2. Grundlagen

Miss alles, was sich messen lässt, und mach alles messbar, was sich nicht messen lässt.

(Galileo Galilei, italienischer Mathematiker, Physiker und Astronom, ⋆1564 †1642)

Inhalt

Das zentrale Thema dieser Bachelorarbeit ist es Unterstützung bei der Erstellung von Metriken in der Softwareentwicklung mithilfe neuester Technologien, wie Enterprise Java Beans 3.0, Webservices etc., zu leisten. Grundlagen der Metriken und Metrikprozesse werden in Abschnitt 2.1 und Abschnitt 2.2 behandelt. Einzelheiten zu den eingesetzten Technologien werden in Abschnitt 2.3 erläutert. Einen Einblick in die aktuelle Forschung am Lehrstuhl bietet der Abschnitt 2.4.1.

2.1. Metriken

Da das Kernthema dieser Bachelorarbeit Metriken und Metrikprozesse sind, wird sich dieser Abschnitt mit dem Thema Softwaremetriken beschäftigen. Zunächst wird der Begriff Metrik und die mit ihm zusammenhängenden Begriffe erläutert. Daraufhin wird der Nutzen von Metriken bei der Softwareentwicklung behandelt. Im Anhang finden sich außerdem Beispiele für Metriken.

2.1.1. Einführung in die Metriken

Nach Fenton u. Pfleeger [9] ist eine Metrik „the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.“ Die ISO/IEC 15939 [16] definiert Metriken als „set of operations having the object of determing a value of a measure“. Die IEEE [24] definiert eine Metrik wiederum so: „A quantitative

measure of the degree to which a system, component, or process possesses a given variable“.

Aus diesen Definitionen folgt, dass Metriken Attribute von reellen Objekten auf klar definierte Art und Weise auf Zahlen oder Symbole abbilden. Auch Ludewig u. Lichter [18] verstehen Metriken als eine „Abbildung einer Software oder eines Prozesses der Software-Bearbeitung auf eine skalare oder vektorielle Größe“. Um Metriken verwenden zu können, benötigt man genau definierte Regeln, die die Art und Weise des Messens festlegen. Des Weiteren müssen Metriken Homomorphismen sein, d.h. sie sind relationserhaltend. Stehen die Attribute x und y in Relation R zueinander, das heißt $(x, y) \in R$, dann muss es auch eine Relation P geben, so dass $(M(x), M(y)) \in P$ ist, wobei M eine Metrik ist. Außerdem benötigt man für die Anwendung von Metriken die Definition einer Skala. Diese entscheidet darüber, welche Operationen auf den Messwerten erlaubt sind.

Als Beispiel diene die „Höhe“. Das reale Objekt sei ein Schrank und das vermessene Attribut ist die Höhe. Diesem Attribut wird eine Zahl zugewiesen und zwar die Höhe des Schrankes z.B. in Zentimeter. Als Skala wird also die „Zentimeter-Skala“ benutzt. Man könnte aber auch einfach eine Abbildung der Höhe auf die Symbole $\{++, +, \circ, -, --\}$ vornehmen. Die Regeln für das Messen umfassen hier beispielsweise wie das Maßband angelegt wird und wie die Distanz abgelesen wird. Da Metriken Homomorphismen sind, müssen die Relationen der Metrikwerte denen der realen Attribute entsprechen. Das heißt, wenn Schrank A höher ist als Schrank B, muss auch die Abbildung der Höhe in Form einer Zahl, eine entsprechende Relation besitzen.

Skalen

Die oben bereits erwähnte Skala ist ein wichtiger Teil einer Metrik, da sie den Rahmen für den Vergleich und die Auswertung der Messergebnisse festlegt. Man unterscheidet zwischen fünf Skalenarten [vgl. 18]:

Nominalskala Hierbei handelt es sich um eine reine Kategorisierung. Die Werte sind nur auf Gleichheit und Ungleichheit prüfbar. Ungleiche Werte können nicht miteinander in Relation gesetzt werden. Ein Beispiel wäre die Kategorisierung von Anwendungssoftware in Bereiche wie betriebswirtschaftliche Software, Kreativsoftware, Spielsoftware, Verwaltungssoftware, Entwicklungssoftware, etc.

Ordinalskala Diese Skala unterscheidet sich insofern von der Nominalskala, als dass sie totalgeordnet ist. Alle Werte sind untereinander vergleichbar und ein Median kann gebildet werden. Das Standardbeispiel für eine Ordinalskala ist die Schulnotenskala. Hier ist „sehr gut“ besser als „gut“ usw. Die bekannte „Durchschnittsnote“ ist aber eigentlich nicht erlaubt, da Differenzen der Werte einer Ordinalskala nicht definiert sind.

Intervallskala Die Intervallskala erlaubt die Differenzbildung und ermöglicht dadurch den Vergleich von Distanzen. Somit ist der Mittelwert bestimmbar. Als Beispiel kann die Celsius-Temperaturskala herangezogen werden. In dieser Skala kann eine Jahresdurchschnittstemperatur berechnet werden. Problematisch ist, dass die Temperaturen selbst nicht ins Verhältnis gesetzt werden können. Eine Aussage wie: „20°C sind doppelt so warm wie 10°C“ klingt zunächst sinnvoll. Von einem US-Amerikaner erhält man aber sicher eine ganz andere Antwort, da dieser die Temperatur in Grad Fahrenheit misst. Das Problem beider Skalen ist der willkürlich gewählte Nullpunkt. [1]

Verhältnisskala Bei der Verhältnisskala ist der Nullpunkt nicht willkürlich gewählt, so dass Verhältnisse gebildet werden können. Dementsprechend ist es in der Kelvin-Temperaturskala erlaubt Verhältnisse zu bilden, da ihr Nullpunkt beim physikalisch bestimmten „absoluten Nullpunkt“ liegt. Bei der Verhältnisskala sind alle Grundrechenarten zwischen den Skalenwerten erlaubt.

Absolutskala Die Absolutskala ist die Skala, die die meisten Axiome besitzt. Jede Transformation außer der Identität überführt eine Absolutskala in einen anderen Skalentyp. Im weitesten Sinne misst sie immer nur Anzahlen von Objekten. So ist bei ihr „der Zahlenwert nicht nur proportional zur gesuchten Größe, sondern selbst die gesuchte Größe“ [18]. Ein Beispiel für eine Absolutskala sind Lines of Code (LOC).

Metriken können außerdem in vielerlei Hinsicht klassifiziert werden. Eine Klassifizierung hinsichtlich verschiedener Bereiche schließt sich an.

Klassifizierung von Metriken

Messgegenstand Eine erste Unterteilung erfolgt nach den Messgegenständen in Produkt- und Prozessmetriken.

Komplexität Weiterhin werden Basismetriken von berechneten Metriken unterschieden. Basismetriken sind sehr leicht abbildbar, aber alleine nicht sehr aussagekräftig. Ein Beispiel sind Lines of Code, da diese einfach durch abzählen der Codezeilen bestimmt werden. Berechnete Metriken sind abgeleitete Metriken, die durch Verknüpfung anderer Metriken entstehen bei denen es sich sowohl um Basismetriken als auch um berechnete Metriken handeln kann (siehe Abbildung 2.1).

Zweck Zusätzlich trifft man eine Unterscheidung bezüglich des Zwecks. So gibt es präskriptive und deskriptive Metriken, d.h. Metriken, die einen

[1] 0°C entsprechen dem Schmelzpunkt von Wasser und 0°F entsprechen einer Kältemischung aus Eis, Wasser und Salmiak oder Seesalz, die der kältesten Temperatur des Winters 1708/1709 in Danzig entsprach.

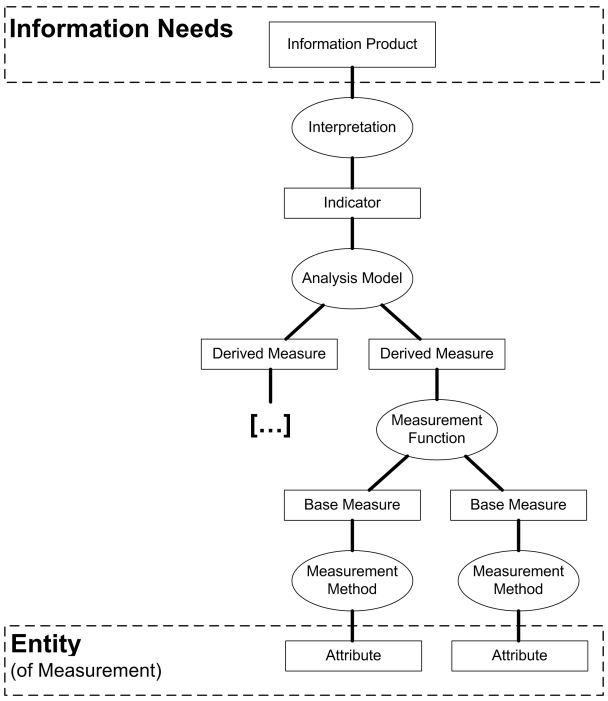

Abbildung 2.1.: Ableitung von Metriken aus einem Informationsbedarf nach ISO 15939 [16]

Soll- bzw. Ist-Zustand beschreiben. Die deskriptiven Metriken werden noch weiter in diagnostische und prognostische Metriken unterteilt, wobei die diagnostischen den momentanen Zustand und die prognostischen den zukünftigen, erwarteten Zustand charakterisieren.

Bestimmung der Messwerte Eine weitere Unterteilung erfolgt anhand der Art und Weise, wie die Metrikwerte bestimmt werden, da Metriken nicht unbedingt nur objektiv sein müssen, sondern auch subjektiv sein können, d.h. auf Schätzungen beruhen.

Basis der Metrik Die Basis der Metrik ist eine weitere Unterscheidungsmöglichkeit, da Metriken entweder modellbasiert oder empirisch belegt sind. Dementsprechend liegt einer Metrik entweder ein Modell zugrunde, das versucht die Realität möglichst genau abzubilden, oder es ist kein erklärendes Modell zwischengeschaltet, sondern die Metrik beruht auf reiner Empirie.

„Unterlaufbarkeit“ Metriken können die negative Eigenschaft der „Unterlaufbarkeit“ besitzen, was bedeutet, dass die Angestellten bei Kenntnis der angewandten Metriken diese unterlaufen können. Wenn der Entwickler weiß, dass eine Metrik benutzt wird, die die Länge von Methoden einbezieht, um die Komplexität des Softwareprodukts zu bestimmen, so kann er eine negative Bewertung vermeiden, indem er jede Methode in mehrere Methoden splittet, ohne tatsächlich die Komplexität zu verringern.

Im Gegensatz zu unterlaufbaren Metriken gibt es robuste Metriken wie z.B. Function Points. Functions Points wurden von Albrecht [1] entwickelt und sind ein Maß für die Komplexität eines SW-Produkts. Innerhalb von einzelnen SW-Organisationen kann mit ihrer Hilfe sehr gut die Komplexität einzelner Produkte unterschieden, so lange man sich an die selben Berechnungsgrundlagen hält. Die heutzutage in der Industrie eingesetzte Variante von Cosmic [7] liefert eine große Menge an Tabellen, aus denen die FP-Werte für bestimmte Softwarebausteine mit Gewichten abgelesen werden können. Den Berechnungsvorschriften folgend ergibt sich am Ende ein FP-Wert für das gesamte Produkt.

Um den Nutzen von Metriken bei der Softwareentwicklung zu verdeutlichen, folgt eine Erläuterung dazu im nächsten Abschnitt.

2.1.2. Nutzen von Metriken

Metriken können in der Softwareentwicklung sehr nützlich sein, werden in der Industrie bisher aber selten eingesetzt (siehe Seibert [23]). Der Nutzen von Metriken wird bereits durch die Zitate am Anfang der Kapitel klar. Durch Metriken lassen sich die Produkte und Prozesse der Softwareentwicklung objektiv messen und vergleichen, sofern die richtigen Skalen verwendet werden. Metriken sind somit unerlässlich, wenn Wissen über Prozesse und Produkte erforderlich ist, wenn sinnvoll geplant werden soll, wenn Missstände aufgedeckt werden sollen, wenn Prognosen erstellt werden sollen, usw. Dies ist für größere Unternehmen und Projekte unerlässlich.

Es ist offensichtlich, dass ein Projektmanager, der keine objektiven Zahlen zu seinem Projekt hat, nur schwer objektive, rationale Entscheidungen treffen kann. So ist schon bei *CMMI*-Reifegrad 2 *Measurement and Analysis (MA)* gefordert, was auch die Verwendung von Metriken einschließt.

Bei MA handelt es sich um einen Prozessbereich des *CMMI (Capability Maturity Model Integrated)*, der festlegt, dass die Softwareorganisation Messungen durchführen können muss, um Reifegrad 2 zu erreichen. CMMI dient der Bewertung von Softwareorganisationen um die Qualität ihrer Produkte und Prozesse abschätzen zu können (vgl. Kneuper [17]). Angenommen wird dabei, dass Software-Prozesse umso besser sind, je besser deren Definition, Planung und Ausführung ist. Letztendlich geht man davon aus, dass die Qualität der Produkte, die Termin- und Kosteneinhaltung umso besser ist, je höher der CMMI-Reifegrad der SW-Organisation ist. Dies ist auch der Grund aus dem CMMI entwickelt wurde, da es das US-amerikanischen Verteidigungsministerium bei der Wahl der SW-Unternehmen, die für einen Auftrag infrage kommen, unterstützen sollte. Unternehmen mit einem höheren CMMI-Reifegrad liefern im Allgemeinen pünktlicher, qualitativ hochwertigere Produkte als Unternehmen niedriger Reifegrade.

Durch die Unterstützung von Metriken wie z.B. dem *C*ost *P*erformance *I*ndex können Probleme viel früher entdeckt und behoben werde. Der CPI eines Projektes sollte möglichst dauerhaft größer oder gleich 1 sein, da ansonsten höhere Kosten als geplant entstehen. Allein durch diese einfache Metrik kann der Projektmanager besser entscheiden, ob er beispielsweise ein Projekt einstellen soll.

Wichtig ist aber nicht nur das Messen der Prozesse im Unternehmen z.B. mittels CMMI, sondern auch die Qualität der eigenen Produkte um seine Kunden nicht zu verlieren. Ohne eine Vermessung der eigenen Produkte kann nicht geprüft werden, ob die eigenen Qualitätsziele erreicht werden. Qualitätsziele sind nach der ISO Norm 9000:2000 (Quelle: DGQ [11]) „Etwas bezüglich Qualität Angestrebtes oder zu Erreichendes“. Das heißt Qualitätsziele eines Unternehmens sind die angestrebten Qualitäten für die eigenen Produkte. Zum Beispiel könnte gefordert sein, dass die Wartbarkeit der Software hoch sein muss. Doch die metriklose Bewertung des Endprodukts hinsichtlich der Wartbarkeit ist sicher nicht objektiv. Als Ansatz zur Messung der Wartbarkeit wäre es z.B. sinnvoll die innere Komplexität des Programmcodes zu messen. In die Bewertung der Komplexität einer objekt-orientierten Software könnte z.B. die Anzahl der Klassen, die Anzahl der Klassenbeziehungen, die Tiefe der Vererbungshierarchien und ähnliches eingehen. Um den Nutzen von Metriken weiter zu verdeutlichen sei auf den Artikel von Seibert [23] verwiesen, wo deutlich gemacht wird, dass man mithilfe von Metriken beispielsweise das Problem zu niedriger Planvorgaben vermindern kann. Im Management ist der Irrglaube, niedrige Planvorgaben steigerten die Produktivität, nämlich immer noch weit verbreitet. Mithilfe von Metriken wie COCOMO (siehe Abschnitt A.1.1) kann man bei vernünftiger Kalibrierung dieses Problem leicht beheben, da eine plausible, objektive, empirisch nachgewiesene Metrik für das Management überzeugender ist, als der Hinweis des Entwicklerteams, die Planvorgaben seien zu niedrig.

Somit ergibt sich, dass Metriken bei der Softwareentwicklung sehr nützlich sein können und wahrscheinlich eine immer größere Rolle in der Industrie spielen werden. Allerdings muss darauf hingewiesen werden, dass Metriken nur bei größeren Projekten sinnvoll sind. Bei kleinen Projekten ist der Overhead, der durch die Entwicklung, Berechnung und Interpretation von Metriken entsteht, zu groß. Um das Konzept einer Metrik besser zu verstehen, finden sich im Anhang Beispiele für Metriken.

2.2. Metrikprozesse

Nachdem im vorherigen Abschnitt eine kurze Einführung in die Softwaremetriken geliefert wurde, geht dieser Abschnitt nun auf den Metrikprozess ein, d.h. unter anderem auf die Entwicklung von Metriken. Metrikprozesse sind das Hauptthema dieser Arbeit, da die hier erstellte Software den Metrikprozess unterstützen soll, indem sie unterstützend bei der Kommunikation zwischen einem