mehr zum thema:
http://en.wikipedia.org/wiki/Service-oriented_architecture
http://www.visual-rules.de
http://www.ilog.de/products/businessrules/
http://www.fiorano.com/whitepapers/fiorano_service_components.htm
http://www.soacenter.com/
http://java.sun.com/developer/technicalArticles/WebServices/soa3/

BUSINESS RULES UND SOA – TECHNOLOGIEN FÜR DYNAMISCHE GESCHÄFTSPROZESSE

Der nun folgende Artikel beschäftigt sich mit zwei technologischen Strömungen, die in den vergangenen Monaten und Jahren an zunehmender Bedeutung im Unternehmensumfeld gewonnen haben. Auf den ersten Blick scheinen sie zwei unterschiedliche Problemsegmente zu lösen. Dies sind der Umgang mit dynamischen Geschäftsregeln und die Zusammenführung verschiedener Software-Dienstleistungen in gemeinsamen technischen Infrastruktur, einer service-orientierte Architektur (SOA). Wir wollen im Folgenden beleuchten, inwiefern sich beide Ideen auf wirksame Weise miteinander kombinieren und zu einem intelligenten Gesamtansatz verbinden lassen.

der autor

Lars Wunderlich (E-Mail: lars.wunderlich@gmx.net) ist Autor zahlreicher Fachartikel und verschiedener Publikationen zu den Themen Java, Eclipse, Rules Engines und Java Management. Er arbeitet als System Architekt und Application Designer bei der TUI InfoTec GmbH in Hannover.

Wenn wir einmal ganz den Blick von heutigen „Software-Hypes“ und „Buzzwörtern“ weg wenden und einfach an uns die technologischen Entwicklungen der letzten Jahrzehnte passieren lassen, so fallen einige Dinge direkt ins Auge.

Mit der zunehmenden Industrialisierung und der späteren Durchsetzung des Computerzeitalters suchten und suchen viele Personengruppen vom Entwickler bis hin zur Firmenkonzernspitze nach einer omnipotenten Softwarestrategie. Während Applikationen auf Einzelrechnern früher mehr Werkzeug für die Beschleunigung einiger weniger Arbeitsschritte waren, haben Internet und Netzwerke Chancen und Möglichkeiten revolutioniert. Ab einer gewissen Unternehmensgröße und einem Wachstum an Softwarekomponenten, Applikationen und Integrationssystemen gewinnt man aber den Eindruck eines Flickenteppichs aus vielen Insellösungen. Er ist verstrickt und verwoben durch zahllose Online- und Offline-Schnittstellen. Sie sprechen unterschiedliche Sprachen, arbeiten zu unterschiedlichen Zeitpunkten, haben unterschiedlichste Aufgaben. Sie bilden einerseits das prozesstechnische Softwarebackend, das notwendig ist, um Aktitäten in Geschäftsprozessen zu unterstützen und zu beschleunigen; andererseits sind sie aber auch Spiegel unserer Unternehmensstruktur. Manche entwickelten sich weiter, bei anderen verwischen die Aufgabenbereiche. Sie decken ebenso wie Abteilungen und Bereiche verschiedene Teilaspekte und Dienste ab und sind ein heterogener Mix. Ebenso wie Mitarbeiterinnen und Mitarbeiter haben sie ihre ganz persönlichen Stärken und Schwächen, Kommunikations- und Berichtsstrukturen.

Trotz dieser überragenden technologischen, Netzwerk-unterstützten Fähigkeiten zeigen uns einige Billiglohnländer oder Anbieter mit einer riesigen Schar von Angestellten manchmal auch auf, dass Software ihre Grenzen in Bezug auf Anpassbarkeit hat. Genauso sind Startup-Unternehmen mit wenigen Mitarbeitern und kleiner Softwareunterstützung a la Microsoft Office und Windows manchmal agiler und flexibler aufgestellt. Das geschieht uns, obwohl sie nicht millionenteure Software am Laufen haben. Großbanken und Versicherungsinstitute „quälen sich“ mit hostbasiertem, undokumentiertem Softwaredurcheinander. Oft ist es über Jahrzehnte historisch gewachsen. Heute läuft es unwart-, unmigrier- und unablösbar auf den verschiedenen Prozessoren. Wer kann heute noch guten Gewissens in der COBOL-Copystrecke die Schnittstelle beschreiben? Wer weiß, welches Feld der Datenbanktabelle für welchen unsemantischen Zweck wiederverwendet wurde?

Umweltpolitische Einflüsse und Konkurrenz beeinflussen die Strategie eines Unternehmens im internationalen Markt. Um Änderungen und Anpassungen im Geschäftsprozess vorzunehmen, sind die Mitarbeiter als menschliche Ressourcen oft viel schneller in der Lage, sich an diese neue Situation anzupassen. Schneller als Software das über all die Jahrzehnte jemals gelernt hat. Eine nicht selten frustrierende Eigenschaft der heutigen Applikationsstrukturen.

Die Hochtechnologisierung von Unterehmen, die Bandbreite heterogener Softwaresysteme, das N-Schnittstellenproblem und die damit einhergehende Komplexität der Unternehmensarchitektur hat den Großteil heutiger Unternehmen in ein technisches Korsett „gequetscht“. Jeder Versuch von Agilität, jede Restrukturierung oder Migration von Softwareteilen verbrauchen scheinbar fast unendliche Aufwände. Viele Unternehmen scheitern in ihrer Entwicklung durch Aufkäufe oder Fusionen mit anderen letztendlich nicht am Widerstand der Mitarbeiter sondern immer mehr an der Nicht-Erreichbarkeit des softwaretechnischen Synergiepotentials. Das eine System will einfach nicht mit dem anderen reden, selbst wenn rein objektiv betrachtet zwei Unternehmen oder zwei Abteilungen ähnliche Geschäftsmodelle haben. Schnell scheitert es so an der exakten Ausgestaltung eines gemeinsamen Softwareprozesses.

Auswege aus dem Dilemma?

Nun verspricht natürlich Software von jeher für jedes noch so komplexe Problem auch eine Lösung zu finden - auch für sich selbst. Eigentlich ist es auch ganz einfach, denn Software muss sich „nur“ so verhalten, wie die Mitarbeiter eines Unternehmens auch. Das menschliche Vorbild überwiegt hier die Software-Designirrtümer der vergangenen Jahrzehnte. Jeder hat als Mitarbeiter und Führungskraft seinen Verantwortungsbereich, seine klar definierte Aufgabe innerhalb der Prozesskette. Er

kann durch jemanden anderen gleichen Knowhows potentiell ersetzt oder unterstützt werden. Darüber hinaus muss er – und das ist das eigentlich entscheidende – in kürzester Zeit in der Lage sein, sich neuen Herausforderungen und Aufgaben stellen zu können, ohne sich zuvor „komplett reprogrammieren“ lassen zu müssen. Software tut sich da schon schwerer.

Wir müssen also in Softwarearchitektur drei Dinge voneinander getrennt bekommen. Wir müssen klare Aufgabenbereiche definieren (nicht zu groß und monolithisch aber auch nicht zu klein und spezialisiert). Wir müssen dynamisierbare Ein- und Ausgabeschnittstellen bauen, um diese Services anzusprechen. Und schließlich müssen wir in der Lage sein, den Prozess, in dem sich diese Services integrieren lassen, also Mikro- und Makroprozessflüsse als auch die Logik innerhalb der Services, dynamisch zu beeinflussen.

Wie das geht? Nur dann, wenn wir Applikationen Verantwortungsbereiche zuordnen können. Zudem müssen sie eine gemeinsame Kommunikationsbasis (Sprache) zwischen allen beteiligten Applikationen finden und die Abfolge der fachlich notwendigen und gewünschten Aufrufe von außen steuern können (Führung). Heutzutage spielt von der Grundidee her also die Plattform (z.B. Betriebssystem, Sprache), auf der eine Software agiert, für den Unternehmenserfolg eine sehr untergeordnete Rolle. Sie fällt hinter den gemeinsamen Geschäftsprozess und die Geschäftslogik, in der sie eingebettet ist, zurück. Diese Idee gilt es zu optimieren und zu beschleunigen. Dies ist weit wichtiger als die Entscheidung, ob eine Java-Software auf Sun Solaris oder ein .NET-C#-Programm auf Windows 2003 Server zum Einsatz kommt.

Jeder stellt jedem anderen potentiell seine Dienstleistung und sein Knowhow eines bestimmten Bereiches zur Verfügung. Dies tut er, um auf abstrakter Basis einen oder n Geschäftsprozesse zu unterstützen, sowohl auf Mitarbeiter/Führungsebene als auch auf Service/Geschäftsregel-Ebene. Damit sind wir logisch gesehen, also bei der Idee eines solchen „überlebensfähigen“ und „anpassbaren“ Architekturmodells angekommen. Es besteht aus Services (inkl. Anbietern und Nutzern) und Geschäftsregeln, die den Prozess als auch Teilaspekte der Aktivitäten steuern. Eine service-orientierte Architektur (SOA) in Kombination mit ein oder n Geschäftsregel-steuernden Komponenten (z.B. Business Rule Management System) ist also nicht einfach eine fixe Idee eines neuen Hypes oder ein einfacher Marketinggag vergangener Jahre. Es handelt sich um eine logisch erklärbare Inferenz eines über die Jahre oft fehlgeleiteten Umgangs mit Software.

BRMS und Business Rules

Ein Business Rule Management System (BRMS) übernimmt also auf natürliche Weise die Funktion einer Prozessführung oder auch eines Prozessbegleiters in Fragen der Dynamisierung von Aktivitätsschritten. Gleichzeitig hilft das BRMS bei der Anwendung von Geschäftsregeln.

Wer sich schon einmal mit dem klassischen Einsatzgebieten eines BRMS auseinander gesetzt hat, wird erfahren, dass das strenge If-Then-Else-Konstrukt der meisten in einer Applikation inhärenten Softwarelogik durch ein BRMS nicht abgelöst werden soll, jedenfalls nicht in Gänze. Dies gilt, solange die somit mehr oder minder statisch monolithische Applikation einem wohldefinierten, unumstößlichen Einsatzzweck auf die gleiche Art und Weise tagein tagaus nachkommen soll. Aber für welche Applikation gilt das schon.

Nehmen wir an, Ihre Applikation verfügt z.B. über ein Kalkulationsmodul. Niemand würde annehmen, dass Preise für Ihre Produkte immer auf die gleiche Art und Weise unabhängig von Währungsschwankungen, Einkaufspreisen, Absatzhöhen, Umsatzmargen, Gewinnbeteiligungen usw. berechnet würden. Da gibt es das eine Sonderangebot hier und das Special dort. Das eine Produkt soll verkauft werden, weil es von der Kategorie so gut zu dem anderen des Käufers passt (Cross-Selling), teurere Derivate von Angeboten sollen als qualitativ hochwertigere Alternativen verkauft werden (Up-Selling) usw. Das führt schnell zu einem Wust an Geschäftsregeln, die hochdynamisch und vom Nutzer selbst beeinflusst werden sollen. BRMS Systeme agieren genau in diese Nische, indem sie Teile der Komponentenlogik auslagern. Sie bieten dem Endanwender visuelle Entscheidungshilfen und halten Test- sowie Dokumentationsmöglichkeiten bereit, um die flexiblen Geschäftslogiken und statische Komponentenlogik zu trennen. Dieser klassische Bereich der BRMS-Nutzung, in dem wir prominente Produkte wie Visual Rules der Firma Innovations aus deutschen Landen ebenso finden wie JRules der Firma Ilog oder Fair Isaac Blaze Advisor ist heutzutage selbstverständlicher Bestandteil vieler Anwendungen.

Zusammen mit dem Business Analyst oder dem Fachexperten lassen sich Logiken frühzeitig in einer Art Prototyping erarbeiten. So entstehen erste Modelle und Dokumentationen von Entscheidungen. Dies spart viel Arbeit in der Analysephase von Projekten und ersetzt sogar teilweise umfangreiche Entscheidungsmodellierung in Pflichtenheftdokumenten. Am realen Softwareobjekt sind die Implikationen getroffener Geschäftsvorgaben oft viel besser abzusehen. Visual Rules beispielsweise bietet dafür eine sehr anwendernahe, visuelle Oberfläche, in der Programmablaufplan-ähnlich Entscheidungsbäume durchlaufen, gedebuggt und getestet werden können. Dies ist sinnvoll, um fachliche Entscheidungen bzgl. Logiken zu verifizieren, lange bevor die Anwendung in Produktion geht und natürlich lange danach, wenn kurzfristig Änderungen nötig werden.

Insgesamt verspricht man sich davon Effizienzsteigerungen, einen engeren Kontakt zum Kunden, der direkt in den Softwareerstellungsprozess involviert ist und ein besseres Time-to-Market-Verhalten. Dies ist auch richtig, vorausgesetzt, man hat den Rest des Systems architektonisch auch auf die Integration eines BRMS vorbereitet.

SOA als Wegbereiter eines BRMS?

Eine SOA spielt in sich gesehen, einem BRMS eigentlich in die Hände. Das liegt daran, dass in einer SOA die vormaligen Prinzipien einer monolithischen Applikation oftmals aufgebrochen werden. Durch absichtlich eingeführte Layer- oder sogar Tiergrenzen (in Form von Prozessen) ergibt sich die zwanghafte Notwendigkeit beim Redesign einer Legacy-Anwendung hin zu einem service-orientierten Ansatz, sich verschärfte Gedanken über Konzepte und Aktivitäten zu machen. Sprich: welcher Applikationsteil hat eigentlich welche Aufgabe? Mit welchen Subjekten und Objekten beschäftigt er sich (z.B. „Abrechnungserstellung“, „Produkteinkaufsplanung“, „Finanzierungskalkulation“)? Diese Delokalisierung führt zu klarer Aufgabenverteilung und oft sogar zu Redundanzminimierung, da bei dieser primären Dekomposition Verantwortlichkeiten entsprechend dem Geschäftsprozess gebündelt werden.

Menschen neigen dazu, mit natürlichsprachlichen Konzepten ihrer Umwelt

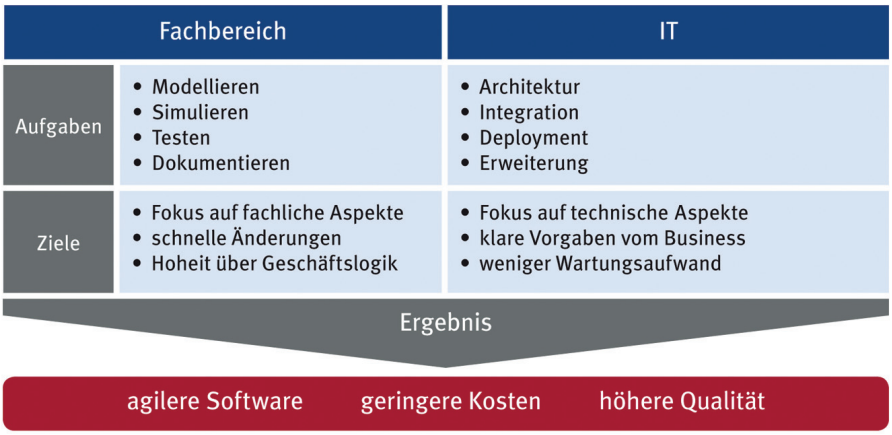

Abbildung 1: Business und IT bei der Entwicklung von Software

leichter umgehen zu können, als bei einer Aufteilung nach Begriffen wie „Präsentationsschicht", „Controllerlogik", „Datenbankpersistenz" und „XML-Transformationshelper". Es fällt Ihnen leichter, Regeln zu formulieren und mit einem Softwaredesigner zu diskutieren, anstatt sich Gedanken zu machen, wie einer Regeln in Form eines XML-Mappings oder einer JSP-Evaluierung wohl von statten gehen mag.

Eine SOA ist also ein Aufteilungsprinzip von Software, das grundsätzlich nicht besser oder schlechter ist als jede andere baumartige Aufteilung von Applikationen zur Reduktion von Komplexität. Eine SOA kommt aber der menschlichen Denk- und Logikstruktur wesentlich näher. Das liegt daran, dass sie bereits vom Namen her Dienstleistungen anbietet. Bei gewissenhaftem Design führt fast unweigerlich das zu einer Struktur, die den Unternehmensprozessen sehr nahe kommt.

Sie können mit einem Fachexperten für Abrechnung (einem Service-Owner) beispielsweise sehr gut über den Aufgabenbereich eines Finanzsystems diskutieren. Das geht allerdings nicht, wenn Sie von den Konzepten wie Rechnung, Kreditor, Debitor und Konten zu rein technischen Aufteilungen wie Datenbanken, Tabellen, Präsentationsschicht und Transportprotokollen kommen. Meist funktioniert es nur wenn Sie auf einem Abstraktionslevel bleiben, auf dem Sie beide verstehen, worum es geht und nicht bewusst oder unbewusst aneinander vorbeireden. Eine SOA führt indirekt zu einer Dekomposition der Software, die den Einsatz eines BRMS grundsätzlich oft begünstigt und eine gemeinsame Sprache zwischen Anwender und Fachexperten fördert. Sie setzt durch die Registrierung von Services, Providern und Consumern (Anbietern und Nutzern von Diensten) klare Strukturen bei der Nutzung der Architektur.

BRMS können auf diesen Services aufsetzend viele dieser Dienstleistungen komponieren, das heißt in sinnvolle sequentielle oder parallele Aktivitätsstränge überführen, um Miniaktivitäten zu Prozessketten und -schritten zu bündeln. Sie bilden leicht durch diese Aggregation Services höherer Abstraktionsebene (grobgranulare Schnittstellen). Zudem sind sie damit oft Bestandteil oder sogar Ausgangsprodukt eines Business Process Management (BPM) Systems.

Zwei Seiten einer Medaille : Fachlichkeit und technische Architektur

Eigentlich sieht es im Projektalltag oft so aus, als seien Business und IT nicht zwei Seiten einer Medaille zur Bildung einer erfolgreichen Software sondern als handele es sich um zwei Welten. Meist erscheint es so, dass der eine den anderen nicht versteht oder nicht verstehen kann. Der Business-Teil, der Kunde Ihrer Software-Dienstleistung, kommt mit unklaren, unstrukturierten Informationen und Regeln daher. Der „IT-Mensch" versucht die Anforderungen durch die Brille der technischen Realisierung von Frameworks und Produkten zu sehen. Dabei können beide zusammenkommen, wenn der Business-Experte die Hoheit über die Geschäftslogik zurückerlangt und der IT-Berater sich auf Architektur und technisches Design konzentrieren kann, ohne jedes fachliche Detail verinnerlicht zu haben.

Hier führen BRM-Systeme wie Visual Rules eine Trennlinie ein, die jeden auf seiner Seite auf sein Expertenwissen konzentrieren lässt, um weniger vom Knowhow des Anderen erweben zu müssen und sich dennoch zu verständigen.

Die Mächtigkeit eines BRMS zeigt sich auch beim Blick auf die typischen Phasen des Entwicklungszykluses. Ein gutes BRMS hilft bereits bei der Modellierung von Entitäten und Services und unterstützt beim Testen und Tracing, also der Prüfung der Anwendung der Regeln. Es stellt Mechanismen bereit, Regelartefakte als Jar-Dateien, XMLs, Regelbäume, Datenbanktabellen usw. jederzeit auch ohne explizit neue Anwendungsversion zu deployen und adhoc zu ändern. Gleichzeitig entsteht in diesem Prozess auch ein erhöhtes Qualitätsbewusstsein des Kunden gegenüber dem fertigen Softwareprodukt. Das liegt daran, dass er direkten Einfluss auf die Logik des Systems hat und damit einen Großteil Verantwortung und Mitbestimmung übernimmt. Er kann oft auch außerhalb der laufenden Applikation bereits seine Regeln testen und ist frühzeitig und ebenfalls aktiv für die Qualität der entstehenden Dokumentation mitverantwortlich. Ihm zur Seite steht ein Business Analyst, oftmals eine Person mit weniger technischem als mehr analytischem Wissen, der in der Lage ist, die vielfältigen Wünsche und Anforderungen mit dem Kunden gemeinsam zu strukturieren und im BRMS zu visualisieren. Hierbei wird implizit bei der Erarbeitung der Regelabläufe bereits ein Großteil der komplexen Anwendungsstellen fertig nutzbar implementiert. Der Bedarf für fachliches Knowhow auf Entwicklerseite und das Risiko von Missverständnissen und Fehlinterpretationen bei der Realisierung der Software sinken. Dies ist ein essentielles Kriterium nicht nur für qualitativ bessere Software

Abbildung 2: Business Rules im Produktionsprozess

sondern auch für deutliche Reduktion der Entwicklungskosten.

Die Aufteilung einer Software hat oftmals wiederum auch Implikationen für den Umgang mit ihr oder die Organisationsstrukturen, in denen Mitarbeiter sich gegenseitig Dienstleistungen anbieten. Zuweilen hilft eine SOA-Strukturierung auch, Zuständigkeiten auf fachlicher Seite klarer herauszuarbeiten, als sie auf rein zwischenmenschlicher Ebene spontan evident wären. Den Regeleinheiten/Regelsätzen sind bedingt durch die Einbettung in und bei Komponenten klare Zuständigkeiten (Service-Ownerships) auf fachlicher Seite zuzuordnen. Dies hilft oft auch bei der fachlichen Aufgabenverteilung im Business-Bereich.

Grau ist alle Theorie?

Um Regeln nunmehr in eine SOA zu integrieren, macht es oft Sinn, sie nicht in der Kapselung eines Services zu verstecken sondern aktiv als eigenen Service in die SOA zu integrieren. Dies geschieht – wenn auch nicht zwanghaft – heute gemeinhin in Form eines Web Services. So können sie in den Enterprise Application Integration (EAI) bzw. Enterprise Service Bus (ESB)-Systemen dieser Welt als eigener Dienstleiter in die Prozesskomposition eingebunden werden. Auch eine dezentrale Bereitstellung der Business Rules ist überdies denkbar als Registrierung in einem Service Repository wie z.B. in Visual Rules.

Nun ist die Theorie von SOA und BRM grau, solange nicht Real-World-Projekte die Nutzbarkeit dieses Ansatzes auch demonstriert haben. Das Softwarehaus SunGard erarbeitete letzthin zusammen mit Innovations, dem Hersteller von Visual Rules, ein Konzept zur Migration eines Komponententeils zur Fondsbuchhaltung bei einem großen Finanzdienstleister. Die Migration bedeutete eine Portierung der monolithischen Struktur auf ein von SunGard empfohlene Common Services Architecture Initiative (CSA). Dies ist eine kollaborative Open Source Arbeitsplattform für Service Orientierte Architekturen. Im Endeffekt weist man auf Grund der engen Zusammenarbeit mit Fachexperten eine um 90% kürzere Entwicklungszeit aus. Statt der Standardimplementierung von 15 Tagen, verlief der SOA-Ansatz bereits in 1,5 Tagen mit Erfolg, berichtet man dort.

Nun gerät man bei solchen Zahlen leicht ins Schwärmen und in die Gefahr Äpfel mit Birnen zu vergleichen. Die Restrukturierung monolithischer Legacy-Systeme zu einer SOA mit BRMS führt nicht wie bei einem Kochrezept automatisch immer zu 90% höherer Entwicklungsperformance. So leicht wollen wir uns die Abstraktion nicht machen. Schließlich spielen da Einarbeitung in die Systeme, Knowhowtransfer und Schulung neuer Softwareprodukte am Anfang auch eine Rolle. Zudem beschleunigen schon unterschiedliche Programmiersprachen an sich, Entwicklungszyklen oft dramatisch. Sicherlich ist aber richtig, dass durch die Verschlankung des Entwicklungsprozesses, die klarere Aufgabentrennung zwischen IT und Business und das Überdenken der eigenen Designansätze sehr schnell effektive Ansätze für eine moderne, flexiblere Architektur gefunden werden können. Bessere Lösungen als sie oftmals bislang vorherrschen.

Viele weitere Fragen der Performance, des IT-Knowhows oder Configuration Managements sind ebenfalls beim Umstieg auf eine neue Architekturrichtlinie zu beantworten. Aber jeder Schritt in Richtung mehr Flexibilität, mehr Qualität und höhere Effizienz bereitet den Boden für eine höhere Wettbewerbsfähigkeit. SOA und BRMS sind zwei Lösungen, die hier häufig heutzutage genannt werden, sicherlich nicht ohne Grund. ■

Literatur

Enterprise SOA. Service Oriented Architecture Best Practices, Dirk Krafzig, Prentice Hall PTR, ISBN: 0131465759

Service-Oriented Architecture. Concepts, Technology, and Design, Thomas Erl, Prentice Hall PTR, ISBN: 0131858580

Enterprise Service Bus. Theory in Practice, David A. Chappell, O'Reilly; ISBN: 05960 06756

Java Rules Engines. Entwicklung von regelbasierten Systemen, Lars Wunderlich, entwickler.press, ISBN: 3935042752