

Terminologie der Softwaretechnik

Ein Begriffssystem für die Analyse und Modellierung von Anwendungssystemen Teil 2: Tätigkeits- und ergebnisbezogene Elemente

W.Hesse¹, G.Barkow², H.von Braun³, H.-B.Kittlaus⁴ und G.Scheschonk⁵

¹ Universität Marburg

² Volksfürsorge, Hamburg

³ MSP, München

⁴ IBM, Stuttgart

⁵ C. I. T., Berlin

Zusammenfassung. Dieser zweiteilige Artikel stellt das Arbeitsergebnis eines Arbeitskreises zur Terminologie der Softwaretechnik dar. Der Arbeitskreis hatte das Ziel, ein fundiertes, abgestimmtes und konsistentes Begriffssystem für die Analyse und die Modellierung von Anwendungssystemen (d. h. die „frühen Phasen“ der Softwaretechnik) aufzustellen und dabei auf terminologischen Vorarbeiten aufzubauen. Im ersten Teil haben wir Klassifizierungskriterien für Begriffe zusammengestellt, Begriffe eingeordnet und eine Begriffssystematik aufgestellt. Weiterhin wurden Grundbegriffe aus verwandten und zugrundeliegenden Gebieten aufgeführt. Im zweiten Teil werden zunächst die Tätigkeiten und Ergebnisse der Analyse und Modellierung dargestellt, anschließend die Definitionen für Elemente von Anwendungsmodellen wie Entität, Attribut, Beziehung, Funktion usw. Einige Bemerkungen zu gegenwärtigen Trends und möglichen künftigen Weiterentwicklungen des Gebiets schließen den Artikel ab.

Schlüsselwörter: Softwaretechnik, Terminologie, (System-) Analyse, Modellierung, Software-Anwendungssystem, Anwendungsmodell, aktive, passive und zusammenhangsbildende Elemente, Projektmanagement, Qualitätssicherung

Summary. This bipartite article presents the results of a task group concerned with software engineering terminology. This work (based on previous terminology work) aimed at building a well-founded, consolidated, consistent system of concepts for the so-called early phases of software engineering, i. e., the analysis and modelling stages of the application development process. In the first part, we concentrated on methodological and conceptual foundations. Concepts were classified according to the selected criteria and a conceptual framework was established. In the second part, main activities and results of the analysis and modelling stages are presented. Furthermore, important elements of application models such as entity, attribute, relationship, function etc. are defined. Some remarks on present trends and a possible further development of the considered area conclude the article.

Key words: Software engineering terminology, (System) analysis, Modelling, Software application system, Application

model, Active, passive and network elements, Project management, Quality assurance

Computing Reviews Classification: D.2, A.2

4. Tätigkeiten und Ergebnisse der Analyse und Modellierung

Der gesamte Entwicklungsprozeß für die Herstellung eines Anwendungssystems besteht aus einer Menge von Tätigkeiten, in denen jeweils spezifische Ergebnisse erstellt werden. Jede Tätigkeit benötigt eine bestimmte Menge von Informationen, die wiederum als Ergebnisse anderer Tätigkeiten zu liefern sind und somit eine logische Abhängigkeit der Tätigkeiten untereinander begründen [29]. So entsteht aus Tätigkeiten und Ergebnissen ein Netz, das aufgrund seiner Komplexität noch nicht für die Planung und Steuerung des Entwicklungsprozesses verwendet werden kann.

Durch eine Gruppierung der Tätigkeiten nach sachlogischen oder ergebnisorientierten Gesichtspunkten wird eine bessere Handhabbarkeit erreicht. Derartige Gruppierungen finden sich in sogenannten Vorgehens- und Phasenmodellen. Dabei dienen Phasen dazu, den gesamten Entwicklungsprozeß mit definierten Zwischenergebnissen und präzisen Prüfpunkten (Meilensteinen) in planbare und kontrollierbare Einheiten zu zerlegen. Vorgehensmodelle dienen der Steuerung des Entwicklungsprozesses mit dem Ziel, Anwendungssysteme so wirtschaftlich wie möglich zu erstellen.

Für die Herstellung von Anwendungssystemen spielt neben der Beherrschung der Komplexität die Einbeziehung der Auftraggeber (Fachabteilung) eine wesentliche Rolle. Viele der gängigen Vorgehensmodelle sehen für die frühen Phasen Tätigkeiten vor, die die Fachabteilungen in den Entwicklungsprozeß mit einbeziehen sollen. Die verwendeten Methoden und Techniken sind entsprechend auf die Belange der Fachabteilungen auszurichten.

Die Entwicklungstätigkeiten der frühen Phasen sind schwerpunktmäßig (in Klammern die Nummern der diesbezüglichen Abschnitte):

- Projektziele und Untersuchungsbereich festlegen (4.1)
- Anforderungen an das Anwendungssystem erarbeiten (4.2)
- Anwendungsmodell (Daten- und Funktionsmodell) erstellen (4.3)
- Organisatorisches und technisches System abgrenzen (4.4)
- Benutzungsschnittstelle entwerfen und beschreiben (4.5)

Das Ergebnis der frühen Phasen ist das Fachkonzept, das das zukünftige Anwendungssystem aus fachlicher Sicht darstellt. Es dient als Grundlage für DV-Konzeption und -Realisierung.

Fachkonzept (Synonym: Aufgabendefinition, fachlicher/funktionaler/sachlogischer Entwurf, Fachspezifikation, funktionale Spezifikation): Zusammenfassende Darstellung des Anwendungssystems aus fachlicher Sicht.

Seine wesentlichen Teile sind: Abgrenzung und Beschreibung des Gegenstandsbereichs, Anforderungen an das Anwendungssystem, Anwendungsmodell, Beschreibung der Struktur.

Neben den Entwicklungstätigkeiten sind die Management-Tätigkeiten des Planens und Steuerns (s. Abschn. 4.6) und die Tätigkeiten der Qualitätssicherung (s. Abschn. 4.7) zu betrachten.

Für große Unternehmen, die in mehreren Geschäftsbereichen und Sparten nebeneinander verschiedene Anwendungssysteme entwickeln und betreiben, stellt sich die Frage der unternehmensweiten Modellierung (s. Abschn. 4.8).

4.1. Projektziele festlegen

Wird ein Projekt zur Entwicklung eines Anwendungssystems in einem formalen Rahmen neu aufgelegt, spricht man von der *Projekt-Initialisierung*, die den Charakter einer administrativen Vorphase hat (vgl. Abschn. 4.6). Inhaltlich bestehen die ersten Tätigkeiten darin, die *Projektziele*, den *Untersuchungsbereich* und den *Gegenstandsbereich* des Systems festzulegen.

Untersuchungsbereich (Synonym: *Problembereich*, *Miniwelt* oder *Universe of Discourse* – kurz: UoD): Teilbereich der realen Welt, der mit dem Ziel untersucht wird, ein Anwendungssystem zu erstellen.

Zum Untersuchungsbereich gehören u. a. die beteiligten Menschen, Organisationseinheiten, Technikkomponenten und ihre Funktionen, Tätigkeiten, Einrichtungen und Beziehungen untereinander.

Projektziele festlegen: Ausgehend von der *Identifizierung des Untersuchungsbereichs* werden die Wirkungen und Leistungen beschrieben, die durch das zu entwickelnde System zu erbringen sind.

Ziele lassen sich unter verschiedenen Gesichtspunkten gliedern, z. B. in Systemziele/Vorgehensziele, Sach-/Formalziele, ökonomische/technische/soziale Ziele oder Muß-/Soll-/Kann-Ziele.

Zielkatalog: Ergebnis der Tätigkeit *Projektziele festlegen*. Die Zielbeschreibung ist i. a. zunächst noch recht grob und wird bei den Sachzielen später zu den *Anforderungen an das Anwendungssystem* detailliert (vgl. Abschn. 4.2).

Gegenstandsbereich: Derjenige Teil des Untersuchungsbereichs, der Gegenstand der Modellierung und Realisierung des Anwendungssystems ist.

Eine wichtige Analyseaufgabe besteht darin, innerhalb des Untersuchungsbereichs den tatsächlichen *Gegenstandsbereich* des Anwendungssystems *abzugrenzen*. Dabei entstehen Grenzen zwischen dem Gegenstandsbereich des künftigen Anwendungssystems und seiner Umgebung, z. B. zu angrenzenden Systemen, die sowohl *technischer* (zu anderen technischen Systemen) als auch *organisatorischer Art* (zu Benutzern und sonstigen Betroffenen) sein können. Die *Analyse des Gegenstandsbereichs* läßt sich in die Teiltätigkeiten *Istzustand beschreiben*, *Stark-/ Schwachstellen analysieren* und *Lösungsmöglichkeiten erarbeiten* untergliedern. Das Abgrenzen und Analysieren des Gegenstandsbereichs ist in der Praxis meist nicht voneinander zu trennen, sondern geschieht in der Regel zusammen.

4.2. Anforderungen an das Anwendungssystem erarbeiten

Wenn das Projekt initialisiert ist und die Projektziele festgelegt sind, werden die Anforderungen an das Anwendungssystem gesammelt und beschrieben. Dies ist der Gegenstand des „Requirements Engineering“, das Methoden, Beschreibungsmittel und Werkzeuge zur Ermittlung, Formulierung und Analyse von Aufgabenstellungen und sonstigen (z. B. qualitativen) Anforderungen an Systeme umfaßt (vgl. [20]).

Anforderungen: Aussagen über zu erbringende Leistungen eines Gegenstands sowie seine qualitativen oder quantitativen Eigenschaften (vgl. [12]).

Zu den Anforderungen an ein Anwendungssystem können gehören (vgl. [20]):

- *funktionale* Anforderungen, gegliedert nach einzelnen Funktionen oder Funktionskomplexen, ihren Ein- und Ausgaben;
- *Qualitäts-*Anforderungen an das System bzw. einzelne Funktionen, z. B. Antwortzeiten, Speicherverbrauch, Zuverlässigkeit, Änderbarkeit, Wartbarkeit, Portabilität;
- Anforderungen an die *Benutzbarkeit* des Systems: ergonomische Gestaltung der Benutzungsschnittstelle, Aufgaben-Angemessenheit, Selbstbeschreibungsfähigkeit, Erlernbarkeit, Steuerbarkeit, Erwartungskonformität, Fehler-Robustheit, Individualisierbarkeit;
- Anforderungen an die *Realisierung* des Systems: Dazu gehören die sogenannte Ziel-Plattform (bestehend aus vorgegebenen HW/SW-Komponenten, Geräten, sonstigen Bausteinen, Schnittstellen), die Benutzungs-Dokumentation des Systems und Handbücher;
- Anforderungen an die *System-Erstellung*, *-Einführung* und *-Betreuung*: Dazu gehören die Entwicklungsumgebung (bestehend aus zu verwendenden Methoden, Verfahren, Beschreibungsmitteln und Werkzeugen auf einer „Entwicklungs-Plattform“), sonstige Entwicklungs-Ressourcen, zu beachtende Vorschriften, Prüfverfahren sowie die Entwicklungs-Dokumentation.

Anforderungen definieren: Tätigkeit mit dem Ziel, Anforderungen an ein System zu beschreiben, zu präzisieren und als Vorgabe für eine künftige Modellierung und Realisierung festzuschreiben.

Anforderungskatalog (Synonym: Anforderungsdefinition, Pflichtenheft, Lastenheft (vgl. [32])): Ergebnis der Tätigkeit *Anforderungen definieren*.

4.3. Anwendungsmodell erstellen

Im Mittelpunkt der fachlichen Tätigkeiten der frühen Phasen steht die Erstellung des Anwendungsmodells, das sich in der Regel aus einem *Datenmodell*, einem *Funktionsmodell* und einem *Ablaufmodell* zusammensetzt.

Modell: Idealisierte, vereinfachte, in gewisser Hinsicht ähnliche Darstellung eines Gegenstands, Systems oder sonstigen Weltausschnitts mit dem Ziel, daran bestimmte Eigenschaften des Vorbilds besser studieren zu können.

Anwendungsmodell: Modell, das sich auf einen Weltausschnitt bezieht, der Gegenstandsbereich einer Software-Anwendung ist.

Bezüglich der in Teil 1, Kap.3 gegebenen Klassifizierung lassen sich Anwendungsmodelle folgendermaßen einordnen: Sie richten sich an Anwender, Angehörige von Fachbereichen/-abteilungen (deren Arbeitsbereich sie beschreiben) sowie an Systemarchitekten und -entwickler (als Grundlage für das von ihnen zu erstellende Anwendungssystem). Sie erstrecken sich auf alle genannten Sprachkategorien. Sie geben Strukturen für die Exemplar-Ebene vor und setzen sich damit aus Elementen der Typ-Ebene zusammen. Sie entstehen in der Analyse- und Definitionsphase, sind aber für den gesamten weiteren Systementwicklungsprozeß maßgeblich.

Anwendungsmodelle werden häufig nach dem zweiten Kriterium aus Teil 1, Kap.3 (Sprachkategorien) untergliedert. So kommt man zu einem

- Datenmodell,
- Funktionsmodell und
- Ablaufmodell.

Datenmodell: Modell, das die statische Struktur des Gegenstandsbereichs beschreibt.

Im Mittelpunkt stehen Beschreibungen der Gegenstände, ihrer Merkmale und Beziehungen in Form von Entitätstypen, Attributen etc. (vgl. Abschn.5.3). Der von uns bevorzugte Datenbegriff (vgl. Teil 1, Abschn.2.3) paßt für diesen Teil des Anwendungsmodells kaum, ein Begriff wie „Entitätsmodell“ wäre angebrachter. Aus pragmatischen Gründen wollen wir jedoch dem üblichen Sprachgebrauch folgen. Faßt man mehrere Datenmodelle, die durch gemeinsame Entitätstypen oder Attribute zusammenhängen, zu größeren Modellen zusammen, kommt man zu Bereichs- und schließlich Unternehmens-Datenmodellen (vgl. Abschn.4.8). Für die Darstellung von Datenmodellen werden sehr häufig Entity/Relationship-Diagramme eingesetzt.

Funktionsmodell: Modell, das die aktiven Elemente des Gegenstandsbereichs beschreibt: die Funktionen, ihre Ein- und Ausgaben, Verarbeitungsvorschriften, die dabei bearbeiteten Masken und Listen.

Diese werden in Form von *Funktionsbeschreibungen* zusammengefaßt. Die Funktionen werden in der Regel hierarchisch untergliedert. Für die Funktionen höherer und niedriger Hierarchie-Ebenen sind Begriffe wie *Funktionskomplex*, *Hauptfunktion* bzw. *Elementarfunktion* gebräuchlich (vgl.

Abschn.5.2). Für die Darstellung der einzelnen Funktionen werden Gliederungsschemata (z.B. Eingabe/Verarbeitung/Ausgabe), Pseudo-Codes, Entscheidungstabellen oder (strukturierte) natürliche Sprache verwendet.

Ablaufmodell: Teil des Anwendungsmodells, der diejenigen Elemente des Gegenstandsbereichs enthält, die den Zusammenhang zwischen aktiven und passiven Elementen und zwischen verschiedenen aktiven Elementen herstellen.

Dazu gehören die Beschreibungen des Ablaufs von Geschäftsvorgängen und -prozessen, der Ablaufbeziehungen verschiedener Funktionen untereinander, ihrer Sichten auf das Datenmodell und der von ihnen bewirkten Datenflüsse, Dialogbeschreibungen, sowie Anweisungen für Ausnahme- und Fehlerfälle. Bevorzugte Darstellungsarten sind Datenflußdiagramme und Petri-Netze.

Neuere, auf eine *objektorientierte Anwendungsmodellierung* zielende Überlegungen gehen dahin, Anwendungsmodelle nicht nach unterschiedlichen Sprachkategorien (d.h. in ein Daten- und Funktionsmodell) zu untergliedern, sondern vorrangig nach fachlich-inhaltlichen, aus der Anwendung begründeten Gesichtspunkten (vgl. [15]). Das bedeutet, daß das Daten- und Funktionsmodell (und soweit möglich auch das Ablaufmodell) für alle Teile des Gegenstandsbereichs (d.h. für jeden Objekttyp) gemeinsam erstellt und beschrieben wird. Die strikte Trennung in ein Daten-, Funktions- und Ablaufmodell wird damit aufgehoben.

4.4. Organisatorisches und technisches System abgrenzen

Die im Funktionsmodell identifizierten Funktionen werden auf der Grundlage der fachlichen Anforderungen hinsichtlich ihrer späteren Betriebsart analysiert. Es wird dabei zunächst eine Zuordnung der Funktionen zum *organisatorischen System* und zum *technischen System* (s. Teil 1, Abschn.2.4) vorgenommen.

Folgende Tätigkeiten führen zur Abgrenzung der Funktionen:

Automatisierung einer Funktion festlegen: Analyse der Funktion dahingehend, ob und inwieweit innerhalb der vorgegebenen technischen, organisatorischen und sozialen Rahmenbedingungen eine DV-technische Realisierung möglich, sinnvoll und gerechtfertigt ist.

Betriebsart: Form, in der eine DV-technische Funktion abläuft. Üblich ist eine Unterscheidung von *interaktivem Betrieb* und *Stapelbetrieb*. Weiter kann nach dem Typ des DV-Systems (Client, Server, Host, Workstation, Abteilungsrechner, PC-Einzelplatz, PC-Netzwerk etc.) unterschieden werden.

Betriebsart zuordnen: Analyse, in welcher Betriebsart die Funktion realisiert werden soll, z. B. sollte bei einem Flugbuchungssystem der Auskunfts- und Reservierungsteil interaktiv abgewickelt werden können; der Berichtsteil kann, sofern er nicht zeitkritisch ist, im Stapelbetrieb ablaufen.

Häufig wird eine Funktion des Funktionsmodells nicht vollständig technisch bzw. organisatorisch realisiert oder läßt sich nicht eindeutig einer bestimmten Betriebsart zuordnen,

sondern muß in Teilfunktionen zerlegt werden, die dann unterschiedlich zugeordnet werden. So wird die Funktion „Flugbuchung“ zerlegt in die Teilfunktionen „Kommunikation mit dem Kunden“, die Teil des organisatorischen Systems wird, und „Flug buchen“, die Teil des DV-Systems wird und interaktiv abläuft. Alternativ ließe sich die Funktion Flugbuchung auch komplett dem technischen System zuordnen, wenn der Kunde die Buchung selbst vornehmen soll, ohne Einschaltung einer Person der Fluggesellschaft.

Die Abgrenzung zwischen dem organisatorischen und dem technischen System erfordert häufig weitere Funktionen. Die Funktionen des organisatorischen Systems bilden die Grundlage für die im Rahmen der weiteren Realisierung anzufertigenden *Arbeitsablaufbeschreibungen*.

4.5. Benutzungsschnittstelle entwerfen und beschreiben

Sind organisatorisches und technisches System voneinander abgegrenzt, so muß für die technisch realisierten Funktionen bzw. Funktionsteile die Benutzungsschnittstelle entworfen und beschrieben werden.

Interaktionsentwurf: Teil der Gestaltung von Arbeitsabläufen, der sich mit dem Wechselspiel von Mensch und Maschine beim interaktiven Betrieb befaßt.

Das *Interaktionsdiagramm* dient dabei als graphisches Beschreibungsmittel des Entwurfs.

Benutzungsschnittstelle: Menge von Vereinbarungen, die die Interaktion von Mensch und Maschine beim Ablauf einer technisch realisierten Funktion beschreiben.

Dialog: Direkter Austausch von Nachrichten zwischen zwei menschlichen Dialogpartnern.

Diese Grundbedeutung wird im Fach-Sprachgebrauch oft auf nicht-menschliche Dialogpartner ausgedehnt. Im Falle eines menschlichen und eines maschinellen Partners wollen wir jedoch lieber von *Mensch-Maschine-Interaktion* sprechen.

Die interaktive Verarbeitung hat die Stapelverarbeitung vor allem dort weitgehend verdrängt, wo ein hoher Aktualitätsgrad der Daten, eine ereignisbezogene Aufgabenerfüllung und die Beschleunigung von Abläufen verlangt wird. Für die Gestaltung interaktiver Systeme stehen unterschiedliche Techniken zur Auswahl.

Maskentechnik: Gestaltung der Benutzungsschnittstelle in Form von *Masken*, worunter die Aufteilung des Bildschirms in Bereiche verstanden wird, deren Inhalt teilweise vom System vorgegeben, teilweise vom Benutzer auszufüllen ist. Zweckmäßig ist dabei eine Untergliederung in Kennzeichnungsbereich, Arbeitsbereich, Steuerungsbereich und Meldungsbereich.

Fenstertechnik: Mensch-Maschine-Interaktion mit Hilfe von *Fenstern*, d. h. Bildschirmbereichen mit Randbegrenzungen, von denen mehrere geöffnet sein können, so daß quasi simultan mehrere Interaktionen zu einer Zeit möglich sind.

Graphische Benutzungsschnittstelle: Anwendung der graphischen Verarbeitungstechnik, die sich für die graphische Darstellung von Daten mit Hilfe grafikfähiger Bildschirme die Adressierbarkeit von Bildpunkten zunutze macht.

Außer am Bildschirm gibt es Benutzungsschnittstellen am Drucker in Form von *Listen* und *Formularen*. Aus Sicht der Benutzer sind sie durch ein Layout strukturierte Informationsträger in Papierform.

In dem Maße, in dem die Verarbeitungstechniken für statische und bewegte Bilder sowie vor allem für Sprache ausgereifter werden, eröffnen sich neue Möglichkeiten für die Gestaltung von Benutzungsschnittstellen und damit für neue Formen der Mensch-Maschine-Interaktion.

4.6. Management-bezogene Tätigkeiten

Management-bezogene Tätigkeiten begleiten die Anwendungssystem-Entwicklung über alle Phasen. Wir wollen hier auf einige Begriffe beschränken, die typischerweise den frühen Phasen, also der Analyse und Modellierung, zuzuordnen sind.

Die Entwicklung eines Anwendungssystems beginnt üblicherweise mit der formalen Initialisierung eines Projektes zwischen einem Auftraggeber und einem Auftragnehmer in Form eines Projektauftrags.

Projekt initialisieren: Gesamtheit der Maßnahmen, um ein Projekt formell ins Leben zu rufen.

Dazu gehören: Projektleiter benennen, Budget bereitstellen, einen Projektauftrag (zwischen Auftraggeber und Auftragnehmer) vereinbaren und Kommunikation regeln.

Auftraggeber: Eine (juristische) Person, die ein Anwendungssystem in Auftrag gibt und bezahlt.

Auftragnehmer: Eine (juristische) Person, die die Verantwortung für die Herstellung eines Anwendungssystems übernimmt.

Projektauftrag: Vereinbarung zwischen Auftraggeber und Auftragnehmer über ein durchzuführendes Projekt, die eine kurze Beschreibung des zu erstellenden Anwendungssystems, der geschätzten Entwicklungskosten und der Verantwortlichkeit enthält.

Die tatsächliche Durchführung des Projekts ist meist an die Ergebnisse bestimmter, in der Analyse-Phase durchzuführender Studien geknüpft, nämlich der Durchführbarkeitsstudie und der Kosten-Nutzen-Analyse. In diese Studien geht eine Aufwandsschätzung auf Basis eines Mengengerüsts ein.

Mengengerüst: Aufstellung über Istmengen, Sollmengen, Mengenentwicklung pro Zeiteinheit, Bestandsdaten und Bewegungsdaten, die quantitative Aussagen über das Projekt ermöglicht.

Aufwandsschätzung: Herstellung eines funktionalen Zusammenhangs zwischen dem Mengengerüst und den im Zusammenhang mit einem Projekt zu erwartenden Aufwänden und Kosten auf der Basis von Erfahrungswerten.

Durchführbarkeitsstudie: Studie zur Erarbeitung einer Empfehlung, ob ein Anwendungssystem unter technischen und Aufwandsaspekten realisiert werden soll.

Ziel der Durchführbarkeitsstudie (feasibility study) ist eine Empfehlung, die dem betroffenen Fachabteilungs- und DV-Management (bzw. dem DV-Steuerungsgremium)

eine Entscheidung darüber ermöglicht, ob das System in der vorgeschlagenen Form realisiert werden soll. In der Studie müssen Aussagen zu folgenden Aspekten erarbeitet werden:

- Ziele
- Veränderungen im Systemumfeld
- Restriktionen durch das Systemumfeld
- Entwicklungsaufwand
- Aufwand für den Betrieb
- zu erwartender Nutzen
- Vorschlag für einen Projektplan

Kosten-Nutzen-Analyse: Untersuchung zur Erarbeitung einer Empfehlung, ob der zu erwartende Nutzen die Realisierung eines Anwendungssystems bei den zu erwartenden Kosten rechtfertigt.

Ausführliche Darstellungen zu Management-bezogenen Tätigkeiten sind zu finden in [9, 12], speziell über Projektmanagement in [3, 13].

4.7. Tätigkeiten der Qualitätssicherung

Qualitätssicherung: Gesamtheit von angemessenen, aufeinander abgestimmten Maßnahmen zur Erfüllung vorgegebener Anforderungen an die Qualität eines Produkts oder Herstellungsprozesses (vgl. [12]).

Qualitätssicherung für Anwendungssysteme unterscheidet sich von der Software-Qualitätssicherung dadurch, daß nicht nur die Software, sondern auch Teile des organisatorischen Systems, z. B. Geschäftsvorgänge, Arbeitsabläufe, Organisationsanweisungen, Gegenstand sind.

Qualitätssicherung dient dazu, bestimmte *Qualitätseigenschaften* zu erreichen und einzuhalten, z. B. Korrektheit, Zuverlässigkeit, Benutzungs- und Wartungsfreundlichkeit, aber auch Aufgabenangemessenheit, Transparenz und Ganzheitlichkeit.

Die Bestimmung von *Qualitätsmerkmalen*, d. h. relevanten Qualitätseigenschaften, ist abhängig vom Gegenstand der Qualitätssicherung sowie von zahlreichen internen und externen Einflüssen in einem Unternehmen, z. B. individuellen Unternehmenszielen, spezifischen organisatorischen Abläufen oder der Größe des Unternehmens. Ein allgemeines, genormtes Qualitätssicherungssystem kann es daher bei diesem Ansatz nicht geben; in DIN- und ISO-Normen werden aber für die wichtigsten Elemente eines Qualitätssicherungssystems Empfehlungen gegeben [18].

Bei der Qualitätssicherung selbst ist zu unterscheiden zwischen *Produktqualität* und *Prozeßqualität*. Bisher liegen die Schwerpunkte der Software-Entwicklung vor allem bei der konstruktiven Verbesserung von Software-Produkten, der Erstellung von Modellen zur Festlegung von Softwarequalität und bei Messungen am Zwischen- und Endprodukt.

Zur Zeit zeichnet sich eine stärkere Konzentration auf den SW-Herstellungsprozeß ab. Dahinter steht die Hypothese, daß die Qualität eines Produkts maßgeblich von der Qualität des Herstellungsprozesses beeinflusst wird. Die Fehlervermeidung wird nicht zuletzt aus betriebswirtschaftlichen Erwägungen einen größeren Stellenwert als die Fehlerentdeckung haben.

Produktorientierte Qualitätssicherung beinhaltet die Prüfung von Softwareprodukten und Zwischenergebnissen im

Hinblick auf vorher festgelegte Qualitätsmerkmale. Dazu gehören Normenkonformität bei Sprachen, Gütebedingungen und Prüfbestimmungen bei Anwendungs-Software. Diese können Gegenstand einer Zertifizierung, d. h. einer Bestätigung durch anerkannte (akkreditierte) Stellen sein, daß bestimmte Normen eingehalten wurden.

Prozeßorientierte Qualitätssicherung bezieht sich auf den Herstellungsprozeß der Software. Dazu gehören Methoden, Werkzeuge, Richtlinien und Standards. Der Grundgedanke besteht darin, ein von allen Mitarbeitern getragenes Qualitätsbewußtsein zu entwickeln, das langfristig den Produktionsprozeß im Sinne einer optimalen Qualität verändert.

4.8. Unternehmensweite Modellierung

Wurden Anwendungssysteme bisher voneinander isoliert entwickelt, nehmen heute die Bemühungen zu, sie in sinnvolle und überschneidungsfreie Zusammenhänge zu bringen im Sinne einer integrierten Systemlandschaft.

Unternehmensmodell: Integrierte Zusammenfassung von Anwendungsmodellen im Unternehmen, die für die Anwendungsentwicklung den Orientierungsrahmen bilden. Sie veranschaulichen unterschiedliche Aspekte – z. B. den organisatorischen Aspekt oder den Informationsfluß – des gesamten Unternehmens oder einzelner Unternehmensbereiche.

Unternehmensweite Modellierung: Gesamtheit der Maßnahmen, die zu einem Unternehmensmodell führen.

Unternehmensdatenmodell: Der Teil des Unternehmensmodells, der den strukturellen Zusammenhang des gesamten Informationsbestands eines Unternehmens veranschaulicht.

Die unternehmensweite Modellierung wird von vielen Autoren beschrieben [2, 25, 28, 34]. Sie folgen dabei in der Regel einem Sichtenansatz, der z. B. die organisatorischen Aspekte des Unternehmens von seinen informationsstrukturellen und funktionalen Aspekten unterscheidet.

Der *organisatorische Aspekt* befaßt sich mit den verantwortlichen Aufgabenträgern, die Anwendungssysteme zur Unterstützung ihrer Aufgaben in Auftrag geben oder nutzen und daher als Anwender in Erscheinung treten. Die häufigste Darstellungsform der Organisationsstruktur ist das *Organigramm*, das die Aufgabenteilung und -koordination einer Organisation beschreibt. Will man die Informationsbeziehung zwischen Aufgabenträgern sichtbar machen, wird ein Kommunikationsmodell erstellt.

Die *informationsstrukturellen Aspekte* (Datenaspekte) werden durch das Unternehmensdatenmodell repräsentiert. Es entsteht in der Regel durch die schrittweise Integration von anwendungsspezifischen Datenmodellen.

Unter die *funktionalen Aspekte* fällt die Gliederung der Gesamtaufgabe des Unternehmens mittels Aufgabenanalyse und Aufgabensynthese, die meist hierarchisiert in Form eines Funktionsbaums dargestellt wird.

Der Detaillierungsgrad dieser Aspekte orientiert sich an der Forderung, daß das Unternehmensmodell sichtbar machen soll, was für die Verfolgung der Unternehmensziele und für die Berücksichtigung kritischer Erfolgsfaktoren vorhanden bzw. einzurichten ist.

Werden das Unternehmensmodell und die bestehenden Anwendungssysteme untereinander in Beziehung gesetzt, so läßt sich auf Unternehmensebene eine Abweichungsanalyse durchführen, die eine Beurteilung der Durchdringung der Organisation mit Anwendungssystemen sowie eine Bewertung der Unterstützungsqualität dieser Systeme erlaubt. Die auf diese Weise festgestellten Anwendungsdefizite sind ein wichtiges Kriterium für die Planung des IKS und die Priorisierung von Entwicklungsprojekten. Information aus abgeschlossenen Entwicklungsprojekten für Anwendungssysteme fließen in das Unternehmensmodell ein und schreiben es fort.

5. Die Elemente von Anwendungsmodellen

Das Anwendungsmodell dient der Darstellung des zukünftigen Anwendungssystems aus fachlicher Sicht. Aspekte der DV-technischen Realisierung sind nicht Gegenstand des Anwendungsmodells. Die hier verwendeten Begriffe orientieren sich an einer standardisierten Fachsprache, für die in Teil 1, Abschn. 2.3 die Basis gelegt wurde.

Die Elemente des fachlichen Modells können datenorientiert (passiv), funktionsorientiert (aktiv) oder zusammenhangsbildend sein (s. dazu auch Teil 1, Abschn. 3.2). Die im folgenden definierten Begriffe gehören zur Meta-Ebene, ihre Ausprägungen zur Typ- oder Exemplar-Ebene.

5.1. Passive (datenorientierte) Elemente

Eine der wichtigsten Aufgaben der fachlichen Modellierung ist die Identifizierung, Abgrenzung und Beschreibung der Elemente des Gegenstandsberichts, über die Daten erhoben, gespeichert, verarbeitet und wiedergewonnen werden sollen. Solche *Gegenstände* werden als *Entitäten* modelliert und durch *Attribute* charakterisiert. Eine Übersicht über die datenorientierten (passiven) Begriffe gibt Abb. 7.

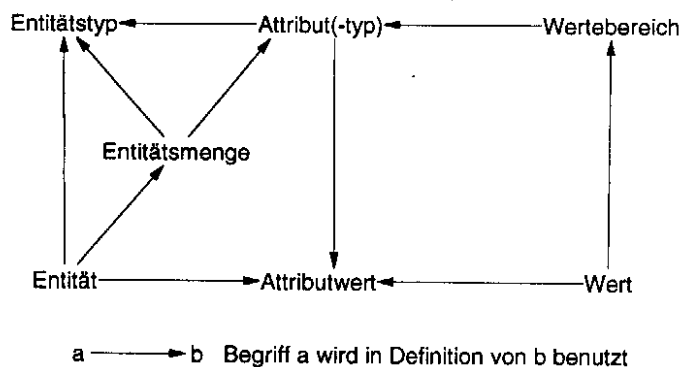


Abb. 7. Übersicht datenorientierte (passive) Begriffe

Entität: Repräsentation eines (konkreten oder abstrakten) Gegenstands, der für ein gegebenes Anwendungssystem von Bedeutung ist.

Synonym: entity, Objekt (im engeren Sinn), Exemplar, instance, Ausprägung

Im relationalen Modell: Tupel

Beispiel: Eine irgendwie geartete Repräsentation des Kunden Alfons Maier, geboren am 31. 12. 45, wohnhaft in 81925 München, Isarstraße 25.

Bemerkung: Wir ziehen den Begriff Entität dem im deutschsprachigen Raum oft benutzten Begriff Objekt vor, um damit Verwechslungen mit der Terminologie objektorientierter Systeme zu vermeiden.

Entitätsmenge: Menge von Entitäten, die bezüglich bestimmter Eigenschaften oder Beziehungen als gleichartig behandelt werden.

Synonym: entity set

Im relationalen Modell: Relation

Beispiel: Die Menge aller Kunden, die den Wohnort Hamburg haben.

Wertebereich: Menge von möglichen Werten, die zur Beschreibung einer Eigenschaft herangezogen werden können. Beispiel: Postleitzahlen nehmen Werte zwischen 00001 und 99999 an.

Attribut(-typ): Zuordnungsvorschrift, die jeder Entität einer Entitätsmenge zu jedem Zeitpunkt jeweils höchstens ein Element aus einem Wertebereich zuordnet.

Synonym: Attributtyp, Eigenschaftstyp, Eigenschaft

Im relationalen Modell: Tabellenspalte

Bemerkung: Attribute dienen zur Beschreibung von Eigenschaften von Gegenständen mit Hilfe von Werten aus vorgegebenen Wertebereichen. Formal kann man ein Attribut definieren als Funktion (mit Entitätsmenge EM , Zeit T und Wertebereich W): $f: EM \times T \rightarrow W$.

Beispiel: Durch das Attribut *Geburtsdag* ist jedem Kunden ein Element aus dem Wertebereich *Datum* zugeordnet.

Attributwert: Ein durch ein Attribut einer Entität zugeordneter Wert, der eine Eigenschaft des modellierten Gegenstands beschreibt.

Synonym: Eigenschaftswert

Bemerkung: Werte können durch Daten repräsentiert werden.

Beispiel: 31. 12. 45 ist dem Kunden Alfons Maier als Wert für das Attribut *Geburtsdag* zugeordnet.

Entitätstyp: Zusammenfassung von Attributen, die auf einer Entitätsmenge definiert sind. Ein Entitätstyp nimmt Bezug auf Gegenstände, die als gleichartig betrachtet werden, und beschreibt deren Eigenschaften.

Synonym: Entitätsklasse, Objekttyp, Objektklasse (im engeren Sinn)

Bemerkung: Formal läßt sich ein Entitätstyp (ET) definieren als:

$ET = \{f_i | f_i: EM \times T \rightarrow W_i\}$, wobei mit EM die Entitätsmenge, mit T die Zeit und mit W_i die Wertebereiche ($i = 1, \dots, n$ mit $n \in \mathbf{N}$) bezeichnet werden.

Die Definition von Entitätstypen dient der einordnenden Bezugnahme auf Entitäten.

Beispiel: Der Entitätstyp *Kunde* ist durch die Menge der Attribute Kunden-Nr., Name, Vorname, PLZ, Ort, Straße definiert.

Subentitätsmenge: Eine Teilmenge einer gegebenen Entitätsmenge.

Beispiel: Zur Entitätsmenge *Mitarbeiter* bilden die *motorisierten Mitarbeiter* eine Subentitätsmenge.

Subentitätstyp: Ein Entitätstyp B , der mit einem Entitätstyp A in folgendem Zusammenhang steht:

- Die Entitätsmenge, auf der B definiert ist, ist eine Teilmenge der Entitätsmenge, auf der A definiert ist. Diese Teilmenge ist bestimmt durch
 - eingeschränkte Wertebereiche bestimmter Attribute und/oder
 - das Bestehen bestimmter Beziehungen zu anderen Entitäten.
- Außer den Attributen und Beziehungen von A kann B weitere Attribute und Beziehungen enthalten.

Bemerkung: Praktisch bedeutet dies, daß man einen Subentitätstyp bildet, indem man zu einem gegebenen Entitätstyp die Entitätsmenge auf bestimmte Attributwerte und/oder bestimmte Beziehungen einschränkt und dabei eventuell zusätzliche Attribute definiert.

Beispiel: Eine Firma hat den Entitätstyp *Mitarbeiter* definiert: *Mitarbeiter* = (Personalnummer, Name, Adresse, Fortbewegungsmittel, ...). Das Attribut Fortbewegungsmittel kann die Werte (Auto, Fahrrad, Motorrad, Fußgänger) annehmen. Nun läßt sich der Subentitätstyp *motorisierte Mitarbeiter* = (Personalnummer, Name, Adresse, Fortbewegungsmittel, ..., Kennzeichen, Hersteller) bilden, dessen Entitätsmenge alle Mitarbeiter mit Auto oder Motorrad enthält und der die zusätzlichen Attribute Kennzeichen und Hersteller beinhaltet.

Bemerkung: Mit diesen Definitionen wird festgelegt, daß „Entitäten“, „Attribute“, „Entitätsmengen“ und „Entitätstypen“ Bestandteile von Modellen sind. *Entitäten* stehen für *Gegenstände der realen Welt* und werden daher im alltäglichen Sprachgebrauch häufig mit diesen identifiziert, sind tatsächlich aber nicht mit ihnen identisch.

Formal kann man Entitäten als Elemente einer Menge auffassen. Durch Attribute werden diesen zu bestimmten Zeitpunkten Werte zugeordnet. (Diese Zuordnung wäre problematisch, wenn man – wie in [4] geschehen – Entitäten selbst als „Dinge der realen Welt“ auffassen würde.) Faßt man alle zu einem gegebenen Element E und einem Zeitpunkt t gültigen Werte zu einem Werte-Tupel zusammen, erhält man eine Entitäts-Zustandsbeschreibung von E zum Zeitpunkt t . Dies entspricht einer Tabellenzeile in einer relationalen Datenbank.

Zu *Entitätsmengen* gelangt man durch Zusammenfassung von Entitäten. Bildet man diese aus gleichartigen, d. h. durch die gleichen Attribute beschriebenen Entitäten, so kann man die Werte-Tupel zu Tabellen zusammenfassen und gelangt zu den Tabellen (oder „Relationen“) einer relationalen Datenbank.

Die Angabe aller für eine Entitätsmenge maßgeblichen Attribute und ihrer Wertebereiche kann zur (intensionalen) Beschreibung einer Entitätsmenge verwendet werden und wird als *Entitätstyp* bezeichnet. *Subentitätsmengen* und *Subentitätstypen* werden in analoger Weise als Teilstrukturen definiert.

Beispiel: Zur Erläuterung dieser Begriffe betrachten wir den Entitätstyp „Konto“ einer Bank. Er ist definiert auf der Entitätsmenge, die der Gesamtheit der Konten der Bank entspricht. Jedes Konto der Bank ist dabei mit genau einer Repräsentation in der Entitätsmenge vertreten und durch ein identifizierendes Attribut (z. B. die Kundennummer oder ein Codewort) eindeutig ansprechbar.

Der Entitätstyp „Konto“ bestehe aus den Attributen „Kontonummer“, „Kontoart“, „Kontoinhaber“, „Eröffnungsdatum“ und „Kontostand“. Das Attribut „Kontostand“ ordnet jeder Entität zu jedem Zeitpunkt einen DM-Betrag zu, der den aktuellen Kontostand angibt, z. B. der Entität „422664“ am 01.09.92, 10 Uhr, den Betrag 1845,23. Dies ist ein typisches Beispiel für ein zeitabhängiges Attribut, d. h. zu einem anderen Zeitpunkt wird der gleichen Entität möglicherweise ein anderer Betrag zugeordnet. Das Attribut „Kontoart“ wird hingegen nicht zeitabhängig verändert.

Die Menge der Konten einer Bank ist in der Realität ebenfalls zeitabhängig, d. h. Entitätsmengen verändern sich im Laufe der Zeit durch Hinzunahme, Veränderung und Fortfall einzelner Elemente.

5.2. Aktive (funktionsorientierte) Elemente

Funktion: Abbildungsvorschrift, die einer Menge von (Eingabe-) Daten eine Menge von (Ausgabe-) Daten zuordnet.

In betrieblichen Anwendungssystemen stehen Funktionen oft für manuelle, zu automatisierende bzw. automatisierte Tätigkeiten oder Vorgänge. Diese werden auch als **Anwendungsfunktionen** bezeichnet. Bei den Ein-/Ausgabedaten handelt es sich in der Regel um Tabellen oder Listen von Werten, einzelne Attributwerte, Steuerdaten (Auslöser) oder Statusanzeigen. Die durch Funktionen gegebene Abbildung kann automatisiert oder manuell erfolgen. Funktionen können hierarchisch gegliedert werden, d. h. der Funktionsbegriff kann rekursiv verwendet werden. Soll die Stellung einer Funktion in der Funktionshierarchie durch die Begriffsbildung widergespiegelt werden, so kann das durch die Wahl geeigneter Präfixe (**Hauptfunktion**, **Elementarfunktion**) oder Suffixe (-gruppe, -komplex) ausgedrückt werden.

Synonym: Business Process

Bemerkung: Im Gegensatz zu den später möglicherweise hinzukommenden *DV-Funktionen* beschäftigt sich die Anwendungsmodellierung ausschließlich mit *Anwendungsfunktionen*.

5.3. Zusammenhangsbildende Elemente

Zusammenhänge in Anwendungsmodellen werden durch Beziehungen gebildet. Wir unterscheiden drei Arten von Beziehungen:

- Beziehungen von Entitäten untereinander: Entitätsbeziehungen
- Beziehungen von Funktionen untereinander: Funktionsbeziehungen
- Beziehungen zwischen Funktionen und Entitäten: sog. Sichten

(Entitäts-) Beziehung: Verbindung mehrerer – in der Regel genau zweier – Entitäten miteinander.

Beispiel: Kunde Meier *besitzt* das Konto mit der Kontonr. 4711.

(Entitäts-) Beziehungstyp: Verknüpfung mehrerer – in der Regel genau zweier – Entitätstypen miteinander.

Ob eine solche Verknüpfung hergestellt wird oder nicht, resultiert aus den Gegebenheiten eines Geschäfts- oder Auf-

gabenbereiches eines Unternehmens. Ein (Entitäts-) Beziehungstyp wird beschrieben durch Bezeichner, Stelligkeit (Anzahl der beteiligten Entitätstypen), Kardinalität (Maximalzahl der beteiligten Entitäten pro Entitätstyp) und Konditionalität oder Optionalität (Aussage darüber, ob zu einem Entitätstyp mindestens eine Entität vorhanden sein muß). Zweistellige Beziehungen können durch einen oder durch zwei Bezeichner, die die jeweilige Beziehungsrichtung ausdrücken, benannt werden.

Beispiel: Kunde *besitzt* Konto ist ein zweistelliger Beziehungstyp mit den Kardinalitäten 1:n (ein Kunde kann n Konten haben, ein Konto hat immer einen Besitzer – zumindest in unserem Beispiel), wobei Konto optional sein kann (ein Kunde kann auch kein Konto haben). Die inverse Beziehung heißt: Konto *gehört* Kunden.

Entitäts-/Beziehungsmodell: Menge von Entitätstypen und Entitätsbeziehungstypen, die eine für ein Anwendungssystem relevante Sicht des Gegenstandsbereiches beschreiben. Synonym: Datenmodell, Informationsstruktur

Entitäts-/Beziehungsdiagramm: Graphische Repräsentation der Entitätstypen und Entitätsbeziehungstypen in ihrem Zusammenhang.

Funktionsbeziehung: Verknüpfung von mehreren – in der Regel genau zwei – Funktionen.

Eine Funktionsbeziehung kann eine Zerlegungsbeziehung (Funktion A wird verfeinert zu Funktionen A21 und A22), eine Steuerungs- oder Kontrollbeziehung (Funktion A löst Funktion B aus) oder eine Kommunikationsbeziehung (Funktion A übermittelt Nachrichten an Funktion B) sein. Im dritten Fall wird die Menge der von Funktion zu Funktion übermittelten Nachrichten oft auch als Datenfluß bezeichnet.

Funktions-/Beziehungsdiagramm: Graphische Repräsentation der Funktionen und bestimmter ausgewählter Funktionsbeziehungen im Zusammenhang.

(Anwendungs-) Funktionssicht: Menge der Attribute, die für die Definition einer (Anwendungs-) Funktion erforderlich sind.

Bemerkung: Eine (Anwendungs-) Funktion besteht in der Regel aus mehreren Teilsichten auf Attribute verschiedener Entitätstypen.

Anwendungssicht: Gesamtheit der Anwendungsfunktions-sichten eines Anwendungssystems.

Benutzersicht: Menge der Attribute, die eine Anwendungsfunktion zur Kommunikation mit dem Benutzer verwendet.

Schnittstelle: Eine Schnittstelle ist eine Menge von Vereinbarungen, die zur Beschreibung des Zusammenwirkens von Systemen oder Systemteilen getroffen werden. Diese können sich auf aktive oder passive Elemente beziehen. In Abhängigkeit davon spricht man von funktionalen oder Daten-Schnittstellen.

Schnittstellen-Element: Einzelnes aktives oder passives Element eines Systems, das Teil einer Schnittstelle ist.

6. Gegenwärtige Trends, Ausblick

Wir haben versucht, ein einheitliches und zusammenhängendes deutschsprachiges Terminologiegerüst für die Analyse und die Modellierung von Software-Anwendungssystemen anzugeben. Diese Terminologie spiegelt den gegenwärtigen „Stand der Kunst“ in einem sich stark verändernden Feld wider. Für diesen Wandel sind verschiedene Faktoren verantwortlich. Einmal sind die „frühen“, der eigentlichen Programmierung vorgelagerten Phasen der Software-Entwicklung in den letzten Jahren stark in das Blickfeld des Interesses gerückt. Methoden und Verfahren für diese Phasen wie „Structured Analysis“ oder das Entity/Relationship-Modell sind sehr populär geworden und haben die Terminologie stark beeinflusst. Weiter wurde ein Gebiet (nämlich die Datenmodellierung), das traditionell als Domäne der Datenbank-Technologie galt, für die Software-Technologie erschlossen. Damit prallten die Begriffswelten zweier Fachgebiete aufeinander, die sich vorher weitgehend unabhängig voneinander entwickelt hatten.

Dieses Zusammenwachsen verschiedener Zweige der Informatik läßt sich auch bei der Werkzeug-Unterstützung beobachten. Werkzeuge für die Verwaltung von „Metadaten“ (d. h. Daten über Daten – und oft auch über Funktionen und Operationen) wie Projektbibliotheken und Datenlexika haben sich zunächst unabhängig voneinander entwickelt, erstere in der Softwaretechnik, letztere im Zusammenhang mit Datenbanken. Mittlerweile ist klar geworden, daß eine durchgängige Anwendungsentwicklung auch durchgängige Verwaltungswerkzeuge erfordert [13]. Konzepte für durchgängige Entwicklungs-Datenbanksysteme beginnen sich zu konsolidieren, bringen aber zwangsläufig wieder Änderungen und Neuerungen der Terminologie mit sich. Dieser Trend wird verstärkt durch das Bemühen großer Hersteller, das Neuartige ihrer Entwicklungen durch eine neuartige Terminologie zu unterstreichen.

Aber auch auf methodischem Gebiet vollzieht sich zur Zeit ein tiefgreifender Wandel in Richtung auf eine einheitliche Betrachtung des gesamten Software-Entwicklungsprozesses, die erstmals durchgängige Verfahren und damit in absehbarer Zukunft auch eine abgestimmte, konsistente Terminologie für dieses Gebiet erwarten läßt. Dieser Wandel ist mit dem Schlagwort „objektorientiert“ verbunden. Als Programmierstil und als (technisches) Entwurfs-Paradigma hat der objektorientierte Ansatz seine Bewährungsprobe längst bestanden, wie erfolgreiche Sprachen und Entwurfssysteme wie Smalltalk oder Eiffel bestätigen [23]. Aber erst im Zusammenhang mit einer „objektorientierten Analyse“ [5, 11, 30] bzw. „objektorientierten Anwendungsmodellierung“ [15] wird dieser Ansatz wirklich durchgängig und verspricht erstmals eine ganzheitliche Behandlung des Software-Entwicklungsprozesses ohne Methoden- und Strukturbrüche.

Uns als Terminologie-Gruppe hat sich die Frage gestellt, inwieweit wir diese neuesten Entwicklungen in die vorliegende Arbeit aufnehmen sollten. Die Erfahrung in unserer eigenen Gruppenarbeit hat uns gelehrt, daß es hoffnungslos ist, eine gerade laufende Technologieentwicklung bis zu ihrem Abschluß abzuwarten, um sie dann auch noch in die Terminologiedarstellung aufzunehmen. Wir haben daher beschlossen, weder neu aufkommende Begriffe im Bereich der Werkzeugunterstützung wie repositories (vgl. [6]) noch spezifisch unterlegte Begriffe aus der objektorientierten Welt wie „Klasse“, „Instanz“, „Nachricht“, „Methode“ (oder den neu zu definierenden Objektbegriff selbst) in die vorliegen-

de Darstellung aufzunehmen. Dies sollte späteren Arbeiten vorbehalten bleiben, für die allerdings nach unserer Einschätzung noch einige Zeit der Erprobung und Konsolidierung der neuen Techniken notwendig ist.

Literatur

1. Achatzi, G., Mistelbauer, H., Münzenberger, H., Sinz, E.J., von Stülpnagel, A.: Effektivs Datendesign - Praxis-Erfahrungen (Hrsg. G. Müller-Ettrich). Köln: Rudolf Müller 1989
2. ESPRIT Consortium AMICE: Open System Architecture for CIM. Res. Rep. ESPRIT. Berlin: Springer 1989, 2nd ed. 1993: CIMOSA
3. Burghardt, M.: Projektmanagement: Leitfaden für die Planung, Überwachung und Steuerung von Entwicklungsprojekten. Berlin - München: Siemens 1988
4. Chen, P.P.: The entity relationship model: Toward a unified view of data. ACM Trans. DB Syst. 1, 9-36 (1976)
5. Coad, P., Yourdon, E.: Object-oriented Analysis. Englewood Cliffs: Yourdon 1990
6. Corzilius, R.: AD/Cycle - Ziele, Konzepte und Funktionen. München: Oldenbourg 1992
7. Denert, E.: Software-Engineering - Methodische Projektentwicklung. Berlin: Springer 1991
8. DIN 44300: Informationsverarbeitung - Begriffe. Berlin: Beuth 1982
9. DIN 69901: Projektwirtschaft - Projektmanagement, Begriffe. Berlin: Beuth 1987
10. Dittrich, K.R.: Objektorientierte Datenbanksysteme. Inf.-Spektrum 12, 215-220 (1989)
11. Ferstl, O., Sinz, E.: Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM). Wirtschaftsinformatik 32, 566-581 (1990)
12. Hesse, W., Keutgen, H., Luft, A. L., Rombach, D.: Ein Begriffssystem für die Softwaretechnik - Vorschlag zur Terminologie. Inf.-Spektrum 7, 200-213 (1984)
13. Hesse, W., Merbeth, G., Frölich, R.: Software-Entwicklung: Vorgehensmodelle, Projektführung und Produktverwaltung. Handbuch der Informatik, Band 5.3. München: Oldenbourg 1992
14. Hesse, W.: Two metamodels for application system development - conventional vs. object-oriented approach. In: M. Broy, M. Wirsing (eds.): Methods of Programming Methodology, LNCS 544, pp. 3-18. Berlin: Springer 1991
15. Hesse, W.: Objekt-orientierte Anwendungsmodellierung - ein Weg zu einem durchgängigen Software-Entwicklungsprozeß. In: G. Kugel (Hrsg.): Praxiserprobte Software-Entwicklungswerkzeuge im Überblick, Reihe Kontakt und Studium. Böblingen: Expert-Verlag 1992
16. IBM: Data Analysis. UK Form 26-8101-0, IBM United Kingdom Ltd. Technical 1981
17. IEEE: Standard Glossary of Software Engineering Terminology. ANSI/IEEE Std 729-1983
18. ISO 9000: Qualitätsmanagement- und Qualitätssicherungsnormen - Leitfaden zur Auswahl und Anwendung. Berlin: Beuth 1990
19. James Martin Associates Ltd.: IEM, Information Engineering Methodology, Introduction. 1987
20. Kühnel, B., Partsch, H., Reinshagen, K.P.: Requirements Engineering - Versuch einer Begriffsklärung. Inf.-Spektrum 10, 433 (1987)
21. De Marco, T.: Structured analysis and system specification. Englewood Cliffs: Prentice-Hall 1978
22. Mayr, H.C., Dittrich, K.R., Lockemann, P.C.: Datenbankentwurf. In: P.C. Lockemann, J.W. Schmidt (Hrsg.): Datenbank-Handbuch. Berlin: Springer 1987
23. Meyer, B.: Object-oriented software construction: Englewood Cliffs: Prentice-Hall 1988
24. Nijssen, G.M., Halpin, T.A.: Conceptual schema and relational database design - a fact oriented approach. Englewood Cliffs: Prentice-Hall 1989
25. Olle, T.W., et al.: Information system methodologies. A framework for understanding. Reading: Addison-Wesley 1988
26. Ortner, E., Söllner, B.: Semantische Datenmodellierung nach der Objekttypenmethode. Inf.-Spektrum 12, 31-42 (1989)
27. Pepper, P.: Grundlagen der Informatik. Automatisierungstechnik 36, Heft 8 (1988)
28. Scheer, A.-W.: Architektur integrierter Informationssysteme. Grundlagen der Unternehmensmodellierung. Berlin: Springer 1991, 2. Aufl. 1992
29. Scheschonk, G., Vogt, A.: Die Erstellung formaler Fachkonzepte in der Software-Industrie. Informatik-Fachberichte 188, 18. Jahrestagung der GI. Hrsg. R. Valk. Berlin: Springer 1988
30. Shlaer, S., Mellor, S.J.: Object-oriented analysis: Modelling the world in data. Englewood Cliffs: Yourdon 1988
31. Spitta, T.: Software Engineering und Prototyping. Eine Konstruktionslehre für administrative Softwaresysteme. Berlin: Springer 1989
32. VDI/VDE 3694: Lastenheft/Pflichtenheft für den Einsatz von Automatisierungssystemen - Entwurf. Düsseldorf: Verein Deutscher Ingenieure 1989
33. Vetter, M.: Aufbau betrieblicher Informationssysteme mittels konzeptioneller Datenmodellierung. Stuttgart: Teubner 1986
34. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. 26, 276-292 (1987)

Eingegangen 21. 12. 1992; in überarbeiteter Form 8. 11. 1993

Prof. Dr. Wolfgang Hesse
 FB Mathematik/Informatik der Philipps-Universität
 Lahnborg, D-35032 Marburg

Dipl.-Betriebswirt Georg Barkow
 Wulfsdorfer Weg 91, D-22359 Hamburg

Hubert von Braun
 Manager Software Products GmbH
 Landsberger Str. 439, D-81241 München

Dipl.-Inform. Hans-Bernd Kittlaus
 IBM Deutschland Informationssysteme GmbH
 D-70548 Stuttgart

Prof. Dr. Gert Scheschonk
 C.I.T. Communication and Information Technology GmbH
 Ackerstr. 71-76, D-13355 Berlin

Index zu Teil 1 und 2 (mit Verweisen auf Kapitel und Abschnitte)

Abbildtheorie 2.1
 Abbildung
 sprachliche 2.1
 Ablaufmodell 4.3
 Ablauforganisation 2.4
 Abstraktion 3.3
 Abstraktionsebene 3

aktives Element 3.1
 Anforderung 4.2
 definieren 4.2
 Anforderungsdefinition 4.2
 Anforderungskatalog 4.2
 Angehörige von Fachbereich/-abteilungen 3.1, 3.5

Anwender 3.1
 Anwendung 1, 2.4, 3.1, 3.3
 Anwendungsentwicklung 3.1, 3.3, 6
 Anwendungsfunktion 3.3, 3.5, 5.2
 Anwendungsfunktionsicht 3.5, 5.3
 Anwendungsmodell 3.5, 4.3, 5
 Anwendungsmodellierung 1, 3.3, 6

- Anwendungsprogramm 2.4
- Anwendungssicht 3.5, 5.3
- Anwendungssoftware 4.7
- Anwendungssystem 1, 2.4, 3, 3.2, 3.3, 4, 4.1, 4.6, 4.8, 5, 5.2, 6
- Arbeitsablaufbeschreibung 4.4
- Attribut 4.3, 5.1
- Attribut(-typ) 3.3, 5.1, 5.3
- Attributwert 3.5, 5.1
- Aufbauorganisation 2.4
- Aufgabe 2.3, 2.4, 3.1, 3.5, 5.1
- Aufgabenanalyse 2.4
- Aufgabendefinition 4
- Aufgabengegenstand 2.4
- Aufgabenträger 2.4
- Auftraggeber 4.6
- Auftragnehmer 4.6
- Aufwandschätzung 4.6
- Automatisierung einer Funktion festlegen 4.4

- Begriffsbildung 1, 2, 2.1, 2.2
 - in der Umgangssprache 2.2
- Begriffsdefinition 2.2
- Begriffskonsolidierung 1
- Begriffssystem 1
- Begriffssystematik 3.3
- Begriffswelt 1
- Benutzer 3.1, 3.5, 5.3
- Benutzersicht 5.3
- Benutzungsschnittstelle 4.5
 - graphische 4.5
- betriebliches Informations- und Kommunikationssystem (IKS) 2.4
- Betriebsart 4.3, 4.4
 - zuordnen 4.4
- Bezeichner 5.3
- Business Process 5.2

- data dictionary 3.3
- Daten 2.3, 6
- Datenaspekt 4.8
- Datenbank 3.3, 6
- Datenmodell 3.5, 4.3, 5.3
- Datenmodellierung 6
- datenorientiert 3.2, 5.1
- Dialog 4.5
- Durchführbarkeitsstudie 4.6
- DV-Funktion 3.5, 5.2
- DV-System 4.4
- DV-Systemarchitekt 3.1, 3.5

- Ebene
 - subjektive 2
- Element 5
 - aktives 3.2, 3.5
 - passives 3.2, 3.5
 - zusammenhangsbildendes 3.2, 3.5
- Elementarfunktion 4.3, 5.2
- end user 3.1
- Entität 2.2, 3.5, 5.1, 5.3
- Entitäts-/Beziehungsdiagramm 5.3
- Entitäts-/Beziehungsmodell 5.3
- Entitäts-Beziehung 5.3
- Entitäts-Beziehungstyp 5.3
- Entitätsmenge 3.5, 5.1
- Entitätstyp 3.3, 3.5, 4.3, 5.1, 5.3
- Entwicklungsphasen 3.5

- Ereignis 2.3, 2.4, 3.3, 3.5
- Exemplar-Ebene 3.3

- Fachkonzept 4
- Fachsprache 2, 2.2
- feasibility study 4.6
- Fenster 4.5
- Fenstertechnik 4.5
- Formular 4.5
- Funktion 3.3, 5.1, 5.2, 5.3, 6
- funktionaler Aspekt 4.8
- Funktions-/Beziehungsdiagramm 5.3
- Funktionsbeschreibung 4.3
- Funktionsbeziehung 5.3
- Funktionshierarchie 5.2
- Funktionskomplex 4.3
- Funktionsmodell 3.5, 4.3, 4.4
- funktionsorientiert 3.2, 5.2

- Gegenstand 2, 3.2, 3.5, 4.3, 5.1
- Gegenstandsbereich 3.5, 4.1, 4.3, 5.1
- Geschäftsprozeß 2.4

- Hauptfunktion 4.3, 5.2

- Information 2.3, 4
- informations-struktureller Aspekt 4.8
- Interaktionsdiagramm 4.5
- Interaktionsentwurf 4.5
- interaktiver Betrieb 4.4
- Istzustand beschreiben 4.1

- Kardinalität 5.3
- Kommunikation 2.3, 5.3
- Konditionalität 5.3
- Kosten-Nutzen-Analyse 4.6
- Kunstsprache 2

- Lösungsmöglichkeiten erarbeiten 4.1
- Lastenheft 4.2
- Liste 4.5

- Maske 4.5
- Maskentechnik 4.5
- Meilenstein 4
- Mengengerüst 4.6
- Mensch 2.1, 2.3, 2.4, 3.1
- Mensch-Maschine-Interaktion 4.5
- Merkmal 2.3
- Meta-Ebene 3.3
- Metadaten 6
- Modell 4.3
- Modellierung 4

- Nachricht 2.3, 5.3, 6

- objektorientierter Ansatz 6
- Optionalität 5.3
- Organisationseinheit 2.4
- Organisator 3.5
- organisatorische Art 4.1
- organisatorischer Aspekt 4.8
- organisatorisches System 2.4, 4.4

- passives Element 3.1
- Pflichtenheft 4.2
- Problem 3.2
- Problembereich 3

- Produktqualität 4.7
- Programmierer 3.1
- Projekt initialisieren 4.6
- Projekt-Initialisierung 4.1
- Projektauftrag 4.6
- Projektziele 4.1
 - festlegen 4.1
- Prototyping-Ansatz 3.4
- Prozeßqualität 4.7

- Qualitätseigenschaft 4.7
- Qualitätsmerkmal 4.7
- Qualitätssicherung 4, 4.7
 - produktorientierte 4.7
 - prozeßorientierte 4.7
- Qualitätssicherungssystem 4.7

- repositories 3.3, 6

- Sachverhalt 2.3, 3.3, 3.5
- Sachziel 2.4
- Schnittstelle 5.3
- Schnittstellen-Element 5.3
- semiotisches Dreieck 2
- Software-Entwicklung 1
- Software-System 2.4
- Sprachelemente 3.2
- Stapelbetrieb 4.4
- Stark-/Schwachstellen analysieren 4.1
- Stelligkeit 5.3
- Subentitätsmenge 5.1
- Subentitätstyp 5.1
- Subsystem 2.3
- System 2.3, 4.6, 5.1
 - organisatorisches 2.4, 4.4
 - technisches 2.4, 3.2, 4.4
- Systembeschreibung 3.2
- Systemelement 2.3
- Systementwickler 3.1
- Systementwicklung 3.4
- Systemkomponente 2.3

- Technik 2.4, 4, 6
- technische Art 4.1
- technisches System 2.4
- Typ-Ebene 3.3

- Unternehmensdatenmodell 4.8
- Unternehmensmodell 4.8
- unternehmensweite Modellierung 4.8
- Untersuchungsbereich 4.1

- Verfahrensziel 2.4
- Vorgang 2.3, 3.2, 3.5
- Vorgehensmodell 3.4, 4

- Wahrnehmung
 - subjektive 2.1
- Wartungsfachleute 3.1
- Wertebereich 5.1
- Wissen 2.1, 2.3

- Zielkatalog 4.1
- Zirkeldefinition 2.2
- zusammenhangsbildendes Element 3.2, 5.3
- Zustand 2.3