NR. 5

Heinz Lothar Grob
Sabine Seufert

Vorgehensmodelle bei der Entwicklung von CAL-Software

INSTITUT FÜR WIRTSCHAFTSINFORMATIK DER WESTFÄLISCHEN WILHELMS-UNIVERSITÄT MÜNSTER
GREVENER STR. 91, 48159 MÜNSTER, TEL. (0251) 83-9752, FAX. (0251) 83-9754
EMAIL GROB@WI.UNI-MUENSTER.DE

Januar 1996

INHALT

1 Vorgehensmodelle der Software-Entwicklung

Mit dem Begriff **Vorgehensmodell** wird ein ablauforganisatorisches Konzept der Software-Entwicklung bezeichnet, bei dem ein komplexer Prozeß in klar definierte, überschaubare Einheiten gegliedert wird. Durch Vorgehensmodelle wird Handlungswissen für die Erstellung von Software in Form von Prinzipien, Methoden und Werkzeugen zur Verfügung gestellt.[1] Aus diesem allgemeingültigen Wissen soll ein Software-Entwicklungsprozeß für *konkrete Anwendungên* abgeleitet werden.[2]

Die einzelnen Entwicklungsaktivitäten werden häufig zu Phasen zusammengefaßt, um die Ablaufstruktur zu fixieren.[3] Ein Vorgehensmodell definiert neben der Struktur aber auch die Inhalte der Phasen sowie die Ergebnisse jeder Phase. Die verschiedenen Paradigmen der Software-Entwicklung beruhen maßgeblich auf der Unterscheidung der zugrundeliegenden Ablauf*strategie*. Dabei lassen sich das lineare, iterative, prototypingorientierte Life Cycle und das evolutionäre Modell voneinander abgrenzen:

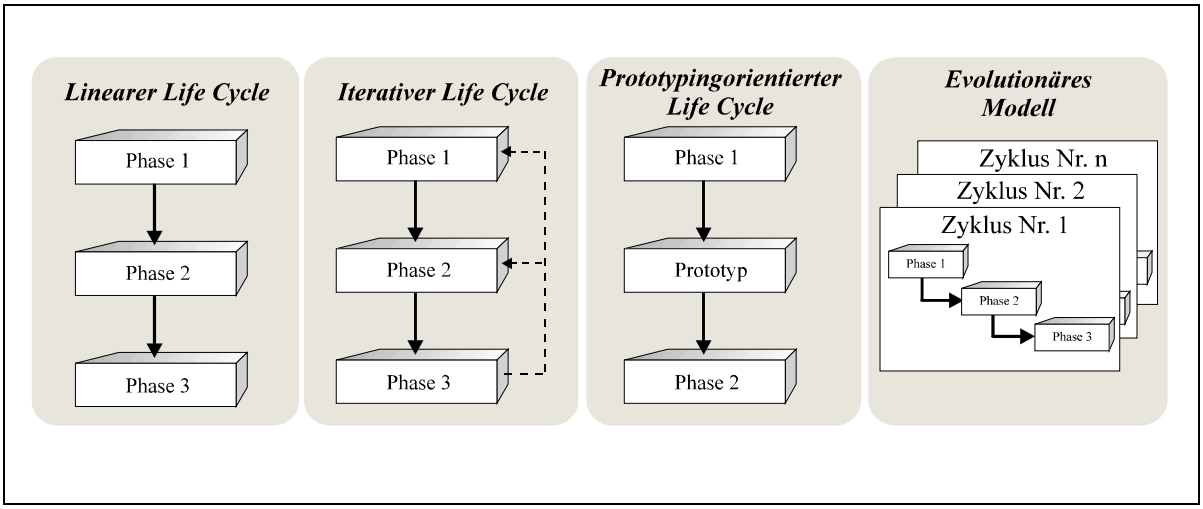

Abb. 1: Paradigmen der Software-Entwicklung

Im Rahmen des **linearen Life Cycles** werden die Entwicklungsschritte in einer sequentiellen Abfolge bearbeitet.[4] Erst wenn die Ergebnisse einer Phase vollständig und fehlerfrei vorliegen, darf mit der nächsten Phase begonnen werden, die wiederum notwendige Input-Informationen für die nachfolgende Entwicklungsphase liefert. Das Endprodukt wird von

---

1 Vgl. Chroust, G. (1992), S. 50.

2 Vgl. derselbe, S. 42.

3 Bezüglich des Ablaufs der Software-Entwicklung stellen die Begriffe Phasenschema, Phasenkonzept oder Ablaufmodell Synonyme für Vorgehensmodelle dar, vgl. Biethahn, J., Muksch, H., Ruf, W. (1990), S. 118.

4 Synonym wird der Begriff *Wasserfallmodell* verwendet, vgl. Hesse, W., Marbeth, G., Frölich, R. (1992), S. 30.

Phase zu Phase zunehmend i. S. eines „Top-down-Refinement“ konkretisiert, wobei Rückverzweigungen zu vorhergehenden Phasen aufgrund aufgetretener Änderungsbedürfnisse nicht vorgesehen sind. Die Implementierung findet in einer relativ späten Phase statt. Zwar sorgt der sequentielle Ablauf für ein klar strukturiertes Vorgehen und diszipliniert die Projektorganisation, jedoch werden Verbesserungen, die bereits abgeschlossene Phasen betreffen, prinzipiell unterdrückt.

Der iterative **Life Cycle** geht ebenfalls nach einem Top-down-Ansatz vor, bei dem schrittweise von allgemeinen Spezifikationsphasen zu immer konkreteren Realisierungsaufgaben übergegangen wird.[1] Zusätzlich wird dieses Modell um Rückverzweigungen zu vorhergehenden Phasen ergänzt, falls bei der Überprüfung der Zwischenergebnisse Fehler aufgetreten sind.[2] Die sich anschließenden Phasen werden nach der Korrektur der fehlerverursachenden Phase erneut durchlaufen. Auch bei diesem Life-Cycle-Konzept findet die Implementierung und damit die Evaluierung der Benutzeranforderungen relativ spät statt.

Um Evaluierungsrisiken entgegenzuwirken, ist der Einsatz von Prototyping entwickelt worden. Der **prototypingorientierte Life Cycle** verknüpft Prototyping-Konzepte mit denen des Software Life Cycles. Ein Prototyp kennzeichnet eine vorläufige und unvollständige Testversion der fertigen Applikation. Mit Hilfe des Prototypen können bereits frühzeitig wesentliche Merkmale des zu entwickelnden Systems untersucht werden. Zudem sind die Phasen dieses Konzepts nicht mehr streng voneinander getrennt, sondern können teilweise überlappend ablaufen. Für die Entwicklung derartiger Testversionen lassen sich die folgenden Arten des Prototyping unterscheiden:[3]

- *Exploratives Prototyping* unterstützt die Kommunikation zwischen Anwendern und Entwicklern, indem anhand eines Prototypen die Benutzeranforderungen (z. B. für den Entwurf von Benutzeroberflächen) ermittelt werden.

- Das *experimentelle Prototyping* versucht unbekannte Eigenschaften der Systemarchitektur zu erproben und dient zur Festlegung des Systementwurfs.

- Beim *evolutionären Prototyping* entsteht das Endsystem durch die kontinuierliche Weiterentwicklung und Verbesserung der anfänglichen Testversion.

Die letzte Form des Prototyping steht im Widerspruch zu den Prinzipien der Life-Cycle-Modelle und geht in ein **evolutionäres Modell** über, das auch als eigenständiges Konzept angesehen wird. Der Software-Herstellungsprozeß wird hierbei in mehreren Entwicklungszyklen geplant, die jeweils gleiche Phasen durchlaufen. Zwar liegt diesem Modell ebenfalls der

---

1 Vgl. Bodendorf, F. (1990), S. 76.

2 Dieses Modell wird auch als Wasserfallmodell mit Rückkopplung bezeichnet, vgl. Chroust, G. (1992), S. 162.

3 Vgl. Floyd, C. (1984), S. 1-18.

Top-down-Ansatz zugrunde, jedoch geht die zunehmende Konkretisierung nicht von Phasen, sondern von Entwicklungszyklen aus. Die Anforderungen des Endsystems werden in jedem neuen Zyklus nach und nach ergänzt.

Im folgenden wird untersucht, welche der oben kurz dargestellten Vorgehensmodelle für die Entwicklung von CAL-Software besonders geeignet sind. Dabei wird von einer Aufgabenkomplexität ausgegangen, die für die Entwicklung von CAL-Software im universitären Bereich typisch ist.[1]

2 Vorgehensmodelle der CAL-Entwicklung

2.1 Einteilung bestehender Vorgehensmodelle

Die Entwicklung von Lernprogrammen weist im wesentlichen *zwei Besonderheiten* auf. Derartige Projekte besitzen einen interdisziplinären Charakter, da für die Konzeption und Realisierung von CAL-Systemen mit Wirtschaftsinformatikern oder Medienspezialisten, Anwendern sowie Pädagogen bzw. Psychologen ganz unterschiedlich ausgebildete Fachexperten notwendig sind.[2] Die Heterogenität des Entwicklungsteams resultiert daraus, daß bei der Gestaltung der Software nicht nur die fachliche, sondern auch die lernpsychologische Situation des Benutzers zu berücksichtigen ist.

Aufgrund dieser Merkmale wurden spezifische Vorgehensmodelle für Lernprogramme entworfen. Zunächst wurden CAL-Vorgehensmodelle entwickelt, die das Handlungswissen, das für die Gestaltung von Software generell relevant ist, für die Erstellung von Lernprogrammen modifizierten. Da sich das Spektrum an Lernsystemen[3] immer stärker ausbreitete, entstanden Vorgehensmodelle, die speziell für einzelne CAL-Varianten und deren Besonderheiten ausgearbeitet wurden. Nachfolgend werden daher *allgemeine* und auf *spezifische* CAL-Varianten zugeschnittene Vorgehensmodelle unterschieden.

2.2 Allgemeine CAL-Vorgehensmodelle

Allgemeine CAL-Vorgehensmodelle, die auf den Paradigmen der Software-Entwicklung basieren, existieren in der Literatur für lineare, iterative, prototypingorientierte Life-Cycle- und evolutionäre Konzepte. Die nachfolgende Übersicht faßt Literaturbeispiele und Inhalte hinsichtlich des Projektmanagements, der CAL-spezifischen Entwicklungstätigkeiten und Methoden zusammen:

---

1 Vgl. Grob, H. L., Grießhaber, W. (1995).

2 Vgl. Kretschmer, M. (1993), S. 1.

3 Das CAL-Spektrum reicht von einfachen Multiple-Choice-Programmen über Simulationsmodelle bis hin zu Planspielen, vgl. Grob, H. L. (1994), S. 80.

Merkmale	***Linearer CAL-Life-Cycle***	***Iterativer CAL-Life-Cycle***	***Prototypingorientierter CAL-Life-Cycle***	***Evolutionäres CAL-Modell***
Literaturbeispiele	• Steppi, H. (1989) • Gabele, E., Zürn, B. (1993) • Janotta, H. (1990)	• Bodendorf, F. (1990)	• Götz, K., Häfner, P. (1991)	• Kretschmer, M. (1993)
Projektmanagement	• Machbarkeitsstudie, Ressourcen-, Zeit-, Kosten- Personal- und Terminplanung • Rollen der Projektbeteiligten sind definiert • Phasenergebnisse als Meilensteine	• Lösungskonzept legt die Rahmenbedingungen des Projekts fest • Rollen der Projektbeteiligten sind (implizit) definiert • Phasenergebnisse als Meilensteine, deren Überprüfung Rückverzweigungen bedingen können	• Eingeschränkte Planungsaufgaben • Endbenutzer als Projektbeteiligte werden früh einbezogen, Rollen der Projektbeteiligten sind definiert • Zwischenergebnisse sind in Form von Prüfpunkten definiert	• Zyklusplanung, Projektrevision erfolgt jeweils zu Beginn eines Zyklus • Rollen der Projektbeteiligten nicht erwähnt, frühe Benutzerpartizipation • Zwischenergebnisse sind in Form von Zyklusergebnissen definiert
CAL-spezifische Entwicklungstätigkeiten	• Unterscheidung in Konzeptions- und Realisierungsphasen • Die Konzeptionsphase beinhaltet überwiegend didaktische Überlegungen • Nahtstelle zur technischen Realisierung bildet die Drehbucherstellung als Implementierungsvorgabe	• Zielanalyse, Lösungskonzept und Pädagogisches Design beinhalten überwiegend didaktische Überlegungen • Nahtstelle zur technischen Realisierung bildet die Drehbucherstellung als Implementierungsvorgabe. Das Drehbuch ist das Ergebnis des pädagogischen Designs	• Analyse, Zielkatalog und Design beinhalten überwiegend didaktische Überlegungen Nahtstelle zur technischen Realisierung bildet die Drehbucherstellung als Implementierungsvorgabe	• Lerntheoretische Analyse mit didaktischen Überlegungen • Unterscheidung in Spezifikations-, Konzeptions- und Realisierungsphasen • Nahtstelle zur technischen Realisierung bildet die Drehbucherstellung als Implementierungsvorgabe
Methoden	• Katalog an Standards, Vordrucke für Feinkonzeption und Drehbuch • umfangreiche Dokumentationen je Phase	• Ablaufplan der Lektionen • explorativer Prototyp (erst relativ spät in der Implementierungsphase)	• Flußdiagramme • Probesequenz, experimenteller Prototyp (bereits in der Anforderungsphase)	• evolutionärer Prototyp

Abb. 2: Allgemeine CAL-Vorgehensmodelle

Die Aufgaben des **Projektmanagements** werden sehr stark vom zugrundeliegenden Entwicklungskonzept beeinflußt. Der *lineare Life Cycle* beinhaltet die umfangreichsten Vorschläge zur Projektplanung und -organisation. In Abhängigkeit von einer Machbarkeitsstudie werden Zeit-, Kosten-, Personal- und Terminplanung detailliert beschrieben. Beim vorliegenden Beispiel des *iterativen Life Cycles* werden die organisatorischen und personellen Rahmenbedingungen in einem Lösungskonzept erarbeitet. Beim *prototypingorientierten Life Cycle* wird die Notwendigkeit einer Projektplanung zwar erwähnt, jedoch nicht auf spezifische Inhalte oder

Unterschiede zu anderen Projekten eingegangen. Der Projektablauf steht stärker als die Projektorganisation im Vordergrund, wobei die Endbenutzer anhand einer Probesequenz sehr schnell einbezogen werden können. Auch beim *evolutionären CAL-Modell* wird möglichst früh ein Prototyp hergestellt. Die Planungsaufgaben konzentrieren sich auf die Anzahl und Abgrenzung der Entwicklungszyklen.

Die **CAL-spezifischen Entwicklungsschritte** sind nicht vom gewählten Entwicklungsparadigma abhängig. Bezüglich der notwendigen Aktivitäten für die Entstehung eines Lernprogrammes herrscht große Übereinstimmung. In jedem Vorgehensmodell werden spezifische Phasen zur Erarbeitung eines didaktischen Konzepts ermittelt. Die Nahtstelle zwischen dem didaktischen Design und der informationstechnologischen Umsetzung bildet in allen Ansätzen das *Drehbuch*, das konkrete Vorgaben für die Implementierung enthält.

Die **Methoden** der Lernprogrammerstellung reichen von Flußdiagrammen, Ablaufplänen der Lektionen bis hin zu Standardkatalogen, um den Entwicklungsprozeß zu unterstützen. Dabei handelt es sich vorwiegend um traditionelle Verfahren, die aus der Software-Entwicklung übertragen wurden.

2.3 Spezifisches CAL-Vorgehensmodell

Die Entwicklung von HyperMedia-Systemen, in denen die Anforderungen des CAL+CAT-Konzepts realisiert werden, erfordert ein spezielles Vorgehensmodell, dessen Eigenschaften im folgenden herauszuarbeiten sind.

In einem HyperMedia-System liegen die Lehrstoffeinheiten nicht linear vor, sondern sind in Form eines Netzwerkes organisiert.[1] Dieses Netzwerk besteht aus Knoten, die die Lehrstoffinhalte einschließen, und aus Querbeziehungen zwischen den Knoten.[2] Bei Hypertext-Systemen bestehen die Knoten aus verschiedenen Textblöcken, die über Querverweise strukturiert sind. Darüber hinaus können bei HyperMedia-Anwendungen mit starren und veränderbaren[3] („kinetischen“) Grafiken, Fotos, Video- und Audiosequenzen, Animationen sowie eigenständige Lernprogramme oder Templates[4] für Berechnungsexperimente integriert werden.[5]

Das HyperMedia-Vorgehensmodell ist in Abb. 3 schematisch dargestellt worden. Das Modell ist aufgrund der Erfahrungen mit HyperMedia-Entwicklungen im Rahmen eines CAL+CAT-Projekts zur Verbesserung der Qualität des universitären Lernens und Lehrens entstanden.[6]

---

1 Vgl. Kuhlen, R. (1990), S. 27 sowie Grob, H. L., Bensberg, F., Bieletzke, S. (1995).

2 Vgl. Gloor, P. A. (1990), S. 3.

3 Vgl. Grob, H. L., v. Brocke, J. (1995).

4 In der folgenden Abbildung als TBK(Tabellenkalkulation)-Templates bezeichnet.

5 Vgl. Grob, H. L., Grießhaber, W. (1995), S. 7.

6 Vgl. Grob, H. L., Griehaber, W. (1995).

Anstelle von Entwicklungsphasen bzw. -zyklen ist das Ablaufmodell durch drei typische *Entwicklungsebenen* gekennzeichnet.[1]

Die erste Entwicklungsebene bezeichnet die Erstellung der Text- und multimedialen Dokumente („*generative* Ebene“). Auf der nachfolgenden Ebene werden die Verknüpfungen zwischen den Texteinheiten erstellt, um aus dem linearen Text den netzwerkartigen Hypertext zu gestalten („*formale* Ebene“). Der Hypertext wird auf der dritten Entwicklungsebene um die multimedialen Objekte zu HyperMedia erweitert („*multimediale* Ebene“).

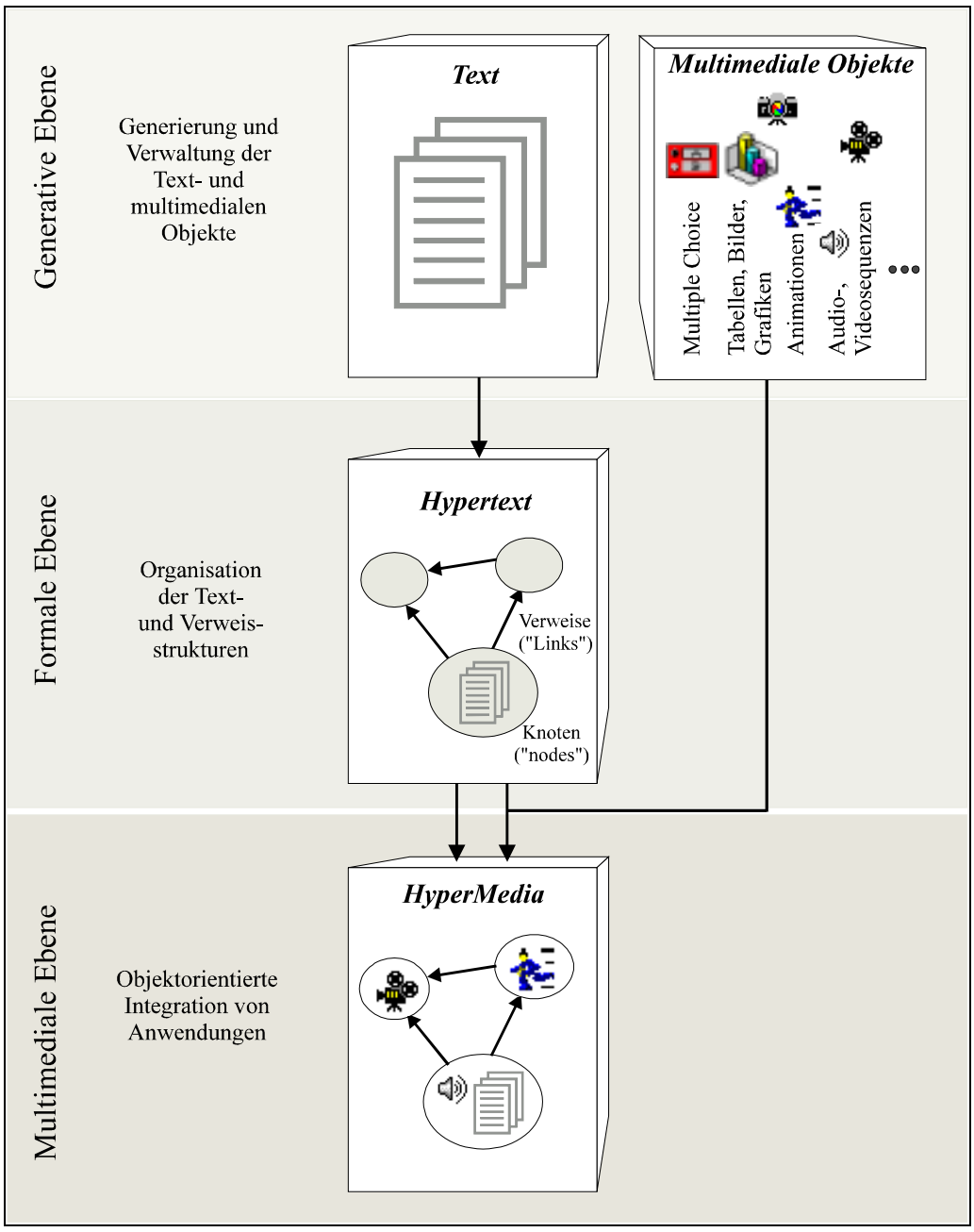

Abb. 3: HyperMedia-Vorgehensmodell

[1] Vgl. Schnupp, P. (1992), S. 130.

Auf der **generativen Ebene** werden die Lehrstoffelemente des HyperMedia-Systems entwickelt. Die Präsentation des Lehrstoffs erstreckt sich auf Texte, Tabellen, Grafiken, Bilder, Animationen, Audio- und Videosequenzen sowie CAL-Software, die Multiple Choice-Systeme, Lernprogramme und Berechnungsexperimente beinhalten.

Der Übergang von Text zu Hypertext wird auf der **formalen Ebene** vollzogen. Die Lehrstoffinhalte werden in übersichtliche Abschnitte, die mit einem Notizzettel oder Lexikoneintrag vergleichbar sind, gegliedert. Diese stellen die Knoten („nodes“) des Hypertextes dar. Für die Definition eines idealtypischen Knotens sind nur schwer allgemeingültige Regeln zu finden. Im Vordergrund sollte stehen, daß ein Knoten für den Benutzer eine plausible Semantik aufzuweisen hat, so daß der Textknoten als logische Texteinheit verstanden werden kann. Ähnlich einem semantischen Netz müssen die Informationseinheiten mit Hilfe von Verweisen („links“) zueinander in Beziehung gebracht werden. Die Bestimmung von Verweisen kann nach verschiedenen Klassifizierungen erfolgen. Häufig werden Verweise nach ihrer Herkunft in strukturbeschreibende, bedeutungsabhängige und benutzerorientierte Arten eingeteilt. Die Verknüpfungsalternativen unterscheiden sich danach, ob sie Beziehungen zwischen oder innerhalb von Textdokumenten herstellen oder vom Benutzer selbst erzeugt werden können.

Der Schritt von der Hypertext- zur HyperMedia-Anwendung wird auf der **multimedialen Ebene** durchgeführt. Die Objekte werden hierbei als selbständige Module in das Netzwerk integriert, so daß sofort nach Aufruf des Knotens die Anwendung startet. Die multimedialen Objekte können darüber hinaus in bestehende Textknoten eingebunden werden, in denen sie durch den Benutzer interaktiv gesteuert werden können. Allerdings können bei den einzelnen Objekten anders geartete Vorgehensmodelle zugrunde gelegt werden.

2.4 Vergleich der Vorgehensmodelle

In den allgemeinen CAL-Vorgehensmodellen werden Konzeptionsphasen, die überwiegend didaktische Überlegungen zum Ziel haben, von Implementierungsphasen unterschieden, die die informationstechnologische Realisierung in den Vordergrund stellen. Der Übergang vom didaktischen Konzept zur technischen Umsetzung wird mit Hilfe eines Drehbuchs bewerkstelligt.

Das Drehbuch - häufig auch als „Storyboard“ bezeichnet - ist die umfassendste, bis in das kleinste Detail gehende Arbeitsanweisung zur Realisierung einer CAL-Software für Multimedia-Objekte.[1] In einem Drehbuch werden die einzelnen Bildschirmseiten in linearer Reihenfolge geplant. Daher entsprechen die Drehbuchseiten den Bildschirmseiten des zu entwickelnden Lernprogrammes. Das Drehbuch sollte so leicht verständlich sein, daß bereits beim Lesen eine Vorstellung vermittelt wird, wie das fertige CAL-Produkt später auf dem Bildschirm aussieht.

---

[1] Vgl. Steppi, H. (1989), S. 162.

Ein Drehbuch kann neben den eigentlichen Lehrstofftexten der Bildschirmseiten Angaben enthalten, die sich auf den Lernweg, die Seitenart, Schlagwörter und Indexkategorien sowie auf Realisierungsanweisungen für die Bildschirmgestaltung und für die Programmierung beziehen.

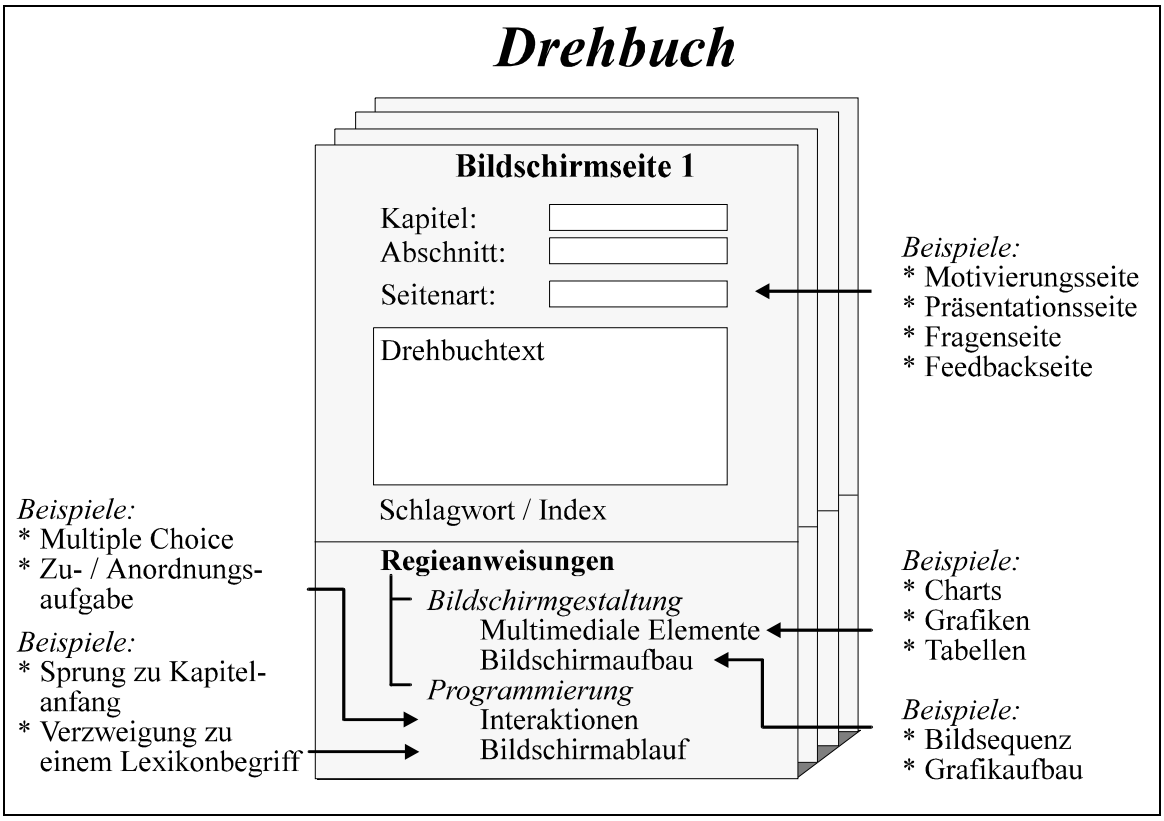

Abb. 4: Inhalte des Drehbuchs

In Lernprogrammen werden häufig die zu vermittelnden Themenbereiche - ähnlich wie in einem Buch - nach Kapiteln und Abschnitten gegliedert. Derartige Gliederungsangaben sollten auch in den Drehbuchseiten angeführt werden. Aus der Aneinanderreihung der fortlaufenden Drehbuch- bzw. Bildschirmseiten kann der lineare *Lernweg* des zu realisierenden CAL-Objektes abgeleitet werden.

Eine Klassifizierung der Drehbuchseiten nach *Seitenarten* kann für einen schnelleren Überblick sorgen. Als Seitenarten können beispielsweise Motivierungs-, Präsentations-, Frage- oder Feedbackseiten unterschieden werden.[1] Dadurch kann auch leichter auf eine sinnvolle und abwechslungsreiche Verteilung von Präsentations- oder Frageseiten in den einzelnen Kapiteln und Abschnitten geachtet werden.

[1] Vgl. Euler, D. (1992), S. 33.

Im Bereich der *Drehbuchtexte* sind die eigentlichen Lehrstoffinhalte aufbereitet, die dem Anwender später auf einer Lernprogrammseite präsentiert werden. Falls die Texte einer Seite nicht komplett, sondern blockweise nacheinander angezeigt werden sollen, müssen die entsprechenden Textblöcke gekennzeichnet werden. Multimediale Elemente, wie z. B. Charts, Grafiken oder Animationen, können mit Positionsangabe im Bildschirmbereich skizziert werden.

Im Drehbuch können darüber hinaus bereits *Schlagwörter* definiert werden, die in einem separaten Schlagwortverzeichnis erklärt werden müssen. Mit der Angabe von *Indexkategorien* wird es dem Lernenden ermöglicht, Lehrstoffinhalte nach einer Indexliste zu bearbeiten, die die Lernseiten nach Stichwörtern strukturiert. Für Indizes ist daher nicht wie bei Schlagwörtern eine zusätzliche Erklärungsseite notwendig. Falls für die Realisierung eines CAL-Produkts Schlagwortverzeichnisse und Indexlisten vorgesehen sind, könnten bereits während der Drehbucherstellung Schlagwörter und Indexkategorien herausgearbeitet werden.

Um eine reibungslose Umsetzung des Drehbuchs in ein Lernprogramm zu ermöglichen, werden häufig zusätzliche Realisierungs- bzw. „Regieanweisungen“ ausgewiesen, die an die Entwicklungsbereiche der Bildschirmgestaltung und der Programmierung gerichtet sein können. Die Realisierungshinweise der einzelnen Drehbuchseiten befinden sich meist am unteren Seitenrand. Falls Drehbuchvordrucke in Papierform verwendet werden, kann auch die Benutzerführung abgebildet sein. Der Drehbuchautor hat dadurch die Möglichkeit, die jeweils aktiven Programmfunktionen innerhalb einer Lernprogrammseite zu kennzeichnen.[1]

Unter *Bildschirmgestaltung* wird die Produktion von Medienereignissen und die Einbindung der multimedialen Daten in die Programmseiten verstanden. Der Betrachtungsgegenstand bei der Bildschirmgestaltung ist die *einzelne* Bildschirmseite, welche Inhalte in welcher Reihenfolge präsentiert werden sollen. Regieanweisungen der Bildschirmgestaltung können daher die Gestaltungsarten (z. B. Text, Chart, Grafik oder Sound) und den Bildschirmaufbau (z. B. aufbauende Grafik, Bildsequenz) der betreffenden Drehbuchseite klassifizieren.

Die Anlage der Bildschirmseiten nach einem meist für das gesamte Lernprogramm einheitlichem Layout und die Verknüpfung *aller* Bildschirmseiten miteinander zu einem ablauffähigen Programm sind dem Entwicklungsbereich *Programmierung* zuzuordnen. Angaben zu Interaktionen und zum Bildschirmablauf können Regieanweisungen darstellen, die die Programmierarbeiten betreffen. Dabei beziehen sich Interaktionen hauptsächlich auf den Fragetyp, wie beispielsweise Multiple-Choice-Aufgaben, Lückentexte oder Freie Eingaben. Falls die Anzahl möglicher Feedbacks auf Antworten einer Frage angegeben wird, ist das Risiko für den Drehbuchautor geringer, Antwortkombinationen zu übersehen. Aufgrund der Angaben des Bildschirmablaufs wird die Verknüpfung zu anderen Bildschirmseiten realisiert, wie beispielswei-

[1] Vgl. Steppi, H. (1989), S. 164.

se der direkte Sprung zu einem bestimmten Paragraphen, zu definierten Schlagwörtern oder zum Kapitelanfang.

Nachfolgende Abbildung beinhaltet ein Beispiel für ein Drehbuch in elektronischer Form. Dargestellt ist der untere Bereich einer Drehbuchseite, in dem sich Realisierungshinweise befinden können. Dabei kann der Drehbuchautor durch vorgegebene Auswahlmöglichkeiten untersützt werden, um beispielsweise Gestaltungs- oder Interaktionsarten zu klassifizieren.

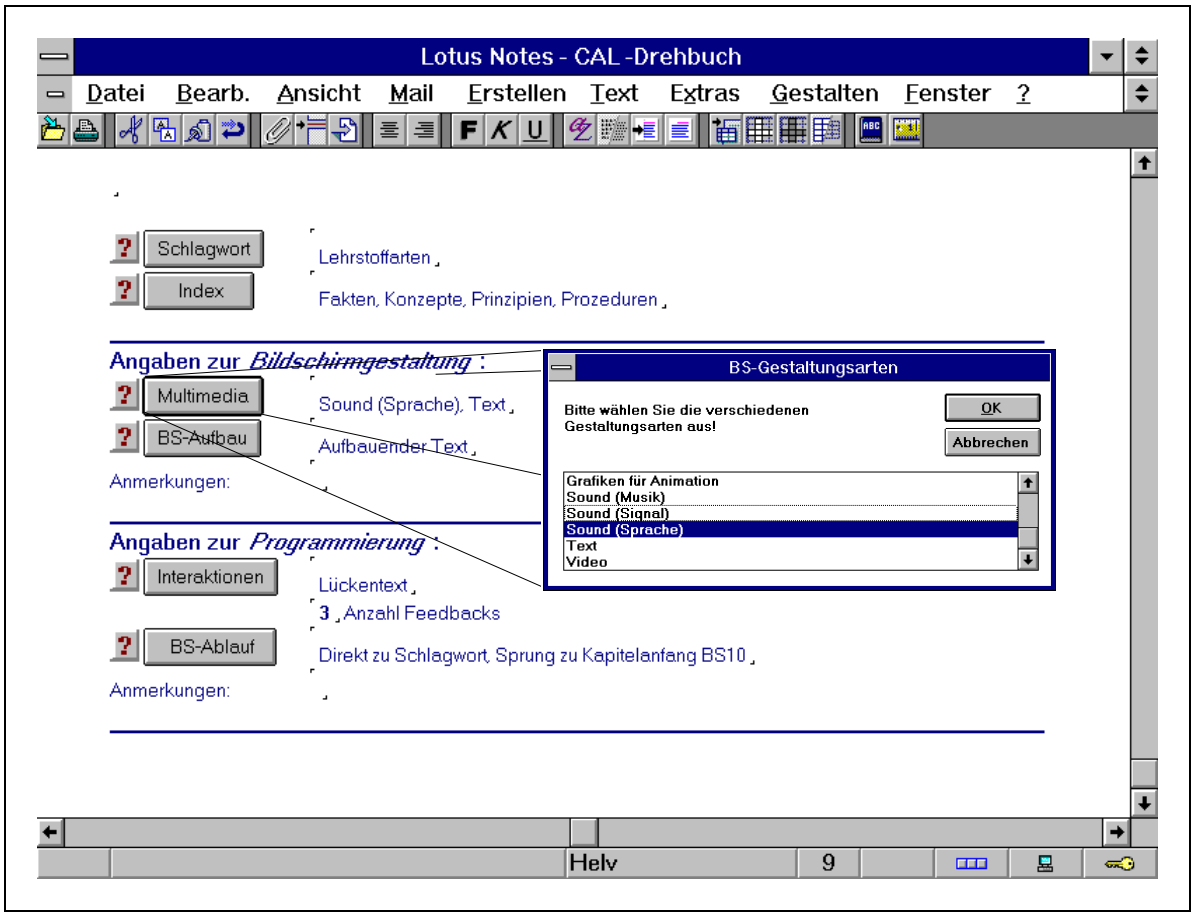

Abb. 5: Beispiel eines Drehbuchs in elektronischer Form

Die bei der Entwicklung der HyperMedia-Software feststellbare Vorgehensweise ist mit dem Top-down-Ansatz allgemeiner Vorgehensmodelle unvereinbar. Bei diesen Klassen von CAL-Programmen wird ein pragmatisch orientierter *Buttom-up-Ansatz* verfolgt, bei dem der permanenten Koordination eine besondere Bedeutung zukommt. Diese kann – über die systematisch durchzuführenden Arbeitssitzungen hinaus – durch ein Repository sinnvoll unterstützt werden.

3 Erweiterung der Vorgehensmodelle um ein Repository

Vorgehensmodelle beinhalten idealtypische Handlungsempfehlungen für die Ablauforganisation von Softwareprojekten. Auf die Lernprogrammerstellung zugeschnittene Vorgehensmodelle bedingen eine spezifische Entwicklungsumgebung (Repository), die als zentrale Ablage für projektrelevante Daten dient und alle Phasen der Softwareherstellung methodisch unterstützt.[1]

Die Anforderungen an ein CAL-Repository ergeben sich aufgrund der charakteristischen Besonderheiten bei der Produktion von CAL-Software. Die nachfolgende Abbildung stellt die typischen Merkmale der CAL-Entwicklung den sich daraus ergebenden Anforderungen an ein CAL-Repository gegenüber.

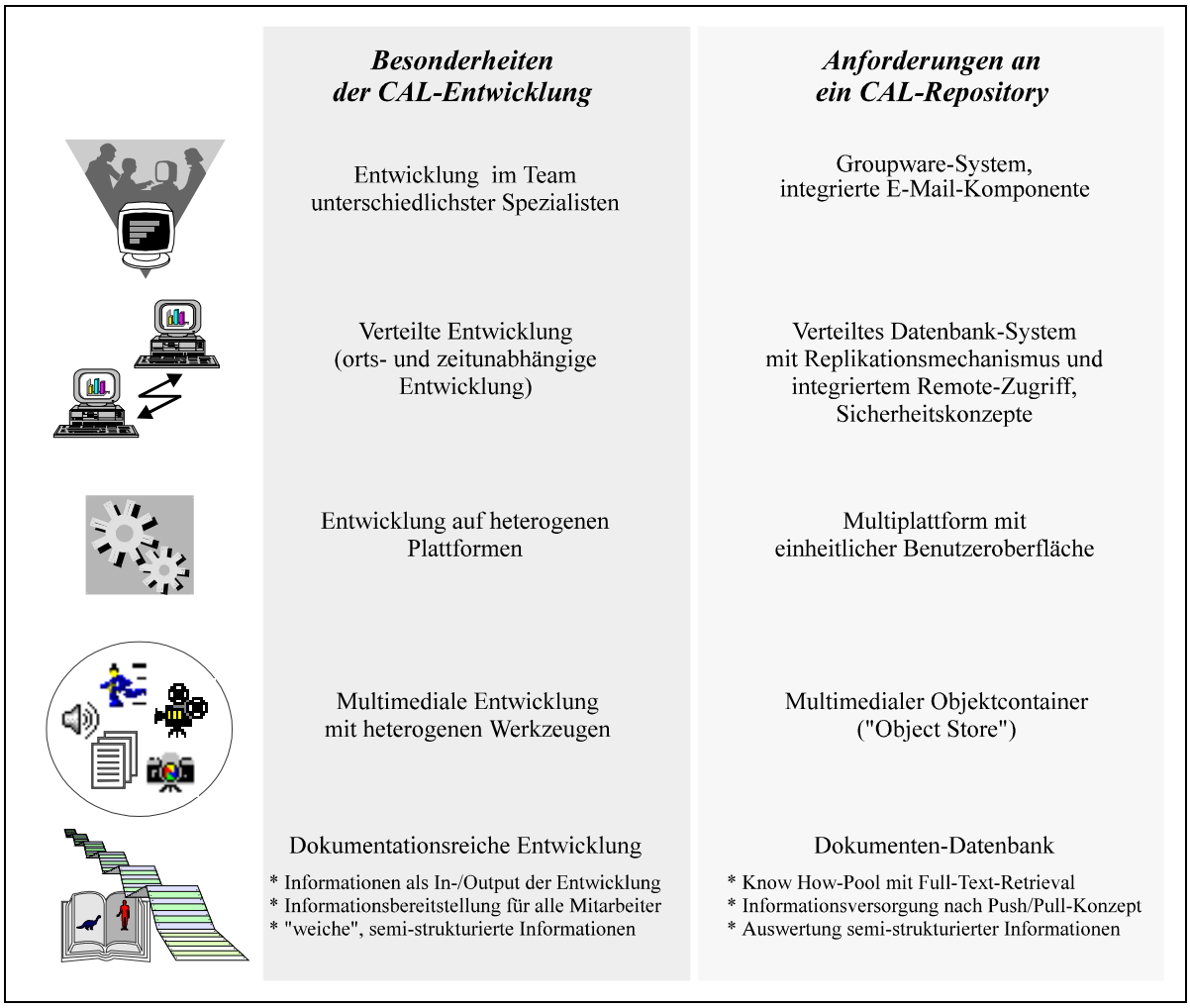

Abb. 6: Anforderungen an ein CAL-Repository

[1] Vgl. Habermann, H.-J., Leymann, F. (1993), S. 15.

CAL-Produkte werden üblicherweise im Team erstellt, das wegen der Interdisziplinarität heterogen besetzt ist. Daher nehmen Aufgaben des Projektmanagements zur Koordination der *Gruppenarbeit* einen besonders hohen Stellenwert ein. Kooperative Prozesse können mit Hilfe von *Groupware*-Systemen unterstützt werden. Ein CAL-Repository sollte daher Groupware-Funktionalitäten beinhalten, die über eine integrierte Mail-Komponente für die Kommunikation der Projektmitarbeiter hinausgehen.

Die Herstellung von CAL-Software kann als „*verteilte Entwicklung*“ bezeichnet werden, da die Projektmitglieder häufig zeitlich und örtlich unabhängig voneinander arbeiten. Daraus ergibt sich die Anforderung an ein Repository, eine dezentrale Datenhaltung im Rahmen eines *verteilten Datenbanksystems* zu ermöglichen. Der Datenabgleich und die Sicherung der Datenkonsistenz kann dabei über *Replikationsmechanismen* sichergestellt werden. Für den Kommunikations- und Datenaustausch sollten ausgefeilte *Sicherheitskonzepte* vorliegen.

Lernprogramme werden häufig auf *heterogenen Plattformen* entwickelt, wie z. B. Apple Macintosh-Rechner für die Grafikerstellung oder Windows als Basis für Autorenwerkzeuge. Eine CAL-Entwicklungsumgebung sollte daher *plattformunabhängig* sind.

Wie bereits im Rahmen der generativen Ebene des HyperMedia-Vorgehensmodells beschrieben, werden *multimediale Objekte* mit den unterschiedlichsten Werkzeugen erstellt, so daß verschiedene Dateiformate in einer CAL-Applikation zusammengeführt werden müssen. Ein Repository sollte daher eine systematische Verwaltung der Multimedia-Objekte übernehmen, so daß Änderungen bzw. Aktualisierungen bei CAL-Produkten sowie die Wiederverwendbarkeit der Multimedia-Elemente für andere Projekte erleichtert werden. Dabei sollten nicht nur Metainformationen über ein Objekt, sondern die Objekte selbst, wie Texte, Grafiken, Bilder, Audio- oder Videosequenzen, in einem *multimedialen Objektcontainer* („Object Store“) gehalten werden.

Die Entwicklungsprozesse der Lernprogrammerstellung sind durch umfangreiche *Dokumentationen* geprägt, die sich sowohl auf Input- als auch auf Output-Informationen beziehen. Als Input können Entwicklungsstandards, „Guide Styles“ für Benutzeroberflächen oder Richtlinien für die Datenhaltung vorliegen, die im Rahmen der Entwicklungstätigkeiten von Projektmitarbeitern beachtet werden müssen. Output-Dokumente fassen Entwicklungsergebnisse zusammen und können beispielsweise Index-, Schlagwortverzeichnisse oder Drehbücher beinhalten. Allen Projektmitarbeitern sollten möglichst frühzeitig sämtliche Informationen zur Verfügung gestellt werden, so daß der einzelne einen guten Überblick über das gesamte Projekt und die Einordnung seines Aufgabenbereiches erhält. Dadurch können „Nahtstellen“, wie beispielsweise der Übergang von der generativen zur formalen Ebene, leichter überbrückt und Schnittstellenprobleme im Vorfeld eingeschränkt werden. Darüber hinaus sollte die Möglichkeit bestehen, vorliegende Dokumente auch nach eher „weichen“ Kriterien, wie beispielsweise Medienereignissen, Interaktionsformen oder Indexkategorien auswerten zu können. Dadurch lassen sich auch diejenigen Entwicklungsschritte methodisch unterstützen, die bislang

nicht sehr transparent waren. Dies gilt insbesondere für die Lehrzielanalyse und die Drehbucherstellung.

Mit Hilfe einer *Dokumenten-Datenbank* könnte ein Know-how-Pool geschaffen werden, der alle erforderlichen In- und Output-Informationen bereithält. Die Teamarbeit kann dadurch auch nach dem „Pull-Konzept“ erfolgen, d. h. jeder Mitarbeiter zieht sich eigenverantwortlich die benötigten Informationen aus dem Repository, und zwar zum adäquaten Zeitpunkt. Der Mitarbeiter ist nicht darauf angewiesen, daß er nach dem „Push-Konzept“ (z. B. Versenden von Mails mit Dateianhängen) mit benötigten Informationen versorgt wird. Für eine effiziente Gruppenarbeit wird eine Kombination von Push- und Pull-Konzepten als sinnvoll angesehen.

Im Gegensatz zu relationalen Datenbanken liegen insbesondere die Stärken von Dokumenten-Datenbanken in den Erfassungs- und Auswertungsmöglichkeiten semi-strukturierter Datenfelder. Durch den Aufbau eines Know-how-Pools können sich neue Projektmitarbeiter schneller und leichter einarbeiten. Verläßt ein Mitarbeiter das Projektteam, bleibt sein Erfahrungsschatz weitestgehend erhalten.

Aufgrund der skizzierten Anforderungen an ein CAL-Repository wird die Entwicklungsplattform Lotus Notes als besonders geeignet betrachtet. Bei Lotus Notes handelt es sich um eine Groupware-Lösung, die eine Dokumenten-Datenbank mit dezentraler Datenhaltung beinhaltet.[1] Zentrales Element von Lotus Notes ist die Compound Document Architektur, die die Funktionalitäten eines multimedialen Objektcontainers unterstützt.

[1] Die Groupwaremerkmale von Lotus Notes werden beschrieben bei Nastansky, L. (1991), S. 8-13. Fachexperten sehen in der Groupwarelösung Notes den Schlüssel zu einer neuen DV-Ära, in der es um die Vernetzung aller Hardwareplattformen geht. Vgl. Kneuse, B. (1995), S. 24.

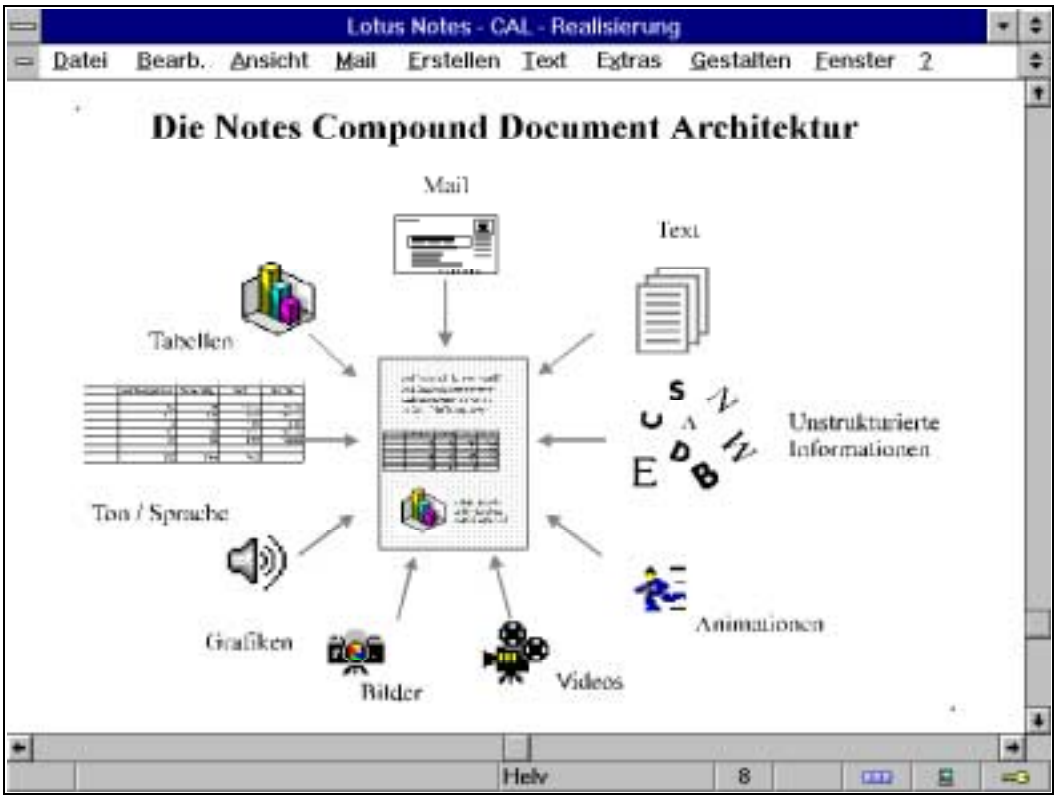

Abb. 7: Notes Compound Document Architektur

Mit dem Begriff „Compound Document“ wird ein Dokument bezeichnet, das sich aus Informationen unterschiedlicher Datenformate, wie beispielsweise Texte, Raster-, Vektorgrafiken, Sprachnotationen oder Videosequenzen zusammensetzt. Mit Hilfe von Hyperlinks können zwischen den einzelnen Dokumenten Verknüpfungen hergestellt werden. Außerdem können Applikationen der CAL-Entwicklungsprozesse integriert und auf deren Daten durch DDE- und OLE-Verbindungen zugegriffen werden.

4 Schlußbemerkung

Abschließend ist festzustellen, daß unabhängig von den zugrundeliegenden Paradigmen und den eingesetzten Werkzeugen (insbesondere dem Repository) bei der Entwicklung von CAL-Produkten ein hohes Maß an Kooperationsbereitschaft und Teamfähigkeit bei den Mitgliedern des CAL-Teams als wesentliche Bedingung anzusehen ist. Praktische Erfahrungen im Rahmen eines Pilotprojekts haben gezeigt, daß allein von der Aufgabe, neue Medien zur Verbesserung der Qualität des universitären Lernens und des Lehrens einzusetzen, eine außerordentlich motivierende Wirkung ausgeht.

Literatur

Biethahn, J., Muksch, H., Ruf, W. (1990), Ganzheitliches Informationsmanagement, Band 1: Grundlagen, München, Wien 1990.

Bodendorf, F. (1990), Computer in der fachlichen und universitären Ausbildung, München, Wien 1990.

Chroust, G. (1992), Modelle der Software-Entwicklung, München, Wien 1992.

Euler, D. (1992), Didaktik des computerunterstützten Lernens, Praktische Gestaltung und theoretische Grundlagen, Band 3 der Reihe „Multimediales Lernen in der Berufsbildung“, Nürnberg 1992.

Floyd, C. (1984), A systematic look at prototyping, in: Approaches at prototyping, Hrsg.: R. Budde, K. Kuhlenkamp, L. Mathiassen, H. Züllighoven, Berlin, Heidelberg, New York, Tokio 1984, S. 1-18.

Gabele, E., Zürn, B. (1993), Entwicklung interaktiver Lernprogramme, Band 1: Grundlagen und Leitfaden, Stuttgart 1993.

Gloor, P. A. (1990), Hypermedia-Anwendungsentwicklung. Eine Einführung mit HyperCard-Beispielen, Stuttgart 1990.

Grob, H. L. (1994), Computer Assisted Learning (CAL) durch Berechnungsexperimente, in: ZfB-Ergänzungsheft 2/94, S. 79-90.

Grob, H. L., Grießhaber, W. (1995), Computergestützte Lehre an der Universität, Arbeitsbericht Nr. 1 der Reihe CAL+CAT, Münster 1995.

Grob, H. L., Bensberg, F., Bieletzke, S. (1995), Hypertext, Arbeitsbericht Nr. 4 der Reihe CAL+CAT, Münster 1995.

Grob, H. L., vom Brock, J. (1996), Kinetische Grafiken, Arbeitsbericht Nr. 6 der Reihe CAL+CAT, Münster 1996.

Habermann, H.-J., Leymann, F. (1993), Repository. Eine Einführung, München, Wien 1993.

Hesse, W., Merbeth, G., Frölich, R. (1992), Software-Entwicklung. Vorgehensmodelle, Projektführung, Produktverwaltung, München, Wien 1992.

Kneuse, B. (1995), Götterdämmerung im PC-Softwaremarkt, Computer Zeitung, Nr. 36 (1995), S. 24.

Kretschmer, M. (1993), Die modellgestützte Entwicklung Intelligenter Tutorieller Systeme, Göttingen 1993.

Kuhlen, R. (1990), Hypertext. Ein nicht-lineares Medium zwischen Buch und Wissensbank, Berlin u. a. 1990.

Nastansky, L. (1991), Gruppenarbeit - Workgroup Computing, in: Office Management, Nr. 6 (1991), S. 8-13.

Schnupp, P. (1992), Hypertext, München, Wien 1993.

Arbeitsberichte der Reihe CAL+CAT

Nr. 1 Grob, H. L., Grießhaber, W., Computergestützte Lehre an der Universität, Arbeitsbericht Nr. 1, Münster 1995.

Nr. 2 Grob, H. L., CAL+CAT, Arbeitsbericht Nr. 2, Münster 1995.

Nr. 3 Grob, H. L., Bensberg, F., Multimedia, Arbeitsbericht Nr. 3, Münster 1995.

Nr. 4 Grob, H. L., Bensberg, F., Bieletzke, S., Hypertext, Arbeitsbericht Nr. 4, Münster 1995.

Nr. 5 Grob, H. L., Seufert, S., Vorgehensmodelle bei der Entwicklung von CAL-Software, Arbeitsbericht Nr. 5, Münster 1996.