

Metrics for Cooperative Development Processes

Dr. Thomas Fehlmann

Euro Project Office AG, Zurich, Switzerland

Thomas.Fehlmann@e-p-o.com

Abstract:

The /ch/open Process is a collection of development methodologies and software development best practice. It is available under the GNU Open Documentation licence and has become popular among software developers in Switzerland.

The methodology integrates software metrics, combinatory metrics, the “Six Steps to Completion” metric for project tracking, and Design for Six Sigma. This framework is particularly useful for agile software development, where metrics are key for successful communication within the development group and between developers, users and sponsors.

This paper summarizes experiences with metrics in collaborative software development environments.

Keywords

eXtreme Programming, Agile Methodologies, Iterative–Incremental Development, Combinatory Metrics, Six Sigma for Software, Design for Six Sigma, Six Steps to Completion, Quality Function Deployment.

Zusammenfassung:

Der /ch/open – Prozess ist eine Sammlung von bewährten Vorgehensweisen in der Software–Entwicklung. Diese werden unter der GNU Open Dokumentation Lizenz angeboten und erfreuen sich in der Schweiz zunehmender Beliebtheit.

Die Methodologie integriert Software Metriken, Kombinatorische Metriken, die „Six Steps for Completion“ – Metrik für Projekt–Fortschrittsverfolgung, sowie Design for Six Sigma. Es hat sich gezeigt, dass diese Metriken sich für agile Entwicklungsmethodiken ausserordentlich gut eignen, denn sie erleichtern die Kommunikation innerhalb des Entwicklerteams wie auch zwischen Entwicklern, Benutzern und Sponsoren.

Dieser Artikel fasst Erfahrungen mit Metriken in kollaborativen Software–Entwicklungsumgebungen zusammen.

Für die zur Begutachtung eingereichte Kurzfassung eines Beitrages ist diese Formatvorlage nicht vorgeschrieben, darf aber natürlich verwendet werden.>>>

Schlüsselbegriffe

eXtreme Programmierung, Agile Vorgehensweisen, Iterativ–Inkrementelles Vorgehen, Kombinatorische Metriken, Six Sigma für Software, Design for Six Sigma, Six Steps to Completion, Quality Function Deployment.

1 Introduction

Although there is much talk about agile software development, little is known about what impact metrics have in cooperative software development. We may expect that agile development groups measure themselves and use metrics to optimize their development practices.

1.1 Paradigm Shift in Software Development Best Practices

This is a major shift away from hierarchical development models, so-called waterfall-models, characterized by the development of requirements, specifications, designs, implementations, test specification and execution. These models suffer from the difficulties to define the details early enough in the development cycle. It means that you have no possibility to predict the outcome of the project, its cost and resource requirements when you already should prepare them. Often enough such projects stuck in the requirements gathering and the specification phases for much longer time than they should. Business has then to wait for results; they are stuck stalled with old and inefficient business processes just because the IT department cannot make the deadline for delivering new functionality when needed. Thus not only among developers, even more for users and customers, cooperative development models are seen as an attractive alternative for various parts of business computing.

1.2 Better Quality

One major benefit of cooperative attempts is better quality. It is commonly recognized that cooperative software development processes produce less error-prone software and possibly also more conservative look and feels than software developed in a strictly hierarchically way.

2 Cooperative Development Processes

2.1 Open Source Projects

Successful examples include open source development. Looking at FOSS (Free and Open Source Software) development platforms such as SourceForge, it appears that such teams are self-motivated and self-organized. They produce significantly better software with less defect density.

Even for embedded software, cooperative processes perform well. Although users are not easily available to guide development teams and influence the development process, quality increases significantly.

Cooperative development is a kind of leadership that evolves in virtual organizations. FOSS development quite often is not done within an established

software development organization. FOSS teams are spread all over the world; membership in the team is governed by the existing team members, and reward for membership is often not in financials but in gaining better insight into the technology that you use anyway for your paid work. Thus it saves you time to contribute to a FOSS project for doing your regular work as a developer.

Cooperative development does not mean: fewer specifications. It means agreed specifications and strict adherence to specifications. You need to persuade a team of peers, not only your boss or the customer to change the specs. Various aspects of process engineering look different in cooperative development processes than in traditional one. It is not less formal.

2.2 The Eclipse experience

However, one significant characteristics of cooperative software development is: they measure. The Eclipse team for instance measures functional size and defect density continuously through the development process. Both measurements are needed to provide release criteria and also to decide whom to give the rights to commit changes or additions to the software. In hierarchical organizations, such criteria are difficult to endorse; this is possibly the reason why open source software is of better quality than proprietary code.

3 Metrics for Cooperative Development Processes

Thus the question is of practical nature: which metrics do we need when we want to use benefits of cooperative software development. Is there a minimum set of metrics?

The reason why cooperative teams measure is: metrics allow for much shorter decision cycles. Agreed metrics in the team shorten the time needed for communications and agreement on the topics under investigation. Whatever metric we propose, it must prove its usefulness for faster and less painful agreement processes in the team.

The following metrics are indispensable for team agreement:

- Customer' Needs analysis, including Kano prioritization
- Quality Function Deployment into solution characteristics and product features (Design for Six Sigma)
- Progress Metrics for project planning and tracking
- Deliverable Sizing
- Metrics for test prioritization and evaluation
- Bug metrics

3.1 Customer Needs Analysis and Kano Prioritization

The standard Voice of the Customer process consists of the Voice of the Customer table for analyzing customer's statements, wishes and eventually complaints, New Lanchester Theory for market analysis, Risk Management, and the Kano inquiry technique. Its combination yields a profile for the customer's needs, or – in case of product development – of market needs. Such a profile is useful to compare importance of one customer's needs topic to another. Thus they are relative metrics.

The Kano method of designing market inquiries distinguishes between linear (stated) requirements, delighters and expectations. It allows distinguishing between those customer's needs that can be communicated and help to persuade customers, and those that require excellence in execution without being particularly helpful in selling the product. Depending on the market, this may yield significant insight for prioritization.

It is an essential need for cooperative software development that all team members agree on customer's needs. Else there will never be the focus needed for success.

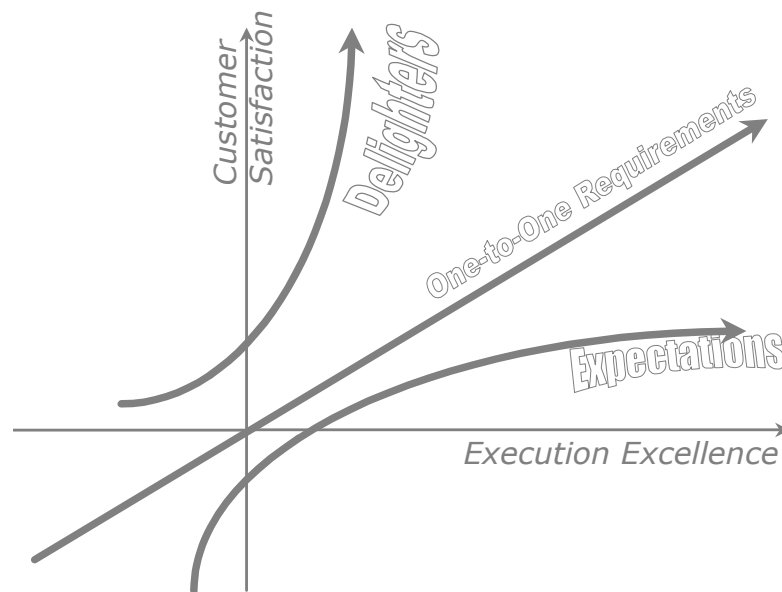


Figure 1: Kano criteria for customer or market inquiries

3.2 Quality Function Deployment into solution characteristics and product features

Design for Six Sigma is – with regard to software development – much in line with comprehensive software development. We deploy customer's needs profile into profiles for solution characteristics, design choice, feature selection, or test

cases. The Six Sigma part is that with the same QFD matrices, combinatory metrics are used to measure how well the delivery processes match the deployment profile.

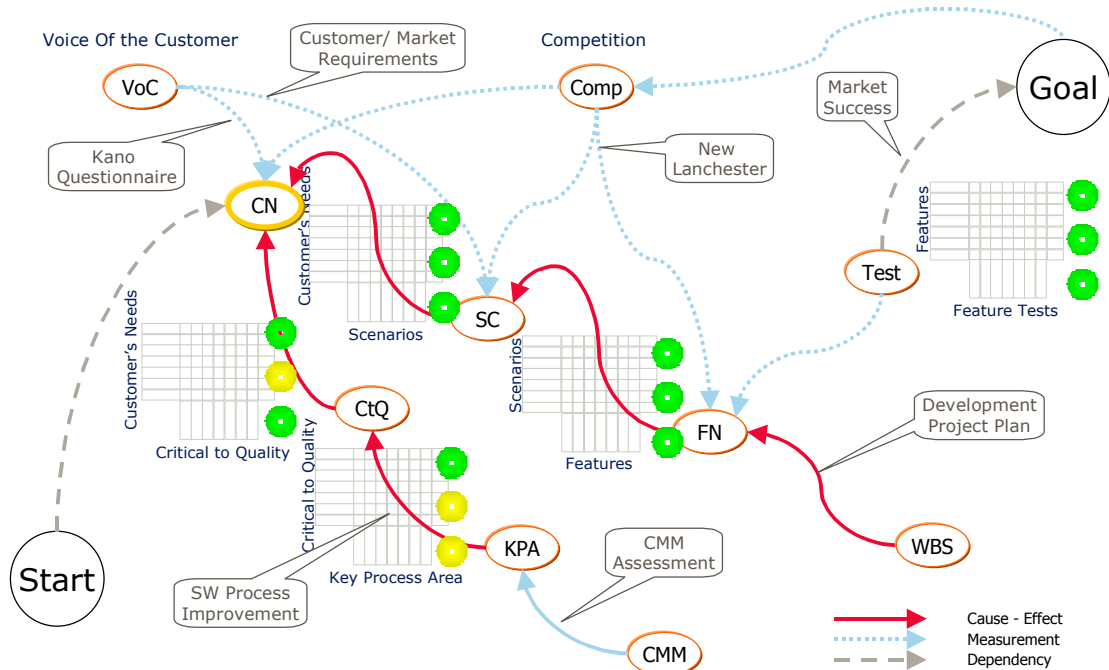


Figure 2: A measurement network for Software Development

Abbreviations used: CN = Customer's Needs; VoC = Voice of the Customer; Comp = Competition; CtQ = Critical to Quality; KPA = Key Process Areas, CMM = Capability Maturity Model; SC = Solution Characteristics; FN = Functionality; WBS = Work Breakdown Structure.

The three signals indicate

- Convergence Factor: This is a metric for the achievement of objectives under ideal conditions;
- Effective Deviation: This is a metric for the actual achievement of objectives;
- Process Maturity Gap: This is a metric for the maturity of your development process.

All three metrics result from Design for Six Sigma and allow the development team to accurately gauge their development effort to market

3.3 Progress Metrics for project planning and tracking

The Six Steps to Completion method measures the developer's progress on his individual task. We use the following model:

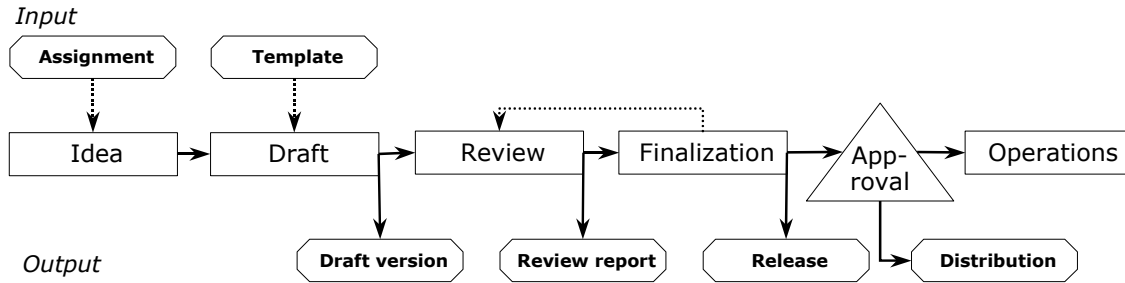


Figure 3: Six Steps to Completion

The model is used to measure actual duration. While effort spent may be significantly higher during drafting of the deliverable, it will decrease for review, finalization and approval. Nevertheless, that time needed is essential for successful project coordination.

Cooperative teams thus use this model to report progress towards the other team members.

3.4 Deliverable Sizing

Here we have the choice between IFPUG 4.2 and COSMIC Full Function Points. Many development teams also use application specific sizing metrics.

3.5 Metrics for test prioritization and evaluation

These metrics derive from the Design for Six Sigma deployment and can be compared with the actually observed bug density for a deliverable. We expect that the Effective Deviation metrics shows a small difference if tests have been focussed on those components that are most important for customer's needs and if bug density is equal among those components. A large effective deviation indicates problems in such area.

4 The /ch/open – Process

The /ch/open Process is an open source project to develop development handbooks, checklists and templates for qualitative software development. It also includes key process metrics and tools for measuring development steps.

4.1 Scoping and Communication

Quality in acquiring knowledge and creating software requires excellent scoping. You cannot describe what you cannot scope. A deliverable is a uniquely defined and validated work result. The project consists of a sequence of deliverables. In the /ch/open process, the scope of the project is described and communicated using individual work tickets. A ticket describes a deliverable of the project. It unites all aspects of a deliverable in software

development: specification of the deliverable, input needed, quality criteria, quality assessment, size, issues, and tests.

Creating such tickets is not an easy task. It means that the team must coordinate itself. The typical roadmap consists of Quality Function Deployment to identify the solution characteristics and features that provide the most value for the customer, and subsequently planning workshops, which identify the tasks and their deliverables. For each deliverable we identify the stakeholders: that might be users of the deliverable, either end-users or other programmers that will later rely on that deliverable for their own tasks. Such users form a review team per deliverable.

4.2 Ticket as Workbench

The ticket is the workbench for the developer. It consists of a form with five different areas for task specification, deliverable identification, effort tracking, issue tracking and test planning and reporting.

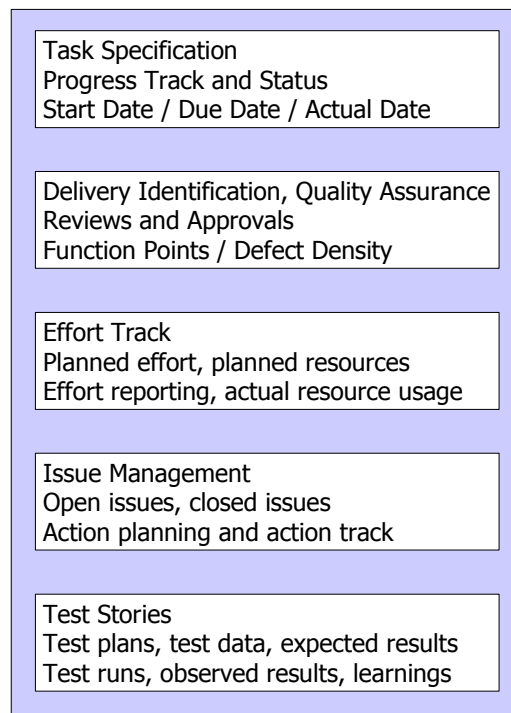


Figure 4: Content of work ticket for the developer

4.3 Implementing the /ch/open – Process

Although the /ch/open Process contains a number of metrics and measurement tools, it is open to accommodate various development methodologies. It works well with the German V-Model as well as the Swiss HERMES II development

process model used in the respective public sectors. Implementing the /ch/open Process always includes adaptation to local processes and organizational responsibilities. The existence of a Project Office is very helpful at least for larger organizations and projects.

The major benefits of using the /ch/open Process stem from the customer-centered metrics. For self-organizing teams this enforces business and customer orientation throughout the development process. The /ch/open Process has been used successfully since 1999 for software development projects in various environments. Experience suggests that its integrated metrics make the difference.

4.4 The Faschina Konzept

The Faschina Group started in 2004 to implement the /ch/open Process together with all metrics mentioned in that paper in the Integrated Software Development Environment (IDE) Eclipse using its Plug-In technology. Software developers thus have access to software metrics and combinatorial metrics with a click of the mouse, the same way they access IDE-ingredients such as configuration management and version control.

References

1. Akao, Y. et.al. (1990), Quality Function Deployment (QFD); Productivity Press
2. Bernet, M. ed. (2002), "The /ch/open/ Software Development Handbook", Excel Templates, AVAILABLE (2004) on www.ch-open.ch/sigs/chop.html
3. Fehlmann, Th. (2000), "Measuring Competitiveness in Service Design", in: QFD Institute (Ed.), 12th Symposium On Quality Function Deployment; Novi, MI
4. Fehlmann, Th. (2001), "Business-oriented testing in E-Commerce", in: Software Quality and Software Testing in Internet Times, ed. Dirk Meyerhoff, SQS AG, Köln
5. Fehlmann, Th. (2002), "Combinatorial Metrics for Software Development", in: 8th International Symposium on QFD, Munich, Germany
6. Fehlmann, Th. (2004), "Six Sigma for Software", in: Proceedings of the 1st SMEF Conference, Rome, Italy
7. Fehlmann, Th. (2003), "Strategic Management by Business Metrics: An Application of Combinatorial Metrics", International Journal of Quality & Reliability Management, Vol. 20 No. 1, Emerald, Bradford UK
8. Humphrey, W.S. (1989), Managing the Software Process, Addison-Wesley
9. 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
10. Taoka N. (1997), Lanchester Strategy – An Introduction, Lanchester Press Inc