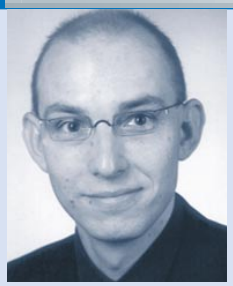
mehr zum thema:
www.systor.com

von gerald baumann

EINE OBJEKTORIENTIERTE ARCHITEKTUR ZUR ELEKTRONISCHEN ARCHIVIERUNG

Im Rahmen eines Projekts für ein großes Telekommunikationsunternehmen bestand die Aufgabe, für unterschiedliche Applikationen ein System zur Archivierung beliebiger Dokumente und Daten zu konzipieren. Der Beitrag stellt die grundsätzlichen Erwägungen zu dieser Aufgabenstellung und eine Architekturlösung vor, die auf Verallgemeinerungen objektorientierter Prinzipien beruht, wie sie im Klassendesign Anwendung finden.

der autor



Dr. Gerald Baumann (E-Mail: Gerald@Dr-Baumann.de) ist Berater für objektorientierte Softwareentwicklung und Systemarchitektur bei der Systor GmbH im Bereich E-Business.

Aufgabenstellung, Ziele und Randbedingungen

Viele Unternehmen – insbesondere solche mit Endverbrauchern als Kunden – archivieren Dokumente elektronisch, weil ein Aktenlager der entsprechenden Größenordnung nicht mehr handhabbar oder finanzierbar ist. Bei den zu archivierenden Dokumenten handelt es sich typischerweise um

- Verträge,
- Rechnungen,
- Schriftverkehr mit Kunden und Lieferanten und um
- Ähnliches.

Grundsätze ordnungsgemäßer Buchführung

Die Interpretation der rein fiskalischen *Grundsätze ordnungsgemäßer Buchführung (GoB)* – Vollständigkeit, Richtigkeit, Zeitgerechtigkeit und Ordnung – und ihre Auswirkungen auf die EDV sind ein sehr komplexes Thema (vgl. [Sch98]). Hier sei nur darauf hingewiesen, dass für erstellte Dokumente eine getrennte Archivierung von Inhalten und Dokument-Formaten nicht praktikabel ist. Zur Rekonstruktion wäre beides zu interpretieren und zusammenzuführen. Damit dabei der Grundsatz der Urschrifttreue der Dokumente erhalten bliebe, wäre man theoretisch dazu verpflichtet, die interpretierende Software, das zugehörige Betriebssystem und die Hardware gleich mitzuarchivieren.

Kasten 1

Das Archiv ist dabei oftmals die Schnittstelle zwischen verschiedenen Geschäftsprozessen. Die Einlagerung ins Archiv erfolgt durch ein anderes System und in einem anderen Geschäftsprozess als das spätere Wiederabrufen. Hierbei ist grundsätzlich im Rahmen der elektronischen Archivierung von Dokumenten nicht die Einlagerung sondern das Wiederfinden die signifikante Problemstellung.

Ein gemeinsames Charakteristikum vieler Dokumente ist, dass sie den gesetzlichen *Grundsätzen ordnungsgemäßer Buchführung (GoB)* (vgl. [Sch98]) unterliegen. Deshalb müssen sie über längere Zeiträume, in der Regel sechs oder zehn Jahre aufbewahrt werden. Aus den GoB wird die Anforderung abgeleitet, dass die elektronische Form bis in die Details des Kleingedruckten und Briefkopfes der Papierform exakt zu entsprechen hat (**siehe Kasten 1**). Sinnvollerweise wählt man deshalb für die elektronische Archivierung ein Grafik-Dateiformat – z. B. TIFF, GIF oder JPEG. Dieses Format ermöglicht zusätzlich eine analoge Archivierung von nicht selbst erstellten oder handschriftlichen Dokumenten durch Scannen.

Aus den GoB folgt weiterhin, dass es keine Möglichkeit geben darf, archivierte Dokumente zu verändern. Dokumente dürfen deshalb in einem elektronischen Archiv nur eingestellt, gelesen oder gelöscht werden. Auf diese Weise ist das Archiv revisionssicher. Eine neue Version eines Dokuments muss immer als neues eigenständiges Dokument und nicht als Update der ursprünglichen Version archiviert werden.

Zusätzlich zur elektronischen Archivierung von Dokumenten bietet es sich an, Objekte – d. h. Instanzen von fachlichen Klassen, wie z. B. Kunden-Stammdaten oder Vertragsdaten – zu archivieren. Damit ist man in der Lage, auch nach längeren Zeiträumen auf ältere Versionen dieser Objekte zuzugreifen oder eine Historie der Objekte zu erstellen. Solche Zugriffe auf „historische" Daten sind z. B. im Rahmen eines *Customer/Relationship-Management* von Interesse. Die Archivierung von Objekten sollte mit XML erfolgen (**siehe Kasten 2**). Obwohl im Weiteren zur Vereinfachung immer nur von Dokumenten gesprochen wird, sind Objekte immer gleichfalls gemeint.

Zusammenfassend gibt es also für ein elektronisches Archiv die Anforderung, dass es verschiedenste Daten aufnehmen und über längere Zeiträume revisionssicher verwahren muss. Die Daten werden von unterschiedlichen Systemen in das Archiv eingestellt und müssen bei Bedarf auch anderen Systemen zur Verfügung gestellt werden. Die Archivierungszeiten sind in der Regel sehr viel länger als die üblichen Update-Zyklen für Software und sogar größer als viele Software-Lebenszeiten. Folglich kann man davon ausgehen, dass die Systemlandschaft außerhalb des elektronischen Archivs sich in der Zeit zwischen Einlagern und Löschen eines Dokuments erheblich verändern wird.

Marktübliche Standardlösungen

Zur Archivierung von Dokumenten existiert auf dem Markt eine Reihe von Komplettlösungen von namhaften großen bis hin zu eher unbekannten mittelständischen Firmen. Zumeist werden diese Lösungen unter der Bezeichnung „Dokumentenmanagement-Systeme" angeboten. Eine Suche zu diesem Stichwort im Internet verschafft einen schnellen Überblick.

Informiert man sich genauer über die Architektur der angebotenen Systeme, so findet man im Kern fast ausschließlich eine relationale Datenbank. Diese speichert die Dokumente als *Binary Large Object (BLOB)* und zu jedem dieser BLOBs eine Reihe von Attributen, die zur Identifikation – d. h. Suche – dienen. Die Attribute in den Suchtabellen enthalten sowohl Metadaten über den BLOB als auch Daten aus dem BLOB, die also grundsätzlich einen kleinen partiellen Inhalt des BLOBs widerspiegeln. Metadaten sind Informationen wie Typ, Format und Erzeugersystem. Daten sind zum Beispiel Kundenname, Kundennummer, Rechnungsdatum etc. Der Begriff „Erzeugersystem" ist hier grundsätzlich in einem erweiterten Sinn zu verstehen. Er meint auch solche Systeme, die z. B. die Bearbeitung von eingegangenen externen Briefen oder Faxen unterstützen und diese Dokumente im Rahmen der Bearbeitung archivieren.

Gleichzeitig mit der Archivierung des Dokuments als BLOB erfolgt das Befüllen der Suchtabelle mit den Attributen. Wird ein Dokument aus dem Archiv benötigt, so erfolgt eine Suche über die Suchtabellen unter Angabe der gewünschten Attribute. Eine Übersicht über diese Architektur gibt **Abbildung 1**. Ein Langzeitspeicher in Form von Magnetbändern oder CDs ist in der Regel an die Datenbank angeschlossen. Auf diesen erfolgt eine Auslagerung der BLOBs nach einem zu definierenden Zeitraum.

Archivierung von Objekten mit XML

Die Verwendung von XML zur Archivierung von Instanzen fachlicher Klassen ist sinnvoll. Durch XML wird es möglich, Objekte zu lesen und zu interpretieren, selbst wenn nach vielen Jahren die zugehörigen ursprünglichen Applikationen oder Klassen nicht mehr existieren sollten. Hier ein selbstredendes Beispiel:

```
<Kunde>
  <Kundennummer> 42 </Kundennummer>
  <Name> Dagobert Duck </Name>
  <Adresse>
    <Strasse> Am Geldspeicher 1 </Strasse>
    <Ort> 11011 Entenhausen </Ort>
  </Adresse>
  <Kunde_seid> 24.12.2000 </Kunde_seid>
</Kunde>
```

Für Grafikdateien kann ebenfalls die Verwendung von XML sinnvoll sein, wenn man Metainformationen – wie z.B. das Grafikformat – zusätzlich archivieren will. Allerdings ist bei der Einbettung von ASCII-formatierten binären Dateien in XML zu beachten, dass innerhalb des ASCII-Codes keine durch XML belegten Zeichen wie < oder > vorkommen dürfen. Aus diesem Grund muss der ASCII-Code vor der Einbettung in XML in einen Code ohne diese Zeichen, z. B. Base64, übersetzt werden.

Ein XML-Beispiel für eine Grafik-Datei:

```
<Rechnung>
  <Grafikformat> TIFF </Grafikformat>
  <Codierung> Base64 </Codierung>
  <Datei>

Und hier der Base64-codierte Inhalt der TIFF-Datei:

  </Datei>
</Rechnung
```

Kasten 2

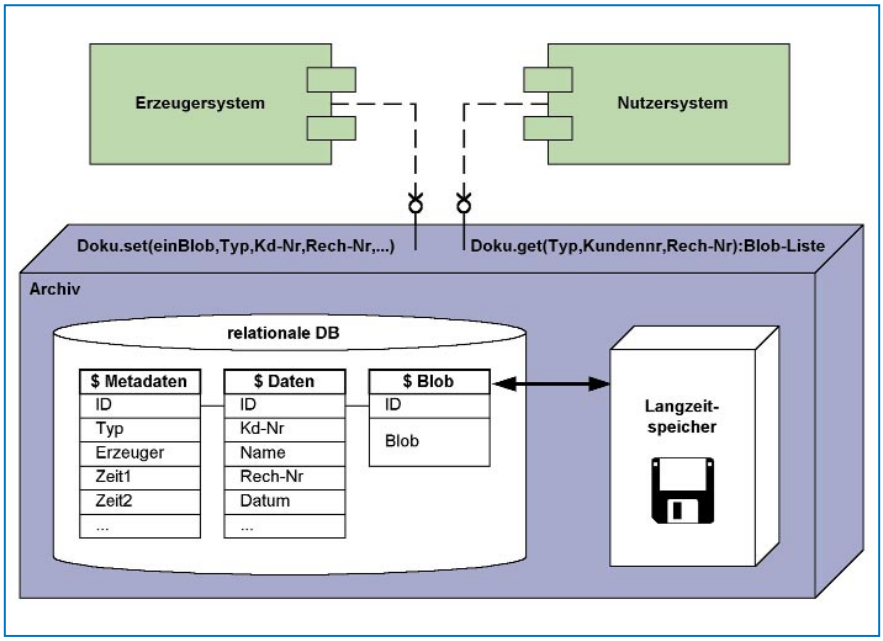
Vordergründige Entkopplung

Die zuvor beschriebene und üblicherweise verwendete herkömmliche Architektur scheint Systeme, die Dokumente erzeugen, und Systeme, die Dokumente nutzen, zu entkoppeln. Ein typisches Szenario ist eine *Customer-Care*-Applikation, die bei einer Kundenbeschwerde eine angeblich fehlerhafte Rechnung aus dem Archiv lädt. Ursprünglich wurde diese Rechnung von einem *Billing*-System erstellt und archiviert. Hier sieht es so aus, als ob die *Customer-Care*-Anwendung mit dem *Billing*-System zunächst nichts zu tun hat.

Um aber in einem solchen Anwendungsfall das richtige Dokument – hier die richtige Rechnung – zu finden, muss die *Customer-Care*-Applikation eine Reihe von eindeutigen Attributen zur Suche an das Archiv übergeben. Sind die Attribute nicht vollständig oder mehrdeutig, wird das Archiv nicht nur ein Dokument sondern eine ganze Reihe von Dokumenten liefern. Eine erfolgreiche Suche setzt somit zum einen voraus, dass das *Billing*-System beim Archivieren der Rechnung wirklich eindeutige Attribute gesetzt hat, und zum anderen, dass die *Customer-Care*-Applikation implizit die zu suchenden Schlüsselattribute kennt. Die *Customer-Care*-Applikation muss wissen, dass sich die Rechnung beispielweise über die Kundennummer und die Rechnungsnummer identifizieren lässt, und sie muss beide kennen.

Nachteile der herkömmlichen Architektur

Eine Entkopplung von Erzeuger- und Nutzersystemen über das Archiv existiert nicht wirklich, wie das Beispiel zeigt. Vielmehr



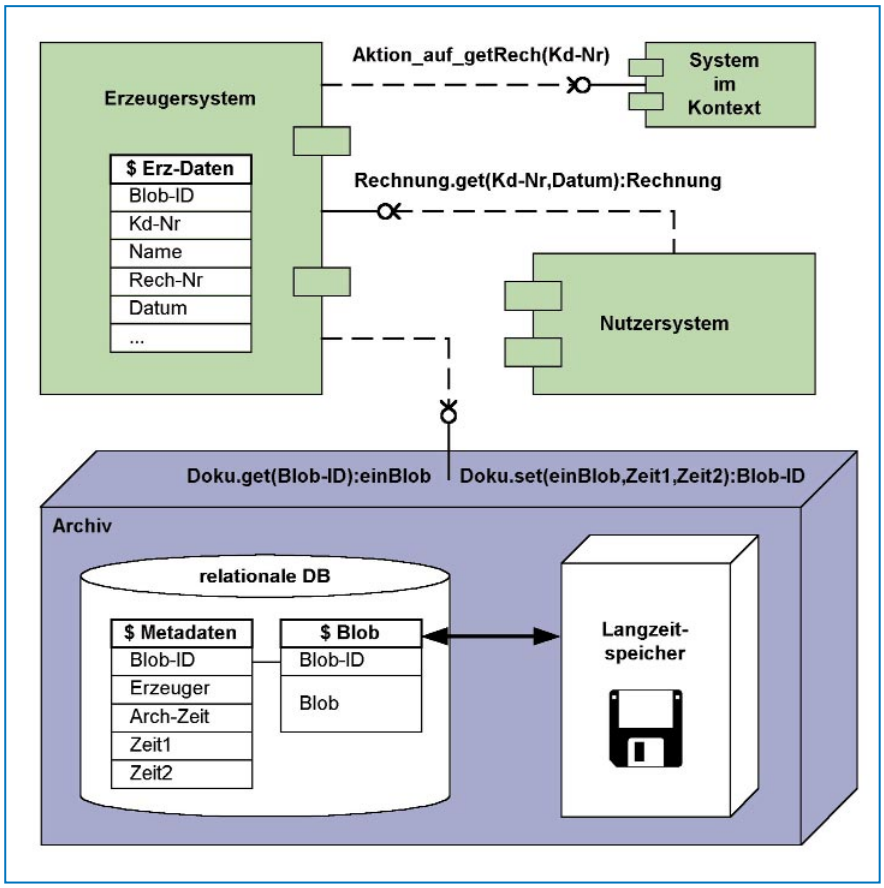


Abb. 2: Objektorientierte Architektur

ergibt sich durch die herkömmlichen Architektur eine ganze Reihe von Nachteilen:

Über die Attribute werden strukturierte Abhängigkeiten zwischen den Erzeugersystemen, dem Archiv und den Nutzersystemen geschaffen. Diese Abhängigkeiten führen dazu, dass bei kleinen Änderungen in jedem der Systeme Anpassungen durchzuführen sind. Das ist meistens mit erheblichen Aufwänden verbunden. Anpassungen müssen erfolgen, sobald

- Versionswechsel von Dokumenttypen oder Metainformationen,
- Änderungen oder Erweiterungen der Suchfunktionalität oder
- Veränderungen in der Attributierung der Dokumente

notwendig werden.

Die Inhalte eines Dokuments stehen zumeist in einem größeren Kontext, der nicht auf reine Daten in Datenbanken beschränkt ist. Der Kontext ergibt sich aus dynamischen Abläufen in Erzeuger-, Nutzer- und/oder anderen angelagerten Systemen. In den Archiv-Suchtabellen existieren keine dynamischen Informationen mit Abhängigkeiten vom Kontext. Im Zusammenhang mit diesen dynamischen Kontexten gibt es im Wesentlichen drei Aspekte:

- Dokumente können möglicherweise nicht identifiziert werden oder sie werden falsch identifiziert. Die Rechnung des Kunden wurde z. B. noch nicht archiviert, weil bei ihm zuvor durch das *Billing*-System noch andere Prozesse – wie Anlegen eines Kundenkontos, Absenden eines Begrüßungsschreibens – abgewartet werden.
- In Folge von Abrufen eines Dokuments können keine Aktionen im Kontext des verantwortlichen Erzeugersystems ausgelöst werden, weil das Erzeugersystem über den Abruf nicht informiert ist.
- Es ist möglich, dass sich die Eigenschaften des Dokuments im dynamischen Kontext durch Vorgänge in den Erzeugersystemen ändern, ohne dass sich das Dokument selber oder seine Attribute ändern. Wer sagt z. B. der *Customer-Care*-Applikation, dass die gefundene Rechnung bereits veraltet ist, weil gerade eine neue erstellt wird?

Die Attribute der Suchtabellen sind im Wesentlichen strukturierte Auszüge aus dem Inhalt des Dokuments. Diese Daten werden meist zu großen Teilen redundant in den Erzeugersystemen gehalten.

Für eine eindeutige Identifizierung muss gewährleistet sein, dass die Attribut-Tupel in den Suchtabellen immer eindeutig sind. Es ist wahrscheinlich, dass es sehr schnell attributgleiche Dokumente geben wird, weil z. B. verschiedene Versionen eines Dokuments existieren. Dies könnte beispielsweise eine zweite korrigierte Version einer Rechnung sein. Alternativ kommt es zu Problemen beim Einstellen ins Archiv, wenn Attribute als Primärschlüssel in der Datenbank definiert sind.

Ein Erzeugersystem hat keine Kontrolle über die Lesezugriffe auf ein Dokument.

Konzept einer objektorientierten Architektur

An Stelle der herkömmlichen Lösung wird eine Architektur (**siehe Abb. 2**) empfohlen, die die Eigenschaften eines objektorientierten Designs besitzt: Flexibilität, Robustheit, leichte Erweiterbarkeit und klare Trennung von Verantwortlichkeiten. Grundlage dieser Architektur ist das *Black-Box*-Prinzip, das sich aus der Abkapselung und dem Verstecken von Informationen zusammensetzt (vgl. [Bas98]).

Um diese Eigenschaften zu erreichen, bleibt das Archiv auf Kernfunktionalitäten (**siehe Kasten 3**) beschränkt und erhält eine möglichst einfache Schnittstelle. Es wird ein Zugriff zum Speichern und Lesen der Dokumente und zum Setzen und Ändern der Speicherzeiträume zur Verfügung gestellt. Einziger Zweck des Archivs ist es, eine von fachlichen Inhalten vollkommen unabhängige und von beliebigen Systemen nutzbare Persistenzschicht für Dokumente zu sein.

Die von den Nutzersystemen benötigte Suchfunktionalität ist nicht vom Archiv sondern als Schnittstelle der Erzeugersysteme bereitzustellen. Das Nutzersystem übermittelt dem Erzeugersystem einen Satz von Parametern, die nach seiner Kenntnis zur Identifizierung eines Dokuments notwendig sind. Das Erzeugersystem besitzt alle relevanten Informationen, um das benötigte Dokument anhand dieser Parameter eindeutig zu identifizieren. Dazu gehören nicht nur die Parameter selbst, sondern insbesondere auch der dynamische Kontext des Dokuments und der Suchanfrage. Weiterhin sind alle Metainformationen inklusive der Doku-

So funktioniert das Archiv.

Die Funktionalität des Archivs ist im Prinzip die Kernfunktionalität kommerzieller „Dokumentenmanagement-Systeme". Man sollte eines dieser Systeme verwenden, um das Archiv aufzubauen. Bei der Auswahl eines Systems ist die Gesamtkapazität sorgfältig zu kalkulieren. Über die betrachteten Zeiträume kommen in größeren Firmen mit Endverbrauchern als Kunden leicht eine Milliarde Dokumente zusammen.

Das Archiv besitzt eine zentrale Schnittstelle zum Einlagern und Abrufen von Dokumenten. Bei der Archivierung wird dieser Schnittstelle ein Dokument als BLOB zusammen mit zwei Zeitangaben übergeben. Bei der Übergabe des BLOBs vergibt das Archiv zunächst eine eindeutige BLOB-ID und liefert diese an das Erzeugersystem zurück. Der BLOB wird dann unter dieser BLOB-ID mit der Angabe von Erzeugersystem, Archivierungszeitpunkt und den beiden Zeitangaben in der Datenbank gespeichert. Die erste Zeitangabe (Zeit1 **in Abb. 2**) ist das Datum, bis zu dem der BLOB in der Datenbank und noch nicht im Langzeitspeicher vorgehalten wird. Wird der erste Zeitpunkt erreicht, verlagert das Archiv den BLOB automatisch von der Datenbank in den Langzeitspeicher. Die zum BLOB gehörenden Informationen, BLOB-ID, Erzeuger und Zeitangaben verbleiben in der Datenbank. Vor diesem Zeitpunkt sind schnelle Zugriffe auf den BLOB möglich. Danach sind Zugriffe sehr viel langsamer. Beim Erreichen des zweiten Zeitpunktes (Zeit2 **in Abb. 2**) erfolgen die Löschung des BLOBs aus dem Langzeitspeicher und die Löschung der BLOB-ID mit den zugehörigen Metadaten aus der Datenbank.

Mit der Rückgabe der BLOB-ID an das Erzeugersystem kann dieses die BLOB-ID zusammen mit anderen das Dokument betreffenden Daten speichern. Für einen späteren Zugriff genügt es, wenn das Erzeugersystem die BLOB-ID an das Archiv übergibt. Das Archiv liefert daraufhin den BLOB als Rückgabewert. Eine Rückgabe des BLOBs an ein anderes System als das Erzeugersystem wird ausgeschlossen, um eine strenge Abkapselung zu gewährleisten.

Dem Erzeugersystemen ist es jederzeit möglich, die beiden Zeitangaben zu ändern, sodass z. B. durch die Angabe eines neuen ersten Zeitpunktes in der Zukunft eine Rückverlagerung des BLOBs aus dem Langzeitspeicher in die Datenbank erfolgt. Auf diese Weise wird wieder ein schnellerer Zugriff auf das Dokument gewährleistet. Die internen Abläufe, wie das Hin- und Herschieben der BLOBs, bleiben den externen Systemen verborgen. Das Archiv funktioniert als *Black-Box*.

Kasten 3

menttypen in allen Versionen und mögliche weitere Attribute, für die eine Rückfrage erfolgen kann, bekannt. Für das Nutzersystem bleiben die internen Abläufe der Suche im Erzeugersystem hinter der Schnittstelle verborgen. Als Rückgabewert erhält das Nutzersystem schließlich das Dokument und eventuell notwendige Metainformationen, wie Grafikformate, um das Dokument auswerten zu können. Das Erzeugersystem hat dazu das Dokument als Ergebnis seiner eigenen Suche aus dem Archiv geholt.

Die *Vorteile* dieser Lösung sind:

- Das Erzeugersystem kennt den dynamischen Kontext des Dokuments und der Suchanfrage und kann dynamisch reagieren.
- Alle Informationen für die Suchfunktionalität sind im Erzeugersystem vorhanden und werden nicht redundant im Archiv gehalten. Die notwendigen Informationen sind vollständig und nicht nur partiell vorhanden.
- Die Schnittstelle zwischen Erzeuger- und Nutzersystem kapselt interne Änderungen der Dokumente, ihrer Attribute, der Metainformationen und der Suchfunktionalität. Die Nutzersysteme hängen daher nicht von den fachlichen Inhalten sondern nur von der Schnittstelle ab.
- Die Verantwortung für das Dokument verbleibt uneingeschränkt beim Erzeugersystem. Es kann somit entscheiden, ob das Nutzersystem überhaupt zum Lesen des Dokuments berechtigt ist.
- Das Erzeugersystem ist in der Lage, auf eine Anfrage zu reagieren und beispielsweise eine Verlängerung der Lagerdauer des Dokuments veranlassen.
- Das Archiv ist leicht erweiterbar, da für neue Erzeuger- oder Nutzersysteme keine Anpassungen im Archiv vorgenommen werden müssen. Das Archiv hängt nicht von fachlichen Daten ab.
- Eine Suche in den Erzeugersystemen ist prinzipiell performanter als im Archiv, da in den Datenbanken der Erzeugersysteme immer nur eine Teilmenge aller Dokumente verwaltet wird.
- Kommt es zu Migrationen – z. B. durch Systemaktualisierungen – so werden im Erzeugersystem die Suchattribute mehr oder minder automatisch „mitmigriert". Im Archiv muss keine separate Migration erfolgen.

Nachteile der Lösung sind:

- Im Vergleich zur herkömmlichen Architektur hat man bei einem suchenden Zugriff einen erhöhten Kommunikationsaufwand zwischen den Systemen. Statt nur eines Zugriffs auf das Archiv wird die Kommunikation zwischen Nutzer- und Erzeugersystem zusätzlich notwendig.
- Fällt das Erzeugersystem aus, ist eine Suche nach Dokumenten nicht mehr möglich.
- Im hypothetischen Fall eines unbekannten Dokumententyps werden Suchanfragen bei mehreren Erzeugersystemen notwendig.

Bei der dargestellten Architektur handelt es sich um die Anwendung des *Black-Box-Prinzips*, weil eine konsequente Kapselung der Dokumente durch ihre Rückführung in den alleinigen Kontext der für sie verantwortlichen Erzeugersysteme praktiziert wird. Hinzu kommt ein zweifaches Verstecken von Informationen. Zum einen werden die Abläufe innerhalb des Archivs vor den Erzeugersystemen versteckt, zum anderen werden den Nutzersystemen Schnittstellen zur Verfügung gestellt, die nicht mehr von den internen Details des Archivs bzw. der Erzeugersysteme abhängen.

Diese objektorientierte Architektur bedeutet gegenüber der herkömmlichen Architektur natürlich einen Mehraufwand bei der Einführung. Auf lange Sicht wird sich eine solche Investition durch eingesparte Änderungs- und Migrationsaufwände aber lohnen. Im Übrigen spricht nichts dagegen beide Architekturen zu mischen. So kann man einen langfristigen Umbau einer bereits bestehenden herkömmlichen Architektur verwirklichen. ■

Literatur

[Bas98] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley, 1998

[Sch98] R. Schuppenhauer, Grundsätze für eine ordnungsmäßige Datenverarbeitung – Handbuch der DV-Revision, IDW Verlag, 1998