

Konzepte und Aufbau einer Repository-gestützten Software-Entwicklungsumgebung

Andreas Schuh, Ludwigshafen

Frank Schönthaler, Karlsbad

Schlüsselworte

Software Engineering, CASE, ORACLE*CASE, Repository, Data Dictionary, Vorgehensmodell

Zusammenfassung

Die EDV-Gruppe des Bereichs Umweltschutz, Arbeitssicherheit und Energie der BASF AG, Ludwigshafen hat in Zusammenarbeit mit der PROMATIS Informatik, Karlsbad einen Data Dictionary als Grundlage für eine Repository-gestützte Software-Entwicklungsumgebung erstellt. Dabei diente ORACLE*CASE als Ausgangspunkt, wobei das CASE*Dictionary mit dem Data Dictionary integriert wurde. Wir beschreiben die Zielsetzungen dieses Projektes vor dem Hintergrund der heutigen DV-Landschaft und die Rolle eines Repositories im eigenentwickelten Vorgehensmodell für Softwareprojekte. Wir formulieren das Datenmodell des Data Dictionary aus der Sichtweise der Benutzerkreise und skizzieren die zugehörige Funktionshierarchie. Dargestellt wird ebenfalls die Projektabwicklung unter Einsatz der Werkzeuge ORACLE*CASE und INCOME. Wir schließen mit Hinweisen zum praktischen Einsatz und einem Ausblick auf die erwarteten Effekte und Nutzwirkungen dieser EDV-Infrastrukturmaßnahme.

1 Hintergrund des Projekts

Die Softwarekrise ist derzeit eines der beherrschenden Themen der DV-Industrie. Bei den Anwendern manifestiert sie sich in einem Mißverhältnis zwischen Wartung alter und Entwicklung neuer Software. Durch den dominierenden Wartungsanteil an den Software-Arbeitsleistungen - man spricht von bis zu 80% aller Software- und DV-Personalaufwendungen bei den Anwenderunternehmen - werden für das operative Kerngeschäft der Unternehmen dringend notwendige Neuentwicklungen blockiert oder kritisch verzögert. Tieferliegende Ursache dieser Situation ist die innere Struktur und Qualität der alten Softwaresysteme. In den meisten Fällen muß man diese aus heutiger Sicht als schlecht bezeichnen. In ihnen bilden sich Fehler, die in der Vergangenheit im Software-Entwicklungsprozeß gemacht wurden, ab und führen zu den bekannten Problemen. In heutiger Terminologie ausgedrückt, handelt es sich dabei meist um Fehler in den frühen Phasen der Projektabwicklung, die später auch durch noch so gute Realisierung nicht mehr wettgemacht werden konnten.

Im folgenden sind einige gravierende Fehler aus den frühen Phasen aufgeführt, die zu wartungsintensiven, ihre Funktionalität verfehlenden oder gar nicht abgeschlossenen Softwaresystemen führen:

- mangelnde Zieldefinition des Systems und Projekts
- keine frühzeitige Einbeziehung der Anwender
- keine exakte und verbindliche Definition der relevanten fachlichen Begriffe
- vorzeitige Dominanz von Implementierungsaspekten über die fachliche Analyse
- fehlende Dokumentation der frühen Phasen
- mangelnde Durchgängigkeit und Auswertbarkeit der Dokumentation

Will man also zukünftig bessere Resultate erzielen, muß man an diesen Fehlern ansetzen und

- die frühen Phasen der Softwareentwicklung deutlicher strukturieren und vollständig dokumentieren
- einen standardisierten Satz von Entwicklungsdokumenten definieren und vorschreiben
- die Dokumente durchgängig durch alle Phasen der Softwareentwicklung hinweg verfügbar, besser sogar maschinell auswertbar, halten und pflegen.

Basis einer effektiven Softwareentwicklung, die zu funktionierenden und wartungsfreundlichen Programmen führt, muß demnach ein Repository sein, das alle Dokumente der Softwareentwicklung aufnehmen, verwalten und bereitstellen kann und integrierende Basis für die eingesetzten Werkzeuge ist, die diese Informationen verwerten.

An dieses Repository werden hohe Anforderungen gestellt:

- Es soll die Definition der Anwenderbegriffe unterstützen, ohne diesen Modellvorschriften aufzuerlegen.
- Es soll sich flexibel im Rahmen vorhandener Vorgehensmodelle einsetzen lassen.
- Es soll auf einem Standard-DBMS aufsetzen, um eine individuelle Anpassungs- und Erweiterungsfähigkeit zu gewährleisten.

- Es soll mit den typischen CASE-Werkzeugen für die einzelnen Phasen und Methoden integrierbar sein.

Von den derzeit kommerziell angebotenen Repository-Systemen und CASE-Tools erfüllt keines diese Anforderungen. Mit ORACLE*CASE 5 konnte jedoch ein Werkzeug gefunden werden, das als Basis eines solchen Repositories verwendet werden konnte. Die fehlende Funktionalität sollte durch Entwicklung zusätzlicher Komponenten erreicht werden.

Nachdem dieses Konzept, ein maßgeschneidertes Repository auf dem CASE*Dictionary basierend aufzubauen, in der Strategieweise des Projektes als tragfähig erkannt wurde, entschloß sich der Funktionsbereich Umweltschutz, Arbeitssicherheit und Energie der BASF AG (im folgenden mit seinem organisatorischen Kürzel „DU“ bezeichnet), diesen Weg in Richtung auf eine zukunftssichere DV-Infrastruktur zu gehen. Der Funktionsbereich DU gehört zu den großen Verwendern von Daten innerhalb der BASF. In einer bereits 1989 durchgeführten Erhebung wurde ermittelt, daß DU Daten aus mehr als 60 Datenbanken verwendet. Diese Datenbanken gehören entweder DU oder anderen BASF-Einheiten. Es wurde festgestellt, daß der ständig weiter steigende Informationsbedarf nur mit Hilfe koordinierender zentraler Einrichtungen in Zukunft effizient und kostengünstig gedeckt werden kann. Als erste Maßnahme wurde daher der DU Data Dictionary (DUDD) als umfassendes Repository zur Verwaltung der Informations-Metadaten geschaffen. Daß eine derartige Entwicklung zuerst in einem einzelnen Unternehmensbereich der BASF beginnt und bei Bewährung in weitere Bereiche und eventuell das ganze Unternehmen übernommen wird, ist Ausdruck des gültigen EDV-Konzeptes der „zentral unterstützten Dezentralisierung“, welches die BASF seit 1987 konsequent verfolgt.

In diesem Artikel werden wir darstellen,

- welche Rolle ein Repository innerhalb der Software-Projektentwicklung einnimmt,
- welche Inhalte im maßgeschneiderten DUDD enthalten sind,
- wie die Projektentwicklung zur Erstellung des DUDD im Rahmen des beschriebenen Vorgehensmodells erfolgte,
- und wie sich Einsatz und künftige Perspektiven des DUDD darstellen.

2 Vorgehensmodell für Softwareprojekte

Die effektive Nutzung eines Repositories und darauf aufsetzender CASE-Tools ist nur dann möglich, wenn diese in einem geeigneten organisatorischen Rahmen eingebettet sind. Demnach mußte zunächst der Prozeß der Abwicklung von Softwareprojekten bei DU analysiert werden. Hierbei waren nicht nur ORACLE-Projekte, sondern auch typische 3GL-Entwicklungsprojekte zu betrachten.

Ausgehend von den Ergebnissen dieser Analyse und der in [Bar90b] beschriebenen CASE*Methode wurde ein Vorgehensmodell für die Abwicklung von Softwareprojekten bei DU definiert (siehe Abb.1).

In diesem Vorgehensmodell sind sowohl die Erkenntnisse aus dem heute anerkannten Stand der Software-Entwicklungsmethodologie als auch die Vorgaben zur Projektabwicklung durch das Unternehmen eingeflossen. Diese Vorgaben wurden BASF-spezifisch implementiert, sind aber so flexibel definiert, daß sie auch abweichenden Verhältnissen angepaßt werden könnten. In diesem Modell sind sowohl die Aktivitäten als auch die relevanten Objekte (i.a. zu erstellende Dokumente) des Software Life Cycle berücksichtigt.

Das Vorgehensmodell wurde in Form von *Petri-Netzen* formal spezifiziert. Hierbei wurden die INCOME CASE-Produkte¹ eingesetzt (siehe [SMM91]). Abbildung 1 zeigt die oberste Ebene der entstandenen Petri-Netz-Hierarchie.

In den Petri-Netzen werden die Aktivitäten graphisch durch Rechtecke repräsentiert. Für die Aktivitäten können zusätzlich Informationen wie textuelle Beschreibungen, Randbedingungen und Referenzen auf die zu verwendenden Methoden und Werkzeuge angegeben werden. Falls Aktivitäten durch weitere Netze verfeinert sind, sind die entsprechenden Rechtecke dick umrandet.

Die Kreise repräsentieren Speicher für Objekte, die im Software Life Cycle erzeugt und verbraucht werden. Die Verwendungsart ergibt sich aus der Richtung der anliegenden Kanten. Die Struktur der Objekte ist durch die Zuordnung von Entities und Attributen des Metamodells (siehe hierzu Kapitel 3) definiert. Damit ist für jede Aktivität unmittelbar ersichtlich, welche Objekte in welcher Weise verarbeitet werden.

Interessant ist, daß das Vorgehensmodell mit dem INCOME/Simulator (siehe [PRO93]) simuliert und graphisch animiert werden kann. In dieser Weise können die wesentlichen Konzepte des Vorgehensmodells anschaulich kommuniziert werden.

Das in Abb. 1 dargestellte Vorgehensmodell bildet nicht nur den Rahmen für die Abwicklung zukünftiger Anwendungsprojekte, sondern wurde auch in der Entwicklung des DUDD selbst konsequent umgesetzt.

3 Inhalt des Repositories

3.1 Positionierung des DUDD

Über die Definition der Begriffe „Repository“ und „Data Dictionary“ herrscht keineswegs Einigkeit in der DV-Welt, sondern es existiert eine Vielfalt von Bedeutungen und Interpretationen. Wir definieren diese Begriffe, indem wir die gesamte Skala der Mannigfaltigkeit an Metainformation, die zu einem Softwaresystem und dessen Entwicklungsprozeß und Lifecycle festgehalten werden kann, betrachten. Am semantisch armen Ende dieser Skala finden wir die reine Metainformation zum konzeptionellen und physikalischen Daten(bank)schema. Beispiel hierfür ist der ORACLE Data Dictionary, wie er dem RDBMS beigelegt ist. Am semantisch reichen bzw. idealerweise vollständigen Ende der Skala steht das Repository, welches die Datengrundlage einer integrierten Software-

¹ Produkt der PROMATIS Informatik, Karlsbad b. Karlsruhe

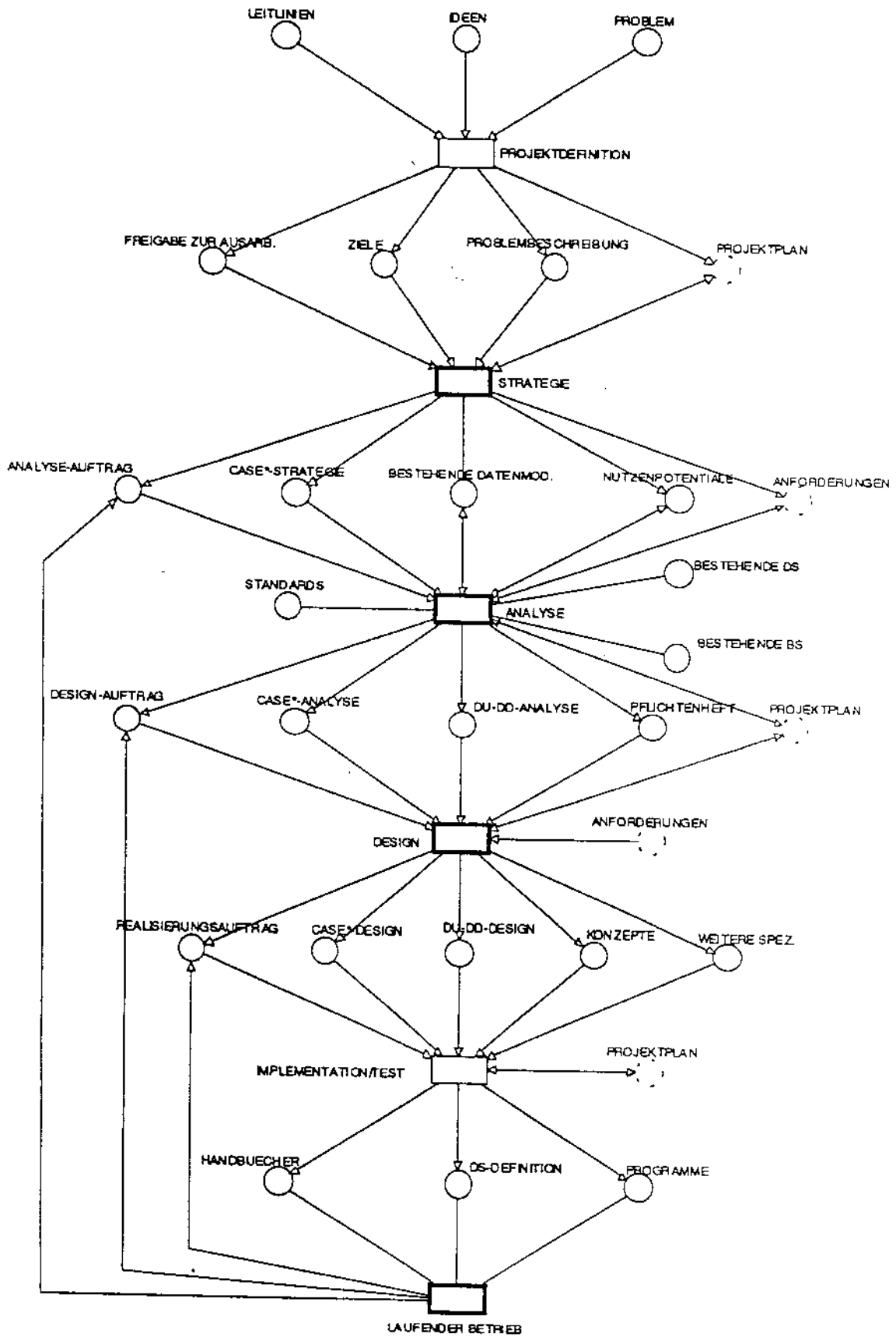


Abbildung 1: Vorgehensmodell für Softwareprojekte

Produktionsumgebung bereitstellt und alle denkbaren Informationen zum Softwareprojekt enthält wie zum Beispiel:

- Fachliche Begriffsdefinitionen
- Anforderungsmodell
- Systementwurf
- Implementationsbeschreibung
- Projektabwicklungsdaten
- Aufzeichnung der Entwicklungsgeschichte
- Versions- und Freigabeverwaltung

Ein solches „ideales“ Repository, welches alle diese Informationen bereithalten und den integrierten Planungs-, Analyse-, Design- und Implementationswerkzeugen zur Verfügung stellen kann, ist unseres Wissens bisher nicht realisiert worden. AD/Cycle von IBM und CDD+ von DEC sind zumindest Konzepte bzw. Baukästen für die Entwicklung eines derartigen Repositories. Mit dem offenen Konzept von ORACLE*CASE war es möglich, ebenfalls nach dem Baukastenprinzip den DUDD als Datenbank für Meta-informationen zu entwickeln, welche, wie wir im folgenden sehen werden, nahe am semantisch reichen Ende der Inhaltsskala steht. Die Bezeichnung als „Data Dictionary“ wurde aus rein historischen und praktischen Gründen beibehalten, nachdem sich dieser Begriff, der von der Analyse der Ausgangssituation bei den von DU verwendeten Datenbanken her kommt, erst einmal fest im Bewußtsein der von diesem Projekt tangierten Personen etabliert hatte.

3.2 Datenmodell und grundlegende Themenbereiche des DUDD

Zur Vorbereitung der Datenmodellierung wurden zunächst die zukünftigen Nutzer des DUDD identifiziert und in Form von *Business Units* definiert. Folgende Benutzerkreise wurden ermittelt:

- Nutzer von DV-Anwendungen des Bereichs DU
- Administratoren für Anwendungen, verschiedene technische Ressourcen und den DUDD selbst
- Systemplaner, Entwickler und Projektleiter
- „Strategen“, die sich mit anwendungsübergreifenden Fragestellungen beschäftigen, etwa der Qualitätssicherung oder der Unternehmensmodellierung

Aus der Sichtweise der verschiedenen Benutzerkreise wurde nun das Datenmodell entwickelt. Zur übersichtlichen graphischen Darstellung wurde es in vier Teilmodelle zerlegt, die unterschiedliche Aspekte des DUDD beschreiben:

- Struktur von Anwendungssystemen
- Beschreibung der Größen
- Verwendung der Größen
- Zugriffsrechte

Bereits bei der Datenmodellierung war zu beachten, daß der DUDD in möglichst hohem Maße von der bereits vorhandenen Funktionalität des CASE*Dictionary Gebrauch machen sollte. Um dieser Forderung gerecht werden zu können, mußten im Datenmodell die relevanten Teile des CASE*-Metamodells integriert werden.

Abbildung 2 zeigt exemplarisch einen Ausschnitt des *Entity/Relationship-Diagramms* zur Beschreibung der Struktur von Anwendungssystemen. Die rechts vom Trennstrich dargestellten Entities und Relationships zwischen diesen sind Bestandteile des CASE*-Metamodells und werden hier nur referenziert (shared elements). Die übrigen Entities und Relationships sind Bestandteile, die zur Erweiterung der Repository-Funktionalität erforderlich waren. Durch den dargestellten Ausschnitt des Datenmodells wird es möglich, Benutzerschnittstellen von Nicht-ORACLE-Anwendungen (etwa 3GL-Programme) einschließlich entsprechender Datenverwendungen im DUDD zu verwalten.

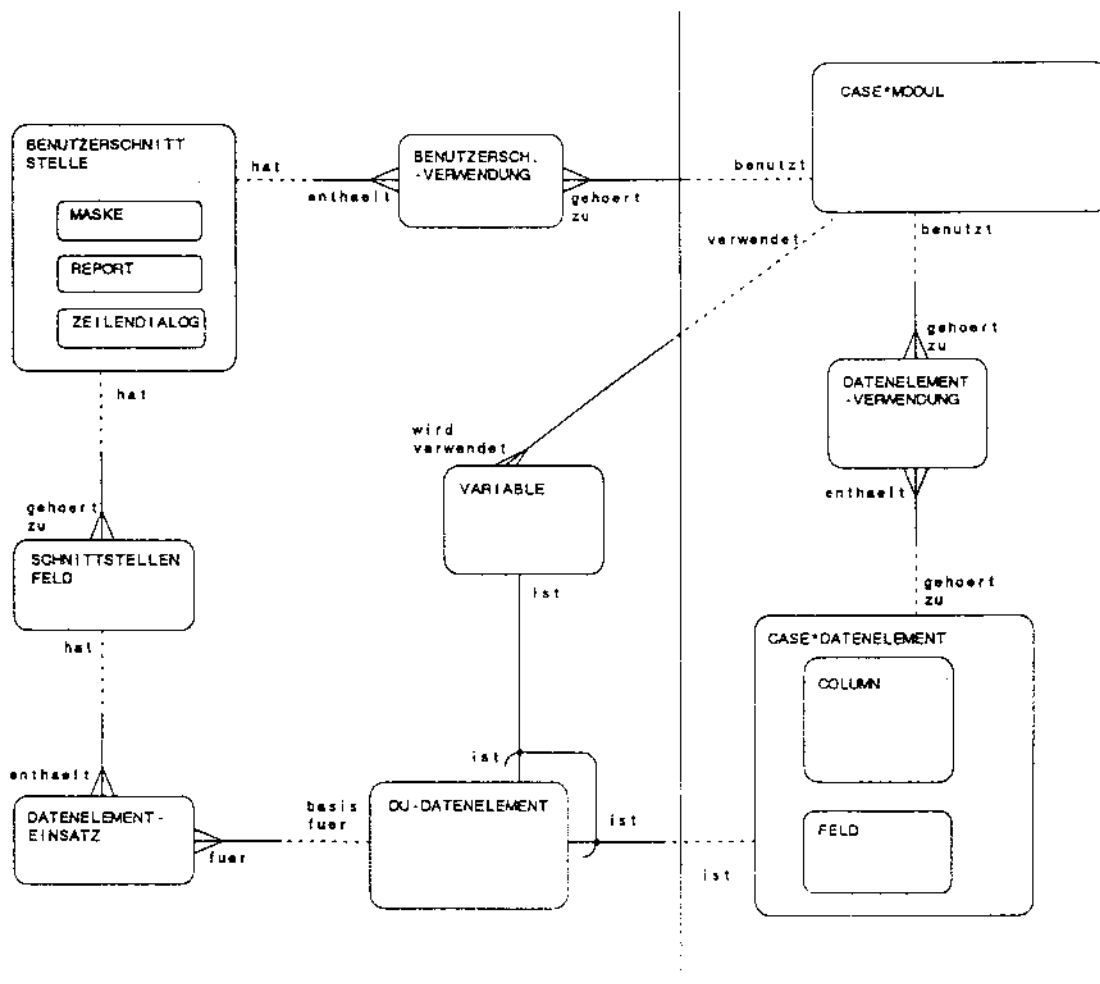


Abbildung 2: Ausschnitt des Datenmodells

Die vier Teilmodelle des DUDD werden im folgenden beschrieben.

Struktur von Anwendungssystemen

Für die Beschreibung von Anwendungssystemen wird auf den bereits in ORACLE* CASE enthaltenen Informationen aufgebaut. Dort können Anwendungssysteme, sogenannte *Applikationen*, durch Datenspeicher- (Tabellen, Views, Datenfiles), deren Datenelemente (Spalten, Felder) und Moduldefinitionen sowie deren Beziehungen beschrieben werden.

Der DUDD ergänzt diese Beschreibung um die Definition von Benutzerschnittstellen und deren Felder, sogenannte Schnittstellenfelder. Die Benutzerschnittstellen beschreiben die Schnittstelle eines Moduls zum Benutzer und sind deshalb eng mit den Modulen aus CASE* verknüpft.

Die Schnittstellenfelder sind zum Beispiel Felder einer Maske und entsprechen normalerweise den Columns aus Tabellen oder Feldern aus Datenfiles. Über diese Beziehung ist nachvollziehbar, welche Datenelemente (Columns, Felder, Variablen) in welchen Benutzerschnittstellen verwendet werden.

Ebenfalls kann im DUDD eingetragen werden, ob ein Anwendungssystem für den Produktionsbetrieb (Einsatz) freigegeben ist.

Ein Ausschnitt der Beschreibung eines Anwendungssystems kann dem Entity/Relationship-Diagramm in Abbildung 2 entnommen werden. Dort werden insbesondere die Aspekte der Erweiterung von CASE* durch die Anknüpfung der neuen Entitätstypen des DUDD exemplarisch dargestellt.

Beschreibung der Größen

Im DUDD können sogenannte *Größen* definiert werden. Eine Größe umfaßt zu dem eigentlichen Zahlenwert einer meßbaren Entität oder einer Textaussage die zur exakten Reproduktion und Interpretation notwendigen Angaben. Sie enthält eine Liste mit Größenelementen, eine formale Beschreibung, Randbedingungen und Quellen.

Größen beschreiben die gesamte Datenlandschaft des Unternehmens. Sie dienen als Basis für die Beschreibung der Herkunft von Informationen und Daten, die in einem bestimmten System existieren oder verwendet werden.

Für die Namensfindung von Größen werden sogenannte *Atome* (atomare Bezeichner) verwendet. Dabei handelt es sich um Wortteile (Wortstämme), die in keine weiteren Bestandteile zerlegt werden können. Diese Atome oder die Kombination solcher Atome bilden die Namen der Größen.

Die Größen werden in Gruppen zusammengefaßt und gegliedert. Die Gruppen wiederum sind in eine Gruppenhierarchie eingebunden.

Die Beschreibung der Größen und ihrer Verwendung (s.u.) waren die wesentlichste Erweiterung, die der DUDD gegenüber dem CASE*Dictionary aufweist. Wozu war es notwendig, eine weitere, quasi vorgelagerte Modellierungsebene einzuführen, wo doch mit der ER-Modellierung das gängigste Datenmodellierungsverfahren in CASE* zur Verfügung steht? Die Erfahrungen aus der Praxis haben gezeigt, daß der nicht-modellierungserfahrene Anwender, auf dessen Angaben die Systemspezifikation schlußendlich beruht, sich mit ER schwer tut. Der Anwender besitzt eine eigene Begriffswelt, geprägt durch seine Arbeitsumgebung, seine Erfahrungen und die Firmenkultur. Diese kann er nur schwer in ER-gemäßer, strukturierter Form reflektieren. Im allgemeinen werden im

Zuge der Planungs- und Analysephasen eines Softwareprojekts zunächst zusammenhanglose Stücke von Informationen erhalten, ohne ein vollständiges oder auch nur teilweises Datenmodell dahinter sehen zu können. Die Größendarstellung erlaubt eine „flache“ Beschreibung und Definition der im Arbeitsablauf des Anwenders auftretenden Daten und Sachverhalte, wenn die Strukturen zunächst nicht nachvollziehbar sind. Die damit vorgenommenen Begriffsbildungen behalten ihre Gültigkeit unabhängig davon, ob später daraus Entitäten oder Attribute oder sofort Datenelemente, Tabellen oder Spalten gebildet werden. Rückwärts, aus der Perspektive der tieferen Modellebenen gesehen, erlauben Größen eine Projektion dieser Termini in die fachliche Welt und Sprache des Anwenders, ohne diesen mit Modellierungsaspekten oder Datenbankterminologie zu belasten.

Die Einführung der Größen bedeutet, daß im DUDD auf drei Modellierungsebenen gearbeitet wird:

- Begriffswelt der Anwender im Größenschema
- ER-Modell
- Logische Daten(bank)ebene und Implementation

Zwischen den drei Ebenen existieren vielfältige Verknüpfungen, die durch entsprechende Beziehungstypen im Metadatenmodell des DUDD repräsentiert werden.

Verwendung der Größen

Die Beschreibung der Verwendung der Größen stellt die Verknüpfung zwischen den Größen und der Beschreibung der Anwendungssysteme her.

Die Größenverwendung wird auf der konzeptionellen Ebene durch eine Zuordnung der Attribute von Entities aus dem Entity/Relationship-Modell zu den Größenfeldern der Größen erreicht. Kann diese Zuordnung nicht vorgenommen werden, da kein Entity/Relationship-Modell vorhanden ist, d. h. die Tabellen bzw. die Datenfiles wurden direkt in ORACLE*CASE eingetragen, so können den Größenfeldern die entsprechenden Datenelemente (Column, Feld eines Datensatzes, Variable) auch direkt zugeordnet werden.

Diese Zuordnungen geben Aufschluß darüber, welche Daten bzw. Informationen in einem System enthalten sind. Bei jeder Systembeschreibung sollte deshalb jedes Datenelement, das für das Unternehmen relevante Informationen trägt, einem Größenfeld zugeordnet sein.

Zugriffsrechte

Die Benutzer von Softwaresystemen und ihre Interaktion mit diesen wurden von Anfang an in das Metadatenmodell mit einbezogen. Dies ergab einerseits über die Rollen der Benutzer die Definition der Business Units, welche essentielle Beiträge zur Strukturierung des Datenmodells und zur Erstellung des Funktionsmodells lieferten. Andererseits entstand so das Teilmodell für die Zugriffsrechte der Benutzer von Software jedweder Art, natürlich auch einschließlich der Benutzer des DUDD selbst.

Im DUDD können verschiedene Informationen über die Zugriffsrechte innerhalb einer Anwendung abgelegt werden. Es werden hierbei die Zugriffsrechte auf

- die Anwendung,
- die Module einer Anwendung,
- die Datenspeicher (Tabellen, Views, Datenfiles) eines Moduls und
- die Datenelemente (Columns, Felder, Variablen) eines Moduls

unterschieden.

Darüber hinaus können auch Zugriffsrechte zu den genannten Objekten unabhängig von Anwendungen oder Modulen definiert werden.

Das Ziel dieser Zugriffsrechtsdefinitionen ist eine Erhöhung der Transparenz, wer mit welchen Anwendungen arbeitet, um vor allem in der Wartungsphase von Anwendungen den Anwenderkreis, der von Änderungen oder Erweiterungen betroffen ist, schneller und genauer erfassen zu können.

3.3 Funktionshierarchie des DUDD

Die Funktionshierarchie des DUDD orientiert sich an den bereits erwähnten Business Units (Benutzerkreisen) und dem in Abschnitt 3.2 beschriebenen Datenmodell. Entsprechend der Aufgaben und Tätigkeiten der verschiedenen Benutzer werden Funktionen zur Verfügung gestellt. Weiterhin wird sichergestellt, daß unter Voraussetzung der entsprechenden Zugriffsrechte die Elementaroperationen (SELECT, INSERT, UPDATE, DELETE) auf den verschiedenen Entitätstypen des DUDD ausgeführt werden können. Dies wird durch Zugriffsmöglichkeiten auf die einzelnen Attribute der Entitäten weiter verfeinert. Die Interpretation der Benutzerkreise und somit der Funktionshierarchie läßt zu, daß eine Person verschiedene Benutzerrollen annehmen kann und dadurch auf verschiedene Äste der Funktionshierarchie zugreift. Dementsprechend wird die Zuordnung der Business Units zur Funktionshierarchie in der *Business Unit/Business Function-Matrix* kontrolliert.

Bei der Funktionsmodellierung wurde zunächst top-down vorgegangen. Allein im Rahmen des Top-down-Vorgehens war es jedoch nicht möglich, ein wirklich vollständiges Funktionsmodell zu entwickeln. Deshalb wurde zusätzlich auf ein Bottom-up-Verfahren zurückgegriffen (siehe [PRO92]), mit dem eine Obermenge von Elementarfunktionen ermittelt werden konnte. Dieses Verfahren nutzt insbesondere das Konzept der Business Views aus, das etwa in [Bar90a] beschrieben wird. Die Elementarfunktionen wurden nach vordefinierten Regeln verschmolzen und in die Funktionshierarchie integriert.

Bei der Ausgabe von Informationen wird weiterhin zwischen Bildschirmausgaben, die in die Bearbeitungsformulare integriert sind, und vordefinierten Reports zur Druckausgabe unterschieden. Neben der Dokumentation des erreichten Standes der Arbeit in einem Softwareprojekt haben diese Reports auch die Funktion, typische Fragen des Managements und der Anwender zu beantworten, wie zum Beispiel:

- Benutzer und ihre Zugriffsrechte
- Anwendungen und berechnete Benutzer
- Potentielle Quellen einer Größe

Bei den vordefinierten Reports wurde weiterhin eine Vielzahl von Qualitätsprüfungen bereitgestellt, die sowohl in der laufenden Entwicklungsarbeit eingesetzt als auch durch

den institutionalisierten Qualitätssicherer bei der Endabnahme durchgeführt werden können. Beispiele für diese Quality Checks sind Reports für

- Größen ohne formale Beschreibung
- Größen ohne Gruppenzuordnung
- Attribute ohne CASE*-Datenelementzuordnung
- Benutzerschnittstellen ohne Modulzuordnung

und so fort, insgesamt 22 derartige Prüfungen auf Vollständigkeit und innere Geschlossenheit der erfaßten Informationen.

Der Benutzer des DUDD wird durch einen Menübaum, der mit SQL*Menu realisiert wurde, durch diese Vielzahl von Möglichkeiten geführt.

4 Durchführung des Projekts

Die Realisierung des DUDD erfolgte nach dem in Kapitel 2 dargestellten Vorgehensmodell. Damit war eine strukturierte Abwicklung nach den allgemein anerkannten Regeln des Software Engineering gewährleistet. Insbesondere wurde auf eine extensive Planungsphase Wert gelegt, um die mit dem Vorhaben zu lösenden Probleme und zu erreichenden Ziele möglichst genau und vollständig zu definieren und die Grobstruktur der Dateninhalte und Funktionen des DUDD für die Detailarbeit der folgenden Projektphasen vorzugeben. In der Planungsphase wurden auch die vorgesehenen Anwenderkreise des DUDD frühzeitig in das Projekt einbezogen, um von Anfang an ein zielgruppengerechtes Produkt zu entwickeln.

Soweit für die Erstellung der im Vorgehensmodell definierten Dokumente einsetzbar, wurde die Projektabwicklung durch die verfügbaren CASE-Werkzeuge gestützt. Im einzelnen waren dies in der Planungs- und Analysephase die Werkzeuge CASE*Dictionary, CASE*Designer, INCOME/Dictionary und INCOME/Designer. Das CASE*Dictionary wurde darüber hinaus auch in der Entwurfs- und Implementationsphase verwendet. Zur Implementation wurde auf die ORACLE SQL-Produkte und die CASE*Generatoren zurückgegriffen.

Nach Fertigstellung des Produkts DUDD wurde das gesamte Projekt in dem gegenüber dem CASE*Dictionary erweiterten Teil nachdokumentiert, um die Vollständigkeit der Dokumente gemäß Vorgehensmodell zu erreichen. In künftigen Projekten, in denen der DUDD von Beginn an als Werkzeug zur Verfügung steht, erfolgt die entsprechende Dokumentation natürlich zum jeweils aktuellen Zeitpunkt in der Projektphase, in der ein Dokument erstellt werden kann oder benötigt wird.

Im folgenden wird die Projektdurchführung in den einzelnen Phasen beschrieben.

4.1 Planung und Analyse

Die Projektgruppe setzte sich in der Planungs- und Analysephase aus drei Mitarbeitern der BASF und zwei Mitarbeitern der PROMATIS zusammen. Die fachlichen Anforderungen wurden in intensiven Gesprächen innerhalb der Projektgruppe erarbeitet, wobei für spezielle Fragestellungen weitere Mitarbeiter von DU hinzugezogen wurden.

Begleitend zu den Projektsitzungen wurden die Anforderungen im CASE*Dictionary formal spezifiziert. Dies hatte den Vorteil, daß die bereits vorliegenden Anforderungen jederzeit auf ihre Qualität überprüft werden konnten. In diesem speziellen Projekt war es auch stets problemlos möglich, die Kommunikation auf der Basis der formalen graphischen Spezifikationen abzuwickeln, da die beteiligten BASF-Mitarbeiter bereits über praktische Erfahrungen mit Software Engineering und speziell ER-Modellierung verfügten. Bei Projekten, in denen EDV-fremde Fachabteilungen involviert sind, ist dies erfahrungsgemäß wesentlich schwieriger.

Die besondere Problematik des Projekts DUDD ergab sich aus der Abstraktheit der Aufgabenstellung. Dies zeigte sich bereits bei der Datenmodellierung, denn es war ja nicht ein „herkömmliches“ Anwendungsdatenmodell, sondern ein Metamodell zu entwickeln bzw. das vorliegende CASE*-Metamodell zu erweitern.

Zum Ende der Analysephase stand im CASE*Dictionary ein umfassend dokumentiertes Anforderungsmodell online zur Verfügung. Es enthielt mehr als 90 neu zu entwickelnde Funktionen und ca. 60 Entitätstypen.

Für die nachfolgende Ausschreibung mußte darauf aufbauend ein am IEEE-Standard angelehntes Pflichtenheft erstellt werden. In dieses waren die CASE*-Dokumente in geeigneter Weise zu integrieren. Das Pflichtenheft enthielt darüber hinaus noch zusätzliche Anforderungen allgemeiner Art, wie die BASF-Standards für Benutzerschnittstellen, Anforderungen an Installation und Betrieb sowie Anforderungen bezüglich der weiteren Projektdurchführung. Auf Grundlage des Pflichtenheftes wurden die nun folgenden Phasen Entwurf und Implementation ausgeschrieben, vergeben und durchgeführt.

4.2 Entwurf und Implementation

Zunächst war ein System- und Datenbankentwurf vorzulegen. Für die nachfolgende Implementation wurde das Gesamtpaket DUDD anhand des Anforderungsmodells in vier Teile zerlegt, die zu festgelegten Zeitpunkten betriebsbereit übergeben werden mußten. Dies hatte den Vorteil, daß die Anwender bereits vor Fertigstellung des Gesamtpakets erste Erfahrungen sammeln konnten.

Technische Abwicklung

Da die Realisierung des DUDD beim externen Auftragnehmer erfolgen sollte, mußte ein Konzept für den gegenseitigen Austausch von Repository-Inhalten und Programmmodulen gefunden werden. Hier kam die Portabilität der ORACLE-Produkte voll zum Tragen. Abbildung 3 zeigt die für das Projekt relevante Konfiguration.

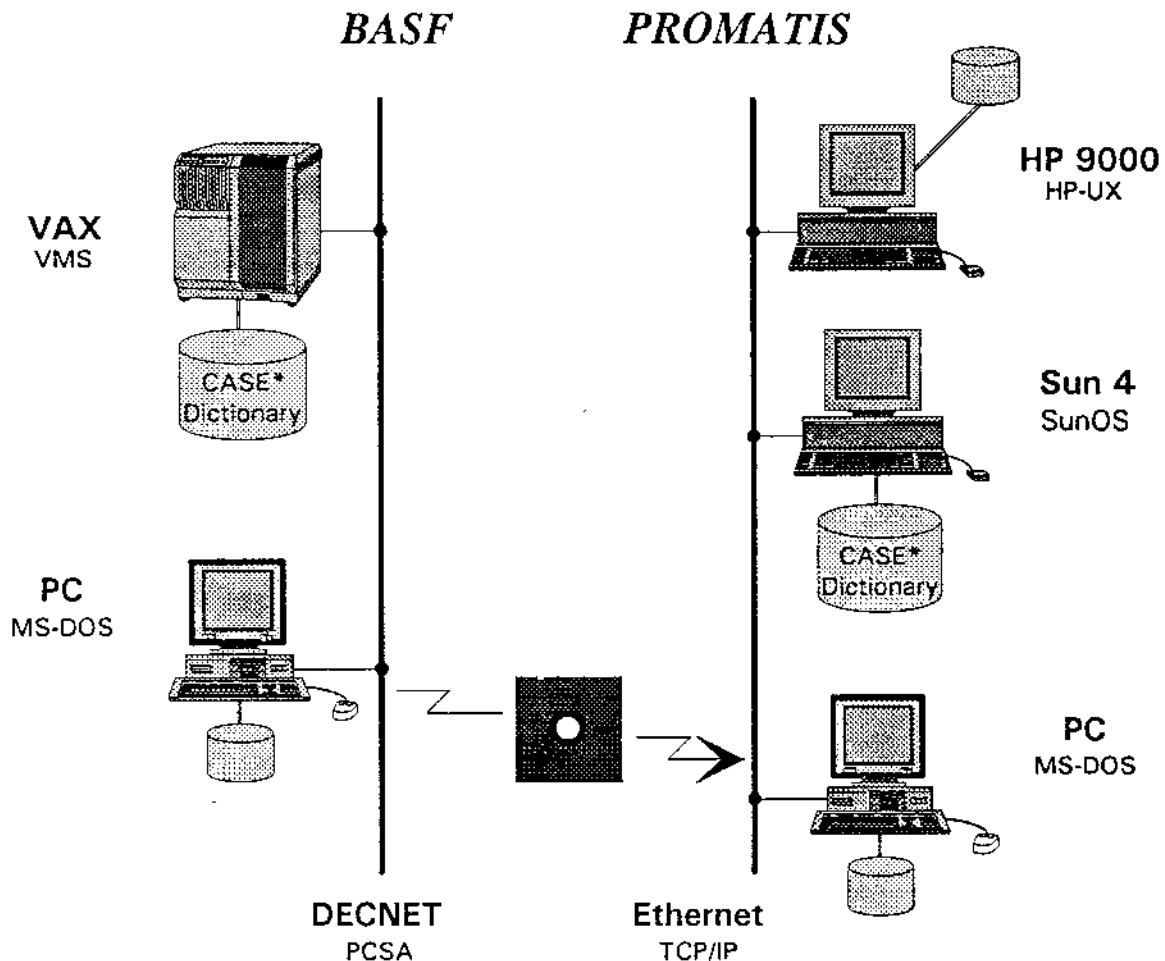


Abbildung 3: Konfiguration bei BASF und PROMATIS

In der Planungs- und Analysephase wurden die CASE*-Produkte bei der BASF auf einer VAX unter VMS genutzt. Vor Beginn der Entwurfsphase wurden deshalb die für das Projekt relevanten Applikationen des CASE*Dictionary extrahiert und die entsprechende Exportdatei über PCSA auf einen MS-DOS PC transferiert. Per Diskette wurde die Exportdatei auf einem MS-DOS PC im Hause PROMATIS eingespielt und über TCP/IP auf eine Sun 4 transferiert. Dort wurden die Applikationen in ein CASE*Dictionary importiert. Die weiteren Arbeiten wurden bei der PROMATIS mit den ORACLE*CASE- und SQL-Produkten auf Sun 4, HP 9000 und MS-DOS PCs durchgeführt.

Bei der Auslieferung von Teilkomponenten wurden jeweils aktualisierte Applikationen des PROMATIS-Dictionary sowie die fertigen Programmodule in der beschriebenen Weise auf die VAX der BASF zurücktransferiert. Dies bedeutet, daß der Auftraggeber nicht nur die entwickelte Software, sondern darüber hinaus die zugehörige Systemspezifikation jederzeit aktuell verfügbar hatte. In dieser Weise war auch eine laufende Qualitätskontrolle durch den Auftraggeber möglich.

Datenbank- und Systementwurf

Aufgrund des Anforderungsmodells wurde zunächst ein Datenbankentwurf durchgeführt. Hierbei mußte vor allem die Integration mit dem CASE*Dictionary berücksichtigt werden. Aufgrund der Komplexität und des Umfangs der Dictionary-Erweiterung, aber auch aus Performance-Gründen, wurde nicht auf die in CASE* 5 verfügbare User Extensibility zurückgegriffen. Die Erweiterung umfaßt deshalb separate Tabellen zur Verwaltung der neuen Repository-Elemente und zugehöriger Beziehungen untereinander, aber auch von Beziehungen zu den Elementen des CASE*Dictionary.

Beim Systementwurf wurden die Funktionshierarchie und die entsprechenden Verwendungsmatrizen auf ein Modul-Netzwerk abgebildet. Dieses wurde zur Erreichung einer einheitlichen Bedienung des DUDD an die CASE*Dictionary-Menüstruktur angepaßt.

Nach Abschluß des Entwurfs wurde dieser in geschlossener Form, sowohl als Papierdokument als auch online im CASE*Dictionary, bei DU präsentiert und von Seite der BASF einer Qualitätskontrolle, insbesondere unter Berücksichtigung der definierten Ziele, unterzogen. Es wurde dabei festgestellt, daß eine sehr gute Abbildung des Fachkonzeptes erreicht worden war.

Feinentwurf und Implementation

Während der Datenbank- und Systementwurf von zwei Mitarbeitern der PROMATIS durchgeführt wurde, mußten für die Implementation bis zu fünf Mitarbeiter eingesetzt werden. Jeder Mitarbeiter war für den Feinentwurf des jeweils zu implementierenden Moduls selbst verantwortlich, d. h. er spezifizierte die entsprechenden detaillierten Table- und Column-Verwendungen im CASE*Dictionary selbst.

Zur Erreichung einer homogenen Benutzerschnittstelle und eines einheitlichen Programmierstils wurden zunächst Templates erstellt, d. h. Vorlagen für die spätere Anwendungsgenerierung. Diese Templates wurden entsprechend den im Pflichtenheft dokumentierten Anforderungen realisiert und mit dem Auftraggeber abgestimmt. Die Templates wurden durchgängig für die Anwendungsgenerierung mit den CASE*Generatoren verwendet.

Da die Datenbank und die Module im CASE*Dictionary sehr detailliert beschrieben waren, lieferten die CASE*Generatoren sehr gute Ergebnisse. Insbesondere die Möglichkeit der evolutionären Programmentwicklung durch die Regenerate-Funktionalität sorgte dafür, daß mit jedem Evolutionszyklus nicht nur das Programm, sondern auch die zugehörige Spezifikation fortentwickelt wurde.

Durch die hohe Qualität des zugrundeliegenden Anforderungsmodells in punkto Konsistenz und Vollständigkeit waren in der Entwurfs- und Implementationsphase nur geringe Nachbesserungen und Abstimmungen erforderlich, so daß die betriebsbereite Übergabe zum geplanten Termin erfolgen konnte.

5 Einsatz und Perspektiven

Der DUDD wurde zum Jahreswechsel 1992/93 als System fertiggestellt, der praktische Einsatz im Bereich DU der BASF AG beginnt in diesem Jahr. Wie bei jeder Einführung einer fortschrittlichen Technologie gibt es Gefahren und Risiken, sind Widerstände zu überwinden und Vorsichtsmaßnahmen zu treffen.

- Die strukturierte Vorgehensweise und Vorgehensmodelle sind noch kein Allgemeingut bei Entwicklern, Projektleitern und DV-Verantwortlichen. Abhilfe kann hier geschaffen werden durch Werbung für die neuen Methoden, durch vergleichende Kosten-Nutzen-Analyse damit entwickelter neuer gegenüber alter Systeme und durch kleine erfolgreiche Neuprojekte als überzeugende Beispiele.
- Die dem DUDD zugrundeliegende Datenmodellierungsmethode (ER) ist ebenfalls noch nicht weit genug bei den Betroffenen verbreitet. Hier ist Schulung erforderlich, die angesichts der überzeugenden inneren Logik der ER-Methode meist sehr schnelle und nachhaltige Erfolge erzielt und dazu führt, daß viele diese Methode harmonisch in ihre tägliche praktische Arbeit integrieren können.
- Da die Erstellung der standardisierten Dokumentation vordergründig mehr Arbeit erfordert, ist zunächst mit mangelnder Akzeptanz eines Repositories beim Entwickler als Hauptbetroffenen zu rechnen. Diesem Widerstand kann zum Beispiel durch rigorose Analyse der jetzigen Arbeitsweise mit Aufzeigen der heutigen zeitlichen Verlustquellen (z.B. Suche nach Informationen) und Hinweis auf Reduzierung der ungeliebten Wartungstätigkeiten begegnet werden.
- Ein fehlendes Unternehmensdatenmodell (UDM) kann verhindern, daß erfolgreich dokumentierte Projekte eine anwendungsübergreifende Nutzwirkung erzielen können. Abhilfe schafft hier die Einrichtung einer Stelle für das Datenmanagement. Eine der Hauptaufgaben dieser Stelle muß es sein, ein UDM mit Unterstützung des Repositories zu erstellen. Hierzu geben wir unten noch eine pragmatische Vorgehensweise an.

Es muß jedoch klar sein, daß die Einführung einer repository-gestützten Softwareentwicklung eine „Kulturrevolution in der Arbeitsweise der Softwareentwickler“ [Mis91] bedeutet. Die Erstellung von Software ist bisher eher Handwerk oder Kunst als Technik oder Wissenschaft. Mit dem Repository schafft man die Voraussetzungen, ein ingenieurmäßiges Vorgehen auf Software zu übertragen. Dies umfaßt ein Konzept, wie global vorzugehen ist, Methoden, die im Rahmen dieses Konzeptes angewendet werden, Techniken, die die Methoden ausführen, und schließlich Werkzeuge, die die Techniken unterstützen. Es hat sich gezeigt, daß jedes isolierte Vorgehen auf einer dieser vier Ebenen erfolglos bleibt, sondern nur integriert die gewünschten Nutzeffekte erzielt werden können.

Welchen Nutzen bringt der Data Dictionary?

Sind die genannten Hindernisse erst einmal überwunden und der Data Dictionary zum breiten Einsatz gelangt, kann man vielfältige und weitreichende Nutzeffekte erwarten. Hierbei ergibt sich der Nutzen des Data Dictionary aus strategischen Vorteilen für das Unternehmen und aus Kostenreduktionen in der EDV.

Zu den strategischen Vorteilen zählen:

- Schnellere Reaktionen auf äußere Anforderungen, gegeben durch Markt oder Gesetzgebung, werden möglich
- Arbeitsabläufe können besser an geänderte Anforderungen oder organisatorische Veränderungen angepaßt werden
- Probleme und Risiken, die auf fehlerhafte Daten zurückgehen, werden durch die Verbesserung der Datenqualität verringert
- EDV-Projekte werden zeitlich und finanziell besser kalkulierbar, da drastische Überziehungen von Kosten- und Zeitplänen, wie sie bisher oft unausweichlich schienen, vermieden werden
- Die Abhängigkeit von vorgegebenen Softwarestrukturen und von einzelnen mit Entwicklung oder Betreuung beauftragten Personen wird verringert

Kostenreduktionen in der EDV ergeben sich bei Einsatz eines Data Dictionary durch

- Straffe und erfolgreiche Durchführung von Neuprojekten, da die Wiederholung von Arbeitsschritten aufgrund von Informationsmängeln entfällt
- Deutlich verbesserte Korrelation zwischen Projektdefinition und Entwicklungsergebnis
- Effektivere Wartung und Betreuung von Software, da durch die standardisierte und stets verfügbare Dokumentation im Dictionary das aufwendige Suchen, Lesen und Verstehen, welches bis zu 50% der Wartungsaufwendungen ausmachen kann, reduziert wird
- Wiederverwendbarkeit von mit dem Dictionary dokumentierten Daten und Programmen, und dadurch Reduktion der Bestände an unkontrolliert redundant gespeicherten Daten mit allen dadurch entstehenden Problemen
- Erleichterung des Datenaustauschs zwischen verschiedenen EDV-Systemen

Die Zeitskala, mit der die Nutzeffekte des Data Dictionary wirksam werden, erstreckt sich von kurz- bis langfristig. Der kurzfristige Nutzen stellt sich für die Abwicklung von Softwareprojekten ein, die mit Unterstützung des Data Dictionary durchgeführt werden. Durch die Formalisierung der in den frühen Phasen (Planung des Vorhabens, Analyse des fachlichen Gegenstandes) zu betrachtenden Informationseinheiten und die sofort verfügbare standardisierte Dokumentation dieser Informationen wird eine straffere und weniger fehlerbehaftete Durchführung dieser Phasen möglich, die ohne Wiederholung von Arbeitsschritten aufgrund „vergessener Fragen“ auskommt.

Mittelfristig wirksam werden

- Wiederverwendbarkeit von Anwendungen und Modulen
- Deutliche Reduktion von isolierten Neuentwicklungen
- Kontrolle und Reduktion redundanter Daten
- Erleichterung und Beschleunigung der Wartung

Die strategischen Vorteile für das Unternehmen werden langfristig erzielt, wenn eine zunehmende Zahl von EDV-Systemen im Data Dictionary dokumentiert wurde und ihre Daten damit integrierbar geworden sind. Damit werden auch anspruchsvolle übergeordnete Konzepte wie Management-Informationssysteme, CIM, Office Automation und verteilte Informationssysteme realisierbar.

An dieser Stelle soll kurz ein pragmatischer Pfad von der heutigen Situation hin zu diesen integrierten Unternehmensdaten skizziert (nach [Mis91]) und die Rolle des Data Dictionary in diesem Prozeß verdeutlicht werden. Ausgangspunkt ist der heutige Zustand mit Anwendungen, die dedizierte Anwendungsdaten besitzen. Im ersten Schritt werden isolierte Datenmodelle für diese dedizierten Anwendungsdaten erstellt. Dies ist auch mit handelsüblichen CASE-Tools zu leisten, der DD kann aber bereits hier eingesetzt werden. Im zweiten Schritt werden integrierte Datenmodelle für dedizierte Anwendungsdaten erstellt, das UDM entsteht. Hier werden Werkzeuge für Definitionen und Abgleich der Teildatenmodelle benötigt. Der DUDD ist ein solches Kontrollinstrument für die Verwendung von Standards und die Konsistenzerhaltung des Datenmodells und zeigt hier seine Stärken gegenüber bisher verfügbaren Werkzeugen. Im dritten Schritt werden integrierte Datenmodelle für integrierte Unternehmensdaten erstellt, die Reorganisation dieser Daten kann erfolgreich abgeschlossen werden. Der Data Dictionary, dann zu einem aktiven integrierten Dictionary ausgebaut, wacht über die effektive Verwendung dieser wertvollen Unternehmensressource.

Die in zunehmendem Maße gefragte anwendungsübergreifende und unternehmensweite Nutzung von Daten und Software wird nur dann möglich werden, wenn diese konsequent mit Unterstützung durch ein Repository aufgebaut werden. Der Grad der Nutzbarkeit von Information als Ressource eines Unternehmens wird auf die Dauer ganz wesentlich von der Adaption und Durchsetzung dieses Prinzips abhängen.

Literatur

- [Bar90a] R. Barker: *CASE*Method - Entity Relationship Modelling*. Addison-Wesley Publ. Comp, Wokingham, England, 1990.
- [Bar90b] R. Barker: *CASE*Method - Tasks and Deliverables*. Addison-Wesley Publ. Comp, Wokingham, England, 1990.
- [Mis91] H. Mistelbauer: *Datenmodellierung und Datenmodellverdichtung*. Vortrag im Symposium „Datenmodellierung '91“ (DECollege), Ludwigsburg 1991.
- [PRO92] PROMATIS Informatik: *Information Engineering mit ORACLE*CASE*. Seminardokumentation, Karlsbad, 1992.
- [PRO93] PROMATIS Informatik: *INCOME - Anforderungsmodellierung und Entwurf mit Petri-Netzen*. Karlsbad, 1993.
- [ScN92] F. Schönthaler und T. Németh: *Software-Entwicklungswerkzeuge - Methodische Grundlagen, 2. Auflage*. B.G. Teubner, Stuttgart, 1992.
- [SMM91] F. Schönthaler, R. Mann und M. Mohl: *INCOME - Modellierung von Systemverhalten mit Petri-Netzen*. *DOAG news, Informationen der Deutschen ORACLE Anwender-Gruppe 6* (Dez. 1991), 22-39.

Dr. Andreas Schuh
BASF AG
Fachinformatik Umweltschutz
DUU/I - Z 75
W-67056 Ludwigshafen
Tel. 0621/60-49896, Fax 0621/60-49077

Dr. Frank Schönthaler
PROMATIS Informatik GmbH & Co. KG
Descostr. 10
W-76307 Karlsbad
Tel. 07248/9145-0, Fax 07248/9145-19