Monograph: Experiences and Advances in Software Quality
(published jointly with Novática*)
Guest Editors: Darren Dalcher and Luis Fernández-Sanz

2 Presentation: Improving Quality in Business Processes, Products and Organizational Systems — Darren Dalcher and Luis Fernández-Sanz
6 Preventative Software Quality Control: Using Human Checking to Change Defective Human Practice — Tom Gilb and Lindsey Brodie
14 The Software Process Improvement Hype Cycle — Miklós Biró
21 Quality Going for Gold — Derek Irving and Margaret Ross
26 Can Teamwork Management Help in Software Quality and Process Improvement? — Esperança Amengual-Alcover and Antònia Mas-Picacho
34 Evidence-based Software Engineering and Systematic Literature Reviews — Barbara Kitchenham, David Budgen, and O. Pearl Brereton
42 Software Project Success: Moving Beyond Failure — Darren Dalcher
51 Software Measurement for Better Project and Process Quality — Christof Ebert
67 A Quality Evaluation Model for Web2.0 e-Learning Systems — Stephanos Mavromoustakos and Katerina Papanikolaou

UPNET (UPGRADE European NETwork)

75 From Mondo Digitale (AICA, Italy)
History of Computing
The Turing Test: History and Significance — Giuseppe O. Longo

CEPIS NEWS

89 Selected CEPIS News — Fiona Fanning

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by Novática, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática) at <http://www.ati.es/novatica>.
Software Measurement for Better Project and Process Quality

Christof Ebert

Software increasingly governs our world and our society. Since software is so ubiquitous and embedded in nearly everything we do, we need to stay in control. We have to make sure that systems and their software run as we intend - or better. Software measurement is the discipline that ensures that we stay in control. Software measurement applies to products (e.g., performance engineering), processes (e.g., productivity improvement), projects (e.g., estimation) and people (e.g., engineering skills). This article will introduce to software measurement in the context of achieving better quality for projects and processes.

Keywords: Empirical Software Engineering, Process Improvement, Project Management, Quantitative Management, Software Measurement.

1 Introduction

Human performance improvement is essentially unlimited. We are driven to try to go beyond what is thought to be our limits. The fastest time for the mile was 4 minutes 30 seconds in 1865, 4 minutes in 1954, and is around 3 minutes today (depending on when you read this introduction). Perhaps we can expect a three-minute mile during this century? Can you imagine how little runners would have improved if there were no stopwatch or measured track?

Unmeasured and unchallenged performance does not improve! Moreover, it will not improve if not fostered by best practices in the discipline. Software development is a human activity and is prone to continuous performance improvements. Software measurement is the approach to control and manage the software process and to track and improve its performance. You can only control what you observe and measure. The act of setting objectives and monitoring them is the way to guarantee receiving the expected results.

Software measurement is however not sufficiently used to keep projects on track and to improve organizations’ performance. Whether it is software engineering for a new embedded automation product, the development of a software application or the introduction of a managed IT-service, demand outstrips the resource capacity of an organization. As a result we see two things: First the acceptance of impossible constraints in time, scope and/or cost. And second, as a direct consequence of the first, we see an increase in rework and chaos, as well as budget overruns, delays and canceled projects. We will show here how software measurements contribute to project success. For more details we refer to the book "Software Measurement" [1].

2 Foundations of Software Measurement

The way software measurement is used in a software company determines how much business value that organization actually realizes. You can hardly imagine any software company operating at its maximum performance without knowing where it is and where it should go. It would be like a fast car in fog that needs to slow down below its capability due to not knowing where it is. Software measurements are used in the following ways:

1. Understand and communicate: Measurements help us understand more about software work products or underlying processes. It also allows us to evaluate a specific situation or (statistical) characteristics of software artifacts in order to make operational decisions leading to special experiences (e.g., project management, rules of thumb, assessments and analysis of situations).

2. Specify and achieve objectives. Measurements are key in identifying and specifying objectives. They ensure we stay on course and eventually reach those objectives. They are used to estimate, and/or forecast software characteristics in order to achieve and improve our understanding leading to improved general knowledge and guidelines (e.g., estimation formulas, development standards and general project profiles).

3. Identify and resolve problems. Measurements help us evaluate work products or processes against established standards and to define and measure specific attributes during the software life-cycle. The latter enable improvements in quality and/or performance (standardized size and complexity measurements). They help to identify, analyze and mitigate risks in order to optimize return on investment (ROI) in an organization’s business.

4. Decision-making and improvement. Measurements allow the monitoring, evaluation, analysis, and control of the business, project or, product or they also permit the con-
trol of process attributes for operational and strategic decisions leading to continuous improvement. This includes portfolio management, removal of the root causes of problems and process improvement.

Today, it is impossible to have a business process without measurements and periodic control. For some companies this is a legal obligation, such as provided by the Sarbanes-Oxley regulations. For all companies it is a simple question of audit and financial control. The measurement process is an inherent part of almost any business process (see Figure 1). Software engineering, as part of the product development business process, is therefore also in need of control and measurement. Software measurements include performance measurements, project control and process efficiency and effectiveness.

Measurements are management tools. Measurements must thus be governed by goals, such as reducing the number of project risks or improving test efficiency, as otherwise they simply generate data cemeteries. Figure 1 shows that independent of the process and its granularity, measurements are first established in terms of goals and measurement processes. They are then extracted in operational activities, where these measurements are evaluated for instance on whether the overarching goals will be reached, and definitive actions have to be executed to ensure that deviations from goals will be removed. This goal-oriented measurement process is called the E4–measurement process.

The E4–measurement process consists of four essential steps:

- **Establish** concrete objectives and the measurement and analysis scope and activities
- **Extract** measurements for the established objectives
- **Evaluate** this information in view of a specific background of actual status and goals
- **Execute** (carry out) a decision to reduce the differences between actual status and goals

The measurement process is governed by ISO 15939 (Software engineering – Software measurement process) [3]. By following this standard we can plan and implement a measurement process based upon best practices. ISO 15939 consists of two parts. First we need to establish the measurement program and prepare it within the project or organization. Then we execute measurements in each single project covering people and processes. We extract data, evaluate it and execute corrective actions – depending on the outcome of analyzing our measurements. These two parts, namely preparing and executing the measurement program, should be considered in all subsequent examples which we provide in this article. It is not enough to simply collect numbers and report them. There is a need for a full framework into which measurements can be embedded as a tool to support decision-making and to further drive the implementation of decisions.

To describe the use of measurement, we will introduce a simplified product life-cycle as it applies to software, systems and IT projects. The top portion of Figure 2 shows this simplified product life-cycle. It consists of archetypal phases as you find them in practically all product and service developments, be it software applications, internet software, middleware, enterprise and IT infrastructure, embedded systems, firmware or services. For each life-cycle phase the corresponding text box shows measurements which are relevant for making the project and product a success.

It is useless to extract information and then only record it for potential further use. If measurements are not used immediately, if they are not analyzed and evaluated, then the chances are high that the underlying data is invalid. If there is no pressure to have accurate measurements available, collection is done with little or no care. Where there is no direct use for the information, the information is worthless and the effort to collect it wasted.

### 3 Project Measurement

Do you have sufficient insight into your engineering and IT projects? If you are like the majority of those in IT and software companies, you only know the financial figures. Too many projects are often run in parallel, without concrete and quantitative objectives and without tracking where they are with respect to expectations. Project proposals are evaluated in isolation from ongoing activities. Projects are started or stopped based on local criteria, not by considering the global trade-offs across all projects and opportunities. Only one third of all software engineering companies systematically utilize techniques to measure and control their product releases and development projects [4].

Project management is the major application of software measurement. Software measurement is an absolutely necessary precondition for project success. The measurement goal is to master the project and to finish whilst meeting or exceeding commitments. We need a way to determine if a project is on track or not. There is a saying that "you cannot control what you cannot measure". Because there is little or no visibility of the status of and plan for projects, it is apparent that some common baseline measurements need to be implemented for all projects in an organization. Such core measurements provide visibility of how engineering projects status is performing against the plan, and allow the early detection of divergence and timely corrective action. Measurements reduce the likelihood and number of "surprises" by giving us insight into when a project is heading towards trouble, instead of discovering it when it is already there. Standardized measurements provide management with indicators to assist in controlling projects and the evaluation of performance in the bigger picture.

Project control answers a few simple questions derived from the following management activities:

- **Decision-making.** What should I do?
- **Focusing Attention.** What should I look at?
- **Performance evaluation.** How well am I doing?
- **Improvement tracking.** Am I doing better or worse than the last period?
- **Target setting.** What can we realistically achieve in a given period?
Experiences and Advances in Software Quality

Figure 1: The E4-Measurement Process with its Four Steps.

Figure 2: During each Phase of the Product Life Cycle, Dedicated Measurements are Extracted that Correspond to Established Objectives.
Planning. What is the best way to achieve our targets?

An initial set of internal project indicators for this goal can be derived from the Software Engineering Institute’s (SEI) core measurements [5] and measurement literature [1]. They simplify the selection by narrowing the focus onto project tracking and oversight from a contractor and program management perspective. Obviously, additional indicators must be agreed upon to evaluate external constraints and integrate with market data.

Here is our short-list of "must have" project measurements:

1. Requirements status and volatility. Requirements status is a basic ingredient to tracking progress based on externally perceived value. Always remember that you are paid for implementing requirements, not for generating code.

2. Product size and complexity. Size can be measured as functional size in Function Points, or code size in lines of code or statements. Be prepared to distinguish according to your measurement goals with code size between what is new and what is reused or automatically generated code.

3. Effort. This is a basic monitoring parameter to assure you stay in budget. Effort is estimated upfront for the project and its activities. Afterwards these effort elements are tracked.

4. Schedule and time. The next basic monitoring measurement to ensure you can keep the scheduled delivery time. Similar to effort, time is broken down into increments or phases which are tracked based on what has been delivered so far. Note that milestone completion must be aligned with defined quality criteria to avoid poor quality/faults being detected too late.

5. Project progress. This is the key measurement during the entire project execution. Progress has many facets and should look to deliverables and how they contribute to achieving the project’s goals. Typically there are milestones for the big steps, and earned value and increments for day-to-day operational tracking. Earned value techniques look to the degree to which results such as implemented and tested requirements or closed work packages relate to effort spent and elapsed time. This then allows cost to complete and remaining time to complete the project to be estimated.

6. Quality. This is the most difficult measurement, as it is hardly possible to forecast accurately whether the product has already achieved the right quality level which is expected for operational usage. Quality measurements need to predict quality levels and track how many defects are found compared to estimated defects. Reviews, unit test and test progress and coverage are the key measurements to indicate quality. Reliability models are established to forecast how many defects need still to be found. Note that quality attributes not only measure functional quality but also relate to performance, security, safety and maintainability. Projects typically aggregate information in similar manner to a dashboard. Such a project dashboard allows all relevant information related to project progress against commitments, including risks and other information summarized on one page. This is typically accessible online and is periodically updated. Examples of project dashboard information are actual milestones reached against the planned dates, or showing the earned value at a given moment.

Project dashboards provide information in a uniform way for all projects, and thus do not overload the user with different representations and semantics to be learnt and understood. They provide immediately available information for decision making. They help in examining those projects that underperform or that are exposed to increased risk. Project managers can look more closely and examine how they could resolve such deviation in real time within the constraints of the project. All projects must share the same set of consistent measurements presented in a unique dashboard. Lots of time is actually wasted by reinventing spreadsheets and reporting formats, where instead the project team should focus on creating value.

A selection of the most relevant project tracking measurements is provided in Figure 3. There can be both direct measurements (e.g. cost) as well as indirect measurements and forecasts (e.g. cost to complete). Figure 3 shows a simplified dashboard used to track projects. It covers the major dimensions of milestone control, budget and cost control, quality level and earned value. These four dimensions give a rapid insight into progress in contrast to commitments and allowed resources. You realize in this dashboard the mixture of plan-driven tracking as we are used to from techniques such as PERT and the forecasting trends with defect predictions or earned value evolution.

These project control measurements are periodically updated and provide an easy overview of project status, even for very small projects. Based on this set of measurements, a small subset can be selected for weekly tracking of work products’ status and progress (e.g., increment availability, requirements progress, code delivery, defect detection), while others are reported periodically to build up a history database (e.g., size, effort…). Most of these measurements are actually byproducts from automatic collection tools related to planning and software configuration management (SCM) databases.

Figure 4 shows a customizable project dashboard that has all necessary information in one sheet, namely risks and open issues, budget and expense control, milestone control, earned value tracking, requirements and their respective implementation status, test planning and tracking and defects status. Built into the commercial eASEE PLM tools suite, it receives parts of its data from operational databases and others from the internal data backbone [6]. This ensures sufficient data quality to compare project status across all projects within a portfolio.

Make sure that numbers are consistent across these different hierarchies. Often aggregation hides insufficient data quality which is then only revealed when it is too late to improve those underlying processes. Figure 5 provides a practical example that we have established for a major Fortune 100 company. The corporate scorecard was the starting point and the necessary links to distributed operational
Experiences and Advances in Software Quality

Figure 3: Measurement Dashboard: Overview Measurements for Schedule, Cost, Quality and Earned Value.

data were established step by step with particular effort going into ensuring data quality by means of periodic reviews, governance and tool support. This small example also indicates that different processes such as corporate control, strategy management, portfolio management and project management are ultimately related. What goes wrong on one level must be visible on the next higher level – if it is beyond normal acceptable noise levels.

4 Process Measurement

Today, software is a major asset of many companies. Engineering investments are primarily spent on software development for the majority of applications and products. In our rapidly changing world, a company will only succeed if it continually challenges and optimizes its own performance. Engineering of technical products is currently undergoing a dramatic change. Ever more complex systems must be developed at decreasing cost and shortened time to market but keeping high quality standards. At the same time, competition is growing and the entry barriers to established markets are diminishing. The result is more competitors claiming that they can achieve better performance than established companies. An increasing number of companies are aware of these challenges and are pro-actively looking at ways to improve their development processes.

Development processes along the product life-cycle determine how things are done – end to end. They provide guidance to those who do and focus on what to do. Guidance means understanding and ensures repeatability. Focus means achieving targets both effectively and efficiently, without overheads, conflict or rework. Good processes are as lean and agile as possible, while still ensuring visibility, accountability and a commitment to results. Inefficient processes limit business opportunities and reduce performance because commitments can then not be kept and delivery is below expectations.

To remain competitive in terms of their software development, many companies are putting in place orchestrated improvement programs for their engineering processes. Process improvement in today’s business is crucial to:

- Improve competitiveness (productivity).
- Expand markets.
- Extend market penetration.
- Better support customers.
- Address their real needs more effectively, to better manage the increasing variety of software assets.
- Be more profitable.

Software measurement is a necessary precondition of performance improvement. Process and product improvement must be combined with a strong focus on business objectives and measurements for following-up on change implementation. Otherwise, the risk is high that too much attention is focused on processes and not enough on what is essential for customers and shareholders. Improvements are directed by business objectives and need to be measured continuously in order to ensure focus, efficiency, and effectiveness. Emphasis is given to setting and dealing with quantitative objectives and thus making the progress of the im-
Experiences and Advances in Software Quality

Figure 4: A Tailorable Project Dashboard View with eASEE Combining the Most Relevant Information.

improvement initiative visible. To survive against today’s global competition with extremely low entry barriers in the software industry, it is important to continuously stretch targets and always keep pushing for improvements.

We have introduced the concept of objective-driven process improvement (ODPI) in order to focus processes on the objectives they must achieve. Processes are a means to an end and need to be lean, pragmatic, efficient and effective – or they will ultimately fail, despite all any amount of effort behind them.

Figure 6 shows this goal-driven relationship from business objectives to concrete annual performance objectives on an operational level to specific process performance measurements. Improvement goals cannot be reached if they
are not quantified and measured, or, as the saying goes, "managers without clear goals will not achieve their goals clearly".

Figure 6 shows concrete instances of objectives and measurements, such as improving schedule predictability or reducing cost. Naturally, they should be selected based on the market and business situation, the maturity and certainly the priorities within the projects.

The following example shows how objective-driven process improvement is translated into concrete actions:

**Step 1:** Identify the organization’s improvement needs from its business goals. These business goals provide the guidance for setting concrete engineering performance improvement objectives on a short-term basis.

Example: The business goal is to improve revenues and cash flows and to reduce the cost of non-performance due to schedule delays.

**Step 2:** Define and agree on the organization’s Key Performance Indicators (KPI’s). Such KPI’s are standardized across the organization and ensure visibility, accountability and comparability. Naturally KPI’s must relate to the business goals. Measurements should drive informed decision-making. They must be used for effectively communicating status and progress against the objectives set for the business, process or project. Measurements, both direct and indirect, should be periodically evaluated in conjunction with the driving objectives in order to identify problems or make decisions.

Example: The selected performance indicator is schedule predictability. It is measured as normalized delays compared to the originally agreed deadline. Schedule changes after project start are not considered in this measurement to avoid the situation where projects would argue that the delay is justified due to changing requirements. This definition is agreed with product managers and business owners to prevent them challenge engineering teams later.

**Step 3:** Identify the organization’s hot spots such as areas where they are feeling the most pain internally or from customers. Typical techniques include root cause analysis of defects and customer surveys.

Example: Average schedule overrun in the company is 45%. A root cause analysis of delays is performed. 40% of delays result from insufficient project management, 30% of delays come from changing requirements, 20% from supplier delays and 10% from other causes. Customer surveys underline that requirements are often included in the project which is perceived as offering higher flexibility but in fact experience shows that these are not critical requirements but rather nice-to-have features.

**Step 4:** Commit to concrete improvement objectives. These improvement objectives should directly address the above weaknesses and at the same time support the overarching business objective. Good objectives are commonly considered to be SMART:

- Specific (or precise).
- Measurable (or tangible).
- Accountable (or in line with individual responsibilities).
- Realistic (or achievable).
- Timely (or suitable for the current needs).
These objectives are reviewed and approved by upper-level managers before proceeding further. This ensures that priorities are appropriate and that nothing relevant has been overlooked or misunderstood.

Example: Two improvement objectives are agreed, namely improving estimations and improving requirements development. The respective performance targets are agreed in a management seminar to achieve buy-in. Each project must have two estimates where the first is allowed to deviate by 20% and the second to deviate by 10%. Requirements change rate after project start has to be below 20% – except that customers pay a fixed fee calculated to be at least double the margins. Time-boxing and incremental development with prioritized requirements was then introduced to achieve those objectives. Requirements priorities were agreed across all impacted stakeholders from product management, engineering and marketing/sales before each new project start.

**Step 5:** Identify specific levers to start improvements and connect them to Return on Investment (ROI) planning. This is typically best done when using a process improvement framework such as CMMI [6]. This framework provides the necessary guidance on which best practices to apply and how processes relate to each other. Without the right levers, chances are high that objectives will not be reached.

Example: Focus will be on requirements development, requirements management, technical solutions, project planning and project monitoring and control. Project managers will be educated in project management techniques and negotiation skills.

**Step 6:** Perform a brief "gap analysis" of the selected process areas to identify strengths and weaknesses. A systematic look at weaknesses helps to focus limited engineering resources where it matters most.

Example: Requirements are collected rather than developed, requirements management satisfies basic needs for change management, engineering is too technology-driven and must be extended to capture business reasoning, project planning shows severe weaknesses in estimation and feasibility analysis, project monitoring and control has weaknesses in getting stakeholder agreement for changes.

**Step 7:** Develop a concrete action plan for the identified weaknesses. Avoid trying to change all of them at the same time. Use increments to subdivide bigger changes. Consider available resources and skills and get external support if you lack competencies, such as change management.

Example: The most urgent need is project planning. A dedicated one-month initiative is launched right away to install a suitable estimation method and train people on it. A tool for feasibility analysis is introduced in parallel because not much historic data was available. A history database is installed for a set of key project measurements. In a second phase, earned value analysis will be introduced. After three months requirements development is launched under the leadership of product management.

**Step 8:** Implement the changes to operational projects and measure progress against the improvement objectives that have been committed to.

Example: Performance measurements are collected from
all ongoing projects. Insufficient performance is not punished but carefully analyzed. Mostly it is found that too many changes still appear for no reason. As a consequence, a strong focus is given towards change management and change review boards. A weekly project review is introduced and after a few weeks enhanced with daily Scrum meetings of development teams. Requirements changes passing the change review board must have their proper business case or are not accepted. Marketing and sales are unhappy – but the results prove valid. It is they who do not want to take ownership and be held accountable for changes. This reduces changes within the first three months dramatically.

The first few projects have 20% schedule overrun (compared to the previous average of 45%) and two of them are even close to 10%. These two are further evaluated to identify best practices. As it turns out, the project managers demanded requirements reviews by product managers and testers before submitting requirements to change review boards.

The quality of requirements substantially improved. This change is immediately pushed forward by senior management to all projects. As it turns out, testers are unhappy because they have not got the time scheduled for doing the reviews. Rather than demanding overtime, senior management asks the projects to budget 5% slots each week for reviews. These are two hours which are sufficient to review 1-3 requirements with sufficient depth.

A goal-oriented measurement approach ensures that process improvement is embedded in a closed feedback loop (see Figure 7). Business-driven improvement objectives are translated into annual targets or key performance indicators. Those are reflected in, and tracked with scorecards. A history database captures process, product and project information. It helps in tailoring processes and with setting specific process and project targets. It also facilitates setting control limits for statistical process control. The feedback loop is built upon measurements from processes that are analyzed against control limits and compared with targets.

Questions such as "are we doing better or worse?" will provide feedback from the currently running engineering projects where changes should be integrated into the workgroups responsible for making such changes happen. People should make realistic commitments and are later held accountable for achieving them. Visibility means trust. The figures are not "made up" for reports; they are a normal (self-) management tool. Not providing visibility or not delivering committed results is failure. Deviations that are out of a team’s or a project’s own control are flagged in a timely manner to allow resolution at the next highest level. Accountability also means setting realistic and measurable objectives. Objectives like "reduce errors" or "improve quality by 50%" are pointless. The objective should be clear and tangible, such as "reduce the number of late projects by 50% for this year compared to the previous year." These objectives must end up in a manager’s key performance indicators. Goals that are measured will be achieved!

Process measurement does not stop with simple indicators for efficiency or effectiveness. In fact, what we have just described is just the beginning of process measurement. Full quantitative process management uses statistical techniques to identify process anomalies, to eliminate them and to ensure the processes will deliver what is asked for by the business (see Figure 8).

Process measurement values represent process behaviors over time, for instance, defect detection rates in different test cycles. The business needs to determine specification limits of the processes, such as a maximum of known defects released to the field. Often these specification limits are asymmetric.

In our example of field defects, a lower specification limit could be impacted by time to market, thus indicating that the product must be of the right quality. Internal process behaviors determine control limits. Control limits are determined by the way the process is defined, disseminated, automated, and executed. They represent the "natural" behavior of the process if executed as specified. For instance if we have a life-cycle with six different quality gates and respective verification and validation steps, we might end up with a remaining defect level of around 5% (assuming we are effective in removing 60% of defects at each step.). Depending on the volatility of the process, these control limits are inside the specification limits and allow adjustments, if for instance a higher quality level is required.

Having 5% residual defects might be good enough if evaluated with a reliability prediction model based on operational usage schemes. It would also need to have a support and service organization in place that can provide the necessary assistance to customers as it might be demanded by service level agreements.

We call this natural behavior of the process the "voice of the process" (VOP). If this natural, built-in process behavior is consistently achieved and exceptions are resolved immediately to prevent any trend away from the control limits, we consider the process to be stable. It is good enough to achieve what we have designed it for. If the voice of the process is within the requirements specification as determined from business needs or the "voice of the customer" (VOC), the process is called capable.

For immature organizations, the control limits are mostly outside the specification limits. Those organizations are have no idea how to adapt processes so that they tend towards specification limits. In fact, these organizations do not even think in terms of control limits, they just hope each time a project is started that it reaches the requested target date or quality level. But process volatility yields continuous surprises. The process is not capable. This explains why the CMMI has in its name the word capability.

In quantitative process management we distinguish between the voice of the process (VOP) and the voice of the customer (VOC). Improvement objectives are the voice of the customer. Business needs and the customer base determine how good processes need to be. Capability baselines are used to identify the normal process behaviors (VOC) and to compare to the process needs (VOC). A small exam-
Experiences and Advances in Software Quality

Figure 7: Goal-oriented Measurement Ensures that Process Improvement is Embedded in a Closed Feedback Loop.

Figure 8: Improving Process Performance with Quantitative Process Management.
more specific VOC derived from the high-level VOC. They are sufficiently detailed to start improvement actions. They do not however answer how to improve the respective processes. This is your own insight and experience. Often you need external help for this step and to identify appropriate improvement actions.

Obviously, the E4–measurement process that we introduced before also applies here. The improvement process has the steps of setting improvement or performance objectives, measuring the process, analyzing the process behaviors and improving the process by first stabilizing it, then keeping it within its specification limits and finally continuously improving it. This goal-driven approach together with using statistical techniques to evaluate trends, behaviors and relationships, is formalized in techniques such as Six Sigma or CMMI [1][7].

5 Practical Measurement Guidelines

Measurement programs mostly fail because they are disconnected from actual business. Too often things are measured that do not matter at all. Alternatively there is a data cemetery with lots of unused data which is just reported because "we need metrics". Perhaps there is no clear improvement objective behind what is measured. The primary question with software measurement is not "What measurements should I use?" but rather "What do I need to improve?" It is not about having many numbers but rather about having access to exactly the information you need to understand, to manage, and to improve your business. This holds for both project and process measurement.

As a professional in today’s fast-paced and ever-changing business environment, you need to understand how to manage projects on the basis of measurements and forecasts. You need to know which measurements are important and how to use them effectively. Here are some success factors for your projects:

- Estimate project time and effort and set realistic deadlines.
- Check feasibility on the basis of given requirements, needs and past project performance (productivity, quality, schedule, and so on). Do not routinely over-commit.
- Manage your requirements and keep track of changes. Measure requirements changes and set thresholds of what is allowed.
- Know what the stakeholders judges as the value of the project. Track the earned value of your project.
- Understand what is behind delays and defects. Do not let delays accumulate. Remove root causes and do not simply treat symptoms.
- Be decisive and communicate with a fact-based approach. Avoid disputes with management, clients and users.
- Use measurements to keep commitments on track.
- Continuously improve by means of looking to your measurements and then taking specific objective-driven actions. Follow through until results are delivered.

As a practitioner encourage your management to take decisions on the basis of measurements. Give them the necessary information before and after the decision so that they can follow the effects and compare to goals. As a manager ensure that you decide based on facts and analysis. Always consider the fourth E-letter in the E4–measurement process, namely to execute. Make sure that your decisions move your business towards agreed objectives.

References


Further Reading

C. Ebert, R. Dumke. Software Measurement. Springer-Verlag, Heidelberg, New York, 2007. ISBN-13: 978-3540716488 (reference no. 1 of this article). Description: People who design and develop software like to call themselves software 'engineers'. Yet few organizations have really institutionalized measurement of their products and processes. The book is right up-to-date in both fields and packed with practical advice which make it the desktop reference for software engineers when it comes to working with software measurements. It provides the necessary foundations for measurement and has many concrete industry case studies to help to transfer the results to your own environments. Additionally it offers a large number of "rules of thumb" and benchmarking data that can be used in projects and in process improvement and assessment activities. This article takes some content from that book.