

Six Sigma for Software

Dr. Thomas M. Fehlmann
Euro Project Office AG, Zeltweg 50
CH-8032 Zurich, Switzerland
phone: +41 44 253 1306
e-Mail: thomas.fehlmann@e-p-o.com

Abstract

Six Sigma has been very successfully applied in manufacturing. Is it also applicable to software, and ICT? Six Sigma is about cost reduction by eliminating defects. Six Sigma uses the following three principles:

1. *Focus on customers*
2. *Process orientation*
3. *Leadership based on metrics*

Applying Six Sigma to software development makes software projects transparent to both management and customers. Transparency requires an important cultural change. As a result, after transparency is achieved, completing accurate project estimations while meeting both deadlines and customer requirements becomes a lot easier. This paper will explain to readers how to:

- *Achieve transparency;*
- *Learn how to implement Six Sigma for managing software development projects;*
- *Understand the six steps for managing software development projects and listen to an explanation of how it impacts software development, software testing, project management, and project controlling;*
- *Discover the fundamentals for leadership, project management and SW architecture.*

In this paper we refer to a case study where Six Sigma has been successfully applied in software development.

1. What is Six Sigma? Does it apply to software?

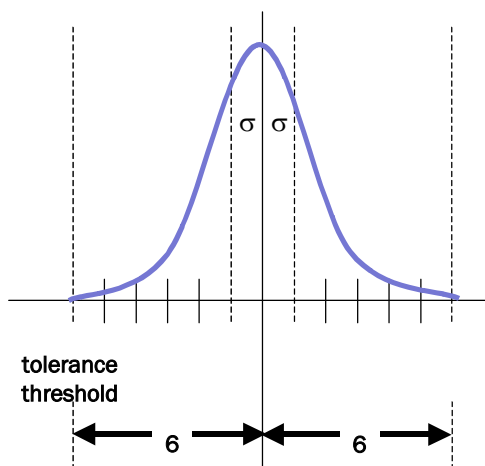
Since the early 90'ies, Six Sigma has been extensively used for achieving total customer satisfaction with innovative products at competitive price. Its objectives are to deliver products when promised, without delivered defects, early life failures, or failures during use.

1.1. The Six Sigma Approach

Six Sigma [10] is about measuring defects in the value chain process in order to systematically reduce them and therefore the corresponding cost factors. Sigma (represented as σ) is a statistical term that measures the deviation from the target set. A process that executes on 6σ level will yield results within the tolerance interval with 99.99966% probability.

Thus 6σ corresponds to only 3.4 defects per 1 million defect opportunities. A defect opportunity is a measurable process result that is critical to customer satisfaction.

Six Sigma is not team driven like TQM. It is a management approach. Defect metrics are the main tool for leaders to impact cost and margin.



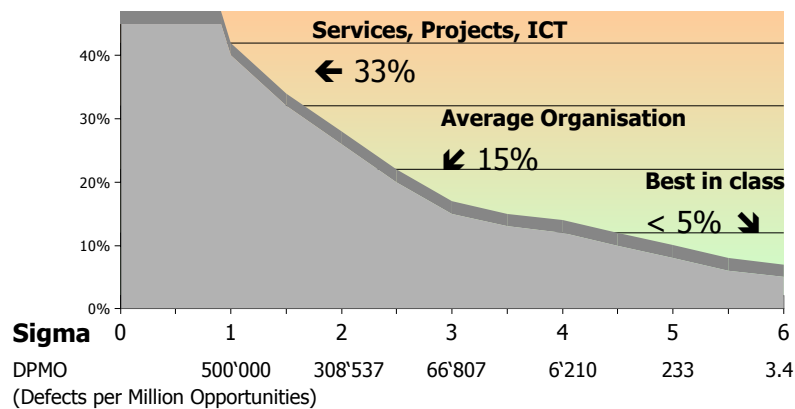
All metrics are customer driven. A defect is only what affects customer satisfaction. Thus before you can start Six Sigma you need to know what your customer really needs. Six Sigma is process – oriented. The term “customer” refers to who is using the output of your process and this is not always the end–customer. It is thus important to define what “defect” exactly refers to, when using a Six Sigma approach.

1.2. Benefits using Six Sigma

In manufacturing industry, hidden costs because of waste and rework in production processes are a major opportunity for cost savings. In Information and Communication Industry (ICT) this is even truer. When using every day’s Office applications, everybody knows how often the many defects in the software affect our productivity and effectiveness. For ICT projects, defect costs are even more important. There exist various estimations that identify hidden cost due to defects in ICT, depending upon the industry and application area. The famous CHAOS Report of the Standish Group [12] noted repeatedly that only about 20% of all ICT projects reach their targets. The rest will either be stopped (~30%) or miss important requirements such as functionality or deadlines (~50%).

Hidden costs

in % of turnover



Source: Concepts, Innsbruck; GE; own measurements

This means that we have an opportunity to get the benefits of ICT for significantly less cost, up to 30% or better. Indeed, organizations that use such an approach consistently report cost saving or margin improvements of this size.

1.3. Does it work for software?

Nevertheless, there were only occasionally attempts to implement Six Sigma for software. Applying Six Sigma to software development makes product development and other projects transparent to both management and customers. However, transparency requires an important cultural change. We know from experience that bad communication is a major reason why projects fail, and software projects in particular. We expect from better transparency that meeting both deadlines and customer requirements becomes easier.

The major problem with early Six Sigma attempts was that there was no connection of software metrics to economic success[6].

Counting mistakes and defects is not a clear indication if the software project is going to be successful (e.g. [4]). Other metrics like time-to-market and user friendliness are much more important in many application areas. Sometimes, reliability is of essence, but not always.

There is no software metrics that serves all. It depends what kind of software we are developing or installing.

Finding good metrics for software development or deployment is a major task in itself.

2. Why do we need Six Sigma For Software?

Software today is responsible for most of the added value in products, and must be blamed for many of its failures. Even if the iron hook breaks, it may be the software embedded in the measurement instrument to blame for not having detected it in time. When in Germany the high speed intercity express train crashed into an overpass, it was software that didn't detect the broken wheel ring well before the accident¹.

Mobile networks are suffering from not being able to provide interconnection to the Internet and interoperability between their own services. It is the software that fails.

In e-Commerce and for making Web Services to work, security, reliability and fault tolerance are of essence. Software and business processes are not cooperating, as they should to make it profitable.

Software is so ubiquitous that we must solve the software development problem to address a lot of other problems the society has.

The Six Sigma approach is:

- Set the goal – *Define*
- Define the metrics – *Measure*
- Measure where you go – *Analyse*
- Improve your processes while you go – *Improve*
- Act immediately if going the wrong path – *Control*

Throughout this talk, we talk about “software development”, however we not only intend writing new software, but also software integration, deployment, and maintenance, as long as it has the character of a project². This means that there is a goal that can be reached or missed. Using Six Sigma, we want to measure how much we are going to miss that goal.

3. Implementing Six Sigma for Software

In our experience with implementing Six Sigma for software we found the following three principles most useful:

- Principle No 1:
 - Measure customer related metrics only
 - Use Combinatory Metrics to cover all topics
- Principle No 2:
 - Adjust to moving targets
 - Your goals may need change; accept change and manage it accordingly
- Principle No 3:
 - Enforce measurement
 - Do not enforce meeting targets

The last principle is the most difficult one, because this is against the project manager's carnet of duties.

Indeed, this is what most often goes wrong in software projects: People want to enforce targets and consequently miss the goals. This is because setting targets that really lead to the goal is so difficult in software projects. Just consider the ever-lurking dilemma: Should we make the milestones, or do it right the first time such that we can later use the knowledge gained to proceed faster? Writing or implementing software is not just an engineering discipline but in essence it is knowledge transfer.

¹ It should be noted that the Columbia accident was *not* due to software, but to organisational failure of not reacting to the problem indications that were indeed abundant. It is an interesting question how much ICT reliability and organisational culture interact.

² Six Sigma for operation of a software – based system has similarities, including Combinatory Metrics.

3.1. Measure customer related metrics only

The goal of Combinatory Metrics is to assign the value delivered to customers to all the deliverables produced during a project

This is not straightforward for things like a concept, a software component, or a test execution during software development.

Impact on customer is dependent from success factors that describe what topics do actually influence the impact on customers. We call these influencing factors “Critical To Quality” (CTQ).

Thus we use QFD as a technique to combine metrics for different topics, especially if those metrics cannot be measured directly.

3.2. Adjust to moving targets

After having learned how to derive measurable targets from goals using Combinatory Metrics, we must face the fact that neither goals nor targets are as stable as should be in a well-controlled development environment. Disturbances such as new requirements, new insights, new technology, and new risks and threads force us to adapt to moving targets.

One of the most common misunderstandings is that goals once set are not to be changed. Six Sigma is not about sacred goal setting. Six Sigma needs a metric system that adjusts to moving targets.

3.3. Enforce measurement

Every project manager in the world (if he knows his job) will easily keep to any deadline you set to him, by reducing scope and quality of the deliverable. Software engineers are less good in that because they fear rework. Most problems in software development projects arise from that fundamental discourse. In the end, the customer will not get what he needs.

The answer to that dilemma is to monitor true work progress using objective evidence for work completion. This is done using the Six Steps to Completion metric.

4. Successful applications of Six Sigma

The following section uses references GMC Software AG, a small Swiss-owned niche company that delivers software for personalized digital printing for a worldwide market. They serve businesses with large customer bases, but with limited resources. As a niche company, they are strongly customer oriented. The software is written in a software development office located in Hradec Králové, Czech Republic.

Their customers want to combine flexibility with reliability and ease-of-use, thus in 2001, GMC set the target of reaching CMM 4 for their software development processes.

4.1. Six Steps to Completion

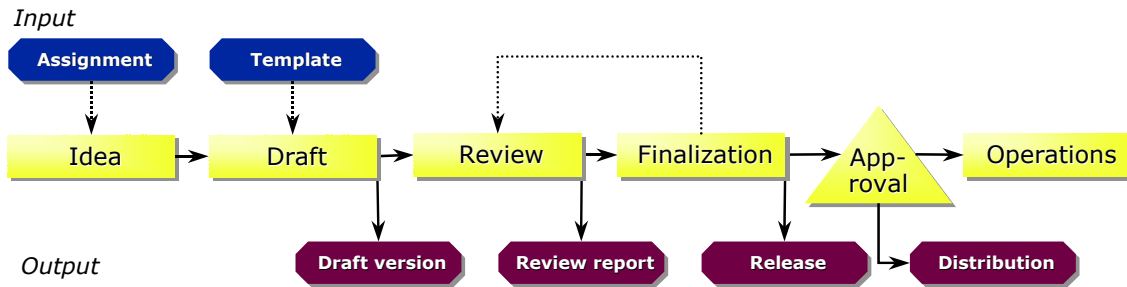
A deliverable is a uniquely defined and validated work result. The project consists of a sequence of deliverables.

Measuring the degree of completion for the deliverables that are needed for the project goals makes it possible to understand the project’s progress as a whole.

We can measure time line, cost and quality targets. Every deliverable must go through six steps. Among these are draft, review, and finalization before approval and usage.

It is possible to get objective evidence to identify the six development steps by recording the first draft version, the review findings, the final version, and the distribution list. Even for complex projects, such an overview is extremely helpful. We calculate a progress index that indicates overall progress and compare with planned and actual resource consumption. This metric enables the responsible project manager to assess whether he can meet the

schedule commitments, and thus identifies the measures. The assessment is based on objective evidence for progress, namely the planned quality assurance activities that must go with the deliverable.



Each of the six steps needs a percentage of the total duration. We set³ 10% for the idea, 30% for the draft, 15% for review, 20% for finalization, 15% to get approval, and 10% to make it operational. This progress grid yields progress metrics that can be accumulated for every project. Note that this metric is about *duration* and has not much to do with *effort*.

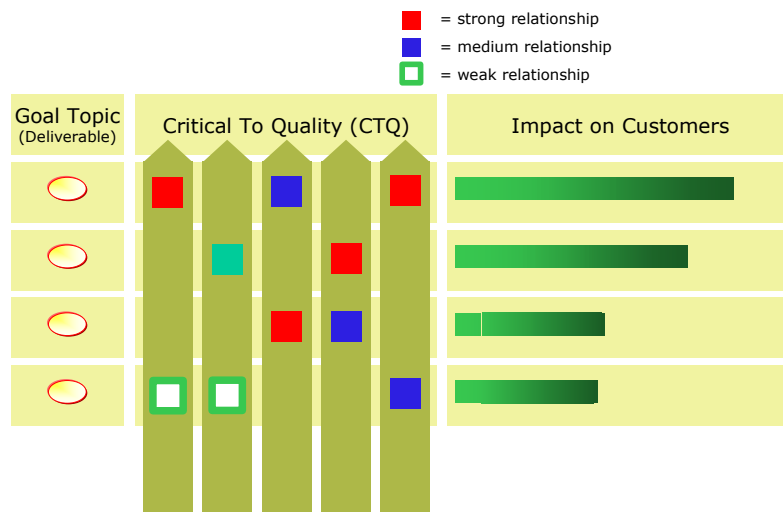
The side effect is that you have to plan for review and approval, which is part of quality assurance. Another side effect is that computing Earned Value and other project management metrics becomes much more reliable. Quality records from reviews and tests provide objective evidence for the completion of a task, if the task has a deliverable. Pending item lists are no longer needed to cope with unfinished tasks [9].

When using an integrated development with a document management repository, the state of the deliverable is monitored automatically. GMC's Czech development office has integrated all their task management in *Bugzilla* [2], an open source environment. Reports and progress metrics are produced automatically, without bothering the developers. Planned schedules are now met regularly, without pain. The defect count is how many deadlines committed to customers are missed.

4.2. Quality Function Deployment

However, meeting deadlines does not yet mean satisfied customers and success on the market. Six Sigma asks for customer orientation and the corresponding metrics. However, impact on customers varies for different deliverables during a software development process.

Quality Function Deployment (QFD, [1]) is the method of choice when doing Design For Six Sigma. QFD provides tools to analyse the impact that design decisions have on customers. We want to assign the impact value



³ Initially the percentages were an educated guess but later experience proved it a good approximation.

relative to customers to all the deliverables that are produced during a project.

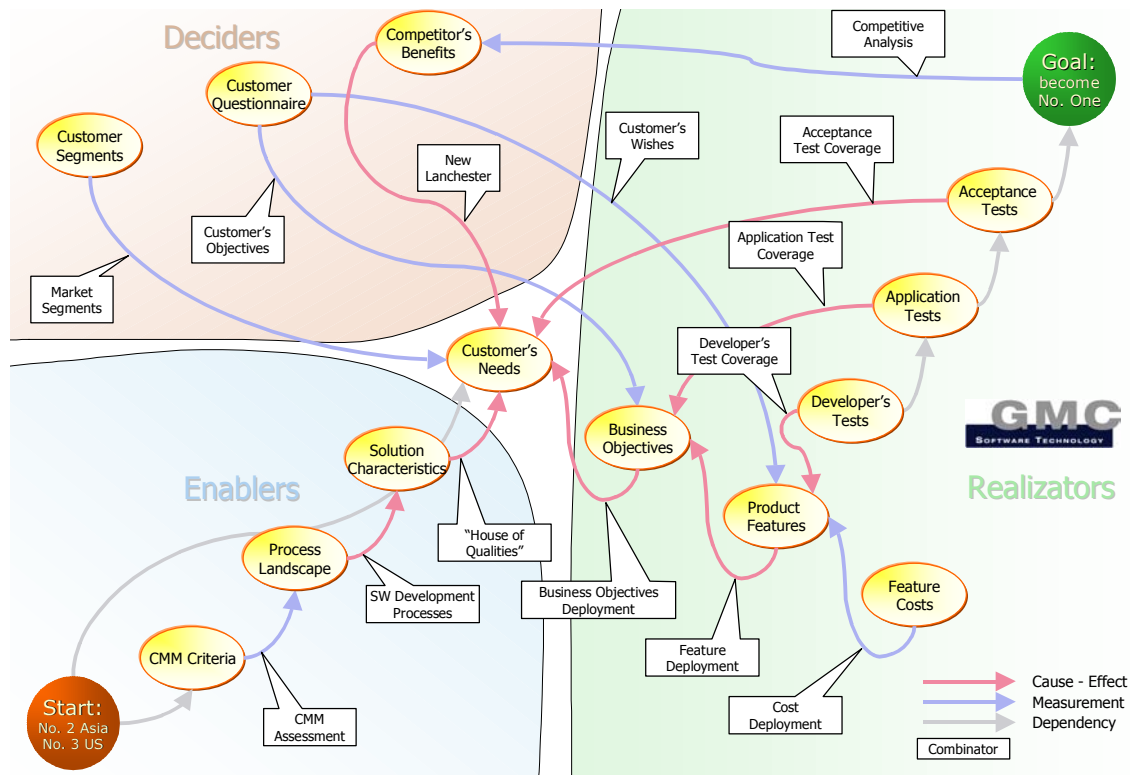
This is not straightforward for deliverables like a concept, a software component, or a test execution during software development. For this reason we use cause-effect matrices that quantify the impact on customer that is dependent from CTQs. These success factors are the parameters in the software development process that influences the result and thus have impact on customers. Such cause-effect influence constitutes a metric for the CTQs.

4.3. A Network of Deployments

In turn, the CTQs constitute the goal topics on the next topic level. For instance, the CTQs for fulfilling the Customer's Needs (CN) are the most important Solution Characteristics (SC) of the software. The solution characteristics result from the software development processes. The corresponding CTQs are the Key Process Areas (KPA) of the Capability Maturity Model (SW-CMM, [13]). Thus we get a chain of QFD deployments: KPA → SC → CN. This is the quality deployment chain, which we call the "Enablers".

There exist similar chains for the product deployment, or "Realizers". Cost deployment for software development depends from Feature Costs (FC) and thus from project costing. Product Features (PF) impact the Business Objectives (BO) supported by the software, which in turn impacts Customer's Needs (CN). The deployment chain reads FC → PF → BO → CN.

The third deployment chain is called the "Deciders". Here we have influencing factors of various kinds such as market share, market segmentation, competition, readiness to pay, and customer perception. Most important is the New Lanchester theory that allows transforming market share data into a metric for competitive advantage as perceived by the prospects.



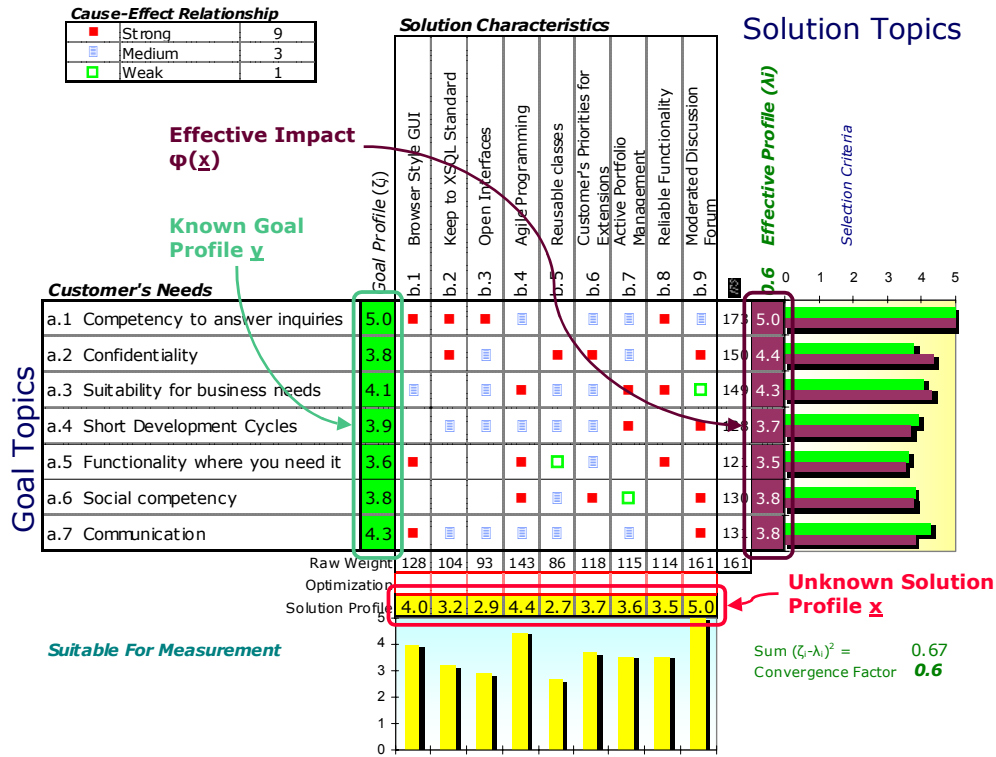
QFD is well established as a technique to combine metrics for different topics, which cannot be measured directly. However, the QFD deployment matrices are not only good for design, they also provide a means to carry measurements over from topic to topic.

4.4. Combinatory Metrics

The QFD matrices are cause-effect mappings between topics. The QFD matrices originate from Ishikava – diagrams that are used to visualise root cause analysis. Thus they always have a direction from the CTQs to its impact on the deliverables, not on the contrary⁴.

You can consider the matrices as linear mappings that map the solution topic's profile on the goal topics. When you choose a particular solution profile then this yields a corresponding goal's profile, by applying the linear mapping defined by the matrix. We call the result "Effective Profile". This compares with the original profile, the weights that we have set for the goal topics.

Combinatory Metrics studies the difference between these two profiles.



The two profiles coincide if the solution is ideal, that is, if the goal can be exactly reached with the given solution topics. This is normally not the case. In all other cases profiles differ; the *Convergence Factor* being a metric for the difference.

The formula for the Convergence Factor is the length of the vector difference between measured weight and derived weight, divided by the number of profile coefficients.

Let \underline{x} be the solution profile and $\varphi(\underline{x}) = \langle \lambda_1, \dots, \lambda_n \rangle$ the corresponding effective profile. Then we may compare $\varphi(\underline{x})$ with the goal profile $\underline{y} = \langle \zeta_1, \dots, \zeta_n \rangle$. The corresponding Convergence Factor is the length of its vector difference $\underline{y} - \underline{x}$ divided by the number of coefficients n , multiplied⁵ by 5:

$$5 * (\sum_{i=1..n} (\zeta_i - \lambda_i)^2)^{1/2} / n$$

⁴ This is a widespread wrong practice in QFD to calculate the profile of the CTQs out of the goal topic's weight. Such calculation is mathematically wrong, but often yields results that approximate the solution sufficiently well [10]. The Convergence Factor must be checked to verify how well the approximation is.

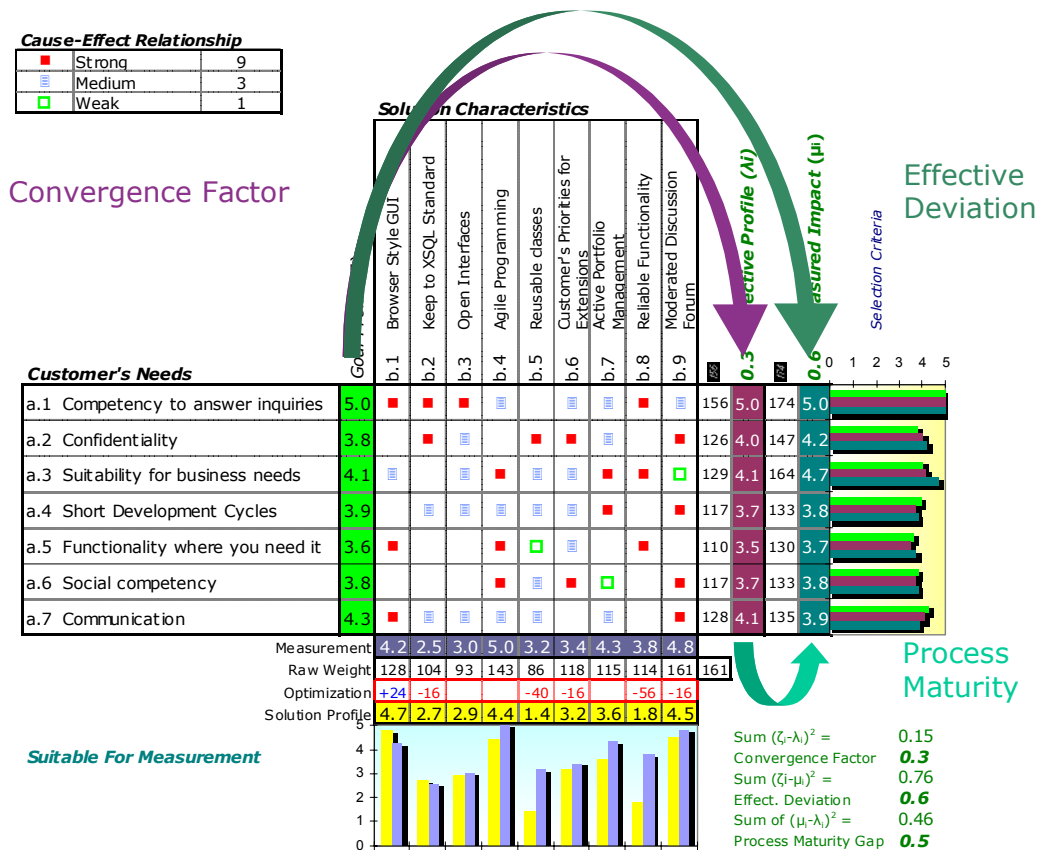
⁵ With the factor 5 we measure the length of the difference vector with the same units as the weights itself. Thus a convergence factor of less than 1 indicates that its difference is tolerable because within the precision order of the QFD matrices.

A Convergence Factor of zero means complete convergence; up to one it is considered acceptable. Convergence Factors greater than one indicate a significant difference between the deployed weight and the measured weight of the topic profile.

To find the optimum solution profile you first have to find the solution components that yield something close to the goal profile, and then you may optimize the solution profile.

4.5. Combining Deployments and Measurements

If you have an actual measurement for the solution topics, you can do the same profile calculation with the measured profile instead the optimum solution profile:

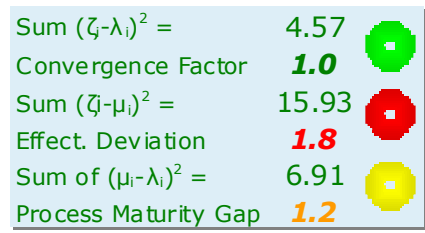


Now we can compare the measured with the optimum solution. This is the means to apply Six Sigma for Software. Defects for the Combinatory Metrics network are defined as differences between convergence factors and effective deviations that exceed 1. Although we have only few defect opportunities, the metric does indeed indicate how well the selected solutions for software development and for product characteristics meets customer's needs and expectations.

4.6. Product Development Process Metrics

If we have QFD Combinators with suitable measurements, we can define two additional metrics similar to the Convergence Factor.

Let y be the goal profile, x the optimized solution profile and x_0 the measured solution profile. Then $\varphi(x)$ is the effective goal profile and $\varphi(x_0)$ the measured impact profile. With these three vectors we define three metrics:



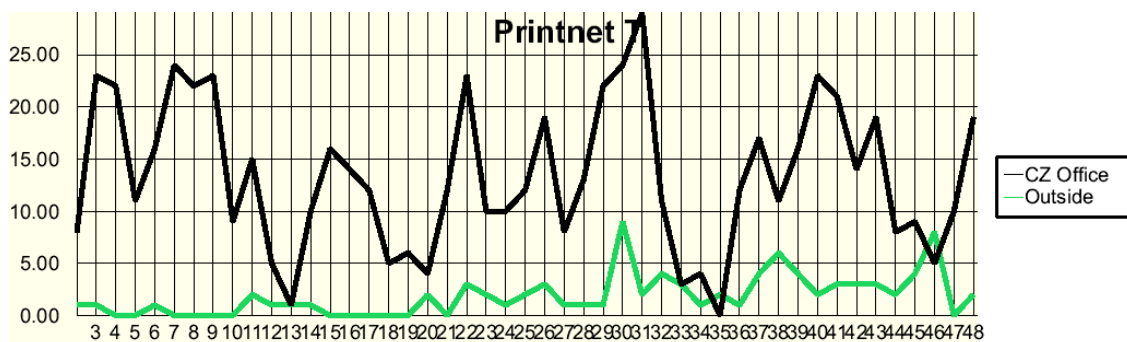
- Convergence Factor: The difference between goal profile ζ and effective goal profile λ shows how accurate the cause-effect – relationship is. It is a metric for the achievement of objectives under ideal conditions.
- Effective Deviation: The difference between goal profile ζ and measured impact profile μ shows how well the target goals are effectively met. It is a metric for the achievement of objectives as perceived by others.
- Process Maturity Gap: the difference between effective goal profile λ and measured impact profile μ shows the gap between what your development processes are aiming for and what they achieved in the real world. It is a metric for the maturity of your development process.

The tolerance indicators in the above figure are set to green for a value below 1.0, yellow for a value between 1.0 and 1.5, and red for values above 1.5. Such metrics are useful for Six Sigma approaches; see [8], [11].

4.7. Bug Count

The third defect count is about bugs delivered to customers. We compare them to the number of bugs the quality assurance department had detected in-house for the same release version.

As sample count we cite the bug count in the flagship product PrintNet T release 4.1 that comes to around $\sigma \sim 3$. The older product lines dwell around $\sigma \sim 2$; however, since the start of the software process improvement this is a tremendous improvement, where σ was well below 1, i.e. almost half of all defects were found by the customer.



The green line indicates the number of bugs found by the customer, the black line is the number of bug reports created during the alpha-internal and beta test phase (the latter includes the support people in the sales offices, but not customers)⁶.

5. Capability Maturity Model and Six Sigma for Software

We have three kinds of metrics:

- Progress track – project related
- Combinatory metrics – product related
- Bug count – quality related

The Six Sigma metrics had been introduced as part of the CMM program, but with Six Sigma in mind. They were also instrumental to get the ISO 9001:2000 certificate in

⁶ Note that this defect count does not include defects found and removed before testing started. There are yet no size metrics available that allow for a better estimate of how many defects are taken out during developer’s testing and with code reviews. For the overall σ estimate we assume that development delivers features with a level of $\sigma \sim 1$; however, it may actually be higher.

November 2003. Since all metrics are related to customer's needs, they serve as well for Six Sigma.

Based on these metrics, GMC reaches now a level well above $3\sigma^7$.

Interestingly this corresponds roughly to the CMM – level. In the last self – assessment, GMC rated in summer 2003 for 3.12, up from 1.57 in winter 2000/2001 according the SW–CMM scale. There is certainly a relationship between the two metrics, and the relation is most probably linear, although it cannot be assumed that the relation is exactly one-to-one.

Another relation is probably of more practical importance: GMC's business results improved tremendously during the last year, and this was certainly not a linear relation. It has more the character of a break-through, as to be expected from a successful Six Sigma approach. The break-through had been correctly predicted by the New Lanxhaster theory [15] that is part of the Combinatory Metrics measurement points.

6. Organizational Considerations and Economic Impact

6.1. Project Office

Establishing the metrics programme and the Combinatory Metrics network needs certainly a professional project office that cares for methods and tools and takes part in the project organisation. Professional project offices are a necessity for any organisation that wants to master the growing complexity in the ICT world with metrics – based improvement strategies such as Six Sigma.

6.2. Liability for Software

Six Sigma metrics for software development allows assessing the risks connected with ICT. The implication of such ability is far – reaching. For instance, insurers can get reasonable evidence for the risk actually involved in software and start offering insurance package. Based on such insurance software manufacturers can start changing their licence terms and offer guarantees for the proper functioning of the software. Liability can be defined in terms of Six Sigma metrics. Such ability will change the market and probably make software engineering indeed the discipline we so often dream about.

6.3. Cost of ICT

However, before this vision becomes true, the immediate benefit we get from any step towards Six Sigma For Software is that the cost of ICT can be dramatically reduced. Studies done in certain areas like e-Government suggest savings of up to 75% of today's cost level; experiences with the progress tracking metrics in particular support that finding. This will not only change the ICT industry but also impact all others that depend from it.

References

- [1] Akao Yoji. et.al: Quality Function Deployment (QFD); Productivity Press 1990, Portland, OR
- [2] Bugzilla Project Home Page <http://www.bugzilla.org/>
- [3] Cohen, Lou: Quality Function Deployment. How to Make QFD Work for You, Prentice Hall PTR 1995, New Jersey, NJ
- [4] Fenton, Norman; Krause, Paul; Neil, Martin: A Probabilistic Model for Software Defect Prediction, IEEE Transactions on Software Engineering
- [5] Fehlmann, Thomas: Risk Exposure Measurements on Web Sites, in: 4th European Conference on Software Measurement and IT Control, FESMA, May 2001, Heidelberg, Germany
- [6] Fehlmann, Thomas: Business oriented testing in e-Commerce, in: Software Quality and Software Testing in Internet Times, April 2002, SQS AG, Köln, Germany

⁷ Since size metrics are yet missing, we cannot pretend a higher accuracy so we give no decimals.

- [7] Fehlmann, Thomas: Combinatory Metrics for Software Development, in: 8th International Symposium in QFD, September 2002, Munich, Germany
- [8] Fehlmann, Thomas: Strategic management by business metrics: an application of combinatory metrics, International Journal of Quality and Reliability Management, Vol. 20, No 1, 2003 pp. 134–145.
- [9] Fehlmann, Thomas: Metrics for Project Management, in: 14th Annual UK Software Metrics Association Tutorials and Conference, August 2003, Wolverhampton, UK.
- [10] Fehlmann, Thomas: Linear Algebra for QFD Combinators, in: 9th International Symposium on QFD, December 2003, Orlando, FL
- [11] Snee, Ronald: Leading Six Sigma: A Step-by-Step Guide Based on Experience with GE and Other Six Sigma Companies, Financial Times Prentice Hall 2002
- [12] The Standish Group: CHAOS Chronicles v3.0, 1994 – 2003,
<http://www.standishgroup.com/chaos/toc.php>
- [13] Humphrey, Watts: Managing the Software Process, Addison Wesley Longman, Reading 1989, MA
- [14] Mizuno, Shigeru, and Akao, Yoji (ed. 1994): QFD: The Customer-Driven Approach to Quality Planning and Deployment, translated by Glenn Mazur, Tokyo: Asian Productivity Organization
- [15] Taoka, Nobuo: Lanchester Strategy – An Introduction, Lanchester Press Inc 1997