When use COSMIC FFP? When use IFPUG FPA?
A Six Sigma View

Dr. Thomas Fehlmann

Euro Project Office AG, Zurich, Switzerland

thomas.fehlmann@e-p-o.com

Abstract:

Six Sigma has become a major drive in industry and is rapidly gaining interest in software development and maintenance as well. The Six Sigma management strategy focuses on measurements for reducing defects early in the value chain processes and thus functional sizing measurements are a must for all Six Sigma Green and Black Belts that dare to deal with IT processes, be it in development or operations. However, which measurement method suits better to Six Sigma, the well established IFPUG 4.2 Function Points Analysis, or the more modern ISO standard ISO/IEC 19761, known as COSMIC FFP V2.2?

Interestingly, both measurement methods seem rather complimentary than competing when used in a Six Sigma setting, a setting rather targeted for defect avoidance than for project estimation with commercial or engineering background. The two methods serve different purposes.

Keywords

Six Sigma for Software, Combinatory Metrics, Quality Function Deployment, Functional Sizing, Function Points Analysis, Full Function Points, Use Case Points.

Zusammenfassung:

Six Sigma ist zu einem wichtigen Motor für die Verbesserung von industriellen Fertigungsprozessen geworden und gewinnt auch für die Software-Entwicklung an Interesse. Es geht darum, Fehler frühzeitig aus den Wertschöpfungsprozessen zu entfernen. Dabei spielen Messungen eine zentrale Rolle. Die funktionale Grösse von Software ist also ein Muss für alle Green Belt und Black Belts, die sich an Software wagen. Bloss: was passt nun besser zu Six Sigma, die bekannte und erfolgreiche IFPUG 4.2 Funktionspunkt Analyse, oder der neue ISO Standard ISO/IEC 19761, bekannt unter COSMIC FFP V2.2?

Interessanterweise haben beide Methode spezifischen Nutzen für Six Sigma. Dabei steht nicht nur die Aufwandsschätzung im Vordergrund, sondern die Vermeidung von Fehlern in der Funktionalität.

Schlüsselbegriffe

Six Sigma für Software, Kombinatorische Metriken, Quality Function Deployment, Funktionale Grösse, Function Points Analysis, Full Function Points, Use Case Points.

1 The Six Sigma Approach

Six Sigma is about eliminating defects in the value chain processes. A defect is a mistake or wrong behavior of the product, or in the service, that affects customer's needs. The process orientation of Six Sigma mandates that not only the end customers are regarded, those that use a process' results can cut costs when they receive error-free input for their process. A software-testing group that receives error-free code can concentrate on detecting application errors, ergonomic failures (all B-defects), or even late requirements errors (A-defects) rather than helping developers in fixing bugs.

However, writing software is a knowledge acquisition process. Since not everything is known from the beginning, knowledge acquisition is ongoing throughout software development and makes requirements volatile and growing. Thus software cannot be completely free of all sorts of defects. However, its statistical defect density in a given moment in its life cycle is computable. For density measurements, we need functional sizing information.

Six Sigma endorses the Define – Measure – Analyze – Improve – Control cycle as its approach to process improvement; in other words, it measures process results and their defect density for each process step. This well-established management method works well also for software. Since all software metrics somehow relate to software size, this means that functional sizing measurements is a must for every Black or Green Belt who wants to address software development, or maintenance. However, which sizing method to choose?

1.1 The Choice of Software Metrics

Functional size measurements is necessary to calculate defect density both for A-defects and B-defects. There are several ISO standards for functional size measurements: among them ISO standard ISO/IEC 20926 that corresponds to IFPUG 4.1 Unadjusted Function Points Analysis (FPA), or ISO standard ISO/IEC 19761, known as COSMIC 2.2 Full Function Points (FFP) [1]. IFPUG has issued the measurement manual V4.2 meanwhile [9]. Other sizing methods such as Use Case Points (UCP) [4] where also investigated.

1.2 The Criteria for Evaluation

One important criterion was the previous inability of the organization to estimate efforts for enhancing their software or writing new software for customers. In Six Sigma terms, this is a B-defect, since due dates are usually known, and if not met, it affects customers. Functional sizing as a base for effort estimation is well established and this organization – having no statistical data from the past – used the ISBSG Project Repository [10] for sizing.

Another criterion was the ability for early estimation. Very often projects are decided almost on the fly, and important investment decisions cannot wait until a sophisticated estimation process finishes. Such estimation must necessarily include all aspects of the project under investigation.

Furthermore, sizing must cope with the multi-tier web service architecture. The predominant applications are not data centric. They involve quite a number of complicated functional processes. All that gave FFP a head start over FPA, at least for the engineering people. How should FPA ever be able to correctly estimate their engineering work?

1.3 The Base Count

Counting existing applications was not much of a problem, in contrary. Both measurement approaches, FPA and FFP, proved capable and could be used to calibrate against each other, and, if available, against actuals. Many applications were services, accessible over browsers or from other applications. As expected, FPA modeled a user–centric view on the applications, FFP identified the functional processes behind it. Much of the contribution to the count came from the many persistent configuration data of the services, managed by transactions in the application and used by its functional processes. Thus there was little systematic difference between the two counting approaches, however, with FPP it was much easier to identify the application boundaries.

The problem was a different kind: it was very difficult to get good requirements to count when doing early estimation. In many cases, important aspects have not be seen early enough. And the biggest problem: early estimation almost never could be based on a use case analysis, or similar.

2 A Sample Sizing

Rather than the said company, we use for this paper a well-known example to demonstrate the issues encountered with the sizing approaches. All details of the counts are available on the web site of the author. [14]

2.1 The Wylie College Case Study

We use the Wylie College course registration system case study – documented in the Rational Unified Process (RUP Version 2003.06.00.65) as an example of Web site project – that was counted in [2] by Khelifi and Abran. All references are citations from [2]. The new system will enable all professors and students to access the system through PCs connected to the Wylie College computer network and through any personal computer connected through the Internet.

Furthermore, the new system will bring the Wylie College to the leading edge in course registration systems thus improving the image of the College, attracting more students, and streamlining administrative functions.

2.2 Count according COSMIC FFP

The *measurement viewpoint* in the Wylie College case study is that of the software developer who is interested in quantifying the functionality of the software he has to develop. The *measurement purpose* is to measure all of the software requirements documented in the vision document, the Release Notes Version 1.0 and Use-Cases Specification of this case study. The *measurement scope* is all of the software functional processes, and only these.

The count according COSMIC FFP [2] identified the following use cases:

Administrative:

- 1.1 Logon
- 1.2 Close Registration

Maintain Professor Information:

- 2.1 Add a professor
- 2.2 Modify a professor
- 2.3 Delete a Professor

Register for Courses:

- 3.1 Create a Schedule
- 3.2 Modify a Schedule
- 3.3 Delete a Schedule
- 3.4 Save a Schedule

Maintain Student Information:

- 4.1 Add a student
- 4.2 Modify a student
- 4.3 Delete a Student
- 4.4 Select Courses to Teach
- 4.5 Submit Grades
- 4.6 View Report Card
- 4.7 Monitor for Course Full

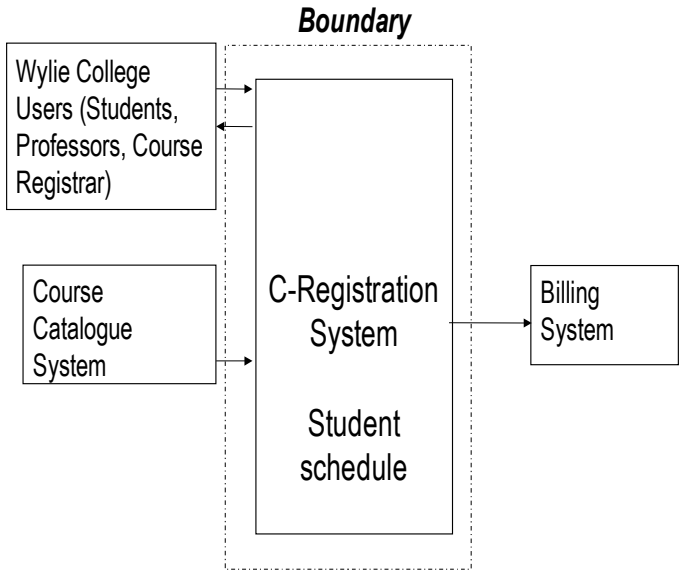

Fig. 1: Application Boundary for C-Registration System

The count yields 137 Cfsu (Cfsu = COSMIC Functional Sizing Units). There is not yet a well established PDR (= Project Delivery Rate) available for bench-

marks. The median for new MIS applications taken from the ISBSG Repository R9 [10] suggests a PDR of 10.2 hours/Cfsu. This corresponds to 1'397 hours total effort (PWE = Project Work Effort).

Note that the counter found the following ambiguities:

- In the 'Close Registration' use case specifications there is an issue stated by the authors 'Need to resolve what to do if too few students registered for a course'.

For this measurement, the following assumptions were made:

- We add the 'Monitor for Course Full' functional process in order to resolve it and to have a more accurate measure.

Thus, the count added an important technical requirement. However, how should we know that we have all functional processes identified, when we count?

2.3 Count with IFPUG FPA

For the business people with our customer, a count based on IFPUG FPA looks more attractive because the input and output requirements are much earlier known than the use cases and functional processes, which require analysis. However, for the engineering people, it is difficult to believe that such measurement ever yields a sizing comparable to a COSMIC FFP count from the developer's viewpoint, because creating the GUI is seldom the major cost driver for development. Other architectural or technical difficulties to solve may matter much more for cost.

For the Wylie case, we identified 3 ILF with 24 UFP (UFP = Unadjusted Function Points), 3 EIF with 15 UFP, 13 EI with 43 UFP, 10 EO with 45 UFP, and 8 EQ with 25 UFP. This yields a total of 152 UFP. For a Value Adjustment Factor (VAF) of 1.01, the total Function Point count according IFPUG 4.2 is 152 FP as well.

Based on the ISBSG Benchmarking database R9 [10], a likely PDR is 10.1, thus the PWE estimation yields 1'528 hours – a little more than the COSMIC estimation.

A side-effect of the count is to clarify the business requirements. For the count, we have to decide, whether we take one ILF "Registered Users" with 3 RETs, or three separate ILFs for Students, Professors, and Registrars.

- Without separate ILFs, a professor would not be able to register for a course. From the available sources of requirements, this is not specified, but that requirement affects the overall count!
- Furthermore, the user's addresses are not specified either. We decided, there must be an EIF for this, since neither use cases nor functional processes are provided for address maintenance, which is a significant application of its own.

These are two typical A-defects. Thus, we conclude, that functional sizing is essential when pursuing Six Sigma for Software, regardless counting approach.

2.4 Count with Use Case Points

Although Use Case Points [4] are no benchmarking standard, we also compared with this method. With a Technical Complexity Factor of 1.01, an Environmental Complexity Factor of 0.83, Unadjusted Use Case Points of 100, and a Productivity Factor of 20 we got a total expected effort of 1'668 hours. This is still in the same range, although the Productivity Factor is an assumption.

The Use Case Count did not give any additional insight into requirements and therefore was not really helpful.

3 The Six Sigma Approach

Combinatory Metrics is a technique that links business requirements to technical requirements – could it be used to combine the two measurements methods as well? FPA for the business requirements and the users' perception of business functionality required, to get early estimations, and FFP for sizing the technical requirements, to incorporate the technical concerns that often enough make an impact on the actual efforts used. In a recent study, the ISBSG has compared those projects in its data set which include estimations with the actual at the end of the project. While about 20% had reasonable estimates, for the balance of the projects, there were significant underestimates or significant overestimates.

Moreover, it is well known that all functional size measurements methods are good at uncovering missing business requirements (such as Data Element Types that are never queried; or data kept and managed in an application). FPA, in particular, is completely independent from the solution approach or architecture and thus helps to keep the focus on the business needs. But it cannot help detecting missing requirements in architecture or technology. In contrary, FFP is able to use developer's viewpoints as well and therefore is capable also to detect missing technical specifications and sometimes even to analyze the suitability for purpose of the selected architecture. Thus, if we could combine the two counting methods, we could possibly get the best from both to reduce our A–defects count.

3.1 Combinatory Metrics – QFD plus Metrics

Combinatory Metrics connects two functional topics. Functional topic describe the knowledge about the functionality of the system under consideration. Business and technical requirements are the two most important functional topics that can be combined using Combinatory Metrics.

Combinatory Metrics is based on the method of Quality Function Deployment (QFD) [3]. QFD is widely used as the vital part of "Design for Six Sigma", see for

instance [6] and [13]. According a communication of Prof. Akao, the matrices of QFD were invented originally as a convenient form to combine several Ishikawa ("Fishbone") diagrams. QFD is a cause/effect analysis in the form of a matrix.

Note that QFD for services – and software – is slightly different from QFD for physical entities. Cause/effect relationships do not describe physical forces, but rather the usefulness of certain solution approaches whether they are suitable means for reaching a goal. You can select which relationships to use; it is not given by physics.

3.2 The Cause/Effect Matrix – Describing Knowledge

Knowledge about two functional topics G and X, is the set of cause/effect relationships, which characterize how X relates to G. Formally, we write X→G.

The functional topics consist of requirements. A functional topic may have a potentially unlimited number of requirements. Usually, we limit our attention to only a selected few of those requirements.

Requirements are bound to functional topics. User requirements and the many kind of technical requirements are different. The failure to understand that difference is the main reason why software development processes are difficult to manage ([8], [11]).

A technical solution supports a business requirement, if that solution is expressed as requirements in the functional topic one step below. The technical solution is the cause why this software supports the said business requirement.

For instance, a certain number of use cases supports a required business scenario. Each use case, in turn, depends from the classes that implement the functionality required in the use case.

The solution requirements are not equally weighted, because they do provide specific contributions to the goal. It is a common practice to distinguish three different levels of cause/effect relationship: *weak*, *medium*, and *strong*. Strong relationship means that the respective solution requirement is a strong influence factor for the respective goal requirement; this corresponds to weight 9. Medium relationship means, it is an indispensable service but might be used in other contexts too; this gives weight 3. "Weak" means useful but not vital, and we assign weight 1. No cause/effect relationship corresponds to weight zero.

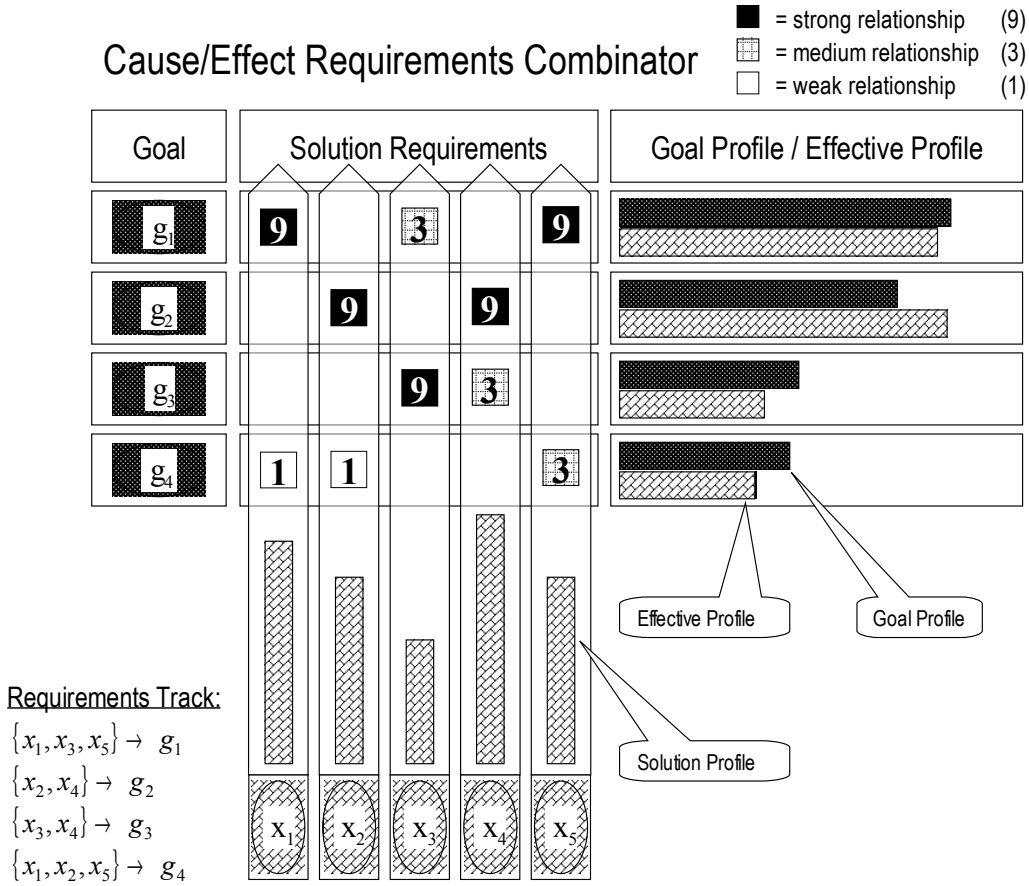

Fig. 2. Combinator relating requirements <g1, ..., g4> with solution <x1, ..., x4>

Well-established techniques exist for characterizing functional topics with only a few requirements [7], [12]. By choosing comprehensive requirements for functional topic G, you can keep the number of requirements low for that functional topic and thus describe the relationship between topics by just a few characteristic requirements on both the input and the output side.

3.3 Topic Profiles – Measuring Knowledge

QFD measures knowledge acquisition. This constitutes the basic idea behind "Design for Six Sigma". We do not need to size knowledge by some absolute scale. It is sufficient to compare the functional topics, assigning them weights. These weights yield a profile. The peaks in the profile correspond to high priorities among the requirements, the lows in the profile are the less important requirements.

Usually, business requirements constitute the goal, technical requirements are the means, or the solution approach, to reach the goal. The profile of the business requirements we call the ***Goal Profile***.

The solution requirements have equal weights either. Their respective importance weights also yield a profile for them: we call it the ***Solution Profile***.

It is natural to ask whether the chosen solution profile characterizes a technical solution that yields desired results. The profile of such a result, which can be achieved with the chosen solution, we call ***Effective Profile***.

Thus if we had a way to compute the effects of a solution profile, we could predict the effective profile for a given solution profile. Comparing the effective solution profile with the goal profile does then allow to find an optimum solution with an optimum solution profile, and optimizing the solution profile accordingly. With a little statistical analysis, we can say how stable the found solution is. Thus we relate technical requirements that describe the solution, to business requirements.

The QFD technique is widely used for Design for Six Sigma. The metrics aspect comes from using the 1-3-9 valuations of the cause/effect relationships. In this view, the QFD matrix becomes a linear mapping.

Let $\langle g \rangle = \langle \gamma_1, \ldots, \gamma_n \rangle$ be the goal profile, let $\langle x \rangle = \langle \xi_1, \ldots, \xi_m \rangle$ be the solution profile. Compute the vector formula:

$$\varphi\left(\langle x \rangle\right) = \langle \varphi_1, \ldots, \varphi_n \rangle = \left\langle \sum_{j=1}^{m} \alpha_{1,j} * \xi_j, \ldots, \sum_{j=1}^{m} \alpha_{n,j} * \xi_j \right\rangle$$

Then $\varphi(\langle x \rangle) = \langle \varphi_1, \ldots, \varphi_n \rangle$ is the effective profile.

3.4 Comparing Profiles – Analyzing Knowledge

However, how do we know whether our requirements combinators are accurate enough to allow for such backtracking? We need a metric that tells us how well our knowledge terms models the software development processes. For this, we need statistical process control for the requirement profiles.

To compare vectors, it is not sufficient to compare their difference. Although you can compute the difference vector as soon as you have the same amount of components, the result may be useless unless the components of the two vectors are of comparable size. In order to achieve that, we need normalization.

The requirement profiles, as normalized vectors, show the "direction" our project has to go on our quest for knowledge acquisition in the vector space of our functional topics. It is possible to eliminate requirements that are not contributing to the desired effect, or change our solution approach. This simply means, find a better solution.

3.5 The Convergence Factor – Improve the Cause/Effect Relationship

We need a metric to measure how well our choice of solution profile matches the goal. This metric we call the C*onvergence Factor*. It is the length of the profile difference between goal profile and effective profile, divided by the number of profile coefficients.

Let $<g> = <\gamma_1, \ldots, \gamma_n>$ be the goal profile, let $<x> = <\xi_1, \ldots, \xi_m>$ be the solution profile and $\varphi(<x>) = <\varphi_1, \ldots, \varphi_n>$ be the effective profile, computed by the vector formula as before. Then the convergence factor is the square root of the length of its profile difference $<g>–\varphi(<x>)$ divided by the number of goal coefficients n:

$$\kappa = \sqrt{\frac{\sum_{i=1..n} (\gamma_i - \varphi_i)^2}{n}}$$

(Convergence Factor)

A convergence factor (κ) of zero means complete match between goal and effective profiles. $\kappa = 0.2$ is generally considered good; $\kappa = 0.5$ is at limits, as this a deviation of direction in the underlying vector space by 10%. κ greater than one indicates a significant difference between the goal profile and the effective profile achieved with the chosen solution approach, meaning that such a solution may cause it go in a totally different direction. The convergence factor is a quality indicator for a cause/effect analysis.

When we have a matrix with a bad convergence factor, there are two resolutions:

1. Add better requirements to the solution profile that better supports the goal profile (e.g. better fit customer's needs), until the convergence factor decreases. This is the preferred way experienced by QFD practitioners.
2. Use the convergence factor as the minimum target for linear optimization. There are standard mathematical algorithms that reliably decrease the convergence factor by optimizing the solution profile.

Linear optimization finds a local optimum but does not detect all possible solutions. Moreover, it may give zero solutions, where real solutions exist indeed. It cannot replace the search for better solution requirements. For more details regarding linear optimization with QFD, see [5].

4 Using the Convergence Factor to get Functional Processes early

We use the convergence factor to complete the choice of functional processes.

4.1 Identifying Customer's Needs

The clue is in section 2.1 of this paper, where – very briefly – the customer's needs are listed for the Wylie College project: "The new system will enable all professors and students to access the system through PCs connected to the Wylie College computer network and through any personal computer connected through the Internet. Furthermore, the new system will bring the Wylie College to the leading edge in course registration systems thus improving the image of the College, attracting more students, and streamlining administrative functions."

Business Objectives Profile

	Business Objectives	Profile of Expert Evalution	Expert Evalution
BO-1 Functional	BO-1.1 Students register for courses on-line	*1.3*	*5*
	BO-1.2 Professors select their teaching courses	*1.0*	*4*
	BO-1.3 Professors maintain student grades	*1.3*	*5*
	BO-1.4 Access the system through Internet	*1.8*	*7*
	BO-1.4 Only browser is needed to use the system	*1.5*	*6*
BO-2 Quality	BO-2.1 Leading edge in course registration systems	*2.0*	*8*
	BO-2.1 Improve the image of the College	*2.3*	*9*
	BO-2.2 Attract more students	*1.5*	*6*
	BO-2.3 Streamline administrative functions	*1.8*	*7*

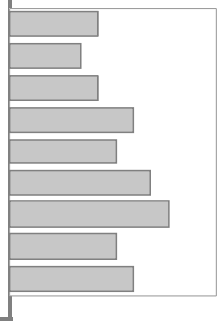

Fig. 3: Prioritizing Customer's Needs

This statement can be converted into business objectives, constituting customer's needs in our case. The priorities may have come from a Six Sigma workshop in the Wylie College IT department. This yields the goal profile.

4.2 Mapping Customer's Needs to Functional Processes

It is quite straightforward to map the business objectives to functional processes, and it cannot surprise that the convergence factor is excellent, almost ideal.

Functional Processes Deployment Combinator

Business Objectives	***Goal Profile***	FP-1.1 Logon	FP-1.2 Close Registration	FP-2.1 Add a professor	FP-2.2 Modify a professor	FP-2.3 Delete a Professor	FP-3.1 Create a Schedule	FP-3.2 Modify a Schedule	FP-3.3 Delete a Schedule	FP-3.4 Save a Schedule	FP-4.1 Add a student	FP-4.2 Modify a student	FP-4.3 Delete a Student	FP-4.4 Select Courses to Teach	FP-4.5 Submit Grades	FP-4.6 View Report Card	FP-4.7 Monitor for Course Full	***Effective Profile***
		Functional Processes																***0.04***
BO-1.1 Students register for courses on-line	1.3	3	9				9	9	3	9		9					9	1.3
BO-1.2 Professors select their teaching courses	1.0	3		9	3	3								9		3	9	1.0
BO-1.3 Professors maintain student grades	1.3	9		9	3	3		3		3					9	9		1.3
BO-1.4 Access the system through Internet	1.8	9	3				1			9				9	9	9	3	1.8
BO-1.4 Only browser is needed to use the system	1.5		3				1				9	9	9	3	9	9		1.5
BO-2.1 Leading edge in course registration systems	2.0		1	9	3	3	9	3	3		9	3	3	9	9	9		2.1
BO-2.1 Improve the image of the College	2.3	3	9	3			9		3		3			9	9	9	9	2.2
BO-2.2 Attract more students	1.5		3				3			3				9	9	9	3	1.6
BO-2.3 Streamline administrative functions	1.8	1	9				1	1	9	3	3	3	3		9	9	9	1.8
Solution Profile for Functional Processes		0.6	1.4	1.0	0.3	0.3	1.3	0.5	0.7	0.9	0.8	0.8	0.5	1.9	2.3	2.4	1.4	Convergence Factor
																		0.04

0.1 Convergence Range
0.3 Convergence Limit

Fig. 4: A QFD Deployment of Business Objectives into Functional Processes

The solution profile is almost a perfect match for the business objectives. The functional processes selected for this solution approach exactly match the objectives. Their solution profile reflects what the software engineering manager should invest into the software. The grading of students and the Report Card provide highest value; he should concentrate efforts there to get the most out of the resources allocated for implementing the project, and to get the highest possible satisfaction among its users. So far, this is pure QFD.

4.3 How to detect missing requirements

Indeed, if we go back to the early stages of the project, when the Use Case analysis was not yet available, what could have happen?

We investigated two likely possibilities to forgot some of the functional processes. In the first example, we forget to include the functional processes for modifying and deleting entries. We only count input functions. The resulting convergence factor immediately degrades.

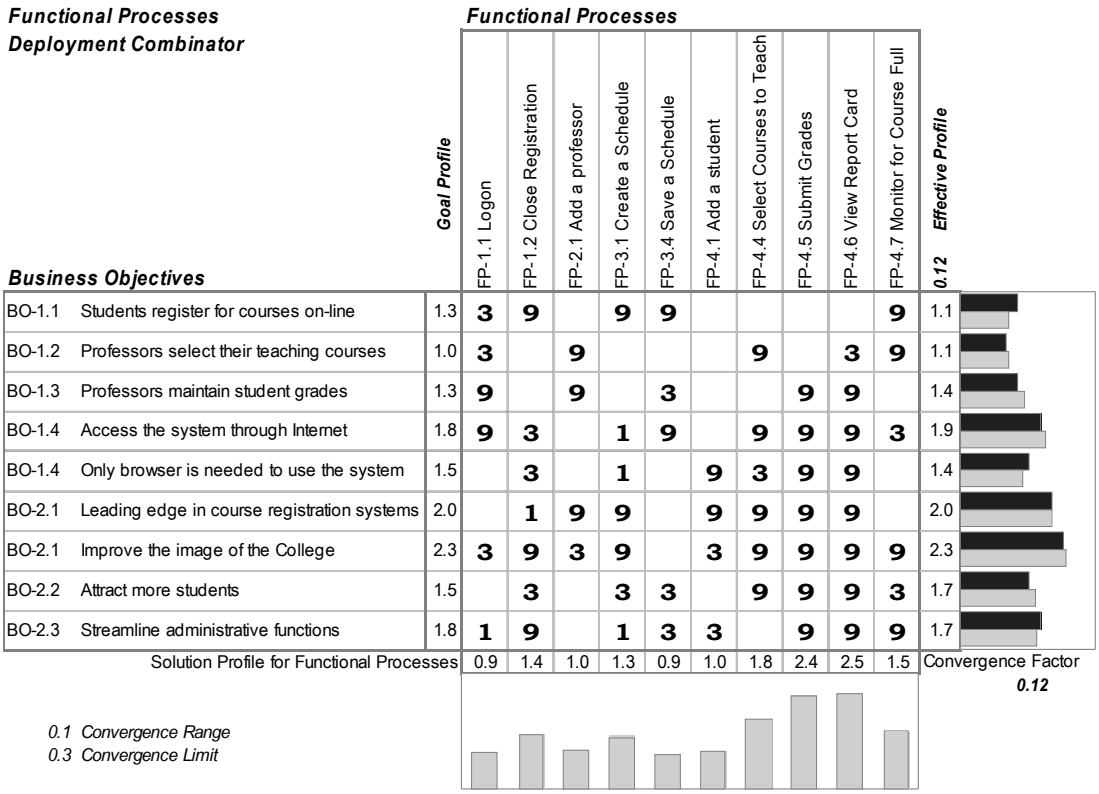

Fig. 6: Effect on convergence factor when omitting input flexibility

Even worse is the effect when the student's grading had been left out. This has been found in the QFD analysis the most important technical requirement, but may not be immediate, because it extends the scope of the old mainframe solution. You may even be temped to postpone that to "phase two"!

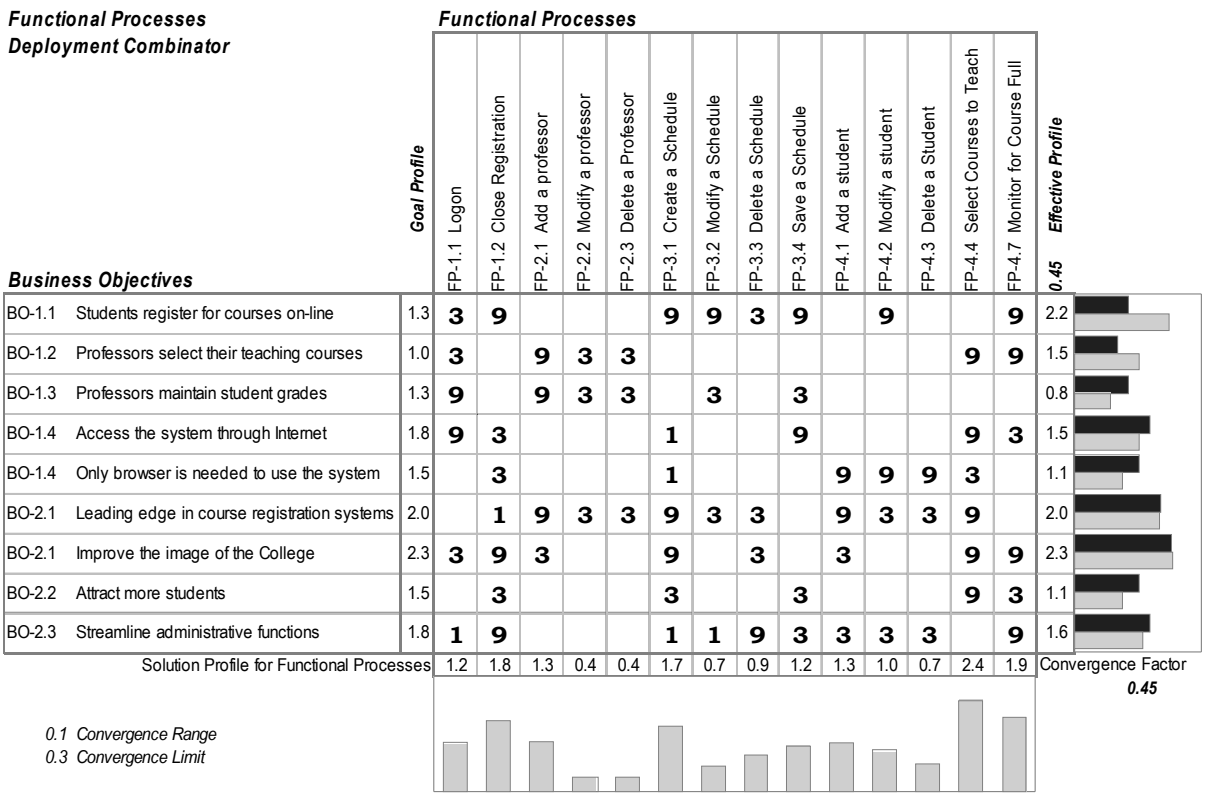

Fig. 7: Effect on convergence factor when Student's grading had been left out

It is obvious that the resulting solution profile is not able to fulfill the business objective BO-1.3 "Professors maintain student grades". The technical requirement needed to meet stated business objectives is missing. This may seem easy detectable in this sample case, but in reality such statements like BO-1.3 are often overlooked. Thus, using the COSMIC FFP counting approach also helps avoiding A-defects, and, on the other hand, using a Six Sigma approach to software development helps identifying the functional processes needed for early functional sizing using the COSMIC FFP counting approach.

5 Results

The Combinatory Metrics profile connects both functional sizing measurements, suggesting that there cannot be a single conversion factor that holds for all kind of FFP or FPA counts within an application area. Conversion between FFP and FPA is rather a linear mapping function hat depends from the relationship matrix between business and technical requirements. The Quality Function Deployment method generates that linear mapping between business requirements and technical requirements.

The Six Sigma approach gives interesting new insights into old problems. Combining Quality Function Deployment with FPA and FFP addresses both kind of problems usually encountered when developing software: the late deliveries (B-defects) and the wrong functionalities (A-defects). FPA is better in business, and FFP in technical requirements . The clue for success are always measurements.

Acknowledgement

Sincere thanks are given to all who have contributed with discussions and suggestions, especially in the session of the Swiss metrics association SwiSMA, an expert group of SwissICT.

References

1. Abran, A. et. al. (2003), COSMIC-FFP Measurement Manual - The COSMIC Implementation Guide for ISO/IEC 19761: 2003, Version 2.2, The Common Software Measurement International Consortium (COSMIC), Montréal, Kanada
2. Abran, A., Khelifi, A. (2004), Software Functional Size with ISO 19761: 2003 COSMIC-FFP Measurement Method, Université du Québec, école de technologie supérieure.
3. Akao, Y. et. al. (1990), Quality Function Deployment (QFD); Productivity Press, University Park, IL
4. Roy K. Clemmons (2006), Project Estimation With Use Case Points, CrossTalk – The Journal of Defense Software Engineering, February 2006
5. Fehlmann, Th. (2004), "The Impact of Linear Algebra on QFD", in: International Journal of Quality & Reliability Management, Vol. 21 No. 9, Emerald, Bradford, UK
6. Fehlmann, Th. (2005), Six Sigma in der SW–Entwicklung, Vieweg–Verlag, Braunschweig–Wiesbaden
7. Herzwurm G., Mellis, W., Schockert, S. (1997): Qualitätssoftware durch Kundenorientierung. Die Methode Quality Function Deployment (QFD). Grundlagen, Praxisleitfaden, SAP R/3 Fallbeispiel.
8. Humphrey, W.S. (1989), Managing the Software Process, Addison-Wesley, Boston, MA
9. International Function Points User Group IFPUG (2004), IFPUG Function Point Counting Practices Manual, Release 4.2, Princeton, NJ
10. International Software Benchmarking Standards ISBSG (2004), ISBSG Estimating, Benchmarking and Research Suite R9, Hawthorn, Victoria, Australia
11. Paulk, M.C., Curtis, B., Chrissis, M.B., and Weber, Ch. V. (1993), Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, Carnegie-Mellon University, Pittsburg, PA
12. Saaty, Th. (1999), „Fundamentals of the Analytic Network Process", in: ISAHP 1999, Kobe, Japan
13. Töpfer, A. ed. (2004), Six Sigma – Konzepte und Erfolgsbeispiele für praktizierte Null–Fehler Qualität, 3. Auflage, Springer Verlag Berlin Heidelberg New York
14. Web Site for this paper: www.e-p-o.com/WhichSizingMethod.htm