2. Zielsetzung, Messung

Ohne definierte *Ziele* ist keine systematische Software-Entwicklung möglich. Eine systematische Zielerreichung ist nur dann möglich, wenn die Ziele kontinuierlich verfolgt und Abweichungen festgestellt werden. Hierzu sind *Messungen* erforderlich.

In diesem Kapitel werden die Problematik der Zielsetzung und Zielverfolgung sowie die Grundlagen der dafür erforderlichen Messtechnik behandelt.

2.1 Das Zielsetzungsproblem

Zielgerichtetes Arbeiten ist eine notwendige Voraussetzung für jegliche Art von systematischer Entwicklung.

In einer Entwicklung ohne formulierte Ziele sind die Eigenschaften und Qualitäten der resultierenden Software zufällig. Sie ergeben sich bestenfalls noch teilweise aus offensichtlichen Bedürfnissen der Problemstellung. Zur Hauptsache hängen sie jedoch dann ab vom Charakter und den Vorlieben der Entwicklerinnen und Entwickler, von Gruppenstrukturen und von den Beziehungen zwischen Entwicklern und Benutzern.

Software Engineering betreiben bedeutet daher zwingend, dass für jede Entwicklung von Software Ziele gesetzt werden und anschließend systematisch auf diese Ziele hin gearbeitet wird.

Es gibt jedoch für die Herstellung von Software kein „natürliches“ Ziel. Viele mögliche Teilziele stehen in Konkurrenz zueinander. Beispielsweise kann eine Software nicht gleichzeitig extrem zuverlässig und besonders kostengünstig in der Entwicklung sein.

Die Wahl der Ziele hat einen erheblichen Einfluss sowohl auf das entstehende Produkt als auch auf den Entwicklungsprozess. Ein von Weinberg und Schulman (1974) durchgeführtes Experiment zeigt dies in eindrücklicher Weise (Bild 2.1). Weinberg und Schulman hatten fünf Entwicklungsgruppen die gleiche Software entwickeln lassen, den fünf Gruppen aber unterschiedliche Projektziele gesetzt. Das Ergebnis zeigt zweierlei:

- Die Qualitäten der erzeugten Programme sind stark korreliert sind mit den Zielen, welche den Entwicklungsgruppen vorgegeben wurden.
- Die optimale Erreichung des gesetzten Ziels ging zum Teil erheblich auf Kosten anderer Qualitäten. Das heißt, dass manche Ziele sich gegenseitig konkurrenzieren.

Es kann daher keinen für alle Software-Vorhaben tauglichen Satz optimaler Ziele geben, sondern die Ziele müssen für jedes Vorhaben entsprechend den jeweiligen konkreten Bedürfnissen und Randbedingungen neu festgesetzt werden.

Bei der Planung muss berücksichtigt werden, dass manche Ziele abhängig voneinander sind (z.B. Termine und Sachziele) und daher keine beliebigen Freiheitsgrade bei der Zielfestlegung bestehen. Dort, wo wesentliche Ziele sich konkurrenzieren, ist es sinnvoll, die Ziele mit Prioritäten zu versehen.

FESTSTELLUNG 2.1. Es gibt keine natürlichen Ziele für Software. Die Ziele müssen für jedes Entwicklungsvorhaben neu festgelegt werden. Dabei ist zu beachten, dass sich Ziele konkurrenzieren können und dass Ziele voneinander abhängig sein können.

FESTSTELLUNG 2.2. Die Wahl der Ziele hat einen erheblichen Einfluss auf das Produkt und den Entwicklungsprozess.

Ziel: Optimiere...	Qualität der Ergebnisse: Erstellungsaufwand	Anzahl Anweisungen	Speicherbedarf	Klarheit des Programms	Klarheit der Ausgaben
Erstellungsaufwand	1	4	4	5	3
Anzahl Anweisungen	2 - 3	1	2	3	5
Speicherbedarf	5	2	1	4	4
Klarheit des Programms	4	3	3	2	2
Klarheit der Ausgaben	2 - 3	5	5	1	1

BILD 2.1. Leistungen von Programmierern in Abhängigkeit von den vorgegebenen Zielen.
1 = beste Leistung, 5 = schlechteste Leistung

2.2 Klassifikation von Zielen

In den meisten Software-Projekten lassen sich die Ziele nach dem in Bild 2.2 gezeigten Schema ordnen. Wesentlich ist die Untergliederung der Ziele in Projekt- und Produktziele, wobei die Erreichung der Produktziele mit ein zentrales Projektziel ist. Die Anforderungen an die bereitzustellende Software sind ein Teil der Zielsetzung.

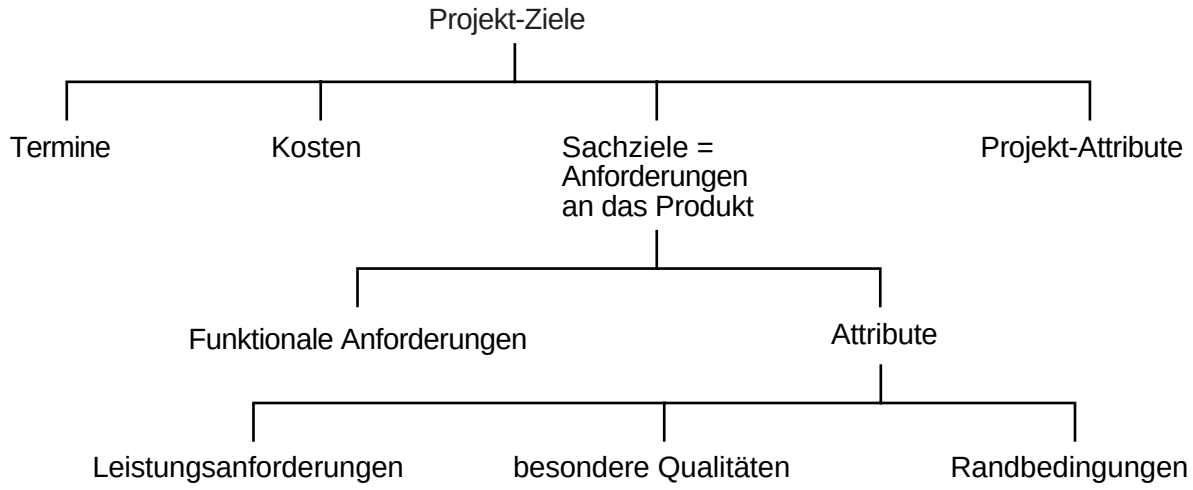

BILD 2.2. Klassifikationsschema für Ziele

2.3 Zielverfolgung

Ziel*setzung* ist notwendig für ein systematisches Vorgehen, aber nicht hinreichend. Nach der Festlegung der Ziele muss systematisch auf das Erreichen der Ziele hingearbeitet werden. Dabei genügt ein einmaliges Fokussieren auf die Ziele zu Beginn nicht. Langsame, aber stetige Abweichungen von den Zielen während der Entwicklung können dazu führen, dass Ziele massiv verfehlt werden, ohne dass dies rechtzeitig bemerkt wird. Es braucht daher eine regelmäßige Ziel-

kontrolle, so dass bei Abweichungen geeignete Gegenmaßnahmen getroffen werden können. Zielkontrolle ist jedoch nur möglich, wenn das Erreichte mit den Zielen verglichen werden kann.

Es muss daher möglich sein, die Ziele zu *messen*. Hierzu müssen zunächst einmal geeignete Maße gefunden oder definiert werden. Dann müssen für jedes Ziel *Referenzwerte* festgelegt werden. Im Minimum sind dies der Zielwert (wenn dieser Wert gemessen wird, ist das Ziel zu 100% erreicht) und ein Schwellwert (wenn dieser Wert mindestens erreicht ist, wird das Ergebnis als nahe genug beim Ziel akzeptiert).

2.4 Messung

Durch Messungen werden interessierende Merkmale eines Gegenstands quantitativ erfassbar gemacht.

2.4.1 Maße

Ein Maß ordnet mit Hilfe eines *Messverfahrens* einer Menge von Gegenständen *Messwerte* auf einer *Skala* zu. Die Messwerte machen eine Aussage über das zu messende *Merkmal* der gemessenen Gegenstände. Dabei müssen die Menge der Merkmalswerte und die Skala *strukturähnlich* sein, das heißt, die Eigenschaften der Skala müssen mit denen des Merkmals übereinstimmen.

DEFINITION 2.1. *Maß*. Sei D eine Menge gleichartiger Gegenstände mit einem zu messenden Merkmal M, sei M(d) die Ausprägung des Merkmals M für den Gegenstand $d \in D$ und sei M die Menge aller dieser Merkmalsausprägungen. Ein *Maß* für das Merkmal M ist eine Abbildung $\mu: D \rightarrow S$, welche jedem $d \in D$ einen Messwert $\mu(d)$ auf einer *Skala* S so zuordnet, dass M und S strukturähnlich sind. Dies ist genau dann der Fall, wenn für beliebige d1, d2 aus D gilt:
Zu jeder Relation R auf der Menge der Merkmalsausprägungen M gibt es eine analoge Relation R* auf der Skala S mit folgenden Eigenschaften

(1) $R(M(d1), M(d2)) \Rightarrow R^*(\mu(d1), \mu(d2))$

(2) $R^*(\mu(d1), \mu(d2)) \Rightarrow R(M(d1), M(d2))$ und die Aussage R(M(d1), M(d2)) ist sinnvoll interpretierbar.

Maße für Software werden in der Literatur oft auch als *Metriken* bezeichnet. Die Bedingungen (1) und (2) werden auch als *Repräsentations-* oder als *Homomorphiebedingung* bezeichnet.

Häufig werden Maße definitorisch eingesetzt, d.h. sie *definieren* ein intuitives Merkmal einer Menge von Gegenständen.

BEISPIEL 2.1. Messung der *Größe* eines Programms. Ein mögliches Maß γ für das Merkmal «Größe» eines Programms besteht aus der Abbildungsvorschrift «Zähle die Programmzeilen, die nicht leer sind oder ausschließlich Kommentar enthalten» und den nicht negativen ganzen Zahlen als Skala.

Auf dem Merkmal «Größe» eines Programms gibt es zwei Relationen: die Merkmalsausprägungen sind *vergleichbar* und sie sind *additiv*. Beide Relationen müssen durch die Abbildungsvorschrift so auf entsprechende Relationen der Skala abgebildet werden, dass die Homomorphiebedingung erfüllt ist; andernfalls wäre γ kein Maß. Es muss also für beliebige Programme P1, P2 und P3 gelten und sinnvoll interpretierbar sein

(1) $\text{Größe}(P1) \leq \text{Größe}(P2) \Rightarrow \gamma(P1) \leq \gamma(P2)$

$$\text{Größe}(P1) + \text{Größe}(P2) = \text{Größe}(P3) \Rightarrow \gamma(P1) + \gamma(P2) = \gamma(P3)$$

(2) $\gamma(P1) \leq \gamma(P2) \Rightarrow \text{Größe}(P1) \leq \text{Größe}(P2)$

$$\gamma(P1) + \gamma(P2) = \gamma(P3) \Rightarrow \text{Größe}(P1) + \text{Größe}(P2) = \text{Größe}(P3)$$

Beides ist mit der gegebenen Abbildungsvorschrift für γ der Fall.

2.4.2 Skalentypen

Abhängig von den Relationen, die es auf den Ausprägungen eines zu messenden Merkmals gibt, sind auf den Skalenwerten bestimmte Operationen möglich oder eben auch nicht möglich. Sind zum Beispiel die Merkmalswerte nicht additiv, so sind Additionen von Skalenwerten unzulässig, selbst wenn dies von der Art der Skala her möglich wäre. Abgestuft bezüglich der erlaubten Operationen werden fünf Skalentypen unterschieden (Tabelle 2.1).

TABELLE 2.1. Skalentypen

Skalentyp	Operationen	Beschreibung
Nominalskala	= ≠	Reine *Kategorisierung* von Werten. Die Skalenwerte sind weder vergleichbar noch miteinander verknüpfbar. Es ist nur nicht-parametrische Statistik möglich.
Ordinalskala	= ≠ < >	Die Skalenwerte sind *geordnet* und miteinander vergleichbar. Die Bestimmung des Medianwerts ist möglich. Im übrigen kann nur nicht-parametrische Statistik betrieben werden. Mittelwerte oder Standardabweichungen können *nicht* berechnet werden.
Intervallskala	= ≠ < > Distanz	Werte sind geordnet; die *Distanz* zwischen je zwei Skalenwerten kann bestimmt werden. Der *Nullpunkt* der Skala ist *willkürlich* gewählt. Mittelwert und Standardabweichung sind bestimmbar.
Verhältnisskala (auch Rationalskala genannt)	= ≠ < > Distanz, (+ –), Vielfaches, %	Werte sind geordnet und in der Regel *additiv*[1]. *Vielfache* und *Prozentwerte* sind bestimmbar. Die Skala hat einen *absoluten Nullpunkt*; dieser repräsentiert das totale Fehlen der gemessenen Eigenschaft. Übliche parametrische Statistik ist möglich.
Absolutskala	= ≠ < > Distanz, (+ –), Vielfaches, %	Die Skalenwerte sind *absolute Größen*, das heißt, sie lassen sich in keine andere Skala umrechnen. Im Übrigen gleiche Eigenschaften wie Verhältnisskala.

BEISPIEL 2.2. Skalentypen
Nominalskala: Testergebnisskala mit den Werten {erfüllt, nicht erfüllt, nicht getestet}
Ordinalskala: Eignungsskala mit den Werten {– –, –, o, +, ++}
Intervallskala: Datumskala für Zeit
Verhältnisskala: Anzahl-Codezeilen-Skala für Programmgröße
Absolutskala: Zählskala für die Anzahl gefundener Fehler in Programmen

2.4.3 Direkte und indirekte Maße

Es gibt einige interessierende Merkmale von Software und von Software-Entwicklungsprozessen, die sich in einfacher Weise direkt messen lassen. Solche Merkmale sind zum Beispiel Entwicklungskosten oder Durchlaufzeiten. Solche Maße nennen wir direkte Maße. Auch das in Beispiel 2.1 erwähnte Maß für die Größe von Programmen gehört zu den direkten Maßen.

Auf der anderen Seite gibt es Merkmale, für die es keine direkten Maße gibt oder wo das Messverfahren zu aufwendig wäre. In diesen Fällen behilft man sich, indem man *messbare Indikatoren* für das Merkmal bestimmt. Die Indikatoren sollen möglichst stark mit dem eigentlichen

[1] Die meisten in der Praxis vorkommenden Verhältnisskalen sind additiv. Additivität ist aber keine zwingende Eigenschaft.

zu messenden Merkmal korreliert sein. Die Indikatormaße bilden zusammen ein indirektes Maß für das interessierende Merkmal.

BEISPIEL 2.3. Messung der *Portabilität* (d.h. des Aufwands, ein System von einer bestehenden Hardware/Systemumgebung auf eine andere Hardware/Systemumgebung zu übertragen). Eine genaue Messung der Portabilität würde die Bestimmung des Quotienten t_{port}/t_{ent} erfordern, wobei t_{port} der Aufwand für die Portierung und t_{ent} der Aufwand für die Neuentwicklung auf der neuen Hardware/Systemumgebung ist. Diese Messung kostet jedoch ein Vielfaches von dem, was sie nützt. Als Ausweg kann man die Messung von Portabilitäts-Indikatoren heranziehen (Bild 2.3).

Indikator	Skala	Messverfahren	Planwert	Schwellwert
Anzahl BS-Aufrufe/Anzahl Prozeduraufrufe	0-100%	Zählen im Code	5%	10%
Anteil Hardware- oder BS-abhängiger Module	0-100%	Zählen im Code	10%	15%
Anteil Nichtstandard Codezeilen	0-100%	Zählen, Vgl. mit ISO-Standard	2%	5%

BILD 2.3. Messung von Portabilität mit Indikatoren (BS=Betriebssystem)

2.5 Wichtige Maße für Ziele in der Software-Entwicklung

Zur Messung von *Produktzielen* interessiert man sich vor allem für Maße zur Messung der Funktionserfüllung (beispielsweise die Erfüllung von Leistungsanforderungen), sowie der Größe, der Zuverlässigkeit, der Benutzerfreundlichkeit oder der Pflegbarkeit. Leider ist zu sagen, dass die meisten dieser Merkmale nur indirekt gemessen werden können und es bis heute keine allgemein anerkannten Sätze von Indikatoren für diese Merkmale gibt. Auch bei scheinbar einfachen Merkmalen wie der Produktgröße gehen die Meinungen auseinander, ob beispielsweise die Zahl der Codezeilen ein adäquates Größenmaß ist.

Im Bereich der Projektziele interessiert man sich insbesondere für Maße zur Messung von Aufwand, Durchlaufzeit (Dauer), Arbeitsfortschritt, Entwicklungskosten und Fehlerkosten. Im Gegensatz zu den Produktmerkmalen lassen sich die meisten Projektmerkmale verhältnismäßig einfach messen, wenn die notwendigen Basisdaten (zum Beispiel eine geeignete Zeitaufschreibung) vorliegen.

Aufgaben

2.1 Warum sind Zielsetzung und Zielverfolgung im Software Engineering besonders wichtig?

2.2 Geben Sie für die Skalentypen Nominalskala, Ordinalskala, Intervallskala, Verhältnisskala und Absolutskala je ein Beispiel aus dem täglichen Leben an.

2.3 In einem Unternehmen mit weltweiten Verkaufsniederlassungen sollen die Verkäuferinnen und Verkäufer durch ein zentrales Produktinformations- und Bestellsystem unterstützt werden. Dieses System soll
- den Verkäuferinnen und Verkäufern aktuelle Produktinformationen liefern (insbesondere auch Angaben über lieferbare Mengen und Lieferfristen)
- die Bestellungen der Verkäuferinnen und Verkäufer aufnehmen und an die Auftragsabwicklung weiterleiten
- Auskunft über Status und Liefertermin von Bearbeitung befindlichen Aufträgen geben.

Die Verkäuferinnen und Verkäufer, welche dieses System benutzen sollen, verlangen in der Zeit zwischen 8.00 Uhr und 18.00 Uhr eine sehr hohe Systemverfügbarkeit.
Nehmen Sie an, Sie hätten den Auftrag, für diesen Benutzerwunsch ein messbares Ziel zu formulieren. Wie gehen Sie vor und wie sieht Ihre Zielformulierung aus?

2.4 Sie sind Leiterin bzw. Leiter eines Entwicklungsprojekts für ein neues Standardsoftwarepaket zur Erstellung von Steuererklärungen. In einem Strategiepapier des Produkt-Marketings ist für dieses Softwarepaket eine «besonders benutzerfreundliche Bedienung» verlangt. Was tun Sie?

2.5 Eine Firma vertreibt Systeme, mit denen technische Prozesse (z.B. Produktionsstrassen, Kraftwerke, Wasserversorgungssysteme) bedient und der Prozesszustand erfasst und angezeigt werden. Ein Kunde, welcher ein solches Leitsystem betreibt, bestellt zusätzlich ein Programm zur statistischen Auswertung von laufend erfassten Betriebsdaten. Nehmen Sie an, Sie sind verantwortlich für dieses Projekt. Sie haben kürzlich ein ähnliches System für einen anderen Kunden entwickelt und installiert. Der neue Auftrag unterscheidet sich von dem zuvor abgewickelten Auftrag wie folgt:

- ca. 50% höheres Datenvolumen
- zusätzliche Auswertungen in graphischer Form (das Vorgängersystem erzeugte nur Tabellen)
- andere Hardware, anderes Betriebssystem.

Weitere Betreiber interessieren sich für Ihr Auswertungssystem; Sie rechnen für die nächsten beiden Jahre mit mindestens fünf Aufträgen der gleichen Art. Sie wissen ferner, dass es in Ihrer Firma nur wenig Know-How über graphische statistische Auswertungen gibt. Ihre Abteilungsleiterin hat Sie beauftragt, einen Vorschlag für die Projektziele vorzulegen und sich zu überlegen, wie Sie die Zielverfolgung sicherstellen.
Welche Projektattribute und welche besonderen Qualitäten für das Produkt schlagen Sie aufgrund der spezifischen Projektsituation vor? Wie wollen Sie die Erreichung der vorgeschlagenen Ziele kontrollieren?

Ergänzende und vertiefende Literatur

Briand, Morasca und Basili (1996) beschreiben grundlegende Eigenschaften wichtiger Produktmaße. In einem Vergleich mit anderen Ansätzen erschließt dieses Papier ferner weitere Literatur zur Fundierung des Messens im Software Engineering.

Fenton (1991) sowie Conte, Dunsmore und Shen (1986) sind Monographien über Messen im Software Engineering. Das Buch von Conte, Dunsmore und Shen ist leider vergriffen und nur noch über Bibliotheken zugänglich.

Zitierte Literatur

Briand, L.C., S. Morasca and V.R. Basili (1996). Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering* **22**(1). 68-85.

Conte, S.D., H.E. Dunsmore, V.Y. Shen (1986). *Software Engineering Metrics and Models.* Menlo Park, CA., etc.: Benjamin/Cummings.

Fenton, N.E. (1991). *Software Metrics. A Rigorous Approach.* London, etc.: Chapman&Hall.

Weinberg G., E. Schulman (1974). Goals and Performance in Computer Programming. *Human Factors* **16**, 1. 70-77.