					5				4			
Complex Processing	4				3		3			1			3
Reusability	2											7	
Installation Ease	4		1				4			2			2
Operational Ease	4						2		6	1			
Multiple Sites	5							8					
Facilitate Change	5	5	1									1	
relative meaning		10,0%	8,9%	6,3%	9,8%	7,7%	8,4%	10,5%	7,9%	9,6%	8,4%	5,6%	7,0%
absolute meaning		43	38	27	42	33	36	45	34	41	36	24	30

abs. rating	Value
45	3
41	
40	2
27	
26	1
6	
5	0
0	

Fig. 2.3. Mapping of the ISO/IEC quality attributes and IFPUG GSC

Table 2.2. Evaluation of IFPUG GSC and ISO/IEC quality attributes

General system characteristics	Mapped to the priority of the quality attribute
0	= No priority (0)
1 and 2	= Small priority (1)
3	= Medium priority (2)
4 and 5	= High priority (3)

attributes from the GSC and vice versa. The connection between the quality attributes and the GSC was ranked from 1 to 9 by the project team, where the sum of each column is 9. Thus, in Fig. 2.3 the quality attribute *Adaptability* (a quality characteristic in ISO/IEC 9126) is connected with the following IFPUG GSC (see column 1 in Fig. 2.3 and Table 2.2 for the mapping of the values):

- 1/9 with data communication
- 2/9 with distributed data processing
- 1/9 with online data entry
- 5/9 with facilitation of change.

The ISBSG (International Software Benchmarking Standards Group) book titled *Practical Project Estimation, 2nd edition*, identifies two alternative methods of addressing these non-functional or quality requirements for software. The first method identified is the COCOMO II set of factors, and the second is the Finnish Software Measurement Association (FiSMA) situation analysis called New Development 21 (ND21) factors (see www. fisma.fi for details).

A second determination factor, besides the classification of estimation into project controlling, is the consideration of its cybernetic control circuit features.

2.1.5 The Cybernetic Estimation Control Circuit

Estimation can be thought of as a cybernetic control circuit. This is an important feature *since control circuits are directable systems that can be controlled by feedback* that enables them to compensate disturbances influencing them. They are able to proceed in a state of equilibrium (called homeostasis) if there are no disturbances or influences exerted on them from the environment. *With the principal model of the cybernetic control circuit the behavior of complex systems can be understood, explained, and controlled.* For better understanding of the cybernetic control circuit of estimation the concept will be explained in more detail here.

Norbert Wiener coined the term *cybernetics* from the Greek word meaning *steersman*. He defined cybernetics as the science of communication and control in mechanisms, organisms, and society. Cybernetics is a general theory of control, a science of the behavior of adaptive complex systems having the important features of feedback and communication as well as information exchange.

A cybernetic control circuit consists of the following four components:

1. *Controller*: The Controller gets information about measures collected by the measurement component, produces decisions, and delivers objectives to the adjustment component. → In the special case of estimation, the controller delivers an estimate to the adjustment component for reaching this objective.
2. *Adjustment Component* (*Actuator*): The adjustment component accepts input from the controller, chooses measures for the mode of activity, and delivers these adjustment factors as (for the model understandable) signals to the object of control to cause changes in it. → In the special case of estimation, the actuator compares this objective with the knowledge base (historical data) and delivers an improved objective to the object of control.
3. *Object of Control* (*Model*): This is the regulating extension, the model that performs the given measures. It is the component where the cybernetic circuit can de disturbed by factors of influence from the environment. *The shorter is this regulating extension* (*e.g.*, *time distance*: *early warning signals*), *the more modest are the measures for steering of the system.* → In the special case of estimation, the object of control sends notifications and data to the measurement component.
4. *Measurement Component*: The measurement component measures the degree of fulfillment of the objectives and accepts notifications telling it that the state of the model has changed. Data are retrieved from the model and used as feedback passed to the controller to further drive the model. → In the special case of estimation, the measurement component measures the actual state of the model, compares it with the objectives, and informs the controller about the deviations. The controller elaborates from this a new estimation and the circulation starts anew.

The whole process is called *feedback loop* and leads to a flexible balance (*homeostasis*), i.e., the system regulates itself when there are no disturbances affecting it. The user (not necessarily human) is not considered to be a component of the cybernetic control circuit but is part of the controller and constitutes the decision-making function that dynamically directs state changes in the model. Figure 2.4 visualizes the cybernetic control circuit of estimation.

The project tasks together with the objectives, the classification, type and class of the project, and project size are input for the controller where the objectives are defined. Furthermore, the controller produces decisions (output, initial value) – based on the comparison of actual versus planned measures from the measurement component – which are delivered to the actuator for comparison with the knowledge base. The actuator chooses a measure (estimated value) and delivers it to the model. This is the object of control and produces – with influences of outside disturbances from the environment – an actual value. This actual value is sent to the measurement component for measurement of the fulfillment

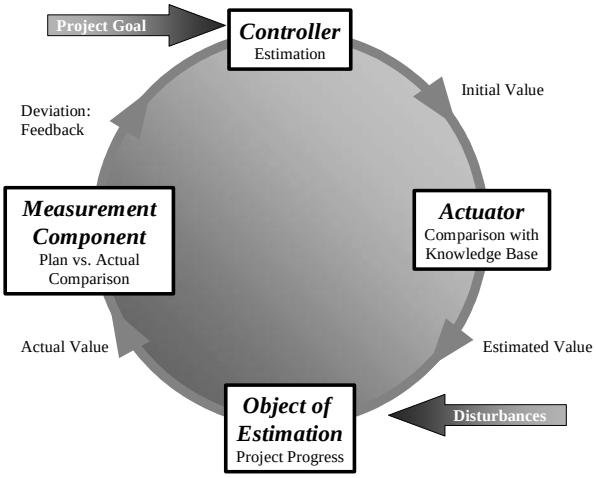

Fig. 2.4. The cybernetic control circuit of estimation

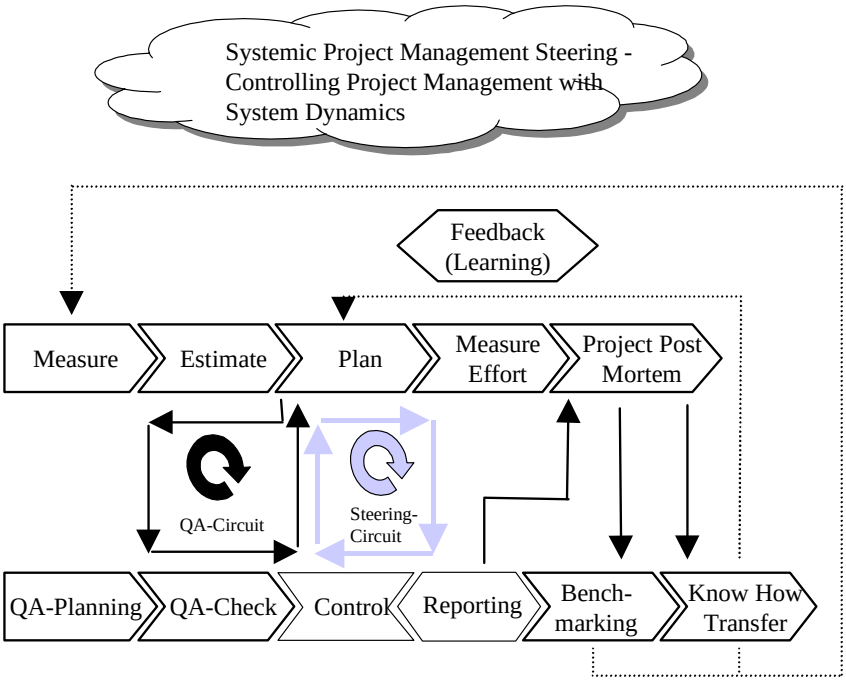

Fig. 2.5. Cybernetic control circuit for estimation

of the objectives. The comparison of planned versus actual values produces a deviation, which is sent as information to the controller.

Severe disturbances can occur, e.g., if the knowledge base does not exist or is qualitatively inadequate, when there are no measurements available (either not done or not documented), or if the estimation process and/or measurements are not controllable. In any of these cases, the cybernetic control circuit is interrupted or it performs in a cumbersome manner. In such cases, quality and the overall benefits of estimation are reduced.

Figure 2.5 shows the tool-based systemic project management concept (realized in an organization) with the partial process of project management and the imbedded cybernetic control circuits for quality assurance and project steering as well as the feedback loops for organizational learning.

2.2 Determining Parameters of Estimation

Strategic estimation is part of strategic project management. Hence, the goals to be reached with estimation should be defined as a necessary prerequisite before introducing estimation.

As an example, strategic project management can have the following goals:

- Continual improvement of the following:
  - Estimation
  - Project planning
  - Project elaboration
- Identification of the following:
  - Cost drivers
  - Efficient methods and tools
- Internal as well as external benchmarking

From these, the following goals for *strategic estimation* can be derived:

- Continual Improvement of the following:
  - Measures of product size
  - Measures for parameters influencing project effort
  - Methods and standards for planning and elaboration of estimation
- Identification of the following:
  - Parameters influencing project effort
  - Efficient methods, standards, and tools.

Figure 2.6 summarizes the determining parameters of estimation, the drivers, constraints, as well as the degrees of freedom. A connection with the devils square of project management can obviously not be neglected.

2.2.1 The Purpose of Estimation

The success of metrics implementation relies on how an organization assesses the principal question: *"What* (*which IT metrics*) *shall we measure?"* After sizing the product (functional size measurement), the effort to be expended shall be estimated in person months (or hours) using the size and additional estimation parameters. Next, the estimated effort is distributed across the phases of the project as the basis for project planning and scheduling. *For the total project plan, an estimate must also be made for the effort for project management and quality assurance. Often the project management and quality assurance efforts are overlooked or forgotten, and this leads to severe miscalculations and underestimating.*

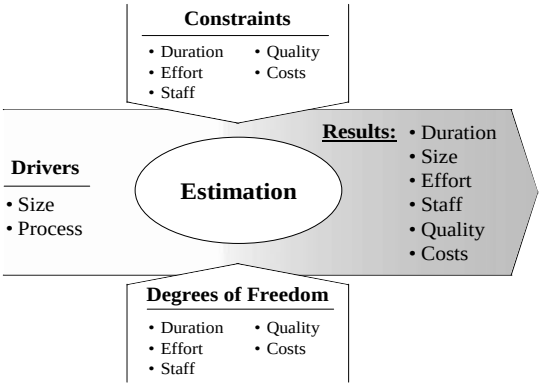

Fig. 2.6. The determining parameters of estimation

Last but not least, estimation contributes to making process improvements measurable. Process capability can be an abstract measure for many factors that influence process improvement. Process capability is a measure of the efficiency of the software development and is measured in effort hours per Function Point, also called PDR (Project Delivery Rate). Putnam and Myers estimate the annual process improvements of organizations with process improvement programs to about 13%. Other authors, including the Software Engineering Institute (SEI), estimate a time span of 2.5–4 years for doubling the process capability of an organization.

The SEI developed a software and systems process maturity model to evaluate the capability of an organization to build software and systems products. Originally, this model was called the Capability Maturity Model (CMM®) for software but today various maturity models for systems, acquisition, and other competencies were combined into what is now known as the Capability Maturity Model Integration or CMMI®. The CMMI® identifies five progressively

mature levels of process capability or maturity for an organization, and the average time to ascend from one step or level to the next is ~18 months.

ISO/IEC developed a process improvement framework called SPICE: Software Process Improvement Capability Determination, which is now represented by a series of standards under the umbrella ISO/IEC 15504. CMMI® and SPICE are both examples of process maturity models.

2.2.2 The Goals of Estimation

The following goals can be reached using estimation methods:

- Holistic and integrated estimation process(es) for IT projects
- Organizational learning (measurement and estimation can highlight best practices that can be leveraged on future projects)
- Concept for training of estimators
- Tool support for host and PC environment
- Standardized estimation process
- Detailed estimation manual
- Documentation manual
- Foundation for benchmarking
- Transfer of experiences with estimation
- Reduction of complexity and uncertainty of estimations
- Increased reliability, precision, and accuracy of project estimates
- Improved requirements documentation and completeness (because functional size measurement relies on good requirements, organizations that implement function points often find that their requirements processes necessarily improve to facilitate the size measurement)
- Improvement of estimation exactness.

It does not matter if the IT projects are classified as new development, enhancement, or maintenance – the objectives that can be achieved are the same. Problems may arise when there are goal conflicts, since the estimators tend to think of estimation as being data-centric, whereas the managers are more likely resource-oriented, and the end users or project sponsors are more likely risk-oriented. *Acceptable estimations must consider and address all three perspectives* (*data-centric, resource-oriented, and risk-oriented*).

2.2.3 The Right Time for Estimation

Determining the right time for estimation is an important consideration for any organization interested in implementing formal estimating procedures. The timing parameter requires a lot of attention since it is the subject of an inherent goal conflict:

Early and precise estimations are necessary and desirable by software customers; however, early estimations are necessarily imprecise and prone to a high degree of uncertainty.

This problem is aggravated by the following effect:
Estimation is done too early and far too seldom!

If one accepts that the precision of estimations at the beginning of an IT project is imperfect and insufficient, and it only increases as the project progresses, then the logical consequence is that multiple estimates are necessary throughout the project. Estimates must be updated and revised whenever important influencing factors change.

In practice, an estimate is usually only done at the project start, and sometimes at a project postmortem. Capers Jones stresses the impact of requirements (scope) creep as causing a 1–3% functional size increase *per month* of project duration.

Additional factors where estimate revisions are necessary include, for example, illness of key staff or resource reallocation. *If such changes are not considered and actualized, the plan made from the original estimation will never be met.* It is critical to revise and repeat the estimation process during project progress especially when there is substantial scope creep or deviations from the project plan. To increase the chance of consensus about the future of the project, the customer should always be kept informed about changes to the estimates (because they reflect changed plans).

Figure 2.7 provides an overview of possible estimation milestones, where milestones 1–7 have the following meaning:

- *Milestone 1: End of Feasibility Study Phase*
  The idea or concept for a new project is constituted. There exists only little information about requirements details and thus Function Points can only be approximated (see chapter “Function Point Prognosis”). Effort estimates can be developed using a tool together with relevant historical data from comparable completed projects. In many companies, the project charter and a preliminary effort estimate are delivered at milestone 1.
- *Milestone 2: Project Start*
  At this point, further information about the project, its resource requirements, and the possible timeframes exists. Furthermore, the IT project team, the development environment, and the programming language are typically known. Hence, a more detailed estimate derived using an estimating tool is possible.
  There still is not enough information available for a complete Function Point count. Hence, the first Function Point Prognosis should be actualized, and estimation with a tool should be done using the documented assumptions for the project.

- *Milestone 3: End of Requirements Analysis*
- At milestone 3, there is now sufficient information for a complete Function Point count followed by documentation and estimation with an estimation tool. The GSCs are classified in an estimation conference as previously described.
- The actual data measured to date on the project become input for the estimation tool, and a revised/updated estimate is carried out on this basis. For tracking and organizational learning reasons, this estimate must be compared with the first estimate.
- *Milestones 4–6: End of IT Design until End of Project*
- Counting and Estimation are actualized at least at critical project dates and confirmed on phase transitions. Changes in the IT project become transparent and part of the process, and are documented to capture the data of the experience. Estimates are tracked continually. The actual measured effort is documented in an estimation tool at least at the end of each phase or preferably on a regular (weekly) basis.
- *Milestone 7: Project Postmortem*
- Here the main task is to collect information and experiences at project completion to improve the counting and estimation processes for subsequent projects. (One of the best ways to capture this data is to conduct a workshop about the experiences in this project.)
- In project postmortems the following effort components are frequently neglected: unpaid overtime, effort for project management, effort for quality assurance, effort for administrative tasks (all effort of the IT project) as well as effort of end users and technical specialists not belonging to the core team. To improve and learn from your own completed projects, it is essential to have a record of all expended project effort so that future projects can gain from the knowledge of complete project data.

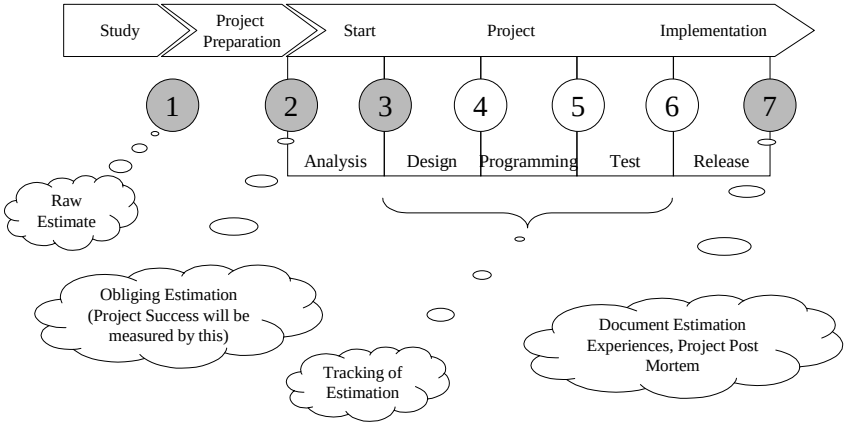

Fig. 2.7. Milestones for estimations in IT projects

The project postmortem must elaborate the realistic amounts for efforts occurring on a one-time basis such as migrations (data conversions) or test drivers (documented separately), as well as effort supplied externally (documented separately). Preferably, this task will be carried out with the assistance of the competence center. It will actualize the metrics for the experience curve as well as IT metrics standards. A competence center will take this data and subsequently use it to determine the productivity measures and other IT metrics. From the viewpoint of estimation the reader is directed to the appendix where we have included a useful checklist for project postmortem of IT projects.

2.2.4 Tracking of Estimates

In its estimation tracking manual, the Software Engineering Laboratory (SEL) of the NASA asks its managers to perform estimates at least six times during the project duration. Each time the estimate for the remaining project work is multiplied with different lower and upper control limits in order to give an interval for the estimate uncertainty at the particular milestone. Table 2.3 shows these multiplication factors.

Table 2.3. Add-ons of the SEL (NASA) for estimate uncertainty

Milestone	Upper control limit	Lower control limit
End of rough requirements concept	×2	×0.5
End of detailed requirements concept	×1.75	×0.57
End of rough IT design	×1.4	×0.71
End of detailed IT design	×1.25	×0.8
End of programming	×1.1	×0.91
End of testing	×1.05	×0.95

For the continuous tracking of estimates, it is advisable to set up a catalogue of continuous activities such as the following:

1. An annual index of applications, projects, and base values to be measured.
2. This registry must only contain objectively measurable data, which must be measured when they occur. In addition to measured values, estimated values must also be recorded. All data must be documented at different aggregation levels to enable later drill down queries into project details. Basic data (esti-mated, planned, and actual) are, e.g., start date, end date, size (in Function Points or SLOC), number of defects, effort by phase, effort of end users, IT and support.
3. For each of these items, a baseline has to be calculated.
4. A comparison of the baseline with the preceding year(s) in order to recognize changes and tendencies.

The following checklist comprises the most important milestones for estimation:

- End of feasibility study
  - Rough estimate based on already known information
  - Depending on estimated project size, add in ~10–30% for each of the following project add-ons: risk, uncertainty, and requirements creep
- Start of project
  - Detailed estimates should be checked by a second estimating professional (e.g., expert estimation). The result becomes the basis for later measurements of the success of the project.
- End of Requirements Analysis
  - Function Point count, project internal estimation conference
- End of each project phase
  - Actualization of the Function Point count due to the requirements creep
- Project postmortem
  - Measurement of success of the IT project
  - Actualization of IT metrics data and repository
  - Workshop for know-how transfer
- Annual baseline and time series.

Project postmortem should be carried out in a meeting documenting all important information about the project, including measures leading to project success as well as those not so successful. *It must be absolutely avoided to search for culprits and attribute blame. Project postmortems are an important prerequisite to foster learning for the future* (*feed forward*). For this reason, the project postmortem information should be readily available for electronic access.

2.3 Management Summary

Estimation is an essential part of project management and must be regarded in the complete context mentioned earlier. Project management without estimation (often justified because it seems to be too time consuming) is like driving a car without planning to refuel along the way.

Estimation is the foundation of viability assessment of IT projects.

Estimation gets its strategic or long-term character through the capability to provide experiences of the past for the improvement of future estimations. This is part of organizational development whereby lessons of the past are used to master the future (feed forward).

Estimation is the sound foundation of planning and thus also the foundation of project control.

Anyone who does not perform the project management task of estimation could be considered as acting grossly negligent!

In particular, the measurement of project size as a basis for estimation additionally delivers the benefit of providing an objective requirements review for the IT project.

The objectives of an IT project must be absolute and clearly defined, and the achievement of its targets must be measurable. This is the main success criteria of an IT project.

The primary goals unavoidably compete with each other for the resources of an IT project. Hence, every additional consumption of one resource leads to reduction in the availability of other resources. This effect is known as the devils square of project management.

The quality of a software product is measured by the degree to which it meets or exceeds the user requirements. The measurement of the functional size for estimation thus becomes of extraordinary significance.

Today, quality is no coincidence, but rather it can and must be planned exactly into a product. Today good quality is built into a product rather than poor quality detected out.

Estimation can be thought of as a cybernetic control circuit. This is an important feature since control circuits are directable systems that can be controlled by feedback that enables them to compensate disturbances influencing them. They are able to proceed in a state of equilibrium (called homeostasis) if there are no disturbances or influences exerted on them from the environment. With the principal model of the cybernetic control circuit, the behavior of complex systems can be understood, explained, and controlled.

The whole process is called feedback loop and leads to a flexible balance (homeostasis), i.e., the system regulates itself when there are no disturbances affecting it.

The success of metrics implementation relies on how an organization assesses the principal question: *What (which IT metrics) shall we measure?*

For the total project plan, an estimate must also be made for the effort for project management, and quality assurance. Often the project management and quality assurance effort is overlooked or forgotten, and this leads to severe miscalculations and underestimating.

Early and precise estimations are necessary and desirable by software customers; however, early estimations are necessarily imprecise and prone to a high degree of uncertainty.

Estimation is done too early and far too seldom!

Capers Jones stresses the impact of requirements (scope) creep as causing a 1–3% functional size increase per month of project duration.

The project postmortem must elaborate the realistic amounts for efforts occurring on a one-time basis such as migrations (data conversions or test drivers documented separately), as well as effort supplied externally (documented separately). Preferably, this task will be carried out with the assistance of the competence center.

In its estimation tracking manual, the SEL of the NASA asks its managers to perform estimates at least six times during the project duration.

For the continuous tracking of estimates, it is advisable to set up a catalogue of continuous activities.

It must be absolutely avoided to search for culprits and attribute blame. Project postmortems are an important prerequisite to foster learning for the future (feed forward).