Universität Augsburg
Prof. Dr. Hans Ulrich Buhl
Kernkompetenzzentrum
Finanz- & Informationsmanagement
Lehrstuhl für BWL, Wirtschaftsinformatik,
Informations- & Finanzmanagement

Universität Augsburg
University

Diskussionspapier WI-10

Entwicklung finanzwirtschaftlicher Planungssysteme

von

Jochen Schneider

Februar 1996

in: Mayr, M., et al., Hrsg., Beherrschung von Informationssystemen, Tagungsband der Informatik '96, Schriftenreihe der österreichischen Computer Gesellschaft, Klagenfurt, (Österreich), September 1996, Oldenbourg, Wien, 1996, S.197-214

Universität Augsburg, 86135 Augsburg
Besucher: Universitätsstr. 12, 86159 Augsburg
Telefon: +49 821 598-4801 (Fax: -4899)
www.fim-online.eu

Entwicklung finanzwirtschaftlicher Planungssysteme

Jochen Schneider[1]

Die Komplexität und Änderungsdynamik des Umfelds ökonomischer Entscheidungen bedingen hohe Ansprüche an die betriebswirtschaftliche Leistungsfähigkeit und Flexibilität von Softwaresystemen zur Unterstützung finanzwirtschaftlicher Planungsrechnungen. Diesen kann mit anwendungsspezifischen Entwicklungsumgebungen in besonderer Weise entsprochen werden. In dieser Arbeit werden aus spezifischen Anforderungen an Finanzplanungssysteme, Anfordungen an spezielle Entwicklungsumgebungen für Finanzplanungssysteme abgeleitet. Mit dem Integrierten Finanzanalysesystem (IFAS) wird der Prototyp einer Entwicklungsumgebung vorgestellt, welche die Entwickler finanzwirtschaftlicher Planungssysteme weitestgehend von nicht betriebswirtschaftlichen Aspekten, einschließlich Oberflächengestaltung und Datenmodellierung, befreit, ohne daß der Anwender eines finanzwirtschaftlichen Planungssystems auf eine komfortable grafische Oberfläche oder die Funktionalität eines leistungsfähigen relationen Datenbankmanagementsystems verzichten muß. Dies wird durch die Übertragung von Konzepten der doppelten Buchführung auf eine anwendungsspezifische Programmiersprache erreicht.

1. Problemstellung

Die hohe Komplexität und Änderungsdynamik des ökonomischen und rechtlichen Umfelds von Investitionsentscheidungen bleiben nicht ohne Einfluß auf die projektbezogene Finanzplanung. Auf der Basis gegebener Handlungsmöglichkeiten, die z.B. in Form verschiedener Angebote vorliegen können, ist bei der Planung u.a. die Aufgabe zu lösen, die aus der Realisation einer Alternative resultierenden Zahlungen zu antizipieren. Dabei ist den steuerlichen Wirkungen der relevanten Entscheidungsalternativen besondere Beachtung zu schenken, da die Nutzung steuerlicher Gestaltungsspielräume erheblichen Einfluß auf die Vorteilhaftigkeit der betrachteten Alternativen ausüben kann[2]. Selbst unter der stark vereinfachenden Annahme sicherer künftiger Ereignisse ist die Antizi-

1 Universität Augsburg, Lehrstuhl für Betriebswirtschaftslehre mit Schwerpunkt Wirtschaftsinformatik, 86135 Augsburg

2 Vgl. z.B. [35].

pation künftiger Zahlungen komplex. Ein Grund hierfür ist die Vielschichtigkeit des für detaillierte Planungsrechnungen erforderlichen Spezialwissens. Dieses entstammt häufig einer Vielzahl betriebswirtschaftlicher Diszplinen, z.B. Finanzwirtschaft, Rechnungswesen und betriebswirtschaftliche Steuerlehre. Die notwendige Kombination des richtigen Spezialwissens konzentriert sich i.d.R. auf wenige, demzufolge knappe Experten in der Unternehmung oder muß extern als Beratungsleistung eingekauft werden. Eine Alternative besteht in einer geeigneten Systemunterstützung, durch die das knappe Expertenwissen einer großen Zahl betrieblicher Entscheidungsträger dezentral verfügbar gemacht werden kann. Diese "Wissensmultiplikation" kann zu einer deutlichen Senkung der Informationskosten - bei gleichzeitiger Verbesserung der Entscheidungsqualität - führen.

Am Markt existiert eine Vielzahl von Standardsoftwaresystemen zur Unterstützung solcher Planungsrechungen. Dabei handelt es sich meist um spezialisierte Pakete, mit denen jeweils bestimmte Kategorien von Handlungsalternativen, z.B. Finanzierungsformen oder Finanzanlageprodukte, analysiert werden können[3]. Prinzipbedingter Nachteil dieser Systeme ist die mehr oder weniger ausgeprägte Inflexibilität im Hinblick auf Anpassungen und Erweiterungen. Neuartige Handlungsmöglichkeiten oder geänderte rechtliche Vorschriften können i.d.R. erst mit einem neuen Release der Software oder mit dem Erscheinen eines neuen Standardsoftwarepakets berücksichtigt werden. Als möglicher Ausweg verbleibt die Eigenentwicklung mittels geeigneter Entwicklungswerkzeuge.

Im folgenden werden zunächst die Anforderungen an ein solches Entwicklungswerkzeug analysiert. Anschließend wird mit dem System IFAS (Integriertes Finanzanalysesystem) ein Konzept für finanzwirtschaftliche Planungssysteme mit integrierter, problemspezifischer Entwicklungsumgebung[4] und dessen prototypische Implementation vorgestellt. Mit IFAS wird ein aus der betriebswirtschaftlichen Anwendungswelt stammendes Konzept - die Technik des betrieblichen Rechnungswesens - auf eine Systemarchitektur - in diesem Fall eine integrierte Anwendungs- und Entwicklungsumgebung - übertragen[5]. Dieser Ansatz wurde bereits in Planungsmodellen zur Analyse von Kauf- und Leasingverträgen in den Systemen KALEM und nachfolgend auch FES verfolgt[6]. Die dabei gemachten positiven Erfahrungen waren Anlaß für die Konzeption einer domänenunabhängigen Entwicklungsumgebung für finanzwirtschaftliche Planungssysteme.

3 Einen Überblick geben z.B. [4].

4 Problemspezifische Entwicklungsumgebungen werden nach Ansicht von [28] generell von zunehmender Bedeutung für die Softwareentwicklung sein.

5 Einer ähnlichen Überlegung folgen auch [1] mit dem Vorschlag, Anwendungssysteme anwendungsnah mit den Metaphern Werkzeug und Material zu konstruieren.

6 Literatur zu KALEM siehe [30] und FES vgl. [32]. KALEM wurde 1993 mit dem Deutsch-Österreichischen Hochschul-Software-Preis ausgezeichnet.

2. Ermittlung spezifischer Anforderungen an Entwicklungswerkzeuge für Finanzplanungssysteme

Aus der Planungssituation resultieren spezifische *Anforderungen an Finanzplanungssysteme*. Aus diesen sind spezifische *Anforderungen an Entwicklungswerkzeuge für Finanzplanungssysteme* ableitbar. Daher bestimmen wir zunächst die Anforderungen an die zu entwickelnden Systeme, bevor im zweiten Schritt die Anforderungen an eine Entwicklungsumgebung folgen.

2.1. Anforderungen an Finanzplanungssysteme

Softwaresysteme für finanzwirtschaftliche Planungsrechnungen unterstützen die Bewertungsphase im Entscheidungsprozeß[7]. Sie dürfen allerdings nicht mit Decision Support Systemen (DSS) gleichgesetzt werden, deren Aufgabe über die Erstellung von Planungsrechnungen für gegebene Alternativen hinausgeht[8]. Die hier behandelten Planungssysteme können anhand des in Abbildung 1 dargestellten Aufbaus beschrieben werden.

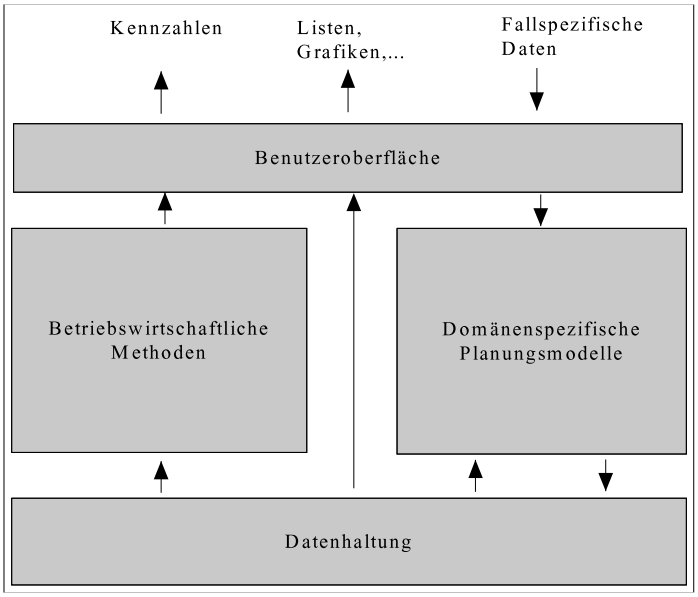

Abbildung 1: Architektur finanzwirtschaftlicher Planungssysteme

7 Hier wird die in der betriebswirtschaftlichen Literatur verbreitete Einteilung in Problemstellung, Alternativensuche, Alternativenbewertung und Entscheidung unterstellt (vgl. z.B. [12] S. 24f.). Daneben existieren in der DSS-Literatur weitere Phasenmodelle, z.B. bei [31] S. 109 ff. und [15] S. 260.

8 Vgl. z.B. [31] S. 105 ff. oder [33] S. 32. Viele Erkenntnisse aus der DSS-Forschung sind jedoch auf die hier diskutierten Systeme übertragbar, da die Zielgruppe und der betriebswirtschaftliche Anwendungskontext übereinstimmen, vgl. z.B. [15] S. 261 und [16] S. 31.

Über die Benutzeroberfläche tätigt der Anwender die für die Planungsrechnung notwendigen Eingaben und nimmt nach Durchführung der Planungsrechnung die gewünschten Auswertungen vor. Das domänenspezifische Planungsmodell ist eine Abbildung der Kausalbeziehungen, die zur Vorhersage der monetären Wirkungen einer Projektentscheidung vom Ersteller des Modells als relevant erachtet worden sind[9]. Damit ist in dieser Systemkomponente das Planungswissen der Fachexperten implizit oder explizit enthalten. Die Datenhaltungskomponente dient zur Speicherung der vom Anwender als Input für das Planungsmodell eingegebenen sowie aller errechneten Daten. Die Planungsdaten können i.d.R. in tabellarischer oder graphischer Form angezeigt oder durch Anwendung betriebswirtschaftlicher Methoden, z.B. der Kapitalwertmethode, zu Kennzahlen verdichtet werden.

Solche Planungssysteme richten sich primär an Mitarbeiter des Middle Managements oder in Stabsstellen. Dies sind nicht-gebundene (non-captive) Gelegenheitsnutzer, die die Nutzung eines Systems schlicht verweigern können[10]. Aus diesem Grunde ist eine nutzungsbezogene, d.h. den besonderen Erfordernissen der Planungssituation angepaßte, ergonomisch gestaltete *Benutzeroberfläche* erforderlich[11]. Insbesondere sollte die Präsentation der Planungsinformationen nicht nur tabellarisch, sondern auch graphisch möglich sein[12].

Die Palette der betriebswirtschaftlichen *Planungsmodelle und Methoden* des Systems sollte dynamisch erweiterbar und an geänderte Rahmenbedingungen anpaßbar sein, um flexibel auf geänderte Informationsbedarfe reagieren zu können. Dabei sollten die betriebswirtschaftlichen Modelle und Methoden voneinander unabhängig sein, damit jede Methode, sofern betriebswirtschaftlich sinnvoll, auf beliebige Planungsdaten, unabhängig vom domänenspezifischen Modell mit dem die Daten berechnet wurden, anwendbar ist. Damit die Transparenz der Planungsrechnung erhalten bleibt, ist zugleich die Option zur Disaggregation von Kennzahlen, z.B. im Rahmen von Drill Down-Analysen erforderlich[13].

Von großer Bedeutung sind kurze *Entwicklungszeiten* für Planungssysteme, da der Systemunterstützungsbedarf häufig ad hoc aus einem aktuell vorliegenden Entscheidungsproblem, mit entsprechenden zeitlichen Restriktionen, resultiert[14]. Darüber hinaus unterliegen viele für eine Finanzanaly-

9 Vgl. [23] S. 38.
10 Vgl. [22].
11 Vgl. z.B. [16] S. 33, [14] S. 124 oder [33] S. 102 ff.
12 Vgl. z.B. [5] S. 42. oder [14] S. 131 ff.
13 Vgl. z.B. [31] S. 35 ff.
14 Vgl. [26] S. 30. Kurze Entwicklungszeiten scheinen allgemein ein kritischer Erfolgsfaktor für die Akzeptanz führungsunterstützender Informationssysteme zu sein. Vgl. hierzu auch [36].

se relevanten Wissensdomänen, z.B. das Steuerrecht, einer hohen Änderungsdynamik. Dies impliziert hohe Anforderungen an die Möglichkeiten zur *Wartung und Pflege* der Prognosemodelle. Deshalb ist die Erstellung der Planungsmodelle sowie deren Wartung und Pflege durch die *Fachabteilung* einer zentralistischen Lösung durch die IV-Abteilung vorzuziehen[15]. Dies ist allerdings nur bei vergleichsweise geringen Anforderungen an die Informatikkenntnisse der hierfür verantwortlichen Mitarbeiter möglich.

Planungsinformationen sind eine Ressource von erheblichem Wert. Daraus resultieren hohe Ansprüche an das Zugangskontrollsystem, die Datensicherung aber auch die generelle Systemverfügbarkeit. Mit den durch eine Fachabteilung technisch beherschbaren Instrumenten, kann diesen Ansprüchen oftmals weder technisch noch wirtschaftlich sinnvoll entsprochen werden. Deshalb sollte die *Betreuung des operativen Betriebs* eine Dienstleistung der *IV-Abteilung* sein. Technische Voraussetzung für diese Arbeitsteilung ist eine hinreichend modulare Systemarchitektur, insbesondere die systemmäßige Trennung der Planungsmodelle von der Datenhaltungskomponente.

2.2. Anforderungen an Entwicklungswerkzeuge für Finanzplanungssysteme

Die betriebswirtschaftlichen Anforderungen an Finanzplanungssysteme implizieren spezifische Anforderungen an problemadäquate Entwicklungswerkzeuge für derartige Systeme. Insbesondere sollte die Entwicklung der Modelle und Methoden soweit wie möglich von nicht betriebswirtschaftlichen Aspekten befreit werden.

Dies gilt in besonderem Maße für die *Benutzeroberfläche.* Diese Komponente hat einen wesentlichen Einfluß auf den Gesamtaufwand, mit dem ein Softwareprojekt realisiert werden kann[16]. Für kurze Entwicklungszeiten ist daher die Minimierung des auf die Benutzeroberfläche entfallenden Gestaltungs- und Implementationsaufwands ein kritischer Erfolgsfaktor. Dies gilt verstärkt, wenn die Implementation und Pflege der Modelle und Methoden durch die Fachabteilung erfolgen und die Benutzeroberfläche zugleich hohen Ansprüchen an Funktionalität und Ergonomie genügen soll. Im Idealfall ist die Benutzeroberfläche ein separates Modul des Systems, so daß die Modelle und Methoden weitgehend ohne Entwurf und Implementation einer Benutzeroberfläche erstellt und gepflegt werden können.

15 Vgl. [34] S. 291.
16 Durchschnittlich 48% ([25] S.23).

Darüber hinaus sollte ein Entwicklungswerkzeug die *inkrementelle Entwicklung* des Gesamtsystems durch eine Architektur unterstützen, in der die Planungsmodelle und betriebswirtschaftlichen Methoden zu prinzipiell unabhängig voneinander nutzbaren Modulen des Gesamtsystems werden[17]. Daher müssen zusätzlich zu den in der einschlägigen Literatur genannten Qualitätskriterien[18] folgende Aspekte berücksichtigt werden:

- Bei der angestrebten inkrementellen Systementwicklung entstehen die Planungsmodelle und Methoden nicht als Ergebnis eines Top Down-Strukturierungsprozesses. Dennoch sollten einmal geschriebene Planungsmodelle in komplexeren Modellen wiederverwendet werden können. Gleiches gilt für die betriebswirtschaftlichen Methoden.
- Die Planungsmodelle und betriebswirtschaftlichen Methoden werden unabhängig voneinander erstellt. Sie sollen aber technisch beliebig miteinander kombinierbar sein, so daß jede Methode auf alle Planungsrechnungen angewandt werden kann.
- Alle Planungsmodelle und betriebswirtschaftlichen Methoden sollten unter einer einheitlichen Benutzeroberfläche verwendet werden können.

Um diese Eigenschaften realisieren zu können, ist die Kompatibilität der Module im Hinblick auf die verwendeten *Datenstrukturen* erforderlich. Dies kann durch die Verwendung eines einheitlichen flexiblen Datenschemas erreicht werden, dessen Einheitlichkeit durch das Entwicklungswerkzeug gewährleistet werden muß. Zugleich sollte das Entwicklungswerkzeug die Implementation der Modelle und Methoden von den technischen Details der Datenhaltung befreien.

Für die Implementation der Module, ist eine geeignete Sprache erforderlich. Aufgrund der angestrebten Implementation durch die Mitarbeiter der Fachabteilung, sollten Syntax und Semantik auf den betriebswirtschaftlichen Anwendungskontext abgestimmt sein. Darüber hinaus sollte die Implementationssprache zusätzlich den Qualitätskriterien des Software Engineering[19] genügen, z.B. durch

- Übersichtlichkeit und Selbstdokumentation der Syntax,
- Existenz von Datentypen und Zwang zu Deklarationen,
- Differenzier- und Überschaubarkeit von Gültigkeitsbereichen der Bezeichner und

17 Eine inkrementelle Einführungsstrategie (vgl. z.B. [7] und [19]) wird oft als Erfolgsfaktor für führungsunterstützende Systeme angesehen ([6]).

18 Vgl. z.B. [29] S. 52 ff. und [23] S. 296 f.

19 Vgl. z.B. [17] S. 26, [24] S. 291 und [29] S. 133 ff. Die nachfolgende exemplarische Aufzählung erhebt keinen Anspruch auf Vollständigkeit.

- Information Hiding in Verbindung mit Abstraktionsmöglichkeiten durch Funktionen, Prozeduren und Parameter.

3. Betriebswirtschaftliche Grundlagen des IFAS

Die Finanzplanung ist auf monetäre Aspekte eines Investitionsprojekts beschränkt. Dies ermöglicht die Realisation eines Entwicklungswerkzeugs für Finanzplanungssysteme auf Basis der Technik des betrieblichen Rechnungswesens, der doppelten Buchführung. Mit deren Kontenformalismus können die unterschiedlichsten Informationsbedarfe gedeckt werden. Trotz dieser Flexibilität wird die doppelte Buchführung bislang nur für vergangenheitsbezogene Dokumentationsrechnungen, nicht aber für zukunftsbezogene Planungsrechnungen eingesetzt[20].

Im Gegensatz dazu kommt im integrierten Finanzanalysesystem IFAS die Technik des betrieblichen Rechnungswesens für zukunftsbezogene Rechnungen zum Einsatz[21]. Da die doppelte Buchführung die konzeptionelle und terminologische Grundlage des IFAS ist, werden deren Grundzüge, soweit für die Konzeption des Entwicklungswerkzeugs relevant, kurz dargestellt[22]. Die Datenhaltungskomponente des IFAS basiert auf einem relationalen Datenbankmanagementsystem; daher ist die folgende Darstellung an eine relationale Sicht auf das betriebliche Rechnungswesen angelehnt.

Ein *Konto* k_i ist der logische Ort der Speicherung von Informationen über wertmäßige Änderungen einer Position aus dem Vermögens- und Kapitalbestand der Unternehmung. Die Menge der vorhandenen Konten wird als *Kontenplan* $P = \{k_1, ..., k_n\}$ bezeichnet. Eine auf einem Konto eingetragene Information wird als *Buchung* bezeichnet. Die Buchung ist im Rahmen dieser Arbeit ein geordnetes Tripel (k_i, t, x), wobei k_i das Konto der Buchung, t den Index für den Buchungszeitpunkt und x den Wert der Buchung angibt. Dabei ist $x \in IR \setminus \{0\}$, d.h. es sind nur Buchungen mit einem Betrag ungleich Null möglich. Die Summe aller Buchungen auf einem Konto wird als *Saldo* des Kontos bezeichnet. Der *Kontenabschluß* ist eine auf dem Saldo aufbauende Funktion: Ein Konto k_i wird auf ein Konto k_j abgeschlossen, indem der Saldo des Kontos k_i auf das Konto k_j gebucht wird. Wir schreiben diese Beziehung als $k_i \rightarrow k_j$. Das Konto k_i wird dann als Unterkonto des Kontos k_j be-

20 Vgl. [27] S. 4 und [3] S. 1099. .

21 Dies hat zudem den Vorteil, daß die Planbuchungen zum Zeitpunkt des "Existenzsprungs" eines geplanten Vorgangs mit den tatsächlichen Ist-Buchungen des begonnenen Vorgangs verglichen werden könnten, und somit die Datenbasis für ein leistungsfähiges Controlling unmittelbar aus der Planungsrechnung heraus entsteht (vgl. [27] S.4).

22 Eine ausführliche Einführung in die Technik des betrieblichen Rechnungswesens findet sich z.B. bei [18] und [8].

zeichnet. Dementsprechend ist das Konto k_j das Oberkonto des Kontos k_i. Ein Konto kann immer nur auf maximal ein anderes Konto abgeschlossen werden. Der Kontenabschluß ist eine informationsverdichtende Funktion der doppelten Buchführung, da das Konto k_j nach dem Abschluß des Kontos k_i dessen aggregierte Information enthält.

Ein kontenbasiertes Datenmodell besteht aus einem Kontenplan und einer zugehörigen Menge von Abschlußbeziehungen. Die Abschlußbeziehungen beschreiben den Weg der Informationsverdichtung innerhalb des jeweiligen kontenbasierten Datenmodells. Ein kontenbasiertes Datenmodell kann mit Hilfe des in Abbildung 2 dargestellten Metadatenmodells beschrieben werden[23].

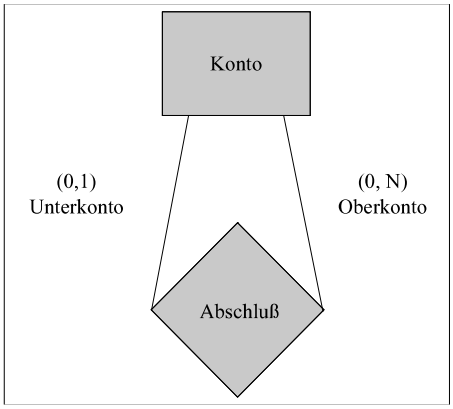

Abbildung 2: Meta-Datenmodell der doppelten Buchführung

Dieses Meta-Datenmodell beschreibt Bäume, deren Knoten die Konten des jeweiligen Kontenplans und deren Kanten die Abschlußbeziehungen zwischen diesen Konten sind. Dabei kann ein Konto als Abstraktionsschicht interpretiert werden, welche die Detailinformationen der Unterkonten vor dem Oberkonto verbirgt. Abbildung 3 zeigt ein Beispiel für ein solches kontenbasiertes Datenmodell. Interpretiert man in diesem Kontenplan das Konto "GuV I" als Bemessungsgrundlage für die Körperschaftsteuer, so ist es für die Steuerberechnung unerheblich, welche Aufwands- und Ertragskategorien in die steuerliche Bemessungsgrundlage einfließen. Diese Eigenschaft ermöglicht die einheitliche Anwendung gegebener Steuerberechnungsalgorithmen auf alle zu erstellenden Planungsmodelle. Dies reduziert den Entwicklungsaufwand und fördert die Konsistenz der Planungsmodelle.

23 Angabe der Komplexitätsgrade in (Min, Max) Notation. Notation nach [9], S. 57 ff.

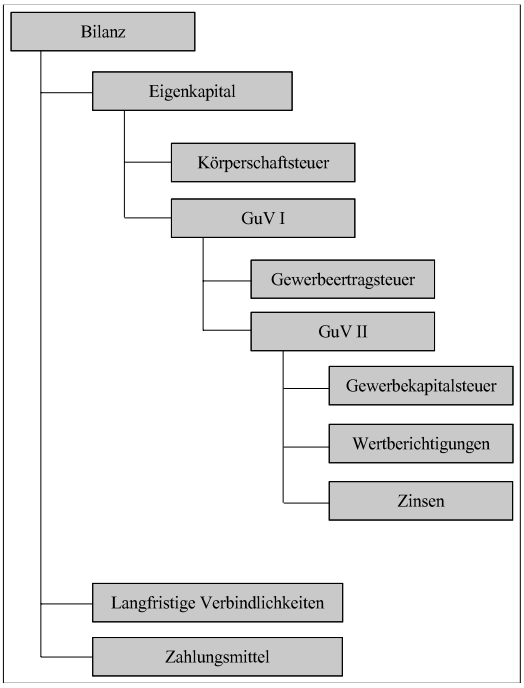

Abbildung 3: Beispiel für ein kontenbasiertes Datenmodell

Als *Transaktion* bezeichnen wir in IFAS ein im Zeitpunkt t eintretendes (geplantes) Realweltereignis, das eine Menge zusammengehöriger Informationen über Veränderungen im Vermögens- und Kapitalbestand der Unternehmung erzeugt[24]. Das Konstrukt zur Abbildung derartiger Transaktionen in der Planungsrechnung ist der *Buchungssatz*. Dies ist eine mindestens zweielementige Menge semantisch zusammengehöriger Buchungen $\{(t, k_1, x_1), ..., (t, k_z, x_z)\}$ für die gilt $\sum_{i=1,...,z} x_i = 0$. Die chronologisch geordnete Menge aller Buchungssätze wird als *Journal* bezeichnet und ist die mit einem Planungsmodell erstellte Datenbasis, die mit betriebswirtschaftlichen Methoden weiter aufbereitet werden kann.

4. Systemarchitektur

Die Planungsrechungen des IFAS beruhen auf fünf Annahmen. Die Annahmen 1 und 2 ermöglichen die Verwendung des Kontenformalismus für das Finanzanalysesystem. Die Annahmen 3 und 4 bilden die Grundlage für die Repräsentation temporaler Aspekte einer Finanzanalyse. Die Annahme 5 bewirkt die autonome Betrachtung einer einzelnen Entscheidungsalternative und ist Vorraussetzung für die Vergleichbarkeit mehrerer Alternativen.

24 Zu dieser Terminologie vgl. [11].

1. Jede Finanzplanungssituation mit einem endlichen Planungshorizont kann in eine nicht leere Menge von Transaktionen zerlegt werden.
2. Die monetären Aspekte einer jeden Transaktion können auf eine nicht leere Menge von Buchungssätzen vollständig abgebildet werden.
3. Der Planungszeitraum ist endlich und umfaßt n aufeinander folgende, gleich lange Perioden (z.B. Tage, Monate oder Jahre), die jeweils durch ihr Anfangs- und Endzeitpunkte definiert sind.
4. Die Planung erfolgt in einer diskreten Betrachtung der n+1 Zeitpunkte des Planungszeitraums. Die Zeitpunkte werden bei 0 beginnend fortlaufend nummeriert.
5. Zu Beginn des Planungszeitraums enthält das Journal nur die leere Menge.

Abbildung 4 zeigt die Systemarchitektur des IFAS[25]. Die Planungsbuchungen werden auf einem domänenspezifischen kontenbasierten Datenmodell durchgeführt. Die hierfür notwendigen Buchungsanweisungen werden in einer für diesen Zweck konzipierten Sprache beschrieben und durch einen zugehörigen Interpreter ausgeführt. Die Interaktion mit dem Benutzer erfolgt einheitlich durch eine von den jeweiligen Planungsmodellen unabhängige graphische Benutzeroberfläche.

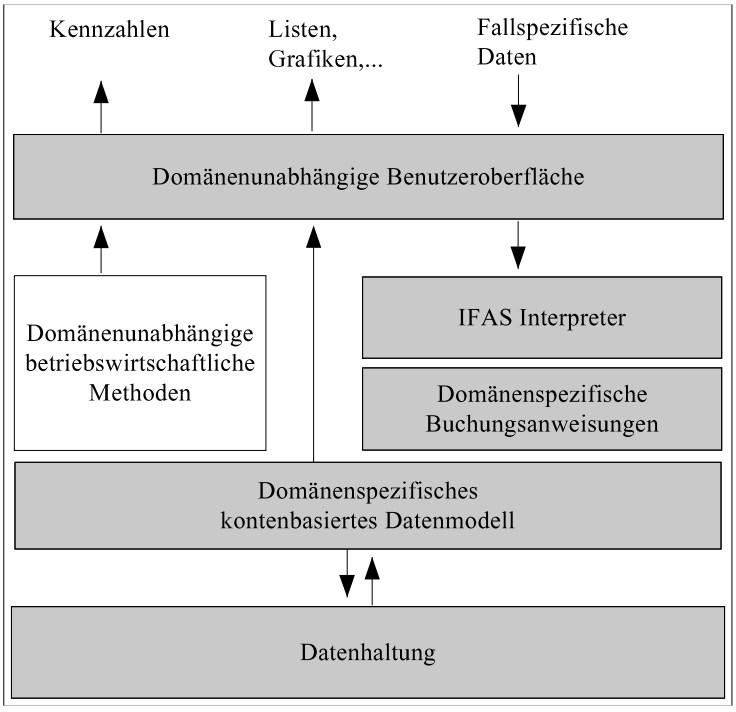

Abbildung 4: IFAS Systemarchitektur

25 Der IFAS Prototyp ist in C unter dem Betriebssystem OS/2 realisiert. Die Anwendungsdaten und die kontenbasierten Datenmodelle werden in einer DB2/2 Datenbank gespeichert. Betriebswirtschaftliche Methoden sind noch nicht implementiert, aber in der Systemkonzeption vorgesehen. Aus diesem Grund ist diese Systemkomponente transparent dargestellt.

5. Die Planungssprache des IFAS

Die Planungsrechnung wird anhand von *Buchungsanweisungen* durchgeführt. Die Buchungsanweisungen repräsentieren das Wissen, welche Transaktionen die Realisation einer Alternative auslösen würde und wie diese im Journal zu dokumentieren wären. Dabei handelt es sich um prozedurales Wissen, das durch eine *prozedurale Sprache* in geeigneter Weise repräsentiert werden kann. Die Programmiersprache des IFAS wird als Knowledge Description Language (KDL) bezeichnet.

Die KDL ist eine *strukturierte Sprache*[26]. Als Strukturierungseinheit werden *Transaktionsketten*[27] verwendet. Eine Transaktionskette ist eine mindestens einelementige Menge inhaltlich zusammengehöriger Transaktionen, die jeweils durch Buchungsanweisungen modelliert werden. Z.B. kann für Leasingverträge jede einzelne Zahlung einer Leasingrate als Transaktion modelliert und die Menge der Leasingzahlungen in einer Transaktionskette zusammengefaßt werden. Außerdem verfügt die KDL über *Steuerungskonstrukte* strukturierter Programmiersprachen, z.B. Sequenz, Verzweigung, Fallunterscheidung, kopf- und fußgesteuerte Schleifen sowie Zählschleifen.

Die KDL ist eine *typisierte Sprache*. Allerdings unterstützt sie lediglich *skalare Datentypen*. Dies stellt für den spezifischen Verwendungszweck keinen Nachteil dar, da die eigentliche Datenmodellierung außerhalb der KDL durch den Kontenplan erfolgt und das Erlernen der KDL für Nicht-Programmierer dadurch erleichtert wird. Die vorhandenen Datentypen stammen aus der Erfahrungswelt der Entwickler der Planungsmodelle. Als Datentypen sind *zeit*, *anzahl*, *geld* und *prozent* verfügbar. Die KDL differenziert die Gültigkeitsbereiche von Variablen in globale und lokale Variable. Bestehende Transaktionsketten können über spezifische Sprachkonstrukte in neue Programme eingebunden werden.

Die Syntax der KDL fördert die *Selbstdokumentation der Programme* durch erläuternde Texte, die für die Hilfefunktion des IFAS notwendig sind. Diese Texte sind für alle wesentlichen Elemente eines Planungsmodells (Konten, Transaktionen (Buchungsanweisungen) und Transaktionsketten) vorgesehen. Eingaben des Benutzers werden durch eine Funktion *user( )* ermöglicht. Diese Funktion zeigt einen als Parameter übergebenen Text am Bildschirm an und liest einen Wert vom Benutzer ein. Dieser Wert wird als Rückgabewert an den Aufrufer übergeben. Eine weitere Gestaltung der In-

26 Vgl. z.B. [20].

27 Ursprünglich wurde hierfür der Terminus Geschäftsprozeß verwendet, dieser ist jedoch in der Wirtschaftsinformatik bereits anderweitig belegt (vgl. z.B. [10]).

teraktion mit dem Benutzer ist in der KDL nicht möglich[28]. Dies ist auch nicht erforderlich, da die eigentliche Benutzeroberfläche als separates Modul vorliegt.

Ebenfalls von großer Bedeutung ist die *saldo( )* Funktion. Diese gibt die Summe aller Buchungen auf einem Konto zwischen zwei Zeitpunkten zurück und ermöglicht so die Aggregation von Buchungen auf einem Konto über beliebige Zeiträume. Die saldo( ) Funktion ist erforderlich, da durch die Buchungen lediglich Informationen über die Wertänderungen eines Kontos explizit in der Datenbasis gespeichert werden. Jede bestandsorientierte Information, z.B. Kassenbestand zum Zeitpunkt t, muß daher aus den Buchungen errechnet werden.

Die Planungsrechnungen sollen in einer Nach-Steuer-Betrachtung durchgeführt werden. Zu diesem Zweck werden bestimmte Konten vorgegeben, deren Saldo die Bemessungsgrundlage für die verschiedenen zu berücksichtigenden Steuern darstellt. Aus Sicht des Software Engineering können diese Konten als *Abstraktionsschicht* interpretiert werden, welche die Details der der steuerlichen Bemessungsgrundlage zugrundeliegenden Transaktionen vor der Steuerberechnung verbirgt und umgekehrt. Dies ermöglicht die einheitliche Verwendung *eines* Steuerberechnungsmoduls für alle Kategorien von Entscheidungsalternativen. Details der Steuerberechnung, z.B. der Steuersatz, werden diesem Modul als Parameter übergeben. Hieraus folgen *deklarative Sprachkonstrukte*, mit denen die Berechung der Steuerwirkungen beeinflußt werden kann. Die folgenden Ausschnitte der KDL-Grammatik vermitteln einen Eindruck von der Syntax der KDL[29].

Ein KDL-Programm gliedert sich in drei Teile. Auf die (optionale) Definition globaler Variablen folgen die Definitionen der Transaktionsketten. Ein KDL-Programm endet mit der Parametrisierung des Jahresabschlusses. Der Programmablauf beginnt mit der syntaktisch vorgeschriebenen Transaktionskette *start( )*. Dies muß nicht die erste innerhalb eines KDL-Programms definierte Transaktionskette sein, sondern die Transaktionsketten können in beliebiger Reihenfolge definiert werden. Dabei enthält der Kopf einer Transaktionskette nicht nur die Definition der für eine modulare Programmierung erforderlichen Parameter, sondern auch einen erläuternden Text, der den Anwender über die betriebswirtschaftliche Bedeutung der Transaktionskette informiert.

---

28 Die user( ) Funktion ist als separater Prozeß implementiert. Das zugrundeliegende Programm kann durch eine Einstellung in einer Konfigurationsdatei flexibel geändert werden. Dies eröffnet flexible Änderungsmöglichkeiten für die Benutzeroberfläche, ohne daß die KDL Programme geändert werden müssen.

29 Terminalsymbole sind fett dargestellt. Hier nicht definierte Nicht-Terminalsymbole sind kursiv dargestellt. Das Symbol ▪ stellt einen leeren String dar. Eine ausführliche Sprachbeschreibung ist auf Anfrage erhältlich.

Programm → *Globale_Variablen* Gesch_Transaktionen Jahresabschluß
Gesch_Transaktionen → GT_Liste Programmstart GT_Liste
GT_Liste → GT_Definition GT_Liste |
GT_Definition → *GT_Bezeichner* **(** GT_Parameter **)** GT_Körper
GT_Parameter → Datentyp *P_Bezeichner* GT_Parameter1
Datentyp → **zeit** | **geld** | **prozent** | **anzahl**
GT_Parameter1 → **,** GT_Parameter2 |
GT_Parameter2 → GT_Parameter | *GT_Erläuterungstext*
GT_Körper → **{** *Lokale_Variablen* Anweisungsfolge **}**
Anweisungsfolge → Anweisung Anweisungsfolge |
Anweisung → Buchungsanweisung | *Funktion* | *Verbundanweisung* |
Verzeigung | *Auswahl* | *kopfgestSchleife* | *fußgestSchleife* | *ZählSchleife* |
Zuweisung | *Transaktionskette* | *Leere_Anweisung* |
Programmstart → **start( )** GT_Körper

Von den in der KDL möglichen Anweisungen wird nachfolgend die Buchungsanweisung näher dargestellt. Sie beschreibt Buchungssätze im Sinne des Abschnitts drei. *Wert* gibt den jeweiligen Wert einer Buchung auf dem vorstehenden Konto *Kontenbezeichner* an. Innerhalb eines Buchungssatzes sind beliebig viele Konten ansprechbar, so daß auch komplexe Ereignisse adäquat in der Planungsrechnung dargestellt werden können. Der in die Buchungsanweisung integrierte Erläuterungstext vermittelt dem Anwender den betriebswirtschaftlichen Hintergrund des jeweiligen Buchungssatzes.

Buchungsanweisung → **bs{** *Zeitpunkt* **,** *Kontenbezeichner Wert* **,** *Kontenbezeichner Wert* **,** BS_Rest **};**
BS_Rest → **,** BS_Rest1 |
BS_Rest1 → *Kontenbezeichner Wert* | *BS_Erläuterungstext*

Im Gegensatz zu den durch prozedurale Transaktionsketten beschriebenen, vor-steuerlichen Aspekten eines Planungsmodells, wird die Planung der steuerlichen Folgen deklarativ gesteuert. Dies erfolgt durch Wertzuweisungen an die Parameter einer einheitlich für alle Planungsmodelle verwendeten "Jahresabschluß"-Funktion des IFAS. Die zu spezifizierenden Parameter sind reservierte Schlüsselworte der KDL. Voraussetzung hierfür ist lediglich die Existenz bestimmter Konten, z.B. des Kontos "GuV I" für die Einkommen- bzw. Körperschaftsteuer. Welche Aufwands- und Ertragskategorien in den jeweiligen Planungsmodellen berücksichtigt werden, ist für den Steuerberechnungsalgorithmus irrelevant[30].

30 Vgl. Abschnitt 3.

Jahresabschluß	→	*Startzeitpunkt* AfA_Daten *GewSt_Daten* ESt_Daten
AfA_Daten	→	**[ABSCHREIBUNG]** AfA_Parameter
AfA_Parameter	→	**KONTO =** *Kontenbezeichner_Vermögensgegenstand*
		BGND = *Betriebsgewöhnliche_Nutzungsdauer*
		METHODE = *Abschreibungsmethode*
		GEGENKONTO = *Kontenbezeichner_Aufwandskonto*
		; AfA_Parameter
Est_Daten	→	**[EINKOMMENSTEUER]** ESt_Parameter
Est_Parameter	→	**EINKOMMENSTEUER =** *Steuersatz*
		KONTO = *Kontenbezeichner_belastetes_Konto*

Die folgenden Programmausschnitte vermitteln einen Eindruck von der KDL-Programmierung. Die Transaktionskette verkauf( ) zeigt die Verwendung der user( ) Funktion für Benutzereingaben. Sie verwendet die saldo( ) Funktion zur Ermittlung des Restbuchwertes einer früher gekauften Maschine im Verkaufszeitpunkt. Bei den beiden folgenden Verzweigungen wird aus Gründen der Übersichtlichkeit auf den optionalen "else" Zweig verzichtet.

```
verkauf("Am Ende der Nutzungszeit wird die Maschine verkauft. Dabei kann, je nach Relation
        zwischen Restbuchwert und Verkaufserlös, ein außerordentlicher Aufwand oder Ertrag entstehen")
{
  ZEIT verkaufszeitpunkt;
  GELD verkaufspreis, restbuchwert;

  verkaufszeitpunkt = USER("Bitte geben Sie den Verkaufszeitpunkt ein");
  verkaufspreis = USER("Bitte geben Sie den Verkaufspreis ein");
  restbuchwert = SALDO(0, verkaufszeitpunkt, MASCHINEN);

  if(restbuchwert > verkaufspreis)

    bs{verkaufszeitpunkt,
       KASSE +verkaufspreis,
       AUFWAND +(restbuchwert-verkaufspreis),
       MASCHINEN -restbuchwert,
       "Verkauf unter Restbuchwert"};

  if(restbuchwert < verkaufspreis)

    bs{verkaufszeitpunkt,
       KASSE +verkaufspreis,
       ERTRAG +(verkaufspreis-restbuchwert),
       MASCHINEN -restbuchwert,
       "Verkauf über Restbuchwert"};
}
```

Ein KDL-Programm kann mehrere Abschreibungsabschnitte enthalten, die jeweils beschreiben, wie die durch ein bestimmtes Konto repräsentierte Vermögensposition abzuschreiben ist. Im Beispiel ist hier die degressiv lineare Abschreibung mit Vereinfachungsregel nach Abs. 44 EstR. durch das Schlüsselwort deglin44 vorgegeben. IFAS unterstützt aber auch andere, u.a. frei definierbare Abschreibungsmethoden.

```
[ABSCHREIBUNG]
Konto = MASCHINEN
BGND = USER("Bitte geben Sie die betriebsgewöhnliche Nutzungsdauer ein")
Methode = deglin44
Gegenkonto = ABSCHREIBUNGEN
```

Der folgende Abschnitt determiniert die Berechnung der Einkommensteuer. Auf ähnliche Weise können zusätzlich die Gewerbekapital- und -ertragsteuer parametrisiert werden.

```
[EINKOMMENMSTEUER]
Einkommensteuer = USER("Bitte geben Sie den Steuersatz ein")
Konto = KASSE
```

6. Die Benutzeroberfläche des IFAS

Die Benutzeroberfläche stellt einen komfortablen Browser für die Planungsbuchungen bereit. Abbildung 5 zeigt einen Screenshot dieser Systemkomponente. Jeder Knoten des abgebildeten Baums enthält bestimmte Informationen. Zusätzlich ist zu jedem Knoten ein erläuternder Text hinterlegt.

Die Wurzel des Baums enthält die Beschreibung der berechneten Alternativen. Aus der Wurzel verzweigt der Baum für jede Alternative in einen separaten Ast. Die Knoten auf der zweiten Stufe enthalten das Journal der jeweiligen Alternative. Auf der dritten Stufe können die Buchungssätze selektiv abgerufen werden. Eine Möglichkeit ist die Selektion nach Transaktionsketten. Dies ist bis auf die Ebene einzelner Buchungen möglich. Auf diese Weise können Fragen nach bestimmten Transaktionsketten, z.B. dem Tilgungsverlauf eines Kredits, beantwortet werden. Die andere Möglichkeit besteht in der Auswahl relevanter Konten des Kontenplans. Dies ermöglicht Aussagen über die wertmäßige Entwicklung von Vermögens- und Kapitalpositionen - z.B. des Zahlungsmittelbestands - unabhängig von der Frage, welche Transaktionsketten diesen beeinflussen.

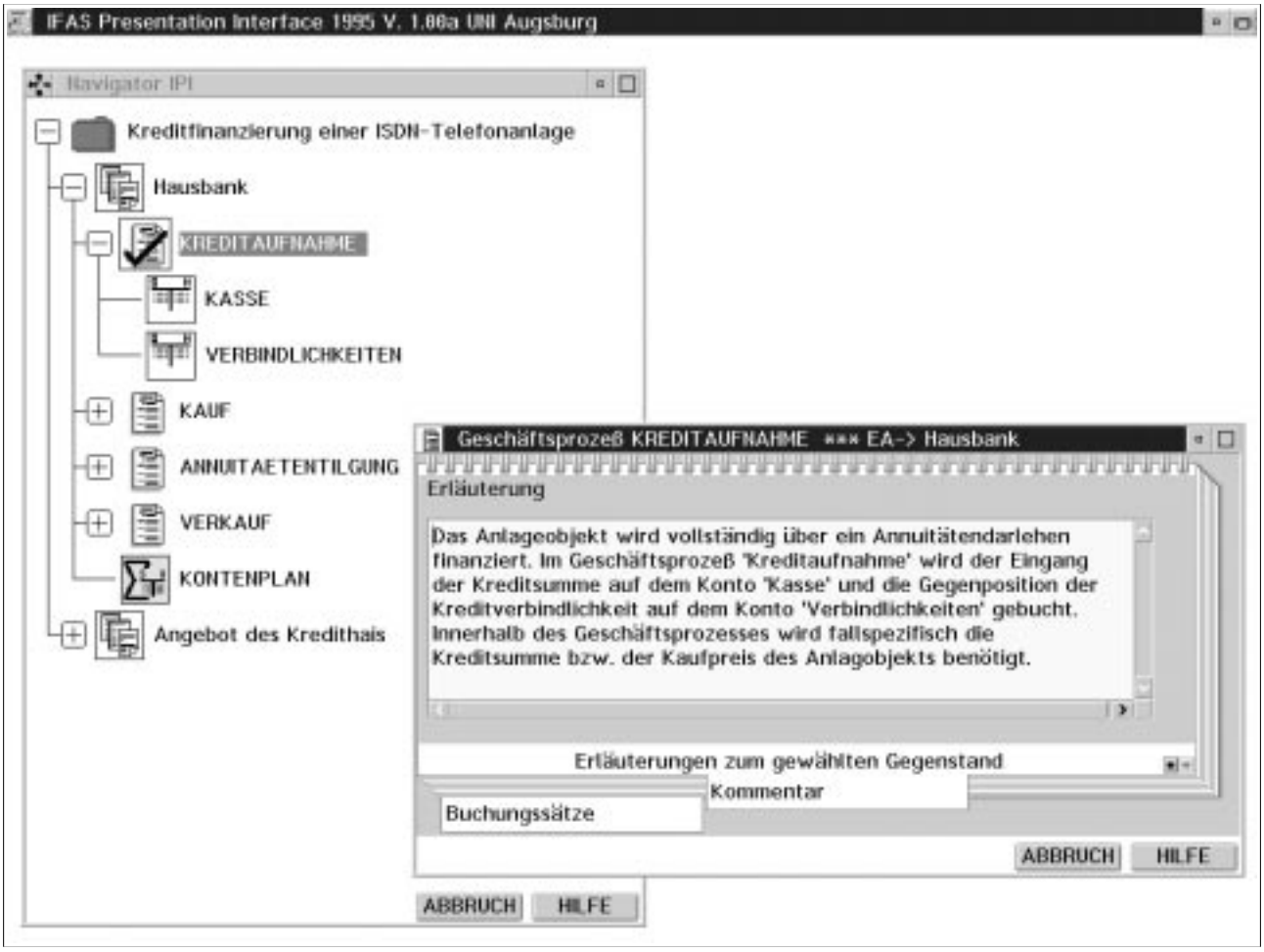

Abbildung 5: IFAS Benutzeroberfläche

7. Zusammenfassung und Ausblick

IFAS stellt eine problemorientierte Programmiersprache für die effiziente und konsistente Entwicklung von Modellen zur Planung monetärer Wirkungen von Investitions- und Finanzierungsentscheidungen bereit. Die Erstellung der Planungsmodelle ist weitgehend frei von nicht-betriebswirtschaftlichen Aspekten und kann somit durch die Fachabteilungen erfolgen. Die modulare Systemarchitektur ermöglicht den Einsatz eines leistungsfähigen relationalen Datenbankservers, ohne die Entwickler der Planungsmodelle mit den damit verbundenen technischen Fragestellungen zu konfrontieren. Es sind z.B. keine Kenntnisse über SQL und Datenmodellierung erforderlich. Der dem IFAS zugrundeliegende Datenbankserver kann durch die IV-Abteilung administriert und auf diese Weise ein hohes Maß an Datenschutz und Datensicherheit realisiert werden. Das zugrundeliegende Meta-Datenmodell ermöglicht zudem ein hohes Maß an Wiederverwendbarkeit. Dies gilt sowohl für einzelne Systembestandteile des IFAS, z.B. die grafische Oberfläche, Steuerberechnungsalgorithmen

und die Datenhaltungskomponente, als auch für die mit IFAS erstellten Planungsmodelle selbst. Die hier erzielbaren Vorteile werden künftig durch die Integration objektorientierter Ansätze in die Entwicklung der Planungsmodelle weiter ausgebaut, indem die domänenspezifischen Kontenpläne in den Planungsmodellen gekapselt werden.

Der Übertragung wissenschaftlicher Erkenntnisse in die Anwendungspraxis, kommt aus Sicht der Wirtschaftsinformatik als anwendungsorientierter Wissenschaft besondere Bedeutung zu. Daher sollen die in IFAS prototypisch implementierten Konzepte in Zusammenarbeit mit einem Praxispartner validiert werden. Dies, die Integration der Planungsrechnung in ein Controllingsystem und der Aufbau einer betriebswirtschaftlichen Methodenbank sind Gegenstand aktueller und zukünftiger Forschungsaktivitäten des Autors.

Literatur

[1] BÄUMER, D.; BUDDE, R.; SYLLA, K.-H.; GRYCZAN, G.; ZÜLLIGHOVEN, H.: Objektorientierte Konstruktion von Software-Werkzeugen und -Materialien, in: Informatik Spektrum 4/1995, S. 203-210.

[2] BEHME, W.; SCHIMMELPFENG K. (Hrsg.): Führungsinformationssysteme - Neue Entwicklungen im EDV-gestützten Berichtswesen, Gabler, Wiesbaden 1993.

[3] BRÜNING, G.: Die Bilanz als Vergangenheits- oder Zukunftsrechnung, in: Zeitschrift für Betriebswirtschaft 12/1979, S. 1099-1124.

[4] BUHL, H.U.; HASENKAMP, U.; MÜLLER-WÜNSCH, M.; ROßBACH, P.; SANDBILLER, K.: Wettbewerbsorientierte IT-Unterstützung in der Finanzberatung, in: Wirtschaftsinformatik 3/1993, S. 262-279.

[5] BULLINGER, H.J.; HUBER, H.; KOLL, P.: Chefinformationssysteme: Navigationsinstrumente für das Topmanagement, in: Office Management 6/1990, S. 40-44.

[6] BULLINGER, H.J.; NIEMEIER, J.; KOLL, P.: Führungsinformationssysteme (FIS): Einführungskonzepte und Entwicklungspotentiale, in: [2].

[7] EBERS, M.: Die Einführung innovativer Informationssysteme - Gestaltungsparameter und Gestaltungsoptionen, in: Zeitschrift für Organisation 2/1991, S. 99-106.

[8] EISELE, W.: Technik des betrieblichen Rechnungswesens - Buchführung, Kostenrechnung, Sonderbilanzen, 2. Aufl., Vahlen, München 1985.

[9] ELMASRI, R.; NAVATHE, S.B.: Fundamentals of Database Systems, Benjamin/ Cummings, Redwood 1994.

[10] FERSTL, O.K.; SINZ, E.J.: Geschäftsprozeßmodellierung, in: Wirtschaftsinformatik 6/1993, 589-592.

[11] GERARD, P.: Der Weg zum unternehmensweit integrierten Informationssystem. Teil 2: Prozeßorientiertes Informationsmanagement bei der Deutschen Bank, in: Computerwoche, 43/1993, S. 19-22.

[12] HAHN, D.: Planungs- und Kontrollrechnung, Gabler, 3. Aufl., Wiesbaden 1986.

[13] HICHERT, R., MORITZ, M. (Hrsg.): Managementinformationssysteme - Praktische Anwendungen, Springer, 2. Aufl., Berlin 1995.

[14] HOFFMANN, H.: Computergestützte Planung als Führungsinstrument: Grundlagen - Konzept - Prototyp, Gabler, Wiesbaden 1993.

[15] HUMMELTENBERG, W.: Realisierung von Management-Unterstützungssystemen mit Planungssprachen und Generatoren für Führungsinformationssysteme, in: [13].

[16] JAHNKE, B: Einsatzkriterien, kritische Erfolgsfaktoren und Einführungsstrategien für Führungsinformationssysteme, in: [2].

[17] KLAEREN, H.: Probleme des Software-Engineering: Die Programmiersprache - Werkzeug des Softwareentwicklers, in: Informatik Spektrum 1/1994, S. 21-28.

[18] KOCH, J.: Betriebliches Rechnungswesen 1: Buchführung und Bilanzen, Physica, Heidelberg 1989.

[19] KRÜGER, W.: Organisatorische Einführung von Anwendungssystemen, in: [21].

[20] KURBEL, K.: Programmentwicklung, 5. Aufl., Gabler, Wiesbaden 1990.

[21] KURBEL, K.; Strunz, H. (Hrsg.): Handbuch Wirtschaftsinformatik, Poeschel, Stuttgart 1990.

[22] KUBICEK, H.;TAUBE, W.: Die gelegentlichen Nutzer als Herausforderung für die Systementwicklung, in: Informatik Spektrum 6/1994, S. 347-356.

[23] LEHNER, F.; HILDEBRAND, K; MAIER, R.: Wirtschaftsinformatik: Theoretische Grundlagen, Hanser, München 1995.

[24] LUDEWIG, J.: Sprachen für das Software Engineering, in: Informatik Spektrum 5/1993, S. 286-294.

[25] MANHARTSBERGER, M.; PENZ, F.; TSCHELIGI, M.: N/Joy - Ein Designbeispiel für eine direkt manipulative, objektbasierte Benutzerschnittstelle, in: Informatik Forschung und Entwicklung 1/1993, S. 23-34.

[26] MERTENS, P.; GRIESE, J.: Integrierte Informationsverarbeitung 2 - Planungs- und Kontrollsysteme in der Industrie, Gabler, 7. Aufl., Wiesbaden 1993.

[27] MÜLLER-MERBACH, H.: Buchhaltung ohne Wandel: 500 Jahre nach Paccioli - Ungenutzte Computerchancen, in: Technologie & Management 1/1994, S. 3-6.

[28] NAGL, M.: Software-Entwicklungsumgebungen: Einordnung und zukünftige Entwicklungslinien, in: Informatik Spektrum 5/1993, S. 273-280.

[29] POMBERGER, G.; BLASCHEK, G.: Software Engineering - Prototyping und objektorientierte Software-Entwicklung, Hanser, München 1993.

[30] SCHNEIDER, J.: KALEM - Entwicklung und Implementation eines zahlungsstromorientierten Analyseverfahrens für Kauf- und Leasingverträge über Mobilien. Diplomarbeit am Lehrstuhl für Betriebswirtschaftslehre mit Schwerpunkt Wirtschaftsinformatik, Universität Giessen.

[31] VETSCHERA, R.: Informationssysteme der Unternehmensführung, Springer, Berlin 1995.

[32] WEINHARDT, CH.; DETLOFF, U.; GOMBER, P.; KRAUSE, R.; SCHNEIDER, J.: IV-Unterstützung in der Finanzierungsberatung - Integration von Methoden und Paradigmen, in: Wirtschaftsinformatik 1/1994, S. 5-14.

[33] WERNER, L.: Entscheidungsunterstützungssysteme - Ein problem- und benutzerorientiertes Management-Instrument, Physica, Heidelberg 1992.

[34] WIESER, H.: EIS/MIS-Einführung in einer prozeßorientierten Betrachtung, in: [13].

[35] WILL, A.: Die Erstellung von Allfinanzprodukten: Produktgestaltung und verteiltes Problemlösen, Gabler, Wiesbaden 1995.

[36] YOUNG, D.; WATSON, H.J.: Determines of EIS acceptance, in: Information & Management 1995, S. 153-164.