cGQM - Ein zielorientierter Ansatz für kontinuierliche, automatisierte Messzyklen

Christoph Lofi

Collaborative Software Development Laboratory, University of Hawai'i Manoa

christoph.lofi@gmail.com

Zusammenfassung:

Messungen, als Grundlage für einen vorhersagbaren und kontrollierbaren Software Entwicklungsprozess, sind ein wichtiger Teilaspekt des Software Engineering. Das Durchführen von Messungen ist essentiell zum Bewerten des aktuellen Projektstatus, dem Erfassen von „Baselines" (Ausgangsmesswerte) und der der Validation von Verbesserungs- und Kontrollaktionen.

Die in diesem Artikel vorgestellten Arbeiten basieren auf Hackystat [1, 2], einem vollautomatisiertem Messwerkzeug für Software Produkt- und Prozessmetriken. Hackystat wurde um die Fähigkeit der zielgerichteten Messung und Analyse nach dem GQM (Goal, Question, Metric) Paradigma [3, 4] erweitert.

Die so entstehende Messplattform implementiert ein ebenfalls hier vorgestelltes Konzept namens cGQM (continuous GQM), einer Spezialisierung des GQM Paradigmas mit der Einschränkung auf lediglich automatisch erfass- und analysierbare Metriken.

cGQM zusammen mit seiner Referenzimplementierung hackyCGQM stellen einen interessanten, vollautomatisierten Ansatz zur werkzeuggestützten Softwareprojektkontrolle dar.

Die aus diesem Ansatz entstehenden Vor- und Nachteile werden in diesem Artikel vorgestellt und kurz diskutiert.

Schlüsselbegriffe

GQM, automatisiertes Messen, kontinuierliche Messzyklen

1 Einleitung

Ein populäres Zitat besagt „*you can neither predict nor control what you can not measure*" [5]. Ohne Messungen ist es nur schwer möglich den aktuellen Zustand eines Entwicklungsprozesses zu erfassen oder die Effekte von kontrollierenden oder steuernden Maßnahmen zu beurteilen.

Während in der Produktionsumgebungen wie z.B. bei der Massenproduktion von Teilen und Maschinen Mess- und Kontrollsysteme weit verbreitet und etabliert sind [6], so ist dies bei der Softwareentwicklung häufig noch nicht der Fall [7]. Die Hauptgründe hierfür sind: a) Ein Mangel an zur Durchführung von Messprogrammen nötiger Kompetenz, b) ein Mangel an Methoden und Werkzeugen zur praktikablen Unterstützung von Messprogrammen sowie c) der mit der Einrichtung und Durchführung von Messprogrammen verbundene hohe Aufwand.

Dieser Artikel stellt *cGQM* (continuous GQM) vor, eine spezialisierte Variante des GQM Paradigmas [3, 4] sowie die zugehörige Referenzimplementierung *hackyCGQM.* HackyCGQM ist ein auf dem GQM Paradigma basierender messgestützter, zielorientierter Projektleitstand [8] mit dem Ziel, bei den drei oben genannten Gründen für den Nicht-Einsatz von Messprogrammen Linderung zu verschaffen.

Das Kernkonzept von cGQM ist die kontinuierliche, automatisierte Messdatenerfassung und –analyse mit minimalem manuellem Aufwand nach der erstmaligen Implementierung eines Messprogramms.

2 Verwandte Konzepte und Technologien

Dieses Kapitel stellt kurz die Konzepte, Werkzeuge und Methoden vor, die für cGQM sowie hackyCGQM relevant sind. Diese sind:

- GQM als Grundparadigma, welches für cGQM eingeschränkt und spezialisiert wird
- Hackystat, die technische Grundlage von hackyCGQM
- SPCC (Software Project Control Center), ein Grundkonzept für Projektleitstände.

2.1 GQM

Das GQM (Goal / Question / Metric) Paradigma [3, 4] ist eine Methode zur Entwicklung von rationalen, verfolgbaren und effizienten Messprogrammen gemäß einem gegebenen Verbesserungs- oder Strategieziels.

Die Hauptidee hierbei ist, dass Messungen zielorientiert sein sollten. Daher besteht der erste Schritt bei GQM in der Definition von Zielen. Zu die-

sen Zielen werden Fragen entwickelt deren Antwort hilfreiche Informationen zur Erfüllung des Ziels liefern. Danach werden die Metriken bestimmt, die zur Beantwortung der Fragen nötig sind. Dieser „Top-Down" Ansatz gewährleistet, dass nur solche Metriken gemessen werden, die auch zur Erfüllung der Ziele beitragen. Die Wahl der Metriken ist somit direkt rechtfertigbar.

Nachdem Messdaten erfasst wurden wird der aufgestellte GQM Plan (welcher aus den oben genannten Zielen, Fragen und Metriken besteht) nach einem „Bottom-Up" Prinzip interpretiert: Die Messwerte liefern Daten für die geforderten Metriken, welche dann zur Beantwortung der Fragen verwendet werden und diese wiederum dienen letztendlich der Erfüllung der Ziele (siehe Abbildung 1).

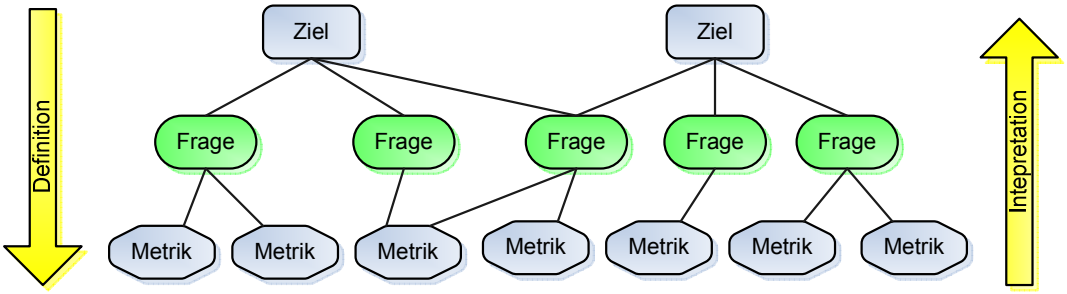

Abbildung 1: GQM Schichten: Ziele werden zu Fragen verfeinert zu welchen wiederum Metriken bestimmt werden. Dabei sind auch Mehrfachassoziationen möglich. Bei der Interpretation der Messdaten wird der Plan „bottom-up" durchlaufen.

Der Ablauf eines GQM Programms erfolgt, wie in Abbildung 2 dargestellt, in den Hauptphasen Planung (Aufstellen des Projektplans), Definition (Aufstellen der Ziele und Fragen, Auswahl der Metriken), Messdatenerfassung und Interpretation.

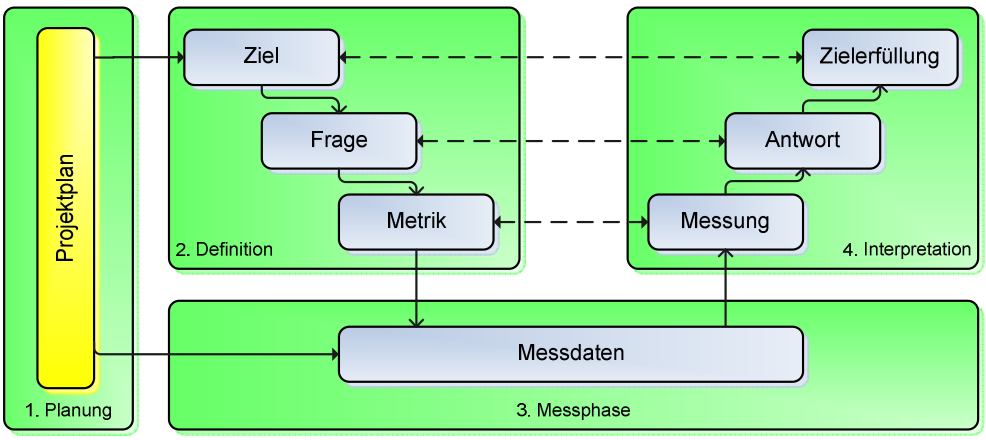

Abbildung 2: Die vier GQM Phasen: Planung, Definition, Messdatenerfassung und Interpretation.

2.2 Hackystat

Hackystat [1, 2] ist ein System zur völlig automatisierten Erfassung von Software Produkt- und Prozessmetriken. Die Entwicklung von cGQM und hackyCGQM basiert auf der ursprünglichen Arbeit an dem Hackystat System, welches zwar ein weites Spektrum an Softwaremetriken erfassen kann, diese aber nicht in einer ziel-orientierten und zur Projektkontrolle geeigneten Form analysiert und repräsentiert. cGQM stellt somit die Verbindung zwischen dem GQM Paradigma und vollautomatisierten Messsystem wie z.B. Hackystat dar.

Hackystat ist als ein Client-Server System mit einem zentralem Mess- und Analyse Server realisiert. Dieser erhält Messdaten von verschiedenen, automatischen Sensoren, welche an Software, die am Entwicklungsprozess beteiligt ist, angekoppelt wird. So können z.B. die Entwicklungsumgebungen auf den Rechnern der einzelnen Programmierer (wie z.B. Eclipse) mit Sensoren ausgestattet werden, die dann völlig autonom Prozessmetriken (z.B. aktive Programmierzeit) oder Produktmetriken (z.B. Größe der bearbeiteten Dateien) an den Server übermitteln. Diese Messdaten können dann entweder über ein Web-Interface analysiert werden oder als Zusammenfassung den Entwicklern per Email zugeschickt werden. Dies ist in Abbildung 3 dargestellt.

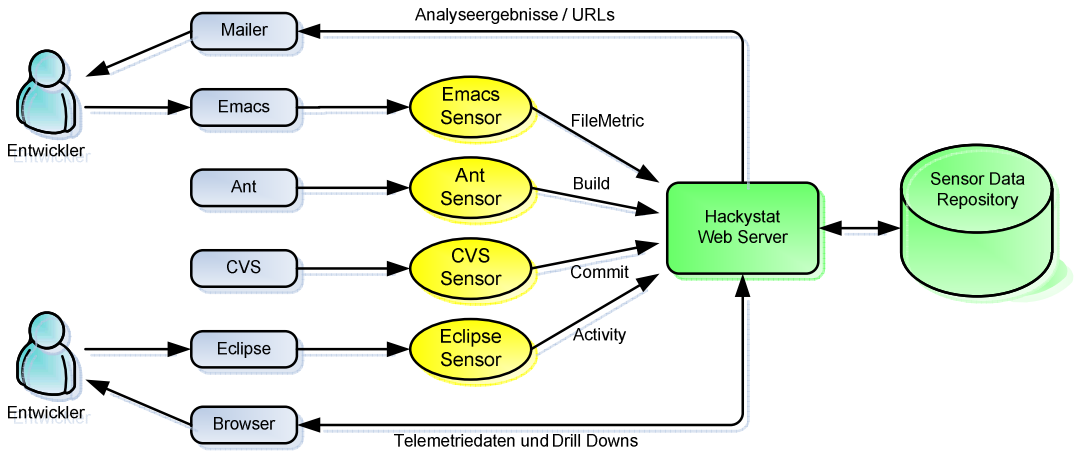

Abbildung 3: Hackystat: Verschiedene Entwicklungswerkzeuge werden mit Sensoren ausgestattet, die Messungen zu bestimmten Metriken an den Hackystat-Server senden.

Zurzeit (September 2005) unterstützt Hackystat 25 Sensoren für populäre Software, die bei Softwareentwicklung eingesetzt wird (z.B. Eclipse, Visual Studio, Ant, CVS, Subversion, JUnit, etc). Diese liefern Messergebisse für eine Vielzahl an Metriken wie z.B. verschiedene Aufwands-, Datei- oder Versionierungsmetriken.

2.3 SPCC

SPCC's (Software Project Control Center) [12] sind Plattformen, um Softwareprojekte anhand von Messdaten zu steuern und zu kontrollieren. Im SPCC Prozessmodell dient das Software-Kontrollzentrum als zentraler Anlaufpunkt zum Visualisieren von Daten und dem Bereitstellen von Feedback aus dem Entwicklungsprozess. Auch das später vorgestellte hackyCGQM kann als Implementierung eines SPCC's angesehen werden.

3 cGQM: Kontinuierliches Messen und Analysieren

Das klassische GQM Paradigma dient dazu, für ein gegebenes Problem oder Umgebung passende Metriken und deren Analysen auszuwählen, die hilfreiche und nützliche Informationen zum Erfüllen eines Zieles liefern. Im Folgenden wird cGQM, ein auf GQM basierender Ansatz vorgestellt.

cGQM greift Konzepte von GQM, wie etwa das Aufstellen von Zielen und Fragen, auf und schränkt diese ein. Im Gegensatz zu GQM, das die eigentliche Implementierung des Messprogramms offen lässt, fordert cGQM völlig automatisierte Mess- und Analysezyklen. Dies hat einige positive Effekte:

- „Kontinuierliche" Analyse-Ergebnisse stehen zur Verfügung. Da Messungen und deren Analyse automatisch verlaufen können beliebig kurze Messzyklen durchgeführt werden, die jederzeit aktuelle Daten liefern können.
- Die Durchführung eines Messzyklus ist „billig" da kein manueller Aufwand erforderlich ist. Daher können auch lang laufende Messprogramme oder solche mit einer hohen Anzahl an Messzyklen ohne Rücksicht auf die anfallenden Messkosten durchgeführt werden. Ein weiterer Vorteil davon ist auch, dass auch eigentlich nicht benötigte Metriken billig erfasst werden können (entgegen dem GQM Para-

digma, das die Beschränkung auf direkt nützliche Metriken zum Ziel hat) die später für Anpassungen und Erweiterungen des Messprogrammes hinzugezogen werden können.

Um automatische Mess- und Analysezyklen realisieren zu können müssen zwei Vorraussetzungen erfüllt sein:

- Am Messprogramm beteiligte Metriken müssen automatisch erfassbar sein. Das heißt, dass Daten von einem autonomen Messsystem („Sensor") erfassbar sein müssen ohne dass dabei Interaktion seitens eines Benutzers erforderlich ist. Dadurch können viele Metriken nicht verwendet werden, vor allem solche, die auf der subjektiven Einschätzung von Experten basieren oder sich auf zu messende Objekte beziehen, die nicht mit Sensoren erfasst werden können. Als Beispiel soll hier die Metrik „Projektaufwand in Stunden" dienen. So bietet Hackystat in diesem Fall eine Näherungsmetrik namens „activetime", die mit Hilfe von in Entwicklungswerkzeugen integrierten Sensoren (z.B. Eclipse, Emacs, Visual Studio, MS Office, etc.) die Zeit erfassen kann, in der ein Entwickler aktiv programmiert bzw. dokumentiert hat. Allerdings lässt sich auf diese Weise nicht erfassen, wie viel Aufwand im Projekt tatsächlich angefallen ist, da zum Beispiel Zeiten für Besprechungen oder andere, nicht mit Softwarewerkzeugen gestützten Aktivitäten nur manuell erfasst werden können. Dies stellt eine gravierende Einschränkung bei dem Entwurf von Messprogrammen dar, da nur ca. 45% der in manuell durchgeführten Messprogrammen verwendeten Metriken automatisiert erfassbar sind (siehe Kapitel 5).
- Die bei der Definition des Messprogramms aufgestellten Fragen müssen automatisch beantwortbar sein. Dies bedeutet, es muss ein Algorithmus aufgestellt werden können, der in der Lage ist, aus den erfassten Messdaten die Antworten auf die im GQM Plan enthaltenen Fragen zu liefern. Zusätzlich kann analog ein „Zielerfüllungsgrad" für die aufgestellten Ziele errechnet werden, der, basierend auf den Antworten der Fragen und den Messwerten kurz darstellt, in wiefern das aktuelle Ziel erfüllt ist. Der Vorteil hierbei ist, dass somit eine knappe und schnell überschaubare Zusammenfassung über den aktuellen Verlauf des Messprograms gegeben ist. Allerdings ist das Aufstellen von Algorithmen, die einen Erfüllungsgrad erzeugen in den meisten Fällen schwierig bzw. nicht möglich. Daher sind Er-

füllungsgrade in cGQM optional. Zur Errechnung von Erfüllungsgrade eignen sich besonders Ziele mit überprüfbaren Zielkriterien (z.B. „reduziere die Anzahl der Integrationsfehler um 10%"), wie z.B. in Abbildung 6 dargestellt.

Die durch automatische Messungen und Analysen gewonnenen Möglichkeiten zur Durchführung von billigen und kurzen Mess- und Analysezyklen vereinfacht besonders die Durchführung von agilen und evolutionären Messprogrammen. Als Beispiel werden im Folgenden drei mögliche Benutzungsmodelle vorgestellt, welche unter Anderem auch gemischt werden können: a) kontinuierliche Analyse, b) retrospektive Analyse und c) alarmbasierte Analyse.

Bei diesen Benutzungsmodellen werden die vier in Kapitel 2.1 vorgestellten Phasen von GQM Messprogramm in unterschiedlicher Reihenfolge ausgeführt.

Die *kontinuierliche* Analyse eignet sich besonders zur Prozesskontrolle. Dabei können kurze (z.B. 1 Tag oder kürzer) Feedbackzyklen eingesetzt werden. Dies erlaubt das kurzfristige Reagieren auf auftretende Ereignisse sowie die Überprüfung von gegensteuernden Maßnahmen (siehe Abbildung 4).

Der Ansatz der kontinuierlichen Analyse kann zur *alarmbasierten* Analyse ausgebaut werden. Hierbei werden ebenfalls kurze Feedbackzyklen eingesetzt. Zusätzlich werden Alarmkriterien definiert, die unerwünschte Verläufe von Mess- und Analyseergebnissen darstellen. Die automatischen Messungen können daraufhin stetig mit den Alarmkriterien verglichen. Bei einer Verletzung der Kriterien wird ein „Alarm" ausgelöst und die Mess- und Analysedaten dem Projektverantwortlichen zugesendet. Bei diesem Ansatz kann das Messprogramm nach Initialisierung völlig unbeaufsichtigt im Hintergrund ablaufen – keinerlei manuelle Interaktion ist vonnöten so lange das unter Beobachtung stehende Projekt die Alarmkriterien nicht verletzt (siehe Abbildung 5).

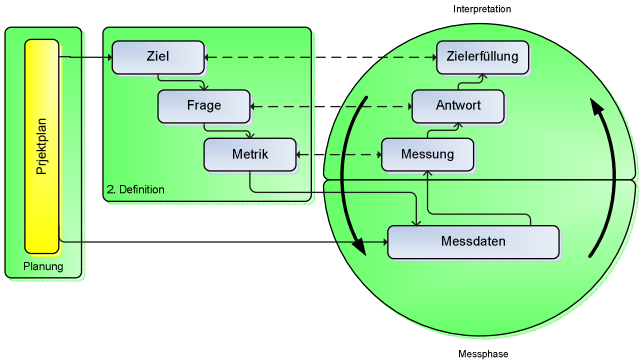

Abbildung 4: Kurze Mess- und Analysezyklen vereinfachen die Projektkontrolle

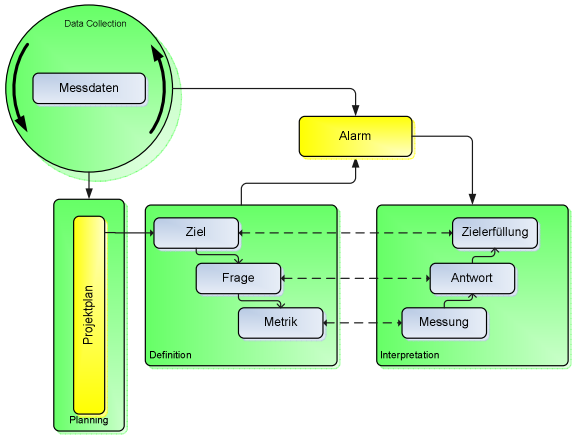

Abbildung 5: Alarmbasierte Analyse: Analyseergebnisse werden nach Verletzung von Alarmkriterien automatisch verschickt

Ein anderer, möglicher Ansatz ist das *retrospektive* Analysieren der Messdaten (siehe Abbildung 6). Dies ist sinnvoll, wenn zu Projektbeginn noch kein Bedarf für ein zielgerichtetes Messprogramm besteht, allerdings ein solches im Projektverlauf zu erwarten ist. Hierbei werden ab Projektbeginn zu allen automatischen erfassbaren Metriken auch Messdaten erfasst. Dies verursacht nach einer Einrichtungsphase aufgrund der Vollautomatisierung der Messungen keine weiteren Kosten und ist unabhängig von Dauer der Messperiode sowie Anzahl der Metriken. Sobald ein zielgerichtetes Messprogramm benötigt wird, kann dieses eingerichtet werden (Aufstellen der Ziele und Fragen und Auswahl von Metriken) und direkt auf die bereits in der Vergangenheit erfassten Messdaten zugreifen.

Die Stärke derartiger Benutungsmodelle kommt besonders in Situationen zu tragen, in denen Anpassungen an dem ursprünglichen Messprogramm vonnöten sind. So kann zum Beispiel während der Durchführung eines kontinuierlichen Messprogramms ein neues Problem entdeckt werden, welches genauerer Analyse zur Lösung bedarf. Neu aufgeworfene Fragen können während dem Verlauf des Messprogrammes integriert werden und dann retrospektiv auf die bereits gesammelten Daten zugreifen. So lassen sich nachträglich vergangene Zeitspannen bis zum Auftreten des Problems gezielt analysieren.

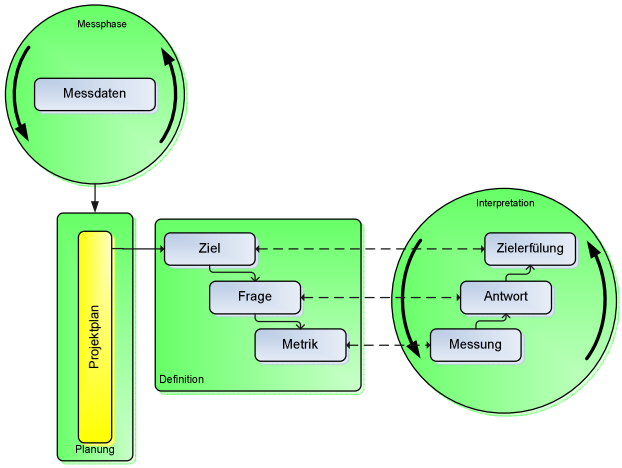

Abbildung 6: Retrospektive Analyse: Alle erfassbaren Metriken werden erfasst und stehen später aufgestellten Messprogrammen retrospektiv zur Verfügung

4 HackyCGQM

HackyCGQM ist die Referenzimplementierung eines cGQM Systems. Es wurde in das automatische Messsystem Hackystat [1, 2] integriert und dient der Evaluation von verschiedenen Aspekten der zielgerichteten, voll-automatischen Softwaremetrikmessung und –analyse.

Benutzer können eigene Analyse-Module („Plugins") entwickeln, welche einen cGQM Plan enthalten. Ein solcher besteht aus der Beschreibung von Zielen, Fragen, Metriken sowie die Codefragmenten, die für deren Ausführung (auf Messdaten zugreifen, Antworten erzeugen, Erfüllungsgrade errechnen) verantwortlich sind („Executables").

Auf die Implementierung wird hier nicht näher eingegangen, Abbildungen 7-9 stellen exemplarisch den Umgang mit dem System dar.

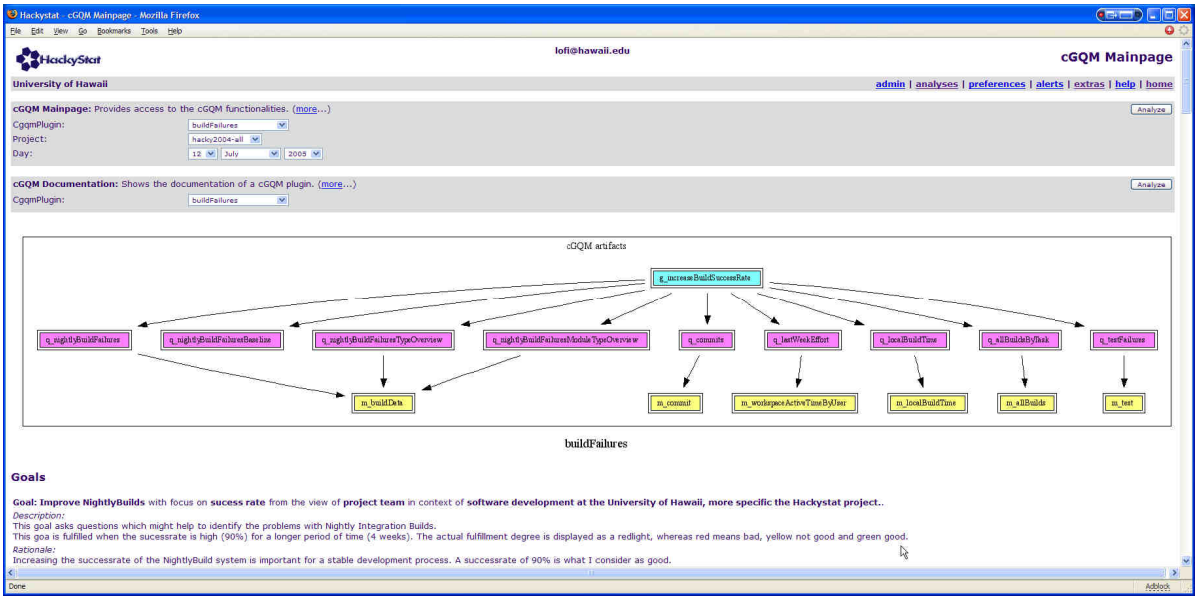

Abbildung 7: HackyCGQM Webinterface: Startseite einer cGQM Analyse mit graphischem cGQM Plan und der Erläuterung der Ziele

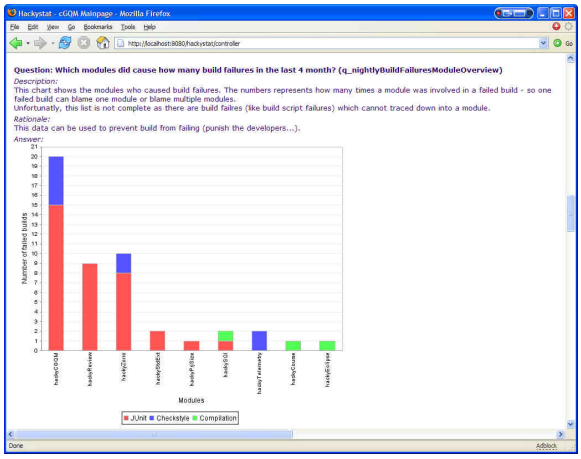

Abbildung 8: HackyCGQM Webinterface: Darstellung der einer Frage und der dazugehörigen Antwort

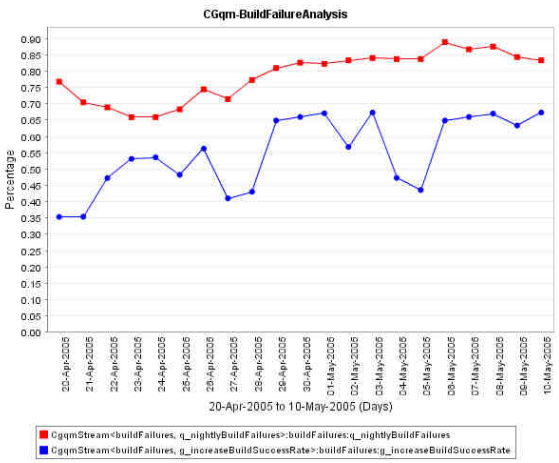

Abbildung 9: Verlauf von „Zielerfülungsgraden" über eine längere Zeitspanne

5 Evaluation

Im Laufe der Entwicklung von HackyCGQM wurden mehrere Evaluationen durchgeführt, darunter eine erste Bestimmung von zu erwartenden Aufwänden bei der Implementierung und Durchführung von cGQM Messpro-

grammen und eine Benutzerevaluation mit hackyCGQM in einem realen Einsatzszenario.

In diesem Abschnitt wird eine strukturelle Evaluation vorgestellt, welche versucht, den möglichen Einsatzbereich von cGQM Messprogrammen zu charakterisieren. Da cGQM sich lediglich auf automatisch erfass- und analysierbare Metriken beschränkt ist der zu erwartende Einsatzbereich nicht so groß wie bei generischen GQM Ansätzen.

In dieser Untersuchung wurden sechs publizierte GQM Fallstudien als Referenz darauf untersucht, ob sie sich mit der aktuellen bzw. mit zukünftigen cGQM Implementierung hätten durchführen lassen können. Es wurden die Studien von Solingen und Berghout (Case A-D) [9], Fugatta [10] und Lindström [11] verwendet. Obwohl die statistische Relevanz dieser Auswahl nicht gewährleistet ist, sind die einzelnen Fallstudien doch unterschiedlich genug, um ein breiteres Spektrum an GQM Einsatzmöglichkeiten abzudecken und dem Ergebnis der durchgeführten Vergleichsevaluation eine ausreichende Signifikanz zu verleihen.

Im Laufe der Evaluation wurden 204 Metriken, 119 Fragen und 16 Ziele untersucht. Dabei stellte sich heraus, dass für 40% der 339 GQM Planartefakten die aktuelle hackyCGQM Implementierung hätte verwendet werden können. Interessanterweise lässt sich dieser Wert durch Verbesserung der Implementierung lediglich auf 45% erhöhen, der restliche Anteil von 55% der GQM Artefakte basiert auf Metriken und Messungen, welche nicht automatisierbar sind (z.B. Expertenurteile).

6 Zusammenfassung

Mit hackyCGQM wurde ein vollautomatischer Mess- und Analyseframework entworfen, der zielgerichtete Messprogramme nach dem GQM Paradigma unterstützt.

Obwohl aufgrund der Beschränkung auf automatisch erfassbare Metriken nicht in jeder Situation einsetzbar, so bietet der Ansatz doch viele Vorteile und ein großes Potential: Lediglich beim Einrichten des Messprogramms fallen Kosten an, die Durchführung der Mess- und Analysezyklen sind weitestgehend kostenfrei. Dies erlaubt kontinuierliche Messungen und Analysen über längere Zeiträume ohne dabei Kostenprobleme zu erzeugen. Des Weiteren bieten sich interessante Prozessmodelle zum Umgang mit den

Messprogrammen an wie evolutionäre oder retrospektive Vorgehensweisen.

In späteren Evaluationen stellte sich heraus, dass ca. 40% von publizierten GQM Programmen automatisierbar sind.

Danksagungen

Ich danke Prof. Dr. Philip Johnson sowie den übrigen Mitarbeitern des Collaborative Software Development Laboratory der Universität Hawai'i für die Unterstützung meiner Arbeit.

Literaturhinweise

1. P.M. Johnson. You can't even ask them to push a button: Toward ubiquitous, developer-centric, empirical software engineering. The NSF Workshop for New Visions for Software Design and Productivity: Research and Applications, Nashville, TN, December 2001.
2. Hackystat Development Website, (http://www.hackystat.org/)
3. V.R. Basili. Software modeling and measurement: The goal question metric paradigm. Technical Report CS-TR-2956, University of Maryland, College Park, 1992.
4. D.M. Weiss V.R. Basili. A methodology for collecting valid software engineering data. IEEE Transactions on Software Engineering, SE-10(6):728–738, November 1984.
5. T., DeMarco. Controlling Software Projects. Yourdon Press, New York,, 1982.
6. R.S. Kaplan. Measures for manufacturing excellence. Harvard Business, School Press, 1990.
7. W. C. Peterson. SEIs software process program - presentation to the board of visitors. Technical report, Software Engineering Institute, Carnegie Mellon University, 1997.
8. J. Heidrich J. Muench. Software project control centers: Concepts and approaches. Technical report, Fraunhofer Institute of Experimental Software Engineering, 2003.
9. E. Berghout R. van Solingen. The Goal/Question/Metric Method: A practical guide for quality improvement of software deleopment. McGraw-Hill Int., London, 1999.
10. A. Fuggetta, L. Lavazza, and S. Morasca. Applying GQM in an industrial software factory. ACM Transactions on Software Engineering and Methology,

7(4):441–448, October 1998.

11. B. Linström. A software measurement case study using GQM. Technical Report LUTEDX(TETS-5522)1-72(2004), Lund Institute of Technology, 2004.
12. J. Heidrich J. Muench. Software project control centers: Concepts and approaches. Technical report, Fraunhofer Institute of Experimental Software Engineering, 2003.