Modellgetriebene Entwicklung im Kontext des Integration Engineering

Stefan Kühne
Universität Leipzig, Institut für Informatik,
Abteilung Betriebliche Informationssysteme
Johannisgasse 26
04103 Leipzig
kuehne@informatik.uni-leipzig.de

Abstract: Modellgetriebene Entwicklungsansätze betonen die zentrale Rolle formaler Modelle und Modelloperationen im Lebenszyklus eines Softwaresystems und werden bereits in verschiedenen Anwendungsbereichen eingesetzt. Gegenstand dieses Beitrags ist die Fragestellung, inwiefern sich Konzepte modellgetriebener Entwicklungsansätze auf den Bereich des Integration Engineering übertragen lassen und welche Auswirkung dies hat. Die Kombination beider Bereiche wird mit dem Begriff „Model-Driven Integration Engineering“ zusammengefasst, welcher anschließend erörtert wird.

1 Einleitung

Der Bereich des *Integration Engineering* (IE) [KR96, FTW05] widmet sich speziell der Lösung von Integrationsproblemen zwischen Informationssystemen und liefert hierfür entsprechende Methoden, Werkzeuge und Vorgehen [Thr05]. Das Integration Engineering versucht dabei ständig, durch neue Methoden, Werkzeuge und Vorgehensweisen eine Effizienzsteigerung zu erzielen. In diesem Zusammenhang ergibt sich die Fragestellung, inwiefern modellgetriebene Ansätze im Bereich des IE sinnvoll Anwendung finden können.

Diese Fragestellung soll im Folgenden erörtert werden. Hierzu wird aus IE-Sicht untersucht, in welcher Weise die den modellgetriebenen Ansätzen zugrunde liegenden Konzepte auf Integrationsphänomene abzubilden und inwiefern deren Zielstellungen mit denen des IEs vereinbar sind. Abschließend wird das Gebiet des *Model-Driven Integration Engineering* motiviert, welches eine Kombination beider Bereiche darstellt, und dessen Aufgabenstellungen für Forschung und Praxis diskutiert.

2 Modellgetriebene Software-Entwicklung

Modellgetriebene Entwicklungsansätze wie das *Model-Driven Engineering* (MDE) [Béz05, FN05, Sch06], das *Model-Driven Software Development* (MDSD) [SV06, S. 21ff.], das *Model-integrated Computing* (MiC) [KSLB03], die *Model Driven Architecture* (MDA),

das *Generative Software Development* (GSD) [Cza04, CE00] oder *Microsofts Software Factories* [GS04] betonen die zentrale Rolle formaler Modelle im Lebenszyklus von Softwaresystemen bzw. Softwaresystemfamilien. Gleichzeitig fordern modellgetriebene Entwicklungsansätze die konsequente Ausnutzung von Modellen durch Modelloperationen, welche zur Automatisierung des Entwicklungsprozesses für bestimmte Aufgaben definiert werden. Auf beide Aspekte soll im Folgenden kurz eingegangen werden.

Ein *Modell* dient als stellvertretende Beschreibung eines zugrunde liegenden Systems (Original), in welchem ausschließlich zweckdienliche Informationen repräsentiert sind. Stachowiak verweist in [Sta73, S. 129ff.] auf drei zentrale Merkmale von Modellen: (i) das *Abbildungsmerkmal,* d. h. Modelle sind Abbilder von etwas, (ii) das *Verkürzungsmerkmal*, d. h. Modelle beschreiben nur die relevanten Eigenschaften eines Originals, und (iii) das *Pragmatische Merkmal*, d. h. Modelle sind für einen bestimmten Modellbenutzer, in einem bestimmten Zeitintervall und für einen bestimmten Zweck dienlich. Zwei weitere Modelleigenschaften sollen ebenfalls erwähnt werden: (iv) Ein Modell und sein Original stehen zueinander in einer N-M-Beziehung [Fav04, S. 14]. Von einem Original können somit mehrere Modelle existieren, welche bspw. unterschiedliche Sichten auf das Original darstellen. (v) Da ein Modell selbst als System aufzufassen ist, kann es das Original eines anderen Modells darstellen. Auf diese Weise kann von einem (bspw. technischen Anwendungs-) System in mehreren Stufen abstrahiert werden. Modelle unterschiedlicher Sichten und Abstraktionsstufen tragen im Kontext der Software-Entwicklung dazu bei, das Verständnis für komplexe Systeme und Vorgänge zu erhöhen.

Die im Entwicklungsprozess erstellten Modelle bilden in modellgetriebenen Ansätzen den Ausgangspunkt für Modelloperationen, wie Modelltransformationen, Modellvalidierungen oder Modelldifferenzen, welche zur Automatisierung bestimmter Entwicklungsschritte definiert werden. Modelltransformationen ermöglichen beispielsweise die Anreicherung von Modellen mit zusätzlichen Informationen (z. B. technischen), die informationserhaltende Formüberführung von Modellen oder die Extraktion und Aggregation bestimmter Eigenschaften zur Bewertung von Modellen. In der Fokussierung auf Automatisierung des Entwicklungsprozesses liegt der wesentliche Unterschied zu modellbasierten Ansätzen, in denen dieser Aspekt weniger stark ausgeprägt ist. Ein höherer Automatisierungsgrad kann sich positiv hinsichtlich der Effizienz des Entwicklungsprozesses oder der Produktqualität auswirken. So erzeugen bspw. Modelltransformationen Artefakte mit definierter Qualität, indem Fehler oder alternative Entwurfsentscheidungen durch manuelle Transformationen ausgeschlossen werden. Die Steigerung des Automatisierungsgrads im Entwicklungsprozess erhöht jedoch auch den erforderlichen Formalisierungsgrad. So müssen im stärkeren Maße Artefakttypen, Methoden und Entwicklungsumgebungen spezifiziert werden, was sowohl den Anwendungsbereich des Entwicklungsprozesses einschränken kann als auch aus wirtschaftlichen Gesichtspunkten betrachtet werden muss. D. h. die zusätzlichen Aufwendungen, welche z. B. durch die Implementierung einer Modelltransformation entstehen, müssen mit dem zu erwartenden Nutzen in Beziehung gesetzt werden.

Erfahrungen zeigen, dass modellgetriebene Ansätze bereits in unterschiedlichen Bereichen, wie bei der Entwicklung verteilter eingebetteter Systeme oder komplexer Enterprise-Anwendungen, erfolgreich eingesetzt werden und zu Verbesserungen im Entwicklungsprozess führen können [Sch06]. Es existieren jedoch auch Projekte, in denen sich

die Erwartungen an modellgetriebene Ansätze nicht erfüllt haben [Rot06]. Ob sich der Einsatz einer modellgetriebenen Entwicklung lohnt, kann somit nicht pauschal beantwortet werden, sondern hängt vielmehr vom konkreten Anwendungsbereich ab. Im Folgenden soll der Anwendungsbereich des Integration Engineering betrachtet werden.

3 Anwendung modellgetriebener Entwicklungsprinzipien im Integration Engineering

3.1 Entwicklung integrationsfähiger Systeme

Rautenstrauch definiert in [Rau93] Integration Engineering „als Erweiterung bzw. Teildisziplin des Software Engineering“, in deren Mittelpunkt „Methoden, Verfahren und Werkzeuge zur Erstellung von Software“ stehen, „die den betrieblichen Integrationsanforderungen gerecht wird.“ [Rau93, S. 32] Er fokussiert dabei auf die Entwicklung eines Konzepts „für die Entwicklung *integrationsfähiger* Software“ [Rau93, S. 31] und betont, dass die Sicherstellung der späteren Integrierbarkeit eines Software-Systems bereits während des zugehörigen Herstellungsprozesses beachtet werden muss.

Die Integrationsfähigkeit bewertet die Möglichkeiten eines Systems, mit anderen Systemen verbunden (Integration im Sinne von Verbinden) oder verschmolzen (Integration im Sinne von Verschmelzen) zu werden. Der Begriff steht somit in engem Zusammenhang mit Begriffen wie Interoperabilität, Portabilität oder Kompatibilität. Die Integrationsfähigkeit eines Anwendungssystems wird durch verschiedene Faktoren bestimmt: (i) Eine wesentliche Anforderung sind Schnittstellen, über die das System mit seiner Umwelt interagieren oder entsprechend einer nicht-invasiven Integration erweitert werden kann. (ii) Die Integrationsfähigkeit verbessert sich außerdem, wenn die für einen bestimmten Bereich relevanten Standards zur Gestaltung von Struktur und Verhalten des Systems (z. B. Kommunikationsstandards oder Referenzarchitekturen) beachtet und erfüllt werden. (iii) Weiterhin bedingt die Integrationsfähigkeit eine gewisse Anpassbarkeit bzw. Flexibilität hinsichtlich nicht vorhersagbarer Umwelteigenschaften bzw. -anforderungen. (iv) Ebenso zu den Anforderungen gehört eine konsistente Dokumentation der Struktur und des Verhaltens des Systems, welche eine Grundlage für die Entwicklung einer Integrationslösung darstellt.

Bereits bei modellbasierten Ansätzen kann explizit die Integrationsfähigkeit eines Systems bspw. durch Modellierung von Systemschnittstellen oder Interaktionsverhalten adressiert werden. Im Vergleich dazu erhöhen modellgetriebene Ansätze jedoch die Konsistenz zwischen Modell und Implementation, da (soweit möglich) der Code aus den Modellen automatisiert abgeleitet wird. Bei der Auswahl bzw. Neuentwicklung von Modelltransformationen wird explizit entschieden, wie Modellinhalte auf zugrunde liegende Kommunikationstechniken, Programmiermodelle und Plattformen abgebildet werden. Dies unterstützt die explizite Definition einer Implementierungsstrategie sowie deren konsequente Umsetzung, was die Einhaltung von Standards begünstigt. Des Weiteren können Anforderungen hinsichtlich des Schnittstellenverhaltens an ein System spezifiziert und anschließend durch entsprechende Modellvalidierungen anhand der Modellierung überprüft werden.

3.2 Desintegration

Eine weiter gehende Definition des Begriffs Integration Engineering liefern Kurbel und Rautenstrauch in [KR96]. Sie erweitern darin den Aufgabenbereich des IE, indem sie nun explizit zwischen der Neu-Entwicklung von integrierten Informationssystemen – *„Integration ex ante“* – und der Zusammenführung bestehender (integrationsfähiger) Systeme – *„Integration ex post“* – unterscheiden. Insofern gehört die in Abschnitt 3.1 beschriebene Problemklasse zur Ex-ante-Integration. Ein weiterer Fall ist die *Desintegration* eines integrierten Informationssystems mit dem Ziel eines Refactoring, sodass eine bessere Integrierbarkeit dieses Systems erreicht wird.

Neben dem modellgetriebenen Forward Engineering ist ebenso modellgetriebenes Reverse Engineering denkbar [Fav04, S. 24]. Im Sinne eines Refactorings sind diese geeignet miteinander zu kombinieren. So können mittels Reverse-Enginnering-Transformationen [Vis01, S. 4] deskriptive Modelle aus einer bestehenden Implementierung erzeugt werden. Diese Modelle können entsprechend den Zielen der Desintegration umgestaltet werden (Refactoring). Anschließend können die Modelle mittels Refinement-Transformationen wieder auf Implementierungscode abgebildet werden.

3.3 Ex-post-Integration

Zur Ex-post-Integration gehören u. a. (i) die Anpassung und Integrierung eines Informationssystems in eine bestehende IT-Infrastruktur, (ii) die Integrierung von Informationssystemen oder deren Komponenten untereinander sowie (iii) die Anbindung bestehender Informationssysteme an eine zentrale Plattform. Die Anwendung modellgetriebener Ansätze in Ex-post-Szenarien bedeutet konkret, dass die entsprechende Integrationslösung nach modellgetriebenen Prinzipien konzipiert und umgesetzt wird.

Ein Vorgehen für Szenario (i) ist bspw. die Anpassungen des zu integrierenden Informationssystems anhand geeigneter Modelle zu beschreiben, aus diesen Modellen über Modelltransformationen geeignete Adapter- und Schnittstellenkomponenten zu erzeugen sowie über Modellvalidierungen die Konsistenz der Anpassungen zu überprüfen. Ein solches Vorgehen wird in [HK06] skizziert. Das E-Commerce-System Intershop Enfinity muss vor Inbetriebnahme an die Umgebungsbedingungen des jeweiligen betrieblichen Umfeldes hinsichtlich Benutzungsoberfläche, Geschäftslogik, Datenhaltung usw. auf verschiedenen Architekturebenen angepasst werden. Diese Anpassungen werden auf Basis domänenspezifischer Modelle beschrieben, aus denen anschließend der Anpassungscode abgeleitet wird. Eine Einsatzmöglichkeit für Szenario (iii) ist in [KW06] beschrieben. Hier basiert die Umsetzung von E-Government-Diensten auf bestehenden Fachanwendungen und Komponenten einer E-Government-Plattform. Die Verknüpfung dieser Komponenten erfolgt in einer zentralen Orchestrierung, welche auf Basis domänenspezifischer Modellierungssprachen und Modelltransformationen entwickelt wird.

Ein Spezialfall der Ex-post-Integration ist die Ersetzung (Migration) eines bestehenden integrierten Informationssystems durch eine aktualisierte Version. Haben sich die Schnitt-

stellen des Systems im Vergleich zur alten Version geändert, was im Zuge fortschreitender technologischer Entwicklungen wahrscheinlich ist, sind die am Altsystem vorgenommen Anpassungen (in Form von Konfigurationen oder Anpassungscode) u. U. nicht mehr gültig. Aufbauend auf oben genanntem Vorgehen kann ein modellgetriebener Entwicklungsansatz in diesem Szenario Abhilfe schaffen: Der Hersteller liefert neben der aktualisierten Produktversion auch aktualisierte Transformationen aus, welche aus den Anpassungsmodellen Adapter- und Schnittstellenkomponenten konform zur neuen Produktversion erzeugen [HK06, S. 39].

Zusammenfassend kann festgestellt werden, dass Konzepte modellgetriebener Entwicklungsansätze im IE-Kontext Anwendung finden (können). Das zentrale Element formale Modellierung kann durch das Explizieren bestimmter Systemeigenschaften (wie Schnittstellenverhalten) als unterstützendes Element aufgefasst werden. Der Unterschied zu modellbasierten Ansätzen wird insbesondere am Spezifika Automatisierung durch Modelloperationen (Modelltransformationen, Modellvalidierungen) deutlich.

4 IE-relevante Zielstellungen modellgetriebener Entwicklungsansätze

Die Zielstellungen modellgetriebener Entwicklung sind vielfältig [SV06, S. 13f.], [GPR06, S. 21f.]. Sie adressieren sowohl Produkt- als auch Prozessqualitätsaspekte und ergeben sich direkt oder indirekt aus den zentralen Elementen Modellierung und Automatisierung durch Modelloperationen bzw. deren Zusammenspiel.

4.1 Steigerung der Integrationsfähigkeit durch verbesserte Produktqualität

Ein wesentliches durch modellgetriebene Ansätze adressiertes Produktqualitätsmerkmal ist *Wartbarkeit*. Diese wird bspw. erhöht durch (i) Steigerung der Softwarequalität durch formale Modellierung, (ii) Steigerung der Synchronität zwischen Entwurfsmodellen und Implementierungscode, (iii) Redundanzvermeidung durch zentralisierte Erfassung von verteilten Implementierungsaspekten in Modellen (Separation of Concerns) und (iv) der einheitlichen Abbildung von (fachlichen) Konzepten auf (technische) Plattformen.

In modellgetriebenen Ansätzen werden fachliche Konzepte und Zusammenhänge aus dem Implementierungscode in Modellierungssprachen und Anwendungsmodellen „herausgelöst“ und über Modelltransformationen wieder auf die zugrunde liegende Plattform abgebildet. Änderungen an der technischen Plattform bedingen im Idealfall lediglich Änderungen an den Transformationen. Dies erleichtert den Austausch der zugrunde liegenden Plattform (*Portierbarkeit* der Anwendung) und erhöht damit die Unabhängigkeit von Plattformtechnologien und Plattformherstellern.

Ein weiteres Merkmal, das insbesondere im Fokus der MDA-Inititiative steht, ist *Interoperabilität*. Diese wird begünstigt durch die konsequente Nutzung von Standards, bspw.

für die technische Umsetzung von Modellrepositories und Transformationen oder Richtlinien hinsichtlich der Strukturierung von Modellen und Transformationen. Des Weiteren verbessert die Trennung von fachlicher Logik und deren technologischer Repräsentation die Definition, Generierung und Anpassung von Schnittstellen, wodurch sich ebenfalls die Interoperabilität erhöht.

Portierbarkeit und Interoperabilität stehen in engem Zusammenhang mit der Integrationsfähigkeit eines Systems. Elemente der Wartbarkeit, wie Modell-Code-Synchronität, begünstigen ebenfalls die Integrationsfähigkeit. Konkrete Integrationsszenarien treten häufig in dezentral gesteuerten oder dynamischen Umgebungen auf (vgl. bspw. [KTRL07, FFSD06]). In beiden Fällen spielt die Wartbarkeit der Integrationslösung eine wesentliche Rolle und ist somit auch für die Ex-post-Integration relevant.

4.2 Verbesserung der Prozessqualität

Ein primäres Ziel modellgetriebener Entwicklungsansätze ist verbesserte *Handhabbarkeit der Komplexität*. Komplexitätreduktion kann durch verschiedene Mechanismen, wie Abstraktion, Separation of Concerns, Teile-und-Herrsche-Prinzip, Verdichtung oder Priorisierung adressiert werden. Modellgetriebene Ansätze fokussieren u. a. auf *Abstraktion* und *Separation of Concerns* (SoC). Abstraktion meint die Beschränkung auf die für eine bestimmte Fragestellung relevanten Merkmale eines Gegenstandsbereichs bzw. die Auslassung aller irrelevanten Merkmale. SoC bezeichnet die Aufteilung einer Problemstellung in *verschiedenartige* Teilprobleme und wird in modellgetriebenen Ansätzen durch die grundlegende Aufgabenteilung in Modellierung, Architekturdefinition, Transformation usw. sowie durch Modellierung hinsichtlich verschiedener Sichten unterstützt. Beide Komplexitätsreduktionsmechanismen werden in modellgetriebenen Ansätzen in Form von Modellierungssprachen und Werkzeugen expliziert, was ein *produktiveres Umfeld* [SV06, S. 13] im Vergleich zu einer ausschließlich auf eine Implementierungstechnologie beschränkten Realisierung ermöglicht. In IE-Projekten spielt die Handhabbarkeit von Komplexität oft eine entscheidende Rolle. Insbesondere in Ex-post-Integrationsszenarien, mit einer einer großen Anzahl an zu integrierenden Systemen, vielfältigen Systemschnittstellen und variantenreichen Integrationsprozessen sind Mechanismen wie Abstraktion und SoC zur Komplexitätsreduktion erforderlich.

Das SoC-Prinzip begünstigt des Weiteren Arbeitsteilung hinsichtlich spezieller Kenntnisse und Fähigkeiten (Separation of Jobs), was zu Spezialisierungen sowie effizienten Ressourcenallokation führen kann. IE-Szenarien betreffen häufig sowohl fachliche als auch technische Belange – beispielsweise bei der prozessbasierten Integration betrieblicher Anwendungssysteme. Eine Spezialisierung beteiligter Rollen und Aufgabenbereiche kann hier zu verbesserter Kommunikation zwischen fachlich- und technisch-orientierten Personengruppen führen. Im Falle einer prozessbasierten Ex-post-Integration kann bspw. ein Anwendungsexperte die fachliche Prozessmodellierung auf Basis für ihn intuitiver Konzepte durchführen, während die Abbildung fachlicher Prozesse auf ein technisches Kopplungssystem von einem Technologieexperten definiert wird.

Zusammenfassend kann festgestellt werden, dass sich die genannten Zielstellungen modellgetriebener Ansätze hinsichtlich Produkt- und Prozessmerkmalen positiv in IE-Problemklassen auswirken. Weitere Zielstellungen, wie (i) Steigerung der Software-Qualität, (ii) Steigerung der Entwicklungsgeschwindigkeit und (iii) Reduzierung von Entwicklungskosten durch Automatisierungs- und Wiederverwendungseffekte sind ebenfalls zu nennen. Sie stehen den IE-Zielen von IE-Problemklassen nicht entgegen – sind sozusagen kompatibel zu diesen. Der Bedarf an der Realisierung solcher Effekte hängt jedoch von allgemeinen, d. h. nicht integrationsspezifischen, Projekteigenschaften ab.

5 Model-Driven Integration Engineering

Nachdem dargestellt wurde, dass modellgetriebene Ansätze im IE für verschiedene Problemklassen Anwendung finden und deren Ziele kompatibel zu IE-Zielen sind – und sich damit unterschiedliche Verbesserungspotenziale eröffnen – soll nun eine explizite Abgrenzung des Bereichs in Form einer Definition vorgenommen werden. Anschließend werden die sich daraus ergebenden Aufgabenfelder skizziert.

5.1 Definition

Unter *Model-Driven Integration Engineering* (MDIE) wird die Auswahl, Adaption, Verfeinerung und Anwendung modellgetriebener Entwicklungsansätze (wie MDE, MDSD, MDA oder GSD) bzw. deren zugrunde liegender Prinzipien, Methoden, Techniken und Vorgehensmodelle, für die Anwendung im Bereich des Integration Engineering verstanden. Modellgetriebene Ansätze werden also zur Lösung verschiedener IE-Problemklassen, wie der Ex-ante-Integration oder der Ex-post-Integration im Sinne des Verbindens und der Vereinigung angewendet. Dabei werden verschiedene Zielstellungen verfolgt. Die Produktaspekte wie Integrationsfähigkeit, Wartbarkeit, Portierbarkeit und Interoperabilität des (weiter-) zu entwickelnden Systems stehen ebenso im Vordergrund wie die Verbesserung von Prozessaspekten insbesondere die bessere Handhabbarkeit von Komplexität, fachgerechte Aufgabenverteilung sowie Flexibilität und Agilität. Aus dieser Definition und den Zielstellungen heraus ergibt sich ein breites Aufgabenspektrum.

5.2 Aufgabenfeld Modellierung

Grundlage für Modellierung im Sinne des MDIE ist es, modellierungsrelevante Integrationskonzepte zu identifizieren und zu beschreiben. Darauf aufbauend kann analysiert werden, welche Abstraktionsmechanismen auf diesen Konzepten angewendet werden können. Anschließend können relevante Modellierungssprachen identifiziert und bewertet werden. Abhängig davon sind im MDIE Sprachen zur Beschreibung von Integrationsphänomenen auszuwählen, anzupassen, zu adaptieren und ggf. neu zu entwickeln. Da davon auszu-

gehen ist, dass verschiedene Sprachen zur Beschreibung unterschiedlicher Integrationsaspekte relevant sind, sind im Sinne von Sprachenarchitekturen und SoC die Beziehungen zwischen diesen Sprachen zu definieren. Dabei sind ebenfalls die Rollen Modellersteller und -benutzer so zu verfeinern, so dass diese fachlichen und technischen Personengruppen zugeordnet werden können.

5.3 Aufgabenfeld Modelloperationen

Der zweite Aufgabenbereich im MDIE ist die Definition von Modelloperationen und deren Implementierung. Hierzu gehören Modelltransformationen (z. B. Refinements, Refactorings), Modellvalidierungen oder Modellmetriken. Diese Operationen sind IE-spezifisch, da sie auf Integrationszuständen und Integrationsprozessen operieren. Beispielsweise sind für die Definition von Refinement-Transformationen abstrakter Prozessmodelle auf ausführbare Orchestrierungssprachen Beziehungen zwischen high-level und low-level Konstrukten festzulegen.

Durch die Beschränkung des Anwendungsbereichs modellgetriebener Ansätze auf Problemstellungen des IE können Lösungsansätze für allgemeine Problemstellungen konkretisiert oder verfeinert werden. Beispielsweise können speziell für Integrationsmodelle Techniken für den Umgang mit manuellen Verfeinerungen nach Transformationen definiert werden. Weiteres Verfeinerungs- und Konkretisierungspotenzial ergibt sich durch die Problemstellung Modellevolution. Aufgrund der Kenntnis der eingesetzten Integrationssprachen sowie deren Anwendungsbereiche können Aussagen über Auswirkungen von Sprach- und Modelländerungen (Integrationssprachen- und Integrationsmodellevolutionen) auch semantisch getroffen und begründet werden. Weitere Problemstellungen in diesem Zusammenhang sind das Transformationsmanagement und die Komposition von Transformationsdefinitionen.

5.4 Aufgabenfeld domänenspezifische Erweiterungen

Integrationsprojekte werden oft in spezifischen (z. B. fachlich motivierten) Umgebungen durchgeführt, bspw. im E-Government- oder E-Commerce-Bereich. Um den dort geltenden Konzepten Rechnung zu tragen muss der Anwendungsbereich allgemeiner MDIE-Sprachen weiter eingeschränkt und die enthaltenen Sprachkonzepte domänenspezifisch erweitert werden. Hierfür sind entsprechende Erweiterungskonzepte zu planen, die die Erweiterungsanforderungen erfüllen. Des Weiteren sind Richtlinien für die Analyse domänenspezifischer Konzepte und deren Repräsentation (Versprachlichung) in Form von domänenspezifischen Sprachkonzepten zu definieren. Analog zur Erweiterung von Modellierungssprachen um domänenspezifische Konzepte können auch für Modelloperatoren (z. B. Transformationen, Validierungen, Metriken, Modellevolutionen) domänenspezifische Erweiterungen definiert werden. In Abbildung 1 sind die Zusammenhänge noch einmal grafisch dargestellt.

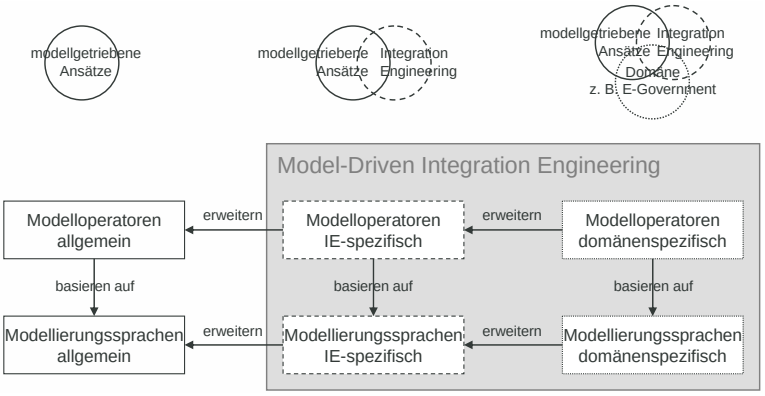

Abbildung 1: Aufgabenbereiche des Model-Driven Integration Engineering.

5.5 Aufgabenfeld Werkzeuge für die praktische Umsetzung

Neben den Grundkonzepten Modellierung und Automatisierung durch Modelloperationen und der zweiten Dimension (domänenspezifische Erweiterungen), kann eine dritte identifiziert werden – die der technischen Umsetzung. Aus praktischer Sicht muss die Modellierung und die Ausführung von Modelloperationen durch entsprechende Werkzeuge hinterlegt werden. Die daraus resultierenden Aufgaben gestalten sich dabei analog zur konzeptionellen Sicht.

Es ist zu untersuchen, welche Modellierungswerkzeuge, insbesondere Modelleditoren und -repositories, zur Unterstützung eingesetzt werden können. Hierbei sind die Anforderungen an diese Werkzeuge aus MDIE-Sicht zu definieren (bspw. die Umsetzung eindeutiger IDs für Modellelemente, Umsetzung von Assoziationen), Bewertungskriterien festzulegen und bestehende Werkzeuge (wie EMF, GMF, MDR oder ARIS) dahingehend zu bewerten. Des Weiteren können bei Bedarf Brücken zwischen verschiedenen Werkzeugen geschaffen werden. Für die Implementierung von Modelloperationen stehen analog zur Implementierung von Modellierungssprachen unterschiedliche Werkzeuge und Techniken zur Verfügung aus denen geeignete ausgewählt, angepasst und ggf. erweitert werden müssen. Analog zur Verfeinerung und Konkretisierung von allgemein unzureichend gelösten Problemstellungen modellgetriebener Ansätze ergeben sich durch die Einschränkung auf den MDIE-Anwendungsbereich auch aus technischer Sicht Potenziale. So können spezifische Aussagen über das Fragmentierungsproblem von Modellierungssprachen und die Verteilung von Modellpartitionierung getroffen werden.

6 Zusammenfassung

In diesem Beitrag wurde für verschiedene IE-Problemklassen untersucht, wie sich der Einsatz modellgetriebener Ansätzen gestalten kann. Die durch diese Ansätze fokussier-

ten Ziele sind kompatibel zu IE-Zielen bzw. begünstigen diese. Für die Umsetzung der Zielstellungen ergibt sich ein breites Fragen- bzw. Aufgabenspektrum (Auswahl geeigneter Modellierungssprachen, Definition entsprechender Modelloperationen, Konkretisierung allgemeiner Problemstellungen bzw. deren Lösungsansätze), welches mit dem Begriff Model-Driven Integration Engineering umrissen ist.

Die eingehende Beschäftigung mit diesem Thema ist sowohl aus IE-Sicht als auch aus Sicht modellgetriebener Ansätze relevant. Zum einen erweitert die Umsetzung modellgetriebener Ansätze im IE das Begriffsverständnis und unterstützt die Untersuchung konzeptioneller Zusammenhänge. Zum anderen trägt die Lösung allgemeiner Problemstellungen im speziellen Anwendungsgebiet zum Erfahrungsschatz modellgetriebener Ansätze bei und leistet auch dort einen Beitrag zu einem besseren Verständnis zugrunde liegender Prinzipien.

Literatur

[Béz05] Jean Bézivin. On the Unification Power of Models. *Software and System Modeling (SoSym)*, 4(2):171–188, 2005.

[CE00] Krzysztof Czarnecki und Ulrich W. Eisenecker. *Generative Programming : Methods, Tools, and Applications*. Addison-Wesley, Boston et. al., 2000.

[Cza04] Krzysztof Czarnecki. Overview of Generative Software Development. In J.-P. Banâtre, Hrsg., *Unconventional Programming Paradigms (UPP) 2004*, LNCS 3566, Seiten 313–328, Mont Saint-Michel, France, 2004.

[Fav04] Jean-Marie Favre. Foundations of Model (Driven) (Reverse) Engineering: Models – Episode I: Stories of the Fidus Papyrus and of the Solarus. In *Post-proceedings of Dagstuhl Seminar 04101 on Language Engineering for Model-Driven Software Development*, 2004.

[FFSD06] Daniel Fötsch, Sven Feja, Sascha Sauer und Andreas David. Problemstellungen agiler Schnittstellen am Beispiel des Commerce Management Systems von Truition/AGETO. In Klaus-Peter Fähnrich, Stefan Kühne, Andreas Speck und Julia Wagner, Hrsg., *Integration betrieblicher Informationssysteme: Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering*, Jgg. IV of *Leipziger Beiträge zur Informatik*, Seiten 49–56. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2006.

[FN05] Jean-Marie Favre und Tam Nguyen. Towards a Megamodel to Model Software Evolution Through Transformations. *Electronic Notes in Theoretical Computer Science*, 127(3):59–74, 2005.

[FTW05] Klaus-Peter Fähnrich, Maik Thränert und Peter Wetzel, Hrsg. *Umsetzung von kooperativen Geschäftsprozessen auf eine internetbasierte IT-Struktur : Arbeiten aus dem Forschungsvorhaben Integration Engineering*, Jgg. Band III of *Leipziger Beiträge zur Informatik*. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, 2005. ISBN: 3934178-52-9.

[GPR06] Volker Gruhn, Daniel Pieper und Carsten Röttgers. *MDA – Effektives Software-Engineering mit UML2 und Eclipse*. Xpert.press. Springer, 2006.

[GS04] Jack Greenfield und Keith Short. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley Publishing, Inc., 2004.

[HK06] Peter Hänsgen und Stefan Kühne. Modellgetriebene Softwareentwicklung zur Lösung von Integrations- und Migrationsproblemen am Beispiel des E-Commerce-Systems Intershop Enfinity. In Klaus-Peter Fähnrich, Stefan Kühne, Andreas Speck und Julia Wagner, Hrsg., *Integration betrieblicher Informationssysteme: Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering*, Jgg. IV of *Leipziger Beiträge zur Informatik*, Seiten 31–41. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2006.

[KR96] Karl Kurbel und Claus Rautenstrauch. Integration Engineering: Konkurrenz oder Komplement zum Information Engineering? – Methodische Ansätze zur Integration von Informationssystemen. In Heidi Heilmann, Hrsg., *Information Engineering: Wirtschaftsinformatik im Schnittpunkt von Wirtschafts-, Sozial- und Ingenieurwissenschaften*, Seiten 167–191. Oldenbourg, München, Wien, 1996.

[KSLB03] Gabor Karsai, Janos Sztipanovits, Akos Ledeczi und Ted Bapty. Model-integrated development of embedded software. *Proceedings of the IEEE*, 91(1):145–164, 2003.

[KTRL07] Stefan Kühne, Maik Thränert, Werner Rotzoll und Jan Lehmann. Model-Driven Integration Engineering in der E-Government-Domäne Meldewesen. In *Wirtschaftsinformatik 2007 - eOrganisation: Service-, Prozess-, Market-Engineering. 8. Internationale Tagung Wirtschaftsinformatik, 28. Februar - 2. März 2007 in Karlsruhe*, 2007.

[KW06] Stefan Kühne und Christian Welzel. Metamodellierung am Beispiel der E-Government-Domäne Meldewesen und Eclipse GMF. In Klaus-Peter Fähnrich, Stefan Kühne, Andreas Speck und Julia Wagner, Hrsg., *Integration betrieblicher Informationssysteme: Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering*, Jgg. IV of *Leipziger Beiträge zur Informatik*, Seiten 59–72. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2006.

[Rau93] Claus Rautenstrauch. *Integration Engineering: Konzeption, Entwicklung und Einsatz integrierter Softwaresysteme*. Addison-Wesley, Bonn ; Paris, 1. Auflage, 1993.

[Rot06] Roman Roth. Modellgestützte Software-Modernisierung. In *Model-Driven Development and Product Lines*, 2006.

[Sch06] Douglas C. Schmidt. Model-Driven Engineering. *IEEE Computer*, Februar 2006:25–31, 2006.

[Sta73] Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer, 1973.

[SV06] Thomas Stahl und Markus Völter. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006.

[Thr05] Maik Thränert. Integration – Eine Begriffsbestimmung. In Fähnrich et al. [FTW05], Seiten 11–22. ISBN: 3934178-52-9.

[Vis01] Eelco Visser. A Survey of Strategies in Program Transformation Systems. *Electronic Notes in Theoretical Computer Science*, 57(2), 2001.