

Projekt MINT – Modellgetriebene Integration von betrieblichen Informationssystemen

Ralf Reussner, Ulrike Steffens, Niels Streekmann

OFFIS

Bereich Betriebliches Informationsmanagement

Escherweg 2

26121 Oldenburg

Kurzfassung

Modellgetriebene Entwicklungsverfahren zur Integration bestehender heterogener betrieblicher Informationssysteme können insbesondere für KMU eine wesentliche Unterstützung darstellen, da sie den flexiblen Umgang mit sich rasch ändernden Anforderungen vor dem Hintergrund gewachsener Systemlandschaften erlauben.

Im vorliegenden Beitrag wird das Projekt MINT beschrieben, in dem die andrena objects AG, die BTC AG, die Delta Software Technology GmbH, das OFFIS und die Universität Oldenburg gemeinsam den Ansatz modellgetriebener Entwicklung auf Integrationsszenarien anwenden. Musterbasierte, domänenspezifische Architektursprachen liefern dabei eine zusätzliche Unterstützung insbesondere bei der Nutzung des modellbasierten Ansatzes für die Integration verschiedener Systeme durch bestehende Infrastrukturen, wie sie z.B. durch Standardsoftware vorgegeben sind, sowie bei der Nutzung des modellbasierten Ansatzes für die Kopplung moderner objektorientiert modellierter Geschäftslogik mit bestehenden relationalen Datenbanksystemen.

1. Integration betrieblicher Software

Eine wichtige Eigenschaft heutiger betrieblicher Software ist es, schnell und kosteneffizient an neue oder sich ändernde Geschäftsprozesse anpassbar zu sein. Oft werden aber bestehende Systeme Anforderungen ausgesetzt, für die sie ursprünglich nicht entworfen wurden, wie zum Beispiel Web-Anbindungen im Bereich des sog. Business-to-business-eCommerce. Dies zeigt sich zum einen in über die Zeit gewachsenen Softwaresystemen, die die beschriebene Flexibilität nicht erreichen, und zum anderen in heterogenen „Softwaresystemlandschaften“, deren Datenhaltung und Arbeitsabläufe nicht integriert sind, obwohl dieses für die Geschäftsprozesse des Unternehmens vorteilhaft wäre. Gerade KMU stellt die Integration bestehender Software mit neuen Anwendungen vor oft kaum zu bewältigende finanzielle und technische Herausforderungen. Die Teilnahme der KMU an endkundenbezogenen oder unternehmensübergreifenden eCommerce-Abläufen wird damit verhindert, wodurch sich offensichtliche Wettbewerbsnachteile ergeben.

Wie in [1] ausgeführt wird, findet ein Großteil der deutschen Softwareentwicklung im Auftrag von hochgradig spezialisierten mittelständischen Unternehmen der sogenannten Sekundärindustrie statt. Die für diese Unternehmen benötigte Software kann i.d.R. nur durch eine enge

Zusammenarbeit zwischen Kunde und IT-Unternehmen entwickelt werden. Neben Maßnahmen zur Steigerung der Effizienz bei der Entwicklung neuer Softwaresysteme ist die effiziente Anpassbarkeit und Integration bestehender Software von hoher Bedeutung. Da in die Entwicklung bereits bestehender Softwaresysteme ein beträchtlicher finanzieller Aufwand geflossen ist und ein für die Geschäftsidee essenzielles Know-how implizit in dieser Software steckt, ist ihre Abschaltung unmöglich. Stattdessen sind Verfahren von Interesse, die eine schrittweise Migration einer solchen unternehmenskritischen Software zu einer neuen und flexiblen Softwarearchitektur erlauben.

2. Modellgetriebene Ansätze zur Integration

Die modellgetriebene Entwicklung versucht, der Forderung nach Flexibilität durch eine Intensivierung der Wiederverwendung Rechnung zu tragen [2]. Dies soll erreicht werden durch den Einsatz von Generortechnologie, wie er beispielsweise für den speziellen Bereich der Software-Produktlinien im BMBF-geförderten Projekt PESOA (Process Family Engineering in Service-Oriented Architectures) bereits erforscht wird [3]. Ergänzt wird dieser generative Ansatz durch die konsequente Trennung zwischen Modellierung der Architektur mit der fachlichen Logik einerseits und dem konkreteren Verhalten andererseits. Die fachliche Logik wird dabei durch das abstrakte, auf die Beschreibung der Anwendungsdomäne zielende Computation Independent Model (CIM) beschrieben. Das konkretere Verhalten wird durch das Platform Independent Model (PIM) repräsentiert, das bereits Problemlösungsverfahren enthält, jedoch noch keinerlei plattformspezifische Eigenschaften umfasst. Diese werden durch einen Generator und seine Konfiguration verborgen, mit dessen Hilfe sich ablauffähige sog. Platform Specific Models (PSM) erstellen lassen. Die PSM können dann anschließend in Code transformiert werden. Bei der modellgetriebenen Entwicklung wird dem Softwareentwickler somit ein Vorgehensmodell an die Hand gegeben, welches beschreibt, wie (teilweise unter Anwendung von Transformationsregeln) aus den Zielvorgaben eines Kunden ein CIM und dann über ein PIM sowie verschiedene PSM schließlich fertige Software erstellt werden kann. Allerdings werden im Bereich der Softwareentwicklung für betriebliche Informationssysteme die antizipierten Vorteile der modellgetriebenen Entwicklung heute noch nicht erreicht, da (a) sich die meisten bestehenden modellgetriebenen Ansätze auf die Neuentwicklung von Softwaresystemen konzentrieren und die gerade bei betrieblichen Softwareanwendungen wichtige Problematik der Evolution und Integration bestehender Systeme vernachlässigen und (b) heutige modellgetriebene Verfahren zwar technische Plattformspezifika kapseln, aber dennoch Eingabemodelle benötigen, die konzeptionell häufig kaum abstrakter sind als Programm-Code und damit nur ähnlich kostenintensiv zu erstellen. Daher werden statt UML-basierender MDA-Sprachen zunehmend domänenspezifische Sprachen diskutiert [4]. Diese Sprachen sind einfacher zu handhaben und können besser an den Besonderheiten ihrer branchenspezifischen Einsatzgebiete orientiert werden. Gerade aber die CIM-Formulierung zur Darstellung kombinierter Geschäfts- und Problemlöseprozesse ist zurzeit mehr Kunst als ingenieurmäßig geplantes Vorgehen. Hier kann der Transfer musterbasierter Vorgehensweisen, wie sie z.B. im BMBF-geförderten Projekt InPULSE erarbeitet werden [5], auf die CIM-Ebene eine erste verbessernde Maßnahme darstellen.

Die im Projekt MINT zu erarbeitenden Ergebnisse unterstützen eine durchgängige modellgetriebene Vorgehensweise bei der Integration von Software in betrieblichen Softwaresystemlandschaften, indem die Herausforderungen in den Bereichen (a) der Integration bestehender Softwaresysteme mit der Fokussierung auf betriebliche Informationssysteme und (b) domänenspezifischer Sprachen mit einander verknüpft werden (vgl. Abbildung 1). Die folgenden Absätze geben einen genaueren Einblick in die im Projekt verfolgten Vorgehensweisen in den

beiden genannten Themenfeldern. Ein dritter Absatz beschreibt die geplante Validierung der Projektergebnisse im Vergleich zu existierenden Verfahren.

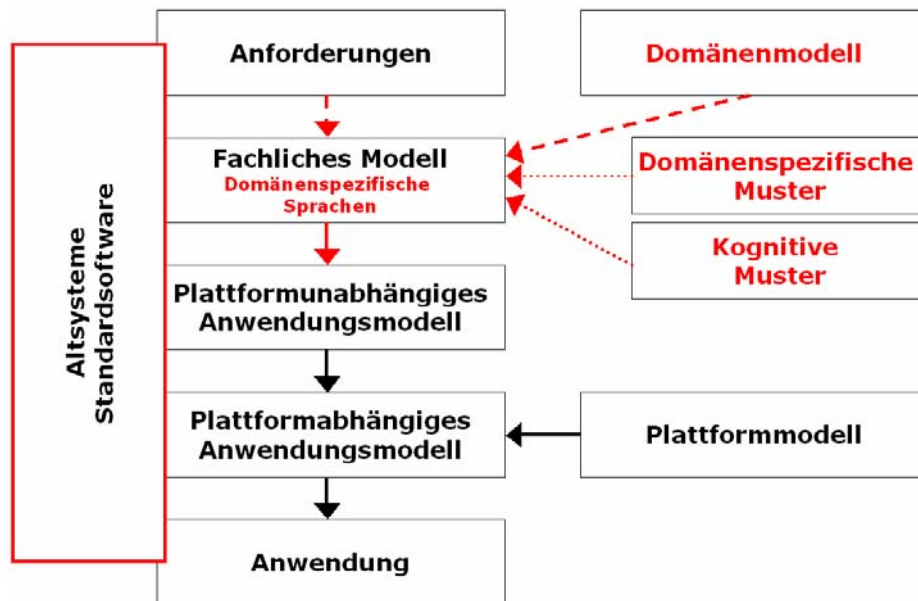


Abbildung 1: Erweiterung des modellgetriebenen Ansatzes um Integrationsaspekte

Integration betrieblicher Softwaresysteme

MINT befasst sich gezielt mit einer Umsetzung des modellgetriebenen Ansatzes in bereits existierenden Softwaresystemlandschaften. Als zwei typische konkrete Einsatzszenarien greift es dabei die Integration von Individual-Software mit betrieblicher Standardsoftware sowie die Kopplung objektorientierter Geschäftslogik an existierende relationale Legacy-Datenbanken heraus.

Obwohl sich die großen Standardsoftwarehersteller mittlerweile von der bisherigen typischen Client-Server-Architektur [6] abwenden, ihre Systeme für die Integration bzw. die individuelle Softwareentwicklung öffnen und sich dabei marktgängigen Standards (J2EE, .NET, Web-Services, BPEL4WS, SOA, ...) anpassen, sind viele der älteren Anwendungssysteme, wie sie in Unternehmen häufig noch im Einsatz sind, weiterhin monolithisch aufgebaut. Die Herausforderung der Integration neuer Software mit diesen monolithischen Altsystemen besteht darin, dass die dort bereits implementierten Fachverfahren sowie der häufig über Jahrzehnte aufgebaute Datenbestand in den neu zu realisierenden Geschäftsprozessen aufgehen müssen. Für das Projekt MINT ergeben sich dabei verschiedene Herausforderungen. Zum einen existieren für die bestehenden Anwendungen häufig keine Modelle, so dass ihre Daten und Verfahren zunächst identifiziert und modelliert werden müssen. Zum anderen muss die modellgetriebene Integration berücksichtigen, dass der Geschäftsbetrieb des einsetzenden Unternehmens nicht unterbrochen werden kann. Die Integration muss somit schrittweise als eine sogenannte "sanfte" Integration vorgenommen werden, die auch in Kauf nimmt, dass sich bestehende Anwendungen während dieser Integrationsphase durch neue Anforderungen ebenfalls noch weiterentwickeln können.

Bereits erforschte Verfahren zum Reverse Engineering werden in MINT durch die Einbeziehung des Domänenexperten nicht nur bei der Erstellung der fachlichen Modelle für die neu zu

entwickelnde Lösung, sondern auch bei der Modellierung von Altsystemen ergänzt. Bei der CIM-Modellierung werden unter Verwendung der Wizard-of-Oz-Technik [7] die kombinierten Geschäftsprozesse des Zielsystems durch ein verteiltes Agentensystem abgebildet. Die Verteilung der Aufgaben zwischen den Akteuren und die notwendige Kommunikation werden mittels der CRC-Karten-Technik festgelegt [8]. Jeder Akteur hat bestimmte Aufgaben, die in seiner Verantwortung liegen, zu lösen. Im Rahmen dieser CIM-Erstellung findet auch die systematische funktionsgetriebene Identifikation möglicher Services des Legacy-Systems statt. Diese Services werden im CIM als eigenständige Akteure repräsentiert.

Für die sanfte Integration bilden existierende Migrationsmuster einen vielversprechenden Ausgangspunkt und werden im Projekt vom Bereich der Systemmigration auf den Bereich der Systemintegration übertragen werden. Eines dieser Muster ist das Dublo-Muster (Dual Business Logic), das darauf setzt, die im bestehenden System vorhandene Geschäftslogik zunächst in einer neuen Geschäftslogikschicht zu duplizieren [9]. Die neue Schicht greift dabei über Adaptoren auf die vorhandene Geschäftslogik und damit die existierenden Datenbestände des Altsystems zu (vgl. Abbildung 2). Durch die Verwendung des Musters kann eine schrittweise Migration vollzogen werden, indem immer mehr Geschäftslogik aus dem alten in das neue Software-System wandert. Zugleich wird eine konsistente Datenhaltung ermöglicht. In Kombination mit den oben beschriebenen Methoden der integrierten CIM-Modellierung kann das Dublo-Muster als Basis für die modellgetriebene Integration eingesetzt werden.

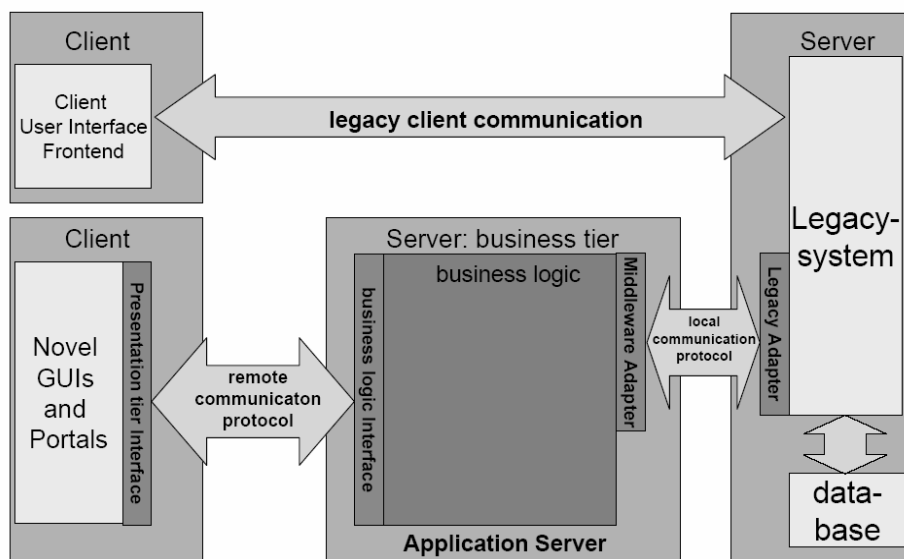


Abbildung 2: Das Dublo-Migrationsmuster [9]

Speziell für die Integration der Persistenzschicht bestehender Anwendungen wird in MINT zusätzlich moderne Generortechnologie [10] eingesetzt und weiterentwickelt. In einem ersten Teilschritt werden hier Schnittstellen für die zu generierenden Objekte (Data Objects) definiert sowie die interne (Klassen-)Struktur der Data Objects festgelegt und in Form von UML-Klassendiagrammen dokumentiert. Diese Definitionen sind plattformunabhängig. Basierend auf den Definitionen wird für eine ausgewählte Plattform (z.B. Java mit JDBC für die Datenzugriffe) ein erster plattformabhängiger Prototyp erstellt. Mit Hilfe dieses Prototyps wird das Klassenmodell hinsichtlich seiner Funktionsfähigkeit und Effizienz überprüft. Aus dem Objektmodell und dem Prototyp werden dann schließlich plattformunabhängige Muster

abgeleitet. Diese Muster bilden die Grundlage zur Definition eines Metamodells, das zur Erstellung plattformunabhängiger Datenmodelle verwendet wird. Das Metamodell hängt ab von den Informationen, die die Muster für die Generierung benötigen, und von den Informationen, die aus den Schemadefinitionen bzw. Datenmodellen der vorhandenen, zu integrierenden Datenbanken gewonnen werden können. Auf dieser Basis wird dann eine plattformspezifische Code-Generierung durchgeführt. Die plattformspezifischen Generatoren für die ausgewählte Plattform werden selbst wiederum mittels des Werkzeugs HyperSenses [3] generiert.

Einsatz domänenspezifischer Sprachen und Muster

Die Anforderung nach Individualisierung und damit nach Integration auf verschiedenen Technologieebenen ergibt sich wie oben bereits beschrieben aus der Tatsache, dass die Standardsoftware branchenspezifischen und kundenindividuellen Anforderungen nicht vollständig gerecht wird. Die Antwort auf die Frage nach der „richtigen“ Individualisierungsebene von Standardsoftware hat sich über die Jahre zunehmend verschoben. Während in den 80er-Jahren als wesentliches Mittel der Integration die Datenintegration (Unternehmensdatenmodell [11]) gesehen wurde, fokussiert sich heute die Mehrzahl der Integrationsprojekte auf die Anwendungsintegration (EAI). Zunehmend werden aber auch Prozess- und Präsentationsebenen in die Integration mit einbezogen (vgl. Abbildung 3).

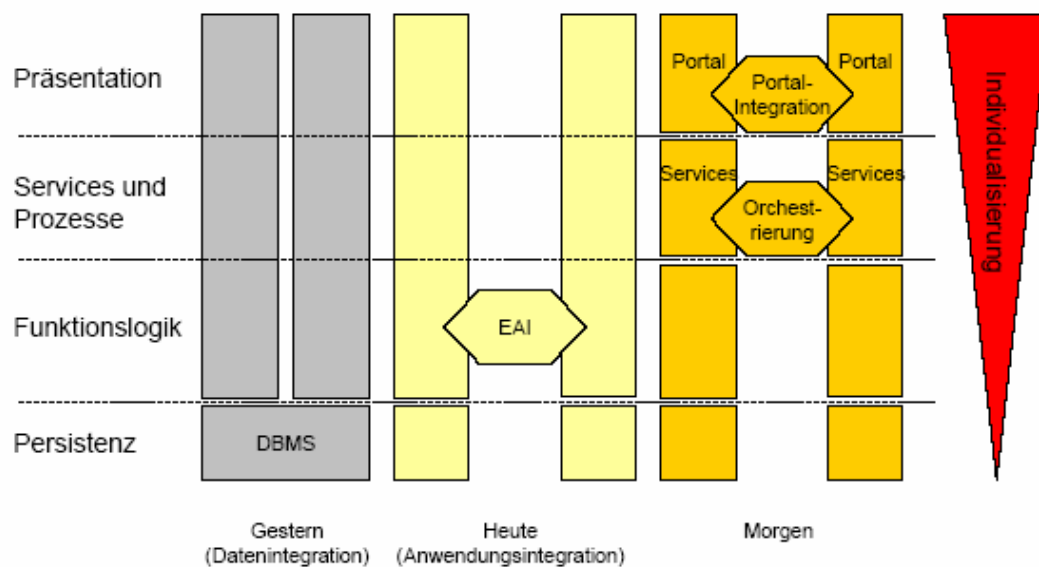


Abbildung 3: Verschiedene Integrations- und Individualisierungsebenen

Während sich in der Vergangenheit die Individualisierungsbemühungen bei Softwaresystemen auf die Entwicklung aller Ebenen konzentrieren mussten, wird sich die individuelle Softwareentwicklung heute und in Zukunft vermehrt auf die Gestaltung der Prozesse, die damit verbundene Integration von Diensten zu neuen Diensten (Orchestrierung) sowie die Integration von Benutzungsoberflächen auf Basis von Portaltechnologien fokussieren. Zwar wird bei der Entwicklung individualisierter Lösungen der Durchstoß bis zur Persistenzschicht auch in absehbarer Zeit nicht obsolet werden, dennoch wird zu erkennen sein, dass sich die Potenziale für die Individualisierung entsprechend immer weiter nach oben, in Richtung der Prozess- und Präsentationsebenen, verlagern werden. Für Entwicklungs- und Integrationsprozesse, die auf

diesem höheren Abstraktionsniveau angesiedelt sind und eine Beteiligung von Fachexperten im Sinne von einfach zu verstehenden und domänenbezogenen Sprachen direkt unterstützen würden, fehlen zurzeit jedoch noch wohlverstandene Vorgehensweisen und Werkzeuge.

Das Projekt MINT begegnet diesen Schwachstellen, indem es einerseits durch seine modellgetriebene Vorgehensweise die verschiedenen Abstraktionsebenen bei der Integration deutlich voneinander entkoppelt. Andererseits stärkt es mit der Abstraktionsebene, auf der das CIM erstellt wird, genau die Ebene, auf der Fachexperten in den Integrations- und Individualisierungsprozess mit eingebunden werden müssen. Diese Stärkung besteht zum einen in der Entwicklung geleiteter Verfahren zur Erstellung von integrierten CIM, die bereits im vorherigen Absatz beschrieben wurden. Zum anderen stellt MINT domänenspezifische Modellierungssprachen zur Verfügung, die ebenso wie domänenspezifische Sprachen allgemein [12] einen Effizienzgewinn durch Fokussierung auf die zu modellierende Anwendungsdomäne erreichen. So kann für die zu beteiligenden Fachexperten ein direkter und intuitiver Zugang zur CIM-Modellierung hergestellt werden. Für die Verwendung dieser Sprachen in der Praxis kommen Language Workbenches [13] zum Einsatz, die der direkten Umsetzung beliebiger Modellierungssprachen innerhalb von graphischen Werkzeugen dienen.

In den entwickelten domänenspezifischen Modellierungssprachen werden Geschäfts- und Problemlöseprozesse auf der Fachebene beschrieben. Um eine Wiederverwendung bereits auf dieser Ebene erreichen zu können, werden die von den Akteuren verfolgten Problemlösungen auf der Basis einer Patternontologie dann mit Hilfe von Cognitive [14], Problem-Solving- [15], Knowledge [16] und Business Patterns [17] gesichert. Durch die verbesserte Verständlichkeit und inhaltliche Transparenz wird eine CIM-Variante gewonnen, die anschließend, ebenfalls auf der fachlichen Ebene, in UML transformiert werden kann. Dieses UML-CIM entspricht in etwa den UML-Analysemodellen, die heute in Softwareprojekten entstehen.

Validierung des Ansatzes

Es gibt bereits heute eine Vielzahl von Ansätzen, um die Kopplung von Datenbanken und Geschäftslogik zu realisieren (z.B. Adaptergenerierung, Nutzung standardisierter Adapter, Persistenz-Frameworks etc). Dabei besitzen die o.a. Kopplungsansätze spezifische Vor- und Nachteile bezüglich ihrer Auswirkungen auf Wartbarkeit, Performanz und Skalierbarkeit des Systems. Der im Projekt MINT zu entwickelnde modellgetriebene Ansatz muss mit Rücksicht auf bestehende Ansätze evaluiert und seine vermuteten Vorteile hinsichtlich Wartbarkeit, Entwicklungskosten und Performanz validiert werden. Diese Auswirkungen sind selbst bei den bestehenden Verfahren bisher nur unzureichend verstanden: Kritische Entwurfsentscheidungen werden im besten Fall von Einzelerfahrungen getrieben, öfter noch in Unwissenheit ihrer Auswirkungen auf die o.a. kritischen Systemeigenschaften. Dies führt zu unvorhersehbaren Projektrisiken durch unzureichende Performanz und Skalierbarkeit, die erst spät bei Integrations- und Stresstests erkannt werden und zu hohen Folgekosten wegen mangelnder Wartbarkeit führen. Eine Änderung der Architektur ist zu diesem Zeitpunkt meist nicht mehr möglich. Erschwerend kommt hinzu, dass Wartbarkeit und Performanz bei einigen Datenbankkopplungstechniken antagonistische Eigenschaften darstellen: Eine Erhöhung der Wartbarkeit durch Einsatz von Persistenz-Frameworks kann zu signifikanten Performanzeinbußen führen.

Daher ist das Ziel dieses Arbeitspakets neben der systematischen Evaluation des im Projekt geschaffenen neuen Ansatzes zunächst die Entwicklung eines systematischen Verfahrens zum Treffen von Architekturentwurfsentscheidungen bezüglich der Datenbankankopplung. Dieses Verfahren berücksichtigt Kosten, Wartbarkeit und die Performanz objektrelationaler Adapter.

Für die Validierung wird eine Testumgebung zum Vergleich verschiedener Kopplungstechniken hinsichtlich ihrer Wartbarkeit, Performanz und Skalierbarkeit entwickelt. Diese Testumgebung basiert auf den Ergebnissen eines bereits durchgeführten Projekts, in dem schon eine Kopplungstechnik in einer realen Architektur und unter realistischen Nutzungsbedingungen untersucht und umgesetzt wurde. Diese Basierung auf einem realen System erleichtert die Interpretation der Messergebnisse bei Entscheidungen in ähnlichen Projekten. In einer artifiziellen Messumgebung hätten Messergebnisse hingegen nur eingeschränkte Aussagekraft.

Die in MINT entwickelte Testumgebung wird sowohl die ursprüngliche Applikation mit generischer Persistenzschnittstelle, an der unterschiedliche Kopplungsansätze erprobt und ausgetauscht werden können, berücksichtigen als auch ein Framework zur Automatisierung von Performanz und Skalierungstests einschließlich einer Anbindung des Frameworks an die Applikation und entsprechender sinnvoller Testszenarien. Das Framework wird die Messung und Protokollierung entsprechender Kennzahlen während und nach den Testläufen erlauben. Darüber hinaus werden einige typische, fachlich sinnvoller Änderungs- und Weiterentwicklungsszenarien zur späteren Durchführung mit den unterschiedlichen Kopplungsansätzen beschrieben und integriert.

3. Erfahrungen

Da das Projekt MINT erst im zweiten Quartal 2006 begonnen hat, liegen noch keine umfassenden Erfahrungen aus dem Projektkontext vor.

4. Literatur

- [1] Bundesministerium für Bildung und Forschung: Analyse und Evaluation der Softwareentwicklung in Deutschland. Siehe <http://www.isi.fhg.de/publ/downloads/isi00b69/software.pdf> (Abruf: 21.08.2005).
- [2] OMG, MDA Guide Version 1.0.1. Siehe <http://www.omg.org/docs/omg/03-06-01> (Abruf: 16.08.2005).
- [3] C. Giese, R. Schilling. Modellgetriebene Generatorentwicklung. In *ObjektSpektrum* 03/2005, SIGS Datacom.
- [4] Martin Kempa, Zoltan Adam Mann. Model Driven Architecture. In *Informatik Spektrum*, Band 28, Heft 4, August 2005, S. 298-302. Springer Verlag, Berlin 2005.
- [5] C. Möbus, H. Seebold, H. Garbe. A Greedy Knowledge Acquisition Method for the Rapid Prototyping of Knowledge Structures, K-CAP 2005, Third International Conference on Knowledge Capture, October 2-5, 2005, Banff, Canada.
- [6] M. Englbrecht und M. Wegelin. EAI-Programmierung mit mySAP. Spektrum, Akademischer Verlag, 2005.
- [7] D. Maulsby, S. Greenberg and R. Mander. Prototyping an intelligent agent through Wizard of Oz. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, Amsterdam, Niederlande, Seite 277-284, Mai 1993, ACM Press.
- [8] Bellin, D. and Simeone, S. S., *The CRC Card Book*, Addison-Wesley, 1999.
- [9] Hasselbring, Wilhelm; Reussner, Ralf; Schlegelmilch, Jürgen; Jaekel, Holger; Teschke, Thorsten; Krieghoff, Stefan; Langnickel, Marc: The Dublo Architecture Pattern for Smooth Migration of Business Information Systems – An Experience Report. In: *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004)*, 2004, S. 117 – 126.

- [10] K. Czarnecki, Eisenecker, U.: Generative Programming – Methods, Tools, and Applications, Boston, Addison Wesley, 2005, ISBN 0-201-30977-7.
- [11] A.W. Scheer. Wirtschaftsinformatik, Springer Verlag, Berlin, Heidelberg, New York, 1988.
- [12] Mernik, Marjan; Heering, Jan; Sloane, Anthony M.: When and how to develop domain-specific languages. ACM Computing Surveys 37 (2005) 4, S. 316-344.
- [13] Fowler, Martin: Language Workbenches: The Killer-App for DomainSpecific Languages? <http://www.martinfowler.com/articles/languageWorkbench.html>, 2005-06-12, (Abruf: 20.07.2006).
- [14] K.M. Gardner, A.R. Rusch, M. Crist, R. Konitzer und B. Teegarden. Cognitive Patterns: Problem-Solving Frameworks for Object Technology, Cambridge University Press, 1998.
- [15] M. Crubézy und M.A. Musen. Ontologies in Support of Problem Solving. In: S. Staab und R. Studer (Hrsg.), Handbook on Ontologies, Seite 321 – 341, Springer, 2004.
- [16] P. Clark, J. Thompson und B. Porter. Knowledge Patterns. In: S. Staab und R. Studer (Hrsg.), Handbook on Ontologies, Seite 191 – 207, Springer, 2004.
- [17] H.E. Eriksson , M. Penker. Business Modeling with UML: Business Patterns at Work, New York: Wiley, 2000, ISBN 0-471-29551-5.