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-pace 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.

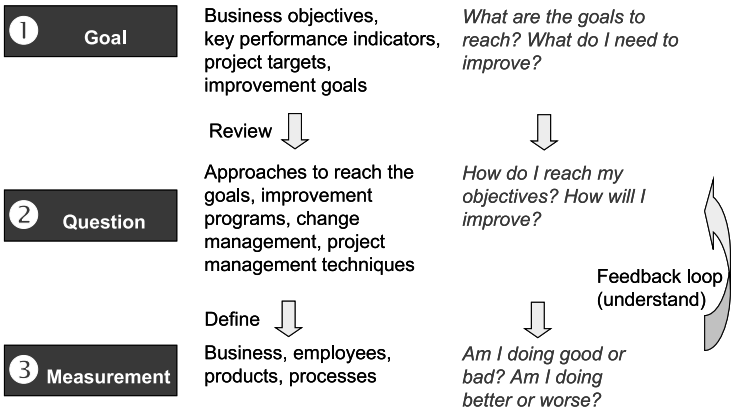

Fig. 2.1. Goal-oriented measurement following the GQM-paradigm

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 ultimatively 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).

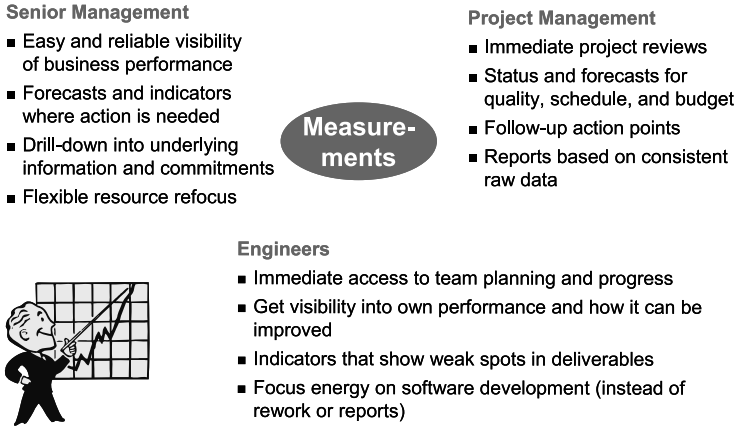

Fig. 2.2. Measurements depend on stakeholder needs. Their goals of what to control or improve drive the selection and effective use of measurements

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.

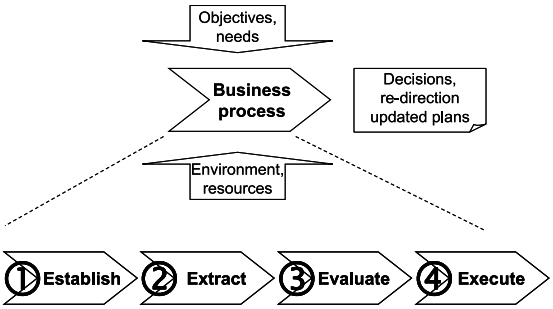

Fig. 2.3. The E4–measurement process with its four steps: Establish objectives and measurement activities, extract measurements, evaluate them, and execute subsequent decisions.

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.

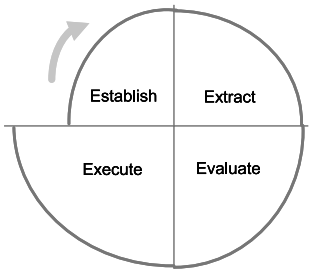

Fig. 2.4. To achieve lasting impact of decisions the E4–measurement process demands four steps in spiral-like continuous improvement circle

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.

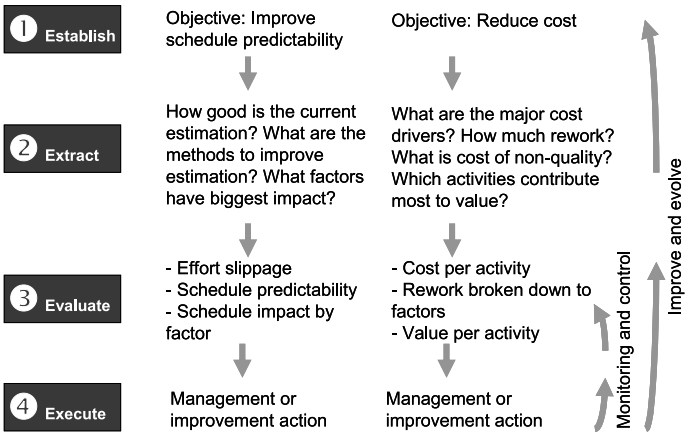

Fig. 2.5. Measurements are derived from goals. They help to achieve concrete objectives – if embedded into the E4–measurement process

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.

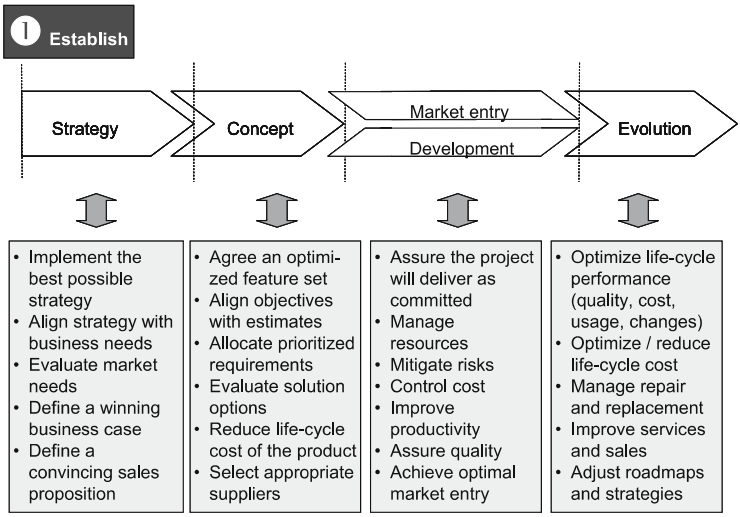

Fig. 2.6. Measurements along the product life-cycle. First the objectives and needs are established.

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.

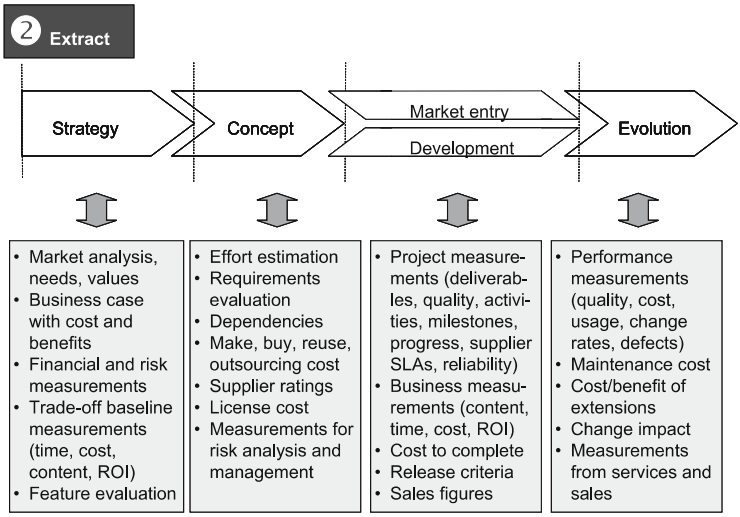

Fig. 2.7. Measurements along the product life-cycle. During each phase of the product life cycle dedicated measurements are extracted that correspond to established objectives.

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

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