

2. Making Measurement a Success – A Primer

*The perception of measures and harmony
is surrounded by a peculiar magic.
Carl Friedrich Gauss*

2.1. Why Measurement?

“The answer is 42” is the popular statement around which Douglas Adams wrote a series of thought-provoking and insightful science fiction books [Adam79, Adam95]. “42” was supposedly “the answer to life, the universe, and everything” in the world, the end to all questions and the final answer that would explain all the rules and logic that make our world move. A huge computer specifically constructed to this endeavor was working for centuries to derive that answer. There was only one difficulty with that answer 42, namely that the question was unknown. The computer seemingly was only asked to provide the answer but not to explain the logic behind and what question it did really address. Therefore, another even bigger computer was built and some books later, it would come back with the question behind that answer. Well, the question was surprising and confused rather than explained anything. Yet it helped to reveal that “there is something fundamentally wrong with the universe”. But that is another story.

There are many parallels to software measurement. We often measure, talk about numbers but hardly know what to do about what we measure.

Software engineering is without any doubt about measurement. IEEE’s definition of the discipline is as follows:

“Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software” [IEEE90].

Engineering requires observation, measurement, and calculation. For instance, a software engineer might be asked how long it will take to build an online game, or he might be asked how many failures will occur after the release of his embedded automation system.

Looking to engineering disciplines, such as civil engineering or mechanical engineering, one can find that the more mature an engineering discipline, the more measurements and indicators are practically used.

The awareness of and need for measurements in software engineering, management, and ICT systems has reached a high level, yet the success rate of a measurement initiative is still poor. Everybody collects measurements, numbers

can be found all over, reports look like telephone book – but hardly anyone makes use of these measurements. The answer of “42” (or other numbers for that matter) is everywhere but the questions behind and the practical use of this answer is invisible.

As industry practitioners, we are doing a poor job in using measurements for day-to-day work. Explanations which we hear when using measurements are manifold and can be summarized as follows:

- The designer: Software is an art and cannot be measured.
- The engineer: Measurements could be misinterpreted and abused by management.
- The team lead: Engineers had never been educated on using measurements.
- The project manager: There is no baseline available from previous projects.
- The product manager: Software engineering projects are highly innovative by nature and this cannot be measured.
- The department leader: Software projects are unique and thus measurements are not comparable.

These are obviously excuses for not doing what is normal behavior in any other engineering discipline. Clearly education matters, and therefore must be addressed while introducing a measurement program (see Chap. 6).

This chapter will provide a quick start to software measurement. It shows how to make use of measurements in concrete situations, such as project management and supplier management. It does not assume that you know statistics, measurement theory or goal-oriented measurement. This will be addressed later in the book. It just assumes three things, namely

- You want to know where you are;
- You want to specify where you want to go;
- You want to find out whether you are taking the right way from where you are to where you want to go.

Information theory had an answer long ago and distinguished between data and information. We need to apply the same logic and distinguish between the numbers and their meaning. We call the latter measurements. And we call the first garbage because it does not tell anything. It is much better to have few but meaningful measurements where we know baselines, trends, and how they can be explained and improved, than having many numbers and graphs but no idea what they say.

The winners in today’s competitive global economy are those who pay attention to their own and their organization’s performance. They need to know the answers to two questions: “Am I doing good or bad?” and “Am I doing better or worse?”

The first question is a look into a mirror and simply tells whether performance is good enough to sustain. The second questions goes beyond performance and compares over time or across projects or organizations. Just as champions and successful athletes practice and train in order to improve their performance, so

must all of us in the world of software engineering and ICT business. Success comes from tuning the process in the right direction. Trends pointing in the wrong direction and that conflict with high performance must be investigated and changed. Measurements help us focus on those things that matter and move us in the right direction. “42” is not always the answer, and even if it is the answer in a given situation, it only matters in a given environment at a given time. Furthermore it only helps if we understand the meaning of the number within that context and take appropriate actions. Let us look into how to make measurement meaningful for your own objectives. **Let us go beyond the numbers!**

During the early eighties both management guru **Peter Drucker** as well as software expert **Tom De Marco** drew attention on measurement by stating that you cannot control what you cannot measure [Druc73,DeMa82]. This statement is widely used to launch a measurement program. However it might be smart to look beyond the statement into the underlying people aspect. What does it really mean to get the expected results? It is attention, not just measurement as was pointed out by **Larry Constantine** [Cons95]. In fact, results are achieved when you monitor people. Parents know it from their children. They get what they pay attention to. If parents like pictures in red colors, they will get them. If they pay attention to singing, children will be happy to sing. The same holds for control theory and systems theory. What needs to be controlled is observed. We all know it from temperature control. Temperature can only be controlled if it is observed. To stay in control, we need to measure.

A famous example is the so-called **Hawthorne effect**. During 1925-1927 Western Electric Company, then the largest manufacturer of electric light-bulbs, wanted to evaluate the effects of lighting on productivity. They set up a series of experiments in their Hawthorne Works, located in Cicero, Illinois. In the first experiment the researchers experimented on three different departments. The striking finding was that they all showed an increase of productivity, whether the lighting increased or decreased. So the scientists extended the experiments with control groups, where in each experiment one group had stable lighting and the second either had an increase or a decrease of lighting. All groups substantially increased production during the controlled experiments. With individualized experiments it was found that if the experimenter said bright was good, the people in the experiment said they preferred the light; the brighter they believed it to be, the more they liked it. The same was true when he said dimmer was good. The series of experiments was concluded that the effects came not from lighting changes but rather from the mere attention to the people. If people receive the right attention and their work is observed and reported, productivity improves.

You can only control what you observe and measure. The act of setting objectives and monitoring them is the guarantee to receiving the expected results.

This chapter introduces to software measurement from a very practical perspective. It is fast-paced and should help you get started. First, section 2 will look into various needs for measurement. Section 3 proceeds with a lean measurement process that we call **E4-measurement process**. We structure the entire measure-

ment and management process into four parts, namely establishing objectives and a measurement process, extraction of information, evaluation and execution. We will not discuss here *how* measurements are selected in specific situations or environments, as this depends on the objectives and goals of a business process or an application area. We will come back to this question during the other chapters of this book. The final two sections provide hints for the practitioner and a summary on how to make measurement a success and how to be successful with measurement.

2.2. The Need for Measurement

It is eight o'clock in the morning. You are responsible for software engineering. On your way to the company building you run into your CEO. He inquires on the status of your current development projects. No small talk. He wants to find out which projects are running, about their trade-offs, the risks, and whether you have sufficient resources to implement your strategy. He is interested in whether you can deliver what he promises to the customers and shareholders – or whether you need his help. This is your chance! Five minutes for your career and success!

Sounds familiar? Maybe it is not the CEO but a key customer for a self-employed software engineer. Or perhaps you are a project manager and one of the key stakeholders wants to see how you are doing. The questions are the same as is the reaction if you have not answered precisely and concisely.

Is there sufficient insight into the development projects? If you are like the majority of those in IT and software companies, you only know the financial figures. Too many projects run in parallel, without concrete and quantitative objectives and without tracking of 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 [CIO03, IQPC03] (for project control see also Chap. 9).

No matter what business you are in, you are also in the software business. Computer-based, software-driven systems are pervasive in today's society. Increasingly, the entire system functionality is implemented in software. Where we used to split hardware from software, we see flexible boundaries entirely driven by business cases to determine what we best package at what level in what component, be it software or silicon.

The software business has manifold challenges which range from the creation process and its inherent risks to direct balance sheet impacts. For example, the Standish Group's Chaos Report annually surveys commercial and government information technology (IT) projects. They found that only 35% of the projects finished on time and within budget, a staggering 19% were cancelled before delivery, and of the remaining projects which finished late or over budget, they only delivered a fraction of the planned functionality [Stan07].

Measurements must be goal-oriented. Since measurements drive management decisions at various levels, they are directly linked to respective targets. Fig. 2.1 shows this goal orientation in practice. It follows the Goal Question Metric (GQM) paradigm of V. Basili [Basi94]. Starting with objectives which can be personal or company-wide it is determined what to improve. This first steps translates goals into what should be achieved in the context of a software project or process or product. A second step means identifying how the improvement should be done. Asking questions helps in clarifying how the objectives of step 1 will effectively (and efficiently) be reached. We should not leapfrog this intermediate step because it reveals whether we have fully understood the objectives in the first place. Once these questions – and the respective answers – have been addressed, the third step is to identify appropriate measurements that will indicate progress and whether the change is pointing in a good direction. As such the entire framework of questions changes. Note in this context that the GQM-paradigm is primarily looking into defining and selecting measurements and not on guiding actions to resolve issues and to implement change and direction.

The important paradigm change in goal-driven measurement is that the primary question 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.

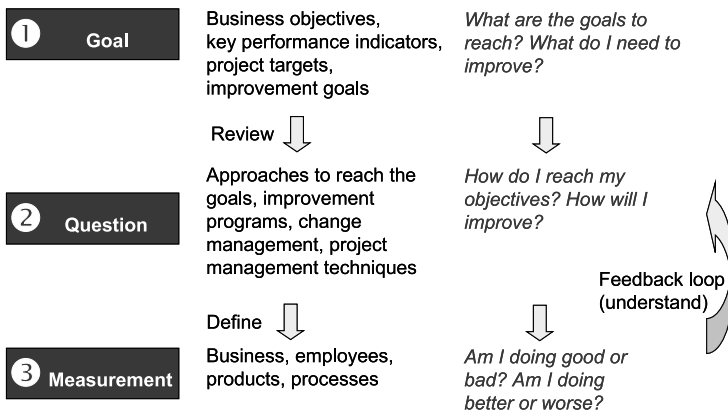


Fig. 2.1. Goal-oriented measurement following the GQM-paradigm

Software measurements ensure that the business is successful. They help us see what is going on, and how we are doing with respect to forecasts and plans. And they ultimately guide decisions on how to do better. Success within a software business is determined and measured by the degree the software projects and the entire product portfolio contribute to the top line (e.g., revenues) and to the bottom line (e.g., profit and loss). Late introduction of a product to market causes

a loss of market share; cancellation of a product before it ever reaches the market causes an even greater loss. Not only is software increasing in size, complexity and functionality, it is also increasing in its contribution to balance sheet and P&L statements.

With the increasing application of software measurement to manage projects and business processes, it became obvious that while GQM was setting the right basis for establishing a measurement program, it failed to drive an action-oriented feedback loop from measurements to direction and change. This is where paradigms such as Establish, Extract, Evaluate, Execute (E4–Measurement Process) come into the picture.

Software measurements do not distinguish first hand between IT projects, development projects or maintenance projects. The approach is the same. Most techniques described in this book can be applied to different types of software engineering, software management and IT management activities.

Fig. 2.2 shows the relationship between different stakeholder needs and the benefits they achieve by using measurements. Each group naturally has their own objectives which are not necessarily aligned with each other, and they need visibility how they are doing with respect to their own goals. On the senior management level, certainly measurements relate to business performance. A project manager needs timely and accurate information on a project's parameters. An engineer wants to ensure she delivers good quality and concentrates on the team's objectives (see Chap. 3).

Senior Management

- Easy and reliable visibility of business performance
- Forecasts and indicators where action is needed
- Drill-down into underlying information and commitments
- Flexible resource refocus

Project Management

- Immediate project reviews
- Status and forecasts for quality, schedule, and budget
- Follow-up action points
- Reports based on consistent raw data

Measurements

Engineers

- Immediate access to team planning and progress
- Get visibility into own performance and how it can be improved
- Indicators that show weak spots in deliverables
- Focus energy on software development (instead of rework or reports)



Fig. 2.2. Measurements depend on stakeholder needs. Their goals of what to control or improve drive the selection and effective use of measurements

Measurements help us

- Characterize and understand the current way of working;
- Provide baselines against which you can compare progress or improvements;
- Evaluate status so that projects can be controlled;
- Assess the impact of technology on products and processes;

- Establish far-reaching yet achievable objectives for project cost, schedule, quality and cost;
- Predict and forecast future performance;
- Communicate progress and improvement needs.

There is some additional literature around taking a business perspective on software. Most of it looks into examples and case studies to generalize principles [Deva02, Benk03, Reme00]. Some of it look into dedicated tools and techniques and their application, such as project control and management [Royc98, McGa01] (see also Chap. 8), management of global development projects [Eber01a, Eber06b] or knowledge management in R&D projects [Auru03]. A good introduction to the topic of business cases for software projects is provided by Reifer [Reif02]. A special issue of *IEEE Software* summarizes the state of the practice of “software as a business” [Mill02].

2.3. A Simple and Effective Measurement Process

2.3.1 The E4–Measurement Process

The measurement process is an inherent part of almost any business process (Fig. 2.3). It therefore easily applies to software measurement, be it performance engineering, project control or process improvement.

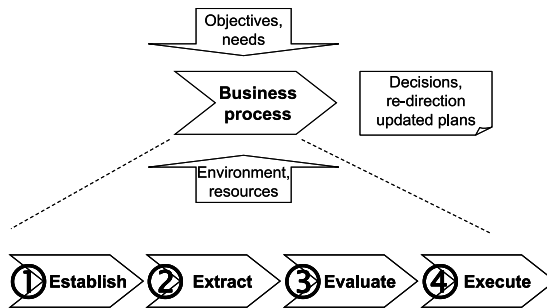


Fig. 2.3. The E4–measurement process with its four steps: Establish objectives and measurement activities, extract measurements, evaluate them, and execute subsequent decisions.

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 need
- **Evaluate** this information in view of a specific background of actual status and goals
- **Execute** a decision to reduce the differences between actual status and goals

The four steps all commence with the letter “E” which explains why we call this simple process the **E4-measurement process**.

The E4-measurement process is based on the Deming-Circle (Plan, Do, Check, Act) and extends the GQM-Paradigm by adding an immediate action-focus. The Deming-Circle has the steps of setting improvement or performance objectives, executing and measuring the process, analyzing the process behaviors and improving the process. It follows the observation that to effectively and continuously improve a process it is relevant to first stabilizing it, then keeping it in its specification limits and finally continuously improving it [Jura00]. The major difference is the clear focus on goal-oriented measurement and execution of decisions at the end of the four steps. Often the Deming-Circle is seen and implemented as a process of continuous improvement and organizational learning with statistical techniques. The underlying power of Plan-Do-Check-Act had been reduced in the past years to statistical process control and continuous improvement (even if Deming had envisioned a much broader scope). We hold that the approach is the major driver for taking and implementing business decisions.

The E4-measurement process (i.e., establish, extract, evaluate, execute) as introduced and widely used by the authors is a management paradigm and goes well beyond software measurement. It is grounded in measurement and quantitative techniques in order to achieve fact-based decision-making and sustainable impact of these decisions.

This E4-measurement process can also be portrayed as a closed control loop. We insist that it be closed by means of the last (execution) step, that is, execute decisions based on the information collected. Without the last step, we end up in collecting measurements but not using it to achieve our objectives and capitalize on concrete improvements.

Fig. 2.4 shows this closed loop where the execution is subsequently followed by a new circle of – adjusted – objectives or new questions to be resolved. We do see from the spiral-like picture that with each successfully finished circle, some improvements have been capitalized and will serve as the basis for further improvements.

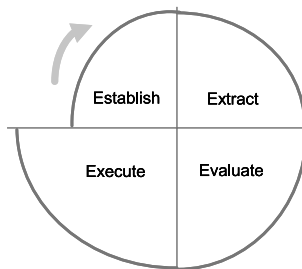


Fig. 2.4. To achieve lasting impact of decisions the E4-measurement process demands four steps in spiral-like continuous improvement circle

There is no use in extracting information and only recording it for potential further usage. If measurements are not used on the spot, if they are not analyzed and evaluated, chances are high that the underlying data is invalid. Without the pressure to have accurate measurements available, collection is done without much care. Where there is no direct use for information, the information is useless and so is the effort behind the collection.

2.3.2 Establish

The very first step in any measurement activity is to establish a scope derived from objectives that should be achieved. Measurement is not primarily about collecting numbers but rather about understanding what information is necessary to drive actions and to achieve dedicated goals. To “establish” therefore means to derive measurement needs from the objectives of the organization (which can be a company, business unit, product line, project or small team), to specify how the measurements are collected and then to extract this information from operational activities. This includes available and needed resources, skills, technologies, reusable platforms, effort, objectives, assumptions, expected benefits and market share or market growth. A consistent set of indicators must be agreed upon, especially to capture software project status.

Measurements selection is goal-oriented (see also Chap. 3). What are the best objectives? What objectives do we talk about? A good starting point is to identify how the projects or activities of an organization are viewed from the outside. Ask questions that affect the organization’s and thus your own future. What is hurting most in the current business climate? Are deadlines exceeded or changed on short notice? Is the quality level so poor that customers may move to another supplier? Are projects continuously over budget? Is the amount devoted to the creation of new and innovative technology shrinking due to cost of poor quality, rework, testing and maintenance? Who feels this pain first in the company? Which direction should the product portfolio take? What exactly is a project, a product or a portfolio? Is this what sales and marketing communicate? Where does management get information? Is it reliable and timely? What targets and quarterly objectives drive R&D? Such questions point to weaknesses and challenges. In order to improve on these observations, dedicated improvements are made which need to be measured.

The trap in most measurement programs is the disconnect between business-driven objectives and the measurement program. Too often things are measured that do not matter at all. Or, there is no clear improvement objective behind what is measured.

A simple example for a measurement need is in project management (see also Chap. 8). The goal is to master the project and to finish according to commitments. An initial set of internal project indicators for this goal can be derived from the Software Engineering Institute’s (SEI) core measurements [Carl92]. They simplify the selection by reducing the focus on 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.

Often the collected measurements and resulting reports are useless, and only create additional overhead (see also Chap. 6 on the misuse and abuse of measurements). In the worst case they hide useful and necessary information. We have seen reports on software programs with over 50 pages full of graphs, tables and numbers. When asked about topics such as cost to complete, expected cost of non-quality after handover or time to profit they did not show a single number. Sometimes they are even created to hide reality and attract attention to what is looking good.

Be aware – and prepared to react where necessary – that measurements are sometimes abused to obscure and confuse reality!

To make measurements a success, more is needed than just facts. It is necessary to look at opportunistic and subjective aspects. What role and impact do you have inside the enterprise? Who benefits most from the projects, and who creates the most difficulties for the projects? Why is that? What could you do to help this person, group, or customer?

Fig. 2.5 shows this goal-driven relationship between business objectives, concrete annual targets or objectives on an operational level and to dedicated indicators or measurements. 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. We show in Fig. 2.5 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 in the projects.

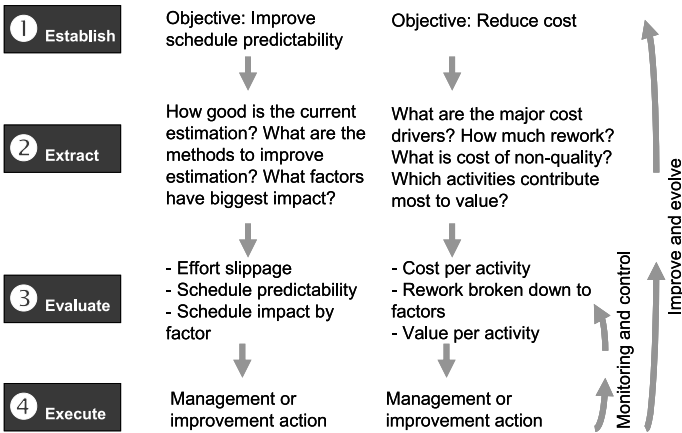


Fig. 2.5. Measurements are derived from goals. They help to achieve concrete objectives – if embedded into the E4-measurement process

To indicate measurement usage during this first step (i.e., establish), we will introduce a simplified product life-cycle as it applies to software, systems and IT projects. Fig. 2.6 shows this simplified product life-cycle in the upper part. It consists of archetypical phases as you find them in practically all product and service development, be it application software, internet software, middleware, enterprise and IT infrastructure, embedded systems, firmware, or services. For each of the life-cycle phases the lower part of the picture shows objectives or needs which are relevant for making the project and product a success. From these objectives, measurements are derived that are used in order to derive the subsequent go/no-go decisions during the life-cycle.

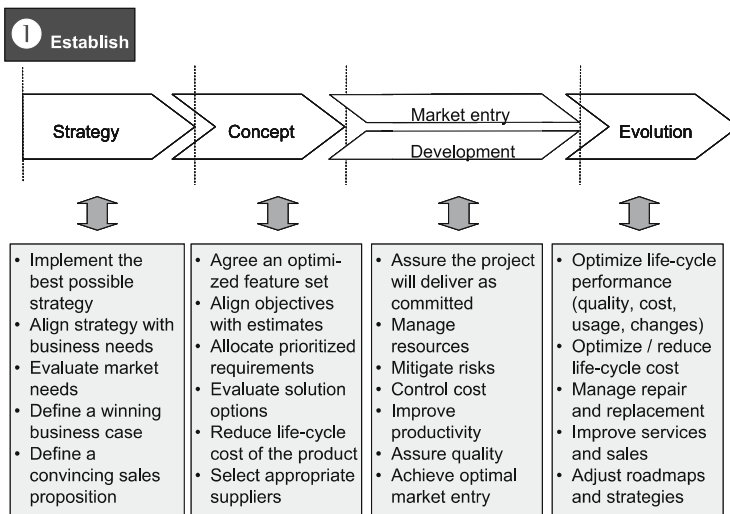


Fig. 2.6. Measurements along the product life-cycle. First the objectives and needs are established.

For instance, during the concept phase suppliers are evaluated. If the suppliers won't be able to commit to the necessary service level or content, the concept phase cannot be concluded and development would not be started. The phase depends on achieving the objectives relevant for the respective phase as stated in the diagram. Measurements help to gain sufficient insight to prepare and drive such decision.

To make the right assessments and decisions, the necessary information must be collected upfront. What factors (or targets, expectations, boundary effects) influence the investment? How did the original assumptions and performance indicators evolve in the project? Is the business case still valid? Which assets or improvements have been implemented? Which changes in constraints, requirements or boundary conditions will impact the projects and results? Are there timing constraints, such as delays until the results are available, or timeline correlations?

2.3.3 Extract

The next step of the measurement process is to extract the right information. To avoid overheads this second step must be closely linked to the first step that provides the measurement objectives. Do not collect information you will not use afterwards. Always be prepared to explain beforehand what you will do with measurements, and how you treat extracted numbers and their trends.

Assuming your objective is to reduce cost and the related action is to reduce rework. The measurements established in step 1 could be a cost measurement related to rework, such as the number of defects not found in the phase in which they are created. From your own previous product development or IT-projects the average values are known. Typically 60% of all defects are found after the phase or activity where they had been inserted. The distribution is in between 50% and 90%. The current project yields 50% at the time of release. Is this good enough? Compared to the previous distribution it looks good, and should be analyzed (next step). However it could also indicate that there were many more defects created than usual and out of this bigger amount a good part had been found immediately. This would be bad. You realize that results could vary dramatically with this one measurement. So this second step can also point to redoing the first step.

For illustration of this second step, we will go back to our simplified product life-cycle as it applies to software, systems and IT projects. Fig. 2.7 shows for each of these standard life-cycle phases some basic measurements that are necessary to steer that respective phase of the project or product release.

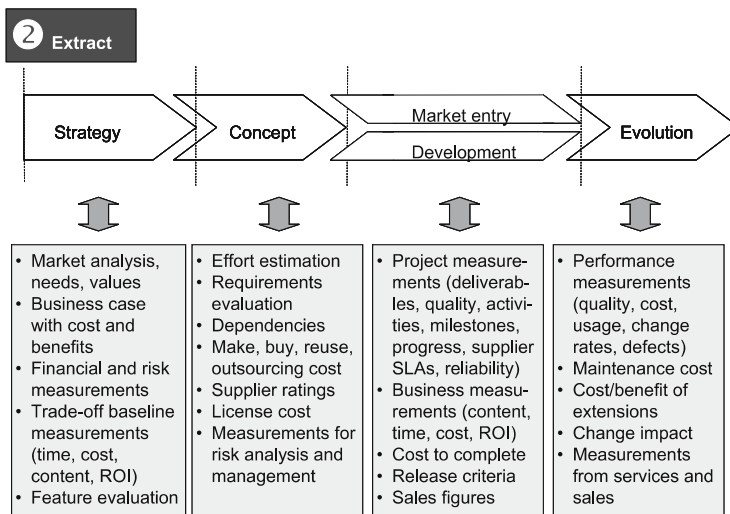


Fig. 2.7. Measurements along the product life-cycle. During each phase of the product life cycle dedicated measurements are extracted that correspond to established objectives.

These measurements are derived with a goal-driven method from the objectives which we indicated in the previous section (Fig. 2.6). Of course, the list is not

comprehensive and rather serves as an example how different measurement will help along the life-cycle.

Fig. 2.8 provides an example of how the same overarching dimensions of quality, productivity, deadlines, and employees are translated into different measurements depending on the perspective taken. An internal process view necessarily looks more into how processes can be improved, while the external perspective takes more a service- or product-related perspective. Often this is underlined by the type of agreements or contracts in the different client schemes of business processes.

	Quality	Productivity	Deadlines	Employees
Internal process view	Fault rate	FP / Person year	Percentage of work products within the 10% time frame	Skills
	Fault density	FP / Calendar month		Willingness
	Cost per fault			Overtime
	Root causes			Absence
		Tool usage		
External customer view	Customer satisfaction	Cost per feature	Delivery accuracy of final product to contracted date	Satisfaction with contact persons (sales, after sales, engineers)
	Delivered product quality			
	Functionality			

Fig. 2.8. Different stakeholder viewpoints determine how goals are translated internally and externally

Forward-looking figures need insight in order for us to be consistent in estimating cost at completion of a project or following up the earned value. Such a “dashboard” or status report compiles exactly those figures one need when comparing all projects. Try to automate the reporting, because for the project manager it is a useless – because overly aggregated – report. He should not be charged with such an effort.

Often indicators are available but are not aggregated and integrated. For instance, a quality improvement program measures only defects and root causes, but fails to look into productivity or shortened cycle times. Or in a newly created sales portal only access and performance information is known, while sales figures or new marketing mechanisms are left out of the picture. Fig. 2.9 shows how this aggregation is implemented in the various levels of an organization

Especially for software engineering projects, it is important to consider mutual dependencies between projects, organizational units, and so on. At the top is the enterprise portfolio which depicts all products within the company and their markets and respective investments. Further down a product-line or product cluster view is detailed which is aligned with platform roadmaps and technology evolution or skill building of engineers. For each product there should be a feature catalogue across the next several releases covering the vision, market, architecture and

technology. From such product roadmap a technology roadmap is derived which allows one for instance to select suppliers or build partnerships. It also drives the individual roadmaps of releases and projects which typically have a horizon of a few months up to a maximum one year. On the project level, these decisions are implemented and followed through to success.

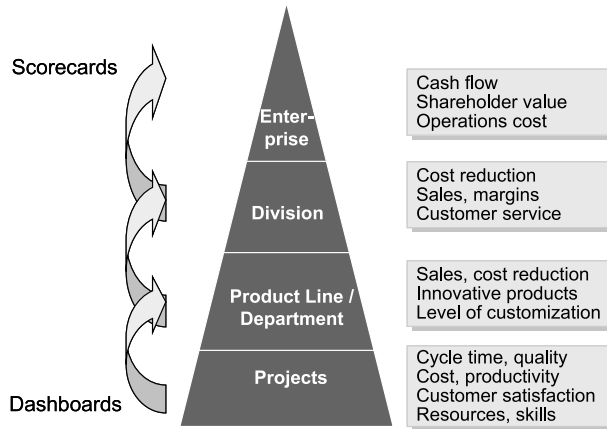


Fig. 2.9. Measurements are aggregated following the organizational needs. At project level focus is on dashboards, while at division or enterprise level it is on scorecards

Ensure 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. Fig. 2.10 provides a practical example that one of the authors had established over the years for a major Fortune 100 company. The corporate scorecard was the starting point and the necessary links to distributed operational data were established stepwise with a decent effort on 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 the acceptable noise level.

Such aggregation is not intended for micromanagement or command and control from the top but rather to ensure that the same data is utilized across the company. This has advantages not only on the cost side due to less rework and redundant data collection mechanisms, but also to ensure that risk management is based on exactly the same insight into operational and strategic baselines and that decisions can be tracked one day in case of external investigations.

Projects typically aggregate information similar to a dashboard. Such **project dashboard** allows to have all relevant information related to project progress against commitments, including risks and other information summarized on one page, typically online accessible with periodically updated data. Examples for project dashboard information are performance of milestones against the planned

dates, or showing the earned value at a given moment (for project dashboards, see also Chap. 8 and for scorecards, see Chap. 6).

Project dashboards provide information in a uniform way across all projects, thus not overloading the user with different representations and semantics that she has to wade through. They provide information at the finger tips – ready to make decisions. They help to examine those projects that underperform or that are exposed to increased risk. Project managers would 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 the project team should rather focus on creating value.

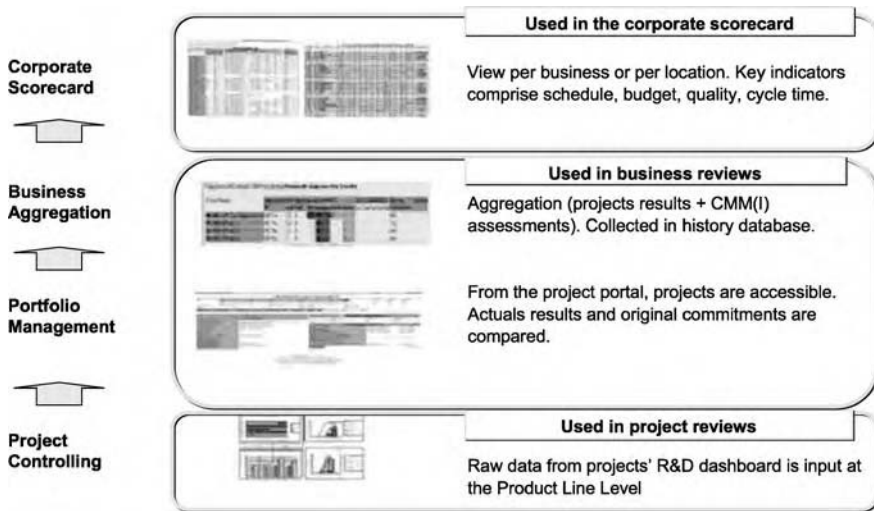


Fig. 2.10. Clearing the fog: Ensure a consistent view on all engineering projects, product lines and business units.

Fortunately, such a dashboard need not be time consuming or complex. Measurements such as schedule and budget adherence, earned value, or quality level are typical performance indicators that serve as “traffic lights” on the status of the individual project. Only those (amber and red) projects that run out of agreed variance (which, of course, depends on the maturity of the organization) would be investigated and further drilled down in the same dashboard to identify root causes. When all projects follow a defined process and utilize the same type of reporting and performance tracking, it is easy to determine status, identify risks and resolve issues, without getting buried in the details of micromanaging the project.

A standardized project dashboard is easy to set up and will not incur much operational cost. It builds on few standard measurements that are aggregated and represented typically in an intranet-accessible format to facilitate drill-down. It leverages on existing project management and collaboration tools from which it

draws its raw data (e.g., schedule information, milestones, effort spent, defects detected, and so on). It is self-contained and easy to learn. Key information is collected in a single repository with access control to protect the consolidated information. Having such a dashboard in place will ensure that project issues remain on the radar screen of the stakeholders.

By fostering early risk management by means of useful measurements and reliable predictions, it will once and for all take away the “I didn’t see that coming” response. A forward-looking attitude will be established automatically when excuses are taken away.

Performance monitoring is key. Standard measurements must be collected from each project and then consolidated for all the projects to evaluate the portfolio’s alignment with business objectives and performance requirements (Fig. 2.9). For the senior management levels the same information is further condensed into a scorecard that relates different businesses to the annual objectives. Scorecards should be balanced [Kap193] in order to avoid local optimization of only one target.

Combining and aggregating the raw data creates useful management information. For instance, a product with a new technology should be looked at from different angles. By using Linux instead of a proprietary operating system one might not only look into skills and introduction costs, but also into financial health of the packaging company or liability aspects in the medium-term. It is necessary to extract indicators from all operational areas and assess them in combination [Kap193, Hitt95].

As an example, let us look at different ways to measure success or returns from software engineering activities. Obviously there are operational figures that benefit calculation, such as phase durations in the product life-cycle, time to profit, time to market, cycle time for single processes, maintenance cost, cost of non-quality, cycle time for defect corrections, reuse rate, and license cost and revenues. Another dimension that is often neglected is improvements in productivity or people, such as cost per engineer, learning curves, cost per employee (in different countries), cost evolution, output rates per engineer or capital expenses per seat.

From these indicators one can select the few that reflect the assumptions of the original business case for the underlying products. They will help to further trace and predict costs and benefits. Indicators should be translated into the “language” of the projects or stakeholders to allow seamless extraction from regular project reviews without much overhead.

Once the necessary indicators are aggregated and available in one central repository or from a single portal, it is easy to generate reports covering the entire set of activities, projects, products or departments. Frequently asked queries are accessible online to save time and effort. Such reports, for instance, provide a list of product releases sorted to their availability. They can show average delays or budget overheads. One can single out projects with above-average cost and compare with their earned value, or one can portray products according to revenues,

market shares and life-cycle positioning. This reduces the effort to identify outliers which need special treatment.

2.3.4 Evaluate

After indicators and relevant project and marketing tracking information have been agreed upon and are available, evaluation of this information starts. This includes the evaluation of cost against benefits, business case, usefulness of the results from projects, and market readiness – all as future scenarios in terms of opportunities and risks. Such evaluation happens continuously and for the totality of projects. Even if product lines are not related technically or perhaps they even address fully different markets, it makes sense to evaluate mutual dependencies or synergies, such as resource consumption or asset generation.

For illustration of this third step, we will go back to our simplified product life-cycle as it applies to software, systems and IT projects. Fig. 2.11 shows for each of these standard life-cycle phases typical analyses that are necessary to steer that respective phase of the development. They are based on the extracted measurements (Fig. 2.7) and correspond to the objectives which were agreed during the first step (Fig. 2.6).

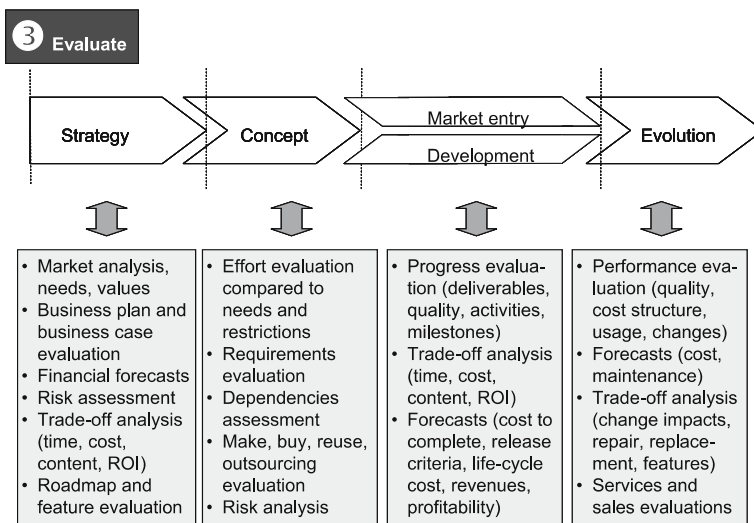


Fig. 2.11. Measurements along the product life-cycle. During each phase of the product life-cycle measurements are evaluated to ensure progress towards established objectives.

Certainly a monthly exercise for projects and products or a yearly exercise for portfolios as was done historically in many companies is far too coarse and falls behind the facts. Even the monthly or quarterly exercises preceding budget cycles and agreements turn out to be illustrative rather than guiding. On the project-level a weekly reassessment of assumptions should be an ideal balance of effort and

benefits. On the level of products and portfolios, a monthly reassessment still allows one to actively steer the evolution reflecting market changes. If projects change or deviate from the agreed objectives or if the risk level gets higher and the chances that the project manager can recuperate within the project get smaller, it is time to reevaluate and realign the project and portfolio with reality.

It is crucial to simultaneously evaluate *cost and benefits* or *trade-offs* between objectives. Regarding cost, the following questions could be addressed: Where are the individual elements of the portfolio (i.e., the products, product releases or projects) with respect to cost and cost structure? Is cost evolution following the approved plans and expectations? Is cost structure competitive (e.g., the share of test effort of the total development cost)? How is the evolution of the entire cost structure (e.g., is the trend going in a direction that allows sustainable growth)? Are the different elements of engineering cost under your control or are they determined from outside? Are all operational cost elements appropriately budgeted (e.g., covering maintenance, service, product evolution, corrections)?

We do the same for the benefits. Do the components of the portfolio follow the original assumptions? Is one investment better than others? Why is this the case? Which factors impact benefits? Which revenue growth relates to certain decisions or changes that have been implemented? What are the stakeholders' benefits from the IT or the R&D, both financially (e.g., return on assets (ROA), return on capital employed (ROCE)) as well as operationally (e.g., value generation for customers' businesses, market expectations, competitive situation)? What are the major impacts on performance and capacity to grow? How do internal processes and interfaces influence innovativeness, capability to learn or improvements?

Today "time to profit" is more relevant than "time to market" when evaluating benefits from different software projects. A delayed market entry in the world of Internet applications as well as other software solutions immediately reduces ROI dramatically. Entry levels for newcomers are so low and the global workforce is growing so fast that the market position is never stable. Even giants like Microsoft feel this in times of open source development and usage and an overwhelming number of companies who all want to have their own piece of the pie. As a rule of thumb one can consider that in a fast-changing market – as is the case for many software applications and services – a delay of only three months impacts revenues by 20% because of being late and the related opportunistic effects such as delays of subsequent releases.

2.3.5 Execute

After having extracted the necessary information and having evaluated the projects comprehensively, it is time to decide and to execute the decisions. Management means to actively take decisions and implement changes. The implications on different managerial levels are comparable. Whether it is on the project or the portfolio level, decisions need to suit the proposed scenarios. For instance, if risks grow beyond what the project can handle, it is time to reconsider features and pro-

ject scope. Maybe incremental deliveries can help with coping too big a scope and unmanageable size or duration.

If the value and benefits from a product release turn out to be below the standard expected return on the investments (e.g., the current interest rates for this risk level), they should be cancelled. Only those projects and products should remain in the portfolio that represent the biggest value and shortest time to returns. A certain percentage of the software budget should always be reserved for new projects and new technology to avoid being consumed by the legacy projects and products and becoming incumbent.

Different alternatives for decisions result from the previous two steps (extraction and evaluation). You decide only on this basis, as you otherwise invalidate your carefully built tools. Decisions should be transparent in order to influence behaviors and maintain trust and accountability. Particularly if projects are going to be cancelled it is important to follow an obvious rule set so that future projects can learn from it. For instance, if there is insufficient budget, a project is immediately stopped or changed instead of dragging on while everybody knows that sooner or later it will fail. Basic decision alternatives include doing nothing, canceling the project or changing the project or the scope of the project.

Do not neglect or rule out any of these three basic options too early. Different options should be further broken down in order to separate decisions which have nothing to do with each other. For instance, whether and which new technology is used in a project should not depend on the supplier. Each decision brings along new risks and assumptions that one need to factor into the next iteration of the evaluations. If one of the key assumptions turns out to be wrong or a major risk materializes, it is the time to implement the respective alternative scenario.

To ensure effective execution on project and product level, software measurement should be closely linked and integrated in the company's product life-cycle management (PLM). Typical software life-cycles might follow ISO 15288 (systems life-cycle [ISO02d]) or ISO 12207 (software life-cycle [ISO95]). They have in common a gating process between major phases based on defined criteria. These gates enforce evaluating the overall status (both commercial and technical) and deciding on whether and how to proceed with the project [Wall02]. PLM is the overall business process that governs a product or service from its inception to the end of its life in order to achieve the best possible value for the business of the enterprise and its customers and partners.

For illustration of this last (and most relevant) step, we will go back to our simplified product life-cycle as it applies to software, systems and IT projects. Fig. 2.12 shows for each of these standard life-cycle phases a sample of decisions and actions that are taken to ensure progress against commitments and objectives. They are based on the underlying measurements (Fig. 2.7) which are evaluated (Fig. 2.11) and correspond to the objectives which were agreed during the first step (Fig. 2.6).

Let us look at the scenario of deciding whether a project should be stopped before its planned end. This is typically a difficult situation, not only for the project manager and for the people working on the project, but because many organizations consider a stopped project a failure. Well, it might be from a financial per-

spective, but it will be an even bigger failure if it is not finished. Only few deal with such “failure” constructively, learn their lessons, and work on the next project. We will look at the decision from a higher-level perspective. There are costs after the decision to stop, such as ongoing infrastructure write-offs, leasing contracts and relocation costs. Often these costs are close to the cost of bringing the project to the intended end which implies a delivery. On the other hand, opportunistic factors should be considered, because the engineers could work on other projects that generate more sales.

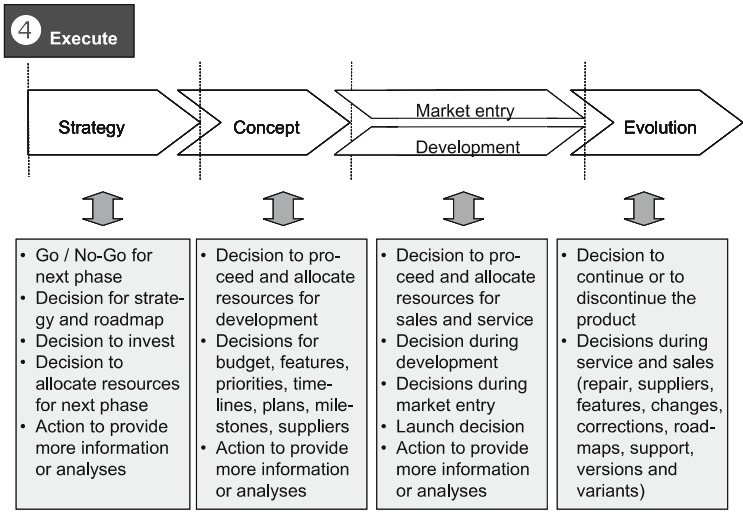


Fig. 2.12. Measurements along the product life-cycle. During each phase of the product life-cycle specific actions must be implemented to ensure that objectives and commitments will be achieved.

Especially for software engineering projects, it is important to consider mutual dependencies (see also Fig. 2.9). At the top is the enterprise portfolio which depicts all products within the company and their markets and respective investments. Further down a product-line or product cluster view is detailed which is aligned with platform roadmaps and technology evolution or skill building of engineers. For each product there should be a feature catalogue across the next several releases covering the vision, market, architecture and technology. From such product roadmap a technology roadmap is derived which allows one for instance to select suppliers or build partnerships. It also drives the individual roadmaps of releases and projects which typically have a horizon of a few months up to a maximum one year. On the project level, these decisions are implemented and followed through to success.

2.4. Hints for the Practitioner

Measurements should satisfy some simple criteria:

- **Goal-oriented.** Measurements must be goal-oriented to address concrete objectives. They are only created and reported if there is a specific need and decision you want to take based on these measurements. 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 and to identify problems or derive decisions.
- **Sustainable.** The measurements must be valid and useful over some period. Data integrity and consistency must be maintained at all times. This can be achieved if the operational measurement process is supported by tools. Measurements should be portrayed and compared over time. Often markets show cyclical behaviors which can only be assessed if the same indicators are followed up for several years.
- **Timely.** The indicators must be available literally on a push button approach. They must be valid and consistent with each other. If market data is one month old and quality reports arrive only half-yearly, this makes no sense. If cost to complete is derived monthly and you have project reviews on weekly basis, decisions are not consistent.
- **Meaningful.** The indicators should provide exactly the information you ask for. They should be aggregated to the level useful to make decisions. To avoid overloading occasional readers there should be few numbers with concise explanations. Further details should be drilled down from your portals and warehouses.
- **Balanced.** Look at measurement results from an overall perspective and do not focus on isolated measurements. They might signal something which can only be interpreted with further sources – or they might direct attention to the wrong issues. For instance, looking only at the number of defects or delays will not help if the impact behind these defects or delays from a business perspective is not considered. Look into all directions that are or will be relevant for your business. Most reporting has some blind spots. Often they appear at interfaces between processes, such as cost of an engineering activity or net present value of a platform with its evolution. Knowing about them is a first step to removing them. Starting from goals, it is easy to see why goals and measurements cannot and must not be separated.
- **Action-focused.** Measurements that result in reports and data cemeteries are good for statisticians but not to improve your business. As a practitioner help your management that they 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 analyses. 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.

Good objectives or goals should be “smart.” Smart goals are (and the first letters obviously show why we call them “smart”):

- Specific (or precise)
- Measurable (or tangible)
- Accountable (or in line with individual responsibilities)
- Realistic (or achievable)
- Timely (or suitable for the current needs)

Objectives should always be reviewed and approved by upper-level managers before proceeding further. This ensures that priorities are appropriate and that nothing relevant had been overlooked or misunderstood.

Here are some practical reporting hints:

- First, summarize in your reports all your results from a business perspective (an “executive summary”). Keep it short and to the point. Indicate conclusions and actions to be taken. Do not talk in this summary about the technology, as everybody trusts your technological competence – and they will probe further anyway once they go to the details.
- Always distill the value proposition to monetary numbers. Your management normally does not like what is not tangible and not traceable. They want to see the impact of a decision in money values.
- Distinguish clearly between capital expenses (e.g., licenses, hardware within your products) and project costs (e.g., persons, infrastructure). Often these means come from different sources or budgets; sometimes they are taxed differently. Certainly they appear in different sections in your profit and loss statement (P&L) and balance sheet.
- Review figures carefully and repeatedly. Many good ideas were killed too early because their presentation was superficial and included obvious errors. A small error in calculating measurements or a wrong underlying assumption will make all your efforts invalid.
- The measurements and numbers will mean different things to different persons. If you provide numbers, ensure that they are accompanied with precise definitions. That is simple with today’s online intranet technology, however, equally necessary for your presentations. Have a short definition in small letters on the same sheet, if the numbers are not well known in your company. Frequently presentations are distributed further or even beyond your reach. Often something is copied or sent out of context, and thus having the explanations on one sheet avoids unnecessary questions.
- Use industry standards as far as possible. Today many figures have standards and agreed-upon definitions that can easily be reused [McGa01, Reif02, Kapl93].

2.5. Summary

The four steps of the measurement process (establish, extract, evaluate, execute) should be introduced in that order.

First, look into the available data and how to achieve or improve visibility. Set up an inventory of all the projects, assets and proposals for changes or investments. Add to this list, and following the same structure break down, the major status measurements per project or proposal and the respective targets in terms of quality, effort, total cost, deadlines, sales and profit expectations. Then evaluate compared to your upfront agreed objectives. Drive actions from the evaluations and monitor the actions to closure. Be fast in suggesting and taking corrective actions where you risk missing objectives.

The two major observations we had over the past years with top-performing organizations were that

- (1) measurements are deeply ingrained into the engineering and management processes and
- (2) measurements are fertilized by the corporate culture.

What does this mean for the company? Measurement is a cultural change. Too often software engineers and management only realize the value of measurement when some unfortunate event has occurred. Projects are failing, product quality is below all expectations, deliveries are delayed, or teams realize that they will never meet the objectives. The culture change means building upon visibility and accountability (see also Chap. 6 for details on these cultural aspects).

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 the failure. Deviations that are out of a team’s or a project’s own control are flagged in due time to allow resolution on the next higher level.

Accountability also means to set 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!



<http://www.springer.com/978-3-540-71648-8>

Software Measurement

Establish - Extract - Evaluate - Execute

Ebert, C.; Dumke, R.

2007, XI, 561 p., Hardcover

ISBN: 978-3-540-71648-8