Nachhaltiges Software Management durch Lebenszyklus-übergreifende Überwachung von Qualitätskennzahlen

Sandra M. Lang, Bernhard Peischl[1]

Softnet Austria
Inffeldgasse 16b/II, 8010 Graz
bernhard.peischl@soft-net.at
sandra.lang@soft-net.at

Abstract: In diesem Artikel motivieren wir den Begriff der Nachhaltigkeit und argumentieren, dass zur Bewertung der Nachhaltigkeit einer Software der gesamte Lebenszyklus betrachtet werden sollte. In diesem Kontext ist es notwendig, Kennzahlen aus verschiedenen Bereichen (Ressourcen, Produkt, Prozess) zu operationalisieren, um Auswirkungen auf Qualität und Nachhaltigkeit quantitativ erfassbar zu machen. Wir gehen in diesem Artikel auf unseren Cockpit Ansatz ein und präsentieren eine Fallstudie aus dem Vorgangsmanagement eines Automobilzulieferers. Daran sieht man, dass für praktische Fragestellungen, die Operationalisierung von Kennzahlen ohne flexible Messinfrastruktur nicht zu bewerkstelligen ist und ein Software Cockpit hier wesentliche Unterstützung leisten kann. Dies gilt im Speziellen für die vernetzten Fragestellungen im Umfeld des nachhaltigen Software Managements.

1 Motivation

Nachhaltigkeit gewinnt in vielen Bereichen der Wirtschaft und des gesellschaftlichen Lebens an Bedeutung. Im ursprünglichen Wortsinn („längere Zeit anhaltende Wirkung“) [Du01] entstammt das Wort von „nachhalten“, mit der Bedeutung „längere Zeit andauern oder bleiben“. Im modernen Sinn kommt der Gedanke hinzu, dass etwas andauern, bleiben, nachwirken oder haltbar sein kann noch lange nachdem es gebaut oder in Bewegung gesetzt wurde. Auf das Management von Software übertragen meint der Begriff eine langlebige, die Ressourcen schonende Nutzbarkeit. Ob und vor allem wie nachhaltig eine Software entwickelt wurde, zeigt sich allerdings erst bei Betrachtung des gesamten Lebenszyklus der Software – Entwurf, Umsetzung, Betrieb, inkrementelle Verbesserung und Wartung müssen in die Nachhaltigkeitsbetrachtung einfließen.

[1] Die Autoren sind in alphabethischer Reihenfolge gelistet. Diese Arbeiten wurden teilweise im Rahmen des Kompetenznetzwerkes Softnet Austria II (www.soft-net.at, COMET Programm der FFG) durchgeführt und vom Bundesministerium für Wirtschaft, Familie und Jugend (bmwfj), dem Land Steiermark, der Steirischen Wirtschaftsförderungsgesellschaft mbH (SFG), und der Stadt Wien durch das Zentrum für Innovation und Technologie (ZIT) unterstützt.

Die hohe IT-Durchdringung von Unternehmen, Behörden und dem Privatbereich impliziert auch, dass Entscheidungen im Software Management nicht alleine (oft quantifizierbare) technische Aspekte beeinflussen, sondern viel mehr auch (kaum oder nur sehr schwer messbare) die Ressourcen betreffende, ökonomische und soziale Auswirkungen haben. Diese Aspekte müssen in die Nachhaltigkeitsbetrachtung einfließen. So gehen z.B. durch Offshoring und Cloud Computing Arbeitsplätze an Billiglohnländer. Gerade in Zeiten von globalen geopolitischen Veränderungen (Währungsturbulenzen, Ressourcen-Knappheit, Herausbildung einer multipolaren Weltordnung, demographische Entwicklungen) wird die Bewertung von Nachhaltigkeit im Sinne einer quantifizierbaren Abschätzung der Auswirkungen von Entscheidungen im Software Management weiter an Bedeutung gewinnen. Während Nachhaltigkeit in manchen Ingenieurdisziplinen (z.B. Produktionsprozesse und Dienstleistungen im Umweltmanagement und im Zivilingenieurwesen) bereits Einzug in die Standardisierung gehalten hat (z. B. ISO 14000 [Is04]) stehen wir diesbezüglich im Management von Software Systemen erst am Beginn dieses Prozesses.

2 Nachhaltiges Software Management

Nachhaltigkeit in der Softwareentwicklung sollte als orthogonaler Aspekt des Gesamtsystems, ähnlich wie andere Qualitätsattribute [Pe11, ZM07] wie z.B. funktionale Korrektheit, Sicherheit (Safety und Security) oder Verfügbarkeit, betrachtet werden. Noch in größerem Ausmaß als diese wohlbekannten (aber nach wie vor nicht einfach zu operationalisierenden) Qualitätsattribute betrifft dies den Aspekt der Nachhaltigkeit. Idealerweise sollte das zusätzliche Attribut der Nachhaltigkeit in einem sehr frühen Stadium eingeführt werden und alle weiteren Phasen wie Entwicklung, inkrementelle Verbesserung, Betrieb und Wartung begleiten. Auch muss sich die Operationalisierung der Attribute zur Bewertung der Nachhaltigkeit in die vorherrschenden Prozessabläufe und – soweit möglich – in die bestehende Toollandschaft integrieren. Ansätze, um die Nachhaltigkeit im Software Management messbar zu machen, werden z.B. in [AXTLZL10] diskutiert. Die Autoren schlagen vor, die Nachhaltigkeit (genauer Sustainability Performance, siehe [AXTLZL10]) als eine Menge an spezifischen Qualitätsattributen zu messen. Durch Erfassung dieser Metriken können so ökonomische, soziale oder ressourcenbezogene Ziele quantifiziert werden. Die Autoren dieser Arbeit zeigen qualitativ die Messung der Nachhaltigkeit und präsentieren eine Fallstudie für eine Software Plattform zum Wassermanagement. Die umfangreiche Liste der Metriken umfassen Ressourcen, Prozessaspekte und produkt-zentrische Metriken. Jedoch lassen die Autoren offen, wie in diesem Modell für Nachhaltigkeit diese Vielzahl von Kennzahlen in der Entwicklung und im Betrieb mit vernünftigem Aufwand operationalisiert werden kann.

Ein nicht unwesentliches Problem in der Erfassung von Kennzahlen über den gesamten Lebenszyklus ist die automatisierte Speicherung und flexible Darstellung der Kennzahlen. Abhilfe kann ein Software Cockpit [Pe11, RBKWL10, ZM07] schaffen, jedoch sind viele dieser Lösungen auf ausgewählte Phasen spezialisiert wie z.B. ausschließlich auf die Entwicklungsphase.

Die sinnvolle Integration dieser hochspezialisierten Werkzeuge ist oft nicht nahtlos möglich, wodurch sich komplexere Abfragen auf den erfassten Datenbeständen, die die Grundlage für Berechnung der Kennzahlen darstellen, nicht oder zumindest nicht projektbegleitend in der Entwicklung oder dem operativen Betrieb einer Software umsetzen lassen. Zur Bewertung der Nachhaltigkeit muss die Auswahl der zu erfassenden Kennzahlen nach Kriterien wie ökonomische Vorteile, soziale Vorteile und Schonung bzw. vorteilhafte Aufteilung von Ressourcen erfolgen. Bevor auf das Cockpit und die den Lebenszyklus übergreifende Erfassung von Kennzahlen eingegangen wird, zeigen wir beispielhaft einige für die Nachhaltigkeit relevante Kennzahlen aus drei Bereichen: Vorgangmanagement (im Sinne einer Prozessmetrik), Aufwandsmanagement (im Sinne einer Ressource Metrik) und Wiederverwendbarkeit (im Sinne einer produkt-zentrischen Metrik, die direkt am Produkt abgegriffen wird).

Kategorie	**Nachhaltigkeitsaspekt**	**Fragestellung**	**Metrik**	**Beispiel**
Ressource	Ökonomisch	Wird das Budget überschritten?	Qualität der Aufwandschätzung	SOLL / IST Aufwände
	Sozial	Werden Überstunden vermieden?		
	Ressourcen	Wer arbeitet an welchen Arbeitspaketen (optimale Teams)?		
Prozess	Ökonomisch	Wie schnell wird auf die Kundenanfrage reagiert?	Bearbeitungszeit	Zeit, die benötig wird, um vom Zustand Neu in den Zustand Fertig zu kommen
	Sozial	Gibt es kritische Zeitschwellen für Feedback?		
	Ressourcen	Ist der 1st Level Support ausreichend dimensioniert?		
Produkt	Ökonomisch	Verbesserung im Time-to-Market?	Wiederverwendbarkeit	Verhältnis Anzahl Abstrakte Klassen / Anzahl konkreter Klassen
	Sozial	Mehr Erfolge mit weniger Aufwand?		
	Ressourcen	Wieviel Ersparnis an Personal bei Wiederverwendung?		

Tabelle 7: Beispielhafte Fragestellungen für nachhaltiges Software Management

Wie die obigen Beispiele zeigen, erfordern konkrete Fragestellungen in Hinblick auf ökonomische, soziale und Ressourcen schonendes Management einer Software die Erfassung und Darstellung von Kennzahlen der verschiedenen Kategorien.

3 Software Cockpits

Metriken lassen sich nach verschiedenen Kriterien klassifizieren. Wie oben angeführt kann z.B. zwischen Produkt- (z.B. Lines of Code, Halstead, Function Points, Bang, McCabe Cycliomatic Complexity, Testabdeckung etc.) und Prozess- (z.B. Durchlaufzeiten, Dauer, Anzahl von Ereignissen, Anzahl abstrakter oder konkreter Klassen etc.) und Ressourcemetriken (z.B. Teamgröße, Werkzeuge, Methoden etc.) unterschieden werden. Schließlich kann man noch bzgl. der Auswertung unterscheiden zwischen primitiven und berechneten Metriken, sowie zwischen Momentaufnahmen und laufenden Betrachtungen, etwa von Trends oder Raten. Gerade im Hinblick auf Fragestellungen der Nachhaltigkeit, die oft die Verknüpfung der drei oben angeführten Kategorien erfordert, ist die Zusammenführung dieser Kennzahlen zu einem Nachhaltigkeitsmodell von entscheidender Bedeutung.

3.1 Vom Ziel zur Metrik: Wie findet man relevante Metriken und aggregiert diese sinnvoll?

Wie in der Motivation erwähnt, ist es wichtig, sich von Anfang an klar zu machen, welche Metriken in einem konkreten Anwendungsfall relevant sind. Da die Fragestellungen im Bereich der Nachhaltigkeit sehr vernetzt sein können, geht es oft nicht nur um einzelne Kennzahlen (oder Gruppen von Kennzahlen); vielmehr muss der CIO oder der Prozessverantwortliche die Daten systematisch analysieren können um geeignete Hypothesen für Nachhaltigkeitsbetrachtungen aufstellen zu können. Daher ist eine effiziente Speicherung und flexible Darstellung der Daten ein zentraler Bestandteil eines modernen Software Cockpits (OLAP [Ni98]). Das Kernstück unseres Cockpits stellt ein Datawarehouse dar, das die Daten in Cubes, wie beispielsweise in [GBLP96] beschrieben, organisiert. Dies ermöglicht eine effiziente aggregierte Darstellung der Metriken (Drill Down, Zoom, Slicing, Dicing). So können Daten aus den für Nachhaltigkeitsbetrachtungen oben erwähnten drei Bereichen (Prozess-, Ressource- und Produktmetriken) anhand verschiedener Dimensionen (Zeit, Module, Versionen) analysiert und zusammengeführt werden. Innerhalb der einzelnen Dimensionen können die Metriken auf verschiedenen Ebenen aggregiert werden. Z.B. kann für die ausgewählte Dimension Zeit über Jahre, Monate, Tage oder auch Quartale und Wochen aggregiert werden. Dabei greifen wir auf MDX Queries (Multi-Dimensionale Expressions [AADG96]) zurück, wobei diese MDX Queries für Endanwender auch durch eine GUI unterstützt werden. Weiter ermöglichen MDX Queries auch das Filtern und Sortieren der großen Datenbestände. Vorgefertigte Schablonen ermöglichen auch dem Endanwender solche Queries zu erstellen.

3.2 Architektur

Abbildung 9 zeigt die Architektur des Cockpits. Das Kernstück ist ein Datawarehouse, dem einerseits ein ausgeflachtes Datenbankschema, in dem die Dimensionen und Fakten gespeichert sind, und andererseits die Definition der Cubes, Metriken und Aggregate, zu Grunde liegen.

Abbildung 9: Architektur unseres Cockpits

Die Daten werden unter Verwendung von Adaptern aus den Artefakten gemappt und importiert. Auch Metriken und eventuell notwendige Aggregattabellen werden in dieser Schicht berechnet. Zu den Artefakten zählen Daten aus dem Issue Management, wie z.B. Bugzilla, und UML Modelle und Source Code aus der Versionsverwaltung, wie beispielsweise SVN. Die Adapter sind so gebaut, dass sie Ressourcen aus der Versionskontrolle holen können und zusätzlich Details wie Datum und Revisionsnummer bzw. Version berücksichtigen und abgleichen.

Ein Administrationsinterface bietet die Möglichkeit Einstellungen für die Adapter, Produkte und Projekte vorzunehmen und zusätzlich den gesamten Batchprozess, der die Adapter anstößt, zu parametrisieren. Die Darstellung erfolgt mittels eines Webinterfaces, das wie vorhin schon erwähnt durch MDX Anfragen die graphische Darstellung, so wie die Drill Down Ansichten steuert.

3.3 Messung der Nachhaltigkeit durch die Frage nach der Wiederverwendbarkeit

Wir zeigen hier eine Produktmetrik für die Wiederverwendbarkeit [Sn97]. Nach der Goal-Question Metrik ist das zugrundeliegende Ziel die Messung der Nachhaltigkeit. Die Frage, die wir uns stellen, ist, inwieweit Klassen wiederverwendet werden können. Klar ist, dass abstrakte Klassen, sofern sie korrekt entworfen und implementiert wurden, sehr wahrscheinlich in anderen Projekten und Produkten, denen ähnliche Use Cases und Problemstellungen zu Grunde liegen, wiederverwendet werden können.

Wir betrachten daher als Maß für die Wiederverwendbarkeit, das Verhältnis von abstrakten Klassen zu konkreten Klassen. Exemplarisch zeigen wir hier wie dieses Verhältnis zwischen der Anzahl der abstrakten Klassen und der Anzahl konkreter Klassen in OCL (Object Constraint Language) realisiert wird. Zur Ausführung einer OCL Query auf UML Diagrammen haben wir das Werkzeug SQUAM [Op09] in das Cockpit eingebunden. So ist die Menge der abstrakten Klassen definiert durch:

```
self.oclIsTypeOf(Class) and self.isAbstract = #true
```

und die Menge der konkreten Klassen:

```
self.oclIsTypeOf(Class) and self.isAbstract = #false
```

Damit lassen sich die Anzahl der abstrakten bzw. konkreten Klassen wie folgt definieren:

```
context Model
def numAbstractClasses:
numAbstractClasses (): Integer =
       self.allOwnedElements() ->    select(e:Element|e.oclIsTypeOf(Class)
and    e.isAbstract =#true ) ->  size()
def numImplClasses:
numImplClasses (): Integer =
       self.allOwnedElements() >     select(e:Element|e.oclIsTypeOf(Class)
and    e.isAbstract =#false ) -> size()
```

daraus ergibt sich das Verhältnis zwischen den beiden durch:

```
context Model
```

Tabelle 1:GQM an einem Beispiel aus der Automobil-Zulieferindustrie

```
query RelAbstrClassesImplClasses:
       let result: Integer =
              self.numAbstractClasses()/self.numImplClasses()
       message result endmessage
```

4 Fallstudie: Vorgangsmanagement bei einem Software Produkt für die Automobilzulieferindustrie

Wir gehen hier nach der Goal-Question-Metric, wie in [BW84] beschrieben, vor. Zuerst wird ein Ziel (Goal) definiert, woraus dann Fragestellungen (Questions) abgeleitet werden. Erst aus diesen ergeben sich die tatsächlichen Metriken. Wir betrachten hierzu eine Fallstudie aus der Automobilzulieferindustrie[2]. Das Ziel bei diesem Projekt ist es die Qualität des Software Engineering Prozesses, wie Durchlaufzeiten von Change Requests und die korrekte Einhaltung des Workflows, zu messen und einen Überblick über die Change Requests und Defects zu bekommen.

[2]Aufgrund der Sensibilität und Vertraulichkeit der Informationen nennen wir kein Unternehmen und präsentieren die Daten anonymisiert. Unser Anwender entwickelt ein Software Werkezeug für die Automobilindustrie.

Die folgende Tabelle zeigt die aus diesem Ziel abgeleiteten Fragestellungen und Metriken jeweils unter Angabe eines einfachen Beispiels.

Fragestellung	**Metrik**	**Beispiel**
Wie viele Issues sind in einen bestimmten Zustand übergegangen?	Number of Transitions	Anzahl Issues im Zustand *new*
Wie alt sind diese Issues?	Age (min, max, avg)	Alter der Issues im Zustand *new*
Wie viele Issues haben einen konkreten Zustand verlassen?	Number	Anzahl der Issues die den Zustand *new* verlassen haben
Wie lange hat es gedauert bis Issues einen bestimmten Zustand verlassen haben?	Elapsed Time (min, max, avg)	Zeit, die Issues im Zustand *new* verbracht haben
Wie viele Issues wechseln von einem konkreten Zustand in einen anderen bestimmten Zustand? (durch eine oder mehrere Transitionen)	Number	Anzahl der Issues die von dem Zustand *new* in den Zustand *ready to review* übergegangen sind (mehrere Transitionen)
Wie lange dauert es bis ein Issue von einem konkreten Zustand in einen anderen bestimmten Zustand wechselt?	Processing Time (min, max, avg)	Zeit, die es benötigt, bis Issues von dem Zustand *new* in den Zustand *ready to review* übergegangen sind (mehrere Transitionen)

Tabelle 2: CQM an einem Beispiel aus der Automobil-Zuliefererindustrie

In unserem konkreten Anwendungsfall ist der Großteil der Metriken einfach in MDX umzusetzen. Es gibt allerdings auch einen Teil an Metriken, die weitaus komplexer sind, da sie die Definition eines Calculated Members, Named Sets oder Funktionen erfordern. Als Beispiel möchten wir hier Metriken nennen, die Prozessdauer von einem Zustand zu einem nicht direkt darauf folgenden berechnen, wie wir es in Tabelle 2 dargestellt haben. Abbildung 2 stellt einen Screenshot des Dashboards dar, das mittels graphischer Darstellung einen Überblick ausgewählter Metriken gibt. Gleichzeitig bietet es die Möglichkeit einer Trendanalyse sowohl im Allgemeinen aber auch auf spezialisierte Weise, nach Produkten, Zeitrahmen und Personen. Das Dashboard ist dahingehend flexibel, als dass es die Möglichkeit bietet, einzelne Ansichten anzupassen.
Das zweite Set an Ansichten, wie in Abbildung 3 gezeigt, stellt Metriken aggregiert dar und bietet dabei die Möglichkeit in einzelne Dimensionen hinein zu zoomen. Dies wird durch die oben erwähnten MDX Anfragen umgesetzt. Die Herausforderung besteht bei der Darstellung der Metriken darin, die Dimensionen und Hierarchieebenen so zu wählen, dass die Tabellen einerseits übersichtlich und andererseits das Hineinzoomen in detaillierte Daten ermöglicht. Auch die Auswahl der Aggregate, ob beispielsweise die Summe, der Durchschnitt oder Maximum verwendet wird, spielt hier eine große Rolle und hängt stark von den anfangs definierten Zielen und Fragestellungen ab.

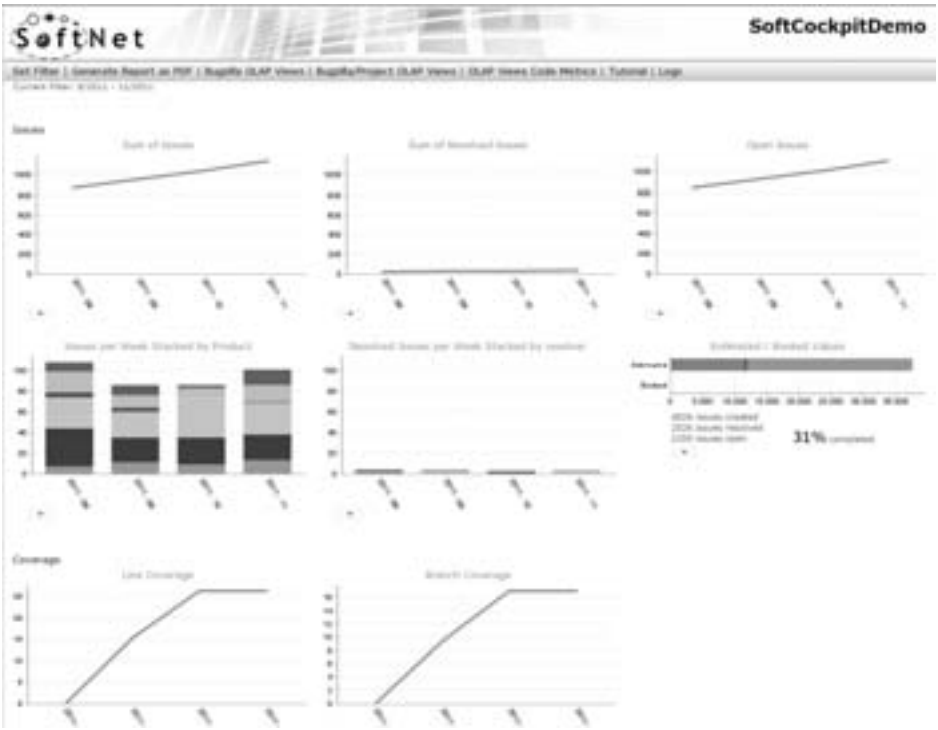

Abbildung 10: Screenshot des Dashboards

So wird in dem Beispiel aus Abbildung 11 die Anzahl der zugewiesenen Vorgänge sowie Minimum, Maximum und Durchschnittsalter dieser Issues in den Dimensionen Zeit x (Produkt und Issue-Art) dargestellt. Da das Alter der Issues in unserem Datenmodell gespeichert ist und die Aggregate im Cube definiert sind, ist dies eine einfache MDX Anfrage, die formal folgender Weise definiert werden kann:
Sei I die Menge aller Issues die im Datawarehouse gespeichert sind. Um Slicer zu beschreiben, die diese Menge nach bestimmten Kriterien $c_0, \cdots, c_l$ einschränken, definieren wir eine Menge $C \subseteq I$ für die gilt $C = \{x | c_i(x) = true, 0 \leq i \leq l\}$.
Zur Darstellung des Drilldowns auf den Spalten 1 bis n, führen wir für jede Spalte eine Untermenge $S_i \subseteq C, 0 \leq i \leq n$ ein, welche die Menge der Issues beschreiben, die das Kriterium der Drilldown-Ebene der Spalte i erfüllen (also beispielsweise in einem gewissen Zeitraum erstellt wurden). Analog dazu definieren wir Mengen $Z_0, \cdots, Z_m$ für die Zeilen 1 bis m.
So können nun die aggregierten Werte in der Tabelle, die sich auf Messwerte $m(x)$ wobei $m: I \mapsto \mathbb{R}$ (z.B. Alter eines Issues) und einer Funktion f (z.B. min, max, sum) beziehen, folgenderweise definiert werden:

$$agg: S_i \times Z_j \mapsto \mathbb{R}$$
$$agg(i,j) = f(\{m(x): x \in S_i \cap Z_j\})$$

Analog dazu kann auch die Wiederverwendbarkeit, wie sie in Abschnitt 3.3 beschrieben wurde, modelliert und nach verschiedenen Gruppierungen, wie beispielsweise Modulen oder Zeitrahmen, abgefragt werden.
Dazu definieren wir eine Messwertfunktion auf der Menge der Klassen K:
Sei $K_a \subseteq K$ die Menge der abstrakten Klassen dann ist $m_a: K \mapsto \{0,1\}$ definiert durch:

$$m_a(x) := \begin{cases} 1 \; iff \; x \in K_a \\ 0 \; else \end{cases}$$

Weiters liefert die von MDX vordefinierte Aggregatfunktion $count: S_i \times Z_j \mapsto \mathbb{N}$ für jede Spalten-Zeilen Kombination die Anzahl der Objekte, die die Kriterien erfüllen, also:
$count(i,j) := |\{x | x \in S_i \cap Z_j\}|$

Die Anzahl der abstrakten Klassen ergibt sich durch folgende Aggregatfunktion:

$$count_a(i,j) := \sum \{m_a(x) | x \in S_i \cap Z_j\}$$

Die Wiederverwendbarkeit ist definiert durch die Anzahl der abstrakten Klassen K_a und der Anzahl der konkreten Klassen K_k als $\frac{|K_a|}{|K_k|}$ also $\frac{|K_a|}{|K|-|K_a|}$.
Somit können wir für die Wiederverwendbarkeit folgende Aggregatfunktion $agg_W: S_i \times Z_j \mapsto \mathbb{R}$ definieren:

$$agg_W(i,j) := \frac{count_a(i,j)}{count(i,j) - count_a(i,j)}$$

Alternativ könnte man statt der Messwertfunktion auch Members definieren, um die Menge der abstrakten Klassen und die Menge der konkreten Klassen festzulegen, und mit der Hilfe des count Aggregats die Wiederverwendbarkeit definieren.

Das komplexere Beispiel aus Abbildung 4 zeigt exemplarisch die Verwendung einer solchen Memberdefinition am Fallbeispiel des Vorgangsmanagement.

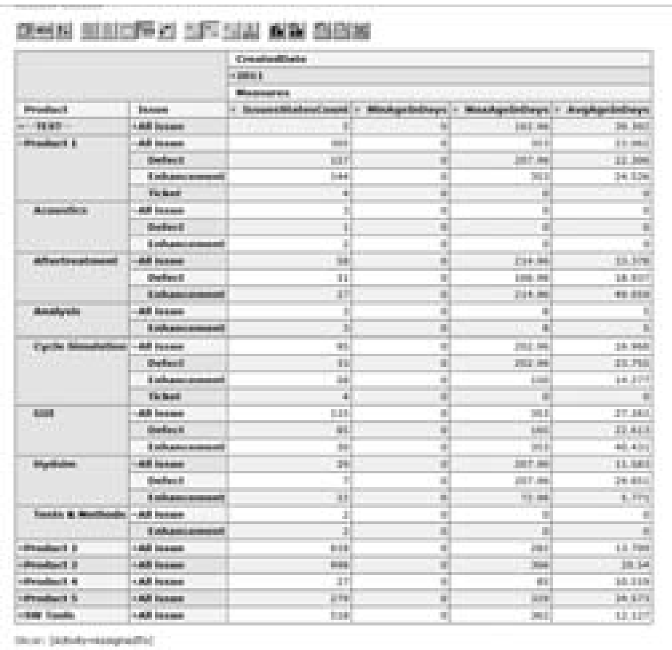

Abbildung 11: Drill-Down Ansicht zu Issues, die bereits zugewiesen wurden

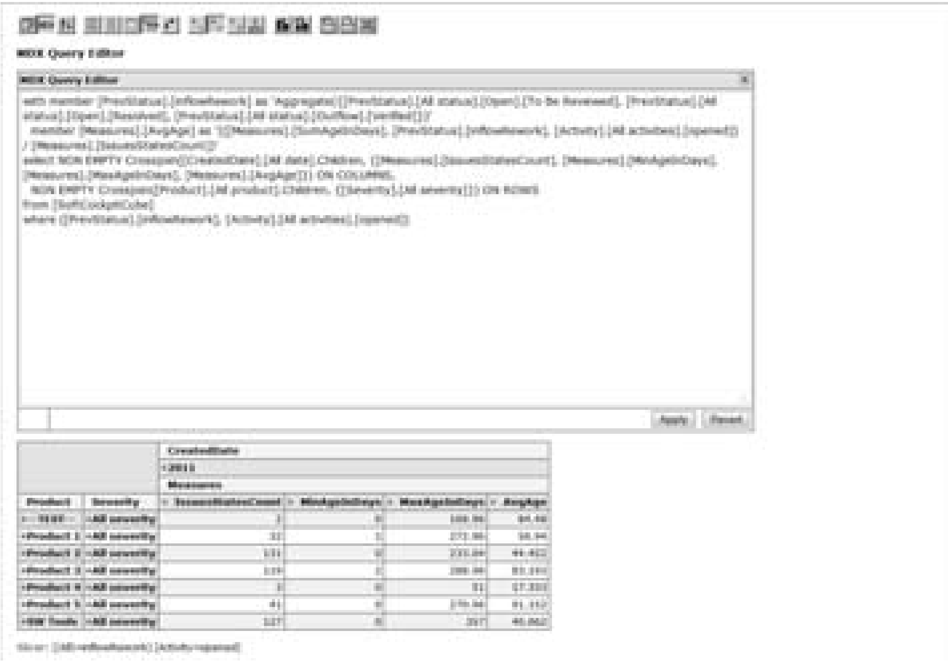

Abbildung 12: Beispiel einer mdx query unter Verwendung einer Member Definition

5 Diskussion und verwandte Forschungs- und Innovationsprojekte

Fallstudien im Bereich Monitoring von Kennzahlen wurden in den letzten Jahren vermehrt durchgeführt. Larndorfer und Ramler [LRB09a, LRB09b] berichten über die Herausforderungen bei der Einführung von Software Cockpits bei einem Software Produktentwickler und diskutieren auch dysfunktionale Effekte bei der Einführung von Software Dashboards. Das Projekt Soft-Pit [ZM07] zielt – wie die hier präsentierte Fallstudie – auf eine Verbesserung der Prozesstransparenz durch Auswertung von Prozesskennzahlen ab. Ähnlich wie auch das hier präsentierte Cockpit Ansatz stellt auch Soft-Pit eine Integrationsumgebung dar, die den Kristallisationspunkt einer standardisierten Infrastruktur für ein umfassendes Management von Kennzahlen bilden soll [BSRR07]. Jedoch greift Soft-Pit die Integration von Kennzahlen aus verschiedenen Qualitätsdimensionen – Prozesse, Produkte, Ressourcen – kaum auf. Unsere Lösung verfolgt den GQM-Ansatz [BW84], der es ermöglicht, ein für den Anwendungsfall und für die Überwachung von bestimmten Qualitätszielen geeignetes Kennzahlsystem festzulegen. Dabei lässt sich das Kennzahlsystem entlang unabhängiger Hierarchien (Zeit, Module, Versionen, Personalressourcen, …) strukturieren. Die Autoren von [BSRR07] berichten über die Anwendung eines Software Cockpits auf das Software Cockpit selbst und auch über erste Erfahrungen in Pilotprojekten.

Als wichtigster Vorteil gegenüber dem Einzeleinsatz von Analysewerkzeugen wird quellen-übergreifende Erfassung von Daten, die miteinander in Beziehung gesetzt werden können, angeführt womit sich Entscheidungsträger ein umfassendes Bild der Qualitäts- oder Nachhaltigkeitsziele verschaffen können. Wie und mit welcher Granularität die Kennzahlen dabei in Relation gesetzt werden ist rollen-, projekt- und auch branchenspezifisch[3]. Neben der Zeitreihenanalyse haben sich die Tabellendarstellungen in den Pilotprojekten bewährt. Weiter berichten die Autoren, dass die angebotene Drill-Down Fähigkeit nur spärlich benutzt wird. Dies steht im Gegensatz zu unserem Anwender, wo eine Drill-Down Fähigkeit entlang der verschiedenen Dimensionen wie z.B. Produkte oder Versionen gewünscht ist.

5.1 Fazit

In diesem Papier argumentieren wir, dass die Beurteilung der Nachhaltigkeit im Management von Software den gesamten Lebenszyklus der Software erfassen muss. Damit z.B. Behörden und Unternehmen Fragestellungen im Bereich des nachhaltigen Software Managements beantworten können, ist eine kontinuierliche Vermessung der Software über den gesamten Lebenszyklus – Entwurf, Umsetzung, Betrieb und Wartung – vonnöten. Die Erfassung solcher Kennzahlen stellt eine große Herausforderung dar, da Fragestellungen in Bezug auf Nachhaltigkeit das Operationalisieren von produktspezifischen Kennzahlen (z.B. Grad an Wiederverwendbarkeit), Prozessmetriken (z.B. Bearbeitungszeit Change Requests) und Ressource-Metriken (z.B. Genauigkeit der Aufwandschätzung) erfordern.

[3] Diese Sicht wurde in einem informalen Gespräch mit erfahrenen Beratern der Capgemini bei den Software Quality Days in Wien aus praktischer Sicht bestätigt.

Wir stellen ein Software Cockpit vor, dass es ermöglicht, Kennzahlen aus den verschiedenen Bereichen entlang verschiedener Dimensionen (Zeit, Module, Version) zu integrieren. Die so aggregierten Datenbestände erlauben es verschiedene Fragestellungen – auch solche aus dem Bereich nachhaltiges Software Management – qualitativ und quantitativ zu beantworten. Neben der Methodik zur automatisierten Erfassung der Kennzahlen präsentieren wir eine Fallstudie im Bereich des Vorgangsmanagements für eine Software aus der Automobilzulieferindustrie. Daran ist klar zu sehen, dass die für praktische Fragestellungen zu operationalisierenden Kennzahlen ohne flexible Messinfrastruktur nicht zu bewältigen sind. Dies gilt im Speziellen für die vernetzten Fragestellungen aus dem Bereich des nachhaltigen Software Managements.

Literaturverzeichnis

[AADG96] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, S. Sarawagi. On the computation of multidimensional aggregates, Proc. 22nd VLDB, pages 506-521, Mumbai, Sept. 1996.

[AXTLZL10] F. Albertao, J. Xiao, C. Tian, Y. Lu, Q. Zhang, C. Liu, Measuring the Sustainability Performance of Software Projects, Proceedings of IEEE 7th International Conference on e-Business Engineering., IEEE, 2010.

[BD02] J. Bansiya, C. Davis. A Hierarchical Model for Object-Oriented Design Metrics as Quality Assessment, IEEE Transactions on Software Engineering, Vol. 28, Nr. 1, pages 4-17, 2002.

[BSRR07] Marcel Bennicke, Frank Steinbrückner, Mathias Radicke, Jan-Peter Richter: Das sd&m Software Cockpit: Architektur und Erfahrungen. GI Jahrestagung (2) 2007: 254-260.

[BW84] Victor R. Basili, David M. Weiss: A Methodology for Collecting Valid Software Engineering Data. IEEE Trans. Software Eng. 10(6): 728-738 (1984).

[Du01] Duden - Deutsches Universalwörterbuch. 4. Aufl. Mannheim 2001.

[GBLP96] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Proc. 12th Int'l IEEE Conf. on Data Engineering (ICDE), New Orleans, February/March 1996.

[LRB09a] Stefan Larndorfer, Rudolf Ramler, Clemens Buchwiser: Experiences and Results from Establishing a Software Cockpit at BMD Systemhaus. EUROMICRO-SEAA 2009: 188-194.

[LRB09b] Stefan Larndorfer, Rudolf Ramler, Clemens Buchwiser Dashboards, Cockpits und Projektleitstände: Herausforderung Messsysteme für die Softwareentwicklung, Object Spektrum - Die Zeitschrift für Software-Engineering und -Management, Ausgabe 04 2009.

[Is04] ISO14000, International Organization for Standardisation, 2004 http://www.iso.org. (15.05.2012).

[Ni98] Nils Clausen: OLAP – Multidimensionale Datenbanken. Addison-Wesley-Longman, Bonn 1998, ISBN 3-8273-1402-X.

[Op09] Joanna Chimiak-Opoka. OCLLib, OCLUnit, OCLDoc: Pragmatic Extensions for the Object Constraint Language. In Andy Schuerr and Bran Selic, editors, Model Driven Engineering Languages and Systems, 12th International Conference, MODELS 2009, Denver, Colorado, USA, October 4-9, 2009, Proceedings. LNCS 5795, pages 665-669. Springer Verlag, 2009.

[Pe11] Bernhard Peischl, Das Softnet Integrationsprojekt: Kennzahl-gestützte kontinuierliche QS im Software Engineering, Software Quality Days 2011, Wien, Österreich, 18-20, January 2011.

[RBKWL10] Rudolf Ramler, Wolfgang Beer, Claus Klammer, Klaus Wolfmaier, Stefan Larndorfer: Concept, Implementation and Evaluation of a Web-Based Software Cockpit. EUROMICRO-SEAA 2010: 385-392.

[Sn97] H.M. Sneed, Metriken für die Wiederverwendbarkeit von Softwaresystemen,.Informatikspektrum, Vol. 6, pages 18-20, 1997.

[ZM07] Th. Zehler, J. Münch, , Soft-Pit: Ganzheitliche Projekt-Leitstände zur ingenieurmäßigen Software-Projektdurchführung, Frauenhofer IESE, VSEK/051/D, http://www.software-kompetenz.de (15.05.2012).