Der folgende Artikel ist im OBJEKTspektrum 2/99, S. 78 - 83 erschienen.

Frameworkbasierte Anwendungsentwicklung (Teil 2): Die Konstruktion interaktiver Anwendungen

Wolf-Gideon Bleek, Guido Gryczan, Carola Lilienthal, Martin Lippert,
Stefan Roock, Henning Wolf, Heinz Züllighoven

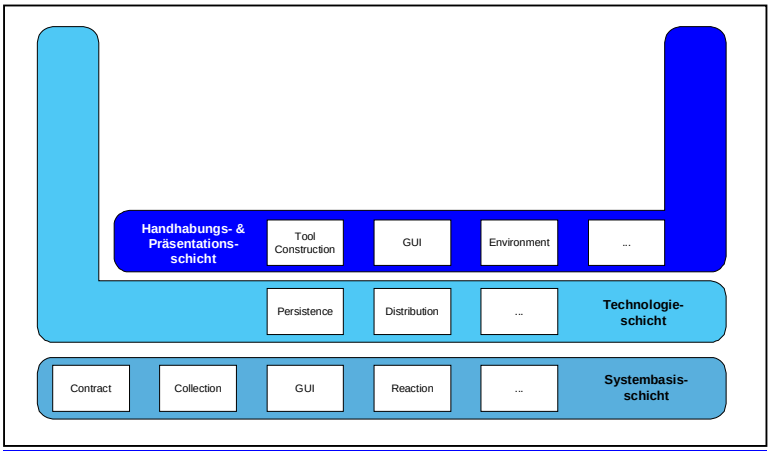
In der letzten Ausgabe von OBJEKTspektrum haben wir eine fachlich motivierte Modellarchitektur für große interaktive Anwendungssysteme vorgeschlagen. Zwei wesentliche Komponenten sind dabei "Werkzeuge" und "Materialien". Werkzeuge, die fachlich die Arbeitsmittel im Anwendungsbereich repräsentieren, kapseln softwaretechnisch die Interaktion mit dem Benutzer. In diesem Beitrag behandeln wir die Probleme beim Bau dieser interaktiven Komponenten und zeigen, wie Frameworks bei der Bewältigung der Probleme eingesetzt werden können. Wir untersuchen, welche Möglichkeiten uns andere Frameworks bieten, und machen deutlich, welche Unterstützung vom JWAM-Framework geboten wird.

Zur Erinnerung: Bei der Erledigung alltäglicher Aufgaben verwenden wir Arbeitsmittel und -gegenstände. Bei der Gestaltung von Anwendungssoftware modellieren wir diesen Zusammenhang in den Entwurfsmetaphern Werkzeug und Material. Softwarewerkzeuge sind die interaktiven Komponenten, mit den wir als Benutzer Materialien bearbeiten.

Werkzeuge sind durch die Menge der benutzten Komponenten und der einzuhaltenden Schnittstellen, die beispielsweise das Fenstersystem vorgibt, aufwendig zu erstellen. Viele der Komponenten, die der Interaktion dienen, orientieren sich nicht an fachlichen Strukturen des Anwendungsgebiets; ihr Zusammenspiel ist ereignisgesteuert. Zudem strukturieren unterschiedliche Entwickler ihre Software verschieden, und es gibt keine Einheitlichkeit über Einzelprojekte hinaus. Dies erschwert die Entwicklung großer durchgängiger Anwendungssysteme aus wiederverwendbaren und wiedererkennbaren Werkzeugen.

Kasten 1: Das JWAM-Framework

Das JWAM-Rahmenwerk wird seit 1½ Jahren am Arbeitsbereich Softwaretechnik der Universität Hamburg von Studenten und wissenschaftlichen Mitarbeitern (weiter-)entwickelt. Die erste Version (JWAM 1.0) basierte auf dem JDK 1.1 und diente als Grundlage für Studien- und Diplomarbeiten. Im April 1998 haben wir die Version 1.1 freigegeben. Diese Version wurde und wird weiterhin in Studien- und Diplomarbeiten verwendet, zusätzlich jedoch auch in verschiedenen Lehrveranstaltungen. Die Version 1.2 wird auf dem JDK 1.2 basieren und ist für Anfang 1999 geplant. Das Framework wird auch in Kooperationsprojekten mit der Industrie eingesetzt.
Das Framework enthält in der Version 1.1 40 Packages, 150 Klassen und 44 Interfaces. Informationen über das JWAM-Rahmenwerk können über das Internet bezogen werden, unter der Adresse:
http://swt-www.informatik.uni-hamburg.de/Software/JWAM



In unseren Kooperationsprojekten haben wir häufig festgestellt, daß die Benutzungsschnittstelle eines interaktiven Werkzeugs weit weniger stabil ist als die zugrunde liegende fachliche Funktionalität. Orientiert sich aber die Struktur des Anwendungssystems an der Art der Interaktion, d. h. ist fachliche Funktionalität direkt mit interaktiven Elementen verknüpft, dann sind die fachlichen Auswirkungen von Änderungen an der Benutzungsschnittstelle nicht mehr überschaubar. Der Aufwand für eine Portierung auf andere Betriebssysteme oder auch nur der Austausch der GUI-Bibliothek kommen dann einer Neuentwicklung gleich. Wir wollen diesen Aufwand möglichst minimieren.

Zu diesem Zweck benötigen wir ein eigenständiges Konzept, daß die nicht-fachlichen Anteile des Werkzeugbaus vereinheitlicht. Die Grundlage hierfür ist für uns die Strukturähnlichkeit (vgl. [Gry99], [Zül98]):

Fachliche Gegenstände und Konzepte sollen sich in den Strukturen der Anwendungssoftware und eines Frameworks wiederfinden lassen. Dieses Grundprinzip ergänzen wir durch eine softwaretechnische Forderung: Die systemabhängigen, fehleranfälligen und generischen Anteile eines Werkzeugs wollen wir in ein Framework verlagern, damit sich der Entwickler auf die Implementierung fachlicher Funktionalität konzentrieren kann.

Konstruktionsprinzipien reaktiver Anwendungssysteme

Wir wollen also die technischen Teile eines interaktiven Anwendungssystems soweit von den anwendungsspezifischen trennen, daß sie in einem Framework vorformuliert und implementiert werden können. Dazu beschränken wir uns auf eine bestimmte Klasse von Softwaresystemen. In Teil 1 dieser Serie haben wir erläutert, daß wir interaktive Anwendungssysteme betrachten.

Jetzt schränken wir dies auf `reaktive interaktive Systeme` ein, d. h. Systeme, die auf Benutzeraktionen reagieren und nicht von sich aus agieren. Diese Beschränkung scheint uns gerechtfertigt, da wir damit eine große Anzahl fachlich motivierter Anwendungen abdecken und gleichzeitig die Bandbreite der Designalternativen einschränken, um zu tatsächlich verwendbaren Abstraktionen zu kommen.

Welche technischen Konzepte des Werkzeugbaus können in einem Framework hinterlegt werden? Wir zählen dazu:

- Werkzeuge erzeugen und löschen (generisch und systemabhängig),
- Werkzeug und Material verknüpfen (generisch),
- Fenstersystembibliotheken anbinden bei größtmöglicher Unabhängigkeit vom konkreten Fenstersystem (systemabhängig),
- Werkzeuge in Teilwerkzeuge sowie in funktionale und anzeigende Komponente zerlegen und gleichzeitig Werkzeugdienste zusammenfassen, die für ein ganzes Werkzeug gelten (generisch), sowie
- Werkzeugkomponenten über Reaktionsmechanismen verknüpfen, z. B. durch einen Ereignismechanismus, eine Zuständigkeitskette, ein Kommandomuster oder eine asynchrone Nachrichtenvermittlung (generisch).

Werden diese Dienste in einem Framework zur Verfügung gestellt und von den Anwendungsentwicklern durchgängig verwendet, haben die in einzelnen Projekten erstellten Anwendungssysteme eine einheitliche technische Struktur. Zudem können sich die Anwendungsentwickler auf die Konstruktion einzelner Werkzeuge für fachlich motivierte Aufgabenstellungen konzentrieren. Die Probleme des technischen Werkzeugbaus selbst können bei der Entwicklung des Frameworks von der dafür spezialisierten Architekturgruppe geklärt werden.

Heute ist es möglich, Frameworks zu finden, die genau eines der genannten Probleme -- z. B. Anbindung eines Fenstersystems -- lösen; die Schwierigkeit liegt jedoch in der Kombination. In [All95] und [Mat97] wird ausgeführt, warum die Kombination unterschiedlicher Frameworks scheitert oder zu einem erheblichem Anpassungsaufwand führt. Wesentlich ist, daß in vielen Frameworks bereits ein Kontrollfluß vordefiniert ist und die vom Anwendungsentwickler erstellten Teile durch Komponenten des Frameworks gerufen werden. Werden mehrere Frameworks eingesetzt, so ist nicht mehr eindeutig, wer den Kontrollfluß bestimmt. Das Anwendungssystem ist dann entweder nicht konstruierbar, oder wir erhalten ein agierendes, nicht ein reaktives Anwendungssystem. Beide Konsequenzen sind für uns unbefriedigend. Bei der Entwicklung von Frameworks für die Werkzeugkonstruktion ist demnach darauf zu achten, daß Anwendungssysteme mit nur einem Kontrollfluß konstruiert werden können.

Frameworks zur Konstruktion interaktiver Anwendungssysteme

Drei Arten von Frameworks erleichtern dem Entwickler heute im wesentlichen die Erstellung interaktiver Anwendungssysteme:

- reine GUI-Frameworks, wie z. B. "Zapp" und "StarView",
- Frameworks zur Bearbeitung zusammengesetzter Dokumente (`Compound Documents`), wie z. B. die "Microsoft Foundation Classes" und "OpenDoc",
- anwendungsspezifische Frameworks, die auch eine Technologieschicht für die Erstellung interaktiver Anwendungssysteme beinhalten, wie z. B. "San Francisco" und "JWAM" (vgl. Kasten 1).

GUI-Frameworks stellen dem Anwendungsentwickler mindestens eine objektorientierte Abstraktion von Oberflächenelementen und eine Klasse zum Starten und Beenden der Anwendung zur Verfügung. Die Anwendung wird in der Regel nach Fenstern und Dialogen strukturiert, an welche die fachliche Funktionalität direkt angebunden ist. Das GUI-Framework allein gibt dem Anwendungsentwickler noch keine Anleitung für die Strukturierung der fachlichen Anteile, da es ihm im wesentlichen die Anbindung an das Fenstersystem erleichtert (vgl. [Bäu96]).

Die Idee von Compound Document Frameworks besteht darin, interaktive Anwendungen als Dokumenten-Editoren mit zugehörigem Dokumenttyp zu konstruieren (siehe [Orf96], Kap. 14). In ein Dokument können dabei Dokumente anderen Typs eingefügt werden. Auf diese Weise lassen sich mit einem Compound Document Framework große monolithische Anwendungen in kleinere, besser handhabbare Elemente aufteilen. Ein solches Framework strukturiert interaktive Anwendungen durch die Trennung in Editor und bearbeitetes Dokument sowie durch das rekursive Zusammensetzen von Dokumenten aus anderen Dokumenten.

Genau wie GUI-Frameworks sind auch Compound Document Frameworks für die von uns diskutierte Klasse von Anwendungen nicht mächtig genug. Dies spricht nicht prinzipiell gegen diese Frameworks. Wie wir aber schon in Teil 1 dieser Serie ([Gry99]) festgestellt haben, brauchen wir für die Entwicklung großer interaktiver Softwaresysteme eine weitergehende Unterstützung. Ein Anwendungsframework muß so in Schichten organisiert sein, daß Entwurfsentscheidungen thematisch gruppiert und gekapselt werden können.

Das Anwendungsframework San Francisco basiert wie JWAM auf einer Schichtenarchitektur und besteht aus unterschiedlichen, sowohl technologischen als auch anwendungsspezifischen Frameworks. Die drei in San Francisco integrierten Schichten (vgl. Abb. 1) sind:

- "Foundation",
- "Common Business Objects" (CBOs),
- "Core Business Processes" (CBPs).

Die Frameworks in diesen Schichten erleichtern es dem Entwickler, interaktive, plattformunabhängige und verteilte Anwendungssystem zu erstellen, indem sie die Struktur interaktiver Anwendungen vorgeben.

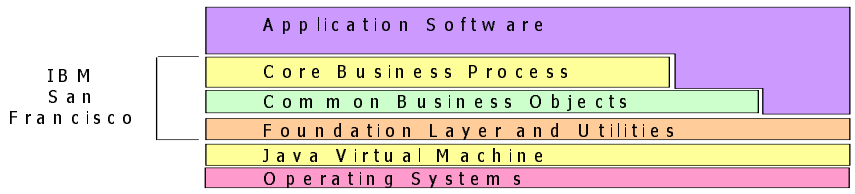


Abbildung 1: Das Schichtenmodell von San Francisco

Der Foundation Layer legt unabhängig vom konkreten Anwendungsgebiet fest, wie interaktive Anwendungssysteme erstellt werden. Diese Schicht ist vergleichbar mit den Schichten Systembasis, Technologie sowie Handhabungs- und Präsentationsschicht des JWAM-Frameworks, die wir in OBJEKTspektrum 1/99 vorgestellt haben.

Eine unter Verwendung dieses Frameworks entwickelte Anwendung wird in die Komponenten Model, View und Maintainer aufgeteilt. Über das Model werden die anwendungsspezifischen Klassen mit dem interaktiven Werkzeug verbunden. Der Entwickler kann sowohl eigene anwendungsspezifische Klassen konstruieren, als auch sich auf die Elemente der beiden höhergelegenen Schichten, der CBOs und der CBPs, abstützen. Letzteres wird von der Firma IBM empfohlen. Dazu werden für betriebswirtschaftliche Anwendungen Business Objects (also passende Datenstrukturen) und Business Processes (Klassen mit vordefinierten betriebswirtschaftlichen Anwendungen) zur Verfügung gestellt. Alle auf diesen beiden Schichten aufbauenden Anwendungen können einfach untereinander "Daten" austauschen, auch wenn sie von unterschiedlichen Herstellern entwickelt wurden.

Die Architektur des JWAM-Frameworks unterscheidet sich vom San-Francisco-Framework im wesentlichen dadurch, daß sie keine Annahmen über die Art der Anbindung der anwendungsspezifischen Teile macht. Wir haben zwar genaue Vorstellungen davon, wie die Schichtenarchitektur so um anwendungsspe-

zifische Schichten erweitert werden kann, daß eine `Strukturähnlichkeit` von anwendungs-fachlichen und technischen Konzepten entsteht (siehe [Bäu98]). Im JWAM-Framework werden hierfür jedoch keine Vorgaben gemacht.

Domänenspezifische Frameworks, wie sie etwa in der GEBOS-Projektfamilie eingesetzt werden (siehe [Bäu97]), enthalten anwendungs-fachliche und organisationsspezifische Komponenten, die außerhalb der jeweiligen Domäne kaum wiederzuverwenden sind. Wir wollen mit JWAM keine bestimmte Domäne unterstützen, sondern die generellen Prinzipien einer interaktiven Anwendung nach dem Werkzeug&Material-Ansatz realisieren. Der Verzicht auf die Integration anwendungsspezifischer Komponenten bedeutet, daß es mit dem JWAM-Framework möglich ist, interaktive Anwendungen für die unterschiedlichsten Anwendungsgebiete -- z. B. den Bankenbereich, die Anlagenautomatisierung oder die Computer/Telefon-Integration (vgl. [Ble97]) -- zu erstellen.

Konstruktion interaktiver Werkzeuge mit dem JWAM-Framework

Im folgenden skizzieren wir, wie mit dem JWAM-Framework interaktive Werkzeuge gebaut werden können. Dazu verwenden wir ein Beispiel -- das "Kassenwerkzeug". An diesem führen wir aus, wie wir mit Hilfe eines Frameworks die Trennung von Funktionalität und Präsentation sowie die lose Kopplung über einen Ereignismechanismus realisieren. Für die jeweiligen Frameworkkomponenten zeigen wir, in welcher der drei Schichten des JWAM-Frameworks -- Systembasisschicht, Technologieschicht oder Handhabungs- und Präsentationsschicht (siehe dazu auch [Gry99]) -- sie angesiedelt sind.

Kasten 2: Das Kassenwerkzeug-Beispiel

Das Kassenwerkzeug steht unter der Adresse *http://swt-www.informatik.uni-hamburg.de/Software/JWAM* unter der Version 1.2 des Frameworks zum Herunterladen bereit. Da das JWAM-Framework englischsprachige Bezeichner verwendet, weichen die Bezeichnungen im Beispiel leicht von denen in diesem Artikel verwendeten ab:

- Funktionskomponenten werden mit ‘fp’ abgekürzt: functional part
- Interaktionskomponenten werden mit ‘ip’ abgekürzt: interaction part
- Werkzeuge werden als ‘tools’ bezeichnet.

Beispiel "Kassenwerkzeug

Das Kassenwerkzeug ist ein "Minibeispiel" (siehe auch Kasten 2) ohne fachlichen Tiefgang, das primär zur Demonstration dient. Abbildung 2 zeigt die Benutzungsoberfläche des Kassenwerkzeugs. Neben Kontonummer und Name des Kontoinhabers wird der Saldo des Kontos angezeigt. Der Benutzer kann in das Betragsfeld einen Betrag eintragen, der entsprechend dem Knopfdruck ein- oder ausgezahlt wird.

Abbildung 2: Oberfläche des Kassenwerkzeugs

Das Kassenwerkzeug arbeitet auf dem Material **`Konto`**. Diese Materialklasse implementiert die Operationen, um ein Konto zu verändern und zu sondieren. Abbildung 3 zeigt die prinzipielle Konstruktion.

Zwischen Werkzeug und Material besteht technisch eine Benutzt-Beziehung. Diese Beziehung ist über das `Vertragsmodell` abgesichert, dessen Realisierung in weiteren Verlauf dieser Serie erläutert wird. Das Material benutzt aus der JWAM-Systembasisschicht nur das `Contract-Sub`-Framework zur Realisierung des Vertragsmodells.

Das Werkzeug ist sehr viel enger als das Material mit dem JWAM-Framework verbunden.

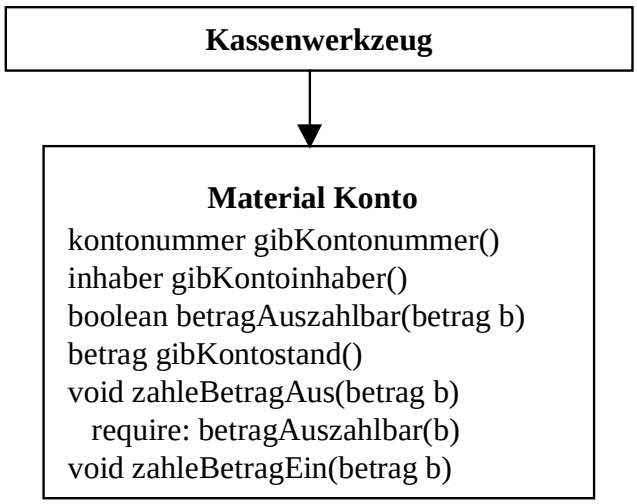


Abbildung 3: Das Kassenwerkzeug arbeitet auf dem Material Konto

Wie oben dargestellt, ist der prinzipielle Kontrollfluß für ein Werkzeug bereits im Framework formuliert. Dabei geht es um das Zusammenspiel der Funktionskomponente mit einer Interaktionskomponente (vgl. [Zül98], [Bäu95]). Folgende abstrakte Oberklassen unterstützen den Bau von Werkzeugkomponenten:

- **`fkBasis`**: Die Funktionskomponente realisiert die fachliche Funktionalität eines Werkzeugs. Dies läßt sich nicht abstrakt vorformulieren. Allerdings muß jede relevante Zustandsänderung der Funktions- an die Interaktionskomponente gemeldet werden. Dazu ermöglicht die entsprechende Basisklasse die Definition von Ereignissen. Zudem unterstützt **`fkBasis`** die Koordination verschiedener Werkzeuge untereinander.
- **`ikBasis`**: Die Interaktionskomponente arbeitet auf der Funktionskomponente. Dazu hält die Oberklasse für Interaktionskomponenten einen Verweis auf die zugehörige Funktionskomponente und bietet die Möglichkeit, sich bei Ereignissen der Funktionskomponente zu registrieren. Die Interaktionskomponente muß auch an die Benutzungsoberfläche angebunden werden. Dazu definiert das Framework spezielle Komponenten, sogenannte `Interaktionsformen`. Diese werden über das `Kommando`-Muster (vgl. [Gam98]) angebunden, das in der Oberklasse **`ikBasis`** formuliert ist.
- **`wzBasis`**: Jedes Werkzeug wird durch ein Objekt repräsentiert. Die entsprechende Basisklasse erzeugt Funktions- und Interaktionskomponenten und verbindet diese. Auf der Grundlage dieser Basisklasse ist im JWAM-Framework die Verwaltung von Werkzeugen implementiert. Dies ist Aufgabe der Umgebung -- sie verwaltet alle Werkzeuge, die am Arbeitsplatz gestartet wurden.

Das Kassenwerkzeug spezialisiert die folgenden Basisklassen (siehe Abb. 4):

- Die Werkzeugfunktionalität wird in einer Funktionskomponente (**`fkKassenwerkzeug`**) realisiert. Diese orientiert sich am fachlichen Umgang, den Anwender mit diesem Werkzeug haben sollen. Daher bietet die Funktionskomponente an der Schnittstelle Operationen wie **`zahleEin`**, **`zahleAus`**, **`speichere`** an. **`fkKassenwerkzeug`** erbt von **`fkBasis`**.
- Die zugehörige Interaktionskomponente (**`ikKassenwerkzeug`**) verwaltet die grafische Benutzungsoberfläche und empfängt von dieser Kommandos, wenn z. B. ein Knopf vom Benutzer gedrückt wird. Sie setzt die Benutzeraktionen (soweit diese anwendungs-fachlich und nicht nur auf die Präsentation bezogen sind) unter Nutzung der Klasse **`fkKassenwerkzeug`** um. **`ikKassenwerkzeug`** erbt von **`ikBasis`**.
- Das gesamte Werkzeug wird durch sein Werkzeugobjekt (**`wzKassenwerkzeug`**) repräsentiert. Dieses Objekt erzeugt erst ein Exemplar der Klasse **`fkKassenwerkzeug`** und anschließend eine passende Interaktionskomponente (hier **`ikKassenwerkzeug`**). Der Interaktionskomponente (kurz **`IK`**) wird dazu ihre Funktionskomponente (kurz **`FK`**) im Konstruktor bekanntgemacht. Die Werkzeugklasse beerbt die Basisklasse **`wzBasis`**.

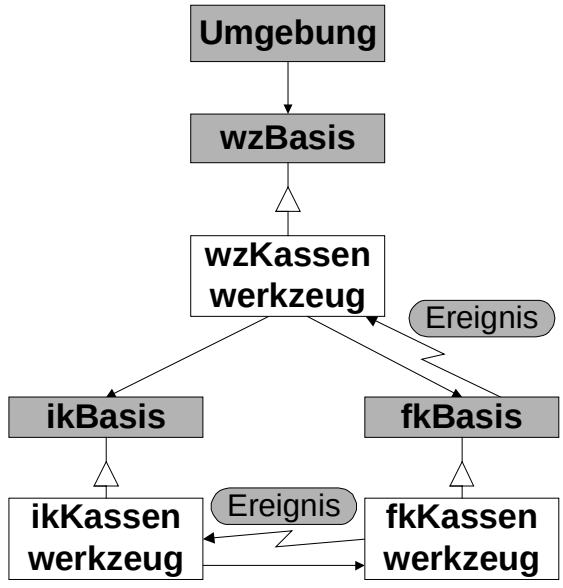


Abbildung 4: Zusammenhang der Klassen eines Werkzeugs. Grau dargestellte Klassen befinden sich im JWAM-Framework.

- IK und FK sind über einen allgemeinen Ereignismechanismus des Frameworks aneinander gekoppelt. Die IK meldet sich im Konstruktor bei den sie interessierenden Ereignissen der FK an. In unserem Beispiel ist das **`kontoGeaendert`**. Immer wenn sich der Saldo des Kontos geändert hat, löst die FK das entsprechende Ereignis aus. So wird die IK benachrichtigt, welche daraufhin ihre Anzeige entsprechend ändern kann.

`ikKassenwerkzeug` und **`fkKassenwerkzeug`** sind von Basisklassen des Frameworks abgeleitet. Funktionalität, wie z. B. das Starten und Beenden von Werkzeugen, sind bereits in den Basisklassen formuliert und werden mit Hilfe einer Zuständigkeitskette (`Chain-Of-Responsibility` , siehe [Gam98]) an die Oberklassen delegiert. Die Zuständigkeitskette kann von der FK darüber hinaus für ihre eigenen Zwecke verwendet werden. Das Kassenwerkzeug ist allerdings zu einfach, um diesen Mechanismus demonstrieren zu können. Er kommt jedoch in sehr vielen komplexeren Werkzeugen zum Einsatz.

Um den Zustand des Werkzeugs (hier größtenteils durch das Material "Konto" bestimmt) mit der Präsentation an der Benutzungsschnittstelle zu synchronisieren, wird der Ereignismechanismus eingesetzt. Dies läßt sich am Einzahlen auf eine Konto verdeutlichen. Nach der Eingabe eines Betrags betätigt der Anwender den Knopf "Betrag einzahlen". Die dadurch ausgelöste Objektinteraktion ist in Abbildung 5 dargestellt.

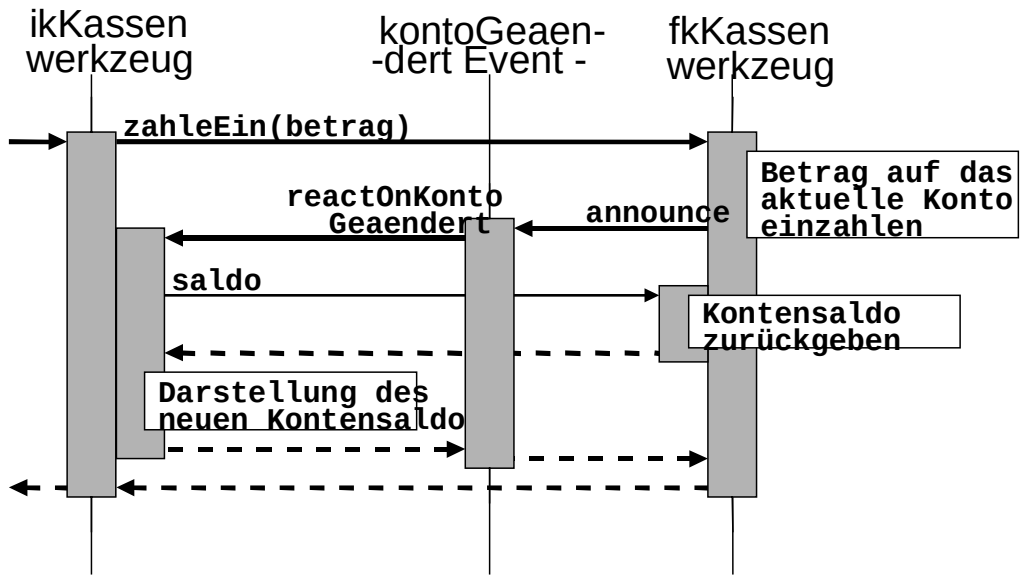


Abbildung 5: Interaktion zwischen Interaktions- und Funktionskomponente beim Aufruf der Methode zum Einzahlen

Die Interaktionskomponente (**`ikKassenwerkzeug`**) ruft die Funktionskomponente (**`fkKassenwerkzeug`**) mit der Methode **`zahleEin`**. Die FK ändert das Material entsprechend. Daraufhin aktiviert die FK das Ereignis **`kontoGeaendert`**, welches die IK benachrichtigt. Als Reaktion auf diese Benachrichtigung aktualisiert die IK die Saldoanzeige: Im Rahmen eines synchronen Kontrollflusses erfragt die IK den aktuellen Saldo an der FK, stellt ihn dar und gibt den Kontrollfluß über das Ereignis an die FK zurück. Mit Hilfe dieser Entkopplung von IK und FK über den Ereignismechanismus kann sichergestellt werden, daß in der FK kein Wissen darüber existiert, wie und in welchem Zusammenhang Informationen präsentiert werden. Die Entscheidung darüber trifft allein die IK.

Resümee

Fassen wir zusammen, welche Teile beim Bau interaktiver Werkzeuge durch ein Framework bereits abgedeckt werden können. Im JWAM-Framework werden die folgenden Konzepte mit passenden (teilweise abstrakten) Implementierungen bereitgestellt:

- Vertragsmodell: Verträge können durch die Benutzung einer speziellen Klasse und die Implementierung eines **`Interface`** benutzt werden.
- Trennung von Werkzeugen in interaktive und funktionale Teile: Um Werkzeuge zu implementieren, müssen die drei oben beschriebenen Basisklassen beerbt werden.
- Benachrichtigungen von der FK zur IK über einen Ereignismechanismus: Der Ereignismechanismus kann von FK und IK durch Benutzung der Ereignisklassen verwendet werden.

- Anbindung von Werkzeugen an die Benutzungsoberfläche mit Interaktionsformen und Kommandos: Ähnlich wie Ereignisse können Interaktionsformen und Kommandos von der IK durch Benutzung verwendet werden.
- Delegation von Verantwortlichkeiten zwischen Werkzeugkomponenten mit der Zuständigkeitskette: Funktionskomponenten können diesen Mechanismus benutzen.
- Starten, Beenden und Verwalten von Werkzeugen durch die Umgebung: Diese Funktionalität ist soweit im Framework gekapselt, daß der Anwendungsprogrammierer im Normalfall nicht damit in Berührung kommt.

In Abbildung 6 haben wir den Zusammenhang zwischen unserem Beispielwerkzeug und den Komponenten des JWAM-Frameworks in den verschiedenen Schichten herausgehoben.

Die Erläuterungen in diesem Abschnitt machen deutlich, daß für das Kassenwerkzeug der größte Codeanteil im Framework steckt. Man kann mit Recht argumentieren, daß sich für ein solch kleines Beispiel die Verwendung eines Frameworks nicht lohnt, denn der Einarbeitsaufwand in das Framework ist viel zu hoch. Allerdings darf nicht vergessen werden, daß ein Framework die einheitliche Konstruktion von Werkzeugen sicherstellt. Darüber hinaus ist bei der Entwicklung mehrerer, vor allem komplexer Werkzeuge der Mehraufwand für die Einarbeitung schnell wieder eingespart.

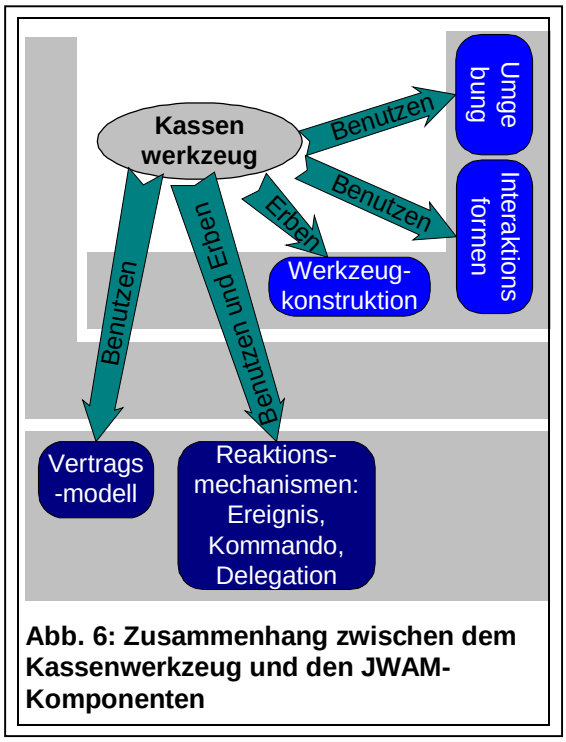


Abb. 6: Zusammenhang zwischen dem Kassenwerkzeug und den JWAM-Komponenten

Ausblick

In diesem Artikel haben wir grundsätzliche Probleme und Lösungen bei der Konstruktion interaktiver Werkzeuge aufgezeigt und Frameworks zur Werkzeugkonstruktion vorgestellt. Im folgenden Teil der Serie beschäftigen wir uns mit einem Thema, das gerade im schnellebigen Umfeld von Java von besonderer Bedeutung ist: die Anbindung einer grafischen Benutzungsschnittstelle an interaktive Anwendungen und damit einhergehend die Bindung an einen GUI-Builder. Wir werden die für JWAM gewählte Lösung vorstellen, die sich dadurch auszeichnet, daß jeder JavaBean-fähige GUI-Builder verwendet werden kann, um die Präsentation von Anwendungsprogrammen zu gestalten.

Literatur

[All95] R. Allen, D. Garlan, J. Ockerbloom, Architectural Mismatch or Why it's hard to build systems out of existing parts, in: Proc. of the 17th International Conference on Software Engineering, Seattle WA, April 1995

[Bäu95] D. Bäumer, R. Knoll, G. Gryczan, W. Strunk, H. Züllighoven, Objektorientierte Entwicklung anwendungsspezifischer Rahmenwerke, in: OBJEKTspektrum 6/95

[Bäu96] D. Bäumer, W.R. Bischofberger, H. Lichter, H. Züllighoven, User Interface Prototyping -- Concepts, Tools, and Experience, in: Proc. of the 18th International Conference on Software Engineering, Berlin, March 1996

[Bäu97] D. Bäumer, G. Gryczan, R. Knoll, C. Lilienthal, D. Riehle, H. Züllighoven, Framework Development for Large Systems, in: Communications of the ACM, October 1997, Vol. 40, No. 10, S. 52-59

[Bäu98] D. Bäumer, Softwarearchitekturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme, Dissertationsschrift am FB Informatik der Universität Hamburg, Januar 1998

[Ble97] W.-G. Bleek, Techniken zur Konstruktion verteilter und technisch eingebetteter Anwendungssysteme, Diplomarbeit, Universität Hamburg, FB Informatik, Juli 1997

[Gam98] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software, 2. Auflage, Addison-Wesley, 1998

[Gry99] G. Gryczan, C. Lillienthal, M. Lippert, S. Roock, H. Wolf, H. Züllighoven, Frameworkbasierte Anwendungsentwicklung (Teil 1), in: OBJEKTspektrum 1/99

[IBM98] IBM, SanFrancisco Concepts and Facilities Chapter 8, siehe http://www.ibm.com/Java/Sanfrancisco/concepts/ibmsf.sf.SFConceptsAndFacilitiesDevelopingApplications.html

[JWAM98] JWAM: Java Framework for the Tools and Materials Approach, Universität Hamburg, FB Informatik, siehe http://swt-www.informatik.uni-hamburg.de/Software/JWAM

[Mat97] M. Mattsson, J. Bosch, Framework Composition: Problems, Causes and Solutions, Proc. TOOLS USA '97

[Orf96] R. Orfali, D. Harkey, J. Edwards, The Essential Distributed Objects Survival Guide, Wiley Computer Publishing, New York, 1996

[Zül98] H. Züllighoven, Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz, dpunkt-Verlag, 1998

Wolf-Gideon Bleek, Dr. Guido Gryczan, Carola Lilienthal, Martin Lippert, Stefan Roock, Henning Wolf und Prof. Dr.-Ing. Heinz Züllighoven arbeiten auf verschiedenen Positionen am Fachbereich Informatik, Arbeitsbereich Softwaretechnik der Universität Hamburg. Die Autoren bilden das JWAM-Kernteam. Sie sind per E-Mail unter jwaminfo@swt1.informatik.uni-hamburg.de zu erreichen.

Wolfgang Strunk ist Abteilungsleiter bei der Fa. Micrologica AG und beschäftigt sich dort mit der Architektur und Konzeption von softwarebasierten Call-Centern. Davor war er mehrere Jahre am Arbeitsbereich Softwaretechnik der Universität Hamburg tätig.