

UWE – Ein Ansatz zur modellgetriebenen Entwicklung von Webanwendungen (UWE – An Approach for the Model-Driven Development of Web Applications)

Alexander Knapp, Nora Koch, Martin Wirsing, Gefei Zhang

Angaben zur Veröffentlichung / Publication details:

Knapp, Alexander, Nora Koch, Martin Wirsing, and Gefei Zhang. 2007. "UWE – Ein Ansatz zur modellgetriebenen Entwicklung von Webanwendungen (UWE – An Approach for the Model-Driven Development of Web Applications)." *i-com* 6 (3): 5–12.
<https://doi.org/10.1524/icom.2007.6.3.5>.

Nutzungsbedingungen / Terms of use:

licgercopyright

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:

Deutsches Urheberrecht

Weitere Informationen finden Sie unter: / For more information see:

<https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publiz/>



Alexander Knapp, Nora Koch, Martin Wirsing und Gefei Zhang

UWE – Ein Ansatz zur modellgetriebenen Entwicklung von Webanwendungen

UWE – An Approach for the Model-Driven Development of Web Applications

Webanwendungen_MDD_Modellierung_Modelltransformationen_UML-Profil

Zusammenfassung. UWE (UML-based Web Engineering) ist ein Ansatz zur modellgetriebenen Entwicklung von Websystemen. Die UWE-Notation ist speziell an die intuitive Modellierung von Webanwendungen angepasst; UWE umfasst einen werkzeugunterstützten Entwicklungsprozess für die semi-automatische Konstruktion von Webanwendungen; und UWE-Modelle ermöglichen eine frühzeitige Validierung. Eines der Hauptcharakteristika von UWE ist seine Verwendung von etablierten Standards: Die Modellierungssprache basiert auf UML und einer UML-Erweiterung, einem so genannten UML-Profil. Dieses Profil ist als eine konservative Erweiterung des UML-Metamodells definiert, wodurch die Verwendung bestehender Modellierungswerkzeuge und deren Erweiterungen ermöglicht wird. Der UWE-Prozess wendet die Model-Driven-Architecture (MDA)-Prinzipien auf die Entwicklung von Webanwendungen an, Modelle und Modelltransformationen bilden den primären Fokus.

Summary. The UML-based Web Engineering (UWE) approach supports the model-driven development of Web systems. The UWE notation is tailored for an intuitive modelling of Web applications. UWE comprises a development process and tool support for semi-automatic construction of Web applications. UWE models make early validation possible. One of the distinguishing features of UWE is its compliance with standards. Modelling with UWE is based on UML and a UML extension – a so-called UML profile. The UWE profile is defined as a conservative extension of the UML metamodel. Metamodelling and conformance with standards enable the use of existing tools or the development of plug-ins for tools already in use. The UWE process applies the Model Driven Architecture (MDA) principles to the development of Web applications focusing on models and model transformations.

1. Einleitung

Die Software-Engineering-Forschung ist geprägt von der Suche nach Abstraktionen, die dem Entwickler helfen, seine Probleme nicht in einem technischen Lösungsraum, sondern in einem domänen-spezifischen Problemraum zu beschreiben und zu lösen. Dies gilt für den Entwurf höherer Programmiersprachen und für die Einführung von CASE-Werkzeugen ebenso wie für den neueren Ansatz der so genannten modellgetriebenen Entwicklung, dem „Model-Driven Engineering“ (MDE).

MDE zielt darauf ab, den Abstraktionsgrad durch die Verwendung domä-

nenspezifischer Modellierungssprachen und automatischer Codegenerierung durch Modelltransformationen zu erhöhen. Modelle dienen der anwendungsreichabhängigen und sichtenspezifischen Repräsentation von Softwaresystemen und damit dem besseren Verständnis der Probleme und ihrer Lösungsmöglichkeiten. Die Verfeinerung und Integration verschiedener Sichten und Modelle auf dasselbe System sowie schließlich die Erzeugung lauffähiger Codes findet durch möglichst weitgehend automatisierte Modelltransformationen statt. Die Softwareentwicklung wird aus Sicht des MDE somit als Folge von Modelltransformationsanwendungen gesehen, in der technologieabhängige Entscheidungen erst, soweit als möglich, am Ende zu tref-

fen sind. Die wohl bekannteste Umsetzung der MDE-Idee ist der MDA-Ansatz der Object Management Group („Model-Driven Architecture“, www.omg.org/mda). MDA integriert viele etablierte OMG-Standards, insbesondere die „Unified Modeling Language“ (UML) für die Modellierung.

Web-Engineering ist eine Teildisziplin des Software-Engineering, die sich mit der systematischen Entwicklung von Webanwendungen befasst. Zu den klassischen Problemen der allgemeinen Softwareentwicklung, wie der Einhaltung eines Budget-, eines Zeit- und eines Qualitätsrahmens, treten die webanwendungsspezifischen Aspekte sich sehr rasch ändernder, kontinuierlich umzusetzender Kundenwünsche und hoher Tech-

nologievolatilität hinzu. Die von MDE propagierte Trennung zwischen der technologieunabhängigen Modellierung und der Transformation in plattformspezifische Modelle einerseits und das Abstraktionsniveau und damit die Änderbarkeit der in einem MDE-Ansatz verwendeten Modelle selbst andererseits, machen MDE zu einer Erfolg versprechenden Methode für den Bereich des Web-Engineering.

Im Laufe der letzten Jahre wurde eine Reihe von Sprachen, Architekturen, Methoden und Prozessen für die Entwicklung von Webanwendungen vorgeschlagen, etwa Hera, OOHDM, OO-H, OOWS, UWE, WebML oder W2000 (für einen Überblick siehe Rossi et al. 2007). Dabei standen die Modellierung von Websystemen, zum Beispiel der Navigation oder der Adaptivität, sowie (eine meist plattformspezifische) Codegenerierung im Vordergrund. In Anerkennung der MDE-Prinzipien und ihrer Nützlichkeit für die Entwicklung von Webanwendungen besteht ein allgemeiner Trend, auch weitergehende Modelltransformationen in die verschiedenen Ansätze zu integrieren.

Der UWE-Ansatz war, teilweise bedingt durch seine Verwendung der UML, einer der Vorreiter dieses Trends. Im Folgenden wollen wir UWE, seine Notationen und Techniken und insbesondere die Umsetzung von MDE-Ideen erläutern.

2. Der UWE-Ansatz

Der UWE-Ansatz (www.pst.ifi.lmu.de/projekte/uwe) entstand Ende der 90er-Jahre mit dem Ziel, die UML-basierte Entwicklung (UWE ist ein Akronym für „UML-based Web Engineering“) von Webanwendungen zu unterstützen und, im Gegensatz zu allen anderen zu diesem Zeitpunkt existierenden Methoden im Webbereich, auf proprietäre Notationen zu verzichten. UWE definiert eine UML-Erweiterung in Form eines so genannten UML-Profiles unter ausschließlicher Verwendung der Erweiterungsmechanismen, die UML selbst anbietet: Stereotypen und Wohlgeformtheitsbedingungen. Das UWE-UML-Profil dient als domänen-spezifische Modellierungssprache für Webanwendungen.

UWE ist aber nicht nur eine Notation zur Modellierung, sondern vielmehr ein Prozess zur Erstellung von Webanwendungen. Der UWE-Prozess folgt den Prin-

zipien des MDA-Standards der OMG, d.h., der Prozess ist modellgetrieben und stützt sich auf Modelle und Modelltransformationen. Das UWE-Metamodell ist „Meta-Object-Facility“ (MOF)-konform, UWE-Modelle können über XML ausgetauscht werden (in der Hoffnung, dass die Werkzeuginteroperabilität in näherer Zukunft verbessert wird) und die UWE-Modelltransformationen sind in Sprachen wie etwa der zukünftigen Standard-Modelltransformationssprache „Query/View/Transformation“ (QVT) formuliert. Werkzeuge zur Modellierung und Transformation von Webanwendungen – eine wesentliche Voraussetzung für die Verwendbarkeit eines Prozesses wie UWE in der Praxis – liegen vor; eine integrierte Entwicklungsumgebung ist in Bearbeitung.

Bei der Modellierung von Webanwendungen hat es sich als nützlich erwiesen, neben den Dimensionen der Entwicklungsphasen (Analyse, Entwurf, Implementierung) und -aspekte (Struktur und Verhalten) eine weitere Dimension, die der Ebenen des Inhalts, der Navigation und der Präsentation, zu unterscheiden. So wie die meisten gängigen Web-Modellierungsansätze berücksichtigt auch UWE diese Trennung und fügt darüber hinaus eine vierte Dimension, die orthogonal zu allen drei erwähnten Dimensionen (Phasen, Aspekte und Ebenen) verläuft, hinzu: Adaptivität. Mit Adaptivität ist hier die dynamische Anpassung der Webanwendung an Benutzerverhalten und Kontextinformation gemeint, ein Thema das immer mehr an Bedeutung gewinnt. Ein Anforderungsmodell stellt den Ausgangspunkt der UWE-Modellierung dar. Darauf aufbauend erfolgt der Entwurf der Webanwendung, der oben genannten Trennung gemäß, in Inhalts-, Navigations-, Präsentations-, Prozess- und Adaptivitätsmodellen, die im folgenden Abschnitt beschrieben werden (für einen Überblick des Entwicklungsprozesses siehe auch Abschnitt 5).

3. Modellierung von Webanwendungen mit UWE

Wir erläutern die Modellierung mit UWE sowohl anhand der Anforderungen an ein Websystem, als auch im Entwurf anhand eines einfachen Musikportal-Beispiels. Das Musikportal bietet im Internet Musikalben und Lieder zum Abruf an.

Webbenutzer können Alben suchen und, wenn sie angemeldet sind, ein komplettes Album oder eine Auswahl von Liedern kaufen. Vor der Anmeldung müssen Benutzer sich beim Websystem registriert haben. Jedes Album und jedes Lied hat seinen Preis; ein Abruf von Musikeinheiten ist nur möglich, wenn der registrierte Webbenutzer genügend Guthaben auf seinem Konto hat; ein Konto kann jederzeit aufgeladen werden.

3.1 Modellierung von Anforderungen

Anforderungen an eine Webanwendung können mit all den unterschiedlichen Techniken ermittelt, analysiert, beschrieben und überprüft werden, die sich in der Entwicklung konventioneller Software bewährt haben. Für die Beschreibung von funktionalen Anforderungen verwendet UWE deshalb die wohlbekannten UML-Anwendungsfälle und Anwendungsfalldiagramme, die eine Übersicht der Anforderungen an das Websystem aus Sicht der Akteure (Personen ebenso wie anderer Systeme) bieten. Eine Besonderheit der Anforderungsanalyse bei Webanwendungen ist die Berücksichtigung der Navigationsfunktionalität, die dem Benutzer das Navigieren durch den Hypertext sowie das Auffinden von Knoten ermöglicht. Der UWE-Ansatz unterscheidet deshalb zwischen funktionalen und hypertext-spezifischen Anwendungsfällen. Hierfür wird jeder so genannte Navigations-Anwendungsfall

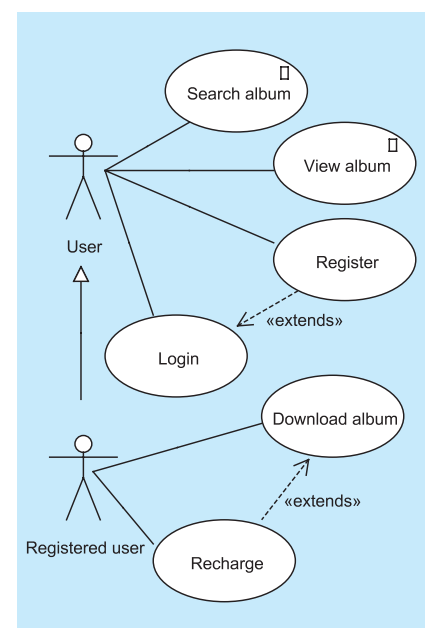


Bild 1: Anforderungsmodell eines Musikportals

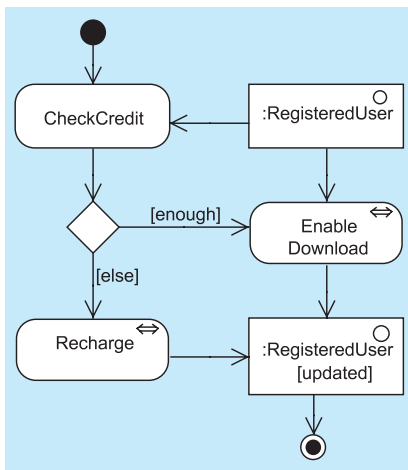


Bild 2: Detaillierte Anforderungen für Anwendungsfall *Download album*








durch das Piktogramm  für den UML-Stereotyp «navigation» gekennzeichnet.

Bild 1 stellt das Anwendungsfalldiagramm unseres Musikportals dar, in dem die Anwendungsfälle *SearchAlbum* und *ViewAlbum* lediglich Navigationsfunktionalität beinhalten. Im Beispiel des Musikportals, wie bei sehr vielen anderen typischen, interaktiven Webanwendungen, können wir zwei Akteursarten unterscheiden: den anonymen und den registrierten Benutzer.

Für eine detailliertere Beschreibung der nicht-trivialen Anwendungsfälle, die mehr Funktionalität bieten als reine Navigation, empfiehlt UWE, die Anforderungsmodellierung um Aktivitätsdiagramme zu ergänzen. UML-Aktivitätsdiagramme sind bestens geeignet, um den Ablauf der Geschäftsprozesse einer Webanwendung, wie etwa des Kaufs eines Musikalbums, intuitiv zu veranschaulichen. Aktivitätsdiagramme bestehen aus Aktionen, Verantwortlichkeiten für die Aktionen, Kontrollflusselementen sowie Objektflüssen. Auch hier können UML-Modellelemente mit von UWE definierten webspezifischen Stereotypen angereichert werden. Diese zusätzliche Information kommt dem modellgetriebenen Prozess, insbesondere der Definition der Modelltransformationen zu gute. In Bild 2 ist beispielhaft ein Aktivitätsdiagramm für den Anwendungsfall *Download album* dargestellt. UWE unterscheidet zwischen Objekten, die Inhalts-, Knoten- oder Präsentationselemente der Webanwendung definieren. Diese Unterscheidung wird mittels der Stereotypen

«content», «node» und Web-Benutzerschnittstelle («WebUI») oder der entsprechenden Piktogramme , ,  visualisiert (siehe Bild 2). UWE verwendet darüber hinaus Stereotypen für spezifische Aktionen im Webumfeld, wie Blättern () Suchen () und Transaktionen () die Datenbankänderungen erfordern.

3.2 Spezifikation des Inhalts

Die von einer Webanwendung zu verwaltenden Daten werden durch die Angabe von UML-Klassendiagrammen visuell spezifiziert: Klassen beschreiben die verfügbaren Entitäten, ihre Attribute und Operationen; Assoziationen die Beziehungen zwischen diesen Entitäten. Es ist dabei bewährte Praxis, zwischen den eigentlichen Inhalten, die sich an den Benutzer richten, in Form eines Inhaltsmodells und den Daten über den Benutzer in Form eines Benutzermodells zu unterscheiden; insbesondere vereinfacht die getrennte Benutzermodellierung die Einbeziehung von Adaptationsmöglichkeiten, um auf Benutzerverhalten zu reagieren (s. Abschnitt 4). Für die Erstellung des Inhalts- und des Benutzermodells geben die detaillierten Anforderungen und die darin verwendeten Objekte nützliche Hinweise.

In unserem einfachen Beispiel eines Musikportals sehen wir für den Inhalt einerseits Lieder (Klasse *Song*), die auf Alben (Klasse *Album*) enthalten sind, vor; andererseits sollen einem Musikportalsneuling die Möglichkeiten durch eine vorgefertigte Präsentation (Klasse *Slide*) nahegebracht werden. Im Benutzermodell wird der Benutzer (Klasse *RegisteredUser*) durch seinen Namen, sein Passwort und

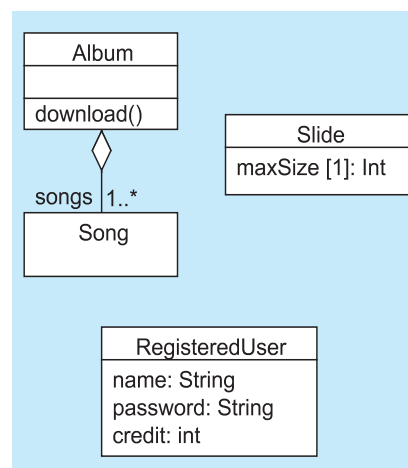








Bild 3: UML Klassendiagramm für das Inhalts- und das Benutzermodell des Musikportals

sein Guthaben auf dem Konto (Attribut *credit*) repräsentiert.

3.3 Modellierung der Navigationsstruktur

Aufbauend auf dem Inhaltsmodell kann nun eine Navigationsstruktur als eine abstrakte Darstellung der Navigationsmöglichkeiten des Benutzers durch die Webanwendung erstellt werden. In einem UWE-Navigationsmodell, das auf UML-Klassendiagrammen basiert, repräsentieren Navigationsknoten die verfügbaren Informationen, die einem Benutzer, der diesen Knoten erreicht, zur Verfügung stehen, und Navigationsverbindungen die Möglichkeiten, von einem Knoten aus weiter zu gehen.

Im Musikportalsbeispiel kann der Benutzer sich von der Einstiegsseite (*Home*) registrieren und einloggen, die Benutzerführung wählen, seinen Kontostand aufbessern oder ein Album suchen. Bei der Suche wird eine Liste passender Alben erstellt, aus der ein Album ausgewählt werden kann. Dieses Album kann dann direkt abgerufen werden, oder der Benutzer kann aus einer Liste der auf diesem Album enthaltenen Lieder eines zum Abruf auswählen. Eine entsprechende UWE-Navigationsstruktur ist in Bild 4 dargestellt. Für eine Auswahl zwischen verschiedenen Navigationsknoten wird von UWE das Konstrukt *Menu* () angeboten. Der Navigationspfad der Suche eines Albums und der Wahl eines der gefundenen Alben wird in UWE folgendermaßen umgesetzt: Eine *Query* () wird an einen *Index* () gekoppelt und dieser an eine Navigationsklasse (, hier *Album*), die den eigentlichen Inhalt aus dem Inhaltsmodell repräsentiert. Die Benutzerführung wird durch eine so genannte *Guided tour* () realisiert, die durch die im Inhaltsmodell spezifizierte Präsentation leitet. Die Registrierung, die Anmeldung, das Aufladen des Kontos sowie die Abrufmöglichkeiten von Alben und Liedern sind durch Prozesse zu implementieren, deren Einstiegspunkte durch Prozessklassen (, *Register*, *Login*, *Recharge*, *DownloadAlbum*, *DownloadSong*) repräsentiert sind.

Tatsächlich bietet UWE methodische Hilfestellungen für die Entwicklung eines Navigationsmodells aus einem Inhaltsmodell in Form eines semi-automatischen Transformationsprozesses an: Die für die Navigation als relevant erachteten Klas-

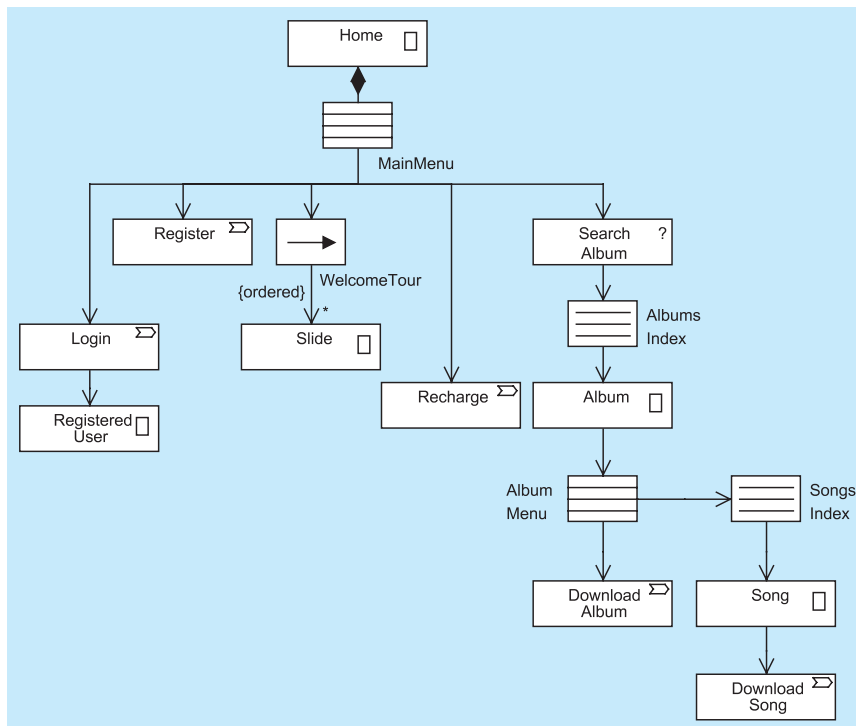


Bild 4: Navigationsmodell des Musikportals

sen und Assoziationen des Inhaltsmodells werden in Navigationsklassen und -verbindungen übersetzt. Da Navigationsverbindungen die Schritte eines Benutzers durch die Webanwendung repräsentieren, müssen alle diese Verbindungen gerichtet sein und dementsprechend werden bidirektionale Assoziationen in zwei, vorwärts und rückwärts gerichtete, Navigationsverbindungen transformiert. Menüs, die eine Auswahl darstellen, werden zu allen Navigationsklassen mit mehr als einer ausgehenden Verbindung hinzugefügt. Schließlich sind alle Verbindungen,

die auf mehrere Instanzen einer Navigationsklasse weisen (in UML-speak: Assoziationsenden mit Multiplizität größer eins), mit einem Index zu versehen. Ein solchermaßen automatisch abgeleitetes Navigationsmodell kann anschließend manuell verfeinert werden, etwa durch Hinzufügung eines Einstiegsknotens und durch Ergänzung um Prozessklassen.

3.4 Präsentationsentwurf

Ein UWE-Präsentationsmodell gibt eine abstrakte Sicht einer Benutzerschnittstelle (UI) einer Webanwendung wieder: Es

wird von konkreten Aspekten der Benutzerschnittstelle, wie etwa den Farben, den Schriftarten und der genauen Positionierung von UI-Elementen auf der Webseite abgesehen, und lediglich die Struktur der Benutzerschnittstelle mit den verwendeten UI-Elementen, zum Beispiel Texten, Bildern, Verweisen und Formularen, modelliert. Damit bleibt das Präsentationsmodell technologieunabhängig.

Die Basiselemente eines UWE-Präsentationsmodells, das wiederum durch eine spezialisierte Form eines UML-Klassendiagramms repräsentiert wird, sind Präsentationsklassen, die den unterschiedlichen Knotentypen des UWE-Navigationsmodells (Navigationsklassen, Prozessklassen, Menüs, usw.) zugeordnet sind. Eine Präsentationsklasse (☐) besteht aus UI-Elementen wie Text (≈), Verweisen (→), Verweisgruppen (≡), Knöpfen (●), Bildern (☐) und Formularen (☐). Bild 5 zeigt eine Präsentationsklasse für den Einstiegsknoten *Home* der Beispiel-Webanwendung. Im Allgemeinen werden mehrere Navigationsknoten auf einer gemeinsamen Webseite präsentiert. Dem entspricht in UWE das Konzept der «page», die es erlaubt, Präsentationsklassen hierarchisch zu gruppieren.

3.5 Geschäftsprozess-Modellierung

Prozessknoten in einem Navigationsmodell repräsentieren die Einstiegspunkte in die Prozesse, die von der Webanwendung zur Verfügung gestellt werden. Der Ablauf, das heißt, der Kontroll- und Datenfluss eines Prozesses wird in UWE durch ein UML-Aktivitätsdiagramm, das Prozessablaufmodell, festgelegt. Eventuell im Prozess notwendige, über das Inhalts- und das Benutzermodell hinausgehende Daten, stellen wir durch ein weiteres UML-Klassendiagramm, das Prozessstrukturmodell, dar. Wie erwähnt ist das Anforderungsmodell die Grundlage für die Erstellung des Inhaltsmodells und des sich daraus ergebenden Navigationsmodells; dasselbe gilt auch für die Prozessmodellierung. Ein hinlänglich verfeinertes UWE-Prozessmodell kann dann auch im Sinne eines *Rapid-Prototyping* direkt ausgeführt werden.

Wir wollen die Prozessmodellierung anhand des in Bild 4 geforderten *Login*-Prozesses veranschaulichen, siehe Bild 6: Um festzustellen, ob bereits ein Benutzer

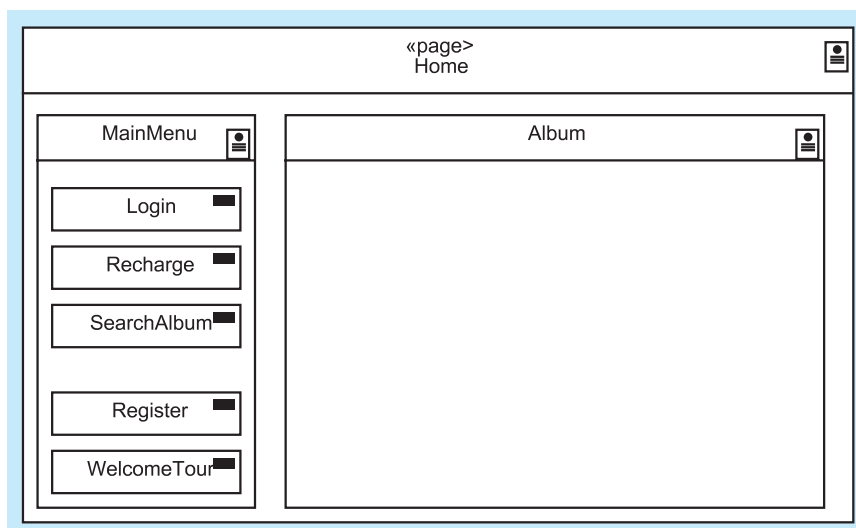


Bild 5: Präsentationsbeispiel für Home

angemeldet ist, wird eine Anmeldung in einem im Prozessstrukturmodell zu definierenden, pro Sitzung neu instanziierten *Session*-Objekt gehalten. Ist noch kein Benutzer in der aktuellen Sitzung angemeldet, so werden der Benutzername und sein Passwort erfragt, anhand derer unter Rückgriff auf das Benutzermodell festgestellt werden kann, ob der Benutzer zur Anmeldung berechtigt ist; ein Fehlschlagen der Anmeldung wird dem Benutzer durch eine Fehlermeldung mitgeteilt. Im UML-Aktivitätsdiagramm werden die entsprechenden Entscheidungen im Kontrollfluss durch Entscheidungsknoten (◇) repräsentiert; für den Datenfluss bieten sich so genannte Ein-/Ausgabe-Pins (□) an, um Parameterübergaben zwischen den verschiedenen Aktivitäten (□) zu spezifizieren.

Die Prozesse für die Benutzerregistrierung, das Abrufen eines Albums oder eines Liedes sowie das Aufladen des Guthabens sind gemäß den eingangs skizzierten Anforderungen analog zu modellieren.

4. Adaptive Webanwendungen in UWE

Adaptivität ist die Eigenschaft von Softwaresystemen, das Verhalten an Benutzereigenschaften anzupassen. Für Webanwendungen ist Adaptivität einer der zentralen Wettbewerbsfaktoren in einem Umfeld, in dem eine enge Benutzerbindung trotz im Allgemeinen a priori unbekannter Benutzer wünschenswert ist. Web-Anwendungen können je nach Benutzerdaten (wie z. B. Wissen, Zielen, Hintergrund, Präferenzen, etc.), Benutzungsdaten (wie z. B. Navigationshistorie) und Umgebungsdaten (wie z. B. Browserversion, vorhandenen Plug-Ins, aktuellem Ort, etc.) die Präsentation ihrer Inhalte und die Navigationsstruktur der Anwendung adaptieren (siehe Kappel et al. 2006).

Die Modellierung der Beobachtung von Benutzerverhalten und der Reaktionen der Webanwendung ist typischerweise mit den Modellen der Basisfunktionalitäten eng verzahnt, da an vielen Stellen Beobachtungen und Reaktionen erfolgen sollen. Um einer zu engen Verwebung solch überall auftretenden Verhaltens mit der eigentlichen Funktionalität zu entgehen und die Modelle wartbar zu halten,

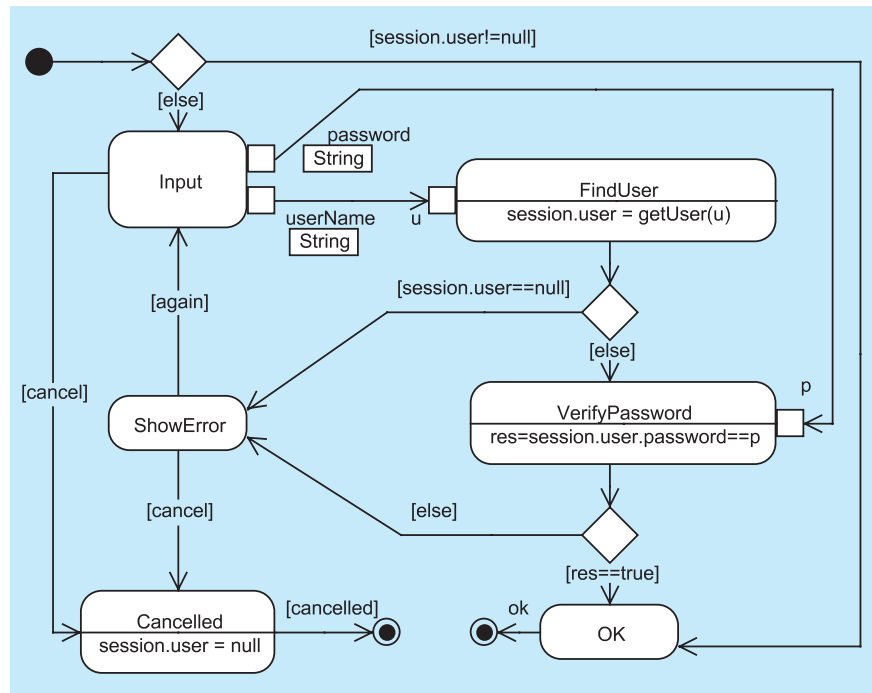


Bild 6: Login Prozess

verwendet UWE die Technik der aspektorientierten Modellierung (siehe Filman et al. 2004), die sich für querschneidende Belange etabliert hat. Wir führen ein neues Modellelement, den *Aspekt*, in die UML ein, der aus einem *Pointcut* und einem *Advice* besteht. Das Pointcut quantifiziert Elemente aus dem Basismodell über ein Muster, wobei ein ausgezeichnetes Element (markiert mit einem „?“) instanziiert wird; der Advice spezifiziert zusätzliche Eigenschaften der selektierten Elemente. Ein Aspekt bedeutet also, dass den vom Pointcut ausgewählten Elementen Eigenschaften hinzugefügt werden, die im Advice spezifiziert sind: Erst die Komposition (*Weaving*) des Basismodells mit dem Advice ergibt die kompletten Eigenschaften der selektierten Elemente. UWE definiert mehrere Typen von Aspekten, um die statische bzw. die Laufzeitadaptivität von Webanwendungen zu ermöglichen (Rossi et al. 2007).

In unserem Musikportalbeispiel können wir die Adaptivitätsanforderung, dass die Anwendung jedem (registrierten) Benutzer eine Favoriten-Liste anzeigt, in der alle Alben enthalten sind, die vom Benutzer mindestens drei Mal abgerufen wurden, wie folgt aspektorientiert modellieren:

- Durch einen (statischen) Modellaspekt (*Visited*) werden alle Klassen mit dem Namen *Album* selektiert (in einem

wohlgeformten Modell kann es nur eine solche Klasse geben) und diesen eine zusätzliche Operation *visited* hinzugefügt mit einem Rückgabewert, der angibt, wie oft ein Album schon abgerufen wurde.

- Ein (dynamischer) Aspekt *Count* stellt sicher, dass *visited* die Abrufe von Alben richtig protokolliert.
- Ein (statischer) Modellaspekt *Favorites* fügt schließlich in das Navigationsmodell eine zusätzliche Verbindung von *MainMenu* über einen Index zu *Album* ein.

Bild 7 zeigt den Aspekt *Visited*, für die graphische Darstellung der anderen zwei Aspekte verweisen wir auf unseren Beitrag in Rossi et al. 2007.

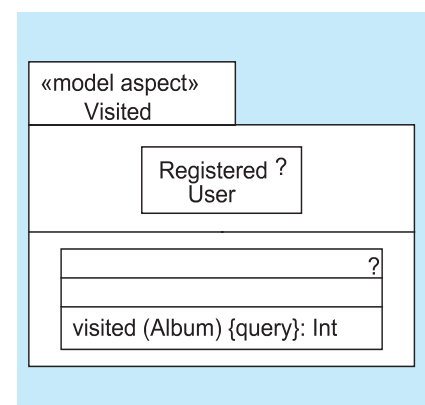


Bild 7: Modellaspekt Visited

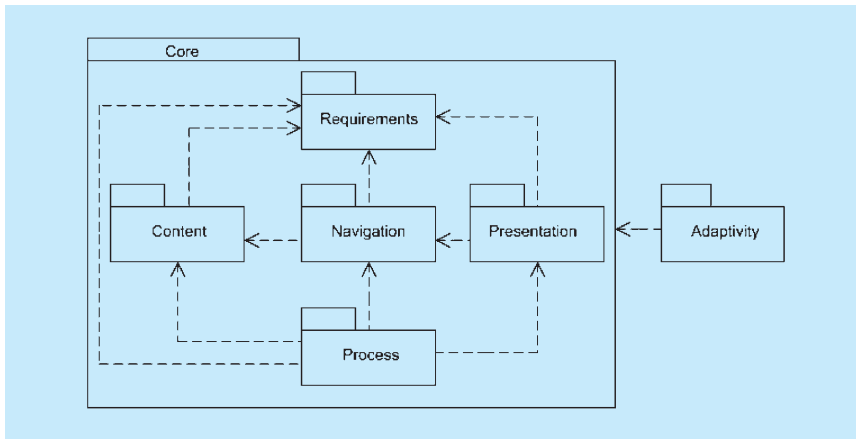


Bild 8: UWE-Metamodell

5. Modellgetriebene Entwicklung

UWE ist als modellgetriebener Ansatz konzipiert, bei dem Modelle und ihre Transformationen im Mittelpunkt stehen. Implementierungen, eventuell auf unterschiedlichen Plattformen, werden durch Transformationen in Codes erzeugt. In MDA werden dabei so genannte plattformunabhängige – und damit technologieunabhängige – Modelle und plattform-spezifische – technologieabhängige – Modelle unterschieden. Die Modellelemente, die für diese Modelle benötigt werden, und die Beziehungen zwischen den Elementen werden auch als Modell dargestellt, ein so genanntes Meta-modell. Basierend auf den MDA-Prinzipien werden für UWE Modell-zu-Modell- und Modell-zu-Text-Transformationen definiert. Modelltransformationen bauen auf Metamodellen auf, d. h. Modelltransformationen bilden Instanzen eines Quell-Metamodells auf Instanzen eines Ziel-Metamodells ab. Die nächsten Abschnitte skizzieren, welche Rolle Metamodellierung (0), Transformationspezifikationen (0) und Werkzeugunterstützung in UWE spielen.

5.1 Das UWE-Metamodell

Die in den vorangehenden Abschnitten erläuterten UWE-Konzepte sind in einem Metamodell als einer konservativen Erweiterung des UML 2.0-Metamodells definiert. Die Konservativität der Erweiterung bedeutet, dass das UML 2.0-Metamodell nicht verändert wird, alle UWE-Metamodellelemente sind durch Vererbung aus bestehenden UML 2.0-Metamodellelementen abgeleitet und nur für

diese abgeleiteten Elemente werden zusätzliche Eigenschaften und Beziehungen definiert. Das resultierende UWE-Metamodell ist durch diese Vorgehensweise direkt in ein UML-Profil (Erweiterung der UML-Notation) übersetzbar und UWE bleibt somit den Standard-UML-CASE-Werkzeugen kompatibel.

Die UWE-Erweiterung, siehe Bild 8, besteht aus den zwei Paketen *Core* und *Adaptivity*, die die separate Modellierung der verschiedenen Ebenen einer Webanwendung und das Querschneiden der Adaptivität durch ihre Abhängigkeiten widerspiegeln. Das Paket *Requirements* enthält die anforderungsspezifischen UWE-Erweiterungen für Anwendungsfälle und Aktivitäten, die Pakete *Navigation* und *Presentation* bündeln die entsprechenden UWE-Konzepte für die Navigationsstruktur und die Darstellung. Dagegen enthalten gegenwärtig die Pakete *Content* und *Process* keine neuen Konzepte: der Webanwendungsentwickler kann alle UML 2.0-Konzepte für den Entwurf von Inhalts- und Benutzermodellen sowie von Prozessmodellen verwenden. Schließlich enthält *Adaptivity* die UWE-Elemente für die aspektorientierte Modellierung von Adaptivität.

5.2 Modelltransformationen im UWE-Prozess

Der UWE-Prozess beginnt mit der Ermittlung und Spezifikation der Anforderungen. Ziel ist es, eine stabile Menge von Anforderungen zu identifizieren und diese mit webspezifischen Modellelementen, wie in Abschnitt 3 skizziert, zu beschreiben. Hiermit ist die Basis für die ersten Modelltransformationen geschaffen, die aus einem Anforderungsmodell

ein Entwurfsmodell, bestehend aus Inhalts-, Navigations-, Präsentations- und Prozess-Modellen, erstellen, siehe Bild 9. Die Ergebnisse dieser Transformationen können vom Webanwendungsentwickler in weiteren Schritten in einem semi-automatischen Prozess verfeinert werden. In der Tat werden einige Verfeinerungsschritte automatisch durchgeführt, wie das Hinzufügen von Indexen und Auswahlmenüs (siehe Abschnitt 3.3). Entwurfsmodelle werden letztendlich in plattformspezifische Implementierungen, etwa JSPs, transformiert. Dieser UWE-Kernprozess wird durch eine Transformation in eine UML-Zustandsmaschine erweitert – das "big picture" (siehe Knapp, Zhang 2006), das die Entwurfsmodelle zu einem Modell integriert und die Validierung von UWE-Modellen mit dem UML-Modellübersetzer Hugo/RT (www.pst.ifi.lmu.de/projekte/hugo) ermöglicht.

Die Entwicklung von UWE und gleichzeitig die Entwicklung der Techniken zur Implementierung der Modelltransformationen resultieren in einer Menge von Transformationstypen, die im Rahmen des UWE-Ansatzes verwendet werden (Koch 2007). Transformationsregeln von Anforderungs- zu Entwurfsmodellen sind als ATL-Regeln (www.eclipse.org/m2m/atl) definiert. Transformationsregeln zur Verfeinerung der Entwurfsmodelle stehen dem Entwickler sowohl als in Java implementierte Funktionen im Modellierungswerkzeug ArgoUWE (siehe Abschnitt 5.3) als auch in ATL zur Verfügung. Die Integration der Entwurfsmodelle ist anhand von Graphtransformationen definiert und mittels des Graphtransformationswerkzeuges AGG umgesetzt (Knapp und Zhang, 2006). Bild 10 stellt exemplarisch die Abbildung von Navigationsknoten auf Zustände dar. Die in ATL definierten Modell-zu-Text Transformationen generieren JSPs und JavaBeans für Präsentations- und Inhaltsobjekte. Komplementär wird die mit UML-Aktivitätsdiagrammen dargestellte Geschäftslogik zur Laufzeit interpretiert und ausgeführt.

5.3 Werkzeugunterstützung

Die UWE-Modelltransformationen sind in einem Eclipse-Plugin implementiert (Kraus et al. 2007). Für die Modellierung stellen wir das Werkzeug ArgoUWE zur Verfügung, eine Erweiterung des Open-Source-UML-Modellierungswerkzeugs

ArgoUML (www.argouml.org). ArgoUWE bietet nicht nur einen maßgeschneiderten graphischen Editor für alle UWE-Diagramme, sondern automatisiert auch einige der oben genannten Modelltransformationen. Ein besonderes Highlight von ArgoUWE ist seine Hilfe für den Erhalt der Modellkonsistenz durch die Erweiterung des ArgoUML-Features *Design Critiques*: Modelle werden ständig überwacht, jede Verletzung einer Modellkonsistenzbedingung generiert eine nichtunterbrechende Warnung.

6. Fazit und Ausblick

UWE unterstützt die modellgetriebene Entwicklung von Websystemen von der Anforderungsspezifikation über die Entwurfsmodellierung bis zur Anwendungsgenerierung. Die Entwicklungsschritte basieren auf Modelltransformationen, die weitgehend in Werkzeugen automatisiert sind. Die Einhaltung von Standards (UML, MDA, usw.) erleichtert die Einarbeitung des Entwicklers und ermöglicht das Verwenden von nicht-UWE-spezifischen Werkzeugen.

UWE kann in mehrere Richtungen erweitert werden: Bei der Modellierung planen wir, auch die so genannten *Rich Internet Applications* zu erschließen, die sich unter anderem durch reichhaltige Benutzeroberflächen, client-seitige Berechnung und asynchrone Client-Server-Kommunikation von konventionellen Webanwendungen unterscheiden. Die Werkzeuge zur Unterstützung von UWE werden in eine Entwicklungsumgebung integriert; außerdem wird im Rahmen des Netzwerks MDWEnet (www.pst.ifi.lmu.de/projekte/mdwenet) angestrebt, die Interoperabilität zwischen UWE und anderen Web-Engineering-Ansätzen zu verbessern.

Danksagung

In den letzten zwei Jahren wurden konzeptuelle Weiterentwicklungen und Implementierungen im UWE-Ansatz durch das DFG-Projekt MAEWA (WI 841/7-1) finanziert.

Literatur

Filman, R. E., Elrad T., Clarke S., Aksit, M. (Hrsg.): *Aspect-Oriented Software Development*. Boston: Addison-Wesley, 2004.

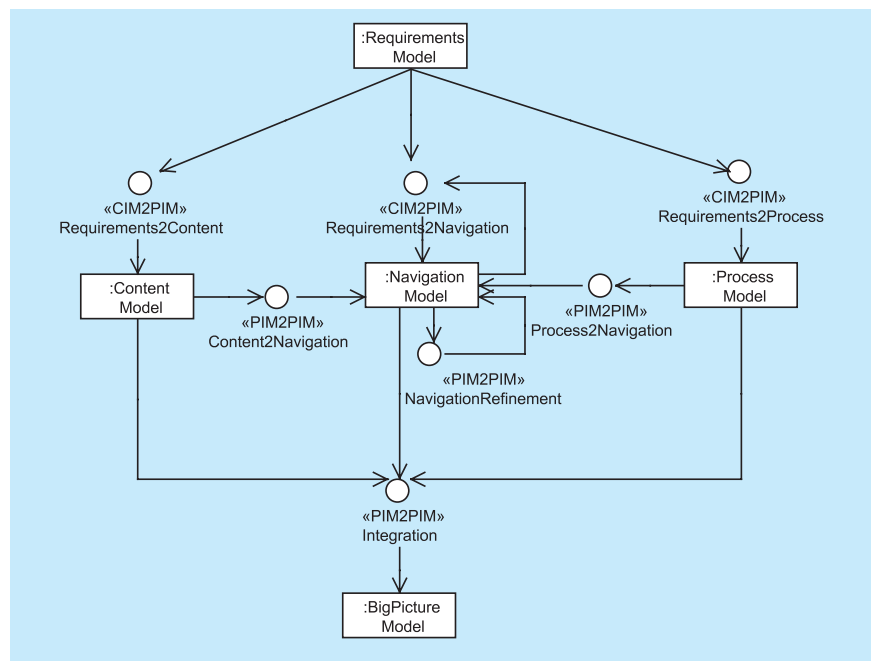


Bild 9: Modelltransformationen in UWE (Auswahl)

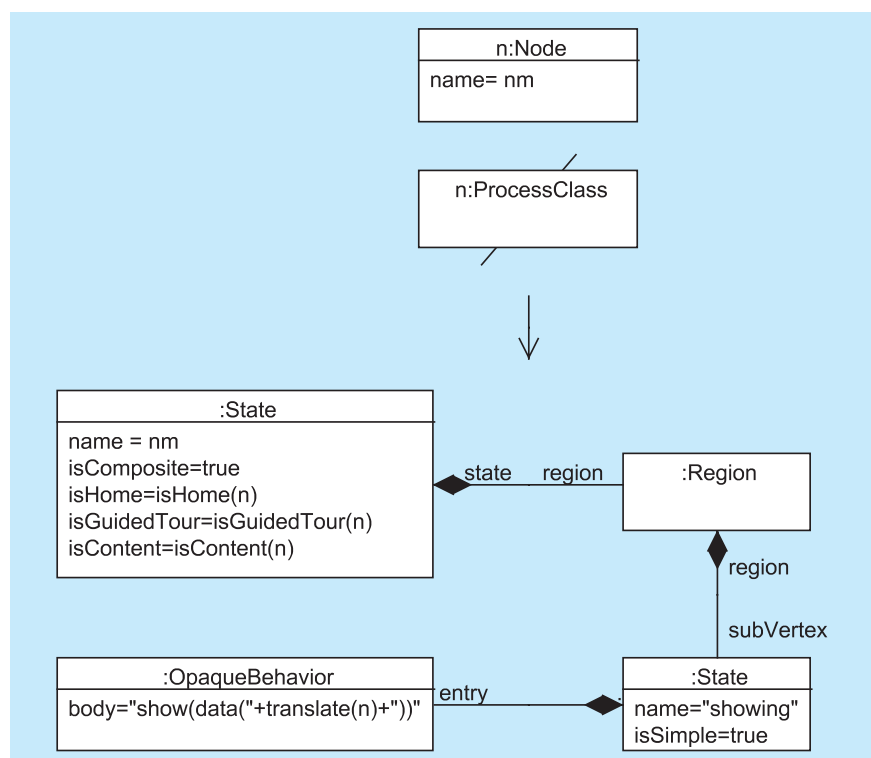


Bild 10: Abbildung von Navigationsknoten auf Zustände

Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (Hrsg.): *Web Engineering*. Chichester: John Wiley & Sons, 2006.

Knapp, A., Zhang, G.: Model Transformations for Integrating and Validating Web Application Models. In: *Proc. Modellierung 2006 (MOD'06)*. (Hrsg. Mayr, H. C., Breu, R.) Bonn: Gesellschaft für Informatik, Lect. Notes Inf. P-82 (2006) 115–128.

Koch, N.: Classification of Model Transformation Techniques used in UML-based Web Engi-

neering. *IET Software Journal* 1 (3) (2007) 98–111.

Kraus, A., Knapp, A., Koch, N.: Model-Driven Generation of Web Applications in UWE. In: *Proc. 3rd Int. Wsh. Model-Driven Web Engineering*. CEUR-WS 261, 2007. <http://ceur-ws.org> (Letzter Zugriff: 19.9.2007).

Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (Hrsg.): *Web Engineering: Modelling and Implementing Web Applications*. Berlin: Springer, 2007.



1



2



3



4

1 Alexander Knapp ist Juniorprofessor für Softwareentwicklung an der Ludwig-Maximilians-Universität München. Seine Hauptinteressensgebiete sind formale Methoden, komponenten- und serviceorientierte Softwareentwicklung, Sicherheit und Web-Engineering. Er leitet das DFG-Projekt „InfoZert“ über Informationsflusssicherheit

und das IUK-Projekt „Grenzenloses Credentialmanagement“. Zur Zeit vertritt Herr Knapp eine Professur für Software and Systems Engineering an der Universität Augsburg.
E-Mail: knapp@pst.ifi.lmu.de

2 Nora Koch ist wissenschaftliche Mitarbeiterin bei Prof. Dr. Martin Wirsing in der Forschungsgruppe Programmierung und Softwaretechnik an der Ludwig-Maximilians-Universität München. Ihre Forschungsschwerpunkte liegen im Bereich Web Engineering und UML mit Fokus auf modellgetriebener Entwicklung von Webanwendungen. Nora Koch ist zusätzlich bei der Firma FAST GmbH im Bereich IT-Consulting tätig.
E-Mail: kochn@pst.ifi.lmu.de

3 Prof. Martin Wirsing ist seit 1992 Inhaber des Lehrstuhls für Programmierung und Softwaretechnik an der Ludwig-Maximilians-Universität München. Seine aktuellen Forschungsinteressen umfassen Software Engineering für verteilte mobile Systeme und für Webanwendungen sowie deren

mathematische Grundlagen. Er ist Autor und Herausgeber von mehr als 20 Büchern und hat mehr als 170 wissenschaftliche Arbeiten veröffentlicht. Martin Wirsing ist Koordinator des EU-Projekts SENSORIA zur systematischen Entwicklung von dienstorientierten Systemen und außerdem an mehreren weiteren nationalen und internationalen Projekten beteiligt. Er ist Mitglied des Senats und des Hochschulrats der LMU München, Vorsitzender des wissenschaftlichen Beirats der INRIA Frankreich und Mitglied im wissenschaftlichen Beirat weiterer angesehener Forschungsinstitute.
E-Mail: wirsing@lmu.de
<http://www.pst.ifi.lmu.de>

4 Gefei Zhang ist wissenschaftlicher Mitarbeiter bei Prof. Dr. Martin Wirsing in der Forschungsgruppe Programmierung und Softwaretechnik an der Ludwig-Maximilians-Universität München. Zu seinen aktuellen Forschungsinteressen gehören Modellbasiertes Web-Engineering und Aspektorientierte Modellierung.
E-Mail: zhangg@pst.ifi.lmu.de



HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

WEITERBILDUNG



Das Weiterbildungsangebot

MAS Human Computer Interaction Design

richtet sich an Personen, die sich mit Entwurf und Entwicklung von User Interfaces befassen.

Werden Sie Expertin oder Experte für

User Interfaces, Interaction Design, Usability

Im deutschsprachigen Raum existiert derzeit kein vergleichbarer Studiengang. Dieses neue, berufsbegleitende, interdisziplinäre Weiterbildungsstudium führt SpezialistInnen aus den Gebieten Informatik, Design und Psychologie zusammen.

Der Masterstudiengang richtet sich an InformatikerInnen, DesignerInnen sowie Psychologinnen und Psychologen im technischen Umfeld. Die Studierenden komplettieren ihre Grundlagen mittels ausgewählter Kurse in den jeweils fremden Gebieten und erlernen in Theorie und Projekten die übergreifende HCI Methodik.

Die Zertifikatskurse können einzeln gebucht und belegt werden.

Das Studium wird von der Universität Basel und der HSR Hochschule für Technik Rapperswil gemeinsam angeboten und in Kooperation mit der Hochschule für Gestaltung und Kunst Basel und mehreren Industriepartnern durchgeführt.

Nächster Einstieg in das Masterstudium im April 2008!

Informationsveranstaltungen in Basel und Rapperswil: www.hcid.ch

Abschluss	MAS Master of Advanced Studies
Studiendauer	3 Jahre inkl. Masterarbeit, 2 Zertifikatskurse à 250 h, Masterarbeit 300 h
Unterrichtszeit	Blockveranstaltungen à 2 Tage; Freitag und Samstag jeweils ganztägig
Studienorte	Basel und Rapperswil
Beginn	18. April 2008,
Informationen	www.hcid.ch
Auskünfte	T +41 (55) 222 49 22, weiterbildung@hsr.ch