4.5.3 Kritik der alten Lösung anhand dieser Lösung

Wenn Sie das erwähnte Buch kennen sollten, wird Ihnen auffallen, dass es erneut Unterschiede zwischen unserer Lösung hier und der des Buches gibt, die nicht nur auf reinen Ergänzungen beruhen, wie sie aus einer Realisierung folgen. Eine solche Ergänzung ist beispielsweise das Attribut „Phonetisch", das die Datei „Schlagwort" im Buch mitführt und das dazu dient, die Wort-Ähnlichkeiten einfacher zu finden. Diese Datei „Schlagwort" stimmt ansonsten mit unserer Lösung gut überein.

Die Datei „Gitter" weicht jedoch erheblich von unserer Lösung ab. Im Buch wurde sie eher beiläufig erwähnt als Datenspeicher für die Schlagwort-Verknüpfungen. Wie und wer diese durchzuführen hat, war dort von untergeordnetem Interesse, da der Tenor die Bewertung die Datei, nicht ihre Festlegung als Objekt war. Deshalb wurde dort die Datei ganz ähnlich der Impuls-Datei unseres kleinen Tools angelegt, als Beziehung zwischen zwei Schlagworten mit Reihenfolge und einem Verknüpfungstyp. Eine Realisierung unseres Gitter-Objektes würde ähnlich aussehen, um „Wortsinn"-Vergleiche, „Hierarchie" und „Verbindung" darstellen zu können, doch würde aufgrund unserer tiefergehenden Analyse diese Beziehungen nicht nur zwischen Schlagworten selbst notwendig werden, sondern sicher auch zwischen ihren Spezifikationen.

Die Datei „Kommentar“ im Buch führt neben ihrer Identifikation und derjenigen ihres übergeordneten Objektes „Dokument“, zu dem sie als Komponentenobjekt gehört, noch ein Textfeld „Anmerkung“ auf, das den eigentlichen Kommentar enthält und in unserem Begriff „Kommentar“ als verbale Darstellung analog der Bezeichnung untergebracht war. Wie unser Objekt enthält sie „Vermerk“, dort „Status“ genannt, „Termin“ und „Priorität“, sie weist daneben jedoch noch Teile unserer „Externen Daten“ auf. Da der Kommentar einem Dokument eindeutig zugeordnet ist, das diese Daten für sich beanspruchen kann, sollte dies eigentlich redundant und damit überflüssig sein.

Im Buch wurde jedoch gefordert, dass die Anwender diese Attribute unabhängig vom Dokument verändern dürfen, weshalb die Lösung im Buch für diese Zwecke korrekt ist.

In unserem Schreiben an Frau Dokumentenkundin, das für unsere Analyse maßgeblich war, ist dies jedoch nicht angedeutet worden, sodass wir bei unserer Lösung bleiben. Damit können die Anwender einem Dokument zuständige Personen und Produkte beifügen, nicht jedoch gesondert einem Kommentar. Dieser bezieht sich für seine Zuständigkeiten demnach immer auf Personen und Produkte des Dokumentes.

Unser Objekt „Externe Daten“ existiert im Buch nicht, dort wurden dessen Attribute mehr oder minder zufällig auf die Dokumenten- und die Kommentardatei verteilt.

Die Datei „Dokument“ hat ansonsten recht viel Ähnlichkeit mit unserem Objekt. Neben ihrem Identifikator, der verbalen Darstellung des Dokuments und einem Textfeld, das einen Teil des Dokumentes zu Prüfzwecken aufweist, enthält sie den Typ des Dokumentes und sein Format, sowie den bereits erwähnten Teil unserer „Externen Daten“. Unser „Schlagwort-Gitter“ wurde realisiert über die Zuordnung von drei Schlagworten mit ihren Spezifikationen zum Dokument.

Sie sehen, dass unsere Überlegungen zur Anordnung des Gitters im Dokumenten-Objekt auch in der „Lösung aus dem Bauch heraus“ selbst ohne differenzierte Vorgehensweise gefunden wurde?

Geringere Unterschiede nach Realisierung

Der große Unterschied zwischen Analysen, die aus der Logik des Problems stammen und solchen, die aus Erfahrung gewonnen werden, ist oft in der endgültigen, realisierten Version weitaus geringer als zu Zeiten der Konzeptionierung, denn die Realisierung zwingt allen Programmierern die Berücksichtigung von Wirkungsketten und Abhängigkeiten auf, sonst „läuft es nicht“. Gesundbeten hilft nämlich nur in der Politik, nicht am Computer.

Weitere Unterschiede treten auf in der Behandlung von Pfad und Status, die wir in das übergeordnete Objekt „Schrank“ gesteckt haben.

„Status“ ist in der im zitierten Buch vorliegenden Bestimmung des Dokumentenstatus beim Dokument nicht optimal untergebracht, da er sich, wie bereits erwähnt, nicht vollständig aus dokumenteneigenen Angaben ermitteln lässt.

Bei Pfad ist das anders. Nach unserem Schreiben an Frau Dokumentenkundin setzt sich der Pfad immer aus Dokumentangaben zusammen.

Ist also unsere oder die alte Buchlösung hier sauberer?

Nehmen wir wieder die EE-Zerlegung zu Hilfe oder mit anderen Worten, fragen wir nach dem Zweck unserer Informationsverarbeitung. Was soll „Pfad“ tun? Im Buch ist Pfad ein Attribut, das entweder eine Verzeichnisstruktur aufweist oder das Sammelbecken ist.

Das Sammelbecken ist jedoch nicht nur eine Lokalität wie die Verzeichnisstruktur, es hat eine besondere Bewandtnis, weswegen es überhaupt separat aufgeführt wurde. Es weist tatsächlich einen erheblichen Unterschied zum sonstigen Pfad auf, denn Dokumente ohne vernünftige Zuordnung zu zuständigen Personen oder

sonstigen Angaben, die für die Verzeichnisstruktur des jeweiligen Dokumententyps erforderlich sind, benötigen besondere Aufmerksamkeit, hier muss etwas getan werden.

Das ist im Buch nicht besonders hervorgehoben worden, „Sammelbecken" wurde ganz im Gegenteil wie das Server-Laufwerk zu einer Konstanten erniedrigt, die zweckmäßigerweise in einer Datei hinterlegt ist, ohne eigentlichen Objektcharakter zu haben.

Objekte versus Daten

Der Unterschied? Objekte müssen Jobs erledigen, Daten sind passiv und warten, bis sie von irgendjemand oder irgendwas abgeholt werden.

Die Aufgabe, die hinter „Sammelbecken" steht, muss also im Buch bei „Pfad" mit aufgenommen worden sein. Da Pfade für mehrere Dokumente gleich sein können und „Sammelbecken" auch die Beaufsichtung aller Dokumente ohne saubere Zuordnungen durchzuführen hat, ist die Sicht der Dinge von Pfad und Sammelbecken aber deutlich „von oben" herab, erfolgt also von einer Dokumenten-Übersicht aus, die es im Buch überhaupt nicht gibt.

Diese Übersicht wurde aber im Schreiben an Frau Dokumentenkundin klar ausgesprochen:

> Alle Dokumente auf dem vorgegebenen Server-Laufwerk sollen angezeigt werden, geänderte sowie gelöschte und neue Dokumente müssen besonders hervorgehoben werden.

Unsere Lösung liegt also genau im Rahmen, der mit der Kundin vereinbart wurde.

Damit ist unsere Lösung dem Problem besser angepasst als die intuitive Version des Buches. Wenn wir diese Dokumentenablage programmieren sollten, ist unsere differenzierte Analyse eindeutig vorzuziehen.

Vorteile der differenzierten Analyse

Ganz abgesehen davon, dass wir Frau Dokumentenkundin eine sehr detaillierte verbale Beschreibung mit klaren Erklärungen der verwendeten Begriffe liefern können, die nicht nur das „wer schreibt, der bleibt"-Prinzip gehörig unterstützt, sondern als Verständigungsangebot auch meist gerne gesehen wird, erlaubt uns die Objekthierarchie einen kompakten Objektaufbau, was unsere ausführenden Programmierer erfreuen wird, da die Arbeitsteilung klar vorhanden ist. Darüberhinaus liefert die Kenntnis der beteiligten Impulse und Threads genauso klare Testanweisungen, was unsere Qualitätskontrolle erleichtert.

All das lassen Erfahrungskonzeptionen im Alltag mehr oder minder stark vermissen und benötigen deshalb Nacharbeiten, die häufig genug erst im Verlauf der Tätigkeiten selbst durchgeführt werden, obwohl sie manchmal konzeptionelle Veränderungen notwendig machen können. Solche Eingriffe, in der Testphase erst erkannt, haben aber ihre Tücken, wie wohl jeder weiß, ganz zu schweigen von Differenzen, die erst beim Kunden selbst offenbar werden.

Vielleicht gibt es ja doch einen „Arbeitserhaltungssatz" der EDV?

Vielleicht ist es eben doch keine Zeitverschwendung,
differenzierte Analysen zu betreiben?