Ein Konzept für ein IV-Controllingsystem

Diplomarbeit

vorgelegt von

Arne-Christian Sigge

Fakultät für Wirtschaftswissenschaften

Universität Bielefeld

im Januar 1999

Themensteller:
Prof. Dr. Thorsten Spitta

Zweitprüfer:
Prof. Dr. Peter Naeve

Inhaltsverzeichnis

1 Einleitung

Die Konzeption eines Softwaresystems ist ein vielschichtiger Prozeß, nicht alle Aspekte dieses Prozesses können deshalb in einer Diplomarbeit behandelt werden. Ziel dieser Arbeit ist es, das Datenmodell eines IV-Controllingsystems für einen realen Einsatz zu entwerfen. Die Arbeit soll dabei drei wesentliche Dinge leisten:

- Das *Problem*, das durch den Einsatz des IV-Controllingsystems auf der Basis des zu entwerfenden Datenmodells gelöst werden soll, muß beschrieben werden.
- Es muß ein *Weg* gefunden werden, *dem Anwender das Datenmodell adäquat zu vermitteln.*
- Das fertige Datenmodell ist als erster Schritt einer *Lösung* des Problems darzustellen.

Die Arbeit orientiert sich an den Anwenderbedürfnissen eines IV-Controllingsystems. Nachdem der Anwender sich darüber bewußt geworden ist, wie sein Problem aussieht und verifiziert hat, daß tatsächlich *sein* Problem vom Entwickler erfaßt wurde, interessiert den Anwender nur noch die Lösung dieses Problems.

Auf den allgemeinen Problemgegenstand *IV-Controlling* wird in Kapitel 2 eingegangen. Die Darstellung eines konkreten Anwenderproblems erfolgt in Kapitel 3. Hier werden die an das IV-Controllingsystem vom Anwender gestellten Anforderungen und die Situation, in der dieses System benötigt wird, geschildert.

Vor der Präsentation des Datenmodells ist zu klären, *wie* man das Modell darstellen will, damit der Anwender die ihm präsentierte Lösung versteht. In Kapitel 4 wird beschrieben *wie* und *warum gerade so* das Modell in dieser Arbeit spezifiziert wird. In Anlehnung an Mordau [Mor98] wird versucht, formale Methoden in die Spezifikation zu integrieren.

Die Lösung des in Kapitel 3 aufgezeigten konkreten Problems wird im fünften Kapitel präsentiert. Das hier vorgestellte Datenmodell ist der eigentliche Kern dieser Arbeit. Beschrieben wird ein Datenmodell, das sich in das existierende Unternehmensdatenmodell einbetten läßt und die Grundlage für ein IV-Controllingsystem bildet. Im Anschluß an die Darstellung wird überprüft, ob das Modell im Widerspruch zu dem von Spitta [Spi98c] vorgestellten Referenzmodell steht.

Bisher unberücksichtigt geblieben ist der Weg der Entstehung des Modells. Für den Anwender ist es in den meisten Fällen zu aufwendig, zu hinterfragen *warum das Problem genau so* gelöst wurde, oder es fehlt ganz einfach die Zeit dazu, sich damit auseinanderzusetzen. Für den Leser, der diese Arbeit aus rein wissenschaftlicher

Sicht - und nicht aus Sicht eines Anwenders - betrachtet, ist der Weg teilweise sogar interessanter als das fertige Modell selbst. In dieser Arbeit steht jedoch das *Ergebnis* und nicht der *Weg* im Vordergrund.

Eine umfangreiche Analyse des Entstehungsprozesses würde genügend Stoff für eine zweite Diplomarbeit bieten und eine tiefere Auseinandersetzung mit soziologischen und psychologischen Aspekten erfordern. Das würde jedoch den Umfang dieser Arbeit sprengen.

Trotzdem soll die Entstehungsgeschichte des Modells nicht vollständig übergangen werden. In Kapitel 6 wird an einigen Stellen exemplarisch gezeigt, wo und warum es Probleme gegeben hat. Dieses Kapitel läßt erahnen, welche Gefahren bei der Formulierung des Problems und auf dem Weg vom Problem zur Lösung zu meistern sind. Ein formal korrekt spezifiziertes System kann immer nur die Lösung des vorher definierten Problems sein.

Daher prägte auch den hier nur ausschnittsweise offengelegten Entstehungsprozeß die für die Entwicklung eines Softwaresystems so wichtige Frage: *„Haben wir unser Problem wirklich verstanden und es korrekt und vollständig beschrieben?“*

2 IV-Controlling

Bevor Anforderungen an ein IV-Controllingsystem gestellt werden und mit dem Entwurf begonnen wird, sollte man sich klar machen, was unter IV-Controlling zu verstehen ist.

IV-Controlling ist das Controlling des IV-Bereiches, wobei das Kürzel IV für *Informationsverarbeitung* steht. Es könnte aber ebenso durch IT (für *Informationstechnologie*), IS (für *Informationssysteme*) oder DV (für *Datenverarbeitung*) ersetzt werden. Eine allgemein anerkannte und griffige Definition von *Controlling* zu finden, ist schwierig.

2.1 Controlling

In der Literatur besteht keine Einigkeit darüber, welche Aufgabengebiete das Controlling abdecken sollte ([Hor96], S. 67; [Bec90], S. 296; [KWZ90], S. 282). Folgt man den Beschreibungen der Tätigkeitsfelder eines Controllers in diesen Literaturquellen, dann kann man den Eindruck bekommen, daß in einigen Unternehmen lediglich das Türschild des Rechnungswesens ausgetauscht wurde, in anderen wiederum sitzen die Controller in der Unternehmensführung oder in der Revision.

Eine Auseinandersetzung mit den zahlreichen Versuchen, den Begriff *Controlling* zu definieren würde an dieser Stelle zu weit führen. Diese Arbeit orientiert sich an zwei Definitionen:

Horváth definiert Controlling folgendermaßen: *„Controlling ist - funktional gesehen - dasjenige Subsystem der Führung, das Planung und Kontrolle sowie Informationsversorgung systembildend und systemkoppelnd ergebniszielorientiert koordiniert und so die Adaption und Koordination des Gesamtsystems unterstützt.“* ([Hor96], S. 141).

Unter *systembildend koordinieren* ist dabei die Etablierung eines Planungs- und Kontrollsystems sowie eines Informationssystems zu verstehen, die durch Schaffung einer Organisationsstruktur zu einem Subsystem der Unternehmensführung verbunden werden (vgl. [Hor96], S. 121ff.). Die *systemkoppelnde* Funktion gleicht Störungen beim Zusammenwirken der verschiedenen Subsysteme aus.

Nach Küpper, Weber und Zünd besteht die Controllingfunktion *„ ...im Kern in der Koordination des Führungsgesamtsystems zur Sicherstellung einer zielgerichteten Lenkung. Sie bezieht sich insbesondere auf die Gestaltung und Überwachung des Planungs-, Kontroll- und Informationssystems“* (vgl.[KWZ90], S. 283).

Vier zentrale Begriffe tauchen in beiden Definition auf: *Planung, Kontrolle, Information* und *Koordination*, wobei der Begriff *Steuerung* an Stelle von *Koordination*

treffender ist, da er der eigentlichen Bedeutung des Begriffes *Controlling*, der von *to control - steuern, lenken* abgeleitet wurde (vgl [Hor96] S. 25f.), näherkommt.

Diese vier Begriffe stehen für die zentralen Funktionen des Controllings, so wie sie in dieser Arbeit gesehen werden:

- **Planung:** Aufstellen von Plänen
- **Kontrolle:** Sammeln und Vergleichen von Ist-Daten mit Plan Daten
- **Information:** Generierung von Informationen aus Plan- und Ist-Daten
- **Steuerung:** Mitwirkung an der Steuerung von Unternehmensprozessen auf Basis der gewonnenen Informationen.

Man unterscheidet zwischen *strategischem* Controlling, das die langfristigen Unternehmensziele im Blick hat (*„to do the right things“*) und *operativem* Controlling, das die kurz- bis mittelfristigen Belange im Auge hat (*„to do the things right“*). Teilweise ist auch vom *taktischen* Controlling die Rede, das speziell den mittelfristigen Bereich abdecken soll. In dieser Arbeit wird jedoch nur zwischen operativem und strategischem Controlling unterschieden.

Der Schwerpunkt liegt heutzutage immer noch im operativen Controlling, da sich das Controlling aus dem konkreten Tagesgeschäft heraus entwickelt hat, und in der kurz- bis mittelfristigen Lenkung von Unternehmen groß geworden ist (vgl. [KWZ90], S. 284).

Welche Zeiträume unter lang-, mittel- und kurzfristig zu verstehen sind, ist stark problem- und branchenbezogen. Küpper sieht einen Zeitraum von 5 bis über 10 Jahren als langfristig, bis ca. 5 Jahre als mittelfristig und bis zu einem Jahr oder weniger als kurzfristig an (vgl. [Küp97], S. 64). Für den IV-Bereich mit seinen schnellen technologischen Entwicklungssprüngen sind diese Zeiträume sicher nicht angemessen. Hier kann durchaus schon ein ein Zeitraum von 3 Jahren als langfristig gelten.

2.2 Controlling im IV-Bereich

2.2.1 Was ist IV-Controlling?

Die Bedeutung, die der *Information* für den Erfolg von Unternehmen zugemessen wird, führt zu der Einsicht, daß *Information* neben den klassischen betriebswirtschaftlichen Faktoren als *Produktionsfaktor* anzusehen ist (vgl. [Sch90] S. 105). Daraus ergibt sich die Forderung, daß der Einsatz des *Produktionsfaktors Information* der Planung, Kontrolle und Steuerung unterzogen werden muß (vgl. [Krü96], S. 3).

Braucht der IV-Bereich ein anderes Controlling als andere Unternehmensbereiche? Fest steht, daß sich der IV-Bereich von Bereichen, in denen sich das „herkömmliche Controlling“ etabliert hat, unterscheidet.

Nach Griese [Gri87] entfallen 45-60% und nach Boehm [Boe81] sogar 90% des Aufwandes im IV-Bereich auf Personalkosten. Daher muß den Personalkosten besondere Aufmerksamkeit geschenkt werden. Im Gegensatz zu beispielsweise einer Fließbandproduktion, ist der Personaleinsatz im IV-Bereich sehr flexibel. Ein Mitarbeiter arbeitet an einem Tag an unterschiedlichen Projekten und betreut verschiedene Systeme. Die Zuordnung seiner Arbeitsleistung zu einem bestimmten Kostenträger kann demnach nicht pauschal über die Daten des Rechnungswesens erfolgen, wie es bei einem Mitarbeiter am Fließband möglich ist. Die Daten über den tatsächlichen Einsatz der Mitarbeiter müssen zusätzlich erfaßt werden.

Neben der Problematik der Zuordnung des Personalaufwands erfordern auch die Aufgaben des IV-Bereiches eine besondere Beachtung. Die für den IV-Bereich relevanten Controllingaufgaben lassen sich nach Sokolovsky und Kraemer [SoKr90] und nach Krcmar [Krc97] einteilen in:

- **Portfolio-Controlling**[1]**:** Die in einem Unternehmen vorhandenen, auf eine IV-Unterstützung abzielenden Ideen müssen koordiniert und einer einheitlichen Entscheidungsvorbereitung zugeführt werden. Vorhandene Projektstudien müssen einem Priorisierungsverfahren unterzogen werden (vgl. [SoKr90], S. 20). Neben den üblichen Maßgrößen wie *Risiko* und *Nutzen* bringt Krcmar noch die Größen *Project-Strategy-Fit* und *Project-Technology-Fit* in die Bewertung des IV-Maßnahmenportfolios mit ein (vgl. [Krc97], S.255).

- **Projekt-Controlling:** Bei der Durchführung von Projekten muß die Planung, Kontrolle und Steuerung von Kosten, Terminen und Leistungen sichergestellt werden.

- **Produkt-Controlling:** Wartung, Pflege und Weiterentwicklung[2] eines erfolgreich implementierten Produktes müssen im Verlauf des weiteren Produktlebenszykluses der Planung, Kontrolle und Steuerung unterzogen werden (vgl. [SoKr90], S. 26).

- **Infrastruktur-Controlling:** Die langfristige Bereitstellung einer IV-Infrastruktur muß unter Berücksichtigung technologischer Entwicklungen, Benutzeranforderungen und Zielvorgaben geplant werden. Der laufende Betrieb ist

[1] Von Krcmar wird der Begriff *Ideen-Controlling* verwendet.

[2] Weiterentwicklung nur in einem Umfang, der kein neues Projekt darstellt.

hinsichtlich Kapazität, Leistungsfähigkeit und einer Kosten- und Nutzenentwicklung zu überwachen (vgl. [Krc97], S. 259ff.). Für eine innerbetriebliche Leistungsverrechnung[3] ist eine Systematisierung der von Benutzern in Anspruch genommenen Leistungen auf der Grundlage von Basiskennzahlen[4] notwendig.

Der Erstanwender des hier vorgestellten IV-Controllingsystems definiert IV-Controlling für sich folgendermaßen: *„IV-Controlling steht hier für das effektive und effiziente Planen, Überwachen und Steuern aller Ressourcen (Mensch Material, Maschine, Mittel und Methode) die zur Gestaltung und für den Betrieb von informationstechnologischen Produkten, Netzen und Anwendungen notwendig sind.“*

2.2.2 Wer braucht IV-Controlling?

Wer glaubt, sich mit der Einführung eines IV-Controllingsystems die eierlegende Wollmilchsau in den Stall geholt zu haben, der irrt. Ein IV-Controllingsystem verursacht auch Wartungs- und vor allem Bedienungsaufwand. Die Frage nach der Notwendigkeit, IV-Controlling zu betreiben, ist also durchaus berechtigt.

Der Einsatz eines IV-Controllingsystems, wie es in den folgenden Kapiteln beschrieben wird, lohnt sich erst ab einer bestimmten Problemgröße. In zwei weiterführenden Auswertungen der Untersuchung von Spitta [Spi98a] werden von Spitta, Ellerbrock und Kuhlmann [SEK98] und von Kuhlmann [Kuhl98] Kriterien für die Notwendigkeit von IV-Controlling aufgestellt. Während sich in [SEK98] die Anforderungen nur auf die Anzahl der Beschäftigten im IV-Bereich (≥ 5), den Anteil der Standardsoftware ($< 75\%$) und die Anzahl der Standorte (> 1) beschränken, stellt Kuhlmann in seiner Auswertung einen komplexeren Bedingungskatalog auf, der neben diesen drei Einflußgrößen noch sechs weitere enthält, die in einem bedingten Zusammenhang stehen. Kuhlmann fordert:

1. Das Unternehmen hat mehr als einen Standort.
2. Die Anzahl der DV-Mitarbeiter ist ≥ 5.
3. Die Anzahl der Mitarbeiter für Wartung und Entwicklung ist ≥ 3.
4. Der Anteil der Mitarbeiter für Wartung und Entwicklung am IV-Personal insgesamt ist $\geq 50\%$.

[3] Diese Aussage bezieht sich nur auf die Kosten der IV-Infrastruktur

[4] Die Erfassung von Basiskennzahlen zur innerbetrieblichen Leistungsverrechnung ist nicht Gegenstand des im folgenden vorgestellten IV-Controllingsystems. Die Erfassung von Basiskennzahlen wird ausführlich bei Krüll [Krü96] beschrieben.

5. Der Verhältnis von Eigen- zu Fremdsoftware ist ≥ 1.
6. Die abgefragten Basisdaten werden nicht für überflüssig gehalten.[5]
7. Es ist ein strategisches Problem, die Daten nicht zu erfassen.
8. Die Wartung wird durch eigenes Personal durchgeführt.
9. Die Entwicklung wird durch eigenes Personal durchgeführt.

Eine vollständige Begründung der einzelnen Punkte würde hier zu weit führen. Sie kann bei Kuhlmann (vgl. [Kuhl98], S. 45ff) nachgelesen werden.

Diese 9 Kriterien werden logisch verknüpft und ergeben so einen Anforderungskatalog, der in Abbildung 1 dargestellt ist.

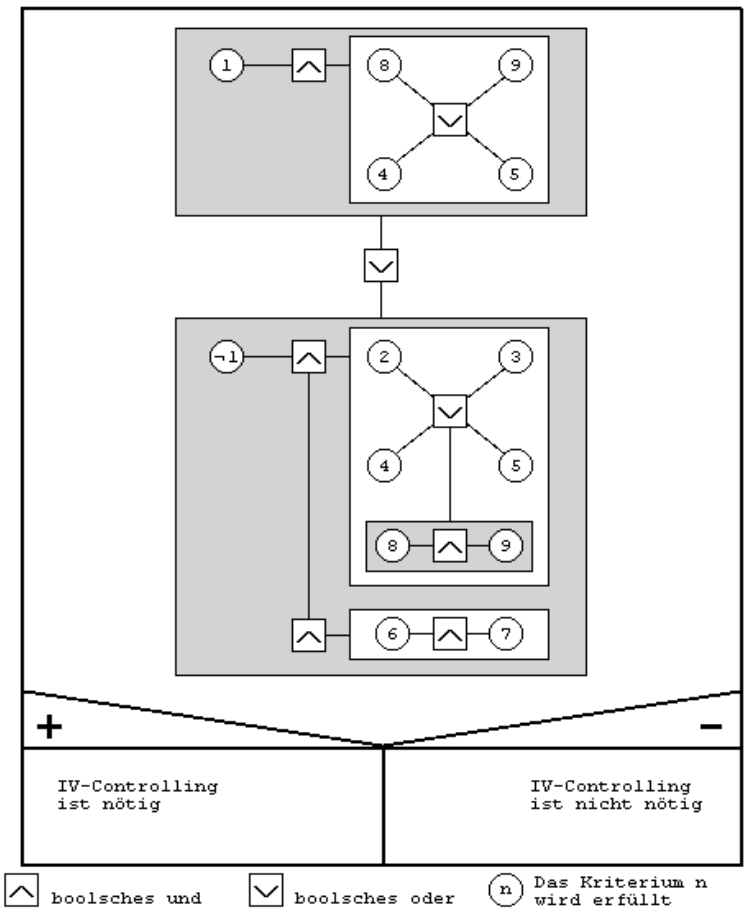

Abbildung 1: Anforderungskatalog für die Notwendigkeit von IV-Controlling.
Quelle: [Kuhl98], S. 48

[5]Diese und die folgende Aussage bezieht sich auf konkrete Fragen in Spittas Untersuchung.

Nach Kuhlmanns Katalog ergab sich in der von Spitta erhobenen Stichprobe für 54,74% der Unternehmen ein Bedarf für den Einsatz von IV-Controlling. Die weniger restriktiven Anforderungen in [SEK98] ergaben für 84% der Unternehmen einen Bedarf für IV-Controlling.

Daß sich diese beiden Werte, obwohl sie sich auf dieselbe Datenbasis beziehen, so signifikant unterscheiden, braucht nicht weiter zu beunruhigen. Der Anforderungskatalog von Kuhlmann ist möglicherweise gewollt komplex, da es unter anderem auch Ziel seiner Arbeit war, zu zeigen, welche Möglichkeiten statistische Werkzeuge wie SAS, SPSS oder S-Plus bieten, die in einer Datenbank gehaltenen Ergebnisse einer Umfrage auszuwerten.[6]

Wer braucht nun IV-Controlling? Diese Frage läßt sich sicher nicht pauschal beantworten, da ein großer Grenzbereich existiert, in dem von Fall zu Fall entschieden werden muß. Da nach [Gri87] 45-60% und nach [Boe81] sogar 90% des Aufwandes im IV-Bereich auf Personalkosten entfallen, ist die Anzahl der Mitarbeiter sicher einer der entscheidensten Faktoren. Die in [SEK98] aufgezeigten Kriterien können in jedem Fall als Faustregel betrachtet werden. Kritisch betrachtet werden sollte dabei jedoch das Verhältnis zwischen Standard- und Eigensoftware. Ein hoher Anteil an Eigensoftware ist wichtiges Indiz für die Notwendigkeit von IV-Controlling. Im Umkehrschluß darf es aber nicht bedeuten, daß ein sehr großer Anteil an Standardsoftware IV-Controlling überflüssig macht.

Darüber hinaus kann es sinnvoll sein zu betrachten, welche Aufgaben die IV-Abteilung zu erfüllen hat. Der Anteil individueller Dienstleistungen, die nicht in Gemeinkosten untergehen sollten, ist sicherlich ein Indiz für die Notwendigkeit einer unternehmensinternen Leistungsverrechnung, für die ein IV-Controlling die notwendige Grundlage bildet.

Neben diesen sehr konkreten Anforderungen für die Notwendigkeit von IV-Controlling gibt es eine sehr allgemeine Aussage von Dobschütz [Dob95], die auch von Horváth ([Hor96], S. 689) unterstützt wird: *„IV-Controlling wird erst zu einer eigenständigen Funktion im Unternehmen, wenn die Aufgaben des Managements zu komplex, undurchsichtig und risikoträchtig werden. Das IV-Controlling schafft dann die notwendige Transparenz, damit das Management verantwortlich planen, entscheiden und führen kann.“*

2.2.3 Wie steht es um das IV-Controlling

In den vorhergehenden Abschnitten ist gezeigt worden, was unter IV-Controlling zu verstehen ist, und unter welchen Bedingungen ein solches notwendig ist.

[6] Kuhlmanns Zahlen sind nicht die Ergebnisse eines manuellen Auszählens, sondern ergaben sich aus den Berechnungen einer entsprechenden, selbsterstellten SAS-Funktion

Im Gegensatz zur Controllingkonzeption, die sich insgesamt in der Unternehmenspraxis durchgesetzt hat, besteht neben anderen Funktionsbereichen auch im Bereich der betrieblichen Informationssysteme großer Nachholbedarf (vgl. [Hor90], S.12).

Die 1997 unter mittelständischen Industrieunternehmen in Ostwestfalen durchgeführte Untersuchung von Spitta [Spi98a] zeigt, daß IV-Controlling praktisch nicht betrieben wird. Ein Problembewußtsein ist in den Unternehmen zwar vorhanden, die Umsetzung erfolgt jedoch nicht oder nur unzureichend (vgl. [Spi98a], S. 430). Die Ergebnisse dieser Untersuchung bestätigen die Erkenntnisse einer empirischen Untersuchung von Krcmar. Nach dieser Untersuchung ist der Implementierungsstand in der Industrie unterdurchschnittlich. Hier behaupten 11,4% der Unternehmen von sich, *„jahrelange Erfahrungen“* mit IV-Controlling zu haben. Etwas besser sieht es bei Versicherungen (22,2%), Unternehmensberatungen (17,4%) und Banken (15,8%) aus (vgl. [Krc92a], S. 13). Diese Abweichung läßt sich dadurch erklären, daß in diesen Branchen schon sehr früh IV-Systeme eingesetzt wurden.

Obwohl offensichtlich Bedarf für IV-Controllingsysteme besteht, werden auf dem Softwaremarkt keine entsprechenden ganzheitlichen Systeme angeboten. Auf dem Softwaremarkt sind allenfalls Teilsysteme erhältlich. Vorwiegend in Großunternehmen ([SoKr90]; [KrFr91]; [Mol94]) finden sich unternehmensspezifische Lösungen.

Es besteht demnach durchaus Bedarf, ein IV-Controllingsystem zu konzipieren, das nicht nur als Individuallösung für ein einzelnes Unternehmen, sondern als vielfältig einsetzbare Lösung dienen kann.

3 Rahmenbedingungen und Anforderungen

3.1 Die Situation des Unternehmens

Der Entwurf dieses IV-Controllingsystems erfolgt in enger Zusammenarbeit mit einem Unternehmen des gehobenen Mittelstandes, das ein entsprechendes System auf Grundlage dieser Arbeit implementieren lassen möchte. Trotzdem soll das Konzept eher für einen universellen als für einen individuellen Einsatz stehen. Der Einfluß des Erstanwenders ist natürlich immer präsent. An vielen Stellen ist diese Situation begrüßenswert, an anderen Stellen bringt sie allerdings auch Probleme mit sich (siehe hierzu auch Kapitel 6).

Es ist daher sinnvoll, an dieser Stelle einen kurzen Einblick in die derzeitige Unternehmenssituation zu geben, in der ein Bedarf für ein IV-Controllingsystem besteht. Dadurch dürften einige Entwurfsentscheidungen klarer werden. Eine Abschätzung des potentiellen Einsatzbereiches ist mit diesem Hintergrundwissen ebenfalls einfacher.

In der DV-Abteilung[7] des Unternehmens am Stammsitz arbeiten rund 200 Mitarbeiter. In den fünf weiteren Werken im Inland arbeiten je 5-10 Mitarbeiter, die jedoch - bis auf grobe Abstimmungen - unabhängig von der DV-Abteilung des Stammsitzes arbeiten.[8]

Neben den klassischen Gebieten einer DV-Abteilung gehören auch das Druck- und Verteilzentrum und der Bereich Telekommunikationstechnik zur DV-Abteilung.

Derzeit sind ca. 90 Systeme und 20 Projekte[9] zu betreuen. Das größte dieser Projekte ist - neben den derzeit überall präsenten *Jahr 2000* und *Euro* Projekten - die unternehmensweite Einführung des SAP R/3 Systems. R/3 soll das derzeit teilweise eingesetzte R/2 und viele selbstentwickelte Systeme ablösen.

Der Einsatz von SAP Software hinterläßt an einigen Stellen beim Entwurf des Datenmodells seine Spuren. Einige Bezeichnungen und Vereinbarungen aus der SAP Welt werden auch in diesem Datenmodell übernommen, um nicht eine weitere Begriffswelt und Vereinbarungen einzuführen, die zu Verwechslungen und einer mangelnden Akzeptanz des Systems führen könnten.

Neben der Betreuung von *Systemen* und *Projekten* bietet die Abteilung noch unterschiedliche *Dienstleistungen* an. Dazu zählen beispielsweise das Aufstellen von

[7] Die DV-Abteilung heißt intern *Organisation Datenverarbeitung*

[8] Zusätzlich arbeiten noch zahlreiche externe Berater und Entwickler im Hause

[9] Ein Vorhaben, für das mehr als als 50 Arbeitstage veranschlagt werden, wird im Hause als *Projekt* eingestuft. Kleinere Vorhaben werden beispielsweise als Systemerweiterung bezeichnet

Telekommunikationseinrichtungen, Druckereiaufträge oder die Bereitstellung eines Internetzugangs.

Ein zentrales Problem im IV-Controlling ist die Planung und Erfassung des Personalaufwands, insbesondere bedarf es der Zustimmung durch den Betriebsrat (vgl. [Spi96a]). Damit ein solches System verwertbare Daten liefert, ist eine Akzeptanz bei den Mitarbeitern notwendig. Bei dem Erstanwender findet man diesbezüglich eine gute Ausgangsposition. Ein eigenes Zeiterfassungssystem, das nun abgelöst werden soll, befindet sich seit längerer Zeit im Echtbetrieb und wird von fast allen Mitarbeitern genutzt.

Eine innerbetriebliche Leistungsverrechnung zwischen Kostenstellen wird über die Innenaufträge[10] der SAP Software abgewickelt. Ein IV-Controlling wird derzeit in „Handarbeit“ betrieben. Aus unterschiedlichen Datenquellen (z.B. Ausdrucke des BAB aus dem R/2-System) werden Zahlen teilweise manuell aggregiert und in EXCEL-Arbeitsmappen aufbereitet. Dadurch ergeben sich erhebliche Probleme:

1. Die Daten sind zum Zeitpunkt der Präsentation vielfach bereits veraltet.
2. Es existieren zahlreiche Quellen für Übertragungsfehler.
3. Die Vollständigkeit der erfaßten Daten ist nicht überprüfbar.
4. Für unterschiedliche Sichten auf die Daten (z.B. Personalstunden pro Kostenstelle, pro Projekt oder pro Mitarbeiter) müssen die gleichen Daten mehrfach erfaßt werden.
5. Ist-Daten können den Plan-Daten nur ungenügend gegenüber gestellt werden.

Der Wandel von der IV-Abteilung zum IV-Profitcenter, wie sie in vielen Unternehmen auf der Tagesordnung steht (vgl. [Kar97]), ist derzeit zwar nicht vordringliches Ziel, aber vereinzelte Gedanken in dieser Richtung bestehen. Daher ist es wichtig, eine möglichst genaue Kostenverrechnung und -kontrolle sicherzustellen. Die DV-Abteilung sieht sich bereits als Anbieter von Waren, wie Hard- und Software und Dienstleistungen. Ein Wettbewerb mit externen Anbietern besteht in einigen Bereichen bereits heute - wie z.B. die Realisation eines Auftrittes im WWW.

Das Unternehmen erfüllt demnach in jedem Fall die in Abschnitt 2.2.2 dargestellten Anforderungen für die Notwendigkeit, IV-Controlling zu betreiben.

[10] Da sich diese Bezeichnung auch bei SAP von Release zu Release ändert, wird im folgenden nur noch die einfachere Bezeichnung *Auftrag* verwendet

3.2 Anforderungen

Basierend auf der in Abschnitt 2.2 skizzierten Vorstellung von IV-Controlling und der Kenntnis der Unternehmenssituation werden im folgenden Anforderungen aufgestellt, die das IV-Controllingsystem erfüllen muß. Die Anforderungen werden an dieser Stelle noch sehr allgemein formuliert und bei der Beschreibung des Datenmodells in Kapitel 5 konkretisiert, damit der Zusammenhang zwischen Anforderung und Realisierung einfacher nachvollziehbar ist.

Das System soll primär das operative Controlling unterstützen und sekundär eine grobe Hilfestellung für das strategische Controlling bieten.

Die Basis des Systems bildet eine relationale Datenbank, für die es ein den Anforderungen entsprechendes Datenmodell zu entwerfen gilt. Auf diese Datenbank setzt später eine Auswertungskomponente auf, die das eigentliche Informationssystem bildet.

Darüber hinaus gibt es noch eine weitere, zweistufige Anforderung, die gewährleisten soll, daß dieser Systementwurf nicht lediglich als Individuallösung implementiert wird. Das System soll nicht unternehmensspezifisch, sondern so variabel, offen und modular ausgelegt sein, daß es auch in anderen Unternehmen als IV-Controllingsystem eingesetzt werden kann. In einem zweiten Schritt sollte es später möglich sein, dieses System auch in anderen Unternehmens*bereichen* einzusetzen.

Der Wandel von einem IV-Controllingsystem zu einem Controllingsystem für beispielsweise die Arbeit in einer Marketing Abteilung erscheint nur auf den ersten Blick radikal. Auch hier wird an Projekten gearbeitet und es werden ständige Vorhaben betreut. Bei genauerer Betrachtung fällt auf, daß in vielen Fällen die wesentlichen Unterschiede zwischen Unternehmensbereichen in Bezeichnungen liegen, die im fertigen System nur auf der obersten Ebene, der Benutzerschnittstelle relevant sind. Eine Anpassung dieser Komponente ist relativ problemlos möglich.

3.2.1 Planen

Ein wesentlicher Bestandteil des Controllings ist die Planung. Daher müssen für alle Objekte, für die später Ist-Daten erfaßt werden, auch Plandaten verarbeitet werden können. Planbar müssen auch solche Daten sein, für die später abgeleitete Ist-Daten errechnet werden, wie z.B. die aggregierte Arbeitsleistung des Personals einer Kostenstelle.

3.2.2 Daten sammeln

Das Aufstellen von Plänen macht nur Sinn, wenn es auch entsprechende Ist-Daten gibt, mit denen die Plandaten verglichen werden können. Diese Daten müssen nicht nur gesammelt, sondern auch Aufgaben eindeutig zugeordnet werden.

Problematisch bei der Ist-Datenbeschaffung ist der Personalaufwand, genauer die Zurechnung des Personalaufwandes zu bestimmten Aufgaben. Um die Korrektheit dieser Daten zu gewährleisten, bedarf die Gestaltung der Erfassungsmaske für diese Daten besonderer Sorgfalt (vgl. [Spi96a]). Vom Datenmodell her muß es möglich sein, individuelle Wünsche und Gewohnheiten der späteren Benutzer bei der Aufwandserfassung zu berücksichtigen.

Das System soll nicht nur Arbeitszeit in einer fest vorgegebenen Zeiteinheit (z.B. Stunden), sondern ganz allgemein *Leistungen* erfassen und zuordnen können.

Alle weiteren Daten, wie der Aufwand für Fremdbezug von Lieferungen und Leistungen, werden über entsprechende Schnittstellen zum Rechnungswesen geliefert (siehe hierzu auch Abschnitt 3.2.4).

3.2.3 Kontrolle, Auswertung und Prognose

Die derzeit schon zahlreich existierenden Übersichten in Form von manuell gepflegten EXCEL-Tabellen bilden eine maßgebliche Anforderung für die Auswertungs- und Informationsfunktion des Systems. Alle bisher in dieser Form existierenden Auswertungen sollen durch automatisch erzeugte Berichte ersetzt werden. Die bestehenden Übersichten geben vor, welche *Fragen* der Unternehmens- bzw. Abteilungsführung das System (mindestens) beantworten können muß. Bis auf einige kleinere Ausnahmen, in denen ein manuell erzeugter Bericht effizienter ist, sind das die Fragen nach:

- Summe aller Kosten pro (Teil-)Projekt,[11] ggf. aufgegliedert nach Eigenleistung, Fremdbezug, Leistungen anderer Unternehmensbereiche
- Summe aller Personalstunden, -jahre oder -kosten pro (Teil-)Projekt, ggf. aufgeteilt nach Werken, Kostenstellen, Organisationseinheiten
- Summe aller Personalstunden, -jahre oder -kosten pro (Teil-)Projekt, ggf. aufgeteilt nach Eigen- und Fremdpersonal
- Aufwand aller Kostenstellen oder Organisationseinheiten für ein (Teil-)Projekt oder System

[11] Die Bezeichnungen *Projekt* und *Teilprojekt* werden hier ganz allgemein verwendet. Eine genaue Definition dieser Begriffe erfolgt in Abschnitt 5.1

- Aufwand (in Geldeinheiten, Personaltagen oder Personaljahren) einer Kostenstelle oder Organisationseinheit für alle (Teil-)Projekte und Systeme
- Summe aller Fremdbezüge pro (Teil-)Projekt oder System
- geplanter Fremdbezug pro (Teil-)Projekt oder System
- tatsächliche Ausgaben pro (Teil-)Projekt oder System
- Auftragsobligo insgesamt, für ein (Teil-)Projekt, ein System, eine Investition oder einen Lieferanten
- Plan/Ist Vergleich der Personalstunden pro Kostenstelle, (Teil-)Projekt oder System auf Monats-, Quartals-, Kalenderjahres-, Geschäftsjahresebene
- relativer Anteil der Arbeit einer Kostenstelle oder Organisationseinheit für Konzeption, Entwicklung, Wartung und Betreuung
- relativer Anteil der Personalkosten eines (Teil-)Projektes für Entwicklung, Wartung und Betreuung ggf. aufgegliedert nach Kostenstellen, Organisationseinheiten, Eigen- und Fremdpersonal
- relativer Anteil Personalkosten einer Kostenstelle, Organisationseinheit oder eines Mitarbeiters für Arbeit an Projekten, Systemen oder Dienstleistungen
- Trendprognosen für Aufwand nach (Teil-)Projekten und Systemen

Als weiteres Auswertungsergebnis werden Daten für die innerbetriebliche Leistungsverrechnung benötigt. Informationen über Kosten für Serviceleistungen, die Betreuung von Systemen und von anderen Kostenstellen in Auftrag gegebene Projekte sollen auf Grundlage der vom IV-Controllingsystem bereitgestellten Daten automatisch über eine Schnittstelle an das Rechnungswesen weitergegeben und verrechnet werden können.

Abstrahiert man von diesen konkreten Anforderungen, dann ergibt sich eine Menge von notwendigen Beziehungen zwischen den verschiedenen Problemgrößen, die Auswertungen wie die oben geforderten ermöglichen:

- Arbeitsleistungen müssen sich müssen sich einem (Teil-)Projekt oder einem System zuordnen lassen
- Die Herkunft der Leistungserstellung muß bis zum leistenden Mitarbeiter nachvollziehbar sein
- Mitarbeiter sind Teil einer kostenrechnerischen und organisatorischen Unternehmensstruktur

- Fremdbezug ist für eindeutige (Teil-)Projekte oder Systeme bestimmt
- Rechnungen lassen sich ebenso wie Bestellungen einem bestimmten (Teil-)Projekte oder Systeme zuordnen
- Plandaten stehen in Beziehung zu Ist-Daten

Diese Zusammenhänge müssen im Datenmodell berücksichtigt werden, um die gewünschten Auswertungen zu ermöglichen.

3.2.4 Schnittstellen

Um wichtige Stammdaten wie z.B. die Personaldaten nicht mehrfach zu erfassen und pflegen zu müssen, muß sich das IV-Controllingsystem in die bestehende „Datenwelt“ des Unternehmens einbetten lassen. Einige Teile des in Kapitel 5 beschriebenen Datenmodells werden nicht in der zentralen Datenbank des Systems gehalten, sondern über Schnittstellen von anderen Systemen zur Verfügung gestellt.

Objekte des Datenmodells werden nur insoweit betrachtet, als es für die Problematik des IV-Controllingsystems von Interesse ist. Für dieses System ist die Sozialversicherungsnummer eines Mitarbeiters unwichtig. In der Personalverwaltung, in der die Stammdaten des Mitarbeiters gepflegt werden, wird dieses Attribut mit Sicherheit von Bedeutung sein. Objekte, deren Daten in anderen Systemen gepflegt werden, werden im Sinne eines Unternehmensdatenmodells unvollständig beschrieben. In Kapitel 5.9 wird auf diese Objekte noch genauer eingegangen.

4 Darstellungsmittel

Dieses Kapitel stellt eine kurze, aber notwendige Unterbrechung auf dem Weg vom Problem zur Beschreibung des endgültigen Systems dar. Bevor man *über* etwas redet, ist es sinnvoll, sich darauf zu einigen, *wie* man über entsprechende Dinge redet, damit zum einen Mißverständnisse vermieden und zum anderen der Leser sich später ganz dem Inhalt widmen kann.

4.1 Darstellung des Datenmodells

Im folgenden Kapitel wird das IV-Controllingsystem bis herunter auf eine sehr technische Ebene beschrieben. Als Autor steht man an dieser Stelle vor einem Problem: Eine rein verbale Beschreibung von Eigenschaften und Zusammenhängen ist zwar leicht lesbar, wird aber sehr schnell unübersichtlich. Eine formale, an der Syntax einer Sprache wie beispielsweise SQL orientierte Beschreibung ist zwar wesentlich kompakter, verlangt dem Leser aber auch erhebliche Konzentration und ein gewisses Grundverständnis einer solchen Sprache ab. Einen Ausweg aus dieser Misere bietet die Mischung beider Stile. Nach einer an den *Leser* gerichteten verbalen Beschreibung, folgt eine formalere, eher an die *Maschine* gerichtete Beschreibung eines Datenobjektes. Die formale Beschreibung hält sich dabei an ein fest vorgegebenes Schema. Diese beiden Formen der Beschreibung wechseln sich ab. Um die Übersicht über die Zusammenhänge zu behalten, sollte das Modell *schrittweise verfeinernd* präsentiert werden.

Wer sich jetzt an den von Donald E. Knuth propagierten Stil des *Literate Programming* [Knu84] erinnert fühlt, liegt genau richtig. Weitere Gedanken in dieser Richtung würden hier zu weit vom eigentlichen Thema ablenken. Ein paar Zeilen zu den Möglichkeiten einer Anwendung dieses Stils auf den Datenmodellierungsprozeß sind aber durchaus berechtigt und finden sich in Abschnitt 4.2.

Eine formale Spezifikationsmethode für Informationssysteme wird in [Mor98] dargestellt. In Anlehnung an diese Methode wird in dieser Arbeit eine Notation verwendet, die sich - ebenso wie die Notation in [Mor98] - an die Spezifikationssprache TROLL (vgl. [Saa93], Kap. 3.1) anlehnt.

Neben den eigentlichen *Objekten* und ihren *Attributen* müssen bei der Beschreibung auch deren *Beziehungen* zu anderen Objekten, *Integritätsbedingungen* und *Wertebereiche* der Attribute berücksichtigt werden.

4.1.1 Objekte

Um eine genaue Abstimmung mit dem Anwender und eine präzise Vorgabe für die Implementierung sicherzustellen, sind zur Beschreibung eines Objektes des Datenmodells Angaben zu folgenden Punkten notwendig:

- Name
- Kurzbeschreibung
- Attribute mit Namen und Typen
- Integritätsbedingungen
- Beziehungen zu anderen Objekten
- Primär- und Fremdschlüssel
- Mußwerte

Der Objektname wird dabei in *Kursivschrift* angegeben.

4.1.2 Verweise

Ein Problem ergibt sich durch die Linearität des Textes. Die einzelnen Objekte des Datenmodells können nicht als auf einer Perlenschnur aufgereiht betrachtet und somit in einer bestimmten Reihenfolge beschrieben werden. So kann es vorkommen, daß Beziehungen zu Objekten bestehen, die dem Leser erst viel später präsentiert werden. Eine gesammelte Beschreibung aller Beziehungen nach der Beschreibung aller Objekte dürfte die Übersichtlichkeit auch nicht erhöhen. Daher werden alle Beziehungen an Ort und Stelle aufgezeigt. Bei Beziehungen zu Objekten, die nicht im selben Abschnitt beschrieben werden, erfolgt ein *Verweis* auf den Abschnitt, in dem das Objekt beschrieben wird. Der Leser kann entsprechende Stellen getrost mit der Einstellung überfliegen: *„Das brauche ich jetzt noch nicht zu verstehen. Das wird später erklärt. Ich behalte es aber schon mal im Hinterkopf.“*

Ähnlich verhält es sich mit nicht elementaren Datentypen[12] von Attributen. Da es sinnvoll ist, bestimmte, selbstdefinierte Datentypen mehrfach zu verwenden (siehe [Spi96b]) und eine Definition eines bestimmten Datentyps auch nur an einer Stelle

[12]Der Terminus *elementarer Datentyp* wird hier etwas großzügig verwendet. Gemeint sind Datentypen, die in praktisch jedem System zur Verfügung stehen wie `Boolean, Integer, Real, Char, String, Date...`

im Text erfolgen sollte, wird es Situationen geben, an denen nur ein Verweis auf ein anderes Kapitel anstelle einer Definition zu finden ist.

Hier besteht die Möglichkeit, alle Datentypen zentral in einem Abschnitt zu beschreiben. Auch in diesem Fall müßte man blättern, wenn man nach der genauen Definition eines Datentyps sucht. Da es für das Verständnis bestimmter Zusammenhänge hilfreich sein kann, die genaue Definition eines Datentyps zu kennen, ist es sinnvoll, diese Definition auch an der Stelle zu plazieren, an der sie benötigt wird: unmittelbar vor der *ersten Verwendung*. Bei allen weiteren Verwendungen erfolgt wieder ein Verweis auf den Abschnitt, in dem der Datentyp definiert wurde.

4.1.3 Datentypen

Die Bezeichnungen der verwendbaren Datentypen hängen sehr von dem gewählten Datenbanksystem ab. Die hier verwendeten Datentypen lehnen sich an Informix und MS-Access Bezeichnungen an. Entsprechende Typen lassen sich in jedem Datenbanksystem finden.

- INTEGER: Ganzzahl
- REAL: Zahl mit Nachkommastellen
- CHAR: Zeichen
- STRING: Zeichenkette
- MEMO: kurzer Text
- DATE: Datum
- CURRENCY: Währungsbetrag
- BOOLEAN: wahrheitswertig

Eine weitergehende Diskussion über die Verwendung spezieller Varianten dieser Typen (z.B. LongInteger) sollte erst geführt werden, wenn entschieden ist, welches Datenbanksystem mit welchen Möglichkeiten tatsächlich zur Implementierung verwendet wird.

Neben diesen „elementaren“ Datentypen ist es in vielen Fällen sinnvoll, eigene Datentypen zu verwenden. Zur Definition eines solchen eigenen Datentyps muß die Menge der zulässigen Werte definiert werden, beispielsweise durch eine Aufzählung. Eigene Datentypen werden durch Spitzklammern gekennzeichnet und folgendermaßen definiert:

⟨BEARBEITUNGS-STATUS⟩ := {`offen; in Arbeit; abgeschlossen`}

Datentypen sind durch Großbuchstaben gekennzeichnet.

4.1.4 Attribute

Attribute werden durch eine Sans-Serif Schrift kenntlich gemacht. Hat ein Attribut die Funktion eines Schlüssels, wird dies durch das angehängte #-Symbol kenntlich gemacht. Primärschlüssel sind zusätzlich unterstrichen. Die Personalnummer als Primärschlüssel würde demnach folgendermaßen dargestellt: Personal#. Als Fremdschlüssel in einem anderen Objekt Personal#.

Soll die Eingabe eines Attributwertes beim Erstellen einer Instanz dieses Objektes zwingend erforderlich sein, so bekommt das Attribut den Zusatz **(M)** für *Mußwert*.

Bei der Verwendung von Fremdschlüsseln kann es durchaus sinnvoll sein, von der Bezeichnung des Attributes in seiner Funktion als Primärschlüssel abzuweichen, um die inhaltliche Bedeutung des Fremdschlüssels deutlich zu machen. So kann die Nummer einer Kostenstelle als Fremdschlüssel für eine *anfordernde Kostenstelle* oder eine *leistende Kostenstelle* verwendet werden. Um die Beziehung des Fremdschlüssels zu seinem Bezugsobjekt deutlich zu machen, wird ihm statt eines Datentyps der Name des referenzierten Primärschlüssels in Spitzklammern zugewiesen.

AnforderndeKst# → ⟨KOSTENSTELLEN#⟩

Die Definition des Attributes Kostenstellen# als Primärschlüssel im Objekt *Kostenstelle* bleibt davon unberührt (z.B. INTEGER).

Aus Gründen der Übersichtlichkeit sollten bei der Namensgebung durch standardisierte Abkürzungen und Vereinbarungen bei der Benennung der Attribute Attributklassen geschaffen werden [Spi96b]:

- Bei Attributen, deren Bezeichnung auf ...# enden, handelt es sich um Schlüssel
- Attribute, deren Bezeichnung auf ...Lfd# enden, beinhalten einen automatisch vergebenen Schlüssel, auf dessen Werte der Benutzer keinen Einfluß hat
- Attributbezeichnungen, die auf ...Dat enden, beinhalten ein Datum
- Das Attribut KurzBez sollte für eine Kurzbezeichnung (möglichst eine Kombination von wenigen Buchstaben) verwendet werden, die beispielsweise in Masken oder Berichten die Übersichtlichkeit erhöht. Eine Langbezeichnung LangBez ist für Stellen gedacht, an denen mehr Platz zur Anzeige zur Verfügung steht oder eine ausführlichere Bezeichnung für Außenstehende benötigt wird.

4.1.5 Integritätsbedingungen

Integritätsbedingungen werden in der Notation der *Prädikatenlogik erster Stufe* angegeben, nachdem sie zuvor verbal formuliert wurden.

Prädikate sind Aussagen, die entweder `wahr` oder `falsch` sind, beispielsweise $3 > 2$ oder $x \in \{\texttt{A}; \texttt{B}; \texttt{C}\}$. Diese einfachen Prädikate können durch Junktoren zu komplexeren Aussagen kombiniert werden:

$\neg$	*nicht*	Negation
$\bigwedge$	*und*	Konjunktion
$\bigvee$	*oder*	Adjunktion
$\Rightarrow$	*wenn...dann*	Implikation
$\Leftrightarrow$	*genau dann, wenn*	Äquivalenz

Durch die Verwendung von Quantoren lassen sich noch weitergehende Aussagen formalisieren:

$\forall$	*für alle*	Allquantor
$\exists$	*es gibt wenigstens ein*	Existenzquantor
$\exists_1$	*es gibt genau ein*	eingeschränkter Existenzquantor

Die Aussage:

> *„Für zwei unterschiedliche Mitarbeiter m1 und m2 aus der Menge aller Mitarbeiter gilt: Wenn m1 und m2 unterschiedlich sind, dann sind auch ihre Personalnummern unterschiedlich.“*

würde in der Notation der Prädikatenlogik erster Ordnung wie folgt dargestellt:

$$\forall\ \text{m1, m2} \in \mathit{Mitarbeiter} \bullet \text{m1} \neq \text{m2} \Rightarrow \text{m1.}\mathsf{Personal\#} \neq \text{m2.}\mathsf{Personal\#}$$

Das $\bullet$-Symbol trennt dabei die Deklaration der Gegenstandsvariablen m1 und m2 von dem zugehörigen Prädikat m1 $\neq$ m2. Es ist zu lesen als *„...für die gilt...“* (vgl. [Mor98], S. 17ff)

4.1.6 Beziehungen

Beziehungen zwischen Objekten werden unter Verwendung von Symbolen nach dem IDEF1X-Schema dargestellt, das auch von graphischen Datenmodellierungstools wie ERWinSQL und Visio verwendet wird.

Dabei stehen

- *Objekt1* $\diamond$——• *Objekt2*
  für eine 0,1 zu 0,n Beziehung zwischen *Objekt1* und *Objekt2*
- *Objekt1* $\diamond$——$\bullet_P$ *Objekt2*
  für eine 0,1 zu 1,n Beziehung zwischen *Objekt1* und *Objekt2*
- *Objekt1* $\diamond$——$\bullet_Z$ *Objekt2*
  für eine 0,1 zu 0,1 Beziehung zwischen *Objekt1* und *Objekt2*
- *Objekt1* ——• *Objekt2*
  für eine 1 zu 0,n Beziehung zwischen *Objekt1* und *Objekt2*
- *Objekt1* ——$\bullet_P$ *Objekt2*
  für eine 1 zu 1,n Beziehung zwischen *Objekt1* und *Objekt2*
- *Objekt1* ——$\bullet_Z$ *Objekt2*
  für eine 1 zu 0,1 Beziehung zwischen *Objekt1* und *Objekt2*

Semantisch bedeutet die Beziehung *Auftrag* —— $\bullet_P$*Auftragsposition*, daß es zu einem *Auftrag* mehrere *Auftragsposition* geben kann, jedoch mindestens eine geben muß. Zu jeder *Auftragsposition* gehört genau ein *Auftrag*.

4.1.7 Beispiel

Als Beispiel wird ein Datenobjekt betrachtet, mit dem ein Mitarbeiter des Unternehmens abgebildet wird. Neben einer den Mitarbeiter eindeutig identifizierenden Personalnummer ist mindestens sein Nachname bekannt. Sein Status muß ebenfalls bekannt sein. Ist dieser Status nicht `Pensionär` oder `freigestellt`, dann ist ihm ein Telefonanschluß zugeordnet, unter dem er zu erreichen ist. Andernfalls bekommt er keine Durchwahl zugeordnet. Ein Mitarbeiter muß genau einer Kostenstelle zugeordnet sein; einer Kostenstelle können mehrere Mitarbeiter zugeordnet werden.

Der Status eines Mitarbeiters kann dabei folgende Ausprägungen haben:

⟨PERSONAL-STATUS⟩ := {`angestellt; gewerblich; freigestellt; Pensionär`}

Objekt: *Personal*

Beschreibung:
Angestellter oder gewerblicher Mitarbeiter des Unternehmens

Attribute:

Personalnummer# → INTEGER

Vorname → STRING

Nachname → STRING **(M)**

Status → ⟨PERSONAL-STATUS⟩ **(M)**

Durchwahl → INTEGER

Kostenstelle → ⟨KOSTENSTELLEN#⟩

IntegritĎtsbedingungen:

$\forall$ pers $\in$ *Personal* • pers.Status $\notin$ { `Pensionär; freigestellt`}
$\Leftrightarrow$ pers.Durchwahl $\neq$ NULL

Beziehungen:

Kostenstelle —• *Mitarbeiter*

4.2 Wie die Zukunft aussehen könnte

Dokumentation und Endprodukt sind untereinander vielfach inkonsistent. Kaum ein Entwickler wird die Dokumentation bei jeder Änderung der Software umgehend anpassen und umgekehrt.

Zu einer Dokumentation gehört meist auch eine graphische Komponente. Vielfach bilden erste „Zeichnungen“ auch die Grundlage für ein Datenmodell. Werkzeuge wie ERWinSQL unterstützen die Verbindung von Graphik und Datenmodell. Über Schnittstellen zur Datenbank wird bei Änderungen in der graphischen Darstellung des Datenmodells die Datenbank angepaßt. Umgekehrt lassen sich entsprechende Grafiken nach Änderungen in der Datenbank anpassen (siehe hierzu auch Abschnitt 6.2). Was fehlt, ist eine feste Beziehung zwischen Datenmodell und textueller Dokumentation.

Knuths *Literate Programming* Ansatz [Knu84], bei dem der Quellcode eines Programms und die zugehörige Dokumentation aus demselben Quelldokument erzeugt

werden, läßt sich auch auf die Datenmodellierung anwenden und umgeht dieses Problem. Dokumentation und Endprodukt (egal ob Programm oder Datenmodell) werden so immer auf dem gleichen, aktuellen Stand gehalten.

Benötigt werden dazu zwei Mechanismen: Der erste scannt das Quelldokument nach Definitionen von Objekten, ihren Attributen und Beziehungen und generiert aus diesen Informationen entsprechenden Code für die gewünschte Zielplattform (z.B. SQL Befehle wie `CREATE TABLE...`). Voraussetzung dafür ist, daß entsprechende Definitionen eindeutig identifizierbar, vollständig und in einer vorher vereinbarten Form aufgeschrieben sind.

Der zweite Mechanismus erzeugt aus demselben Quelldokument den Input für eine Textverarbeitungssoftware wie beispielsweise TEX, wobei er eventuelle, für den Inhalt irrelevante Steuerzeichen ausblendet und notwendige Verweise (z.B. *Dieser Datentyp wird in Kapitel 5.2 definiert*) innerhalb des Dokumentes einfügt.

Die im vorhergehenden Abschnitt beschriebene Art des Aufschreibens kommt dieser Vorstellung schon sehr nahe. Die fehlenden Elemente für die gerade angedachte *Literate Database Design* Umgebung sollten sich mit vergleichsweise geringem Aufwand erstellen lassen. Varianten für andere Sprachen wie APL, Pascal, C^{++} oder S-Plus Code existieren bereits. Wer jetzt auf den Geschmack gekommen ist und in dieser Richtung aktiv werden will, findet nützliche Hinweise bei Wolf [Wol98].

5 Datenmodell für das IV-Controllingsystem

Aus der geschilderten Situation des Unternehmens, den Anforderungen und der Vorstellung von IV-Controlling ergeben sich für die Gestaltung eines Datenmodells, das die Grundlage für ein adäquates IV-Controllingsystem bildet, folgende Fragen:

- Wie sieht die zu erbringende Leistung des Unternehmensbereiches aus, in dem das IV-Controllingsystem eingesetzt wird?
- Wer erbringt diese Leistung genau?
- Wer fordert die Leistung an?
- Wie wird der Aufwand für die Leistung verrechnet?
- Wer erbringt sonst noch Leistungen?
- Wie kann das Ganze geplant werden?
- Was beeinflußt das IV-Controllingsystem sonst noch?

Durch grobe Antworten auf diese Fragen lassen sich die notwendigen Teilbereiche des Gesamtmodells identifizieren:

- **Servicearten**: Es werden unterschiedliche Produkte und Dienstleistungen angeboten und erbracht
- **Unternehmensstruktur**: Die Leistung wird in einem Unternehmen erbracht, das in verschiedene Teilbereiche aufgeteilt ist
- **Personal und Organisation**: Für diese Unternehmensbereiche arbeiten Mitarbeiter, denen Aufgaben und Arbeitsbereiche zugewiesen werden
- **Aufträge**: Leistungen werden erbracht, weil es einen Auftraggeber gibt, der diese Leistungen angefordert hat
- **Leistungskontierung**: Erbrachte Leistungen müssen den betreffenden Aufträgen zugeordnet werden
- **Fremdbezug**: Neben den eigenen gibt es auch Leistungen Dritter, die in die Gesamtleistung mit eingehen
- **Planung**: Für zahlreiche Komponenten wie Mitarbeiter, Aufträge oder Investitionen müssen Planwerte festgehalten werden
- **Weitere Komponenten**: Es sind Dinge wie Recheneinheiten, Umrechnungsverhältnisse, Preise und Wertebereiche zu organisieren

5.1 Servicearten

Die IV-Abteilung ist als Serviceabteilung zu sehen, die anderen Abteilungen ihre Produkte und Dienste anbietet. Diese *Serviceleistungen*, und insbesondere deren struktureller Zusammenhang, werden in diesem Abschnitt beschrieben und modelliert.

Alle Vorhaben, Tätigkeiten oder Objekte, auf die sich die Arbeit der IV-Abteilung bezieht, werden als *Service*[13] bezeichnet.

Betrachtet man die Serviceleistungen der IV-Abteilung genauer, lassen sich drei (primäre) Service*arten* identifizieren, für die Leistungen erbracht werden:

- **Projekte**: zeitlich begrenzte, einmalige Vorhaben, für die mehr als 50 Arbeitstage eingeplant werden.[14] Projekte werden später vielfach zu *Systemen*. Ein treffendes Beispiel hierfür ist die Einführung von SAP R/3.
- **Systeme**: Eine dauerhafte Einrichtung, die betreut, gewartet und ggf. weiterentwickelt werden muß - beispielsweise ein Vertriebs-Informationssystem
- **Dienstleistungen**: Kleinere Tätigkeiten oder Lieferungen, die häufiger anfallen, die aber weder einem Projekt, noch einem System zugeordnet werden können. Dazu zählen beispielsweise das Aufstellen und Einrichten eines Druckers oder einer Telefonnebenstelle, die Vervielfältigung eines Manuskriptes oder auch die Bereitstellung einer Softwarelizenz.

Ein Service kann demnach von drei unterschiedlichen Arten sein: `Projekt (PJ)`, `System (SY)` oder `Dienstleistung (DL)`.

Da in vielen Fällen eine Zerlegung eines Problems in Teilprobleme notwendig ist, muß es möglich sein, einen Service in mehrere *Teilservices* zu unterteilen. Ein Service muß sich daher einem anderen Service zuordnen lassen. Die Struktur der einzelnen Service-Elemente ist streng hierarchisch angelegt. Daher kann die Beziehung eines Service-Objektes zu seinem Vorgänger durch ein zusätzliches Attribut TeilServiceVon abgebildet werden.

Die Bezeichnungen für die untergeordneten Serviceobjekte richten sich nach Bezeichnungen, die nach DIN genormt sind. Ein Projekt kann in `Teilaufgaben (TA)` zerlegt werden. Diese Teilaufgaben enthalten wiederum `Arbeitspakete (AP)`. Systeme lassen sich in `Teilsysteme (TS)` zerlegen. Auch Teilsysteme werden in `Arbeitspakete (AP)` unterteilt.

[13] Auf die naheliegende Bezeichnung IV-Service wurde bewußt verzichtet, um einen Einsatz in anderen Bereichen nicht schon durch Bezeichnungen im Datenmodell zu erschweren.

[14] Diese Grenze kann je nach Organisationsgröße und Branche variieren.

Die unterste, ausführbare Ebene bildet die `Aktivität (AK)`. Sie beschreibt Tätigkeiten wie *Code schreiben, Modul testen* oder *Handbuch erstellen.*

⟨SERVICE-ART⟩ := {`PJ; SY; DL; TA; TS; AP; AK`}

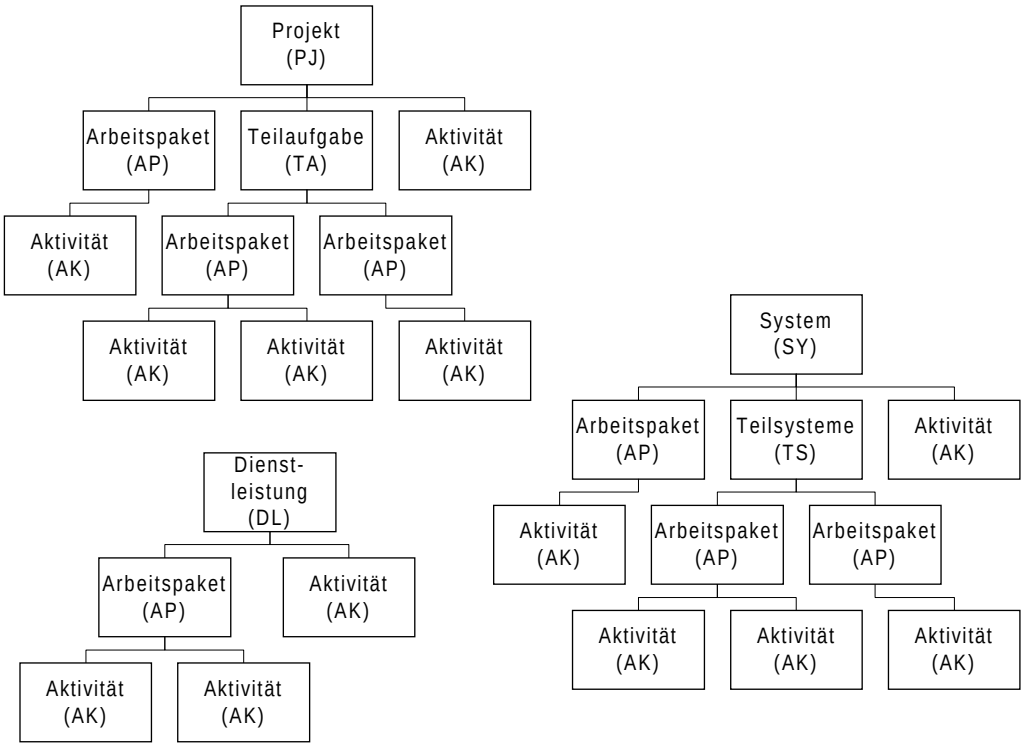

Abbildung 2: Mögliche Strukturen der unterschiedlichen Servicearten

Die Positionen der unterschiedlichen Servicearten in diesem Strukturbaum sind dabei keinesfalls beliebig, sondern unterliegen folgenden Bedingungen:

1. Ein Service, der von der Art `PJ, SY` oder `DL` ist, kann keinen Vorgänger haben.
2. Ein Service von der Art `TA, TS, AP` oder `AK` muß einen Vorgänger haben.
3. Ein Vorgänger kann nie von der Art `AK` sein.
4. Die Vorgänger einer Teilaufgabe `(TA)` oder eines Teilsystems `(TS)` können nur von der Art `PJ` oder `SY` sein.
5. Arbeitspakete können Vorgänger von der Art `PJ, SY, DL, TA` oder `TS` haben.

Durch Variationen dieser Bedingungen und des Datentyps SERVICE-ART läßt sich die maximale Tiefe des Servicestrukturbaumes beeinflussen. Über die maximal zulässige Tiefe kann es unterschiedliche Auffassungen geben (vgl. [Spi96a]). Daher muß sich über diese beiden Parameter eine individuelle Reglementierung realisieren lassen.

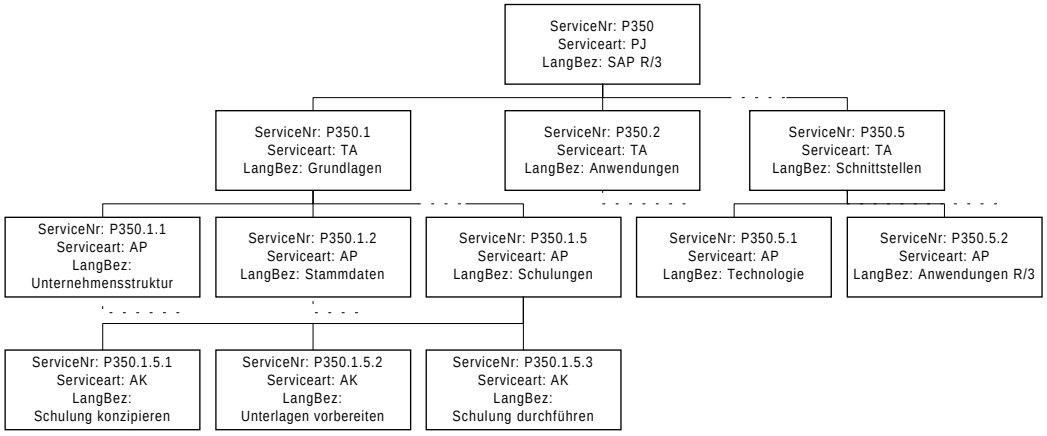

Abbildung 3: Beispiel für mögliche Positionen der unterschiedlichen Servicearten im Servicestrukturbaum eines *Projektes*

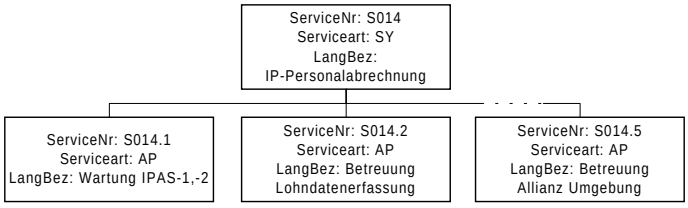

Abbildung 4: Beispiel für mögliche Positionen der unterschiedlichen Servicearten im Servicestrukturbaum eines *Systems*

Ein Service befindet sich in einem Status. Ist noch keine Leistungskontierung auf einen Service oder einen diesem Service untergeordneten Service erfolgt, so hat der Service den Status `offen`. Mit der ersten auf diesen oder einen untergeordneten Service bezogenen Leistungskontierung wechselt der Status auf `in Arbeit`. Wird durch Leistungskontierungen zu einem Service und allen untergeordneten Services signalisiert, daß die Arbeiten an dem entsprechenden Service abgeschlossen sind, tritt der Status `technisch abgeschlossen` ein. Trotzdem muß es noch möglich sein, Buchungen - etwa von Rechnungen über Fremdbezug - vorzunehmen. Nach einiger Zeit tritt der Status `buchhalterisch abgeschlossen` ein, in dem keine Buchungen mehr auf Aufträge dieses Services durchgeführt werden können. Zwischenzeitlich

kann es auch passieren, daß ein Projekt abgebrochen oder ein System außer Betrieb genommen wird. Dann bekommen entsprechende Services den Status `abgebrochen`.

⟨SERVICE-STATUS⟩ := {`offen; in Arbeit; technisch abgeschlossen; buchhalterisch abgeschlossen; abgebrochen`}

Objekt: *Service*

Beschreibung:
Element in der Hierarchie aller Aufgaben und Tätigkeiten einer Abteilung

Attribute:

Service# → STRING

ServiceArt → ⟨SERVICE-ART⟩ **(M)**

TeilServiceVon# → ⟨SERVICE#⟩

KurzBez → STRING **(M)**

LangBez → STRING

Verantwortlich# → ⟨PERSONAL#⟩

Status → ⟨SERVICE-STATUS⟩ **(M)**

IntegritĎtsbedingungen:

1. ∀ serv ∈ *Service* • serv.ServiceArt ∈ {`PJ`; `SY`; `DL`}
   ⇔ serv.TeilServiceVon = NULL
2. ∀ serv ∈ *Service* • serv.ServiceArt ∈ {`TA`; `TS`; `AP`; `AK`}
   ⇔ serv.TeilServiceVon# ≠ NULL
3. ∀ s1, s2 ∈ *Service* • s1.ServiceArt ∈ {`TA`; `TS`; `AP`; `AK`}∧
   s1.TeilserviceVon# = s2.Service# ⇔ s2.ServiceArt ≠ `AK`
4. ∀*s*1, *s*2 ∈ *Service* • s1.ServiceArt ∈ {`TA`; `TS`}∧ s1.TeilserviceVon#
   = s2.Service# ⇔ s2.ServiceArt ∈ {`PJ`; `SY`}
5. ∀ s1, s2 ∈ *Service* • s1.ServiceArt ∈ {`AP`; `AK`}∧ s1.TeilserviceVon#
   = s2.Service# ⇔ s2.ServiceArt ∈ {`PJ`; `SY`; `DL`; `TA`; `TS`}

Beziehungen:

Service ◇——• *Service* (5.1)

Service ——• *MitarbeiterAuftrag* (5.3)

Service —● *LeistungsKontierung* (5.5)

Service ◇—● *Auftrag* (5.4)

Service ◇—● *PlanWert* (5.7.1)

5.2 Unternehmensstruktur

Aus kostenrechnerischer Sicht unterteilt sich ein Unternehmen in mehrere *Werke*. Ein Werk darf dabei nicht unbedingt mit einem Produktionsstandort gleichgesetzt werden. Die Bezeichnung *Werk* stammt aus dem SAP-System und wird übernommen.

Objekt: *Werk*

Beschreibung:
Werk, bzw. Standort des Unternehmens

Attribute:

Werk# → INTEGER

KurzBez → STRING (M)

LangBez → STRING

IntegritĎtsbedingungen:

keine

Beziehungen:

Werk —●$_P$ *Kostenstelle* (5.2)

Werk —● *MitarbeiterKstEinsatz* (5.3)

Werk ◇—● *LeistungsKontierung* (5.5)

Werk ◇—● *PlanWert* (5.7.1)

Werke sind unterteilt in Kostenstellen. Da in verschiedenen Werken gleiche Kostenstellen vorhanden sein können, ist es sinnvoll, als Primärschlüssel einen zusammengesetzten Schlüssel aus Werk# und Kostenstellen# zu bilden.

Kostenstellen können zu sogenannten *Knotenkostenstellen* verdichtet werden. Auch hier muß eine hierarchische Anordnung der Kostenstellen möglich sein. Um Kostenstellen von Knotenkostenstellen unterscheiden zu können, ist ein Attribut KostenstellenArt notwendig, das die Ausprägungen `Kst` für eine einfache Kostenstelle und `KnotenKst` für eine verdichtende Kostenstelle annehmen kann.

⟨KOSTENSTELLEN-ART⟩ := {`Kst; KnotenKst`}

Die jeweiligen Werte für das Attribut KostenstellenArt richten sich danach, ob eine Kostenstelle einen Nachfolger hat oder nicht. Nur wenn ein Nachfolger existiert, handelt es sich um eine Knotenkostenstelle.

In dem Attribut KostenstellenArt wird demnach ein abgeleiteter Wert gespeichert. Normalerweise sollte das vermieden werden, um möglichen Inkonsistenzen vorzubeugen. Seine Bestimmung erfordert allerdings einen vergleichsweise hohen Rechen- bzw. Suchaufwand. Da dieser Wert bei zahlreichen Datenbankabfragen ausgelesen wird, ist ein Attribut KostenstellenArt in diesem Fall sinnvoll, das nur vom System selber, aber nicht vom Benutzer verändert werden kann.

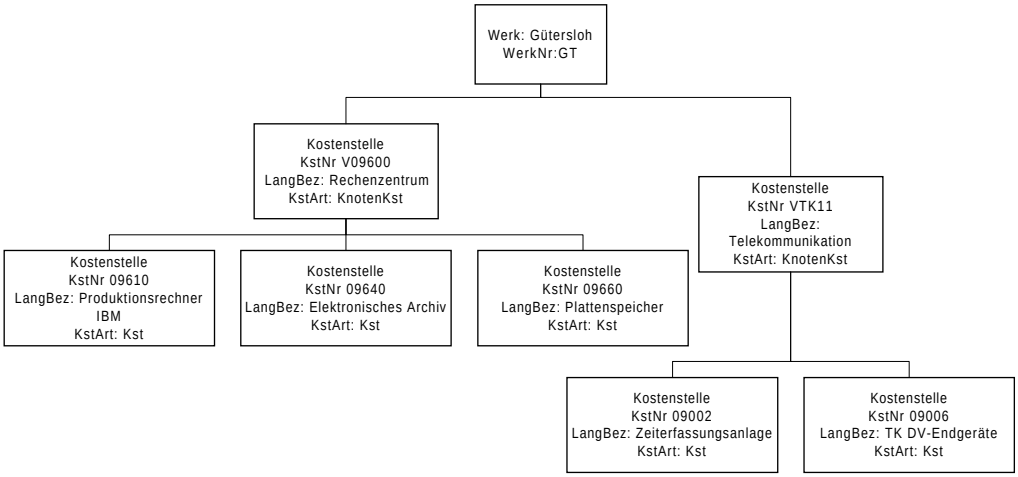

Abbildung 5: Ausschnitt aus der kostenrechnerischen Unternehmensstruktur

Objekt: *Kostenstelle*

Beschreibung:
Kostenstelle laut Kostenstellenplan des Unternehmens

Attribute:

Kostenstellen# → STRING

Werk# → ⟨WERK#⟩

KurzBez → STRING **(M)**

LangBez → STRING

TeilWerkKstVon# → ⟨WERK#⟩

TeilKstVon# → ⟨KOSTENSTELLEN#⟩

KstArt → ⟨KOSTENSTELLEN-ART⟩

Verantwortlich# → ⟨PERSONAL#⟩ **(M)**

Stundensatz → CURRENCY

IntegritĎtsbedingungen:

∀ k1, k2 ∈ *Kostenstelle* •¬∃ k2 • k2.TeilKstVon#
= k1.Kostenstellen# ⇔ k1.KostenstellenArt = `Kst`

Beziehungen:

Kostenstelle ◇——• *Kostenstelle* (5.2)

Kostenstelle •$_P$—— *Werk* (5.2)

Kostenstelle ——• *MitarbeiterKstEinsatz* (5.3)

Kostenstelle ◇——• *LeistungsKontierung* (5.5)

Kostenstelle ◇——• *PlanWert* (5.7.1)

Neben der kostenrechnerischen Sicht des Unternehmens gibt es noch eine organisatorische Sicht des Unternehmens. Die organisatorische Sicht bildet feinere Strukturen ab als die Kostenstellenstruktur. Dabei muß die Organisationssicht nicht unbedingt deckungsgleich mit der Kostenstellen- und Werkstruktur sein.

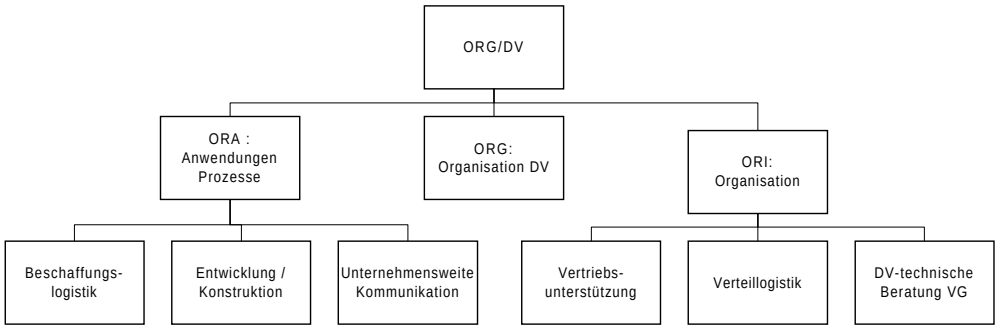

Abbildung 6: Ausschnitt aus der Organisationssicht der Unternehmensstruktur

Objekt: *OrganisationsEinheit*

Beschreibung:
Organisationseinheit eines Unternehmens

Attribute:

OrgEinheit# → INTEGER

TeilEinheitVon# → ⟨ORGEINHEIT#⟩

KurzBez → STRING **(M)**

LangBez → STRING

Verantwortlich# → ⟨PERSONAL#⟩ **(M)**

IntegritĎtsbedingungen:

keine

Beziehungen:

OrganisationsEinheit ◇—● *OrganisationsEinheit* (5.2)

OrganisationsEinheit —● *MitarbeiterOrgEinsatz* (5.3)

OrganisationsEinheit —● *AngebotPosition* (5.4)

OrganisationsEinheit —● *Ziele* (5.7.2)

OrganisationsEinheit ◇—● *PlanWert* (5.7.1)

5.3 Personal

Neben einer den Mitarbeiter eindeutig identifizierenden Personalnummer muß sein Nachname und sein Status bekannt sein.

⟨MITARBEITER-STATUS⟩ := {`angestellt; gewerblich; freigestellt; pensioniert`}

Je nach Branche, Unternehmen oder sogar Unternehmensbereich ist es üblich, den Stundensatz für eine Arbeitsstunde nicht an die leistende Kostenstelle zu koppeln, sondern an den Mitarbeiter. Dadurch können unterschiedliche Qualifikationen berücksichtigt werden (z.B. Junior-Berater/Senior-Berater oder Meister/Geselle). Ob in einer Auswertung nun der Stundensatz der Kostenstelle oder der Stundensatz des Mitarbeiters verwendet wird, muß an anderer Stelle festgelegt werden.

Manche Mitarbeiter arbeiten nicht Vollzeit (`VZ`), stehen also nicht 35 Stunden in der Woche zur Verfügung. Es gibt Klassen von Mitarbeitern mit unterschiedlicher Verfügbarkeit. Denkbar sind beispielsweise Teilzeitverträge über 15 Stunden (`TZ15`) pro Woche, oder über 20 Stunden (`TZ20`). Andere Mitarbeiter können längerfristig gar nicht zur Verfügung stehen (`NV`). Diese Aufzählung ist erweiterbar. Weitere Klassen können über die Umrechnungsdefinitionen (siehe Abschnitt 5.8.2) definiert werden.

⟨VERFGSTD⟩ := {`VZ; TZ15; TZ20; NV`}

Objekt: *Mitarbeiter*

Beschreibung:

Angestellter oder gewerblicher Mitarbeiter des Unternehmens.

Attribute:

Personal# → INTEGER

Vorname → STRING

Nachname → STRING **(M)**

Status → ⟨MITARBEITER-STATUS⟩ **(M)**

Verfügbarkeit → ⟨VERFGSTD⟩ **(M)**

Stundensatz → CURRENCY

IntegritĎtsbedingungen:

keine

Beziehungen:

Mitarbeiter —●$_P$ *MitarbeiterKstEinsatz* (5.3)

Mitarbeiter —●$_P$ *MitarbeiterOrgEinsatz* (5.3)

Mitarbeiter —● *MitarbeiterAuftrag* (5.3)

Mitarbeiter —● *LeistungsKontierung* (5.5)

Mitarbeiter ◇—● *PlanWert* (5.7.1)

Mitarbeiter müssen einer Kostenstelle zugeordnet sein. Diese Zuordnung muß für jeden Zeitpunkt nachvollziehbar sein. Daher ist das Objekt *MitarbeiterKstEinsatz* notwendig, um den Einsatz eines Mitarbeiters für unterschiedliche Zeiträume in unterschiedlichen Kostenstellen dokumentieren zu können. Für jeden Mitarbeiter gibt es genau eine aktuelle[15] Einteilung zu einer Kostenstelle.

[15] Aktuell bedeutet hier: noch kein Endtermin festgelegt. Nach dem SAP Standard ist in diesem Fall der Endtermin der `31.12.9999`.

Objekt: *MitarbeiterKstEinsatz*

Beschreibung:
Zuordnung eines Mitarbeiters zu einer Kostenstelle für einen bestimmten Zeitraum

Attribute:

Personal# → ⟨PERSONAL#⟩

VonDat# → DATE

BisDat# → DATE

Kostenstellen# → ⟨KOSTENSTELLEN#⟩ **(M)**

Werk# → ⟨WERK#⟩ **(M)**

IntegritĎtsbedingungen:

$\forall$ ma $\in$ *Mitarbeiter* $\exists_1$ makein $\in$ *MitarbeiterKstEinsatz* •
makein.Personal# = ma.Personal# $\wedge$ makein.BisDat = `31.12.9999`

Beziehungen:

MitarbeiterKstEinsatz •$_P$—— *Mitarbeiter* (5.3)

MitarbeiterKstEinsatz •—— *Werk* (5.2)

MitarbeiterKstEinsatz •—— *Kostenstelle* (5.2)

Aufgrund der möglichen Unterschiede in Struktur und Gliederungstiefe zwischen Kostenstellen und Organisationseinheiten muß es auch eine Zuordnung der Mitarbeiter zu Organisationseinheiten geben. Auch hier muß es pro Mitarbeiter einen aktuellen Zuordnungssatz geben.

Objekt: *MitarbeiterOrgEinsatz*

Beschreibung:
Zuordnung eines Mitarbeiters zu einer OrganisationsEinheit für einen bestimmten Zeitraum

Attribute:

Personal# → ⟨PERSONAL#⟩

VonDat# → DATE

BisDat# → DATE

OrganisationsEinheit $\rightarrow \langle \text{ORGEINHEIT\#} \rangle$ **(M)**

IntegritĎtsbedingungen:

$\forall$ ma $\in$ *Mitarbeiter* $\exists_1$ maoein $\in$ *MitarbeiterOrgEinsatz* $\bullet$
maoein.Personal# = ma.Personal# $\wedge$ maoein.BisDat = 31.12.9999

Beziehungen:

MitarbeiterOrgEinsatz $\bullet_P$—— *Mitarbeiter* (5.3)

MitarbeiterOrgEinsatz $\bullet$—— *OrganisationsEinheit* (5.2)

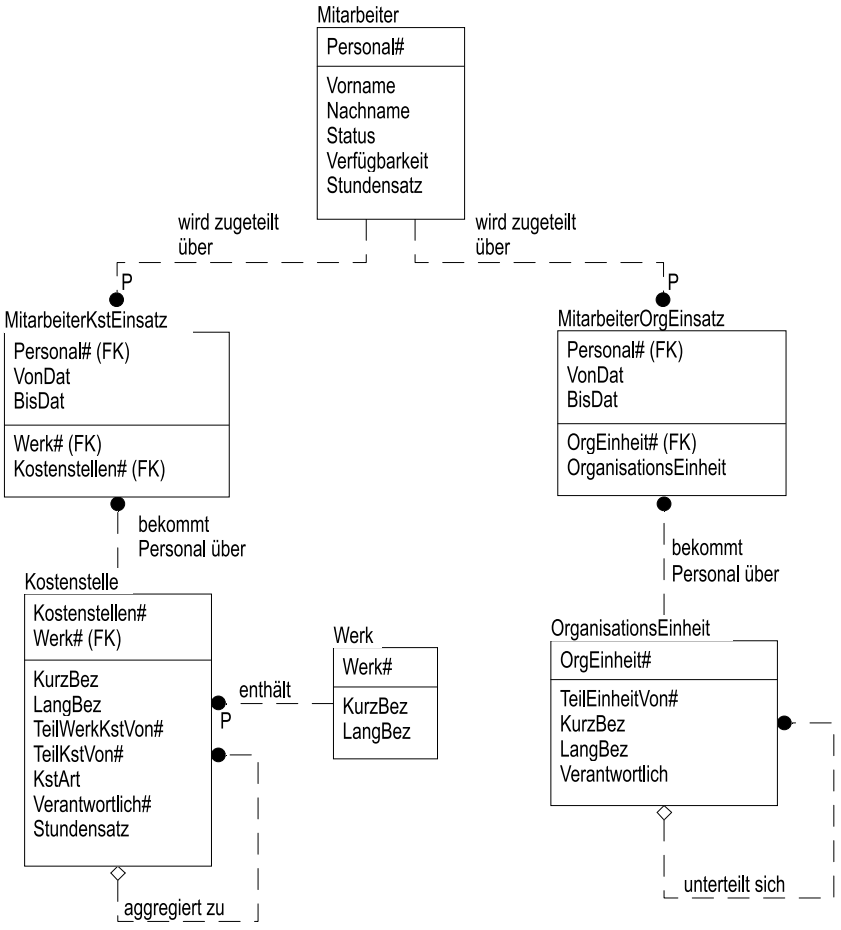

Abbildung 7: Zuordnung eines Mitarbeiters zu Kostenstelle und Organisationseinheit

Um zu verhindern, daß Mitarbeiter unkontrolliert an verschiedensten Aufträgen arbeiten und möglicherweise ohne Wissen von Vorgesetzten Leistungskontierungen auf Services durchführen, an denen Mitarbeiter nicht arbeiten sollen, muß es eine Möglichkeit geben, Mitarbeiter zu der Arbeit an bestimmten Services einzuteilen. Ein solcher Auftrag befindet sich in einem Status:

⟨MITARBAUFTR-STATUS⟩ := {`offen; in Arbeit; abgeschlossen; abgebrochen`}

Objekt: *MitarbeiterAuftrag*

Beschreibung:
Auftrag für einen Mitarbeiter, an einem Service zu arbeiten

Attribute:

Personal# → ⟨PERSONAL#⟩

Service# → ⟨SERVICE#⟩

Status → ⟨MITARBAUFTR-STATUS⟩ **(M)**

IntegritĎtsbedingungen:

keine

Beziehungen:

MitarbeiterAuftrag —•$_Z$ *MitarbAuftrProt* (5.3)

MitarbeiterAuftrag •— *Mitarbeiter* (5.3)

MitarbeiterAuftrag •— *Service* (5.1)

MitarbeiterAuftrag ◇—• *PlanWert* (5.7.1)

Ein Problem ergibt sich bei Mitarbeitern, die sehr flexibel eingesetzt werden oder „kurz und unbürokratisch“ ein Problem lösen. Hier darf die Haltung *„Ohne Auftrag vom Chef mach ich erst mal gar nix!“* durch das System nicht aufgezwungen werden, da Vorgesetzte oder Einteilungsbefugte in Notfällen möglicherweise nicht verfügbar sind.

Um auch solche Leistungen kontieren zu können, muß es einem Mitarbeiter, oder auch nur einem bestimmten Kreis von Mitarbeitern, möglich sein, in einem speziellen Kontierungsmodus auf einen Service zu kontieren, für den sie nicht eingeteilt sind. In diesem Fall muß der *MitarbeiterAuftrag* automatisch angelegt und protokolliert werden.

Objekt: *MitarbAuftrProt*

Beschreibung:
Protokolliert die „Selbsteinteilung“ eines Mitarbeiters zur Bearbeitung eines Services

Attribute:

MitarbAuftrProtLfd# → INTEGER

Personal# → ⟨PERSONAL#⟩ **(M)**

Service# → ⟨SERVICE#⟩ **(M)**

AnlegDat → DATE **(M)**

IntegritĎtsbedingungen:

keine

Beziehungen:

MitarbAuftrProt $\bullet_Z$—— *MitarbeiterAuftrag* (5.3)

5.4 Aufträge

Grundlage für die unternehmensinterne Leistungsverrechnung ist der *Auftrag*.[16] Zu jedem Auftrag ist ein Kostenverrechnungsschlüssel hinterlegt, über den festgelegt ist, welche Kostenstellen mit den auf diesen Auftrag aufgelaufenen Kosten belastet werden. Dieser Verrechnungsschlüssel wird in diesem Modell nicht abgebildet, da er im SAP R/3 implementiert ist und vom IV-Controllingsystem weder gepflegt noch ausgewertet wird.

Es kann mit dem Auftraggeber vereinbart worden sein, daß der Auftrag zu einem Festpreis abgerechnet wird. In diesem Fall richtet sich die Verrechnung nach dem verbindlichen Angebot oder sonstigen Vereinbarungen und nicht nach den auf diesen Auftrag aufgelaufenen Leistungen. Für das IV-Controlling ist es jedoch notwendig, die tatsächlich aufgelaufenen Leistungen zu kennen.

Neben den Standard-Aufträgen gibt es noch sogenannte *statistische Aufträge*.[17] Bei dieser Art von Werkaufträgen ist kein Verrechnungsschlüssel hinterlegt. Bei jeder Buchung auf einen solchen statistischen Auftrag muß die *zu belastende Kostenstelle* mitgebucht werden. Durch diese Auftragsart ist es möglich, kleinere, häufig auftretende Leistungen, wie z.B. das Aufstellen eines PC, abzurechnen, ohne daß aufwendig ein eigener Auftrag eingerichtet werden muß.

[16] Im SAP R/2 System wurde die Bezeichnung *WerkAuftrag* verwendet. Im R/3 System wird nur noch von *Auftrag*, vereinzelt von *InnenAuftrag* gesprochen

[17] Auch diese Bezeichnung stammt aus dem SAP System. In anderen Systemen finden sich entsprechende Auftragsarten

Eine weitere Auftragsvariante sind die `aktivierungspflichtigen`[18] Aufträge, die zur Abwicklung von Fremdbezug dienen.

⟨AUFTRAG-ART⟩ := {`Standard; aktivierungspflichtig; statistisch`}

Für die Zuordnung von Aufträgen gelten folgende Bedingungen:

1. Ist ein Auftrag einem Service zugeordnet, kann er keiner Investition zugeordnet sein
2. Ist ein Auftrag einer Investition zugeordnet, kann er keinem Service zugeordnet sein
3. `Aktivierungspflichtige` Aufträge müssen einer Investition zugeordnet werden
4. Alle anderen Auftragsarten werden einem Service zugeordnet

Wird ein Auftrag einem Service zugeordnet, so gilt er auch für im Servicestrukturbaum nachfolgende Services. Er wird also einer definierten Gruppe von Services zugeordnet.

Objekt: *Auftrag*

Beschreibung:
Ein Auftrag ist das für das Rechnungswesen zentrale Kontierungsobjekt

Attribute:

AuftragArt → ⟨AUFTRAG-ART⟩ **(M)**

Auftrag# → INTEGER

Service# → ⟨SERVICE#⟩

Invest# → ⟨INVEST#⟩

Angebot# → ⟨ANGEBOT#⟩

KurzBez → STRING **(M)**

[18] Der Begriff `aktivierungspflichtig` darf nicht als *in der Bilanz zu aktivieren* mißverstanden werden. Leistungen externer Berater können nicht in der Bilanz aktiviert werden, obwohl sie über aktivierungspflichtige Aufträge abgewickelt werden. Die korrekte steuerliche Behandlung ist Aufgabe des Rechnungswesens.

LangBez → STRING

Festpreis → BOOLEAN

IntegritĎtsbedingungen:

1. $\forall$ auf $\in Auftrag \bullet$ auf.Service# $\neq$ NULL
   $\Leftrightarrow$ auf.Invest# = NULL
2. $\forall$ auf $\in Auftrag \bullet$ auf.Invest# $\neq$ NULL
   $\Leftrightarrow$ auf.Service# = NULL
3. $\forall$ auf $\in Auftrag \bullet$ auf.AuftragArt = `aktivierungs-pflichtig`
   $\Leftrightarrow$ auf.Invest# $\neq$ NULL
4. $\forall$ auf $\in Auftrag \bullet$ auf.AuftragArt $\neq$ `aktivierungs-pflichtig`
   $\Leftrightarrow$ auf.Service# $\neq$ NULL

Beziehungen:

Auftrag •——◇ *Service* (5.1)

Auftrag •——◇ *Investition* (5.4)

Auftrag ——• *Bestellung* (5.6)

Auftrag ——• *LeistungsKontierung* (5.5)

Auftrag •$_Z$——◇ *Angebot* (5.4)

Ein Auftrag kann sich auf ein *Angebot* beziehen, das auf eine Anfrage eines Kunden - der anfordernden Kostenstelle - hin erstellt wurde. Dieses Angebot wird vom Kunden `angefordert`, befindet sich dann `in Arbeit` und wird schließlich `angeboten`. Vom Kunden wird es daraufhin `angenommen` oder `abgelehnt`. Es kann auch `verworfen` oder `überarbeitet` werden.

⟨ANGEBOT-STATUS⟩ := {`angefordert; in Arbeit; angeboten; angenommen; abgelehnt; überarbeitet; verworfen`}

Angebote können teilweise nur für einen bestimmten Zeitraum gültig sein.

Objekt: *Angebot*

Beschreibung:
Angebot an einen Kunden für eine angeforderte Leistung

Attribute:

Angebot# → INTEGER

KurzBez → STRING **(M)**

LangBez → STRING

Bearbeiter# → ⟨PERSONAL#⟩ **(M)**

AnfWerk# → ⟨WERK#⟩ **(M)**

AnfKst# → ⟨KOSTENSTELLEN#⟩ **(M)**

Status → ⟨ANGEBOT-STATUS⟩

GültigBisDat → DATE

IntegritĎtsbedingungen:

keine

Beziehungen:

Angebot —•$_P$ *AngebotPosition* (5.4)

Angebot ◇—•$_Z$ *Auftrag* (5.4)

Angebot •— *Mitarbeiter* (5.3)

Angebot •— *Werk* (5.2)

Angebot •— *Kostenstelle* (5.2)

Ein Angebot besteht aus mehreren Positionen, wobei eine Position die Leistung einer Organisationseinheit beschreibt. Dabei muß neben dem Volumen der Leistung auch die Einheit bekannt sein, in der das Leistungsvolumen angegeben wurde.

⟨ANGEBOTS-EINHEIT⟩ := {`Personalstunden; Personaltage; DM; $; Euro`}

Objekt: *AngebotPosition*

Beschreibung:
Position eines Angebotes, das die Leistung einer Organisationseinheit beschreibt

Attribute:

AngebPosLfd# → INTEGER

Angebot# → ⟨ANGEBOT#⟩ **(M)**

KurzBez → STRING **(M)**

LangBez → STRING

LstOrgEinheit# → ⟨ORGEINHEIT#⟩ **(M)**

Volumen → REAL **(M)**

Einheit → ⟨ANGEBOTS-EINHEIT⟩ **(M)**

IntegritĎtsbedingungen:

keine

Beziehungen:

AngebotPosition •$_P$— *Angebot* (5.4)

AngebotPosition •— *OrganisationsEinheit* (5.2)

Für den Fremdbezug von Lieferungen und Leistungen ist eine *Investition* notwendig, für die ein Budget in einer bestimmten Währungseinheit[19] besteht.

⟨WAEHRUNG⟩ := {`DM; Euro; $`}

Eine Investition ist für einen bestimmten Service vorgesehen. Nachdem eine Investition `beantragt` wurde, wird sie `freigegeben` oder `abgelehnt`. Sind alle zu einer Investition gehörenden Bestellungen abgewickelt oder ist das Budget ausgeschöpft, ist die Investition `abgeschlossen`. Zwischenzeitlich kann eine Investition auch `gesperrt` werden.

⟨INVEST-STATUS⟩ := {`beantragt; freigegeben; gesperrt; abgeschlossen; abgelehnt`}

Objekt: *Investition*

Beschreibung:

Eine Investition bildet die Grundlage für jede Bestellung

[19] Gespeichert werden sollte nur die Hauswährung. Um bei Währungsumstellungen (z.B. auf den Euro) sicher zu sein, auch den gesamten Datenbestand berücksichtigt zu haben, sollte ein Währungskennzeichen immer zusätzlich zur Kontrolle gespeichert werden. Die Eingabe kann in unterschiedlichen Währungen erfolgen und wird automatisch auf die Hauswährung umgerechnet.

Attribute:

Invest# → INTEGER

KurzBez → STRING (M)

LangBez → STRING

Service# → ⟨SERVICE#⟩

Budget → CURRENCY

Währung → ⟨WAEHRUNG⟩

Status → ⟨INVEST-STATUS⟩

IntegritĎtsbedingungen:

keine

Beziehungen:

Investition —• *Auftrag* (5.4)

Investition •— *Service* (5.1)

5.5 Leistungskontierung

Alle Leistungskontierungen müssen einem *Auftrag* zurechenbar sein. Aufträge können für unterschiedliche Services eines Servicestrukturbaumes erteilt werden, gewöhnlich für ein oder mehrere Arbeitspakete. Die Kontierung erfolgt jedoch häufig auf einer tieferen Ebene, beispielsweise auf eine Aktivität. Folgt man dem Strukturbaum von dem kontierten Objekt in Richtung der Wurzel, stößt man auf einer Ebene auf einen Service, dem ein Auftrag zugeordnet ist.

Wenn die Struktur der Aufträge zur Servicestruktur paßt, sollte es für jeden Service nur einen kontierbaren[20] Auftrag geben. Andernfalls ist die Servicestruktur nicht fein genug abgebildet.

Da einem Service mehrere Aufträge zugeordnet werden können, ist der Auftrag, über den die Kontierung verrechnet werden soll, nicht eindeutig identifizierbar. Daher ist es notwendig, vor jeder Kontierung die Menge der für einen Mitarbeiter möglichen Kombinationen von Aufträgen und Services zu bestimmen, auf die eine Kontierung erfolgen kann.

[20] Aktivierungspflichtige Aufträge zählen in diesem Zusammenhang nicht dazu, da es hier nur um die Kontierung von *eigenen* Leistungen geht

Ist der Mitarbeiter eingeteilt zur Arbeit am Arbeitspaket `P350.1.5`, für das die Werkaufträge `280872` und `250478` gelten, dann muß es für ihn Kontierungsmöglichkeiten auf `P350.1.5.280872` und `P350.1.5.250478` geben.

Der elegantere und konsistentere Weg wäre, nur einen Auftrag pro Service zuzulassen. Dann würden Auftrags- und Servicestruktur zueinander passen und der Mitarbeiter braucht sich nicht mehr darum zu kümmern, wie seine Leistung kostenrechnerisch verbucht wird. Er braucht sie lediglich den *Aufgaben*[21] zuzuordnen, zu denen er eingeteilt wurde, und nicht den kostenrechnerischen *Aufträgen*.

Die Auftragsnummer des tatsächlich kontierten Auftrages wird im Attribut Auftrag# festgehalten. Dieser Wert muß automatisch ermittelt werden und darf vom Benutzer nicht verändert werden können.

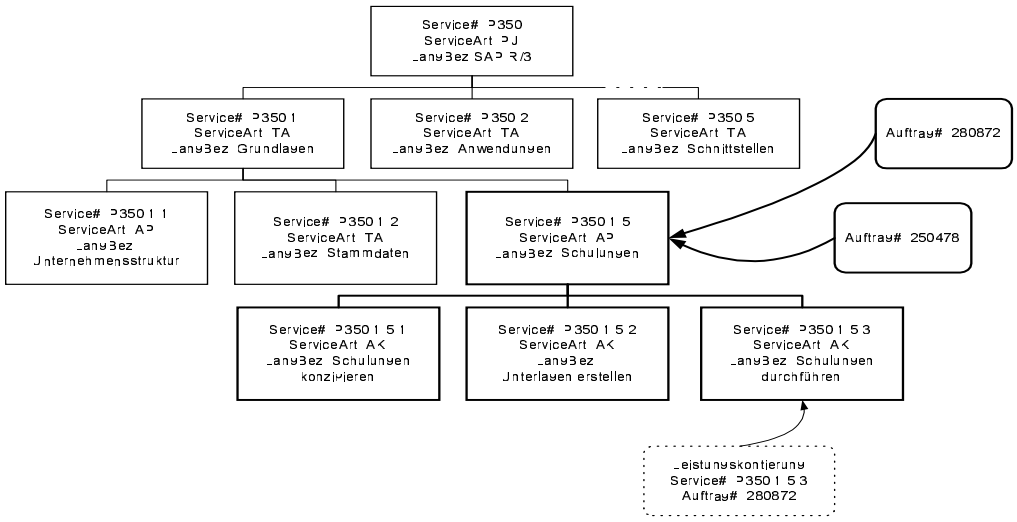

Abbildung 8: Ausschnitt aus dem Servicestrukturbaum für den Service P350. Eine Leistungskontierung auf den Service P350.1.5.3 kann über die Aufträge 280872 oder 250478 abgerechnet werden.

Handelt es sich bei einem Auftrag, auf den kontiert werden soll, um einen sogenannten *statistischen Auftrag*, muß die zu belastende Kostenstelle in den Attributen BelastetesWerk# und BelasteteKst# festgehalten werden. Nur wenn einer Kontierung ein statistischer Auftrag zugrunde liegt, dürfen die Erfassungsfelder für die zu belastende Kostenstelle auf der Kontierungsmaske aktiv sein.

Leistungen können in verschiedenen Einheiten erbracht werden. So sollen nicht nur `Arbeitsstunden`, sondern auch `Arbeitswerte`, wie sie in anderen Branchen üblich

[21] Abgebildet durch einen Service mit ServiceArt = `Aktivität`

sind, oder anderweitig vereinbarte Leistungseinheiten kontierbar sein, wie beispielsweise die zu einem Pauschalpreis angebotene Installation inklusive Lizenzgebühren für das Betriebssystem Windows 95.

⟨LEISTUNGS-ART⟩ := {`Stunden; Arbeitswert; Win95Inst`}

Dem Benutzer steht es frei, für welchen Zeitraum er seine Leistungen kontiert. Standardmäßig sollte täglich - möglichst unmittelbar nach Abschluß der Leistung - kontiert werden. Arbeitet ein Mitarbeiter über einen längeren Zeitraum an sehr wenigen Aufgaben, hat er die Möglichkeit, die Kontierungen für einen vollständigen Monat durchzuführen.

⟨KONT-ZEITRAUM⟩ := {`Tag; Monat`}

Bei der Entwicklung von Software lassen sich unterschiedliche Phasen identifizieren (siehe [Boe76], S. 1227ff und [Spi89], S. 26ff). Leistungskontierungen lassen sich diesen Phasen zuordnen. Eine Kontierung kann sich dabei auf eine Phase beziehen, die vom aktuellen Stand des Gesamtprojektes abweicht. So ist es denkbar, daß ein Entwickler noch die Spezifikation ergänzt oder korrigiert, obwohl sich das System bereits in der Testphase befindet. Als Defaultwert sollte in der Eingabemaske allerdings immer der aktuelle Projektstand vorgegeben werden.

⟨PHASENNUMMER⟩ := {`0..6`}

Ebenso wie die Kontierung einer Phasennummer ist die Kontierung einer Fehlernummer nur bei bestimmten Servicearten sinnvoll. Die Kontierung von Phasennummern ist nur bei Projekten sinnvoll, während Fehlernummern nur bei Systembetreuungen oder Dienstleistungsaufgaben Verwendung finden sollten. Die Eingabemaske für die Leistungskontierung sollte diesen Umstand berücksichtigen und in Abhängigkeit von der kontierten Serviceart entsprechende Felder aktivieren bzw. deaktivieren. Ausführlichere Gedanken zur Kontierung von Fehlernummern finden sich in Abschnitt 5.8.3.

Die Arbeiten an einem Service können nach einer Leistungskontierung noch andauern (`in Arbeit`) oder sie werden mit der aktuellen Leistungskontierung regulär `abgeschlossen` oder vorzeitig `abgebrochen`.

⟨LEISTUNGS-STATUS⟩ := {`in Arbeit; abgeschlossen; abgebrochen`}

Objekt: *LeistungsKontierung*

Beschreibung:

Eine Leistung, die einem Service, und somit einem Auftrag zugerechnet wird

Attribute:

KontLfd# → INTEGER

Mitarbeiter# → ⟨PERSONAL#⟩ **(M)**

Service# → ⟨SERVICE#⟩ **(M)**

Auftrag# → ⟨AUFTRAG#⟩ **(M)**

IstLeistung → REAL **(M)**

Leistungsart → ⟨LEISTUNGS-ART⟩ **(M)**

BelasteteKst# → ⟨KOSTENSTELLEN#⟩

BelastetesWerk# → ⟨WERK#⟩

Status → ⟨LEISTUNGS-STATUS⟩

Phase → ⟨PHASENNUMMER⟩

Fehler# → ⟨FEHLER#⟩

KontDat → DATE

Zeiteinheit → ⟨KONT-ZEITRAUM⟩

Bemerkung → MEMO

IntegritĎtsbedingungen:

$\forall$ lk $\in$ *LeistungsKontierung*, auf $\in$ *Auftrag* •
lk.Auftrag# = auf.Auftrag# $\wedge$ auf.AuftragArt = `statistisch`
$\Leftrightarrow$ lk.Werk# $\neq$ NULL $\wedge$ lk.Kostenstellen# $\neq$ NULL

Beziehungen:

LeistungsKontierung •—— *Service* (5.1)

LeistungsKontierung •—— *Auftrag* (5.4)

LeistungsKontierung •—— *Mitarbeiter* (5.3)

LeistungsKontierung •——◇ *Werk* (5.2)

LeistungsKontierung •——◇ *Kostenstelle* (5.2)

LeistungsKontierung •——◇ *Fehler* (5.8.3)

5.6 Fremdbezug

Nicht alle Leistungen, die in einen Service einfließen, sind selbsterstellte Leistungen. Zu einem nicht unerheblichen Teil gehen auch Leistungen fremder *Lieferanten* ein. Dabei kann es sich um `externe` Lieferanten wie SAP, IBM, Lynx etc oder um `interne` Lieferanten handeln, die aus dem eigenen Unternehmen stammen, wie beispielsweise die Marketingabteilung.

Objekt: *Lieferant*

Beschreibung:
Externer Lieferant von Waren und Dienstleistungen

Attribute:

Lieferant# → INTEGER

KurzBez → STRING **(M)**

LangBez → STRING

IntegritĎtsbedingungen:

keine

Beziehungen:

Lieferant ◇—● *Bestellung* (5.6)

Bevor Lieferanten eine Leistung erbringen, muß eine *Bestellung* erfolgt sein. Bestellungen werden für jede Kostenart getrennt erteilt. Es wird unterschieden zwischen Bestellungen für `Hardware, Software` und Dienstleistungen durch `Fremdpersonal`.

⟨KOSTENART⟩ := {`Hardware; Software; Fremdpersonal`}

Neben der unternehmensintern vergebenen Bestellnummer vergeben einige Lieferanten eigene Auftrags- oder Bestellnummern. Um bei Rückfragen ggf. schnell reagieren zu können, sollte diese fremde Bestellnummer als AuftrNrLieferant abgelegt werden.

Eine Bestellung befindet sich im Verlauf ihrer Abwicklung in verschiedenen Stadien:

⟨BESTELL-STATUS⟩ := {`in Arbeit; erteilt; Teillieferung erfolgt; ausgeliefert; abgerechnet; storniert`}

Wie bei den eigenen Angeboten können auch für Bestellungen Festpreise vereinbart werden.

Eine Bestellung, die an einen unternehmensinternen Lieferanten geht, wird zwar gewöhnlich als *Bedarfsanforderung* bezeichnet, kann aber ebenso als Variante einer *Bestellung* gesehen werden, bei der statt der Nummer des Lieferanten die Nummer der Leistenden Kostenstelle erfaßt wird (Integritätsbedingungen 1 und 2).

Ohne einen aktivierungspflichtigen Auftrag (siehe Abschnitt 5.4) kann keine Bestellung ausgelöst werden (Integritätsbedingung 3).

Objekt: *Bestellung*

Beschreibung:

Bestellung für Fremdbezug von Lieferungen und Leistungen extern und intern

Attribute:

Bestell# → STRING

Auftrag# → ⟨AUFTRAG#⟩ **(M)**

Lieferant# → ⟨LIEFERANT#⟩

AuftrNrLieferant → STRING

LeistendesWerk → ⟨WERK#⟩

LeistendeKst → ⟨KOSTENSTELLEN#⟩

BestellDat → DATE

Festpreis → BOOLEAN

KostenArt → ⟨KOSTENART⟩ **(M)**

Status → ⟨BESTELL-STATUS⟩ **(M)**

IntegritĎtsbedingungen:

1. $\forall$ best $\in$ *Bestellung* • best.Lieferant $\neq$ NULL $\Leftrightarrow$ best.LeistendesWerk = NULL $\wedge$ best.LeistendeKst = NULL
2. $\forall$ best $\in$ *Bestellung* • best.LeistendesWerk $\neq$ NULL $\wedge$ best.LeistendeKst $\neq$ NULL $\Leftrightarrow$ best.Lieferant = NULL
3. $\forall$ auf $\in$ *Auftrag*, best $\in$ *Bestellung* • auf.Auftrag# = best.Auftrag# $\Rightarrow$ auf.AuftragArt = `aktivierungspflichtig`

Beziehungen:

Bestellung —•$_P$ *BestellPosition* (5.6)

Bestellung —• *Rechnung* (5.6)

Bestellung •— *Auftrag* (5.4)

Bestellung •—◇ *Lieferant* (5.6)

Eine Bestellung sollte mehr als eine *Bestellposition* enthalten können. Wichtig ist die Angabe der Einheit, in der bestellt wird.

⟨BESTELLEINHEIT⟩ := {`Stück; Packungen; Stunden; Kilogramm`}

Neben der Einheit sind die BestellImenge, der Einzelpreis und die Währung, in der die Einzelpreisangabe erfolgt, von Bedeutung.

Objekt: *BestellPosition*

Beschreibung:
Position einer Bestellung

Attribute:

BestellPosLfd# → INTEGER

Bestell# → ⟨BESTELL#⟩

KurzBez → STRING **(M)**

LangBez → STRING

BestellEinheit → ⟨BESTELLEINHEIT⟩ **(M)**

BestellMenge → REAL **(M)**

Einzelpreis → REAL **(M)**

Währung → ⟨WAEHRUNG⟩ **(M)**(5.4)

IntegritĎtsbedingungen:

keine

Beziehungen:

BestellPosition •$_P$— *Bestellung* (5.6)

Nachdem fremde Leistungen bestellt bzw. angefordert und erbracht wurden, folgt eine Rechnung.

Der Rechnungsbetrag kann aus verschiedenen Gründen vom Bestellwert abweichen und muß daher ebenfalls bekannt sein.

Auch hier kann es eine vom Lieferanten vergebene Rechnungsnummer geben.

Objekt: *Rechnung*

Beschreibung:
Rechnung eines externen Lieferanten, die aus einer Bestellung von Lieferungen und Leistungen resultiert

Attribute:

RechnungsLfd# → INTEGER

Bestell# → ⟨BESTELL#⟩

Status → ⟨RECHNUNGS-STATUS⟩

Rechnungsbetrag → REAL **(M)**

Währung → ⟨WAEHRUNG⟩ **(M)**(5.4)

RechNrLieferant → STRING

RechnungsDat → DATE

EingangDat → DATE

ZahlungsZielDat → DATE

IntegritĎtsbedingungen:

keine

Beziehungen:

Rechnung •—— *Bestellung* (5.6)

Erfolgt die Fremdleistung nicht durch einen externen sondern durch einen unternehmensinternen Lieferanten, wird keine Rechnung erstellt. Es wird lediglich eine unternehmensinterne Verrechnung durchgeführt, über die ähnliche Informationen wie über eine Rechnung vorhanden sind.

Objekt: *IntVerrechnung*

Beschreibung:

Daten der unternehmensinternen Verrechnung, die aus einer Bedarfsanforderung an einen anderen Unternehmensbereich resultiert

Attribute:

IntVerrechnungLfd# → INTEGER

Bestell# → ⟨BESTELL#⟩ **(M)**

Verrechnungsbetrag → CURRENCY **(M)**

Währung → ⟨WAEHRUNG⟩ **(M)**

IntegritĎtsbedingungen:

keine

Beziehungen:

IntVerrechnung —● *Bestellung* (5.6)

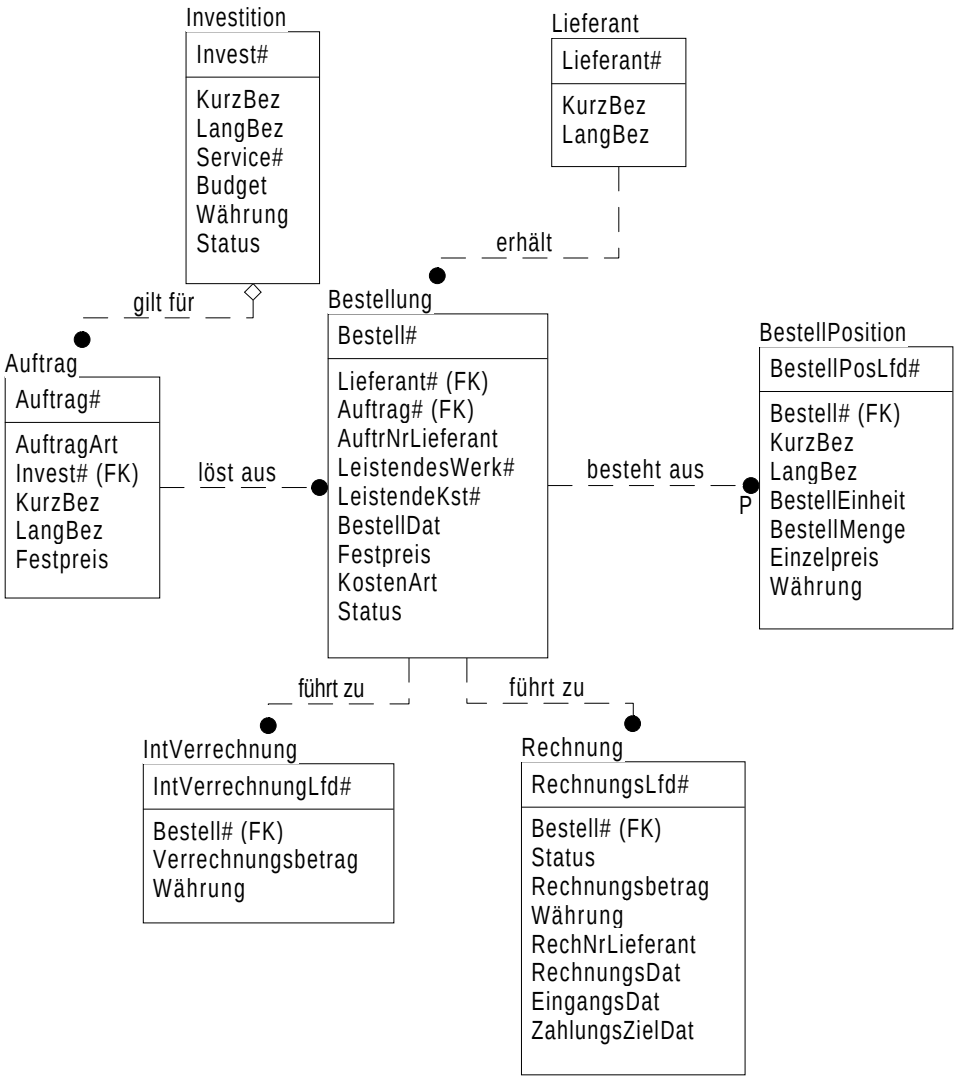

Abbildung 9: Bestellungen sind Aufträgen zugeordnet, für die eine Investition genehmigt sein muß

5.7 Plankomponente

5.7.1 Operative Planung

Pläne beziehen sich auf die unterschiedlichsten Objekte. Ein Mitarbeiter plant seine Überstunden und Urlaubstage, die Projektleitung plant 120.000 DM für den Kauf von Hardware über den Lieferanten IBM ein und der Gruppenleiter plant 10 Arbeitsstunden des Mitarbeiters Schmidt für das Projekt P350.1.2.4 ein.

All diese Planungen haben gleiche Attribute, beziehen sich lediglich auf unterschiedliche Objekte. Es ist daher notwendig zu wissen, auf welchen Objekt*typ* sich die Planung bezieht.

⟨BEZUGSOBJEKT⟩ := {`Auftrag; Mitarbeiter; Kostenstelle; OrgEinheit; Service; Investition; Bestellung`}

Aus dem BezugsObjekt läßt sich ablesen, welcher Fremdschlüssel eines Planwertes, der die Beziehung zu dem beplanten Objekt herstellt, gefüllt sein darf, und welche Fremdschlüssel auf den Wert `NULL` gesetzt sein müssen. Die Angabe des Bezugsobjektes scheint nur auf den ersten Blick überflüssig, da man aus der Belegung der Fremdschlüsselattribute den Typ des Bezugsobjektes bestimmen könnte. Für Integritätsprüfungen der Plandaten und deren Auswertung ist die Angabe des Bezugsobjekttyps hilfreich.

Auch Pläne sind hierarchisch aufgebaut. Die Personalplanungen der Gruppen einer Kostenstelle werden zur Personalplanung der Kostenstelle verdichtet. Die Personalplanungen mehrerer Kostenstellen lassen sich zu einem Planwert einer Knotenkostenstelle aggregieren. Dies wäre eine `BottomUp`-Planung. Umgekehrt läßt sich beispielsweise `TopDown` das Budget für ein Projekt auf die einzelnen Teilaufgaben und Arbeitspakete aufsplitten. Es muß lediglich die PlanRichtung verbindlich festgelegt sein.

⟨PLANRICHTUNG⟩ := {`TopDown; BottomUp`}

Planwerte, die nicht von Hand erfaßt werden, sondern sich errechnen lassen (wie die Personalplanung der Knotenkostenstelle) werden als *abgeleitete* Planwerte bezeichnet. Ob es sich um einen abgeleiteten Planwert handelt, der durch den Benutzer nicht verändert werden kann, oder nicht, wird im Attribut abgeleitet festgelegt.

Wird ein Planwert angelegt oder verändert, dann muß ein Algorithmus angestoßen werden, der - abhängig von der Planungsrichtung - den nächsten abgeleiteten Vorgänger bzw. Nachfolger des Planwertes ermittelt und aktualisiert.

Um die hierarchische Struktur der einzelnen Planwerte abzubilden, wird im Attribut TeilPlanVon der Primärschlüssel des übergeordneten Planwertes festgehalten.

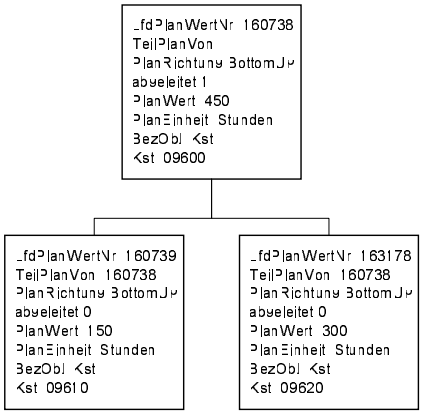

Abbildung 10: Beispiel für eine `BottomUp` Planung

Durch die Angabe einer PlanEinheit ist es nicht nur möglich, gleiche Sachverhalte aus unterschiedlichen Sichtweisen (z.B. Personalaufwand nach Stunden, Tagen oder in DM) sondern auch unterschiedliche Sachverhalte wie Überstunden oder Urlaubstage zu planen.

⟨PLANEINHEIT⟩ := {`DM; Euro; $; Stunden; Personaltage; Personaljahre; Überstunden; Weiterbildung; Urlaubstage`}

Eine Planung erfolgt zu einem bestimmten Zeit*punkt* für einen bestimmten Zeit*raum*, der an einem bestimmten Datum beginnt.

⟨PLANZEITRAUM⟩ := {`Tag; Woche; Monat; Geschäftsjahr; Kalenderjahr`}

Pläne, die nach Abschluß des geplanten Vorhabens 100%ig erfüllt werden, lassen drei Schlüsse zu:

1. Es ist sehr gut geplant worden
2. Die Ist-Daten wurden so manipuliert, daß der Plan genau erfüllt wird
3. Die Planwerte wurden nachträglich so verändert, daß sie zu den Ist-Daten passen

Im ersten Fall gibt es keine Probleme. Der zweite Fall sollte bei der Implementierung an anderen Stellen - wie der Gestaltung der Erfassungsmasken - im Hinterkopf behalten werden. Um sich vor dem dritten Fall zu schützen, sollten zwei Schritte unternommen werden. Der Planwert muß einen Status besitzen - beispielsweise `in Arbeit, freigegeben` und `endgültig fest`. Hat der Planwert den Status `endgültig fest` kann a) der Status und b) der Planwert selbst nicht mehr verändert werden. Ein veränderter Planwert hat den Status `verändert`. Ein schlechter oder überholter Plan kann auch verworfen werden.

⟨PLAN-STATUS⟩ := {`in Arbeit; freigegeben; angepaßt; endgültig fest; verworfen`}

Objekt: *PlanWert*

Beschreibung:
Planwert für ein beplanbares Objekt

Attribute:

PlanWertLfd# → INTEGER

TeilPlanVon# → ⟨PLANWERTLFD#⟩

PlanRichtung → ⟨PLANRICHTUNG⟩ **(M)**

abgeleitet → BOOLEAN

PlanWert → REAL **(M)**

PlanEinheit → ⟨PLANEINHEIT⟩ **(M)**

PlanDat → DATE **(M)**

PlanBeginnDat → DATE **(M)**

PlanZeitraum → ⟨PLANZEITRAUM⟩ **(M)**

Status → ⟨PLAN-STATUS⟩ **(M)**

Verantwortlich# → ⟨PERSONAL#⟩ **(M)**

BezugsObjekt → ⟨BEZUGSOBJEKT⟩ **(M)**

Service# → ⟨SERVICE#⟩

Werk# → ⟨WERK#⟩

Kostenstellen# → ⟨KOSTENSTELLEN#⟩

Mitarbeiter# → ⟨PERSONAL#⟩

OrgEinheit# → ⟨ORGEINHEIT#⟩

Auftrag# → ⟨AUFTRAG#⟩

Bestell# → ⟨BESTELL#⟩

Invest# → ⟨INVEST#⟩

IntegritĎtsbedingungen:

∀ pw *inPlanWert* ∃$_1$ pw.?# ∈ {Service#; Werk#; Kostenstellen#; Mitarbeiter#; OrgEinheit#; Bestell#; Invest#} ≠ NULL

Beziehungen:

PlanWert —• *PlanProtokoll* (5.7.1)

PlanWert •—◇ *Service* (5.1)

PlanWert •—◇ *Mitarbeiter* (5.3)

PlanWert •—◇ *OrganisationsEinheit* (5.2)

PlanWert •—◇ *Kostenstelle* (5.2)

PlanWert •—◇ *Werk* (5.2)

PlanWert •—◇ *Auftrag* (5.4)

PlanWert •—◇ *Investition* (5.4)

PlanWert •—◇ *Bestellung* (5.6)

Änderungen an freigegebenen Plänen sollten protokolliert werden. Dabei interessiert, *wer wann welchen* Plan *wie* verändert hat.

Objekt: *PlanProtokoll*

Beschreibung:
Protokoll der Änderungen an Planwerten

Attribute:

PlanProtLfd# → INTEGER

Mitarbeiter# → ⟨PERSONAL#⟩

PlanWert# → ⟨PLANWERT#⟩

Differenz → REAL

ÄnderDat → DATE

Status → ⟨PERSAUFTR-STATUS⟩ **(M)**

IntegritĎtsbedingungen:

keine

Beziehungen:

PlanProtokoll •—— *PlanWert* (5.7.1)

Die Planwerte für alle Bezugsobjekte in einer Entität abzubilden, kann möglicherweise zu einem erheblichen Implementierungsaufwand führen. Zur Einhaltung der Integritätsbedingungen müssen beim Anlegen eines Planwertes zahlreiche Überprüfungen durchgeführt werden. Daher kann es einfacher sein, für jeden zu beplanenden Objekttyp ein separates Planobjekt vorzusehen. Dieses Objekt enthält mit Ausnahme des Attributes Bezugsobjekt und den dann überflüssigen Fremdschlüsseln die gleichen Attribute wie das Objekt *PlanWert*. Das Planobjekt für einen Mitarbeiter wird stellvertretend für alle anderen Planobjekte dargestellt.

Objekt: *MitarbeiterPlan*

Beschreibung:
Planwert für einen Mitarbeiter

Attribute:

MitarbeiterPlanLfd# → INTEGER

TeilPlanVon# → ⟨PLANWERTLFD#⟩

PlanRichtung → ⟨PLANRICHTUNG⟩ **(M)**

abgeleitet → BOOLEAN

PlanWert → REAL **(M)**

PlanEinheit → ⟨PLANEINHEIT⟩ **(M)**

PlanDat → DATE **(M)**

PlanBeginnDat → DATE **(M)**

PlanZeitraum → ⟨PLANZEITRAUM⟩ **(M)**

Status → ⟨PLAN-STATUS⟩ **(M)**

Verantwortlich# → ⟨PERSONAL#⟩ **(M)**

Mitarbeiter# → ⟨PERSONAL#⟩ **(M)**

IntegritĎtsbedingungen:

keine

Beziehungen:

MitarbeiterPlan •—— *Mitarbeiter* (5.3)

5.7.2 Strategische Planung

Neben der operativen Planung soll es in diesem System auch die Möglichkeit geben, strategische Ziele wie etwa die Senkung der Ausfallzeit um 20% für jede Organisationseinheit festzuhalten. Soll- und Ist-Werte werden in einfacher numerischer Form erfaßt.

Objekt: *Ziele*

Beschreibung:
Übersicht über strategische Ziele und Zielerreichung einer Organisationseinheit

Attribute:

ZielLfd# → INTEGER

OrgEinheit# → ⟨ORGEINHEIT#⟩

Beschreibung → MEMO

StartDat → DATE

IstWert → REAL

SollWert → REAL

Verantwortlich# → ⟨PERSONAL#⟩

IntegritĎtsbedingungen:

keine

Beziehungen:

Ziele •—— *OrganisationsEinheit* (5.2)

5.8 Weitere Komponenten

5.8.1 Schlüsseltabellen

Um eigene Datentypen zu realisieren, gibt es unterschiedliche Möglichkeiten. Am sinnvollsten, da plattformunabhängig und leicht konfigurierbar, erscheint die Verwendung von *Schlüsseltabellen*. Aus Gründen der Übersichtlichkeit sollten sich Schlüsseltabellen bereits durch die Bezeichnung von Tabellen unterscheiden, mit denen die eigentlichen Objekte des Datenmodells abgebildet werden.

Das Präfix k_[22] könnte solche Schlüsseltabellen kennzeichnen.

Neben einer Kurzbezeichnung, die gleichzeitig als Primärschlüssel dient, ist auch eine Langbezeichnung nützlich. Ein Attribut Beschreibung erlaubt die Eingabe eines längeren erläuternden Textes für jedes Element der auf diesem Weg definierten Wertemenge. Ein Aktivitätskennzeichen mit den Ausprägungen `aktiv (A)`, `inaktiv (I)` und `Prognose (P)` erlaubt es, bestimmte Ausprägungen zwar vorzusehen, für den sofortigen Gebrauch jedoch zu sperren oder nur als vorläufig zu betrachten.

⟨AKTIVKENNZ⟩ := {`A; I; P`}

Die Schlüsseltabelle zur Definition des Datentyps ⟨SERVICE-ART⟩ würde folgendermaßen angelegt werden:

Objekt: *k_SERVICE-ART*

Beschreibung:
Definiert die möglichen Ausprägungen des Datentyps ⟨SERVICE-ART⟩

Attribute:

KurzBez# → STRING

LangBez → STRING

Beschreibung → MEMO

AktivKennz → ⟨AKTIVKENNZ⟩

[22] k steht für Key

5.8.2 Umrechnungskomponente

Die im System verwendeten Recheneinheiten werden in dem Objekt *Einheit* zentral gepflegt. Zu jeder Recheneinheit existiert eine Kurzbezeichnung, die gleichzeitig als Primärschlüssel dient, eine Langbezeichnung und ein kurze Beschreibung.

Objekt: *Einheit*

Beschreibung:
mögliche Recheneinheiten

Attribute:

KurzBez# → STRING

LangBez → STRING

Beschreibung → MEMO

AktivKennz → ⟨AKTIVKENNZ⟩ **(M)**(5.8.1)

IntegritĎtsbedingungen:

keine

Beziehungen:

keine

KurzBez	LangBez	Beschreibung	AktivKennz
PT	Personaltag	Leistung eines Mitarbeiters pro Arbeitstag	A
EU	Euro	Währungseinheit Euro	I
DM	Deutsche Mark	Währungseinheit DM	A
...	...	...	...

Für die Umrechnung von einer Einheit in eine andere müssen Informationen über den rechnerischen Zusammenhang vorhanden sein. Ein `Personaljahr (PJ)` beinhaltet beispielsweise 220 `Personaltage (PT)`. Zwischen diesen Einheiten existiert ein Umrechnungs*faktor* von 220. Diese Angaben sind zu verschiedenen Zeitpunkten möglicherweise unterschiedlich, da sich diese Werte beispielsweise durch Änderungen von Tarifverträgen oder die Streichung von Feiertagen von Jahr zu Jahr ändern

können. Daher ist es notwendig zu wissen, für welchen Zeitraum dieses Verhältnis gültig ist. Auch hier ist ein Aktivitätskennzeichen sinnvoll, das angibt, ob eine erfaßte Umrechnungsvorschrift auch verwendet werden darf.

Objekt: *Umrechnung*

Beschreibung:

Gibt das Umrechnungsverhältnis zwischen den Recheneinheiten A und B an

Attribute:

UmRechLfd# → INTEGER

A# → ⟨KURZBEZ#⟩ **(M)**

B# → ⟨KURZBEZ#⟩ **(M)**

AinB → REAL **(M)**

VonDat → DATE **(M)**

BisDat → DATE **(M)**

AktivKennz → ⟨AKTIVKENNZ⟩ **(M)**(5.8.1)

IntegritĎtsbedingungen:

keine

Beziehungen:

keine

So gelten beispielsweise vom 01.01.96 bis zum 31.12.97 218 Personaltage (PT) als ein Personaljahr (PJ). Seit dem 01.01.98 müssen für ein Personaljahr 220 Personaltage geleistet werden. Ab dem 01.07.99 werden 215 Personaltage für ein Personaljahr prognostiziert. Der entsprechende Eintrag in der Umrechnungstabelle sieht folgendermaßen aus: [23]

A	B	AinB	VonDat	BisDat	AktivKennz
PT	PJ	218	01.01.96	31.12.1997	A
PT	PJ	220	01.01.98	31.12.9999	A
PT	PJ	215	01.07.99	31.12.9999	P
...	...	...	...	...	...

[23] Auf den laufenden Schlüssel wird hier und im folgenden Beispiel aus Gründen der Übersichtlichkeit verzichtet

Ebenso wie ein Umrechnungsverhältnis zwischen zwei Einheiten muß auch der *Preis* einer Leistungseinheit bekannt sein. Der Einzelpreis einer Leistung wird in einer Währung angegeben und gilt für einen bestimmten Zeitraum.

Objekt: *Preis*

Beschreibung:

Legt den Stückpreis einer Leistungseinheit für einen bestimmten Zeitraum fest

Attribute:

PreisLfd# → INTEGER

Einheit# → ⟨KURZBEZ#⟩ **(M)**

Einzelpreis → CURRENCY **(M)**

Währung → ⟨WAEHRUNG⟩ **(M)**(5.4)

VonDat → DATE **(M)**

BisDat → DATE **(M)**

AktivKennz → ⟨AKTIVKENNZ⟩ **(M)**(5.8.1)

IntegritĎtsbedingungen:

keine

Beziehungen:

keine

Als Beispiel für einen Eintrag in der Preistabelle wird die zu einem Pauschalpreis angebotene Leistung *Installation und Lizenz des Betriebssystems Windows 95* verwendet. Bis zum 31.12.97 kostete diese Leistung 180,50 DM. Seit dem 01.01.98 ist der Betrag auf 210,00 DM gestiegen. Bereits vorgesehen aber noch nicht verfügbar ist die Leistung *Installation und Lizenz des Betriebssystems Windows XX (95 oder 98)*, die in Euro abgerechnet wird.

Einheit	Einzelpreis	Währung	VonDat	BisDat	AktivKennz
Win95Inst	180,50	DM	01.01.96	31.12.1997	A
Win95Inst	210,00	DM	01.01.98	31.12.9999	A
WinXXInst	120,00	Euro	01.07.99	31.12.9999	I
...	...	...	...	...	...

5.8.3 Fehler

Dieses Objekt soll nicht nur als Schlüsseltabelle für die Verwendung von Fehlernummern bei der Leistungskontierung dienen, sondern kann auch als Ansatz für eine „Knowledge-Base“ verwendet werden. Die Kontierung von Fehlernummern kann dazu dienen, ungewöhnliche Häufungen von Fehlern aufzudecken (siehe Abschnitte 5.5). Abrufbare Hinweise zu den Fehlern können die Effizienz der Fehlerbehebung erhöhen.

Objekt: *Fehler*

Beschreibung:
Sammlung von Fehlern und Vorkommnissen

Attribute:

Fehler# → INTEGER

KurzBez → STRING

LangBez → STRING

Hinweis → MEMO

IntegritĎtsbedingungen:

keine

Beziehungen:

Fehler ◇—● *LeistungsKontierung* (5.5)

Ein Eintrag in die Fehlertabelle, der eine fehlerhafte Clusterverwaltung bei bestimmten Quantum Festplatten beschreibt, würde so aussehen:

FehlerNr	KurzBez	LangBez	Hinweis
42	HDDQuant	Quantum Festpl. def.	Fehlerhafte Clusterverwaltung bei Quantum HDDs der Typen QHDGX 3700 und QHDGX 4300 sind bekannt. Kulanzaustausch über Lieferant 4711
...	...	...	...

5.9 Gesamtsystem

5.9.1 Beziehungen zwischen den Teilmodellen

Die in den vorhergehenden Abschnitten beschriebenen Teilmodelle können jetzt zu einem Gesamtsystem zusammengesetzt werden. Die Beziehungen zwischen allen Objekten sind bereits in den Objektdefinitionen aufgezeigt worden, sollen hier aber nochmals erläutert werden.

Einem *Service* können *Aufträge* zugeordnet sein, über die die Anbindung an das Rechnungswesen erfolgt. Im Idealfall sollte einem Service nur ein nicht aktivierungspflichtiger Auftrag zugeordnet werden. Andernfalls besteht eine Asymmetrie zwischen der Servicestruktur und der Auftragsstruktur, die vermieden werden sollte.

Ein Auftrag kann dabei für mehrere Services gelten, indem der Auftrag *einem* im Servicestrukturbaum höher stehenden Service zugeordnet wird. Der Auftrag gilt dann für diesen und alle untergeordneten Services. Dadurch ist eine feinere Aufgliederung von Leistungen für das IV-Controlling möglich, die für die Leistungsverrechnung nicht notwendig ist.

Leistungskontierungen können nur auf Services erfolgen, für die ein Auftrag identifizierbar ist. Damit wird sichergestellt, daß die unternehmensinterne Leistungsverrechnung vollständig über Aufträge abgewickelt wird, gleichzeitig aber ein Controlling von Projekten, Systemen und Dienstleistungstätigkeiten detaillierter, da unabhängig von Aufträgen, durchgeführt werden kann.

Für einen *Mitarbeiter* kann es mehrere *MitarbeiterAufträge* geben, die sich jeweils auf genau einen *Service* beziehen. Die Leistungskontierungen eines Mitarbeiters erfolgen nicht auf den Arbeitsauftrag, sondern direkt auf den betroffenen Service. Über diese Kontierung läßt sich eine Beziehung zu dem Auftrag herstellen, über den die kontierte Leistung abgerechnet werden soll.

Bestellungen können nur ausgelöst werden, wenn ihnen ein aktivierungspflichtiger Auftrag zugrunde liegt, der sich auf eine Investition bezieht. Eine Investition kann für mehrere Aufträge gelten. Ein *Auftrag* kann mehrere *Bestellungen* auslösen. Um diese Bestellungen einem *Service* zurechnen zu können, muß es eine Beziehung zwischen einer Investition und einem Service geben. Für einen Service (z.B. das Projekt P350 - Einführung SAP R/3) kann es mehrere Investitionen geben. Eine Investition kann jedoch nur für einen Service gelten.

Eine Planung kann für alle Objekte angelegt werden, für die in dem Objekt *Planwert* ein Fremdschlüssel vorgesehen ist.

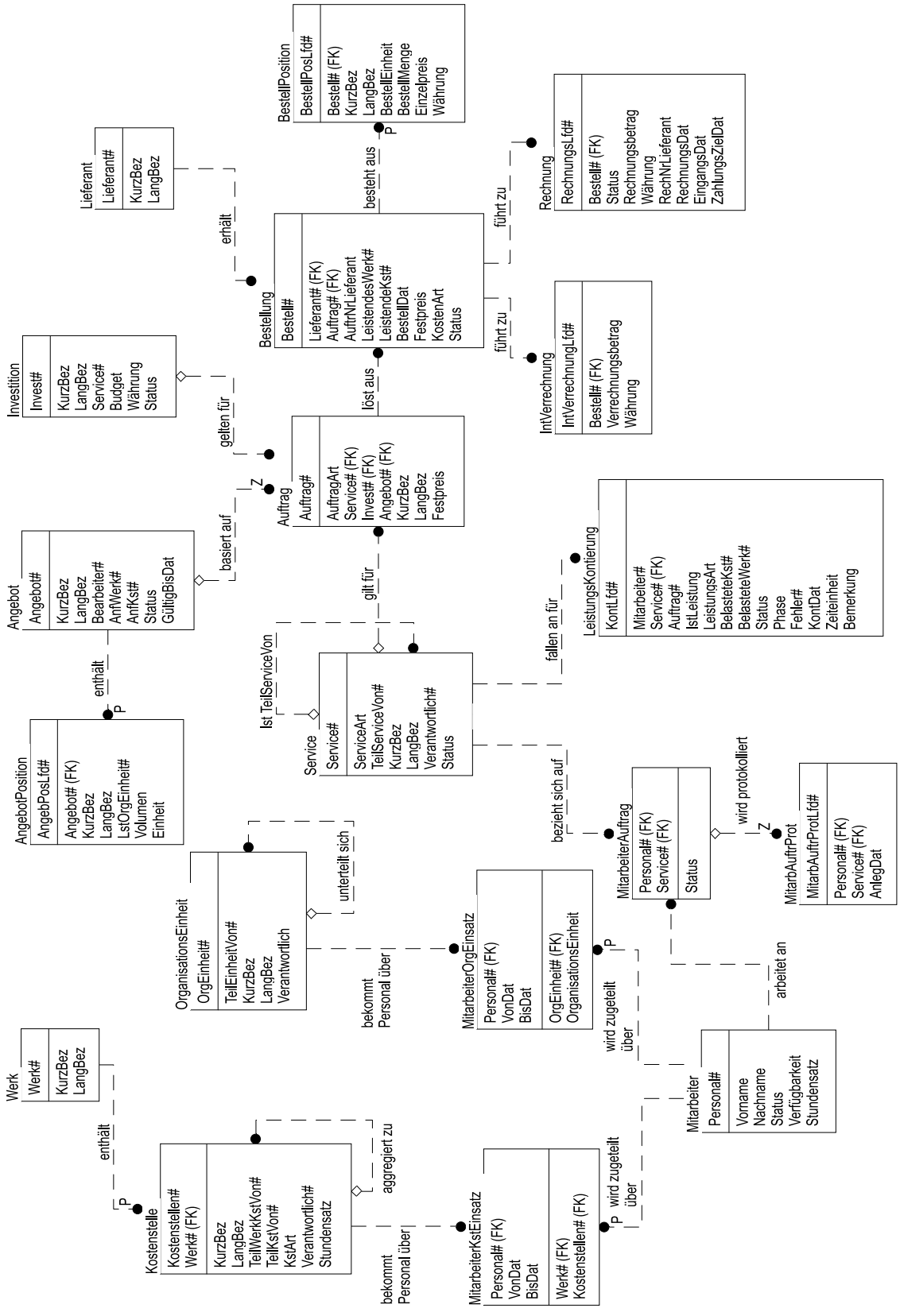

Abbildung 11: Gesamtmodell ohne Plankomponente und Hilfstabellen

5.9.2 Datenherkunft und -pflege

Einige der in diesem Modell beschriebenen Objekte werden auch von anderen Softwareprodukten abgebildet. Die Daten eines Mitarbeiters beispielsweise werden bereits in der Personalverarbeitungssoftware verarbeitet. Dort werden sie auch von den zuständigen Sachbearbeitern gepflegt. Betroffen von dieser Pflege in Fremdsystemen sind folgende Objekte:

- Mitarbeiter
- MitarbeiterOrgEinsatz
- MitarbeiterKstEinsatz
- Werk
- Kostenstelle
- OrganisationsEinheit
- Auftrag
- Investition
- Lieferant
- Bestellung
- Rechnung

Auch die Fehlertabelle kann in einem speziellen System wie CCM[24] extern gehalten werden.

In mehreren Fällen wird es notwendig sein, mit aus anderen Systemen *replizierten* Daten zu arbeiten, wenn aus datenschutzrechtlichen, technischen oder Perfomancegründen[25] ein direkter Datenzugriff nur mit hohem Aufwand oder gar nicht möglich ist. In diesem Fall muß jedoch sichergestellt werden, daß regelmäßig ein automatisches Update der Daten durchgeführt und eine eigene Pflege dieser Daten im IV-Controllingsystem unterbunden wird.

[24] CCM ist ein System zur Fehlerverwaltung

[25] Die Performance wird in vielen Fällen wohl den Aussschlag geben

5.9.3 Erfassungsmasken und Algorithmen

Das Datenmodell ist so angelegt, daß viele individuelle Vorlieben und Gewohnheiten der Anwender realisiert werden können. So ist beispielsweise die Verrechnung von Personalstunden wahlweise über den Stundensatz des einzelnen Mitarbeiters oder über den Stundensatz der Kostenstelle möglich. Ebenso können die Berechtigungen zur Einteilung von Mitarbeitern zu Arbeitsaufträgen über die Maskenlogik gesteuert werden. Kontierungsfelder wie Phasennummer oder Fehlernummer können, müssen aber nicht verwendet werden. Je nach Anwendungsgebiet, Serviceart oder Benutzer können diese Felder ausgeblendet werden. Auch die Anzeige von Informationen wie *Summe der geleisteten Stunden für diesen Auftrag* oder *Summe der noch möglichen Stunden bis zur Planerfüllung* bei der Leistungskontierung, hängt im Sinne einer möglichen Verhaltenssteuerung der Mitarbeiter von den Vorstellungen des jeweiligen Vorgesetzten ab.

Dieser Teil des IV-Controllingsystems wird hier nicht spezifiziert. Zu diesem Punkt ist eine umfassende Abstimmung mit Anwendern und Vorgesetzten im konkreten Anwendungsgebiet notwendig. Der Entwurf des Datenmodells ist so allgemein gehalten, daß zahlreiche Implementierungsvarianten des IV-Controllingsystems möglich sind.

Die Einhaltung der Integritätsbedingungen muß an zwei wesentlichen Stellen sichergestellt werden. Bei der Datenerfassung müssen Algorithmen durch die Erfassungsmasken angestoßen werden, die die Zulässigkeit der erfaßten Daten überprüfen und abgeleitete Daten wie die KostenstellenArt (vgl. Abschnitt 5.2) oder die Service# in einer *LeistungsKontierung* ermitteln. Zur Kontrolle des Datenbestandes sollten Trigger eingesetzt werden [LSK97].

5.9.4 Vergleich mit dem Referenzmodell von Spitta

Spitta hat in [Spi96a], [Spi97] und [Spi98c] ein verallgemeinertes Referenzmodell für ein Leistungserfassungssystem in der Softwareentwicklung und Wartung vorgestellt. Dieses Modell wurde evolutionär entwickelt und basiert auf Erfahrungen, die Spitta über 6 Jahre lang mit dem Einsatz eines solchen Systems gemacht hat. Das System war dabei an 5 Standorten im Einsatz und erfaßte die Leistung von 40 Entwicklern. Spitta sieht darin ein „gewisse Streuung an empirischer Erprobung“, die das Design seines Systems über die Fallanwendung hinaus allgemein empfiehlt (vgl. [Spi97], S. 107). In diesem Abschnitt soll Spittas Modell mit dem vorgestellten IV-Controllingsystem verglichen werden, um zu überprüfen, ob Spittas Anspruch, ein Referenzmodell entworfen zu haben, durch diese Fallstudie bekräftigt wird.

Prinzipiell gibt es keine bedeutenden Unterschiede zwischen den beiden Modellen. Es fällt lediglich auf, daß Spitta vier Entitäten (*ApplSystem, ApplPartSystem, Program*

und *IsService*) verwendet, um den zentralen Arbeitsgegenstand einer IV-Abteilung, eine zu entwickelnde oder zu wartende Software, abzubilden. Lediglich *IsService* enthält dabei eine rekursive Beziehung. Dadurch wird die mögliche Systemstruktur bewußt auf zwei bzw. drei Ebenen begrenzt. Diese Beschränkung beruht auf Erfahrungen mit Aufwandserfassungssystemen in einem Großkonzern (Chemie) und einem mittelständischen Konzern (Textilien). Eine tiefer geschachtelte Struktur ist nach Spittas Erfahrungen unübersichtlich und führt zu Zuordnungsfehlern (vgl. [Spi96a], S. 476f).

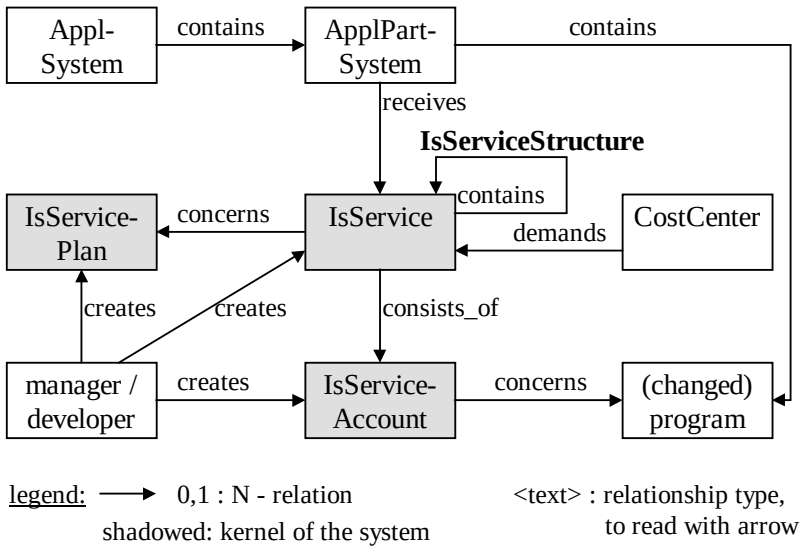

Abbildung 12: Spittas Datenmodell zur Leistungserfassung. Quelle: [Spi98c], S. 6

In dem in dieser Arbeit vorgestellten Konzept wird das zu wartende oder zu entwickelnde System über die eine Entität *Service* abgebildet. Die maximale Tiefe eines so aufgebauten Servicestrukturbaumes ist durch Integritätsbedingungen und die Definition von Servicearten steuerbar (siehe Abschnitt 5.1). Da Spittas System für den Einsatz in der Softwareentwicklung konzipiert wurde, ist seine exaktere Abbildung des zentralen Entwicklungs- und Wartungsgegenstandes erklärbar. Spitta berücksichtigt in seinem Modell Fremdbezug nur eingeschränkt. Durch den von ihm vorgesehenen Wert `purchase` des Attributes EffortClass wird die Beschaffung von Anlagegütern nicht berücksichtigt. Die Planungskomponente und Anbindung an das betriebliche Rechnungswesen sind nur ansatzweise modelliert, lassen sich aber ausbauen.

Betrachtet man die Teilkomponente Aufwandserfassung, dann läßt sich auch mit dieser Fallstudie die Position untermauern, daß es sich bei Spittas Modell um ein Referenzmodell handelt. Die an das hier beschriebene IV-Controllingsystem gestell-

ten Anforderungen gehen jedoch über eine reine Aufwandserfassung in der Softwareentwicklung und Wartung hinaus. Sie fordern an Stellen wie der Anbindung zum Rechnungswesen oder der Plankomponente eine exaktere Modellierung und an anderen Stellen wie der Servicestruktur wiederum eine größere Abstraktion von der Softwareentwicklung als zentralem Arbeitsgegenstand hin zu einer allgemeineren Serviceleistung.

5.10 Ausblick und Schlußbetrachtung

Das in dieser Arbeit vorgestellte Konzept für ein IV-Controllingsystem reicht noch nicht aus, um umgehend mit einer Implementierung beginnen zu können. Da das System sehr allgemein entworfen wurde, müssen mehrere individuelle Anforderungen des Anwenders noch genauer spezifiziert werden. Das trifft insbesondere auf den Entwurf der Erfassungsmasken zu. Aus diesen individuellen Anforderungen lassen sich auch weitere Integritätsbedingungen ableiten. Die in diesem Kapitel aufgezeigten Integritätsbedingungen sollten - ebenso wie die Werte der Aufzählungstypen - mit dem jeweiligen Anwender diskutiert und ggf. ergänzt werden.

Ebenfalls anwenderspezifisch zu klären ist die Herkunft bestimmter Daten, die aus anderen Systemen angeliefert werden. Auch die Schnittstellen zu diesen Systemen müssen abhängig von der Systemwelt des Anwenders spezifiziert werden. In diesem Zusammenhang stellt sich auch die Frage, welche Teilsysteme bereits am Markt vorhanden sind und ggf. zugekauft werden können.

Ist ein Intranet vorhanden, ist es vorstellbar, die Leistungserfassungsmaske auf der Basis eines Java Applets zu realisieren. Derzeit scheint die Performance eines solchen Applets mit Datenbankzugriff noch nicht ausreichend, um eine Leistungserfassung komfortabel durchzuführen.

Das Ziel, ein Konzept für ein vielfältig einsetzbares IV-Controllingsystem zu entwickeln, ist erreicht worden. Durch die Verwendung von Schlüsseltabellen und die Möglichkeit, Integritätsbedingungen zu ändern, sind zahlreiche Konfigurationsmöglichkeiten gegeben, um eine individuelle Anpassung an die Wünsche eines Anwenders zu ermöglichen. Es ist sicherlich kein universell einsetzbares System entstanden. An einen potentiellen Anwender werden einige Anforderungen gestellt, die erst ab einer bestimmten Unternehmensgröße anzutreffen sind. Dazu zählt beispielsweise das auftragsbezogene Arbeiten und das Vorhandensein einer Softwareinfrastruktur wie sie unter anderem durch SAP R/3 bereitgestellt wird, bei der ein Großteil der Stammdaten nicht vom IV-Controlling Personal gepflegt werden muß.

Aus den letzten Abstimmungsgespächen mit dem Erstanwender ist erkennbar, das auch die Form der Darstellung des Modells richtig gewählt wurde. Unklarheiten bei

der Vorstellung des Modells beschränkten sich primär auf kleinere Unterschiede in Begriffsdefinitionen, die schnell geklärt werden konnten und nicht aus der eigentlichen Darstellung des Modells resultierten. Inwieweit die sehr formale Definition von Integritätsbedingungen tatsächlich notwendig war wird sich erst bei der Implementierung zeigen. Dem Autor hat sie jedenfalls geholfen, genauer über die verbale Formulierung der Integritätsbedingungen nachzudenken.

6 Veni, vidi, vici - schön wär's

Dieses Kapitel fällt ein wenig aus dem Rahmen. Die primären Ziele dieser Arbeit sind mit Abschluß des fünften Kapitels bereits erreicht worden. Die Arbeit hätte an dieser Stelle demnach beendet sein können. Es ist aber durchaus wichtig, sich nicht nur mit der Lösung eines Problems zu beschäftigen. Die vorhergehenden Kapitel erwecken möglicherweise den Anschein, als sei der Autor dahergekommen, habe das Problem kurz betrachtet, in Teilprobleme zerlegt und das Datenmodell in einem Rutsch zu Papier gebracht. Die Realität sah natürlich etwas anders aus. Das beschriebene Modell ist das Ergebnis eines langwierigen Abstimmungsprozesses, in dessen Verlauf sich zahlreiche Modellvarianten als unbrauchbar oder falsch erwiesen. Ob das endgültige Datenmodell nun *richtig* ist - wobei man darüber streiten kann, was denn in diesem Zusammenhang *richtig* ist - wird sich, wenn überhaupt, mit der Einführung des Systems zeigen.

6.1 Keep Talking

Wie kam es dazu, daß nicht gleich das endgültige Modell gefunden wurde? Warum mußten so viele Umwege gegangen werden, um zum Ziel zu kommen? Die Antwort ist ebenso simpel wie alt: Menschen haben miteinander kommuniziert. Menschen mit unterschiedlichen Vorkenntnissen, Denkweisen und Begrifflichkeiten.

Mißverständnisse sind in einer solchen Situation fast unausweichlich. Das beginnt bei der Wahl von Begriffen und der Semantik, die diese Begriffe individuell für die Beteiligten haben. Zu großer Verwirrung führten anfänglich die unterschiedlichen Vorstellungen darüber, was unter einem *Arbeitspaket* zu verstehen ist. Es ist durchaus sinnvoll, im Anfangsstadium eine Begriffsanalyse durchzuführen und eine Fachterminologie einzuführen (vgl. [Mor98], S. 37ff). Gleiches trifft auf die Verwendung von Beispielen und Testdaten zu. Stammen diese nicht aus dem Unternehmen, geht für die Beteiligten der Bezug zum Problem verloren. Bei Präsentationen beschäftigen sich die Beteiligten unter Umständen mehr damit, das Beispiel zu verstehen, als sich um die dargestellten Inhalte und Zusammenhänge zu kümmern.

Sind die Begrifflichkeiten geklärt, ergibt sich das nächste Problem bei der Schilderung der Zusammenhänge. Für ein Problem gibt es unterschiedliche Sichtweisen:

1. Die Vorstellung des Vorgesetzten über die Arbeitsabläufe und Zusammenhänge in den ihm unterstellten Bereichen
2. Die Schilderung der Abläufe und Zusammenhänge aus Sicht der Mitarbeiter
3. Die von Außenstehenden beobachteten Abläufe und Zusammenhänge

Keine dieser Sichtweisen kann allein zu einem die Realität korrekt abbildenden Datenmodell führen. Der Vorgesetzte gibt Richtlinien vor, ist aber nicht in das Tagesgeschäft seiner Mitarbeiter involviert. Eventuell hat er sogar nie selber die Aufgaben seiner Mitarbeiter durchgeführt und kennt sich beispielsweise auch nicht in der Bedienung der verwendeten Werkzeuge aus. Die Mitarbeiter wiederum halten kleinere Probleme und viele nicht offizielle „Kunstgriffe“, mit denen sie Schwierigkeiten im Tagesgeschäft umgehen, von ihrem Vorgesetzten fern. Weicht ihr Handeln von der Vorstellung des Vorgesetzten ab, versuchen sie nicht aus eigenem Antrieb diese Vorstellung zu korrigieren. Das wäre einerseits anstrengend und könnte andererseits von Vorgesetzten auch falsch verstanden werden.

Die Schilderungen der Mitarbeiter sind für die Standardfälle sicherlich zutreffend, doch auch bei diesen Schilderungen existiert eine große Portion impliziten Wissens, das nur situationsbezogen preisgegeben wird. Viele Sonderfälle, die für das Datenmodell von entscheidender Bedeutung sein können, werden möglicherweise gar nicht, nur auf direkte Nachfrage, oder erst sehr spät dargestellt. Hinzu kommt, daß der einzelne Mitarbeiter teilweise eine eingeschränkte Sicht des Problems hat. Größere Zusammenhänge mit Bereichen, die im Tagesgeschäft nicht mit ihm zusammenarbeiten, bleiben möglicherweise unberücksichtigt.

Ebenso bekommt der außenstehende Beobachter nur einen bestimmten zeitlichen und inhaltlichen Ausschnitt der Realität präsentiert. Dinge, die für einen Mitarbeiter selbstverständlich sind, entgehen ihm. Fragen, die er nicht stellt - oder nicht meint stellen zu müssen, da er *glaubt* den Zusammenhang verstanden zu haben - werden ihm auch nicht beantwortet.

All diese Verhaltensmuster sind in vielen Situationen anzutreffen. Vermeidbar sind sie kaum, dem Tagesgeschäft schaden sie auch nicht. Sie wirken sich an anderen Stellen wie der Datenmodellierung aber negativ aus.

Treffendes Beispiel für einen solchen Fall bei der Entstehung dieser Arbeit ist die in Abschnitt 5.5 beschriebene Beziehung zwischen *Aufträgen* und *Services*.[26]

Aus den Aussagen :

1. Ein Auftrag kann für mehrere Arbeitspakete gelten
2. Aufträge beziehen sich eigentlich immer auf Arbeitspakete
3. Pro Arbeitspaket ein Auftrag

[26] Auch hier wird natürlich nur die subjektive Meinung des Autors wiedergegeben, der glaubt, das Problem erfaßt zu haben.

folgerte der Autor, daß es sich um eine `1:n` Beziehung zwischen *Auftrag* und *Service* handelt.

Stutzig machte ihn dann einige Zeit später die Aussage: *„Bei einem Auftrag kann man entweder die Nummer des Projektes, oder eine Investitionsnummer eingeben.“*

In einer Diskussion über einen ganz anderen Themenbereich, den Fremdbezug, stellte sich heraus, daß der Fremdschlüssel, über den die Beziehung zwischen *Service* und *Auftrag* gesteuert wird, nicht im Objekt *Service* sondern im Objekt *Auftrag* abgelegt ist. Plötzlich wurde aus der `1:n` Beziehung eine `n:1` Beziehung. Auf die Frage, wie denn die in Aussage 1 beschriebene Zuordnung zu mehreren Arbeitspaketen realisiert wird, kam die Antwort: *„Dann trage ich die Nummer des nächsten Knotens ein“*. Das bedeutet, daß der Auftrag der übergeordneten *Teilaufgabe* zugeordnet wird, was nach anderen Aussagen nicht gemacht wird: *„ Aufträge gelten nicht für Teilaufgaben, sondern für Arbeitspakete.“*

Ebenso stellte sich sehr spät - genauer gesagt auf der vorletzten Besprechung - heraus, daß es zu einem Arbeitspaket tatsächlich mehrere Aufträge geben kann. Die von einem Vorgesetzten, der an dieser Besprechung nicht teilnahm, getroffene Aussage 3 wurde plötzlich relativiert: *„Das dürfen Sie nicht so genau nehmen, was Herr X sagt. Er meint, es gibt nur einen Auftrag pro Arbeitspaket, aber im SAP machen wir das anders. Es soll zwar generell nur einen Auftrag geben, aber Herr Y z.B. legt immer mehrere an.“*

Zu einem Service läßt sich demnach nicht eindeutig ein Auftrag identifizieren. Bei zahlreichen vorhergehenden Gesprächen und der Demonstration eines Prototyps, wurde diese eindeutige Identifizierung jedoch verwendet. Die Frage: *„Und was ist, wenn ich mehrere Aufträge für ein Arbeitspaket habe?“* wurde nie gestellt, vermutlich weil der Vorgesetzte anwesend war, der kurz zuvor noch betont hatte: *„Pro Arbeitspaket ein Auftrag!“*

Hier zeigt sich, daß Aussagen von Personen, die noch nie selber ein relationales Datenmodell entworfen haben, mit äußerster Vorsicht zu genießen sind. Der Entwickler leitet aus einzelnen Aussagen sofort Beziehungen ab. Der Mitarbeiter, der diese Aussagen trifft, denkt jedoch nicht unter dem Aspekt der Datenmodellierung über die Inhalte nach und ist sich daher der Konsequenzen seiner Aussagen nicht bewußt. Alle Beteiligten glauben fatalerweise, das *Richtige* gesagt bzw. verstanden zu haben, obwohl das Gegenteil der Fall ist.

Interessant ist, daß in der ursprünglichen Fassung des Modells die später „wiedergefundene“ `n:1` Beziehung zwischen *Service* und *Auftrag* bereits vorgesehen war. Sie führte allerdings durch die Verwendung unternehmensfremder Beispiele und eine eigene Definition von unternehmensintern definierten Begriffen in ersten Besprechungen zu erheblicher Verwirrung und wurde durch die oben aufgeführten Aussagen

vermeintlich korrigiert. Auch Probleme, die durch eine mehrfach mögliche Zuordnung von Leistungen zu Aufträgen entstanden, wurden durch die Umkehrung der Beziehung gelöst. Die Korrektur wurde daher damals dankend angenommen, aber nicht intensiv genug hinterfragt.

Ein ganz anderes Problem liegt in dem Auseinanderfallen von Rahmenbedingungen und Anforderungen. Als großer Anforderungskatalog ist die Sammlung von Auswertungen und Berichten auf der Basis von EXCEL zu sehen, die bisher von Hand mit Daten gefüttert wurden. *„Das soll das System automatisch leisten!“* war eine der zentralen Anforderungen. Für ein datenbankgestütztes Berichtswesen gilt jedoch: *„Ganz oder gar nicht!“* Alle in den Berichten auftauchenden Größen müssen vom Datenmodell abgebildet und in der Datenbank gepflegt werden. Das gilt auch für Sonderfälle, wie die *Anzahl der DV-Beschäftigten in ausländischen Vertriebsniederlassungen*, bei denen bisher immer die Zahlen aus dem Vorjahr übernommen wurden. Einige in den Auswertungen auftauchende Größen wie die *Reisespesen pro Projekt* können gar nicht bekannt sein, da die Abrechnung dieser Kosten durch andere Unternehmensbereiche über ein Budget erfolgt, ohne daß ein Kostenbezug zu einem Projekt nachvollziehbar ist. Die in den bisherigen Berichten auftauchenden Werte sind vermutlich durch die bekannte $\pi * Daumen$ Formel entstanden. Solche „Kunstgriffe“ sind mit einem datenbankgestützten Berichtswesen jedoch nicht möglich.

Im Modell gibt es auch einige Zugeständnisse an den Erstanwender, die seine bisherigen Gewohnheiten berücksichtigen und auch künftig ermöglichen, obwohl diese aus Sicht des Autors vermieden werden sollten. Die monatliche Kontierung von Leistungen ist ein Beispiel für solche Zugeständnisse, die gemacht werden müssen, damit das entworfene System von den späteren Anwendern auch akzeptiert und genutzt wird und nicht als *unpraktisch und realitätsfern* abgestempelt in einem Regal verstaubt.

Die Forderung nach einen neuen System hat meist zwei Gründe. Es gibt Probleme, die mit den bisher zur Verfügung stehenden Mitteln zwar auch, aber nur sehr umständlich und unzureichend gelöst werden können. Darüber hinaus gibt es Problemstellungen, die mit der derzeitigen Softwareausstattung nicht gelöst werden können, sich in der Vorstellung des Auftraggebers mit einem neuen System aber lösen lassen müßten. Solche *Visionen* sind wichtig, denn ohne sie kann eine Weiter- und Neuentwicklung von Softwaresystemen - wenn überhaupt - nur unzureichend betrieben werden. Diese Visionen sind in der Anfangsphase sehr hilfreich, um zu verstehen, in welche Richtung sich das System entwickeln soll. Kommen jedoch zu einem späteren Zeitpunkt ständig neue Vorstellungen hinzu, kann sich dies auch negativ auf den Entwicklungsfortschritt auswirken. Die Zahl der möglichen Abstimmungstermine zwischen Entwickler und Auftraggeber sind erfahrungsgemäß begrenzt und müssen daher effizient genutzt werden. Ein ständiges *Weiterträumen* in solchen Sitzungen kann zu sehr von dem eigentlich in der Entwicklung befindlichen System ablenken. Die wichtige Verifikation des eigentlichen Modells, die eigentlich Inhalt

einer solchen Sitzung im fortgeschrittenen Entwicklungsstadium sein sollte, kommt zu kurz, wenn diese Sitzungen nicht mit konkreten Fragen und Präsentationen *beider* Seiten vorbereitet wurden. Erst im Anschluß daran sollte das begrüßenswerte *Weiterträumen* laut Tagesordnung wieder erlaubt sein.

6.2 Welcome To The Machine

Neben den Kommunikationsproblemen zwischen Menschen gibt es auch noch die Technik, die Probleme bereiten kann. In Kapitel 4 wurde schon angedeutet, daß bereits bei einem Datenmodell dieser Größe der Einsatz von Softwarewerkzeugen sinnvoll ist, insbesondere wenn neben der Dokumentation auch ein Prototyp entsteht. Diese Arbeit soll keine Werkzeugevaluation beinhalten, sondern kurz aufzeigen, welche Softwareprodukte verwendet wurden und inwieweit diese Produkte unzureichend waren.

Als Datenbanksystem für den Prototypen diente *Access 97*. Das Produkt aus dem Hause Microsoft läßt sich problemlos auf vergleichsweise leistungsschwachen Rechnern[27] installieren. Von Microsoft als „Datenbank für Jedermann“ angepriesen, erlaubt Access dank der von anderen Office Produkten bekannten Benutzeroberfläche einen einfachen Einstieg in die Datenbankwelt, der auch schnell zu Resultaten führt. Darin verbirgt sich allerdings eine große Gefahr: Viele entscheidende Einstellungen werden von Access automatisch, ohne den Benutzer zu informieren vorgenommen. Diese Arbeitserleichterung führt dazu, daß der Benutzer möglicherweise erst sehr spät, oder sogar gar nicht merkt, daß Datenbankobjekte andere als von ihm gewünschte Eigenschaften haben. Eine manuelle Konfiguration solcher Eigenschaften ist lästig, da die Benutzerführung in diesem Fall zahlreiche Wechsel von Maus und Tastatur erfordert. Schmerzlich wird ein Data-Dictionary vermißt, daß die Konfiguration und Dokumentation von Datentypen erheblich vereinfachen würde.

Access bietet in der aktuellen Version zahlreiche Möglichkeiten, um schnell funktionsfähige Prototypen zu erstellen, da sich durch die Integration der ereignisorientierten Sprache *Visual Basic* auch Erfassungsmasken und aufwendige Algorithmen realisieren lassen. Über eine ODBC Schnittstelle ist eine Anbindung an andere Datenbanksysteme möglich, so daß sich das Berichtssystem von Access auch als einfache Auswertungskomponente verwenden läßt.

Problematisch, was die Portabilität einer unter Access entworfenen Datenbank angeht, sind die Namenskonventionen. Access erlaubt in Tabellenbezeichnungen beispielsweise Umlaute und Blanks, die in anderen Datenbanksystemen nicht zulässig sind. Unter den Gesichtspunkten Performance und Datensicherheit betrachtet, ist Access einer Informix oder Oracle Datenbank immer noch unterlegen.

[27] Ein Pentium mit 32 MB RAM sollte es schon sein.

Wichtigste Erkenntnis: Resultate erreicht auch ein ungeübter Anwender schnell. Ob diese Ergebnisse auch *richtig* sind, steht auf einem ganz anderen Blatt. Access erlaubt viele Fehler, die dem Anwender gar nicht bewußt werden. Für einen erfahrenen Anwender ist Access 97 ein nützliches Datenbanksystem zur Erstellung eines Prototyps.

Als graphisches Datenbank Design Tool wurde ERWinSQL eingesetzt. Zwei Versionen standen zur Verfügung: *ERWinSQL 2.2a* von 1993 und die auf fünf Entitäten beschränkte Demoversion *ERWin for SQLWindows 2.1* [28] von 1995. Bereits die 93er ERWinSQL Version beeindruckt durch zahlreiche Konfigurationsmöglichkeiten für Datentypen und die Datenintegrität. Die 95er Version bieten hier noch mehr Möglichkeiten.

ERWinSQL bietet eine bidirektionale Schnittstelle zu Access, über die der ERWinSQL Entwurf und die Accessdatenbank jederzeit synchronisiert werden können. Die Schnittstelle zu Access ist allerdings eine der unwichtigeren. Wichtiger für den professionellen Anwender sind sicherlich die Schnittstellen zu Datenbanksystemen wie Informix, Oracle oder DB2. ERWinSQL wird auch von Sommer [Som96] empfohlen, der ERWinSQL für das Datenbankdesign von Access-Datenbanken[29] eingesetzt hat.

Leider sind die Vertriebswege von ERWinSQL in Deutschland nur sehr dürftig. [30] Eine direkte Faxanfrage nach einer Updatemöglichkeit und einem Händlerverzeichnis blieb vom Hersteller unbeantwortet.

Als Alternative zu ERWinSQL wurde versucht *VISIO Professional 5.0* einzusetzen. VISIO hat sich bei der graphischen Darstellung von Geschäftsprozessen und Organisationsstrukturen einen Namen gemacht und kann in diesem Bereich als etablierte Software mit hohem Verbreitungsgrad angesehen werden. Die Version 5.0 Professional beinhaltet die Möglichkeit, ein ERM-Modell nach dem IDEF1X Schema zu erstellen. Über die ODBC-Schnittstelle ist ein Zugriff auf Access Datenbanken möglich, über den die in VISIO entworfene Datenbank in Access angelegt werden kann. Dabei kam es aber immer wieder zu unbrauchbaren Fehlermeldungen wie: *„Nicht alle Objekte konnten angelegt werden“*. Nach Veränderungen der Datenbank ließ sich die Grafik nur manuell anpassen.

Realisiert wurden diese Funktionalitäten über ein Makro, dem man in der deutschen Version eine hastige und oberflächliche Übersetzung aus dem Englischen anmerkt. Etwas stiefmütterlich ist auch die Behandlung in den Handbüchern. Lediglich in

[28] Bei beiden Programmen handelt es sich um 16bit Anwendungen für Windows. Scheinbar hat zwischenzeitlich ein Namenswechsel und ein Neubeginn der Versionszählung stattgefunden.

[29] Auch Sommer hält Access für ein mit Vorsicht zu genießendes Datenbanksystem (vgl. [Som96] S.19)

[30] Der Hersteller Logic Works scheint inzwischen auch von einer anderen Gesellschaft übernommen worden zu sein.

Kurzreferenz wird die Datenbankkomponente ansatzweise angesprochen. Im eigentlichen Handbuch sucht man vergeblich nach einer Dokumentation. In der derzeitigen Form ist diese VISIO-Komponente nicht zu empfehlen.

6.3 Near The End

Nach der Schilderung einiger Probleme, die es bei der Konzeption des Datenmodells gegeben hat, muß man sich fragen: *„Was lernen wir daraus?“*

Softwareseitig hat sich eine altbekannte Erkenntnis bekräftigt: Die verbreitetsten Systeme leisten oft weniger als sie vorgeben oder scheitern im Zusammenspiel mit anderen Softwareprodukten an Versionsinkompatibilitäten. Daß es besser geht, zeigt ERWinSQL. Doch was gut ist muß noch lange nicht erfolgreich sein. Viele ERWinSQL Anwender scheint es in Deutschland nicht zu geben, andernfalls wäre der Vertrieb und Support besser organisiert.

Auch was den Kommunikationsprozeß mit dem Anwender betrifft wurden bekannte Erfahrungen bestätigt. In diesem Fall ist es allerdings gut gelaufen. Bei dem Erstanwender des IV-Controllingsystems standen drei kompetente und engagierte Ansprechpartner zur Verfügung, die sehr genau wußten, was sie wollten. Solch eine hervorragende Ausgangsposition findet sich in den wenigsten Fällen. Trotzdem verlief der Abstimmungsprozeß nicht völlig problemlos. Mißverständnisse zwischen Menschen, die ein Problem aus unterschiedlichen Sichten betrachten, sind letztlich unvermeidlich. Da aber unterschiedliche Sichtweisen notwendig sind, um die Vielzahl der Aspekte eines Problems zu berücksichtigen ist es wichtig, sich immer im klaren darüber zu sein, daß das Problem vom Entwickler möglicherweise nicht richtig erfaßt wurde.

Die in Abschnitt 6.1 geschilderten Situationen wird der Autor bei künftigen Sitzungen sicherlich wieder in einem anderen Licht sehen. Die Abstimmung mit einem Anwender ist eben kein linearen, sondern ein eher zyklischer Prozeß, bei dem sich der Entwickler vor und nach jedem Schritt fragen muß: *„Habe ich das Problem wirklich verstanden?“*

Literaturverzeichnis

[Bec90] Becker, W.: Funktionsprinzipien des Controlling. In: ZfB 60 (1990) 3, 295-318

[Boe76] Boehm, B.W.: Software Engineering. In: IEEE Transactions on Computers 25 (1976) No. 12, S. 1226-1241.

[Boe81] Boehm, B.W.: Software Engineering Economics. Prentice Hall, Englewood Cliffs 1981

[Dob95] von Dobschütz, L.: IV-Controlling: Theoretische Sicht und praktische Bedeutung. In: Controlling 6 (1995) 5, S. 306-312

[Gri87] Griese, J.; Obelode, G.; Schmitz, P.; Seibt, D.: Ergebnisse des Arbeitskreises Wirtschaftlichkeit der Informationsverarbeitung. In: zfbf 39 (1987) 7, S. 515-551

[Hor90] Horváth, P.: Effektives Informationscontrolling. In: Office Management 1-2/1990, S. 12-15

[Hor96] Horváth, P.: Controlling. 6. Auflage. Vahlen, München 1996

[Kar97] Kargl, H.: Der Wandel von der DV-Abteilung zum IT-Profitcenter. In: Arbeitspapiere WI, 1/1997, Hrsg.: Lehrstuhl für Allg. BWL und Wirtschaftsinformatik, Johannes Gutenberg-Universität: Mainz 1997, im Internet erhältlich unter http://wi.bwl.uni-mainz.de/apap/kaapvd.htm (Stand 10/98)

[Knu84] Knuth, D. E.: Literate Programmig; The Computer Journal, 27 (2) 1984, S. 97-111

[Krc90] Krcmar, H.: Informationsverarbeitungs-Controlling – Zielsetzung und Erfolgsfaktoren. In: Information Management (1990) 3, S. 6-15

[Krc92a] Krcmar, H.: Informationsverarbeitungscontrolling in der Praxis. In: Information Management (1992) 2, S. 6-18.

[Krc92b] Krcmar, H.: Leise Töne im Informationsmanagement des Mittelstandes. In: Information Management 7 (1992) 4, S. 79-83

[Krc97] Krcmar, H.: Informationsmanagement, Springer Verlag, Berlin, Heidelberg, 1997

[KrFr91] Krause, O.C.; Fröhling, C.: DV-Controlling für Rechenzentren. In: Controlling 3 (1991) 5, S. 270-277

[Krü96] Krüll, J.: Die Erfassung informationstechnischer Basisdaten. Diplomarbeit, Universität Bielefeld, Fakultät für Wirtschaftswissenschaften, April 1996

[Kuhl98] Kuhlmann, A.: Datenanalyse zum IV-Controlling unter Berücksichtigung von Werkzeugaspekten. Diplomarbeit, Universität Bielefeld, Fakultät für Wirtschaftswissenschaften, Juli 1998

[KWZ90] Küpper, H.-U.; Weber, J.; Zünd, A.: Zum Verständnis und Selbstverständnis des Controlling. In: ZfB 60 (1990) 3, 281-293

[Küp97] Küpper, H.-U., Controlling, 2 Auflage, Stuttgart, Schäfer-Poeschel, 1997

[LSK97] Lufter, J.; Schaarschmidt, R.; Küspert, K.: Aktive Datenbankmechanismen. In: HMD 195 (1997), S. 102-127

[Mol94] Moll, K.-R.: Informatik-Management. Springer, Berlin/Heidelberg et al. 1994

[Mor98] Mordau, J.: Die Integration formaler Methoden zur Spezifikation von Informationssystemen. Dissertation, Universität Bielefeld, Fakultät für Wirtschaftswissenschaften, Juni 1998

[Saa93] Saake, G. Objektorientierte Spezifikation von Informationssystemen. In: Volume 6 of Teubner-Texte zur Informatik. Teubner, Stuttgart, Leipzig, 1993

[Sch90] Schwarze, J. Betriebswirtschaftliche Aufgaben und Bedeutung des Informationsmanagements. Wirtschaftsinformatik 32 (1990) 2, S. 104-105

[SEK98] Spitta, Th.; Ellerbrock, R.; Kuhlmann, A.: IV-Controlling und Informationsmanagement im Mittelstand - Abschließende Ergebnisse einer Feldstudie, September 1998, eingereicht ZfB, 17 Seiten

[SoKr90] Sokolowski, Z.: Kraemer, W.: Controlling der Informationsverarbeitung. In: Information Management 5 (1990) 3, S. 16-27

[Som96] Sommer, M.: Datenbankdesign für Access 7.0/2.0. International Thomson Publishing, Bonn 1996

[Spi89] Spitta, T.: Software Engineering und Prototyping - Eine Konstruktionslehre für administrative Softwaresysteme. Springer, Berlin - Heidelberg et al. 1989

[Spi96a] Spitta, T.: Die Aufwandserfassung von IS-Dienstleistungen In Wirtschaftsinformatik 38 (1996) 5, S. 473-484.

[Spi96b] Spitta, T.: Wiederverwendbare Attribute als Ordnungsfaktor der Unternehmensdaten. In: Ortner, E.; Schiennmann, B.; Thoma, H. (Hrsg.): Natürlichsprachlicher Entwurf von Informationssystemen, Workshop Mai '96 Tutzing, Universitätsverlag, Konstanz 1996, S. 79-93.

[Spi97] Spitta, T.: Die Gewinnung korrekter Daten aus manuellen Aufschreibungen. In: Grün, O.; Heinrich, L.J. (Hrsg): Empirische Forschung in der Wirtschaftsinformatik, Springer, Wien - New York 1997, 105-118

[Spi98a] Spitta, T.: IV-Controlling in mittelständischen Industrieunternehmen – Ergebnisse einer empirischen Studie. In: Wirtschaftsinformatik 40 (1998) 5, S. 424-433

[Spi98b] Spitta, T.: Personal IS-Effort Collection in Industrial Environments, 9th Intern. Symp. on Software Reliability Engineering (ISSRE 98), Paderborn, November 1998, Industrial Practices, S. 111-120

[Spi98c] Spitta, T.: Data Collection of Development and Maintenance Effort, Discussion Paper No. 401, Universität Bielefeld, Fakultät für Wirtschaftswissenschaften, August 1998

[Wol98] Wolf, P.Was hat es mit *literate programming* auf sich?, Arbeitspapier, Universität Bielefeld, Fakultät für Wirtschaftswissenschaften, Oktober 1998

Name: Sigge
Vorname: Arne-Christian

Versicherung

Ich versichere, daß ich die Diplomarbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe.

Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen habe ich als solche kenntlich gemacht.

Bielefeld, den 11. Januar 1999

Arne-Christian Sigge