Software Controlling mittels Kennzahlen

Ausarbeitung im Rahmen des Proseminars
„IT-Kennzahlen und Softwaremetriken“
im Sommersemester 2010 an der TU München

Timo Besenreuther
besenreu@in.tum.de

Inhalt

Das Projektmanagement in der Software Branche entwickelt sich unter wachsenden Anforderungen stetig weiter. Dazu müssen klassische Vorgehensweisen überdacht und erweitert werden.

Diese Arbeit gibt einen Überblick über die Entwicklung des Projektmanagements hin zum Software Controlling und beschreibt Ansätze und Lösungen zur Unterstützung des Controllings mit Kennzahlen und Kennzahlensystemen.

1. Projektmanagement im Wandel

Software ist überall

Unternehmen verlassen sich immer mehr auf IT-Systeme, die ihre Geschäftsprozesse unterstützen, aber auch im privaten Leben nimmt der Einfluss von Software-Systemen zu. Ob Facebook, das neueste 3D Spiel, oder der Anruf per Skype nach Kanada, wir benutzen Software täglich und mit zunehmender Selbstverständlichkeit.

Die Software Branche schreitet voran

Die Endprodukte der Softwareindustrie werden immer ausgereifter, selbst mobile Geräte bauen heute auf Betriebssystemen auf, die einem PC in wenigem nachstehen. Anwendungen werden als verteilte Systeme entwickelt und Projekte werden von verteilten Teams bearbeitet, in denen sich die Rollen zunehmend spezialisieren. Außerdem wird der Termindruck immer größer, wobei von Anwendern immer kürzere Release-Zyklen gewünscht werden.

Softwareentwicklung muss verlässlich sein

Das klassische Verständnis von IT-Projektmanagement betrachtet hauptsächlich die Einhaltung von Terminen und Budget, sowie die Erfüllung von Anforderungen. Dabei wird meist lediglich das Endprodukt geprüft, das dem Kunden geliefert wird.

Um den oben angedeuteten wachsenden Anforderungen gerecht werden zu können, muss dieses Verständnis allerdings erweitert werden. Die Prozesse zur Softwareentwicklung müssen verlässlich, das Projekt vorausschauend geplant sein.

Dazu reicht es nicht aus, die Qualität des Endprodukts zu prüfen. Es müssen über den gesamten Projektverlauf hinweg Kontrollen auf den einzelnen Komponenten des Systems durchgeführt werden. Nur so kann gewährleistet werden, dass nach dem Zusammenfügen der Komponenten die Qualität des Gesamtsystems auf einem hohen Niveau ist.
Auch im Hinblick auf die Messpunkte von Qualität muss das bisherige Verständnis in Frage gestellt werden. Klassische Tests wie Komponenten- oder Integrationstests, die erst am Ende einer Phase durchgeführt werden, bergen beispielsweise die Gefahr, dass Fehler im Entwurf erst nach der Realisierung erkannt werden. Um dem vorzubeugen sollten auch die Anforderungsanalyse, das Lastenheft, sowie Gesamt- und Komponentenentwürfe auf Vollständigkeit, Korrektheit und Qualität überprüft werden, bevor die Entwicklung in die nächste Phase voranschreitet.

Weiterhin greift es zu kurz, nur Fertigstellungstermine zu überwachen. Bereits auf niedriger Ebene muss auf die Einhaltung von zeitlichen Vorgaben geachtet werden, um frühzeitig die Teile des Projekts zu erkennen, die ihre Termine nicht einhalten können und damit den Verlauf des gesamten Projekts gefährden. Denn nur wenn Risiken frühzeitig erkannt werden, kann reagiert und Einfluss genommen werden.

1.1 Software Controlling

Diese Art der Herangehensweise ähnelt der eines Controllers, der im klassischen Verständnis hauptsächlich an finanzieller Budgeteinhaltung interessiert ist. Darauf aufbauend wurde der Begriff des Software Controllings geprägt, der ein funktions- und bereichsübergreifendes Koordinationssystem beschreibt, das den gesamten Verlauf des Projekts überwacht wird und die Möglichkeit bietet, möglichst früh auf Risiken reagiert zu können, um das Projekt zum Erfolg zu führen.

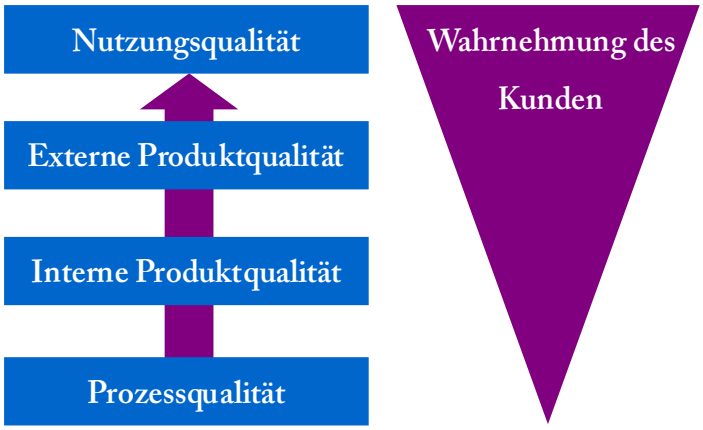
Von Kargl wird Software Controlling nahe am allgemeinen Controlling Ansatz beschrieben: *„Ziel ist die Unterstützung, Informationsversorgung und Koordination von Zielfindung, Planung und Überwachung des Projekts.“* [Kargl99].

Kütz geht etwas weiter und definiert als Ziele des Software Controllings die *„Wirtschaftlichkeit und Effektivität der Planung, Steuerung und Kontrolle aller IT-Prozesse, deren Ressourcen und der Infrastruktur in der Organisation.“* [Kütz05].

Ein weiteres wichtiges Ziel des Software Controllings ist es, Transparenz über den Zustand der Software und des Projekts zu geben. Der Verantwortliche soll sich zu jeder Zeit ein umfassendes Bild des aktuellen Projektstandes machen können und auf dieser Grundlage fundierte Entscheidungen treffen können.

Außerdem wird das Qualitätsbewusstsein aller Beteiligten gestärkt, weil vom Management mehr auf Qualität und nicht nur Funktionalität geachtet wird. Diese Mentalität gepaart mit Ergebnissen wie der Einhaltung von Plänen stärkt das Vertrauen beim Kunden, und kann so Wettbewerbsvorteile gegenüber anderen Unternehmen schaffen, die kein effektives Software Controlling betreiben.

1.2 Qualität, ihre Sichtweisen und passende Metriken



Die Qualität einer Software lässt sich in vier Ebenen untergliedern: Prozess-qualität, interne Produktqualität, externe Produktqualität und Nutzungsqualität [Bennicke08]. Auf jeder dieser Ebenen können Metriken ein Indikator für das Qualitätsniveau sein.

Eine Metrik ist definiert als eine Abbildung von einer Eigenschaft eines Objekts (in diesem Fall ein Qualitätsaspekt des Produkts, bzw. des Prozesses) auf einen konkreten Zahlenwert. Damit können verschiedene Werte verglichen werden, die Ergebnisse können auf Komponenten heruntergebrochen werden, und man kann sich einen Überblick über die Entwicklung des Wertes über die Zeit verschaffen.

Im Folgenden betrachten wir die vier Ebenen der Qualität und betrachten dazu passende Metriken.

Nutzungsqualität

Hier liegt der Fokus des Kunden. Die Nutzungsqualität ist der durch den Kontext bestimmten Nutzen, den ihm das Produkt bringt, also genau der Grund, warum der Kunde das Projekt in Auftrag gegeben hat.

Diese Ebene ist zu weich, um automatisiert gemessen zu werden, dennoch sind Metriken denkbar: Man könnte z.B. einen Fragebogen entwerfen, in dem der Kunde verschiedene Eigenschaften des Produkts mit Zahlenwerten belegt.

Externe Produktqualität

Diese Ebene beeinflusst die Nutzungsqualität am stärksten, sie enthält Eigenschaften der Software, die für den Benutzer sichtbar sind. Beispiele dafür sind Faktoren wie Funktionalität, Zuverlässigkeit und Effizienz.

Metriken dieser Ebene sind z.B. die Anzahl von Bugreports oder die Ergebnisse von Unit Tests, um die Funktionalität des Systems zu messen. Diese Werte können zum Großteil automatisch erhoben werden, was einer manuellen Erhebung generell vorzuziehen ist, da automatisierte Messungen öfter, konstanter und mit weniger Zeitaufwand durchgeführt werden können.

Interne Produktqualität

Die interne Qualität betrachtet nicht direkt die Anforderungen, sondern die Qualität des Systems an sich. Eigenschaften dieser Ebene sind beispielsweise Wartbarkeit oder Portierbarkeit.
Diese Ebene ist nicht weniger wichtig als die vom Kunden wahrgenommenen darüberliegenden Ebenen, im Gegenteil: Die interne Qualität beeinflusst die höheren Ebenen stark und muss deshalb

mindestens genauso beachtet werden.

Metriken hierfür können beispielsweise der Kommentaranteil oder die Anzahl der Verletzungen des Architekturentwurfs sein.

Prozessqualität

Die unterste Ebene des Modells ist die Prozessqualität, welche sich teilweise nicht einmal direkt auf das Projekt bezieht, sondern auf die Prozesse der Entwickler. Darunter fällt unter anderem die Flexibilität der Prozesse, also wie schnell auf externe Einflüsse reagiert werden kann.
Ein weiterer Faktor der Prozessqualität ist der Umfang von durchgeführten Code-Reviews, was mittels einer Metrik gemessen werden kann. Das Resultat dieser Reviews fällt wiederum eher unter die interne Produktqualität.

1.3 Qualitätsziele

Das klassische Projektmanagement hat eine Optimierung der externen Qualität zum Ziel. Software Controller haben allerdings verstanden, dass die externe Qualität von der internen Qualität beeinflusst wird, welche wiederum von der Prozessqualität abhängt.

Dabei ist die negative Beeinflussung besonders ausgeprägt. Ohne einen effektiven Entwicklungsprozess ist es unmöglich, in einem komplexen Umfeld anspruchsvolle Software erfolgreich zu entwickeln. Auf der anderen Seite kann der Prozess zwar perfekt geplant sein, das Endprodukt aber trotzdem eine geringe Qualität aufweisen, weil an anderer Stelle ein Fehler gemacht wurde.

An diesem Beispiel lässt sich wieder das Ziel des Software Controllings erkennen, Risiken frühzeitig zu identifizieren und auf die Ursachen Einfluss zu nehmen, damit sich das Problem nicht auf weitere Phasen oder Komponenten auswirkt. Dieses ganzheitliche Verständnis ist es, was den Unterschied zwischen Erfolg und Scheitern in der Software Branche machen kann. Das gilt genauso für die Durchführung von Projekten wie für regelmäßige Wartungsarbeiten oder die tägliche Arbeit von IT Abteilungen.

2. Kennzahlensysteme

2.1 Metriken und Kennzahlensysteme

Eine Metrik ist wie oben bereits definiert ein Indikator für nur einen Aspekt der Qualität und hat somit für sich genommen wenig Aussagekraft. Deshalb werden die gewonnenen Kennzahlen zu Kennzahlensystemen zusammengeführt, indem sie gewichtet, interpretiert und an geeigneter Stelle aggregiert werden.

Mit einem Kennzahlensystem ist es möglich, den Projektzustand widerzuspiegeln: Es lässt unter anderem den Fortschritt gegenüber dem Zeitplan oder die Einhaltung von verschiedenen Standards und Abläufen erkennen. Außerdem können Teile des Projekts identifiziert werden, die Risiken oder Effizienzhemmnisse darstellen.

2.2 Goal Question Metric

Goal Question Metric ist eine Sammlung von Ansätzen, Richtlinien und Good Practices zum Erstellen von Kennzahlensystemen, die in der Praxis benutzbar sind.

Grundlegender Gedanke ist das Definieren von Zielen, Fragen und Metriken [Basili84].
Ein Ziel beinhaltet eine oder mehrere Fragen, welchen Metriken zugeordnet werden. Die Ergebnisse der Metriken werden dann zu einer Antwort auf die Frage aggregiert, und die beantworten Fragen geben einen empirisch gestützten Überblick über die (Nicht-)Erreichung des Ziels.

Zum Beispiel könnte ein Ziel sein, Change Requests 10% schneller zu bearbeiten.
Fragen und Metriken dazu sind:

- Wie schnell bearbeiten wir CRs im Moment?
  - Metrik „durchschnittliche Bearbeitungszeit heute“
  - Metrik „durchschnittliche Bearbeitungszeit nach CR Typ gegliedert“
- Wie hat sich die Bearbeitungszeit verändert?
  - Metrik „durchschnittliche Bearbeitungszeit nach Monaten“
  - Metrik „durchschnittliche Bearbeitungszeit nach CR Typ gegliedert“

Aspekte von Zielen

Ein weiterer Teil des Goal Question Metric Ansatzes ist, dass Ziele immer fünf Aspekte haben müssen: Messobjekt, Zweck, Qualitätsfokus, Blickwinkel und Kontext.

Im obigen Beispiel hieße das

Analysiere	die CR Bearbeitung	(Messobjekt)
zum Zwecke	der Verkürzung	(Zweck)
in Bezug auf	die Bearbeitungszeiten	(Qualitätsfokus)
vom Blickwinkel	des Entwicklers	(Blickwinkel)
im Kontext	von Projekt X	(Kontext)

Diese fünf Aspekte sollen den gedanklichen Prozess des Nutzers beim Definieren von Fragen anregen und ihm helfen, Ziele vollständig zu definieren, um später klare Aussagen über die Erfüllung machen zu können.

Implikationen

Kennzahlensysteme, die nach diesem System entworfen wurden, haben klar definierte Ziele, die durch Metriken gemessen werden können. An welchen Stellen Aggregationen sinnvoll sind, wird dadurch angedeutet, welche Metriken gemeinsam zum Beantworten von Fragen genutzt werden. Außerdem gibt es eine klare Struktur, nach der Reports erstellt werden können.

2.3 Kennzahlensysteme und klassische Qualitätsmodelle

Qualitätsmodelle sind mit Kennzahlensystemen eng verwandt. Liggesmeyer definiert: *„Ein Qualitätsmodell beschreibt die kausalen Beziehungen zwischen nicht greifbaren Sichten auf Qualität und greifbaren Softwaremaßen. Es setzt sich aus verbundenen und hierarchisch geordneten Qualitätsaspekten zusammen, die schlussendlich auf Softwaremaße führen.“* [Liggesmeyer09].

Die meisten Qualitätsmodelle erfüllen einen großen Teil dieser Definition, indem sie Aspekte der Qualität in verschiedene Klassen unterteilen und Begriffe sprachlich definieren. Oft sind einige der Faktoren allerdings zu weich, um automatisch gemessen zu werden, wie z.B. die sinnvolle Selbstdokumentation des Codes aus McCalls Qualitätsmodell von 1977. Für die Faktoren, die gemessen werden könnten, werden teilweise keine Metriken angegeben, sondern die Eigenschaften müssen wie im Beispiel von McCall von Hand mit Zahlenwerten belegt werden.

Ein modernes Qualitätsmodell ist beispielsweise die Norm ISO / IEC 9126. Sie ist zwar umfangreicher als das Modell von McCall, hat aber die selben Schwächen. Teilweise sind die Qualitätsaspekte sogar noch abstrakter und passende Metriken fehlen komplett.

Als weiteren Schwachpunkt vieler Qualitätsmodelle ist noch zu erwähnen, dass sie lediglich die komplette Software und nicht Zwischenprodukte oder Prozesse betrachten. Für effektives Software Controlling ist allerdings beides nötig.

3. Das Capgemini sd&m Kennzahlensystem

Die Firma Capgemini sd&m hat ein Kennzahlensystem entwickelt, das auf dem Goal Question Metric Ansatz basiert, allerdings in seiner Struktur nicht sofort den Aufbau nach Zielen, Fragen und Metriken erkennen lässt. Dieses System wurde 2008 in einem Artikel im Informatik Spektrum vorgestellt.

Das Capgemini sd&m Kennzahlensystem ist eine Entwicklung aus der Industrie. Die Firma hat erkannt, dass viele Kennzahlensysteme unzureichend sind und für sich und ihre Kunden bestehende Systeme erweitert und verbunden, um ein umfassendes Kennzahlensystem zu erstellen. Dazu werden die Eigenschaften von Software und Softwareprojekten in sechs Bereiche gegliedert:

- Systemumfang und -komplexität
- Technische Code-Qualität
- Effizienz
- Korrektheit und funktionale Vollständigkeit
- Fortschritt
- Allgemeiner Projektstatus

Systemumfang und -komplexität

Auf höchster Ebene dient dieser Wert der allgemeinen Information über die Größe des Projekts.

Betrachtet man die Größen der einzelnen Komponenten im Vergleich, können andere Messwerte normiert werden: z.B. ist in einer großen Komponente eine höhere Anzahl an Bugreports zu erwarten als in einer kleinen. Außerdem lassen sich Schwächen im Architekturentwurf erkennen, wenn Komponenten sehr unterschiedlichen Umfang haben.

Technische Code-Qualität

Hier wird die Einhaltung von Standards und Vorgaben geprüft. Das kann entweder das Einhalten des Architekturentwurfs und konkreter Coding Guidelines sein, oder es wird etwas weniger scharf das Erfüllen des aktuellen State-of-the-Art geprüft.

Dazu werden bestimmte Guidelines als verpflichtend definiert. Wenn diese nicht eingehalten werden handelt es sich um einen Regelverstoß, der behoben werden muss.
Die Einhaltung der restlichen Guidelines und Good Practices wird zwar überprüft, es handelt sich bei Verstößen allerdings nur noch um Warnungen. Das bedeutet, diese Regeln dürfen zwar in Ausnahmefällen verletzt werden, das Kennzahlensystem gibt allerdings Auskunft über Bereiche mit gehäuften Verletzungen. Die entsprechende Metrik heißt Warnungsdichte.

Effizienz

In diesem Bereich werden Leistungsziele bewertet.

Dazu kann entweder ein Wert allgemein gemessen und beobachtet werden, oder es kann die Erfüllung von konkreten nichtfunktionale Anforderungen überprüft werden.

Weiterhin kann die Erhebung in zwei Klassen unterteilt werden: Das System kann bewusst belastet werden, um zu sehen, wie es sich unter Last verhält (Stresstest), oder man kann im Produktivbetrieb die Leistung überwachen (Performancelog).

Korrektheit und funktionale Vollständigkeit

Ziel dieses Bereiches ist es, die funktionale Qualität der Software zu beurteilen, was unter anderem die Erfüllung von funktionalen Anforderungen enthält.

Gefundene Fehler lassen sich klassifizieren, indem man betrachtet, wie sie entdeckt wurden:
Das kann nämlich entweder in automatischen Unit Tests geschehen, oder die Fehler werden von Menschen gemeldet und in einem Bugtracker abgelegt.

Fortschritt

Ein wichtiger Teil des Capgemini sd&m Kennzahlensystems ist die Überwachung des Fortschritts. Es wird der Status und die Termineinhaltung von Artefakten und Zwischenprodukten genauso überwacht wie der Fortschritt des Gesamtprojekts.

Diese Art von Metriken fehlt in vielen Kennzahlensystemen, weil meist nur das Produkt an sich betrachtet wird. Für eine ganzheitliche Betrachtung ist etwas in dieser Art aber unerlässlich, um sich einen Überblick über die Entwicklungsdynamik und die Qualitätssicherung der verschiedenen Teile des Systems zu verschaffen.

Allgemeiner Projektstatus

Der allgemeine Projektstatus soll die unter Fortschritt erhobenen Informationen vervollständigen und ergänzen. Dazu werden subjektive Aussagen von Mitarbeiter einbezogen, die z.B. über ein Voting im Intranet erhoben werden können. Durch die Einbeziehung von menschlichen Meinungen kann auch das gesamte Kennzahlensystem validiert werden, indem man die Überdeckung der verschiedenen Aussagen prüft.

Mögliche Fragen an Mitarbeiter sind „Wie beurteilen Sie die Qualität in Komponente X?“, „Wird der nächste Meilenstein planmäßig erreicht?“ oder „Wie ist die Stimmung?“.

3.1 Erhebung der Werte

	Metriken	Werkzeuge und Quellen für Zahlenwerte
Systemumfang und -komplexität	SLOC Anzahl Klassen Anzahl Abhängigkeiten	javaNSCC ConQAT
Technische Code-Qualität	Anzahl Regelverstöße Warnungsdichte	SonarJ Findbugs ConQAT
Effizienz	Aufrufhäufigkeit Antwortzeiten	Performancelog Stresstest
Korrektheit und funktionale Vollständig	Fehleranzahl Fehlerdichte Fehlerhafte Testfälle	xUnit Bugzilla
Fortschritt	Bearbeitungsstatus Anzahl offener Issues Testüberdeckung Codereview-Aufwand	Artefaktliste Aktivitätenliste Projektmanagement-Tool
Allgemeiner Projektstatus	Subjektive Aussagen von Mitarbeitern	Voting im Web Fragebögen

Basiert auf [Bennicke08]

4. Verwendung von Kennzahlen

4.1 Unterteilung

Die Verwendung von Kennzahlensystemen lässt sich in zwei Kategorien einteilen:
Zieldefiniert und explorativ.

Zieldefinierte Verwendung

Bei diesem Ansatz geht man von konkreten Fragen zu einem bestimmten Zeitpunkt aus.
Zur Beantwortung werden Kennzahlen gemessen und Reports erstellt.

Zum Beispiel möchte der Verantwortliche wissen, wie die Qualität des nächtlichen Builds war.
Dazu lässt er sich einen nach Komponenten aufgeschlüsselten Report generieren, der ihm anzeigt, wie viele Testfälle fehlgeschlagen sind und wie oft Guidelines verletzt wurden.

Explorative Verwendung

Wird ein Kennzahlensystem explorativ verwendet, so möchte der Verantwortliche einen Überblick über die Entwicklung des Messobjekts, also des Projekts und der Software bekommen. Es geht darum, Trends zu erkennen und die verschiedenen Zeitpunkte und Komponenten zu vergleichen. Dadurch können Auffälligkeiten gefunden werden und Untersuchungen angestoßen werden.

Beispielsweise könnte dem Qualitätsverantwortlichen auffallen, dass Komponente X sehr viele Warnungen aufweist. Daraufhin muss er weitere Untersuchungen anstellen, wie z.B. die Überprüfung der Größe der Komponente. Wäre die Komponente besonders groß, würde das die hohe Anzahl an Warnungen erklären. Je nach dem kann er die Sache auf sich beruhen lassen oder weitere Maßnahmen ergreifen.

4.2 Kennzahlensysteme im Entwicklungsprozess

Projektleitstand

Der Projektleitstand setzt sich aus Reports zusammen, die wieder aus kontinuierlich erhobenen Kennzahlen bestehen. Er soll einen explorativen Überblick über das gesamte Projekt geben und auf Auffälligkeiten hinweisen

Meilensteine

Vor dem Erreichen eines Meilensteins kann zieldefiniert ein Report generiert werden, der darüber Auskunft gibt, ob die Qualität auf einem ausreichenden Niveau ist, um den Meilenstein zu erreichen.

Quality Gates

Ein Quality Gate ist ein Tor zwischen den Phasen des Projekts, das sich erst öffnet, wenn die Qualität mit positivem Ergebnis geprüft wurde. So darf die Umsetzung des Entwurfs beispielsweise erst beginnen, wenn der Entwurf selbst geprüft wurde.

4.3 Kritische Betrachtung von Kennzahlensystemen

Aussagekraft der Kennzahlen

Die Kennzahlen dienen lediglich als Orientierung und sollen nicht als Ziel gesehen werden, gegen das optimiert wird. Denn ein guter Wert einer Metrik bedeutet nicht, dass das Messobjekt tatsächlich in einem guten Zustand ist. Genauso kann es in Ausnahmefällen vorkommen, dass Metriken schlecht Werte aufweisen, obwohl das Objekt eine hohe Qualität aufweist.

Die Werte müssen immer interpretiert und in Kontext gesetzt werden. Das Kennzahlensystem muss als Frühwarnsystem verstanden werden, das den Verantwortlichen auf eventuelle Risiken aufmerksam macht.

Aggregation von Werten

An einigen Stellen machen Aggregationen offensichtlich Sinn, wie z.B. die Größe der Komponenten zur Gesamtgröße des Projekts zusammenzufassen. Oft sind Aggregationen aber enorm trügerisch, da einzelne Werte in der Masse verschwinden können. Außerdem können die einzelnen Zahlen häufig nicht automatisch gewichtet werden, da jede Firma und jedes Projekt individuell ist und dementsprechend die Werte auch unterschiedlich bewertet werden müssen.

Aufwand und Nutzen

Die Entwicklung eines Kennzahlensystems kann sehr aufwändig sein, genauso die Integration in den Entwicklungsprozess. Man darf nicht aus den Augen verlieren, dass ein Kennzahlensystem den Entwicklern dienen muss und nicht umgekehrt. Kennzahlen sind ein ausgezeichnetes Hilfsmittel, um den Überblick über ein Projekt zu behalten, aber leider kein Allheilmittel.

Literaturverzeichnis

[Bennicke08]
Bennicke et al. - Software Controlling
Erschienen im Informatik Spektrum 31.06.2008

[Kütz05]
Martin Kütz – IT Controlling für die Praxis
Erschienen im dpunkt.verlag, 1. Auflage 2005

[Kargl99]
Herbert Kargl – DV Controlling
Erschienen in der Oldenbourg Verlagsgruppe, 4. Auflage 1999

[Liggesmeyer09]
Peter Lieggesmeyer – Softwarequalität: Testen, analysieren und verifizieren von Software
Erschienen in Spektrum Akademischer Verlag Heidelberg, 2. Auflage 2009

[Basili84]
Basili VR, Weiss DM (1984): A methodology for collecting valid software engineering data.
IEEE Transactions on Software Engineering (IEEE Trans Software Eng) -10(6): S. 728–738

Wikipedia: Goal Question Metric, zuletzt aufgerufen am 29.07.2010
de.wikipedia.org/wiki/Goal_Question_Metric