Integration von prozessorientierten Anwendungen

Thomas Bauer

DaimlerChrysler Research and Technology
Abt. Data and Process Management
Postfach 2360, D-89013 Ulm
Thomas.TB.Bauer@DaimlerChrysler.com

Zusammenfassung. Existierende Altanwendungen enthalten oft explizites oder implizites Prozesswissen. Ziel muss es sein, dieses bei ihrer Integration in eine unternehmensweite Anwendung verfügbar zu machen und zu nutzen. Im vorliegenden Beitrag wird untersucht, welche Ansatzpunkte und Anforderungen es hierfür gibt. Von diesen ausgehend wird eine entsprechende systemneutrale Architektur und eine exemplarische Lösung für IBM WebSphere Business Integration (WBI) vorgestellt.

1 Einleitung

Ein wichtiger Aspekt der Enterprise Application Integration (EAI) sind Prozesse. Diese werden meist verwendet, um existierende Anwendungen zu integrieren (siehe z.B. [2]). Eine in der wissenschaftlichen Literatur bisher kaum betrachtete Tatsache ist, dass die zu integrierenden Altanwendungen häufig selbst Prozesslogik enthalten. Ziel muss es sein, dieses Prozesswissen explizit zu machen, um es in neu zu erstellenden (übergeordneten) Applikationen nutzbar zu machen. Das bedeutet, die Altanwendung soll nicht mehr nur als „Black Box“ eingebunden werden.

Dies ist insbesondere deshalb eine Herausforderung, weil existierende Legacy Applications normalerweise nicht wohl-strukturiert z.B. auf Basis von Workflow-Management-Systemen (WfMS) [7] realisiert sind. Außerdem verbieten sich deren Neuimplementierung oder umfangreiche Anpassungen häufig aufgrund ihrer komplexen Geschäftslogik und aus Kostengründen. So ist der einzig verbleibende Integrationsweg, die Altanwendungen mittels eines Adapters an die neuen Applikationen anzubinden. Dieser Adapter muss natürlich Zugriff auf die benötigten Daten und Ereignisse aus der Legacy Application haben [8]. Bei Eigenentwicklungen sind zur Adapteranbindung auch geringfügige Erweiterungen der Altanwendung vorstellbar (z.B. die Protokollierung relevanter Ereignisse in einer Datenbanktabelle, auf die dann vom Adapter zugegriffen wird).

Im Abschnitt 2 werden Anforderungen an die Integration von prozessorientierten Altapplikationen vorgestellt. In Abschnitt 3 wird eine generische Integrationsarchitektur vorgestellt und exemplarisch erläutert, wie diese für eine bestimmte Systemlandschaft (WBI) realisiert werden kann. Nach einer Diskussion verwandter Ansätze in Abschnitt 4 schließt dieser Beitrag mit Zusammenfassung und Ausblick.

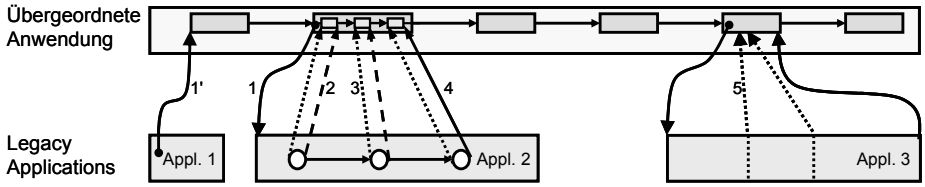

Abb. 1. Interaktionsformen zwischen Altanwendungen und der integrierenden Applikation

2 Anforderungen an die Integration von Prozessanwendungen

Im Rahmen eines Forschungsprojektes haben wir gemeinsam mit DaimlerChrysler-Unternehmensbereichen Anforderungen an eine Integration von prozessorientierten Altanwendungen gesammelt. Besonders relevant sind die funktionalen Anforderungen, da aus ihnen die notwendigen Interaktionsformen zwischen den Applikationen abgeleitet werden können. Wie in Abb. 1 dargestellt, müssen folgende Aktionen unterstützt werden:

1. Starten einer Prozessinstanz der Legacy Application durch die übergeordnete Anwendung. Hierbei müssen außer den Eingabeparametern des Prozesses auch noch der Prozesstyp und weitere Metainformationen übergeben werden. In Sonderfällen kann ein Prozess der übergeordneten Anwendung auch durch eine Aktion einer Legacy Application gestartet werden (Aktion 1' in Abb. 1).
2. Die Beendigung einer Aktivität einer Legacy Application wird an die zugehörige übergeordnete Prozessinstanz signalisiert. Diese enthält sog. „Schattenaktivitäten“, die nicht aktiv ausgeführt werden, sondern nur den aktuellen Zustand und die Ergebnisdaten der korrespondierenden externen Aktivitäten widerspiegeln. Eine solche Schattenaktivität kann in der übergeordneten Applikation z.B. als Ausgangsknoten von Kontroll- und Datenflusskanten verwendet werden. Aktivitäten können dadurch die Daten der Legacy Application nutzen, bevor diese komplett beendet ist. Die Daten von Schattenaktivitäten können außerdem für Monitoring, Prozessvisualisierung [3] und Ähnlichem genutzt werden.
3. Das Starten einer Aktivität der Legacy Application wird ebenfalls an die Schattenaktivität übermittelt. Diese Aktion kann entfallen, wenn sie in der entsprechenden Anwendung nicht erkannt werden kann (z.B. für Operationen wie „Dokument erstellen“). In diesem Fall wird der Start gemeinsam mit der korrespondierenden Aktivitätenbeendigung (Aktion 2) signalisiert.
4. Die Beendigung eines Applikationsprozesses ist für die übergeordnete Anwendung prinzipiell auch relevant. Allerdings fällt diese Aktion normalerweise mit der Beendigung seiner letzten Aktivität zusammen, so dass eigentlich eine Aktion vom Typ 2 ausgeführt wird.
5. Es gibt auch Applikationsprozesse, die nicht wirklich separate Aktivitäten durchlaufen. Dennoch kann Information über den Ausführungszustand des Prozesses anfallen, wie ein Reifegrad eines Bauteils oder ein Abarbeitungsgrad (in %). Solche Information ist für das Monitoring des übergeordneten Prozesses ebenfalls sehr relevant. Deshalb muss sie übertragen und in der jeweiligen Aktivität gespei-

chert werden. Hierbei ist essentiell, dass die Statusinformation nicht erst bei Beendigung der übergeordneten Aktivität gespeichert wird (weil die Information einen aktuellen Zwischenzustand beschreibt) und auch mehrfach aktualisiert werden kann.

Des Weiteren gibt es auch nicht-funktionale Anforderungen, die primär die Kosten und den Betrieb des Gesamtsystems betreffen:

- Für die Entwicklungskosten ist entscheidend, dass nicht (mehrfach) projektspezifische Lösungen realisiert werden. Stattdessen wird einmalig eine Infrastruktur zur Applikationsintegration entwickelt, mit der dann beliebige Altanwendungen integriert werden können.
- Um in einem Rechenzentrum einen stabilen Betrieb mit akzeptablem Aufwand garantieren zu können, darf die Infrastruktur nur ohnehin eingesetzte Serverkomponenten enthalten. Es ist nicht möglich, die Verfügbarkeit einer Applikation zu gewährleisten, die nach einem Absturz manuell gestartet werden muss. Deshalb sind z.B. Programme mit Callbacks, die von Legacy Applications gerufen werden, ausgeschlossen. Stattdessen sollten diese über Nachrichten kommunizieren, welche dann automatisch die Ausführung von Programmcode (z.B. message-driven Enterprise Java Beans) auslösen.
- Um Lizenzen und Know-how effizient nutzen zu können, muss sich die Infrastruktur auf Komponenten abstützen, die in dem jeweiligen Unternehmen Standard sind. Im vorliegenden Fall ist dies IBM WebSphere Business Integration (WBI). Außerdem muss natürlich eine geeignete Betriebssystem-Plattform ausgewählt werden.

3 Konzept und Architektur

Da IBM WBI [1, 5, 6] als Basis verwendet werden soll, wird MQ Workflow für die Realisierung von neuen Prozessanwendungen verwendet, und damit für die Implementierung des übergeordneten Prozesses (siehe Abb. 2). Zur sicheren Kommunikation dient in diesem Umfeld der Messaging-Dienst WebSphere MQ, mit dem XML-Nachrichten unter Transaktionsschutz ausgetauscht werden können. Er wird ergänzt durch den WBI Message Broker, mit welchem (automatisch ausgeführte) Microflows definiert werden können, deren Knoten u.a. folgende Operationen ermöglichen:

- Transformation eines Eingabe- in ein Ausgabedatenformat (graphisch definierbar)
- Zugriff auf Datenbanken mittels ESQL (extended SQL)
- Verzweigungen (exklusiv aufgrund von Bedingungen oder parallel)
- (inhaltsbasiertes) Routing von Messages
- Publish-/Subscribe-Mechanismus zur Registrierung für bestimmte Ereignisse

Die Ein- und Ausgabe eines solchen Microflows erfolgt stets mittels einer Message-Queue. Über eine solche kommuniziert der Message-Broker mit Adaptern. Deren Aufgabe ist es, Nachrichten entgegenzunehmen und die entsprechende Information über einen beliebigen Mechanismus (z.B. API oder Datenbankzugriff) an eine Legacy Application zu übergeben bzw. dort eine entsprechende Funktion aufzurufen. Außerdem erkennt der Adapter Ereignisse, die in der Legacy Application eingetreten sind,

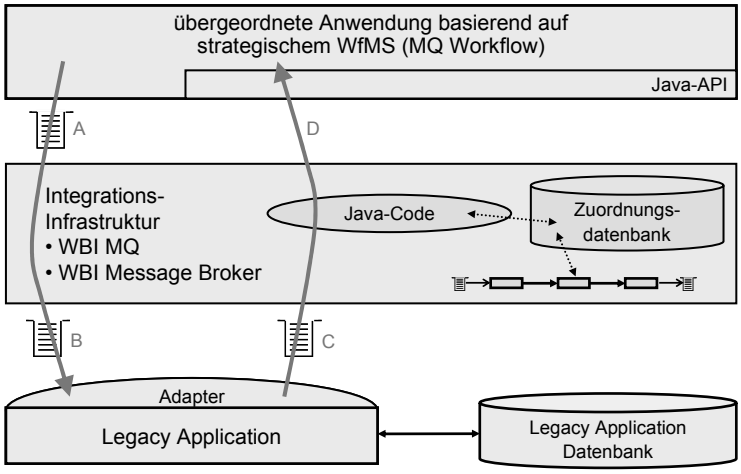

Abb. 2. Einbettung der Infrastruktur

greift auf die zugehörige Information zu, und übergibt eine entsprechende Nachricht an die Eingangs-Queue des Message Brokers.

Der eigentliche Kern der Integrations-Infrastruktur (vgl. Abb. 2) wird durch die Abläufe des Message-Brokers, Java-Code und eine Datenbank gebildet. In letzterer wird z.B. die zur Modellierungszeit spezifizierte Zuordnung einer Aktivität einer Legacy Application zu einer bestimmten Aktivität eines Workflow-Typs von MQ Workflow gespeichert. Sie verwaltet aber auch Run-Time-Daten, wie die Zuordnung von konkreten Legacy-Prozessinstanzen zu Workflow-Instanzen. Eine solche Zuordnung kann je nach Gegebenheit sowohl durch interne IDs (z.B. ID einer MQ Workflow Instanz), wie auch durch die Speicherung von Anwendungsdaten (z.B. Auftragsnummer) erfolgen.

Mit den prozessorientierten Altanwendungen und dem neuen Anwendungssystem kommuniziert die Infrastruktur über eine (intern) standardisierte Nachrichten-Schnittstelle. Dadurch wird es möglich, Adapter wiederzuverwenden und Altanwendungen schnell in unterschiedliche Prozesse zu integrieren. Die in Abb. 1 dargestellten Aktionen führen bei ihrer Ausführung in der Infrastruktur zu folgenden Operationen (mit jeweils korrespondierender Nummer):

1. Beim Starten eines Anwendungsfalls in der Altapplikation wird aus der bei der Infrastruktur eingehenden Nachricht (A in Abb. 2) ermittelt, welcher Prozesstyp in welcher Applikation gestartet werden soll. Aus der Zuordnungsdatenbank werden die entsprechende Funktion und der Adapter ermittelt, und mit den entsprechenden Eingabedaten (die zuvor ggf. vom Message Broker in das Zielformat transformiert wurden) aufgerufen (B). Das vom Adapter zurückgegebene Funktionsergebnis beinhaltet die Prozessinstanz-ID in der Legacy Application, die dann gemeinsam mit der ID aus MQ Workflow in der Zuordnungsdatenbank gespeichert wird. Analog wird verfahren, falls die Legacy Application einen übergeordneten Workflow auslöst (Aktion 1' in Abb. 1).

2. Die Beendigung einer Aktivität der Anwendung wird vom Adapter automatisch erkannt (ggf. durch Polling) und per Nachricht (C in Abb. 2) an die Infrastruktur übermittelt. Diese ermittelt aus den Modellierungsdaten der Zuordnungsdatenbank den betroffenen Workflow-Typ und die Aktivität ebenso wie die zur Ausführungszeit (durch Operation 1) hinterlegte Workflow-Instanz. Die hierdurch referenzierte MQ Workflow Aktivitäteninstanz wird dann beendet und die (evtl. transformierten) Ergebnisdaten werden übergeben (D).

   Da die Messaging-Schnittstelle von MQ Workflow die hierfür notwendige API-Funktion „CheckIn" [6] nicht unterstützt (ebenso wie die nachfolgend benötigten), muss das Java-API verwendet werden. Deshalb wird es erforderlich, in der Infrastrukturschicht Java-Code auszuführen. Wie in Abschnitt 2 beschrieben, ist es nicht möglich ist, Callbacks (z.B. per RMI) eines laufenden Java-Programms zu rufen. Deshalb werden message-driven Enterprise Java Beans verwendet, die im Falle einer eingehenden Nachricht automatisch gerufen werden und an welche die Eingabedaten übergeben werden. Diese EJB kann dann die Methode „CheckIn" des MQ Workflow Java-APIs aufrufen und die Aktivität beenden. (Eine Alternative, die ohne Application Server auskommt, ist die Verwendung einer sog. Trigger-Queue von WebSphere MQ [5].)
3. Der Start einer Aktivität der Legacy Application wird analog zur Aktion 2 erkannt und mittels der API-Funktion „CheckOut" an MQ Workflow signalisiert. Der einzige Unterschied ist, dass beim Starten einer Aktivität noch keine Ergebnisdaten entstanden sind, und deshalb nur Metainformationen (z.B. Bearbeiter, Startzeit) an MQ Workflow übermittelt werden.
4. Die Beendigung eines Legacy Application Prozesses wird bei einer Aktion vom Typ 2 mit entsprechend gesetztem Attribut an die Infrastruktur „mit signalisiert". Diese hat nun prinzipiell die Möglichkeit, den entsprechenden Zuordnungseintrag aus der Datenbank zu löschen. Hiervon wird aber abgesehen, da der Platzbedarf für eine solche Prozessinstanz-Zuordnung sehr klein ist, und der Eintrag weiterhin der Nachvollziehbarkeit und Dokumentation dient.
5. Zur Speicherung von Statusinformation einer Aktivität werden Anwendungsvariablen der übergeordneten Applikation verwendet. Diese werden hierfür reserviert und in der Zuordnungsdatenbank wird gespeichert, welche Information in welcher Variablen gespeichert wird. Signalisiert eine Legacy Application nun die Änderung eines solchen Statuswerts (C in Abb. 2), wird wieder die zugehörige Aktivitäteninstanz ermittelt (ggf. gestartet) und die Variable wird geschrieben (D).

   Problematisch ist hierbei für die konkrete Realisierung, dass MQ Workflow eigentlich nur vorsieht, Ergebnisdaten bei der Beendigung der zugehörigen Aktivität zu schreiben [6]. Da dies für einen solchen Zwischenstatus aber keinen Sinn macht, wurde im Java-Code folgendes Vorgehen implementiert: Wie in Abb. 3 dargestellt, befindet sich eine eigentlich in der Legacy Application ausgeführte MQ Workflow Aktivität im Zustand „CheckedOut". Mit der Operation „Force-Restart" ist es nun möglich, den neuen Status als Parameter in den InContainer[1] der

[1] Jede MQ Workflow Aktivität liest ihre Eingabedaten aus einem InContainer und schreibt ihr Ergebnis in einen OutContainer. Dieser kann auf die InContainer von Nachfolgeaktivitäten abgebildet werden. Über API-Funktionen [6] ist es aber jederzeit möglich, auch die Daten

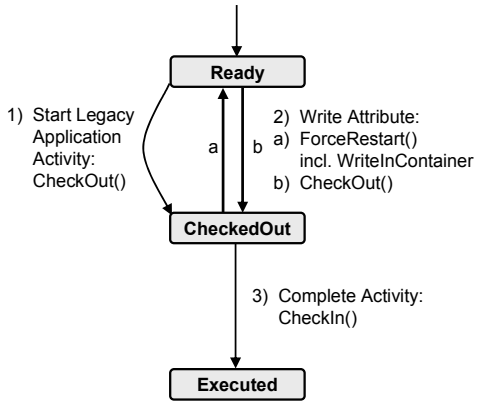

Abb. 3. Übergänge des Aktivitätenstatus bei Operation 5 (vgl. [6])

Aktivität zu schreiben. Dadurch geht der Zustand in „Ready" über. Dies macht der Java-Code durch Aufruf der API-Funktion „CheckOut" rückgängig, so dass sich die Aktivität wieder im korrekten Zustand befindet.[2] Im InContainer befindet sich dann die aktuelle Statusinformation, die durch dieses Vorgehen mehrfach aktualisiert werden kann. Beendet wird die Aktivität wie üblich mittels CheckIn.

Eine Alternative zum soeben beschriebenen Vorgehen bietet MQ Workflow durch die Definition einer Exit-Condition für die betroffene Aktivität: Beim „CheckIn" wird der Zwischenstatus in den OutContainer geschrieben und die Exit-Condition auf False gesetzt. Dadurch wird die Aktivität zurückgesetzt und kann erneut gestartet werden. Damit die Daten aus dem OutContainer hierbei allerdings nicht verloren gehen, müssen sie mittels eines Loop-Connectors in den InContainer derselben Aktivität „zurückgeschleift" werden. Da diese zusätzlich notwendige Datenflusskante und die Exit-Condition das Prozessmodell deutlich unübersichtlicher machen, haben wir uns für die zuvor beschriebene Variante entschieden. Bei dieser ist die gesamten Algorithmik in dem einmalig erstellen, anwendungsunabhängigen Java-Code der Integrations-Infrastruktur konzentriert.

Beide Ansätze haben den Nachteil, dass ein Neustart der Aktivität im Log-File des WfMS protokolliert wird, was irrtümlicherweise als Problem- oder Fehlersituation interpretiert werden könnte. Idealerweise würde das verwendete WfMS das Schreiben von Anwendungsdaten auch dann erlauben, wenn die zugehörige Aktivität noch läuft. Dann wäre es problemlos möglich, (Zwischen-)Zustandsdaten im

---

eines InContainers zu lesen. Deshalb sind Daten aus dem InContainer sowohl für Nachfolgeaktivitäten wie auch für externe Programme (z.B. für Monitoring) voll nutzbar.

[2] Die API-Funktion „CheckOut" löst (im Gegensatz zu „Start") nicht tatsächlich das Starten einer Anwendung aus. Deshalb hat ihr (mehrfacher) Aufruf auch keine negativen Seiteneffekte. Es ist sogar schon beim ersten „CheckOut" (in Operation 3 aus Abb. 1) der Fall, dass die Aktivität in der Legacy Application bereits läuft (deren Start hat die Aktion schließlich angestoßen).

WfMS abzulegen. Da dies bei MQ Workflow aber nicht gegeben ist, wird der beschriebene „Work-around“ gewählt.

Für das Monitoring und die Visualisierung von Prozessen [3] ist es notwendig, auch Zugriff auf Anwendungsdaten der Applikation zu haben, und nicht nur auf die Prozess-Zustandsinformation. Zu diesem Zweck werden in Operation 2 Ergebnisdaten an die übergeordnete Prozessanwendung übergeben. Um diese Workflow-Applikation hierbei aber nicht zu überlasten, muss unterschieden werden, für welche Daten dies tatsächlich sinnvoll ist: Workflow-relevante Daten [10] müssen selbstverständlich übergeben werden, da sie für die Steuerung des Kontrollflusses vom WfMS benötigt werden. Kleinere Anwendungsdaten-Objekte können ebenfalls an das WfMS übergeben werden, wobei beachtet werden muss, dass bei einigen WfMS (u.a. auch MQ Workflow) die Anzahl der Elemente eines Daten-Containers beschränkt ist. Große Datenobjekte (z.B. CAD-Modelle, digitalisierte Bilder, Videosequenzen) sollten ausschließlich im Anwendungssystem gehalten werden, so dass dem WfMS lediglich ein Suchschlüssel übergeben wird. Zum Zwecke einer Prozessvisualisierung kann mit diesem dann auf die Anwendungsdaten zugegriffen werden.

4 Verwandte Arbeiten

In [8] wird dargelegt, dass es aktuell in der wissenschaftlichen Literatur kaum Ansätze gibt, die im Zuge einer Integration von Legacy Applications versuchen, deren Prozesswissen zu erschließen. Zwar wird in [4] eine prozessorientierte Kopplung realisiert, es werden aber keine Altanwendungen integriert. Stattdessen wird angenommen, dass die zu integrierende Anwendung selbst auf einem WfMS basiert. Dessen Kopplung wird notwendig, weil es zur Steuerung der übergeordneten Anwendung ungeeignet ist (z.B. da es sich um ein internes WfMS eines Produktdaten-Management-Systems handelt).

Weitere Ansätze, die noch entfernter mit der vorliegenden Fragestellung zu tun haben, werden in [8] ausführlich diskutiert. So untersucht z.B. [9], wie auf Komponententechnologie basierende Anwendungen in übergeordnete Workflows integriert werden können. Da allerdings nicht angenommen werden kann, dass jede Legacy Application diese Eigenschaft erfüllt, kann von diesem Ansatz im Allgemeinen nicht ausgegangen werden.

5 Zusammenfassung und Ausblick

In dem Beitrag wurde untersucht, wie eine prozessorientierte Legacy Application unter Erhaltung des Prozesswissens in eine Workflow-Anwendung integriert werden kann. Hierbei konnten alle identifizierten Anforderungen erfüllt werden, ohne bzgl. der Legacy Application irgendwelche Einschränkungen zu machen. Allerdings muss der Adapter, über den die Altanwendung angebunden ist, natürlich Zugriff auf diejenigen Daten und Ereignisse haben, die nach außen sichtbar gemacht werden sollen. Das vorgestellte Konzept ist insgesamt allgemeingültig, auch wenn der Ansatz für die

IBM-Produkte WebSphere MQ, WBI Message Broker und MQ Workflow detailliert und prototypisch implementiert wurde. Produkte mit ähnlicher Funktionalität sind aber auch von anderen Herstellern erhältlich.

Eine noch weitergehende Integration von Legacy Applications lässt sich erreichen, wenn zusätzlich zu deren Prozessmodell und Ausführungszustand auch noch weitere Modellinformationen integriert werden. Hierfür würde sich insbesondere das Organisationsmodell anbieten, da häufig sowohl Altanwendungen wie auch WfMS über ein solches verfügen. Werden diese (partiell) integriert, so reduziert sich der Aufwand für deren Pflege. Außerdem kann die Funktionalität der Prozesssteuerung erhöht werden, weil es durchaus die Anforderung gibt, dass die Bearbeiter einer Workflow-Aktivität von denen einer Legacy Application abhängig sind (z.B. selber Bearbeiter, Vorgesetzter, 4-Augen-Prinzip). Die Frage ist nun, wie solche Abhängigkeiten definiert und mit einer Ausführungssemantik versehen werden können, wenn die involvierten Systeme strukturell und inhaltlich unterschiedliche Organisationsmodelle aufweisen.

Denkt man die Idee einer Integrations-Infrastruktur noch weiter, so sollte sie nicht nur erlauben, Altanwendungen mit solchen prozessorientierten Anwendungssystemen zu integrieren, die auf einem WfMS basieren. Es sollte der Informationsaustausch zwischen beliebigen Anwendungssystemen unterstützt werden, auch zwischen zwei Legacy Applications. Dann übernimmt die Integrations-Infrastruktur die Rolle eines zentralen Verteilers, bei dem sich Anwendungen für gewisse Events registrieren können, der zur Run-Time die Zuordnung von Prozessinstanzen verwaltet, Informationen an (evtl. mehrere) Adapter und Zielsysteme weitergibt und dort ggf. die jeweils spezifizierten (ggf. unterschiedlichen) Operationen durchführt.

Literatur

1. O. Anspacher: Integration von heterogenen Anwendungen in Workflow-Management-Systeme. Diplomarbeit Fachhochschulen Ulm/Neu-Ulm. (2005)
2. T. Bauer: Integration heterogener Applikationstypen durch Workflow-Management-Technologie. In: Proc. Workshop on Enterprise Application Integration, Oldenburg. (2004) 39-47
3. T. Bauer: Visualisierung laufender Prozesse. In: Proc. Informatik 2004, Workshop Geschäftsprozessorientierte Architekturen, Ulm. (2004) 543-548
4. T. Bauer, T. Beuter, M. Weitner, T. Bär: Integration of Engineering and Production Planning Workflows. Journal of Advanced Manufacturing Systems 4(1). (2005) 37-52
5. IBM: MQSeries Application Programming Guide. (2000)
6. IBM: WebSphere MQ Workflow Programming Guide. Version 3.5. (2004)
7. S. Jablonski, M. Böhm, W. Schulze: Workflow-Management: Entwicklung von Anwendungen und Systemen. dpunkt-Verlag. (1997)
8. R. Pryss: Enterprise Application Integration: Anforderungen, Ansätze und Technologien. Diplomarbeit Universität Ulm. (2005)
9. S. Schreyjak: Coupling of Workflow and Component-Oriented Systems. Lecture Notes in Computer Science, 1357. (1998)
10. Workflow Management Coalition: Terminology & Glossary. Document Number WFMC-TC-1011. (1999)