Gut koordiniert

jBPM-Einsatz zur Modellierung und Ablaufsteuerung von Geschäftsprozessen mit Human Task Integration

Hongguang Yang, Christoph Uhe

Das Open-Source-Prozess-Framework jBPM liegt seit 2011 in der Version 5 vor und wartet gegenüber den beiden Vorgängerversionen mit einigen Neuerungen in der Konzeption, der Architektur und in der Implementierung auf, beispielsweise mit Prozessmodellierung mit BPMN 2 (Business Process Model and Notation), Benutzerinteraktion basierend auf der WS-HT-Spezifikation (Web Services for Human Task), Anbindung an eine Rule Engine sowie Separierung von Historie- und Laufzeitinformationen für die Prozessinstanzen. In diesem Artikel wird ein typisches Einsatzszenario der jBPM-Technologien anhand eines konkreten Projekts der Versicherungsbranche aufgezeigt. Was das jBPM-Framework mit seinen Erneuerungen seit der Version 5 für die Anwendungspraxis bringt, wird aus der gewonnenen Projekterfahrung betrachtet. Abschließend spricht der Artikel die noch offenen Punkte dieses Frameworks an.

Projektvorstellung

Der zentrale Anwendungsfall beschreibt einen Geschäftsprozess (Beschwerdemanagement), wie er typisch für Versicherungsunternehmen ist. Das betreffende Versicherungsunternehmen benötigt mehrere fertige Anwendungen von verschiedenen Softwareanbietern, um die spezifischen, teamorientierten Aufgaben im Bereich Beschwerdemanagement ausgehend vom Callcenter, über die Sachbearbeitung bis hin zur Vorgangsüberwachung (Management von Service Level Agreements – SLAs) zu koordinieren und durchzuführen. Im Vordergrund stehen deshalb die Integration von Softwareanwendungen und die Interaktionen zwischen den Prozessen und den Benutzern bzw. ihren Organisationseinheiten. Dazu werden die gesamten Arbeitsabläufe mit den jBPM-Prozessen unter intensivem Einsatz von Benutzeraufgaben (Human Tasks) modelliert.

Da die existierenden Anwendungen auf die jeweiligen internen Arbeitsschritte zugeschnitten sind, kann jede Benutzeraufgabe (Human Task im Sinne des Prozessmodells) auf eine Anwendungskomponente abgebildet werden. Dies hat zu einer der entscheidenden Prozessmodellvorgaben geführt, die besagt, dass eine Benutzeraufgabe immer mit einer der vorgegebenen Anwendungskomponenten erledigt wird. Sobald ein Anwendungswechsel stattfindet, wird er im Prozess explizit als Übergang zwischen zwei Benutzeraufgaben modelliert.

Eine zweite wichtige Modellvorgabe legt fest, dass der Wechsel eines Benutzers bzw. einer Organisationseinheit im Rahmen einer Aufgabenzuweisung im Prozess ebenso explizit als Übergang zwischen zwei Benutzeraufgaben modelliert wird.

Prozessmodellierung mit BPMN 2.0

jBPM unterstützt die Modellierung von Prozessen mittels *Business Process Model and Notation* [BPMN] in der aktuellen Version 2.0. Diese standardisierte Vorgehensweise für die Handhabung von Geschäftsprozessen spiegelt sich bei jBPM im mitgelieferten Eclipse-Plug-in wider, das sowohl das Editieren von Prozess-Template-Dateien als auch ihr Deployment durch ein integriertes REST-Interface zu einem internen Repository (*Guvnor*) ermöglicht. Anstatt die Entwicklungsumgebung Eclipse einzusetzen, können alternativ die Prozess-Templates durch ein externes Tool (*Prozess-Designer*) verarbeitet werden.

Abbildung 1 zeigt das modellierte Prozessbild. Jede Beschwerde seitens eines Kunden (=> „*Partner*“) führt zum Anlegen einer jBPM-Prozessinstanz gemäß diesem Template und durchläuft die vorgegebenen Bearbeitungsschritte.

Das Prozessbild enthält die einzelnen Schritte in abgerundeten Kästchen (*Aufgaben* oder *Task*), die beim Durchlauf von Verbindungen sowie von den Knoten angesteuert werden. Bei den verwendeten Kästchen handelt es sich mit einer Ausnahme (*InitProcess*) nur um *Human Tasks*, die durch das Symbol des Männchens mit einer Akte in der Hand gekennzeichnet sind. Bei den Knoten gibt es zusammenführende (*converge*) und abzweigende (*diverge*) Knoten. Bei den abzweigenden Knoten handelt es hier immer um eine „XOR“-Entscheidung, die den Ablauf der Prozessinstanz dynamisch in einen der Zweige führt.

- *Prozessstart*: Der Startknoten (grüner Kreis in Abb. 1) kennzeichnet das Anlegen einer Prozessinstanz gemäß dem modellierten Template. Es erfolgt ein automatischer Übergang

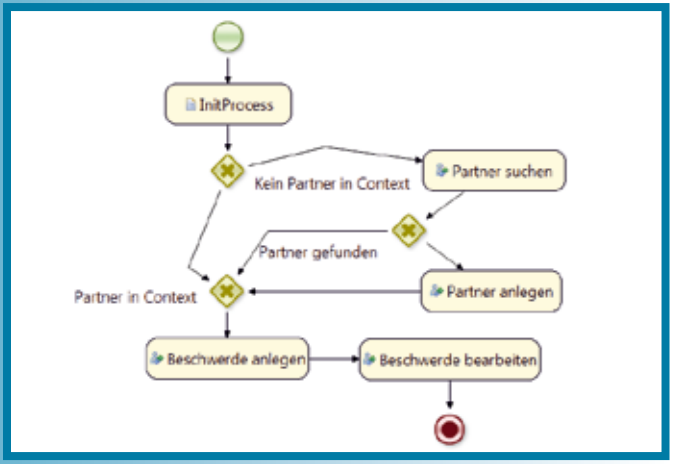

Abb. 1: Beschwerdeprozess

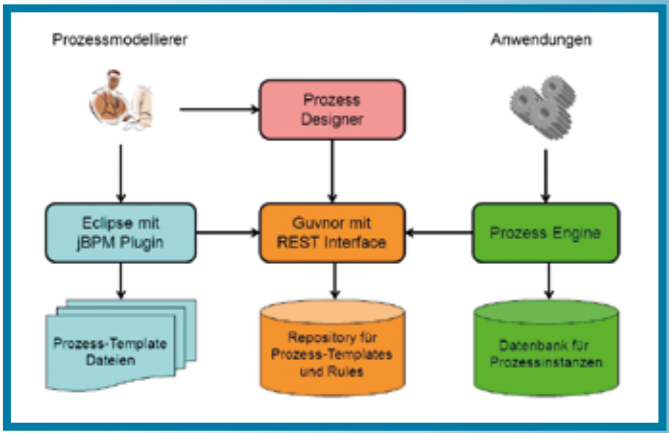

Abb. 2: Technisches Vorgehen beim Modellieren, Deployment und Nutzen von Prozessen

vom Startknoten zum ersten Kästchen, das eine skript-basierte Aufgabe darstellt (Dokumentsymbol).

- *InitProcess*: Das ist eine sogenannte Skript-Aufgabe: Über die Inline-Java- oder Mvel-Codes werden alle erforderlichen Prozessvariablen initial gesetzt.
- *Abzweigung in Abhängigkeit vom Vorhandensein des Partner-Contexts*: Liegt die Kundeninformation bereits im Anwendungskontext (*Partner in Context*) vor, geht die Bearbeitung sogleich zu *Beschwerde anlegen*. Wenn nicht (*Kein Partner in Context*), dann verlangt das System, dass der Sachbearbeiter einen Kunden zuerst sucht (Benutzeraufgabe *Partner suchen*). Danach setzt das System die Kundeninformation in den Anwendungskontext.
- *Abzweigung in Abhängigkeit vom Suchergebnis*: Wird der Kunde gefunden, kann eine Beschwerde für ihn angelegt werden (*Beschwerde anlegen*). Ist noch keine Kundeninformation in der Datenbank vorhanden, wird diese zuerst angelegt (*Partner anlegen*).
- *Zusammenführung vor „Beschwerde anlegen“*: Der zusammenführende Knoten vor dem Kästchen *Beschwerde anlegen* stellt den Ausgangspunkt dar, an dem alle erforderlichen Rahmenbedingungen für das Anlegen und Bearbeiten einer Kundenbeschwerde erfüllt worden sind.
- *Beschwerde anlegen*: Hierbei handelt es sich um die eigentliche Benutzeraufgabe, nämlich für die Kundenbeschwerde eine Bearbeitungsinstanz zu erstellen und alle weiteren relevanten Informationen zu erfassen.
- *Beschwerde bearbeiten*: Die weitere Bearbeitung der Beschwerde führt i. d. R. ein anderer Sachbearbeiter durch. Dies wird in der Prozessmodellierung durch ein weiteres Benutzeraufgaben-Kästchen *Beschwerde bearbeiten* festgelegt. Das Bearbeiten der Beschwerde wird bzgl. Zeitüberschreitung überwacht. Im Falle einer Zeitüberschreitung wird die Beschwerde an die nächst höherer Organisationseinheit eskaliert, da ein entsprechender SLA (Service Level Agreement) existiert.
  Es sei nur erwähnt, dass das Auslösen einer Eskalation durch einen festgelegten Parameter unter Nutzung des jBPM-*Timers* modelliert wurde.
- *Prozess Ende*: Ist der Beschwerdevorgang abgearbeitet (Beispielsweise wenn der Kunde die Bestätigung der abgeschlossenen Bearbeitung per Brief erhalten hat), wird die Prozessinstanz beendet.

Die Prozess-Template-Datei kann von der Prozessmaschine direkt eingelesen, in die sogenannte Knowledge Base eingefügt und somit für die Erstellung von Prozessinstanzen genutzt werden. Dazu gibt es ein entsprechendes Java-API. Darüber hinaus unterstützt das jBPM-Eclipse-Plug-in ein direktes Deployment der Prozess-Templates in ein vorgegebenes Guvnor Repository, um u. a. eine Clusterumgebung besser zu unterstützen. Das ist in einem Unternehmensumfeld sinnvoll, da die Prozessmodellierung, der Test, das Deployment und die Nutzung im Laufzeitsystem in der Gesamtheit betrachtet werden sollen. Abbildung 2 stellt das technische Vorgehen zur Unterstützung dieser Aspekte dar.

Verwaltung von Benutzeraufgaben mit WS-HT

Wie bereits erwähnt, stellt jede Benutzeraufgabe die Nutzung einer in sich geschlossenen Anwendungskomponente dar, die neben ihrer Verarbeitungslogik in der Fachlichkeit eine, aber auch mehrere GUI-Masken für die Benutzerinteraktionen beinhalten kann. Die spezifischen Anforderungen des Projektes, dass mehrere existierende Anwendungen in ihren Prozess- und Dialogabläufen interagieren müssen, macht eine standardisierte Schnittstelle zwischen den modellierten jBPM-Prozessen und implementierten Benutzeraufgaben notwendig.

Hierfür eignet sich die WS-HT-Spezifikation, aktuell in der Version 1.1 vorliegend [WSHT]. Sie standardisiert den Lebenszyklus einer Benutzeraufgabe in Form von Task-Status und den Status-Übergängen, unabhängig davon, ob es sich bei ihr um eine einfache GUI-Maske, um eine vom Benutzer gesteuerte Job-Verarbeitung oder gar um die langdauernde Bearbeitung eines komplexen Anwendungsfalls handelt. Die WS-HT legt insbesondere auch die Service-Schnittstelle zur Kommunikation zwischen dem sogenannten Task Parent (in diesem Fall der Prozessinstanz), dem

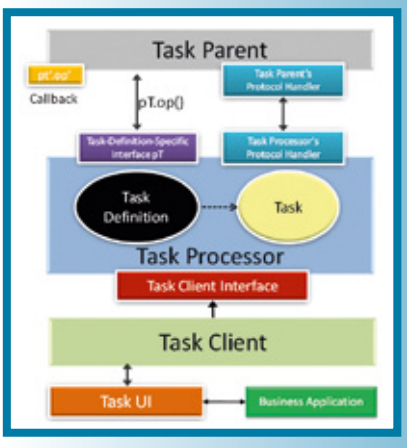

Abb. 3: Task Parent, Task Processor und Task Client (Figure 7 aus [WSHT])

Status/Übergang	Created	Ready	Reserved	inProgress	Completed	Failed
`Created`	–	`activation (a)`	`activation (b)`	`activation (c)`	–	–
`Ready`	–	`forward`	`claim||delegate`	`start`	–	–
`Reserved`	–	`release||forward`	`delegate`	`start`	–	–
`inProgress`	–	`release||forward`	`stop||delegate`	–	`complete`	`failed`
`Completed`	–	–	–	–	–	–
`Failed`	–	–	–	–	–	–

Tabelle 1: Wichtigste Status und Status-Übergänge einer Benutzeraufgabe

Task Service (Task Processor) und dem Task Client (Anwendung) fest. Abbildung 3 zeigt die Architektur.

Der wesentliche Unterschied zwischen dem Task Parent und dem Task Client besteht darin, dass der Task Parent für die Erstellung einer Benutzeraufgabe sowie für die rückwärtige Weiterverarbeitung von Status-Änderungen zuständig ist, da er ja in einem übergeordneten Anwendungskontext (Prozessinstanz) die Benutzeraufgabe repräsentiert. Demgegenüber sorgt der Task Client für die Erfüllung der Lebenszyklusfunktionen einer Benutzeraufgabe, also für die Statusänderungen selbst. Hinter einem Task Client verbirgt sich die eigentliche Anwendungskomponente (*Task UI* und *Business Application*), die der Benutzer zur Erledigung von Aufgaben benutzt.

Zwischen dem Task Parent und dem Task Processor läuft ein asynchrones Protokoll, basierend auf der speziellen Webservice-Porttypdefinition (pt) und der speziellen Operation (op). Darauf soll hier nicht näher eingegangen werden. WS-HT gibt hierfür nur eine generelle Empfehlung, aber keine Spezifikation. Das ist insofern logisch, da der Task Parent ein recht weites Spektrum des spezifischen Anwendungskontexts annehmen kann – in unserem Fall handelt es sich speziell um den Prozessinstanz-Kontext. Das Projekt nutzt eine eigene Implementierung hierfür.

Das Task-Client-Interface ist hingegen in der WS-HT soweit durchspezifiziert, dass aus der vorgegebenen WSDL die Webservices generiert werden können. Das Interface regelt insbesondere die Statusänderungen einer Benutzeraufgabe. Tabelle 1 fasst, ohne auf die Details einzugehen, die wichtigsten Status und die Methoden zu den Statusübergängen (von der Zeile zur Spalte) zusammen.

In der Tabelle ist z. B. zu sehen, dass eine neu erstellte Benutzeraufgabe (Status: `Created`) durch die Methode `activation (c)` in den Status `inProgress` (also in Bearbeitung) übergeht. Dieselbe Task mit dem Status `inProgress` kann dann durch die Methode `complete` in den Status `Completed` übergehen, was eine fehlerfreie Bearbeitung durch einen Benutzer kennzeichnet.

jBPM stellt eine native Implementierung des Task Service in Anlehnung an die WS-HT-Spezifikation bereit, die jedoch statt Webservices ein Java-API bedient. Dafür ist die Nutzung unkompliziert. jBPM schreibt jedoch nicht die Nutzung dieser Implementierung vor, sondern erlaubt die Nutzung anderer Implementierungen durch das Einbinden mittels einer Java-Interface-Klasse (`WorkItemHandler`).

Von dieser Möglichkeit machte das angehende IT-Projekt Gebrauch und implementiert einen eigenen Task Service mit allen spezifizierten Webservices. Er fungiert als WS-HT-Provider und siedelt sich in einer speziellen Anwendungskomponente zur Verwaltung von „Arbeitslisten“ an.

Die Nutzung von WS-HT hat sich insbesondere dadurch bewährt, dass alle im Projekt verwendeten Anwendungen zwar sehr inhomogen sind, sie aber alle ausschließlich die WS-HT-Status-Behandlungsschnittstellen realisieren müssen, um in der Gesamtheit mitwirken und ein vergleichbares Handling für die Benutzer bieten zu können. Abbildung 4 zeigt die Implementierungsarchitektur. Darin sind die jBPM-Framework-Komponenten in Grün, die WS-HT-Implementierungen in Blau und die Anwendungskomponenten in Gelb dargestellt. Die Bezeichnungen *BP* (business partner), *CRM* (customer relationship management) und *DMS* (document management system) sind Abkürzungen für die spezifischen Anwendungen.

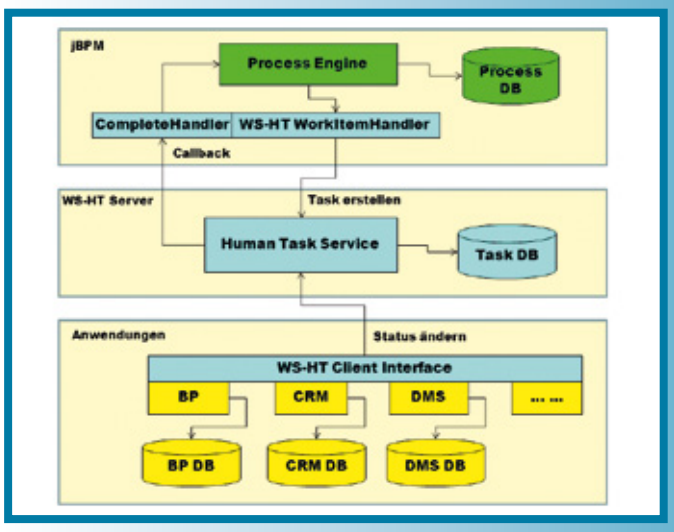

Abb. 4: Task Service als WS-HT-Implementierung und verschiedene Anwendungen

Die größten Vorteile der Nutzung von jBPM und damit verbunden der WS-HT liegen in der Möglichkeit, die vorhandenen Business-Funktionalitäten aus verschiedenen Anwendungen über eine einheitliche Vorgehensweise miteinander zu verknüpfen, ohne die bestehenden Anwendungen umfangreich zu ändern.

jBPM-Framework-Integration ins Laufzeitsystem

Eine der technischen Herausforderungen beim Einsatz von jBPM (speziell in der Version 5) besteht in der Integration des Frameworks in die Laufzeitumgebung. Hierbei handelt es sich um den JBoss Application Server Container. Ironischerweise liefert weder das jBPM-Entwicklungsteam noch die JBoss Group eine schlüsselfertige Lösung für die Nutzung des jBPM-Frameworks im hauseigenen Applikationsserver – ein Wermutstropfen schlechthin. Während die jBPM-Version 3 noch ein fertiges Set von EJB-Interfaces bereitgestellt hat, fehlt in der jBPM-Version 5 jeglicher konzeptionelle Hinweis zur Einbettung des Frameworks in ein EJB-Umfeld.

Das Projektteam hat allerdings die Quelloffenheit von jBPM genutzt und durch die Code-Analyse einen möglichst sinnvoll erscheinenden Mechanismus für die folgenden technischen Einzelfacetten abgeleitet, die für die Integration umgesetzt wurden:

- Knowledge Base Handling (Initialisieren, Wiederverwenden),
- Knowledge Session Handling (Nutzung von stateless bzw. stateful Knowledge Sessions),
- Transaktions-Handling (Festlegung von Strategie, Demarkation und Binding an Sessions),
- Nutzung von Guvnor Respository (Speichern und Bereitstellen von Prozess-Templates),
- Nutzung von jBPM Management Console (Konfigurationen einer Vielzahl von Persistenz-Definitionen).

Als Ergebnis ist ein dediziertes EJB-Modul als *Process Engine Adapter* entstanden, der über diese technischen Facetten hinaus auch alle Kommunikationsschnittstellen (JMS, Webservice und EJB) zu den anderen verwendeten Softwarekomponenten beinhaltet. Abbildung 5 zeigt die Einbettung der Prozessmaschine in den JBoss Application Server Container mittels des selbst implementierten Adapters.

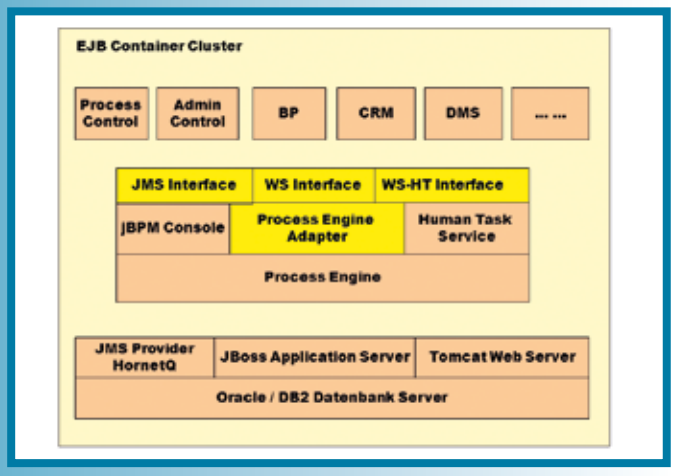

Abb. 5: Einbettung der Prozessmaschine in eine Server-Umgebung mittels des Adapters

Der Prozess Engine Adapter und die von ihm unterstützten Schnittstellen sind in Gelb dargestellt. In der oberen Reihe stehen die Anwendungskomponenten, die über unterschiedliche Schnittstellen via Adapter mit der Prozessmaschine oder dem WS-HT-Service kommunizieren. Im unteren Bereich sind die Infrastrukturkomponenten des Applikationsservers dargestellt. Speziell kommt die *JBoss HornetQ*-Messaging-Komponente zum Einsatz, die asynchrone Anfragen und Antworten gegen ein System-Shutdown absichern kann, indem die Nachrichten persistiert werden.

Fazit

Alle wesentlichen Erneuerungen von jBPM seit der Version 5 sind von der Architektur, über die Konzeption bis hin zur Implementierung als gelungen zu betrachten. Insbesondere stellt die umfassende Unterstützung der BPMN- und WS-HT-Standards ein essenzielles Plus im praktischen Einsatz dar.

Die Zusammenführung von jBPM und Drools macht die übergangsfreie Nutzung von Prozessen und Rules möglich, was von der Grundidee her als brillant gesehen werden kann. Dies spiegelt sich in der Tatsache wider, dass die Process und die Rule Engine (ebenso die verwendeten Knowledge Base und Sessions) dieselben sind. Aufgrund des Umfangs dieses Artikels kann nicht auf die Nutzung von Rules im Projekt eingegangen werden.

Das jBPM-Framework ist robust und die Engine im Sinne der Performance und Funktionalität leistungsfähig. Eine Ausnahme davon stellt das administrative Werkzeug jBPM Console dar, dessen Darstellungsfehler in der Software von einer Version zur anderen weiter geschleppt werden. Die Quellcodes und eine Vielzahl von Beispielen und Testklassen unterstützen das Verständnis des Frameworks. Die vorhandenen Dokumentationen geben allerdings die konzeptionellen Merkmale nicht hinreichend wider.

Ein weiteres Manko sehen die Autoren im Fehlen einer Unterstützung für die Nutzung von jBPM in verschiedenen Laufzeitumgebungen. Dies konnten wir aus eigener Erfahrung im Fall des Applikationsservers von JBoss feststellen. Aber auch für die Nutzung durch das Spring-Framework fehlt ein wirklich hilfreicher Hinweis. Dieses Manko liegt nicht einmal in den fehlenden technischen Möglichkeiten. Vielmehr würde ein verstärktes Mitdenken im Bereich praktischer Softwaretechnik helfen, die Brücke vom Framework zur Anwendung zu bauen. In der letzten JBoss Enterprise SOA Edition (Version 5) hat die JBoss Group das jBPM-Framework Version 3.2 als Infrastrukturkomponente integriert. Es bleibt die spannende Frage an die JBoss Group, wie schnell sie in der künftigen SOA-Plattform die neue Version 5 von jBPM berücksichtigt. Wenn dies bereits geschehen wäre, wäre unser *Process Engine Adapter* komplett überflüssig gewesen.

Literatur und Links

[BPMN] Business Process Model and Notation (BPMN), Version 2.0, OMG, 2011, `http://www.omg.org/spec/BPMN/2.0`

[Drools] `http://www.jboss.org/drools`

[Guvnor] `http://www.jboss.org/guvnor`

[jBPM] `http://www.jboss.org/jbpm/`

[WSHT] Web Services – Human Task (WS-HumanTask) Version 1.1, OASIS, 2010, `http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803`

[Yang11] Hongguang Yang, Erfahrung zur Ablaufsteuerung mittel jBPM, in: JavaSPEKTRUM, 01/2011

Dr. Hongguang Yang beschäftigt sich seit zehn Jahren mit Java. Sein Schwerpunkt ist die Architektur und Implementierung von IT-Anwendungen für Unternehmen. In den letzten Jahren bediente er Firmen in den Branchen Öffentlicher Dienst, Geoinformatik, Verkehrstelematik und Versicherung. Dr. Yang ist Mitarbeiter und Lead IT Consultant des IT-Unternehmens msg systems ag.
E-Mail: hongguang.yang@msg-systems.com.

Christoph Uhe beschäftigt sich seit fünfzehn Jahren mit Java und seit 2004 intensiv mit serviceorientierten Architekturen und deren Umsetzung in der Versicherungsbranche. Seine Schwerpunkte liegen im Bereich Beratung bzgl. Aufbau von Enterprise-Architekturmanagement (EAM) für Versicherungsunternehmen. Er ist Abteilungsleiter für Architekturen in der Branche Insurance des IT-Unternehmens msg systems ag.
E-Mail: Christoph.Uhe@msg-systems.com