Das JWAM-Framework und Komponenten – Eine konzeptionelle Bestandsaufnahme[1]

Henning Wolf[+], Stefan Roock[+], Andreas Kornstädt[*],
Heinz Züllighoven[+], Guido Gryczan[*]

[+] *Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, Vogt-Kölln-Straße 30, 22527 Hamburg, Deutschland, Tel.: +49 (40) 42883 - 2413, Fax: -2303, E-Mail: {wolf|roock|zuelligh}@informatik.uni-hamburg.de, URL: http://www.jwam.de*

[*] *Apcon WPS GmbH im Verbund der itelligence AG, Friedrich-Ebert-Damm 143, 22047 Hamburg, Deutschland, Tel.: +49 (700) APCONWPS, Fax: +49 (40) 69424-210, E-Mail: {Andreas.Kornstaedt|Guido.Gryczan}@itelligence.de, URL: http://www.jwam.de*

Zusammenfassung. Im vielfach diskutierten Idealbild der Wiederverwendung von Softwarekomponenten werden komplette Anwendungssysteme mit minimalem Entwicklungsaufwand der Entwickler durch Glue-Code und Parametrisierung zusammengestellt. Gestützt auf unsere Erfahrungen aus Projekten mit dem JWAM-Framework stellen wir eine Sichtweise vor, bei der das Framework als architektonisch einheitliche Grundlage dient, auf der Komponenten überhaupt erst sinnvoll miteinander kombinierbar werden. Dazu grenzen wir zunächst die existierenden Komponentenkonzepte begrifflich voneinander ab und diskutieren anschließend die im Kontext von JWAM entstandenen Konzepte der Austattungs- und der Einschubkomponenten.

Schlüsselworte: Klassifikation; Framework; Ausstattungs- und Einschubkomponenten.

1 Kontext

Software-Komponenten sind schon seit langer Zeit ein Wunschtraum für viele Entwickler und vor allem IT-Manager. Dieser Wunschtraum beruht auf der Vorstellung, man könne Softwaresysteme so zerteilen, dass daraus für viele Anwendungsbereiche oder Aufgaben vorgefertigte Bausteine entstehen.

Die Komponentendiskussion wird aktuell angeheizt von der Idee eines boomenden Komponentenmarkts, auf dem sich IT-Manager preiswert und schnell fertige Subsysteme einkaufen können.

Die kühnsten Komponenten-Anhänger versprechen, dass die Entwicklung von Anwendungssoftware sich absehbar drastisch verändert. Geschulte Anwender sollen Standard-Komponenten auswählen, die sie anhand vorgegebener Einstellmöglichkeiten an die konkrete Situation anpassen und mit einem graphischen Editor zu einem fertigen System komponieren. Diese Ansicht teilen wir nicht.

---

[1] Der folgende Text ist eine deutsche Überarbeitung eines Abschnitts aus dem in 2001erscheinenden Konstruktionshandbuch für objektorientierte Anwendungssysteme, Hauptautor: Heinz Züllighoven.

Wenn wir die überzogenen Hoffnungen aber einmal außer Acht lassen, bleiben immer noch viele Argumente übrig, die es aus unserer Sicht lohnend machen, sich mit Komponenten zu beschäftigen. Zudem zeigen Komponentenansätze wie DCOM oder Visual Basis mit seinen VBXs, dass vorgefertigte Softwarekomponenten durchaus einsetzbar sind.

Im Folgenden werden wir darstellen, wie wir den Begriff Komponente und die Einsatzmöglichkeiten des damit verbundenen Konzepts im Kontext unseres Frameworks JWAM™ interpretieren. Mit JWAM entwickeln wir objektorientierte interaktive Anwendungssysteme für eine große Bandbreite von Kunden aus unterschiedlichen Anwendungsbereichen in Java. Wir entwickeln JWAM, damit wir für unterschiedliche Kunden JWAM-Komponenten schnell und mit hoher Qualität an Kundenbedürfnisse anpassen können (Gryczan et al. 1999).

2 Hintergrund: Der Ansatz WAM und das JWAM-Framework

Der Werkzeug- und Material-Ansatz (WAM-Ansatz; Züllighoven 1998) zur Entwicklung objektorientierter Anwendungssysteme ist die konzeptionelle und methodische Basis für die Entwicklung des Frameworks JWAM. Nach dem WAM-Ansatz entwickelte objektorientierte Systeme zeichnen sich dadurch aus, dass für Anwender Arbeitsmittel und Arbeitsgegenstände, d.h. Werkzeuge und Materialien bereitgestellt werden. Dieses einfach nachvollziehbare *Benutzungsmodell* für Anwender hat für Entwickler dann hohe Bedeutung, wenn der Anwendungsgegenstand der Entwickler, Software, leicht auf eine konstruktive Basis gestellt werden kann. Mit JWAM haben wir unter Verwendung der Programmiersprache Java™ dieses Mittel geschaffen. Gleichzeitig und zusätzlich haben wir mit JWAM unsere Erfahrungen bei der Begleitung und Durchführung von Anwendungsprojekten derart vergegenständlicht, dass wir auf eine Bandbreite von Kundenanforderungen bei der Entwicklung objektorientierter Anwendungssysteme gut reagieren können.

In diesem Sinne ist JWAM als Rahmenwerk nicht mehr und nicht weniger als ein Mittel zum Zweck: Wir, bzw. die Entwicklungsabteilungen unserer Kunden setzen JWAM als Entwicklungsplattform ein, um Lösungen für unsere Kunden marktgerecht herstellen zu können.

Unsere Diskussion über die Weiterentwicklung von JWAM ist ebenfalls pragmatisch geprägt. Die zentrale Fragestellung lautet hier: Wie muss JWAM weiter entwickelt werden, so dass wir eine immer breitere Klasse von Applikationen durch Einsatz vorfabrizierten Halbzeugs unterstützen können?

Einen wichtigen Beitrag zur Beantwortung dieser Frage liefert die aktuelle Diskussion über Komponenten. Wir greifen diese Diskussion auf, um die weitere Entwicklung von JWAM voran zu treiben.

3 Der Begriff Komponente

In der Literatur werden unterschiedliche Komponentenbegriffe diskutiert. Eine bekannte Definition hat Clemens Szyperski vorgelegt (Szyperski 1998):

> Komponenten sind in Binärform vorliegende Lösungen softwaretechnischer Probleme, die so zusammengefügt werden können, dass sie ein lauffähiges Softwaresystem bilden. Sie können unabhängig voneinander hergestellt, erworben und eingesetzt werden. Dazu besitzen Komponenten eine feste Schnittstelle und enthalten potentiell auch eigene Ressourcen, auf die sie bei

der Ausführung zurückgreifen.

Doch das Spektrum an Vorschlägen für den Komponentenbegriff ist weiter. Eine grobe Einteilung ergibt:

- *Entwurfskomponenten* liegen als technische oder fachliche Lösung im Quelltext vor und sind zur Wiederverwendung in einem ähnlichen Kontext geeignet.
- *Implementationskomponenten* liegen als Lösung in Binärform vor; sind unabhängig voneinander hergestellt und können er werden. Sie werden statisch ins Compilat eingebunden.
- *Laufzeitkomponenten* liegen als Lösung in Binärform vor; sind unabhängig voneinander hergestellt und können er werden. Sie werden zur Laufzeit hinzugeladen.

In der jüngsten Zeit läuft die Diskussion um den Komponentenbegriff auf folgende Merkmale hinaus:

1. Komponenten sind Bausteine für die Wiederverwendung.
2. Sie haben eine oder mehrere explizite Schnittstellen, über die sie fachlich kohärente Dienstleistungen anbieten.
3. Sie weisen über eine explizite Schnittstelle aus, welche Dienstleistungen sie von anderen Komponenten oder ihrer einbettenden Umgebung benötigen.
4. Sie kapseln ihre innere Struktur und ihre Implementationen.
5. Die unterschiedlichen Versionen und Varianten einer Komponente können „spät“, d.h. beim Installieren, Laden oder während der Ausführung des Anwendungssystems ausgewählt werden.

Für unseren Ansatz steht die Anwendungsorientierung im Vordergrund. Entsprechend verstehen wir unter einer Komponente allgemein:

Komponente

Eine Komponente repräsentiert die softwaretechnische Lösung eines technischen oder anwendungsfachlichen Problems. Eine Komponente ist für den Einsatz in einem Anwendungssystem mit einer fachlichen Schnittstelle ausgestattet, über die sie ihre Dienstleistungen anbietet. Komponenten können andere Komponenten voraussetzen und müssen dies durch definierte Schnittstellen explizit machen. Komponenten können austauschbar in die Schnittstellen eines Framework eingepasst werden.

Wir betrachten Komponenten als grobgranulare Einheiten oder Module, die mehrere Schnittstellen und Klassen umfassen können.

4 Aktuelle Bedeutung von Komponenten

Nach unser Erfahrung aus vielen Projekten kommen Komponenten meist in zwei Größen vor: Sehr klein und sehr groß, also etwa visuelle Komponenten zur Anzeige und Auswahl des Datums in Form eines Kalenderblatts oder als komplettes ERP-System wie SAP R/3.

Viele Komponentenmodelle bauen auf die Entstehung eines Komponentenmarktes, aber nur für sehr wenige Domänen existieren solche Märkte. Der wohl erfolgreichste Komponentenmarkt sind die grafischen Oberflächenelemente für GUIs. Die verwendeten Komponententechnologien sind Microsofts VBX (bzw. OCX) oder Suns JavaBeans.

Auf diesen Komponentenmodellen werden immer wieder Entwicklungsumgebungen gebaut, die ein grafisches Zusammenstellen der Komponenten über ihre direkte Manipulation erlauben. Dies finden wir sowohl für Visual Basic als auch für die Java Beans. Ereignisausgänge einer Komponenten können mit Ereigniseingängen einer anderen Komponente grafisch verknüpft werden. Nach unserer Erfahrung ist dies aber selbst beim Prototyping nur für sehr kleine Anwendungen zu gebrauchen, da die Entwickler sehr schnell den Überblick verlieren.

Wir ziehen daraus den Schluss, dass Komponenten besser über ausprogrammierten Glue-Code miteinander verbunden werden.

Mit den Enterprise JavaBeans steht ein Komponentenmodell zur Verfügung, dass zwar auf Binärkompatibilität setzt, aber weniger mit Blick auf einen allgemeinen Komponentenmarkt entwickelt wurde. Die sog. EJBs werden vorrangig zum Zwecke des Deployments von großen Softwaresystemen und der Verteilung im Netz verwendet. Ein System soll in anwendungsfachliche Komponenten ohne grafische Repräsentation und repräsentationsspezifische Teile zerlegt werden. Für die Repräsentation werden meist JavaServlets oder Java Server Pages verwendet.

Wir sehen in Komponenten als Entwurfs- und Konstruktionseinheiten für große verteilte Systeme den aktuellen und tragfähigen Trend. Daneben haben die generischen GUI-Komponenten ihren Markt. Einen Markt für domänenspezifische Anwendungskomponenten können wir derzeit nicht erkennen.

5 Komponenten und Frameworks

Komponenten- und Frameworkansätze scheinen auf den ersten Blick die gleichen Ziele zu verfolgen, nämlich Wiederverwendbarkeit in größerem Stil und bessere Strukturierung von Anwendungssystemen. Technisch unterscheiden sich die Ansätze. Beim Einsatz von Frameworks wird programmiert. Vorgefertigte Klassen werden für die eigene Anwendungsentwicklung benutzt oder beerbt. Beim Einsatz von Komponenten steht eher die Idee der Komposition im Vordergrund. Vererbung wird gar nicht verwendet und die Komponenten werden als Black-Boxes betrachtet. Programmiert wird idealerweise nur minimal, um den notwendigen Glue-Code zur Verbindung der Komponenten herzustellen. In diese Richtung können Frameworks aber auch gehen. Andererseits sind Komponenten, die als Entwurfskomponenten gelten, oder die für verteilte Architekturen z.B. als EJBs verwendet werden auch als Bausätze eines White-Box-Frameworks zu verstehen.

Wir unterscheiden weiterhin zwischen Komponenten und Frameworks:

- *Frameworks* bestimmen die Anwendungsarchitektur im Großen und stellen die wesentlichen Grundkonzepte dieser Architektur abstrakt bereit.
- *Komponenten* hingegen stellen zusätzliche und abgegrenzte technische bzw. anwendungsfachliche (Teil-) Lösungen zur Verfügung, unter denen der Anwendungsentwickler auswählen kann.

Komponenten- und Framework-Technologie ergänzen sich aus unserer Sicht sehr gut (siehe den nächsten Abschnitt). Wenn wir als integrierenden Rahmen für eine Familie von Komponenten dasselbe Framework voraussetzen, können die einzelnen Komponenten auf einer einheitlichen technischen Infrastruktur der verwendeten Programmiersprache und des verwendeten Komponentenmodells aufbauen. Dazu kommen die Basiskonzepte und Abstraktionen des zugrundeliegenden Frameworks, wie etwa ein einheitlicher Benachrichtigungsmechanismus, eine Interpretation des Vertragsmodells oder die durchgehend einheitliche Verwendung von Wert- und Referenzsemantik.

Damit lassen sich fachlich höhere Schnittstellen definieren, als dies beispielsweise bei den Java Beans der Fall ist. So können die Komponenten eines Frameworks für den WAM-Ansatz bereits die Konzepte Material und Werkzeug kennen.

6 Komponenten in JWAM

Das JWAM-Framework besteht aus einem Kern, der die Basiskonzepte wie Werkzeug, Material, Fachwerte des Werkzeug und Material-Ansatzes realisiert und darauf aufbauenden Komponenten. Wir versuchen den Kern so klein und schlank wie möglich zu halten. Daher haben wir zusätzliche nicht in jedem Anwendungsprojekt benötigte anwendungsfachliche Funktionalität in Komponenten ausgelagert.

Wir unterscheiden dabei zwei Arten von Komponenten: Ausstattungs- und Einschubkomponenten.

6.1 Ausstattungskomponenten

> *Definition*: Eine Ausstattungskomponente löst als Entwurfskomponente ein anwendungsfachliches Problem. Sie kann in ein Anwendungssystem zur Entwicklungszeit eingebunden werden und bietet ihre Dienstleistungen mittels einer fachlichen Schnittstelle an. Eine Ausstattungskomponente macht explizit, von welchen anderen Komponenten sie ggf. abhängt. JWAM-Ausstattungskomponenten setzen den JWAM-Kern implizit voraus.

Ausstattungskomponenten bieten Anwendungsentwicklern für Entwurf und Implementierungen vielfältige Entscheidungsmöglichkeiten. Der Anwendungsentwickler kann in einem konkreten Projekt entscheiden, ob er sein System mit einer bestimmten Komponente, beispielsweise den Desktop oder dem Formularwesen, ausstatten will. Ausstattungskomponenten sind in sofern optional, als sie rein additiv zu einem bestehenden Framework-Kern hinzugenommen werden können.

In JWAM visualisiert die Komponente Desktop das Konzept der (Arbeits-)Umgebung nachvollziehbar für den Anwender. Die Komponente wird als Black-Box in das System eingebunden, aufgerufen und präsentiert sich dem Anwender. Die auf dem Desktop initial liegenden Gegenstände können vom Anwendungsprogrammierer bestimmt werden.

Das Formularwesen ist in zwei verschiedene Komponenten unterteilt: Eine Komponente stellt die Grundkonzepte beispielsweise als Formularelement und Formular zur Verfügung. Eine weitere Komponente bietet generische Werkzeuge zum Editieren und Anzeigen von Formularen für Anwender. Beide Konzepte werden in der Regel im Sinne der White-Box-Vererbung verwendet und

stellen somit Entwurfskomponenten dar. Sie geben eine fertige Konstruktion für ein festgelegtes Problemfeld vor, welches gut zu den Konzepten und Leitbildern des Rahmenwerkskerns passt.

Ausstattungskomponenten verallgemeinern meist Konstruktionen, die bereits in unterschiedlichen Anwendungsentwicklungskontexten verwendet wurden. Sie vergegenständlichen eine anwendungsfachliches Konzept für die direkte Wiederverwendung .

6.2 Einschubkomponenten

Die Funktionalität von Ausstattungskomponenten kann oft mit unterschiedlichen Technologien implementiert werden. Dabei hat es sich als nützlich erwiesen, wenn diese alternativen Technologien auf einfache Art und Weise vom Anwendungsentwickler ohne großen Programmieraufwand ausgetauscht werden können. Untragbar wäre, für jede neue Technologie die gesamte Ausstattungskomponente neu zu implementieren. Daher führen wir das Konzept der Einschubkomponenten ein.

> *Definition*: Einschubkomponenten sind gekapselte Implementierungen für eine Einschub-Schnittstelle, meist als Teil einer Ausstattungskomponente. Einschubstellen werden verwendet, um von der konkreten verwendeten Technologie zu abstrahieren. Einschubkomponenten können zur Entwicklungszeit oder zur Laufzeit ausgetauscht werden. Einschubkomponenten machen explizit, welche anderen Komponenten sie voraussetzen.
>
> JWAM-Einschubkomponenten setzen den JWAM-Kern implizit voraus.

Einschubkomponenten werden aus einer anderen Motivation erstellt als Ausstattungskomponenten. Im Vordergrund steht bei ihnen die Skalierbarkeit. Wir verwenden Einschubkomponenten in Rahmenwerken und Anwendungssystemen dann, wenn wir uns explizit auf eine Schnittstelle abstützen, uns aber unterschiedliche Implementationen offen halten wollen. Damit können wir je nach Einsatzsituation verschieden aufwendige Technologien im Sinne der Skalierung austauschen.

Einen weiteren Vorteil zeigen Einschubkomponenten innerhalb eines zyklischen Entwicklungsprozesses. Zu Beginn eines Projektes können wir nach der Definition der Schnittstelle einer Einschubkomponente zunächst eine sehr primitive Implementierung wählen und das System fachlich weiterentwickeln. Später, also etwa während der Konstruktionsetappe, kümmern wir uns dann um eine effizientere Implementation.

Die Technik der Einschubkomponenten bietet uns generell die Möglichkeit, technische Entscheidungen wegzukapseln und z.B. eine Entscheidung für ein konkretes Datenbankprodukt aufzuschieben.

Generell passt das Konzept der Einschubkomponenten sehr gut zu den Ausstattungskomponenten. Ausstattungskomponenten liefern für den Anwendungsentwickler die fachlichen Schnittstellen und die anwendungsorientierten Gegenstände, mit denen er einen Arbeitsplatz versehen will. Die Einschubkomponenten liefern eine unabhängige technologische Dimension für diese Ausstattungskomponenten. Dies bedeutet, dass ein Anwendungsentwickler die fachliche Entscheidung für eine Ausstattungskomponente weitgehend trennen kann, von der Festlegung auf eine bestimmte Technologie. Damit skaliert das System. So kann es bei der Einführung eines Arbeits-

platzsystems sinnvoll sein, die Registratur mit den Kundendaten auf das File System eines Anwendungsservers aufzusetzen. Wenn die Anwenderzahl und der Kundenstamm stärker anwachsen, lässt sich die Einschubkomponente „Persistenz auf File-Basis“ austauschen durch die Einschubkomponente „Persistenz durch RDBMS“, ohne dass die Ausstattungskomponenten Registratur nach Außen verändert werden muss.

Im aktuellen JWAM-Rahmenwerk finden wir keine solchen "puren" Einschubkomponenten. Zu der Ausstattungskomponente Registratur, die einen fachlichen Persistenzdienst darstellt, gibt es eine definierte Einschubstelle „Registraturbehälter“. Diese definiert die Schnittstelle (im Sinne eines Outgoing-Interface), die eine konkrete Implementation erfüllen muss, um Materialien persistent machen zu können.

Unsere erste einfache Implementation für diese Einschubkomponente basierte auf dem Filesystem und der JAVA-Serialisierung. Mittlerweile existieren auch Einschübe, die auf JDBC oder auf konkreten Datenbankprodukten basieren (Beispielcode kann unter http://www.jwam.de bezogen werden.)

7 Diskussion

Die Aufteilung unseres Rahmenwerks JWAM in einen Kern und darauf aufbauende Komponenten (Design-, Implementations- oder Laufzeitkomponenten) bringt viele Vorteile:

Der Kern wird schlanker und verständlicher. Er ist dadurch für Anwendungsentwickler leichter einzusetzen und für Framework-Entwickler einfacher zu warten.

Die Komponenten um den Kern herum sind nur über die Konzepte des Kerns miteinander verbunden. Sie können unabhängig voneinander erstellt, gewartet und benutzt werden. Dadurch realisieren wir eine Skalierbarkeit im Prozess - nur die Dinge werden hinzugenommen, die aktuell benötigt werden.

Das Prinzip von Ausstattungs- und Einschubkomponenten trennt fachliche und technologische Dimension. Da Einschubkomponenten von Anwendungsprogrammierern ohne Programmieraufwand ausgetauscht werden können, entspricht dies den Ideen der Black-Box-Frameworks. Nicht zuletzt können wir durch die Aufteilung von Rahmenwerkskern, Austattungs- und Einschubkomponenten unseren Kunden jeweils maßgeschneiderte Lösungsbestandteile anbieten: Nur die Komponenten, die für die Lösung eines kundenspezifischen Problems gebraucht werden, müssen eingesetzt, bzw. lizensiert werden.

Literatur

Szyperski, C.: Component Software: Beyond Object-Oriented Programming. 2. Aufl., Addison-Wesley, Harlow 1998.

Gryczan, G.; Lilienthal, C.; Lippert, M.; Roock, S.; Wolf, H.; Züllighoven, H.: Frameworkbasierte Anwendungsentwicklung (Teil 1). ObjektSpektrum (1999) 1, S. 90-99.

Züllighoven, H.: Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz. dpunkt, Heidelberg 1998.