

Halbformale versus formale Beschreibungen

Matthias Riebisch
FG Prozeßdatenverarbeitung
Fakultät Informatik und Automatisierung
Technische Hochschule Ilmenau
PO Box 327; O-6300 Ilmenau, Germany

Fax (+49) (3677) 69 1614 Tel (+49) (3677) 69 1463
E-Mail riebisch@PrakInf.TH-Ilmenau.DE

Zusammenfassung

Die Festlegung eines Beschreibungsmittels zur Benutzung in Informations- und Entwurfssystemen stellt eine grundlegende Entscheidung über Mächtigkeit, Eignung und Akzeptanz eines solchen Software-Systems dar. Meist wird aufgrund der formalen Auswertbarkeit formalen Beschreibungen der Vorzug gegeben. Sollen sehr komplexe oder allgemeine Sachverhalte ohne hohe Sicherheitsforderungen beschrieben werden, kommt es häufig zu Problemen beim Einsatz, z.B. wegen hoher Komplexität und geringer Übersichtlichkeit der Darstellungen, sowie hohem Lernaufwand und geringer Akzeptanz beim Nutzer. Die Aufnahme informaler Teile in formale Beschreibungen kann in solchen Fällen Vorteile bieten, zumal für diese Teile mit den Mitteln des Information Retrieval auch geeignete Auswertungsmethoden zur Verfügung stehen.

Einleitung

Vor nunmehr etwa zwanzig Jahren wurde die Softwarekrise konstatiert, seitdem arbeitet man gezielt an der Einführung softwaretechnologischer Unterstützung in die Programmentwicklung. Zahlreiche der entwickelten Methoden fanden nur geringe Akzeptanz bei den Anwendern, häufig ist sogar eine generelle Ablehnung der Anwender gegenüber Methoden-Einführungen zu bemerken. Mit dem Ziel der Verbesserung dieser Situation werden immer neue Methoden (und Werkzeuge dazu) entwickelt.

Auf der anderen Seite sind Anwender auf der Suche nach Werkzeugen, die bei der Lösung ihrer Probleme Unterstützung leisten. Ein Ausweg liegt in der verbesserten Anpassung der Werkzeuge an die Anwender. Die Anstrengungen bei der Weiterentwicklung von Werkzeugen der Softwaretechnik und ihrer praktischen Anwendung beachten zunehmend die Erfordernisse der Ergonomie, insbesondere bei der Gestaltung der Werkzeuge. Nicht nur für Oberflächen und Werkzeuge ist die Ergonomie von großer Bedeutung. Auch die Methoden und die ihnen zugrunde liegenden Beschreibungsmittel müssen an die Fähigkeiten und Kenntnisse der Nutzer angepaßt sein; sie müssen in der täglichen Programmierpraxis beherrscht werden können.

Für zahlreiche Aufgaben, die auf einem von Details abstrahierenden Niveau gelöst werden müssen, werden Lösungen statt durch Anwendung formaler, algorithmischer Methoden mit den Mitteln der KI gesucht. Eine der Ursachen dafür dürfte in der zu geringen Ausdrucksmächtigkeit für verallgemeinerte Informationen und der zu geringen Flexibilität formaler Beschreibungsmittel liegen.

Die Versuche der KI-Nutzung sind nur zu einem kleinen Teil von Erfolg gekrönt.

Durch eine Erweiterung formaler Beschreibungen um informale Teile wie z.B. Text ist in einigen Fällen eine Lösung möglich, die einerseits gut verständliche und handhabbare Darstellungen ermöglicht, andererseits ausreichend gut auswertbar ist.

Anforderungen an Beschreibungen

Methoden beispielsweise in Entwurfssystemen basieren auf Beschreibungsmitteln. Die Implementierung dieser Methoden stellen den Kern entsprechender Werkzeuge dar. Die Festlegung dieser Beschreibungsmittel ist eine grundlegende Entscheidung, die über Einsatzbreite, Anwendbarkeit und Akzeptanz einer Methode und eines Werkzeugs bestimmt.

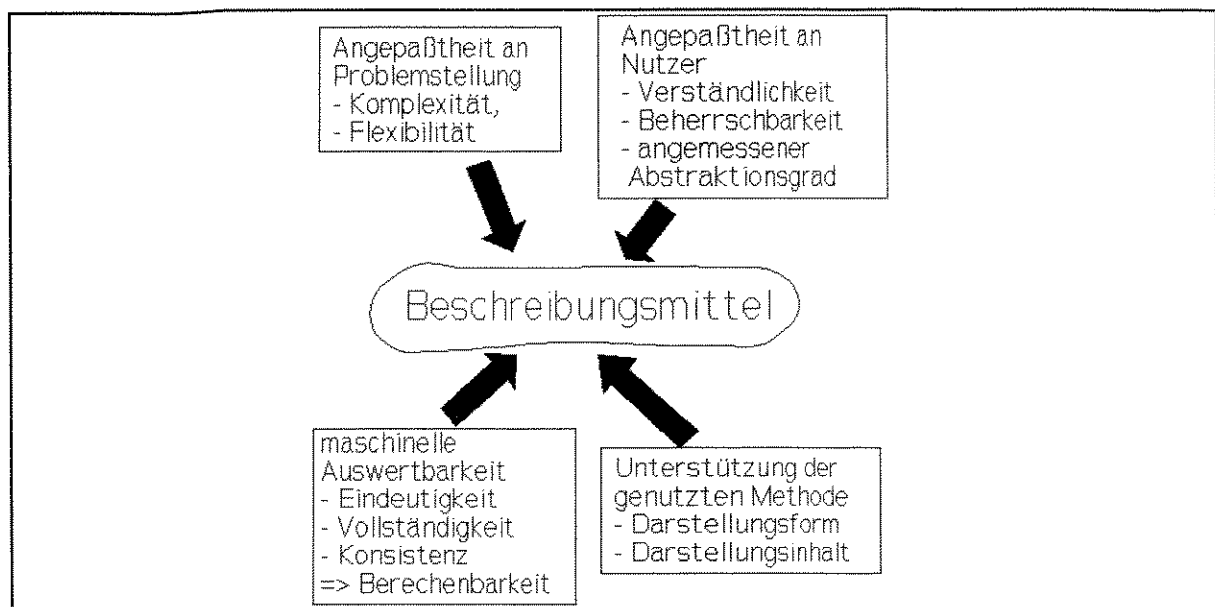


Abb. 1 Anforderungen an Beschreibungsmittel

Was für Anforderungen muß eine solche Beschreibung erfüllen?

Im wesentlichen gehören dazu

1. eine adäquate Darstellung des Problemraums,
2. Verständlichkeit für den Benutzer,
3. eine weitgehende Auswertbarkeit und
4. Unterstützung der gewählten Methodik der Problemlösung.

zu 1. adäquate Darstellung des Problemraums

Die Beschreibung muß dazu geeignet sein, die relevanten Eigenschaften des Beschreibungsobjekts abzubilden, die im Fall von Software sehr vielfältig sind. Dazu gehören beispielsweise Eigenschaften wie Funktionalität, Datenstrukturen, Ablaufstruktur, Aufbau der Schnittstelle, Beziehungen zur rechentechnischen Umgebung und zum Anwendungsgebiet bzw. -ziel der Software.

Stellt man die Eigenschaften gegenüber und vergleicht ihren Gehalt, so zeigen sich große Unterschiede bezüglich Abstraktionsgrad, Struktur-Aussagen und Kontextabhängigkeiten. Einige Beschreibungsteile können ähnlich einer Black Box weitgehend auf

Details der Struktur verzichten, andere müssen die Struktur, Wirkungsweise und die Beziehungen der Beschreibungsobjekte im Sinne einer White Box präzise darstellen. Der Abstraktionsgrad und der Anteil allgemeiner Informationen ist für die zuerst genannten Teile wesentlich höher als für die zweiten. Die Beschreibung muß solche unterschiedlichen Abstraktionsgrade ausdrücken können.

Wird die Beschreibung aus den unter 2. genannten Gründen der Ergonomie in Sichten und Abstraktionsebenen gegliedert, entsteht die Notwendigkeit, Beziehungen zwischen diesen Sichten darzustellen. Bezogen auf eine Sicht der Beschreibung stellt eine Beziehung zu einer anderen Sicht jeweils eine Kontextabhängigkeit dar.

zu 2. Verständlichkeit für den Benutzer

Als Aspekte der Ergonomie sind besonders der Lernaufwand zur Beherrschung der Beschreibung und die kognitiven Fähigkeiten der Menschen bezüglich der Erfäßbarkeit zu berücksichtigen.

Um den Lernaufwand bei der Einführung einer Beschreibung gering zu halten, sollte diese zu gebräuchlichen Beschreibungsmitteln ähnlich sein. Außerdem sollte das ihr zugrundeliegende System bzw. die Methodik insofern "reif" sein, als es nicht zu viele theoretische Kenntnisse vom Nutzer verlangt. (Viele Menschen sind nach Meinung des Autors zu stark abstraktem Denken nicht fähig bzw. haben es nicht gelernt.)

Formale Algorithmen, die eine automatische Generierung formaler Semantikbeschreibungen aus informalen Angaben zulassen, sind bisher nicht bekannt. Es ist fraglich, ob sie überhaupt existieren.

Zu den zu berücksichtigenden kognitiven Fähigkeiten gehört die Erfäßbarkeit von Darstellungen durch den Menschen. Er kann Beschreibungen dann verstehen, wenn sie in einem gewissen Maß homogen sind, d.h. wenn ihre Darstellung nicht zu viele unterschiedliche Elemente und Darstellungsformen enthält. Die Komplexität von Software ist beschreib- und beherrschbar, wenn verschiedene Sichten (Ebenen) der Betrachtung eingeführt werden. Es hat sich als günstig erwiesen, solche Sichten einmal anhand von Eigenschaften und Nutzungsaspekten und zum anderen entsprechend geeigneter Abstraktionsebenen zu bilden.

Die Gliederung einer Beschreibung in verschiedene Abstraktionsebenen wie Gesamt-Aufgabenstellung, einzelne Entwurfsentscheidungen und Details der Struktur ist erforderlich, um Überschaubarkeit im Sinne von Homogenität der Beschreibung zu erreichen. Eine Erklärung mittels Funktionen jedoch kann nur verständlich sein, wenn diese Funktionen vom Abstraktionsgrad her auf dem Niveau der zu beschreibenden Funktionalität liegen. Es müssen folglich Beschreibungsebenen existieren, auf die sich die Beschreibung bezieht. Eine Darstellung der Funktionalität eines Programmsystems z.B. auf der Ebene von Prozessorbefehlen ist nicht erfäßbar.

Im Fall der Beschreibung von Software bestehen besondere Forderungen an die Festlegung solcher Beschreibungsebenen. Sie müssen genügend statisch sein, um als Basis einer Beschreibung dienen

zu können und um erlernbar zu sein. Andererseits sind viele Ebenen der Software variabel. Selbst Betriebssystemelemente können ggf. modifiziert und ergänzt werden, wie auch Teile der Datenhaltung, der Kommunikationsdienste, der Nutzerschnittstelle sowie von Anwenderprogrammen.

Bei der Wahl von Beschreibungsebenen muß ein Optimum erreicht werden zwischen Stabilität, Flexibilität gegenüber Änderungen und leichter Erfassbarkeit durch den Menschen.

zu 3. weitgehende Auswertbarkeit

Beschreibungsmittel stellen die Basis für die Anwendung von Methoden dar. Um die Methodenabarbeitung zu automatisieren oder wenigstens zu unterstützen, müssen die verwendeten Beschreibungsmittel einer maschinellen Auswertung möglichst gut zugänglich sein. Der geforderte Grad an Eindeutigkeit und Widerspruchsfreiheit ist je nach Beschreibungsziel unterschiedlich. Für Verifikation und Simulation ist er wesentlich höher als für Suche und Klassifikation.

Die Auswertbarkeit muß auf der gewählten Beschreibungsebene gegeben sein. Eine Überführung der Beschreibung auf eine tiefere, mehr elementare Beschreibungsebene zum Zweck der Auswertung würde eine adäquate Abbildung des Problems verhindern.

zu 4. Unterstützung der gewählten Methodik der Problemlösung

Da Beschreibungsmittel und Methodik unmittelbar zusammengehören, muß ein Beschreibungsmittel vom Gehalt und der Strukturierung der Informationen her die Methodik unterstützen. Ist beispielsweise die zu beschreibende Software objektorientiert strukturiert, müssen sich Vererbungs- und Benutzungsbeziehungen beschreiben lassen.

Diese Anforderungen widersprechen sich teilweise. Insbesondere zwischen 2. und 3. gilt es, ein Optimum zu finden.

In welcher Weise erfüllen formale Beschreibungen diese Anforderungen?

Einschätzung formaler Beschreibungen

Formale Beschreibungsmittel wie z.B. funktionale Algebren und Petri-Netze bieten jeweils Ausdrucksmittel auf relativ elementarer Ebene. Beschreibungen auf höherer Ebene sind zwar möglich, es wurde jedoch noch kein Standard ihrer Benutzung entwickelt, der eine Beschreibungsebene für allgemeinere Beschreibungen zur Verfügung stellt. Die Darstellung von Kontextabhängigkeiten verkompliziert die Darstellung, statt den Überblick für den Betrachter zu erleichtern.

Noch wichtiger als gebräuchliche Ausdrucksformen ist, daß Auswertungs- und Analysemöglichkeiten auf höheren Beschreibungsebenen bisher nicht unterstützt werden. Damit wird der Vorzug der formalen Beschreibung aufgegeben.

Soll Software beispielsweise mittels Petrinetzen beschrieben werden, so sind Darstellungen des Steuerflusses (d.h. von Programmzuständen mit Synchronisations- und Zeitbedingungen), des Datenflusses, der Ressourcen-Nutzung, der Abhängigkeit zwischen

Systemzuverlässigkeit und Struktur, sowie von weiteren Eigenheiten möglich.

Solche Darstellungen müssen jedoch getrennt erfolgen, da die Ausdrucksmittel wie Marken und Transitionen jeweils unterschiedliche Sachverhalte repräsentieren. Um einen Überblick über die Zusammenhänge zu ermöglichen, sind Querverbindungen zwischen den Teil-Darstellungen erforderlich. Solche Querverbindungen sind mit den eigenen Mitteln der Petri-Netze nicht darstellbar; Erweiterungen haben den Nachteil, formalen Analysen nicht zugänglich zu sein. Für höhere Netzformen wie z.B. Prädikat-Transitions-Netze stehen weniger Analysemöglichkeiten zur Verfügung; eine Überführung in Standard-Petrinetze zerstört jedoch die adäquate Abbildung der Problemstellung oder -lösung.

Bei der Anwendungen mit hohen Sicherheitsforderungen treten die Vorteile wie Verifizierbarkeit stark in den Vordergrund. In zahlreichen Fällen ohne Sicherheitsforderungen behindern beispielsweise

- der Aufwand der Erstellung der formalen Beschreibung und
- die Unübersichtlichkeit
 - aufgrund schwer erkennbarer Zusammenhänge zwischen einzelnen Beschreibungsteilen und
 - aufgrund geringer Ähnlichkeit zwischen formaler Darstellung und Problem

den praktischen Einsatz formaler Beschreibungen.

Halbformale Beschreibungen - Aufbau und Auswertung

Worin besteht nun ein optimales Beschreibungsmittel, das gut erfaßbar und leicht erlernbar ist, das komplexe Eigenschaften von Software darstellen kann und weitgehend maschinell auswertbar ist?

Informale Beschreibungen wie natürliche Sprache und freie Grafiken bieten keine Alternative - ihre Syntax und Semantik ist völlig unbeschränkt und läßt eine Auswertung von Mehrdeutigkeiten nicht mit vertretbarem Aufwand zu. Ihre Erschließung hängt in sehr hohem Maße von Kontextbedingungen wie Begriffsverständnis, Vorkenntnissen und Verständnis des Autors der Beschreibung ab.

Der hier vorgestellte Ansatz besteht deshalb darin, in eine formal definierte Beschreibung informale Teile aufzunehmen und diese maschinell auszuwerten¹. Die Beschreibung wird dadurch durch den Menschen leichter erstell- und erfaßbar und flexibler in den Ausdrucksmöglichkeiten bezüglich ihrer Semantik.

Mit formalen Mitteln wird ein Rahmen definiert. In diesem Rahmen werden Einträge untergebracht, deren Semantik durch Bezug auf den Rahmen definiert ist. Die Einträge können formalen oder auch informalen Charakter haben, wenn das erforderlich ist: Einzel-

¹ Einige bekannte Beschreibungsmittel für Entwurf und Spezifikation enthalten ebenfalls informale Teile, wie z.B. EPOS-S [Lau90] mit 'PURPOSE:'. Diese Einträge dienen meist zur Aufnahme derjenigen Informationen, die in das Schema der Beschreibung anders nicht aufgenommen werden können. Hier geht es jedoch darum, diese Teile zu strukturieren und auszuwerten.

begriffe, Wortgruppen oder Text. Da jeder informale Eintrag innerhalb des Rahmens eine definierte Semantik besitzt, ist eine Auflösung von Mehrdeutigkeiten weitgehend möglich.

Die Auswertung der informalen Einträge ist mit den Mitteln des Information Retrieval möglich [SMG87] [Lag89] [Dtz89]. Dabei wird nicht der Sinngehalt oder die Bedeutung des Textes analysiert, sondern lediglich die Verwendung sinntragender Begriffe. Auf diese Weise ist beispielsweise ein Vergleich, eine Ähnlichkeitsbewertung und eine Inhaltserschließung möglich.

Bei der automatischen Auswertung von Patentbeschreibungen z.B. werden im Text Begriffe gesucht und für sie ein Wichtungsfaktor berechnet. Die so gebildete Menge von Begriffen mit Wichtungsfaktor (der sog. Deskriptorvektor) beschreibt den Inhalt (die Semantik) in einer Weise, die für automatische Klassifikation und Retrieval ausreichend auswertbar ist.

Trotz der Strukturierung mittels formalem Rahmen ist die Beschreibung leicht handhabbar und flexibel genug, um eine gewisse Variationsbreite an Abstraktionsgraden und z.B. Anwendungsgebieten der Software zuzulassen. Die Auswertbarkeit ist um so größer, je besser die Gliederung des Rahmens der Struktur des Problems und den Eigenschaften des Beschreibungsobjekts angepaßt ist.

Wie sieht nun die Anwendung einer solchen Beschreibung konkret aus?

Beispiel einer halbformalen Beschreibung für Retrieval und Klassifikation wiederverwendbarer Software-Komponenten in Entwurfssystemen

Bei der Wiederverwendung von Software werden wiederverwendbare Teile, sog. Komponenten, in einer Bibliothek gesammelt. Nach Spezifikation, Grob- und Feinentwurf sucht der Entwerfer Komponenten in der Bibliothek, die zur Implementierung seines Entwurfs beitragen. Er wählt aus einer Rangfolge nach Eignung geordneter Komponenten eine aus, paßt sie an und fügt sie zur entstehenden Software hinzu.

Neue Komponenten entstehen durch Verallgemeinerung bereits getesteter und weitgehend fehlerbereinigter Softwareteile. Sie werden in die Bibliothek eingeordnet, d.h. klassifiziert. Die korrekte Klassifikation ist von sehr großer Bedeutung für effektives Retrieval. Das Retrieval ist dann effektiv, wenn der Anteil gefundener an der Gesamtmenge geeigneter Komponenten hoch ist und dabei nur wenige ungeeignete Komponenten mit nachgewiesen werden.

Die Komponenten der Bibliothek werden jeweils durch eine Beschreibung zusätzlich zum Code und zur Dokumentation repräsentiert, der sog. Komponentenbeschreibung². Eine solche Komponen-

² In anderen Ansätzen werden zur Beschreibung von Software-Teilen auch freie, natürlichsprachliche Texte genutzt, z.B. [Maa91] [ArS87] [BAB87]. Der Auswertungsaufwand ist jedoch dann wesentlich höher, der Auswertungsgrad ist aufgrund fehlender semantischer Bezüge zwangsläufig niedriger.

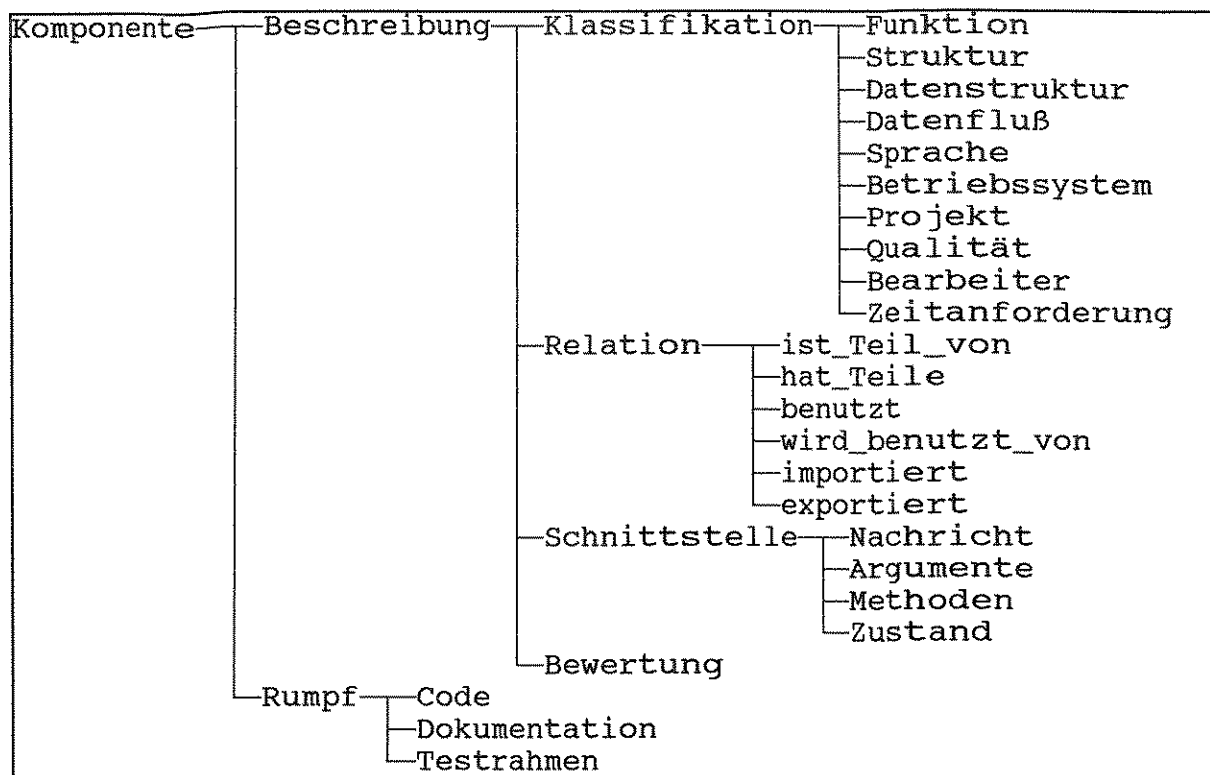


Abb. 2 Gliederung der Komponentenbeschreibung

tenbeschreibung kann auch zur Formulierung einer Suchanfrage und zur Beschreibung zwecks Klassifikation einer neuen Komponente genutzt werden³. Die Komponentenbeschreibung stellt damit ein zentrales Dokument in Retrieval und Klassifikation dar (Abb. 3).

Abb. 2 zeigt die Gliederung der Komponentenbeschreibung, Abb. 4 ein konkretes Beispiel.

Die Beschreibung der wiederverwendbaren Software-Komponenten dient dazu, ihre relevanten Eigenschaften in einer für Vergleiche geeigneten Form zu repräsentieren.

Der Grad der Detail-Abstraktion ist bei diesen Informationen unterschiedlich hoch. Eigenschaften, für deren Beschreibung definierte Bezeichner zur Verfügung stehen, wie

Programmiersprache, Betriebssystem, Datenbank- und Kommunikationsdienste, benötigte Ressourcen (Hard- und Software, Zeit), Grad der Parallelität, bei objektorientierter Gliederung der Komponenten auch Vererbungsbeziehungen, Instanzierungsbeziehungen, Aufruf- bzw. Benutzungsbeziehungen und Enthaltenseinsbeziehungen

werden formal beschrieben. Zu ihrer Beschreibung werden Einträge geschaffen, die jeweils einen oder mehrere definierte Bezeichner aufnehmen. Diese Bezeichner können z.B. einen Verweis auf andere Komponenten oder in der Klassifikation enthaltene Merkmalsklas-

³ Außerdem benötigt man eine solche Komponentenbeschreibung noch für Modifikation, Integration und Test der Komponenten im Zuge der Implementierung, was jedoch wegen des unterschiedlichen Charakters der benötigten Informationen hier nicht betrachtet werden soll.

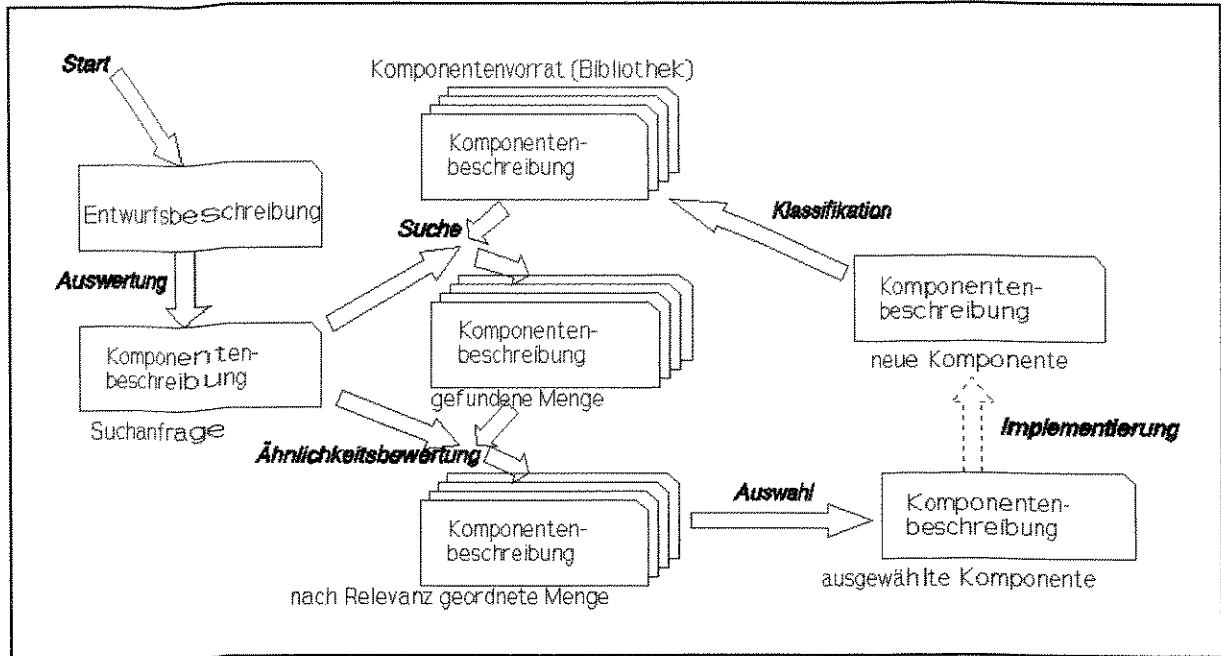


Abb. 3 Nutzung der Komponentenbeschreibung

sen darstellen.

Für Eigenschaften, deren Beschreibung einen hohen Grad an Detail-Abstraktion erfordern, wie Funktionalität und Architekturmerkmale, enthaltene Entwurfsentscheidungen, Komplexität, aber auch Grundstrukturen von Datenfluß, Steuerfluß und enthaltener Datenstruktur, sind in den meisten Fällen keine definierten Bezeichner oder andere formale Beschreibungen in anwendbarer Form verfügbar. Sie werden dann mit informalen Mitteln wie beliebigen Wörtern oder Text ausgedrückt.

Die Gliederung der Beschreibung erfolgt durch einen Rahmen aus Schlüsselwörtern, der in Abb. 2 dargestellt ist. Die Schlüsselwörter folgen einer Syntaxdefinition. Die Semantik ist durch die jedem Eintrag zugeordneten Auswertungsmethoden definiert; außerdem existiert eine Anwenderbeschreibung dazu.

Die Semantik formaler Einträge zur Beschreibung von Eigenschaften zwecks Klassifikation und Retrieval besteht z.B. darin, daß verwendete Bezeichner Verweise auf gleichnamige Merkmalsklassen in der Bibliothek darstellen.

Die Semantik informaler Einträge wird definiert

- durch Festlegung der Auswertungsschritte für die Gewinnung sinntragender Begriffe,
- durch Festlegung der Referenz-Begriffsmengen für Wichtung und Vergleich sowie
- durch Festlegung der Nutzung der gebildeten Deskriptorvektoren.

Die Einträge sind entsprechend ihrer Verwendung durch den Rahmen zu Gruppen zusammengefaßt. Die Informationen der Gruppen "Relation" und "Schnittstelle" dienen für Modifikation und Integration.

```

Beschreibung:
  Klassifikation:
    Funktion: "Die Suchanfrage ist eine Deskriptorliste, deren
      Deskriptoren alle ein Gewicht haben. Sie verfügt über
      Methoden, um aus einer Clustermenge bzw. einem Cluster
      Ranglisten relevanter Cluster bzw. Komponenten her-
      auszusuchen. Sie verfügt über das Cosinus-Ähnlich-
      keitsmaß und über das Maß der Überlappung."
    Struktur: Funktionsbaustein
    Datenstruktur: Cluster, Dokument, Rangliste, Deskriptor-
      liste, Set 'Cluster'
    Datenfluß: 'Cluster'-'Array'
    Sprache: Smalltalk-80
    Betriebssystem: UNIX-V/7.02
    Projekt: RESQUE
    Qualität: verifiziert
    Bearbeiter: Franke
    Zeitanforderung:
  Relation:
    ist Teil von: Sucher
    hat Teile: ...
    ist Subklasse von: DeskriptorListe
    hat Subklassen: -
    ist Instanz von: -
    hat Instanzen:
    benutzt: GefundenesDokument, RangListe
    wird benutzt von: Clustermenge, Sucher
  Schnittstelle: ...
  Bewertung: ...
  ...

```

Abb. 4 Beispiel einer Komponentenbeschreibung für eine Komponente 'SuchAnfrage' (Ausschnitt)

Eigenschaften der Komponenten zwecks Suche, Ähnlichkeitsbewertung und Auswahl der Software-Komponente sind in der Gruppe "Klassifikation" beschrieben.

Für jede Eigenschaft, die als Suchkriterium benötigt wird, ist in der sog. Facettenklassifikation [DIN87] [PrD86] eine Facette vorhanden, in der alle Komponenten ihren Eigenschaften entsprechend einer Hierarchie von Merkmalsklassen zugeordnet sind (Abb. 5).

Jeder Eintrag im Teil "Klassifikation" der Komponentenbeschreibung (Abb. 2) verweist auf eine Facette der Klassifikation (Abb. 5). Der Inhalt des Eintrags bezeichnet diejenige Merkmalsklasse der betreffenden Facette, der die Komponente zugeordnet ist. Steht z.B. in einer Komponentenbeschreibung unter "Klassifikation"/"Sprache" der Bezeichner "BC++", so ist die betreffende Komponente in der Facette "Sprache" in der Merkmalsklasse "BC++" eingeordnet. Solche Bezeichner stellen damit ähnlich dem Hypertext-Prinzip Beziehungen zwischen Komponentenbeschreibung und Klassifikation der beschriebenen Komponente in der Bibliothek her.

Für die Komponenteneigenschaften, für die eine informale Beschreibung erforderlich ist, existieren in der Bibliothek unscharfe Merkmalsklassen zur Klassifikation der Komponenten. Unscharf bedeutet hier, daß kein eindeutiges Kriterium exi-

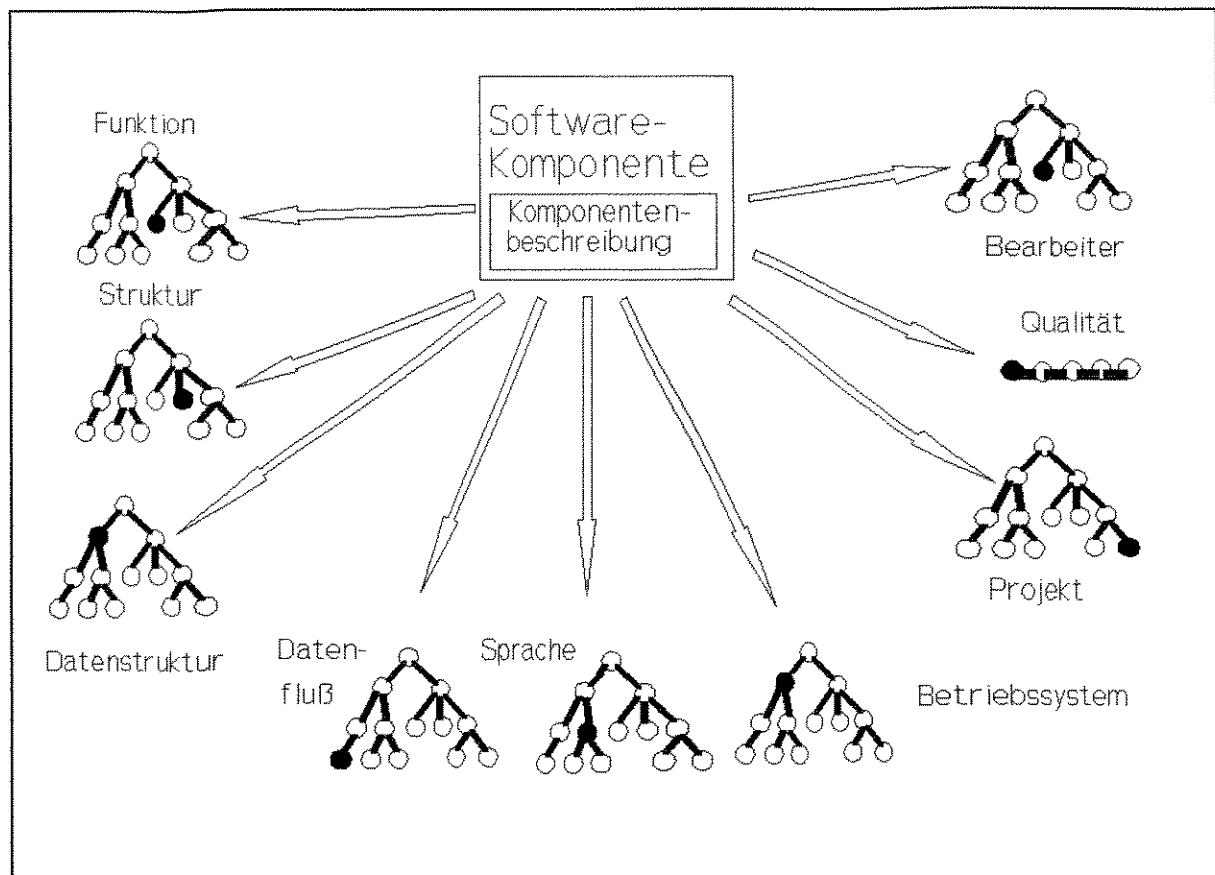


Abb. 5 Klassifikation der Komponenten in der Bibliothek

stiert, ob eine Komponente zu einer Merkmalsklasse gehört oder nicht. Bei Eigenschaften wie Funktionalität oder Architekturmerkmalen beispielsweise ist die eindeutige Bildung von Merkmalsklassen nicht möglich, oder sie würde zu Widersprüchen bei der Einordnung neuer Komponenten in die Klassifikation führen. Zur Bildung unscharfer Merkmalsklassen werden Cluster [siehe SMG87] benutzt, die sich auch in Form einer Hierarchie anordnen lassen. Eine Clusterstruktur läßt sich mit Hilfe der Clusteranalyse aufbauen [Crc72] [Pan86].

Ein Cluster, das eine Merkmalsklasse repräsentiert, wird durch einen Deskriptorvektor beschrieben. Bei Suche, Ähnlichkeitsvergleich und Klassifikation (d.h. Clusteranalyse) wird dieser Deskriptorvektor mit demjenigen verglichen, der aus einem informalen Eintrag einer Komponente gewonnen wurde.

Nutzung der halbformalen Komponentenbeschreibung in einem Werkzeug zur Wiederverwendung von Software

Im Rahmen eines Projekts zur Suche, Klassifikation, Bewertung und Auswahl wiederverwendbarer Software-Komponenten [Rie91] [RKU91] wird derzeit ein Prototyp eines Wiederverwendungs-Tools geschaffen, das auf dem dargestellten Beschreibungsmittel basiert.

Zentraler Teil des Werkzeugs ist die Bibliothek wiederverwendbarer Software-Komponenten. Sie besteht aus einem Datenbank-Teil, der die Komponenten enthält, und einer Facettenklassifikation, die die Ordnung der Komponenten nach ihren Eigenschaften verwaltet.

Die für Suche, Ähnlichkeitsbewertung und Klassifikation benutzten Komponentenbeschreibungen werden aus bereits vorhandenen Spezifikationen und Entwurfsbeschreibungen gewonnen. Deren Auswertung wird von einem auf ihre Definition zugeschnittenen Übersetzer vorgenommen. Mit Hilfe konventioneller Scanner und Parser erfolgt die Überführung in eine Komponentenbeschreibung. Außerdem werden dabei auch die informale Beschreibungsteile ausgewertet und Deskriptorsätze gebildet.

Ein Sucher wertet die Verweise und Deskriptorsätze der Komponentenbeschreibungen aus und berechnet Ähnlichkeitsangaben für jede Facette der Klassifikation.

Von einem Bewerter werden die einzelnen Ähnlichkeitsangaben entsprechend des Bewertungsschlüssels zu einer Ähnlichkeits-Rangfolge zusammengeführt, aus welcher der Nutzer eine geeignete Komponente auswählt. Der Sucher ermöglicht dabei eine manuelle Suche und Auswahl aus den angebotenen Suchergebnissen und innerhalb der Klassifikation.

Für die Neuaufnahme von Komponenten in die Klassifikation wird ein Clusteranalysator benutzt. Er führt die Zuordnung neuer Komponenten zu Clustern und bei Bedarf die Neuorganisation der Clusterstruktur aus.

Im Zuge des schrittweisen Aufbaus des Prototyps läßt sich bereits abschätzen, daß die entwickelte Komponentenbeschreibung ein effektiv benutzbares Beschreibungsmittel zum Wiederfinden und Wiederverwenden darstellt.

Danksagung

Ich danke Prof. D.L. Parnas für die Anregung zu diesem Paper und meinem Kollegen Herwig Unger für seine Diskussionsbereitschaft und die kritische Durchsicht des Manuskripts.

Literatur

[Ars87]

Arnold, S.P.; Stepoway, S.L.: The reuse system: Cataloging and retrieval of reusable software. In [Trc87] S.138-141.

[BAB87]

Burton, B.A.; Aragon, R.Wienk; Bailey, S.A.; Koehler, K.D.; Mayes, L.A.: The Reusable Software Library. in [Trc87] S.129-137.

[Crc72]

Crouch, Donald B.: A clustering Algorithm for Large and Dynamic Document Collections. PhD Thesis. Southern Methodist Univ. Dallas: 1972.

[Dtz89]

Dietze, Joachim: Einführung in die Informationslinguistik - Die Linguistische Datenverarbeitung in der Informationswissenschaft. Leipzig: Verlag Enzyklopädie, 1989.

[DIN87]

-: DIN 32 705 / Erstellung und Weiterentwicklung von Klassifikationssystemen. Ausgabe 01.87.

- [FrN87]
Frakes, W.B.; Nejme, B.A.: Software Reuse Through Information Retrieval. in: Proc. of the 20th Annual HICSS. Kona, HI, 1987. S. 530-535.
- [Lag89]
Langbein, Dierk: Zum Aufbau wissensbasierter Informationssysteme in Forschung und Entwicklung. Ilmenau; Suhl: TH Ilmenau, 1989.
- [Lau90]
Lauber, R.[Hrsg.]: EPOS-Kurzbeschreibung. GPP mbH, Oberhaching, 1990.
- [Maa91]
Maarek, Yoëlle S.: An Information Retrieval Approach for Building Reuse Libraries. in: Proceedings First Int. Workshop on Software Reusability. SWT-Memo, Memorandum des Lehrstuhls Software-Technologie, FB Informatik, Univ. Dortmund, 3.-5. Juli 1991, S. 248-255.
- [Pan86]
Panyr, Jiri: Automatische Klassifikation und Information Retrieval. Anwendung und Entwicklung komplexer Verfahren in Information-Retrieval-Systemen und ihre Evaluierung. Tübingen: Max Niemeyer, 1986
- [PrD86]
Prieto-Diaz, Ruben: A Software Classification Scheme. PhD Thesis. Dep. of Information and Computer Science, Univ. of California at Irvine, 1985. Ann Arbor: Microfilms Int., 1986.
- [Rie91]
Riebisch, Matthias: Klassifikation und Suche halbformal beschriebener Software-Komponenten zum Zweck ihrer Wiederverwendung. in: Zorn, Werner; Bender, Klaus [Hrsg.]: Software- und Datenbank-Management / Transputer-Architekturen und -Anwendungen / TOOL91, RISC91. Tagungsband, 2. int. Fachmesse und Kongreß in Karlsruhe von 26.-28. Nov. 91. - Berlin, Offenbach: vde-verlag, 1991. S. 387-398.
- [RKU91]
Riebisch, Matthias; Krannich, Ralf; Stoinski, Daniel; Usbeck, Christian: Implementierung von Algorithmen zur Deskriptorgewinnung. Forschungsbericht. TH Ilmenau, Fakultät Informatik und Automatisierung, Fachgebiet Rechnerarchitekturen, Dezember 1991.
- [SMG87]
Salton, Gerard: Information Retrieval - Grundlegendes für Informationswissenschaftler. Hamburg: McGraw-Hill, 1987.
- [Trc87]
Tracz, W. (Ed.): Software Reuse - emerging technology. Computer Society Press, 1987.

eingereicht am 8. Juli 1992