Flexibilität in Betrieblichen Informationssystemen

Andreas Oberweis, Wolffried Stucky

J.W. Goethe-Universität
Institut für Wirtschaftsinformatik
60054 Frankfurt/Main
E-Mail: oberweis@computer.org
WWW: wiwi.uni-frankfurt.de/~oberweis

Universität Karlsruhe (TH)
Institut AIFB
76128 Karlsruhe
stucky@aifb.uni-karlsruhe.de
WWW: aifb.uni-karlsruhe.de/~stucky

Abstract Flexibilität, d.h. die Fähigkeit zur Anpassung an geänderte Umstände, gehört zu den wichtigsten Anforderungen, die an betriebliche Informationssysteme gestellt werden. Trotzdem verfügen Informationssysteme in der Praxis vielfach nur über unzureichende Möglichkeiten zur Anpassung bzw. erforderliche Änderungen sind oftmals nur mit unvertretbar hohem Aufwand möglich.
Dieser Beitrag beleuchtet das Problem der Flexibilität, beschreibt Möglichkeiten zur Lösung und schlägt workflowbasierte Informationssysteme vor, um ein größtmögliches Maß an Flexibilität zu ermöglichen.

1. Einleitung

Die Anforderungen von Anwendungsseite an betriebliche Informationssysteme lassen sich grob den folgenden Gestaltungsbereichen zuordnen:

- ***Funktionalität***: Das Informationssystem soll eine zur Aufgabenstellung passende (d.h. nicht zu viel und nicht zu wenig) Funktionalität besitzen.
- ***Effizienz***: Das System soll einen sparsamen Umgang mit den technischen und personellen Ressourcen, etwa Prozessorlaufzeit, externer Speicher, Arbeitsspeicher, Kommunikationszeiten und Mitarbeiterzeit, ermöglichen.
- ***Zuverlässigkeit***: Ausfallzeiten des Systems sollen sich innerhalb bestimmter garantierter anwendungsspezifischer Toleranzgrenzen bewegen.
- ***Beherrschbarkeit***: Die Reaktionen des Systems sollen jederzeit vorhersehbar, kontrollierbar, nachvollziehbar und steuerbar sein.
- ***Usability***: Das System soll für die Benutzer in dem vorgesehenen Anwendungskontext gebrauchstauglich sein, d.h. ihm helfen, mit möglichst geringem Aufwand ein bestimmtes Ziel zu erreichen.
- ***Flexibilität***: Das System soll über die Fähigkeit zur Anpassung an veränderte Umstände verfügen. Sowohl automatische als auch manuelle Anpassungsmaßnahmen sollen mit vertretbarem Aufwand durchführbar sein.

Die Informatikforschung hat sich in der Vergangenheit hauptsächlich mit Fragen der Funktionalität, der Effizienz und der Zuverlässigkeit befasst. Trotz der großen Relevanz für die Praxis sind Beherrschbarkeit, Usability und Flexibilität wegen ihrer schwereren formalen bzw. quantitativen "Greifbarkeit" vernachlässigt worden.

Der vorliegende Beitrag befasst sich mit Fragen der Flexibilität in betrieblichen Informationssystemen. Software (also auch ein Informationssystem) ist im Gegensatz zur Hardware prinzipiell (weitgehend) beliebig "änderbar", da sie ein immaterielles Gut ist. Es mag daher auf den ersten Blick für Außenstehende überraschend sein, dass Flexibilität überhaupt ein Problem darstellt. Letztendlich geht es bei Fragen der Flexibilität von Informationssystemen nicht um die grundsätzliche Änderbarkeit, sondern um die *effiziente* Änderbarkeit. Bei der Beurteilung der Effizienz sind nicht nur die direkten Kosten einer Änderung zu berücksichtigen, sondern auch die damit verbundenen Folgekosten (für Schulung, Installation usw.).
Dieser Beitrag ist wie folgt gegliedert: Im anschließenden zweiten Kapitel wird der Begriff "Flexibilität" näher betrachtet, unterschiedliche Formen von Flexibilität werden herausgestellt. Kapitel 3 betrachtet Flexibilität im Zusammenhang mit Indvidualsoftware, Kapitel 4 im Zusammenhang mit Standardsoftware. In Kapitel 5 werden grundsätzliche Methoden der Angewandten Informatik zur Unterstützung von Flexibilität kurz beschrieben, Kapitel 6 stellt flexible, workflowbasierte Informationssysteme vor. Kapitel 7 schließt den Beitrag mit einem kurzen Ausblick auf zu erwartende zukünftige Entwicklungen im Bereich der Flexibilisierung digitaler Produkte ab.

2. Was ist Flexibilität?

Flexibilität eines Systems bezeichnet die Eignung des Systems zur Anpassung an veränderte Umstände, unter denen das System zum Einsatz kommt. Im Lebenszyklus eines Informationssystems hat das Thema Flexibilität Relevanz vor der Einführung (d.h. zur Entwicklungszeit), im laufenden Betrieb und im Rahmen von Wartungsmaßnahmen.
Der Begriff Flexibilität tritt in der Literatur und in der Praxis unter verschiedenen Bezeichnungen auf: Begriffe wie Parametrisierbarkeit, Änderbarkeit, Wartbarkeit, Erweiterbarkeit oder Tuningfähigkeit werden synonym oder mit geringen Bedeutungsunterschieden verwendet. Gelegentlich wird der Begriff "Flexibilität" abgegrenzt zum Begriff "Robustheit" [SHN01]: Während Flexibilität die Fähigkeit eines Systems bezeichnet, sich ändernde Anforderungen nach Fertigstellung des Systems zu erfüllen, bezeichnet Robustheit die Fähigkeit, gegebene Anforderungen auch bei geänderten Umgebungsbedingungen zu garantieren.
Es ist außerdem zu unterscheiden zwischen "universeller Einsetzbarkeit" und "Flexibilität" [SHN01]. Software, die in vielen unterschiedlichen Umgebungen ohne Änderung einsetzbar ist (etwa durch ein unter verschiedenen Betriebssystemen laufendes Webbrowser-Interface), muss nicht zugleich auch flexibel sein.
Gründe für notwendige Anpassungen von betrieblichen Informationssystemen sind:

- Änderung von Marktbedingungen (z.B. wünschen Kunden schnellere Auskünfte über den aktuellen Bearbeitungsstatus eines Auftrags).
- Verfügbarkeit neuer Technologien (z.B. ermöglichen moderne Mobilfunknetze eine Kommunikation der Unternehmenszentrale mit Außendienstmitarbeitern unabhängig von deren geographischen Standorten).
- Änderung rechtlicher Rahmenbedingungen (z.B. Erhöhung des Mehrwertsteuersatzes).

- Organisatorische Umgestaltung (z.B. Änderung eines Geschäftsprozesses zur Verbesserung der Time to Market).
- Auftreten von unvorhergesehenen (Fehler-)Situationen (z.B. bei Internationalisierung des Geschäfts müssen Kundenadressdaten verarbeitet werden können, die von dem Standard-Datentyp "Kundenadresse" abweichen, etwa weil die Postleitzahl ein unübliches Format besitzt).
- Wiederverwendung von Software (z.B. bei Einsatz eines Programms in unterschiedlichen Branchen).
- Anwenderzufriedenheit (z.B. wünschen Anwender Gestaltungsspielraum bei der Ausführung von Tätigkeiten oder unterschiedlich gestaltete Benutzungsoberflächen).

Bei den Anpassungsarten kann unterschieden werden zwischen:

- langfristigen / kurzfristigen (vorübergehenden) Anpassungen: langfristige Anpassungen haben bleibenden Bestand für die Restlebenszeit eines Informationssystems, wohingegen kurzfristige Anpassungen später wieder rückgängig gemacht werden.
- einmaligen / mehrmaligen / regelmäßigen Anpassungen: Anpassungen können einmalige oder mehrmalige Änderungen betreffen oder aber in regelmäßigen Zeitabständen vorgenommen werden.
- geplanten / ungeplanten ("ad hoc", spontanen) Anpassungen: Anpassungen können geplant sein, beispielsweise bereits beim Entwurf des Systems, oder ungeplant sein. Damit auch ungeplante Anpassungen nicht vollständig unkontrolliert ablaufen, müssen sie im Rahmen bestimmter Regelungen erfolgen, die nicht verletzt werden dürfen.
- automatischen / manuellen Anpassungen: ein System kann für Änderungen vorgesehen sein, die Anpassungen können dann beim Eintreten bestimmter Bedingungen automatisch erfolgen. Andere Anpassungen erfolgen manuell.
- strukturbezogenen / verhaltensbezogenen Anpassungen: die Struktur eines Systems kann geändert werden und / oder das Verhalten.
- funktionalen / nicht-funktionalen Anpassungen: Funktionen des Systems können angepasst werden oder nicht-funktionale Eigenschaften (z.B. Performance).

In [SHN91] wird darauf hingewiesen, dass Flexibilität nicht nur ermöglicht, angemessen auf Änderungen in der Umgebung zu reagieren, sondern auch, Änderungen in der Umgebung durch Änderungen im Informationssystem hervorzurufen. Die Flexibilität eines Systems kann insofern sogar als "strategic weapon" eines Unternehmens gesehen werden.

In den folgenden beiden Kapiteln werden spezielle Aspekte der Flexibilität in Individualsoftware und in Standardsoftware betrachtet.

3. Individualsoftware

Es gibt prinzipiell zwei Möglichkeiten zur Software-Beschaffung: Implementierung von Individual-Software (in Eigen- oder Fremdentwicklung) sowie Kauf von Standard-Software. Individual-Software wird speziell zu den Bedürfnissen eines Unternehmens

passend implementiert. Dem Entwicklungsprozess liegt üblicherweise ein Vorgehensmodell zugrunde, etwa eine Variante des Wasserfallmodells (vgl. Abbildung 1).
Meistens ist Individual-Software auf bestimmte kleinere Änderungen bzw. Anpassungen bereits vorbereitet: so ist es beispielsweise (ohne Änderung der zugrunde liegenden Programme) möglich, neue Benutzerkennungen mit entsprechenden Rechten einzurichten oder die Benutzungsoberfläche nach individuellen Bedürfnissen einzustellen. Die möglichen Änderungsarten müssen allerdings bereits zur Entwurfszeit bekannt sein und mittels entsprechender Parameter im System implementiert sein. Darüber hinausgehende Anpassungen sind nicht möglich, ohne die Programme selbst zu ändern.
Änderungen am Programmcode installierter Informationssysteme werden üblicherweise als Wartung bezeichnet.

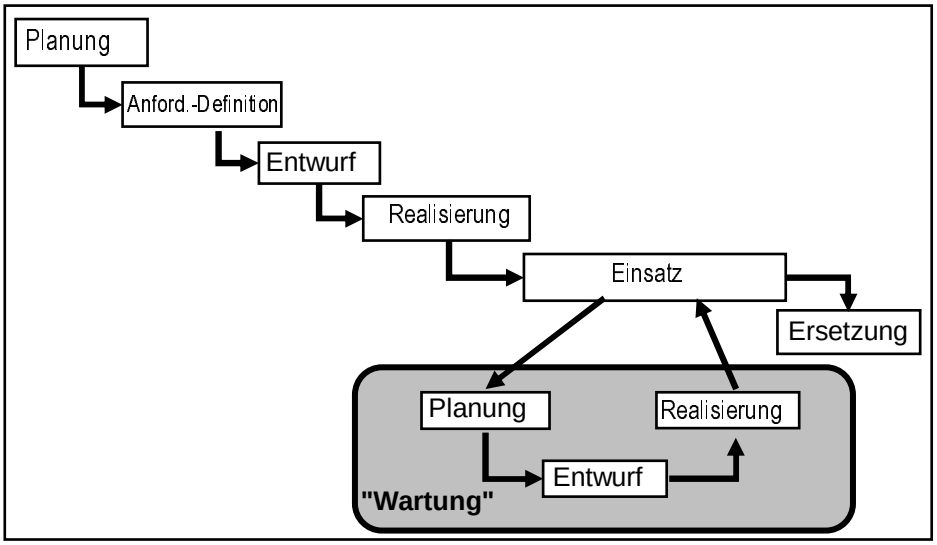

Abbildung 1: "Lebenszyklus" eines Informationssystems

Das Durchlaufen eines Wartungszyklus, bestehend aus einem Planungs-, einem Entwurfs- und einem Realisierungsschritt (vgl. Abbildung 1), führt jeweils zu einer neuen Version eines Informationssystems. Es stellt sich – insbesondere bei komplexen Systemen – die Frage, wann bzw. wie oft ein Versionswechsel stattfinden soll und wie umfangreich die Unterschiede zwischen aufeinander folgenden Versionen sein sollen (Abbildung 2).
Da die Einführung einer neuen Version mit zusätzlichem Aufwand verbunden ist (etwa für Installationsarbeiten und Schulungsmaßnahmen), sollte die Zahl der Versionen verhältnismäßig niedrig gehalten werden. Andererseits gibt es von Anwenderseite von Zeit zu Zeit wichtige Änderungswünsche, die möglichst schnell berücksichtigt werden sollen. Durch häufigere Änderungen werden die Unterschiede zwischen aufeinander folgenden Versionen (in Abbildung 2 als Δ_i bezeichnet) geringer, dadurch sinkt das Entwicklungsrisiko.

Flexibilität in diesem Zusammenhang bezeichnet die Eigenschaft der Software, dass Änderungswünsche in der Einsatzphase mit verhältnismäßig geringem Aufwand erfüllt werden können. Das Konfigurationsmanagement als Teilbereich des Software Engineering Management umfasst auch das Management von Änderungen.

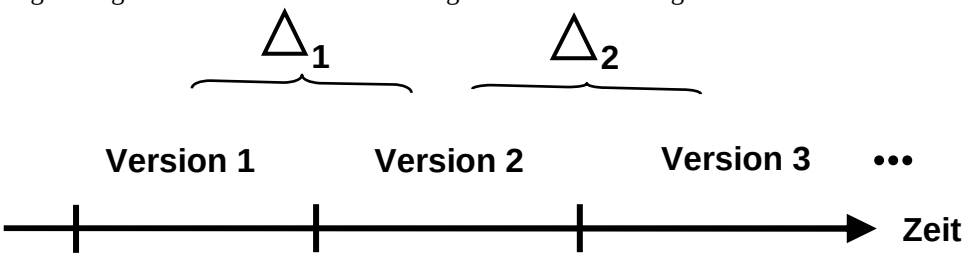

Abbildung 2: "Versionierung" eines Informationssystems

4. Standardsoftware

Betriebswirtschaftliche Standardsoftware wird für einen "anonymen" Markt hergestellt. Sie muss daher die (potentiellen) Anforderungen einer Vielzahl von Unternehmen erfüllen, und sie muss dafür vorbereitet sein, an die Bedürfnisse eines konkreten Unternehmens individuell angepasst zu werden. Thomas Davenport stellt 1999 in diesem Zusammenhang die provokative Frage [Dav99]: "Passt Ihr Unternehmen zur Software?" Vielfach ist es in der Praxis nämlich gerade so, dass die Geschäftsprozesse an die Software angepasst werden. Aus Kundensicht stellt sich bei der Einführung betriebswirtschaftlicher Standardsoftware die Frage nach dem optimalen Mix zwischen Anpassung der Geschäftsprozesse und Customizing der Software.
Beim modellgestützten Customizung erfolgt die Anpassung anhand von Modellen, die von Prozessen und Objekten des Anwendungsbereichs erstellt werden. Diese Ist- bzw. Soll-Modelle (in Abbildung 3 als $Modell_U$ bezeichnet) werden mit den Referenzmodellen der Standardsoftware (in Abbildung 3 $Modell_{SW}$) abgeglichen. Zwischen $Modell_U$ und $Modell_{SW}$ besteht in der Ausgangsituation ein Unterschied $\Delta_{U\text{-}SW}$. Der Abgleich führt zu einem neuen Zielmodell ($Modell_{IS}$). Die Überbrückung der Unterschiede Δ_{Umg} bzw. Δ_{Cus} ist mit Umgestaltungs- bzw. Customizing-Kosten verbunden, die zu minimieren sind.
Diese Anpassung ist aber üblicherweise keine einmalige Aktivität. Denn immer dann, wenn neue Versionen der Standard-Software installiert werden, muss die Anpassung wiederholt werden. Auch wenn im Laufe der Zeit Prozesse im Unternehmen umgestaltet werden sollen, sind entsprechende Anpassungen bei der Standardsoftware nötig. Insofern ist das Ziel, die Umgestaltungs- und Customizing-Kosten nicht nur einmalig bei der Erstinstallation sondern über die gesamte Lebenszeit der Standard-Software hinweg zu minimieren.
Neben der individuellen Anpassung an die spezifischen Bedürfnisse eines Unternehmens gibt es noch länderspezifische Anpassungen (Sprache, Zeichensatz, steuerrechtliche Regelungen, Währung usw.) sowie branchenspezifische Anpassungen. Auch die Größe des Unternehmens spielt eine Rolle: Kleine, mittlere und große Unternehmen brauchen üblicherweise jeweils unterschiedliche Funktionalität der Software.

In multi-nationalen Konzernen muss zudem geklärt werden, welche Objekte und Prozesse allgemeingültig (d.h. unternehmensweit standardisiert) sein sollen und was individuell entsprechend den regionalen Gegebenheiten realisiert werden darf.

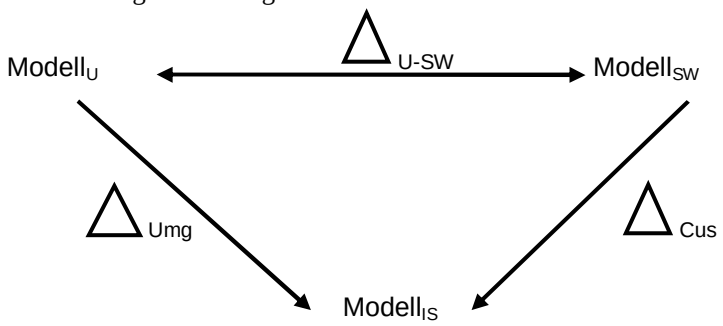

Abbildung 3: Modellgestütztes Customizing

Aus Sicht des Standardsoftware-Herstellers stellen sich im Zusammenhang mit der Produktpolitik verschiedene Fragen. Es ist zu klären, wie viel Flexibilität durch Parametrierung in die Software eingebaut wird. Bei Versionswechsel ist der Umfang der Neuerungen (d.h. das Δ zwischen den Versionen) festzulegen. Auch der Zeitpunkt, zu dem eine neue Version bereit gestellt wird, muss bestimmt werden. Schließlich sind Fragen der Kompatibilität zu früheren Versionen zu beantworten.

5. Methodische Unterstützung

Zur Unterstützung von Flexibilität stellt das Software-Engineering verschiedene grundlegende Methoden bereit.
Die **Modularisierung** bei Modellierung und Programmierung beispielsweise erlaubt eine Trennung (Kapselung) unterschiedlicher Aufgabenbereiche voneinander. Ein Modell oder ein Softwaresystem ist modular aufgebaut, wenn es aus voneinander abgrenzbaren Einheiten zusammengesetzt ist und wenn diese Einheiten einzeln ausgetauscht, verändert oder hinzugefügt werden können, ohne dass andere Teile des Modells oder Systems hierdurch beeinflusst werden. Bei Änderungen in einem Modul brauchen die anderen Module also nicht berücksichtigt zu werden.
Die Architektur eines Informationssystems setzt sich beispielsweise aus den Modulen Benutzungsoberfläche, Schnittstellen, Funktionskern, Datenhaltung und Basistechnologie zusammen.
Auch die 3-Ebenen-Architektur von Datenbanksystemen (Abbildung 4), bestehend aus externer, konzeptueller und interner Ebene, setzt dieses Prinzip um und bietet damit flexible Einsatzmöglichkeiten: Änderungen der internen Ebene haben keinen Einfluss auf die externe Ebene, Änderungen auf externer Ebene haben nur begrenzten Einfluss auf die interne Ebene, datenbankbezogene Änderungen in einem Anwendungsprogramm machen keine Änderungen in anderen Anwendungsprogrammen erforderlich.
Beim Entwurf von Software kann Flexibilität auch zunächst durch **"Unterspezifikation"**, d.h. durch Offenlassen von Detailentscheidungen, realisiert werden (vgl. [Wed01]).

Außerdem ist eine **Berücksichtigung aller (denkbaren) Alternativen** möglich. Weiterhin können **Ausnahmesituationen** definiert werden, die im laufenden Betrieb manuell behandelt werden müssen (Exception-Handling [ObS91, StM95]).

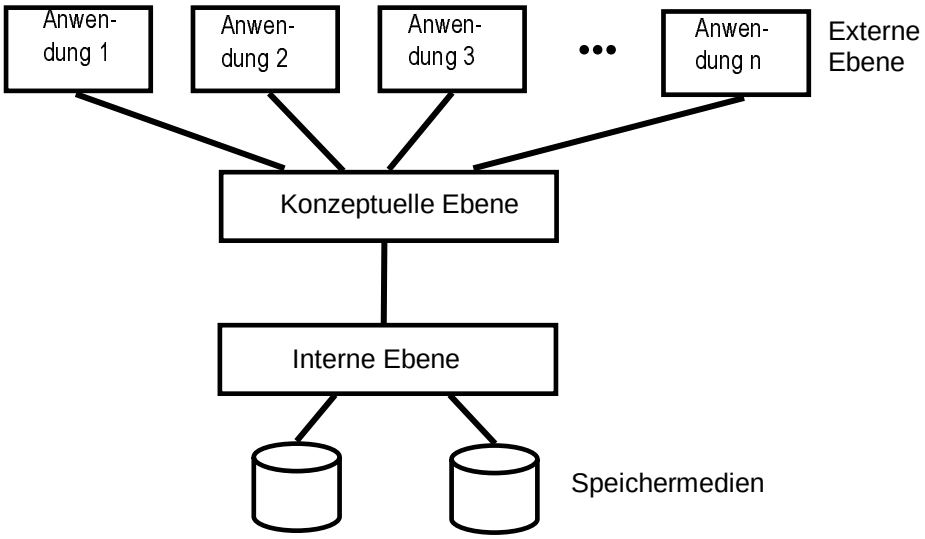

Abbildung 4: 3-Ebenen-Architektur von Datenbanksystemen

Auch die Abstraktionstechniken **Generalisierung** und **Spezialisierung** ([SmS77], vgl. auch [Wed01]) in Verbindung mit der Vererbung von Struktur und Verhalten tragen zur Flexibilität von Prozess- bzw. Objekt-Modellen bei.
Abbildung 5 zeigt als Beispiel einen Ausschnitt aus dem Objektstrukturschema eines Krankenhausinformationssystems. Der Super-Typ Person wird spezialisiert zu den Sub-Typen Mitarbeiter und Patient, Mitarbeiter wird weiter spezialisiert zu Pfleger, Verwaltungsangestellter und Arzt. Neben den Attributen eines Objekt-Typs enthält die graphische Darstellung auch Methoden, die von den Objekten eines Typs aufgerufen werden können. Sub-Typen erben von den Super-Typen Eigenschaften und können aber auch über zusätzliche Eigenschaften verfügen. Änderungen, die mehrere Objekt-Typen betreffen (z.B. Änderung des Attributs Adresse), brauchen nur in dem übergeordneten Objekt-Typ durchgeführt zu werden und vererben sich dann automatisch auf die untergeordneten Objekt-Typen. Dadurch sinkt der Änderungsaufwand, und die Fehlerwahrscheinlichkeit bei Änderungen wird verringert.

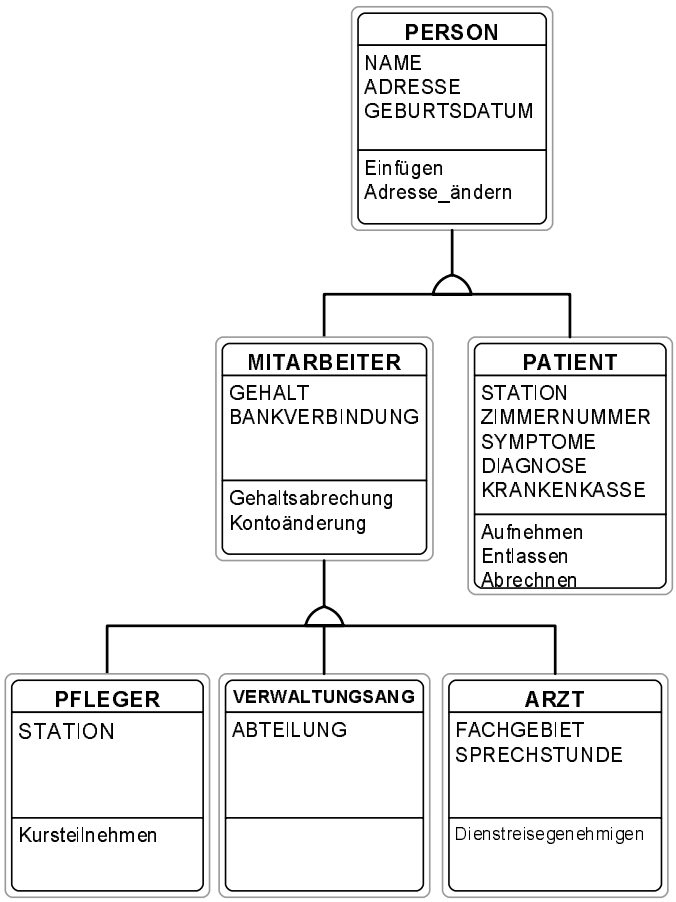

Abbildung 5: Objektstrukturschema (Ausschnitt) eines Krankenhausinformationssystems

Im folgenden Kapitel wird als weiteres Konzept zur Unterstützung von Flexibilität in Informationssystemen die **Trennung von Ablauflogik und Programm** betrachtet, wie sie Workflow-Managementsysteme unterstützen. Ähnlich wie bei Informationssystemen das Datenbank-Managementsystem alle wichtigen Funktionen im Zusammenhang mit der Datenhaltung übernimmt, übernimmt ein Workflow-Managementsystem alle Kontroll- und Steuerungsaufgaben, die mit der Ausführung von betrieblichen Abläufen zusammenhängen.

6. Workflowbasierte flexible Informationssysteme

Bei der traditionellen Informationssystementwicklung enthält der Programm-Code die Ablauflogik "fest verdrahtet". Der Programm-Code wird übersetzt und als Maschinen-Code auf dem Zielrechner ausgeführt (vgl. Abbildung 6). Jede Änderung an der Ablauflogik erfordert eine Änderung des Source-Codes, eine anschließende Übersetzung und eine Neu-Installation. Offensichtlich kann eine solche Änderung des Quell-Codes ein(e) Endanwender(in) üblicherweise nicht selbst vornehmen.

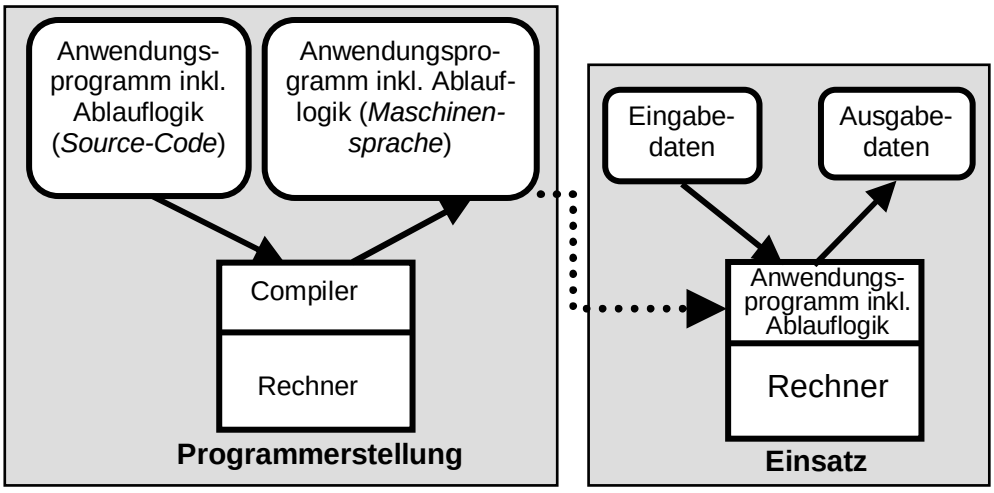

Abbildung 6: Traditionelle Programmerstellung

Demgegenüber sind in workflowbasierten Informationssystemen betriebliche Abläufe nicht fest programmiert über die Ablauflogik im Anwendungsprogramm. Stattdessen werden die Abläufe im Ablaufschema beschrieben [Obe96]. Diese Ablaufschemata werden durch eine Workflow-Engine interpretiert, die Interpretations ersetzt somit die Compilierung von Programm-Code. Interpretation heisst, dass das Ablaufschema Schritt für Schritt abgearbeitet wird. Dabei werden benötigte Anwendungsmodule mit entsprechenden Eingabe- und Steuerdaten aufgerufen und ausgeführt (vgl. Abbildung 7).
Zu einem gegebenen Ablaufschema können zur Laufzeit unterschiedliche Ablauf-Instanzen erzeugt werden (Abbildung 8). Dieses Prinzip der Trennung zwischen Schema und Instanz ist im Datenbankbereich schon lange bekannt und kann auf den Workflow-Bereich übertragen werden.
Änderungspotential für Ablauf-Instanzen ergibt sich vor Beginn der Ausführung und während der Ausführung (vgl. [CGD01]). Durch entsprechende Regeln muss festgelegt werden, welche Änderungen zulässig sind und wer über die Änderungen entscheidet. Durch vorausschauende Simulation können Änderungen vor der Durchführung validiert werden.

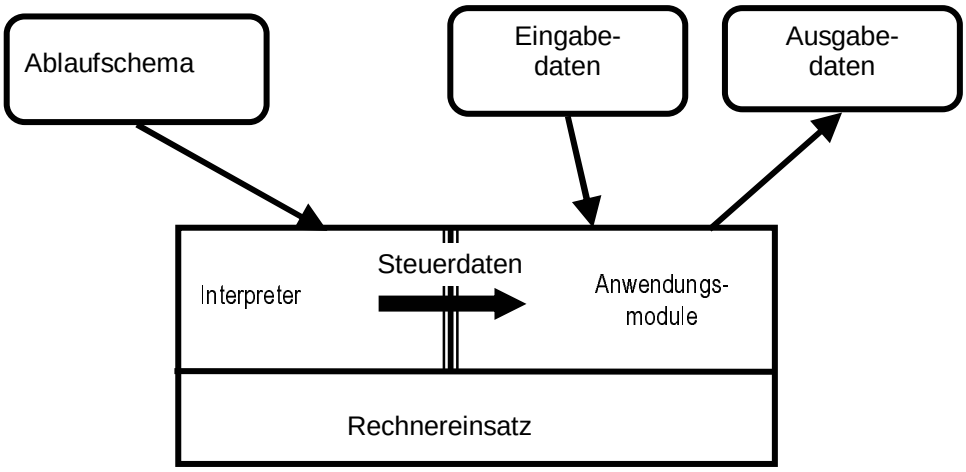

Abbildung 7: Grundprinzip workflowbasierter Informationssysteme

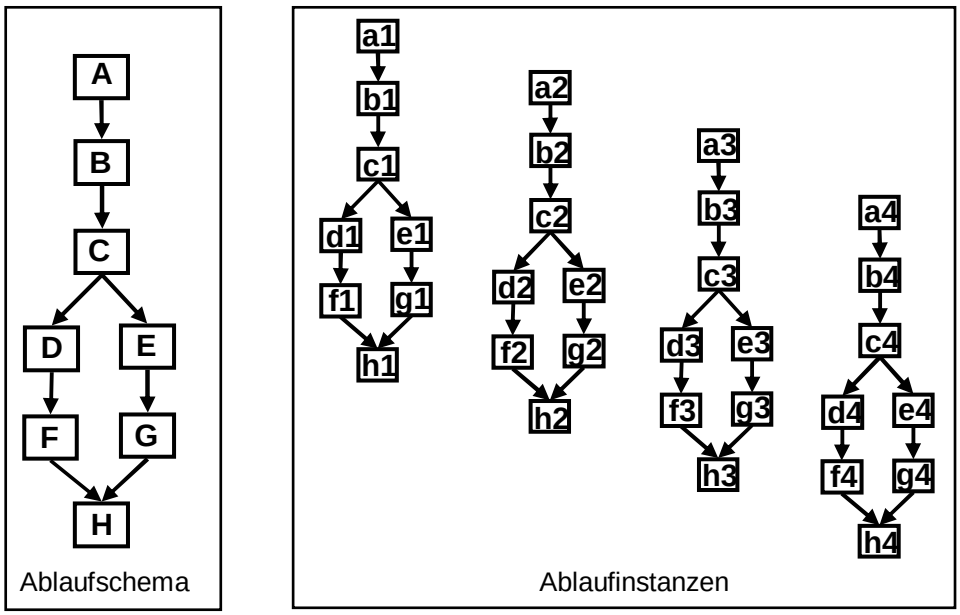

Abbildung 8: Trennung von Ablaufschema und Ablaufinstanzen

Demgegenüber betreffen Änderungen am Ablaufschema alle künftigen Ablauf-Instanzen zu diesem Schema. Wichtig ist, dass alte Ablaufschemata archiviert werden, um später Ablaufausführungen der Vergangenheit nachvollziehen zu können. Bei Änderungen an einem Ablaufschema muss geklärt werden, ob die gerade laufenden Ablaufausführungen bereits davon betroffen sind, oder ob diese Abläufe noch nach dem alten Schema ausge-

führt werden sollen. Am einfachsten ist die Änderung, wenn alle laufenden Ablaufausführungen vor Änderung des Schemas abgeschlossen werden. Bei lang laufenden Abläufen ist das aber offensichtlich nicht immer möglich.
Abbildung 9 zeigt, wie die Abstraktionskonzepte Generalisierung und Spezialisierung auf Abläufe angewandt werden können: Zu einem gegebenen Meta-Ablaufschema können verschiedene Ablaufschemata als Spezialisierung angegeben werden.
In Abbildung 9 werden zu dem Meta-Ablaufschema AM die Ablaufschemata AM1, AM2 und AM3 als Spezialisierung angegeben. In AM1 gibt es zu den Aktivitäten D und F zwei zusätzliche Aktivitäten E und G, die parallel ausgeführt werden, in AM2 gibt es zwei parallele Teilabläufe zu D und F (E und G sowie F und J), in AM3 gibt es zu Aktivität B die parallele Aktivität K und zu D und F die parallelen Aktivitäten E und G.

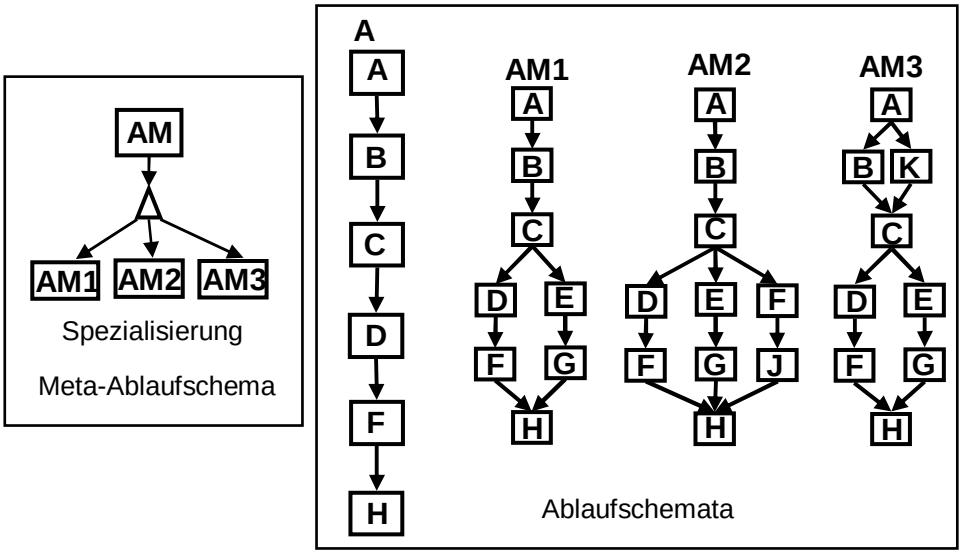

Abbildung 9: Meta-Ablaufschema mit Spezialisierung

In workflowbasierten Informationssystemen wird die Änderung am Source-Code ersetzt durch Änderungen am Ablaufschema. Idealerweise sollte die Änderung am Ablaufschema durch komfortable Editoren unterstützt werden, so dass auch (geschulte) Endanwender diese Änderungen durchführen können sollten. Bisherige Modellierungssprachen und -werkzeuge erfüllen diesen hohen Anspruch aber noch nicht. Insbesondere die Änderung laufender Workflows (und damit verbunden Fragen der Modifikation des Transaktionskonzeptes) stellt sich als schwieriges Problem dar, zu dem es verschiedene Vorschläge in der Literatur gibt (vgl. etwa [BDK99, GHM96, HSW01, ZeZ02] und die dort angegebene weiterführende Literatur).

7. Ausblick

Traditionelle Workflow-Managementsysteme unterstützen die Ausführung von unternehmensinternen Geschäftsprozessen. E-Business ist demgegenüber durch unternehmensübergreifende Geschäftsprozesse gekennzeichnet, die über das World Wide Web eine direkte Kopplung der Workflows eines Unternehmens an die Kunden ermöglichen. Die an der Schnittstelle zum Kunden derzeit eingesetzten Technologien sind allerdings noch relativ starr im Hinblick beipielsweise auf die (schnelle) Berücksichtigung von individuellen Sonderwünschen der Kunden. Dabei hat der Kunde üblicherweise weniger Interesse an der Struktur des entsprechenden Geschäftsprozesses, die im Rahmen konventioneller Geschäftsprozessoptimierung betrachtet wird. Der Kunde interessiert sich vielmehr für das Produkt, das er letztendlich erhält. Gewünscht wird insbesondere im Finanzdienstleistungsbereich oder im Telekommunikationsbereich eine möglichst weitreichende Individualisierung der angebotenen Produkte im Hinblick auf spezifische Kundenbedürfnisse. Herkömmliche Verfahren zur Informationssystementwicklung etwa bei der Bestandsführung von Versicherungen sind bzgl. Time-to-Market zu langsam, um Kundenbedürfnisse zeitnah in passende Produkte mit den entsprechenden Softwaresystemen umzusetzen. Eine schnelle und effiziente Umsetzung von Produktidee bzw. Kundenbedürfnissen in entsprechende Produkte und dazu passende Softwaresysteme wird der entscheidende Wettbewerbsfaktor im globalen Finanzdienstleistungsmarkt der Zukunft sein.
Ein Produkt (etwa ein spezielles Finanz-Derivat oder eine Lebensversicherung) besitzt aus Informatiksicht einen strukturellen Teil mit entsprechenden Datenwerten und einen Prozessteil. Eine passende Produktmaschine ist als erweitertes Workflow-Managementsystem zu verstehen, das neben der Prozessbeschreibung auch die Produktbeschreibung interpretieren und so in ein lauffähiges System umsetzen kann.
In einem laufenden Kooperationsprojekt zwischen der Goethe-Universität Frankfurt (Institut für Wirtschaftsinformatik) und der Universität Karlsruhe (Institut AIFB) werden entsprechende generische Modellierungswerkzeuge entwickelt, die Kunden die komfortable Online-Spezifikation von Finanzdienstleistungsprodukten ermöglichen. Dazu soll ein Interpreter (Produktmaschine, im Sinne einer erweiterten Workflow-Engine) für Produktbeschreibungen konzipiert werden. Als Grundlage für die Beschreibung des strukturellen Anteils von Produkten sollen XML bzw. die entsprechenden XML-Dialekte für den Finanzdienstleistungsbereich eingesetzt werden. Für den Prozessbeschreibungsteil werden höhere Petri-Netze (sog. XML-Netze) eingesetzt, die hier als dynamische Erweiterung für XML dienen [LeO03]. Neben den genannten Anwendungsbereichen sollen auch Digitale Bibliotheken bzw. Wissensportale exemplarisch betrachtet werden.

Literaturverzeichnis

[BDK99] Bernstein, A.; Dellarocas, C.; Klein, M.: Towards Adaptive Workflow Systems, CSCW-98 Workshop Report, in: SIGMOD Record, Vol. 28, No. 3, Sept. 1999, S. 7-8.

[CGD01] Chang, E.; Gautama, E.; Dillon, T.S.: Extended Activity Diagrams for Adaptive Workflow Modelling, in: Proc. of the Fourth Int. Symp. on Object-Oriented Real-Time Distributed Computing (ISORC'01), IEEE, 2001.

[Dav99] Davenport, Th.: Passt Ihr Unternehmen zur Software?, Harvard Business manager, 1/99, S. 89-99.

[GHM96] Georgakopoulos, D.; Hornick, M.F.; Manola, F.: Customizing Transaction Models and Mechanisms in a Programmable Environment, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 4, 1996, S. 630-649.

[HSW01] Halliday, J.J.; Shrivastava, S.K.; Wheater, S.M.: Flexible Workflow Management in the OPENflow system, in: Proc. of the Fifth Int. Enterprise Distributed Object Computing Conference (EDOC'01), IEEE, 2001.

[LeO03] Lenz, K.; Oberweis, A.: Interorganizational Business Process Management with XML Nets, in: Ehrig, H.; Reisig, W.; Rozenberg, G; Weber, H. (Hrsg.): Petri Net Technology for Communication Based Systems, LNCS 2472, Springer-Verlag, 2003.

[Obe96] Oberweis, A.: Modellierung und Ausführung von Workflows mit Petri-Netzen, B.G. Teubner-Verlagsgesellschaft, 1996.

[ObS91] Oberweis, A.; Stucky, W.: Die Behandlung von Ausnahmen in Software-Systemen, Wirtschaftsinformatik, 33. Jahrgang, Heft 6, 1991, S. 492-502.

[SHN01] Saleh, J.H.; Hastings, D.E.; Newman, D.J.: Extracting the Essence of Flexibility in System Design, in: Proc. of the Third NASA/DoD Workshop on Evolvable Hardware (EH'01), IEEE, 2001.

[SmS77] Smith, J.M.; Smith, D.C.P.: Database Abstractions: Aggregation and Generalization. ACM Transactions on Database Systems, Vol. 2, No. 2, Juni 1977, S. 105-133.

[StM95] Strong, D.M.; Miller, S.M.: Exceptions and Exception Handling in Computerized Information Processes, ACM Transactions on Information Systems, Vol. 13, No. 2, April 1995, S. 206-233.

[Wed01] Wedemeijer, L.: Engineering for Conceptual Schema Flexibility, in: Proceedings of the 11th International Workshop on Research Issues in Data Engineering (RIDE 01), Heidelberg, 2001, pp. 85-94.

[ZeZ02] Zeng, D.D.; Zhao, J.L.: Achieving Software Flexibility via Intelligent Workflow Techniques, in: Proc. of the 35th Hawaii International Conference on System Sciences, 2002.