Verarbeitung von ARIS-EPK-Modellen im Eclipse Modeling Framework

Heiko Kern und Stefan Kühne
Universität Leipzig, Institut für Informatik,
Betriebliche Informationssysteme
Johannisgasse 26
04103 Leipzig
{kern, kuehne}@informatik.uni-leipzig.de

Abstract: Die Architektur integrierter Informationssysteme (ARIS) ist ein technischer Raum für den Bereich des Geschäftsprozessmanagements. Ein wesentlicher Bestandteil von ARIS ist der Modelltyp Ereignisgesteuerte Prozesskette (EPK), der in verschiedenen ARIS-Modellierungsmethoden Verwendung findet. Die Interoperabilität von ARIS-Modellen mit anderen Räumen zur Wiederverwendung bzw. Weiterverarbeitung ist durch ARIS-spezifische Export- und Import-Schnittstellen beschränkt.

In diesem Beitrag wird ein umfassender Syntaxdefinitons-sensitiver Export/Import-Mechanismus für ARIS-Modelle zum Eclipse Modeling Framework (EMF-Raum) in Form einer ARIS-EMF-Brücke vorgestellt. Er ermöglicht Interoperabilität mit den vielfältigen EMF-Werkzeugen zur Modellverarbeitung. Das Anwendungspotenzial der Brücke wird anhand von Transformationsszenarien für EPK-Modelle skizziert und konkret am Beispiel der Syntaxprüfung mit ATL demonstriert.

1 Einleitung

Die Architektur integrierter Informationssysteme (ARIS) [Sch91, Sch98a, Sch98b, STA05] ist eine Referenzarchitektur zur Unternehmensmodellierung. Sie ist in fünf Sichten (Organisations-, Daten-, Kontroll-, Funktions- und Leistungssicht) sowie in drei Abstraktionsebenen gegliedert. Diesen Sichten und Abstraktionsebenen sind verschiedene Modellierungssprachen zugeordnet. Eine zentrale Modellierungssprache ist die (Erweiterte) Ereignisgesteuerte Prozesskette (EPK) [KNS92]. Der Prozesssicht zugeordnet, dient sie der Beschreibung von Geschäftsprozessen, indem sie Elemente statischer Sichten mit einem Kontrollfluss in Beziehung setzt.

ARIS kann als ein „technischer Raum“ betrachtet werden – ein Begriff, der erstmals in [KBA02] Erwähnung findet. Als solcher stellt ARIS eine Menge von Ideen, Konzepten, Werkzeugen, Möglichkeiten und erforderlichen Fähigkeiten dar und ist einer Gruppe/Gemeinschaft zugeordnet, in der ein gewisser Konsens hinsichtlich des Wissensgebietes bzw. der akzeptierten Literatur herrscht und welche sich über Workshops und Konferenzen austauscht.

Als technischer Raum stellt ARIS eine Reihe von Werkzeugen [IDS07] zur Verfügung, wie den ARIS SOA Architect oder den ARIS Business Architect. Sie werden für verschie-

dene Zielstellungen, wie ISO-9000-Zertifizierung, Performance Management, oder der Geschäftsprozessorientierten Ausrichtung von IT-Strukturen, in verschiedenen Branchen, wie E-Government, E-Finance oder Automotive, eingesetzt. Die Werkzeuge repräsentieren eine ganzheitliche Modellierungs- und Modellverwaltungsumgebung. Zur automatisierten Verarbeitung von Modellen stehen imperative Skriptsprachen und Makros zur Verfügung, mit denen beispielsweise Auswertungsreporte erzeugt werden können.

Ein technischer Raum, der sich aufgrund offener Standards, frei verfügbarer Open-source-Implementierungen und zahlreicher Werkzeuge für die Verarbeitung von Modellen empfiehlt, ist durch das Eclipse Modeling Framework (EMF) gegeben. Beispielsweise existieren für die Umsetzung von Modell-zu-Modell- und Modell-zu-Text-Transformationen Werkzeuge unterschiedlicher Ansätze (Graph-basierter Ansatz, Relationaler Ansatz, Funktionaler Ansatz, Template-basierter Ansatz). Diese Ansätze bieten im Vergleich zum imperativen Ansatz hinsichtlich Modularität, Anpassbarkeit, Wiederverwendbarkeit oder Benutzbarkeit ein signifikantes Verbesserungspotenzial.

Für die Interoperabilität mit anderen technischen Räumen stellt ARIS XML-basierte Schnittstellen zur Verfügung. Hierzu gehört der sprachunabhängige Ansatz der ARIS Markup Language (AML), der jegliche Art von ARIS-Modellen unabhängig der zu Grunde liegenden Modellierungsmethode exportiert. Des weiteren existieren sprachabhängige Exportmöglichkeiten, die bestimmte ARIS-Modelle, die anhand einer bestimmten Modellierungsmethode erstellt wurden, in ein bestimmtes Format (bspw. BPEL4WS) exportieren. Beide Möglichkeiten sind hinsichtlich der Weiterverarbeitungsmöglichkeiten in EMF-Werkzeugen, die sich im Allgemeinen auf Modelle und deren zu Grunde liegender Sprachdefinition gleichermaßen beziehen, beschränkt.

Ein allgemeiner Ansatz ist durch die im Folgenden skizzierte und demonstrierte ARIS-EMF-Brücke [KK07] gegeben. Sie erlaubt den Export in das EMF-spezifische Format für jegliche Art von ARIS-Modellen bei gleichzeitiger Extraktion der Sprachdefinition. Im EMF-Raum können die exportierten Modelle mit den zur Verfügung stehenden Werkzeugen bearbeitet werden und bei Bedarf anschließend wieder in ARIS-Werkzeuge importiert werden. Die zur Verfügung gestellte Interoperabilität vergrößert den Entscheidungsspielraum hinsichtlich zur Verfügung stehender Werkzeuge. Die Brücke basiert ausschließlich auf Meta-sprachlichen Konzepten und Beziehungen und folgt daher einem ähnlichen Ansatz [BDD+05] vergleichbarer Brücken zwischen EMF und den Microsoft DSL Tools [BHJ+05] oder EMF und MetaGME [BBC+05]. Sie ist damit unabhängig von einer gewählten ARIS-Modellierungsmethode. Des Weiteren sichert dieser Ansatz die Einheitlichkeit und Konsistenz der Umsetzung verschiedenartiger Sprachdefinitionsansätze.

Im Folgenden werden zunächst die Sprachdefinitionselemente des EMF- und ARIS-Raums erörtert. Aufbauend auf diesen Konzepten werden Beziehungen zwischen diesen Räumen hergestellt und der Aufbau der ARIS-zu-EMF-Brücke vorgestellt. Anschließend wird das Anwendungspotenzial skizziert und die Brücke anhand der Syntaxüberprüfung von EPKs mit ATL demonstriert.

2 Eclipse Modeling Framework

Das Eclipse Modeling Framework ist ein Open-source-Framework, das die Entwicklung von Java-Anwendungen unterstützt und als Plug-In für die Eclipse-Plattform zur Verfügung steht. Es ermöglicht die Beschreibung von Datenstrukturen in Form von Datenmodellen. Die Verwaltung von Instanzen (z. B. Serialisierung, Überwachung von Änderungen) erfolgt durch aus dem Datenmodell generierte Java-Klassen oder einer EMF-Laufzeitkomponente, welche Datenmodelle dynamisch interpretiert.

Im Sinne der modellgetriebenen Software-Entwicklung [SV06] wird ein EMF-Datenmodell als Metamodell und dessen Ausprägungen bzw. Instanzen als Modelle bezeichnet. Metamodelle werden in EMF mit Hilfe von Ecore [BSM+04], dem so genannten Metametamodell, beschrieben. Durch die Ausprägungs- bzw. Instanzbeziehungen zwischen Modell, Metamodell und Metametamodell ergibt sich eine Metamodellhierarchie, die aus den drei Ebenen M1–M3 besteht. Die Ausprägungs- bzw. Instanzbeziehung zwischen den Modellen auf den einzelnen Ebenen muss nicht einer objektorientierten Instanzbeziehung entsprechen, daher wird sie im Folgenden als Konform-zu-Beziehung [Béz05] bezeichnet.

Abbildung 1 zeigt ein Beispiel für die Metamodellhierarchie. Auf M3-Ebene ist ein Ausschnitt aus Ecore dargestellt, der die zentralen Modellelemente `EClass`, `EAttribute`, `EReference` und `EDataType` zeigt. Eine EClass (Ausprägung von `EClass`) beschreibt auf M2-Ebene eine Klasse von Modellelementen mit bestimmten gemeinsamen Eigenschaften. Relationen zwischen EClasses können mit Hilfe von EReferences (Ausprägung von `EReference`) ausgedrückt werden. Weiterhin können EClasses Attribute besitzen. Diese Attribute werden durch Ausprägungen von `EAttribute` beschrieben. Der Wertebereich eines EAttributes wird durch dessen EDataType festgelegt. Dies kann beispielsweise `EString`, `EInt` oder `EBoolean` sein.

Auf M2-Ebene des Beispiels ist ein Metamodell dargestellt, welches aus der EClass „Funktion“ und „Ereignis“ besteht. Ein Ereignis steht in Relation zu einer Funktion durch die Aktiviert-Relation und umgekehrt durch die Erzeugt-Relation. Diese beiden Relationen werden durch die EReference „Aktiviert“ und „Erzeugt“ realisiert. Auf M1-Ebene ist ein entsprechend zu dem Metamodell konformes Modell dargstellt. Es besteht aus der Funktion „F1“ und dem Ereignis „E1“ und der einer Erzeugt-Referenz „Erz1“, die „F1“ und „E1“ verbindet.

3 Architektur integrierter Informationssysteme

Analog zum EMF-Raum lässt sich der technische Raum ARIS ebenso in eine Drei-Metaebenen-Hierarchie gliedern [STA05]. Auf Basis eines ARIS-spezifischen Metametamodells (im Folgenden als A3-Modell bezeichnet) sind die zur Verfügung stehenden Modellierungskonzepte definiert. Im Gegensatz zu EMF ist dieses A3-Modell jedoch nicht explizit beschrieben.

Im Sinne eines Reverse Engineering lassen sich jedoch auf Basis verfügbarer Schnittstellen die zu Grunde liegenden Konzepte und Beziehungen identifizieren und in einem ein-

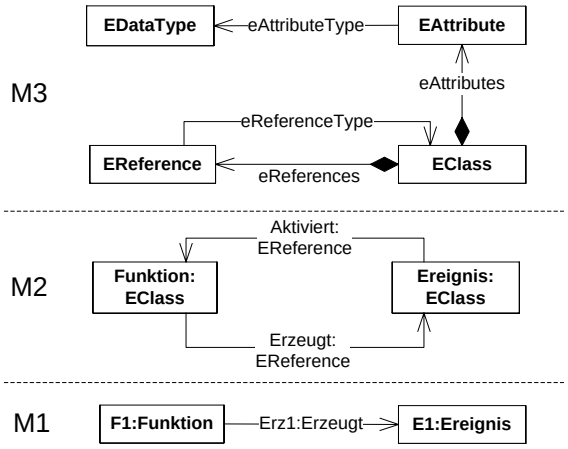


Abbildung 1: Metamodell-Hierarchie in Eclipse EMF

heitlichen Modell integrieren. Abbildung 2 zeigt die Ergebnisse einer Analyse des Werkzeugs ARIS Business Architect, insbesondere dessen Benutzungsschnittstelle, der Export-Schnittstelle AML, dem ARIS Filter-Mechanismus und die zur Verfügung stehende Programmierschnittstelle.

Im A3-Modell stellt `Modelltyp` das zentrale Element dar. Es besteht aus Kantentypen und Objekttypen, sowie deren grafischen Repräsentationen – Symbole und Linien. Modelltypen sind beispielsweise die „Ereignisgesteuerte Prozesskette“ oder das „Funktionszuordnungsdiagramm“. `Objekttyp` repräsentiert eine Menge von Objekttypen, wie „Ereignis“ oder „Funktion“, im A3-Modell. Hinterlegungsbeziehungen zwischen Modelltypen und Objekttypen legen fest, welche Modelltypen welchen Objekttypen hinterlegt werden können. Beziehungen zwischen Objekttypen werden durch Kantentypen definiert. Dabei sind Kantentypen, wie beispielsweise „aktiviert“ oder „erzeugt“, eigenständige Entitäten mit zwei mehrstelligen Referenzen auf mögliche Quell- und Zielobjekttypen. Modell-, Objekt- und Kantentypen können Eigenschaften, wie „Beschreibung“ oder „Name“ besitzen. Eigenschaften werden durch `Attributtyp` beschrieben und entsprechend von Modell-, Objekt- und Kantentypen referenziert. Der Wertebereich eines Attributtyps wird durch einen Datentyp spezifiziert. Die Festlegung der grafischen Repräsentation von Objekt-, Kanten- und Attributtypen erfolgt in Abhängigkeit von Modelltypen durch Symbole, Linien und Beschriftungen.

4 ARIS-EMF-Brücke

Ansatzpunkt für die Konstruktion einer M3-basierten Brücke ist zunächst die Konzeptabbildung der beiden Metametamodelle aufeinander (siehe Abbildung 3). Für M3-Konzepte, die sich nicht direkt aufeinander abgebilden lassen, ist die Nachbildung auf Basis zur Verfügung stehender Konzepte erforderlich. Auf Basis der M3-Ebenen-Beziehungen kön-

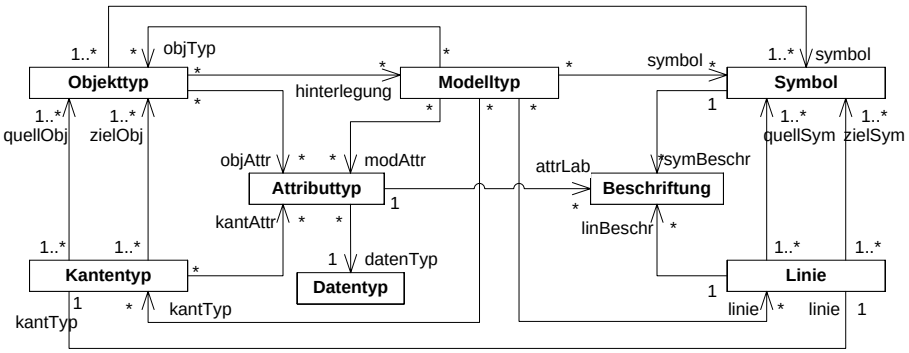


Abbildung 2: Definition der ARIS-Method auf Basis des ARIS-spezifischen Metametamodells

nen die Metamodelle auf M2-Ebene durch eine entsprechende Transformation aufeinander abgebildet werden. Der nächste Schritt besteht in der Abbildung auf M1-Ebene. Hierfür liegt ebenfalls die Konzeptabbildung auf M3-Ebene zu Grunde. Zusätzlich ist für die Definition einer entsprechenden Transformation auf M1-Ebene die Realisierung der Konform-zu-Beziehung in beiden Räumen relevant, d. h. in welcher Weise Metamodellelemente auf M1-Ebene ausgeprägt werden.

Die ARIS-EMF-Brücke bildet, wie im vorigen Absatz beschrieben, zunächst Modellelemente aus dem A3-Modell auf Ecore ab. Beispielsweise kann `Objekttyp` auf `EClass` abgebildet werden. `Kantentyp` könnte auf `EReference` abgebildet werden. Allerdings können im Gegensatz zu EReferences (ARIS-)Kantentypen Attribute besitzen. Deswegen werden Kantentypen auf EClasses abgebildet, die zwei EReferences besitzen. Diese beiden EReferences stellen die Beziehung zu den Quell- und Zielobjekten her. Weiterhin wird `Attributtyp` auf `EAttribute` abgebildet, wobei der Datentyp auf den entsprechenden EDataType abgebildet wird. Modelltypen werden ebenfalls auf EClasses abgebildet. Für die Unterscheidung der unterschiedlichen EClasses (Objekt-, Modell- und Kantentypen) werden entsprechende Oberklassen in den EMF-Metamodellen eingeführt. Aufbauend auf der Konzeptabbildung wurde eine Transformation in ARIS-Skript implementiert, welche ausgehend von einem Methodenfilter das entsprechende EMF-Metamodell in Form von XMI erzeugt. Die entgegen gesetzte Richtung – von einem EMF-Metamodell zu einem Methodenfilter – ist nur unvollständig realisierbar, da der Methodenfilter bzw. die zu Grunde liegende Methode lediglich eingeschränkt vom Nutzer geändert werden können.

Auf Basis der Abbildungen auf M3- und M2-Ebene wird eine Transformation auf M1-Ebene definiert. Hierbei ist die Speicherstruktur von ARIS zu berücksichtigen. Auf M1-Ebene sind Objekttypen als Objektdefinitionen, Kantentypen als Kantendefinitionen, Symbole als Objektausprägungen und Linien als Kantenausprägungen ausgeprägt. Für die Transformation auf M1-Ebene resultiert hieraus, dass beispielsweise Objektdefinitionen auf EClasses abzubilden sind. Die EClass auf M1-Ebene ist eine Ausprägung der EClass (M2-Ebene), die zu dem entsprechenden Objekttyp der Objektdefinition korrespondiert. Die Transformation von ARIS-Modellen zu EMF-Modellen wurde ebenfalls mit einem ARIS-Skript realisiert. Dieses Skript erzeugt in Abhängigkeit eines Methodenfilters EMF-

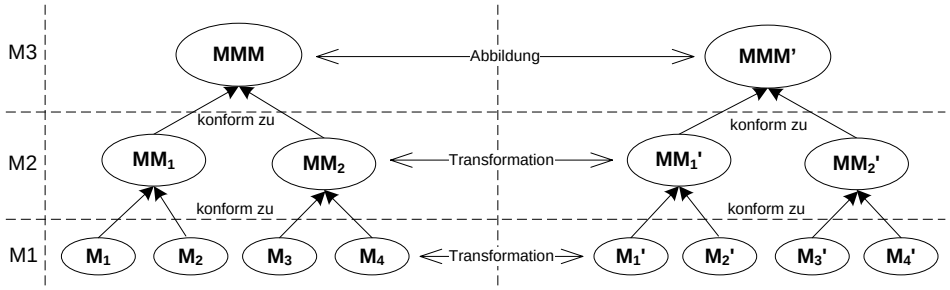


Abbildung 3: Ansatz für eine M3-Ebene-Brücke

Modelle im XMI-Format, welche konform zum Methodenfilter korrespondierenden EMF-Metamodell sind.

Im Gegensatz zur Transformation auf M2-Ebene können ARIS-Modelle nicht nur nach EMF exportiert, sondern auch wieder in ARIS importiert werden. Zum Erzeugen von ARIS-Modellen aus einem EMF-Modell wird mit Hilfe von XPand von openArchitectureWare (oAW) [Völ07] ein ARIS-Skript generiert, welches Kommandos für das Erstellen der ARIS-Modelle enthält. Die ID-Vergabe von Modellelementen erfolgt somit innerhalb des ARIS-Raums. Die eben beschriebene Brücke ist in Abbildung 4 in der Übersicht dargestellt. Eine ausführlichere Beschreibung der Transformationen ist in [KK07] zu finden.

5 Anwendungsmöglichkeiten

Die Verarbeitungsmöglichkeiten von ARIS-Modellen auf Basis von Modell-zu-Modell-Transformationen (M-zu-M-Transformation) sind vielfältig. Der EMF-Raum bietet sich für diese Aufgabe an, da eine Reihe von Transformationswerkzeugen für EMF existiert. Beispiele für solche Werkzeuge bzw. Transformationssprachen sind die ATLAS Transformation Language (ATL) [JK05], das Model Transformation Framework (MTF) [IBM04], XTend [Eff06] von oAW oder das Tiger EMF Transformation Project [BEK+06]. Diese Transformationswerkzeuge basieren auf unterschiedlichen Transformationsansätzen, wie dem Graph-basierten, relationalen, operationalen oder Template-basierten Ansatz.

Praktische Anwendungsfälle für M-zu-M-Transformation von ARIS-EPK-Modellen lassen sich hinsichtlich des Abstraktionslevels des Quell- und Ziel-Modells klassifizieren. Beispiele sind:

- die Transformation in eine Modellierungssprache auf niedrigerem Abstraktionsniveau (technische Verfeinerung, Synthese): z. B. Transformation von EPK-Modellen nach BPEL4WS,
- die Transformation in eine Modellierungssprache auf höherem Abstraktionsniveau (Reverse Engineering): z. B. Transformation von EPK-Modellen nach Wertschöpfungsdiagrammen,

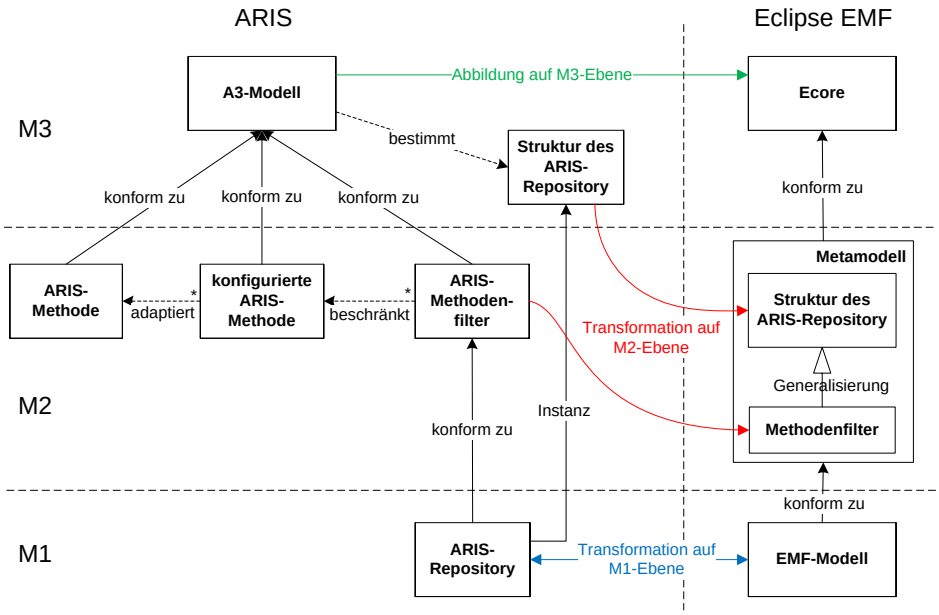


Abbildung 4: Überblick über die ARIS-EMF-Brücke

- die Transformation in eine andere Modellierungssprache auf gleichem Abstraktionsniveau (Migration): z. B. Transformation von EPK-Modellen nach UML-Aktivitätsdiagrammen,
- die Restrukturierung: z. B. Transformation von EPK-Modellen in normalisierte EPK-Modelle,
- die Extraktion bestimmter Modell-Eigenschaften (Analyse), wie die Berechnung von Prozessmetriken für EPK-Modelle oder die Überprüfung der syntaktischen Korrektheit.

Die syntaktische Überprüfung von EPKs mit Hilfe einer M-zu-M-Transformation wird im folgenden Abschnitt anhand der Transformationssprache ATL demonstriert.

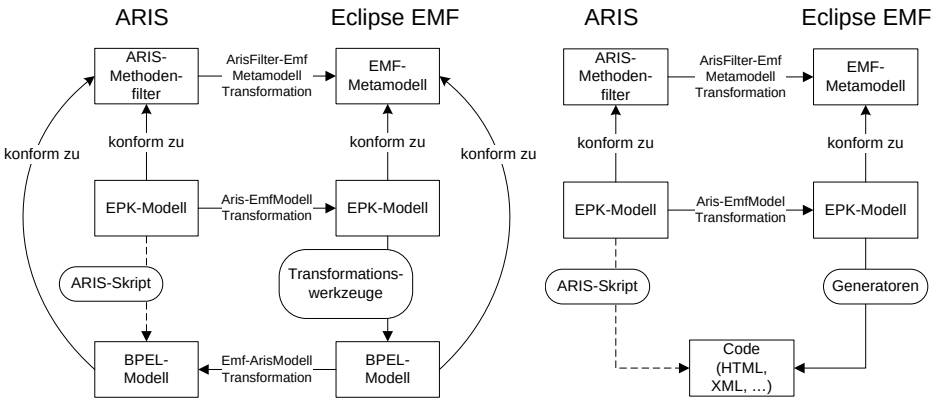
Eine weitere Möglichkeit für die Verarbeitung von ARIS-Modellen ist die Modell-zu-Text-Transformation (M-zu-T-Transformation). Im Gegensatz zur M-zu-M-Transformation, bei der das Ausgabeartefakt konform einem Ziel-Metamodell ist, liegt bei der M-zu-T-Transformation keine explizite Syntaxdefinition der Zielsprache vor bzw. wird bei der Transformation nicht berücksichtigt. Die Ausgabeartefakte sind textuell, beispielsweise Java-Code oder HTML-Code. Beispiele für Werkzeuge, die eine M-zu-T-Transformation in EMF unterstützen, sind Java Emitter Templates (JET) oder XPand [Eff06] von oAW. Ein möglicher Anwendungsfall für ARIS-EPK-Modelle ist deren Repräsentation in HTML-Form.

Neben der Transformation von ARIS-Modellen im EMF-Raum sind noch weitere Modelloperationen [Béz05] möglich, wie der Vergleich von Modellen, die Verwebung von ARIS-Modellen oder die Differenzenbildung von ARIS-Modellen. Für die Modellverwebung kann beispielsweise der ATLAS Model Weaver (AMW) [FBJ+05] eingesetzt werden und für den Modellvergleich EMF Compare [BMT07] aus dem Eclipse Modeling Framework Technology Project.

Die genannten Verarbeitungsmöglichkeiten von ARIS-Modellen im EMF-Raum erfordern den Einsatz der ARIS-EMF-Brücke. Dabei kann zwischen zwei unterschiedlichen Szenarien unterschieden werden. Zum einem kann die Brücke eingesetzt werden, um ARIS-Modelle in den EMF-Raum zu exportieren, dort zu verarbeiten und anschließend wieder in den ARIS-Raum zu importieren.

Ein solches Szenario ist in Abbildung 5(a) beispielhaft dargestellt. Hierbei wird ein Methodenfilter für EPK- und BPEL-Modelle zu Grunde gelegt. Die technische Verfeinerung von EPKs zu korrespondierenden BPEL-Modellen kann im ARIS-Raum durch ARIS-Skript realisiert werden. Die Nutzung der Brücke ermöglicht alternativ den Einsatz spezieller Transformationswerkzeuge. Erfahrungen zeigen, dass für diese Aufgabe auf Modelltransformation spezialisierte Werkzeuge im Vergleich zu transformationsunspezifischen imperativen Ansätzen Verbesserungspotenziale hinsichtlich Kompaktheit, Modularisierbarkeit, Anpassbarkeit und Komponierbarkeit eröffnen.

Im zweiten Anwendungsszenario der Brücke (siehe Abbildung 5(b)) werden ARIS-Modelle aus ARIS exportiert. Auf einen Re-Import der Modelle wird verzichtet. Die Modelle werden zur Verarbeitung in den EMF-Raum exportiert, in dem die Export-Funktion alternativ zu ARIS-Skript implementiert wird. Dabei können sowohl M-zu-M-Transformations- oder M-zu-T-Transformationswerkzeuge (Generatoren) zum Einsatz kommen.



(a) ARIS zu EMF mit Re-Import

(b) ARIS zu EMF

Abbildung 5: Anwendungsszenarien der ARIS-EMF-Brücke

Tabelle 1: EPK-Methodenfilter – Objekttypen, Symbole und Linien des EPK-Modelltyps

(a) Objekttypen und Symbole

Objekttyp	**Symbol**
Ereignis	Ereignis
Funktion	Funktion

(b) Linien und deren Symbole

Quellsymbol	**Linie**	**Zielsymbol**
Ereignis	aktiviert	Funktion
Funktion	erzeugt	Ereignis

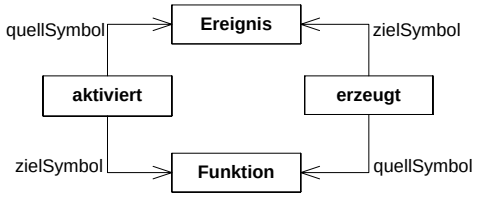


Abbildung 6: Ausschnitt aus dem EMF-Metamodell des EPK-Methodenfilters

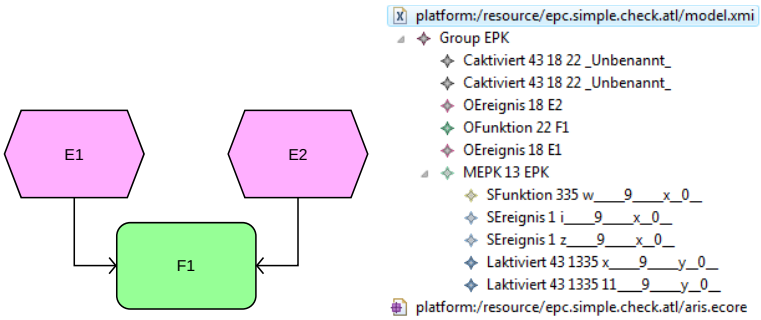
6 Syntaxüberprüfung von EPKs mit ATL

In diesem Kapitel wird auf die Analyse, speziell die Syntaxüberprüfung, von ARIS-EPK-Modellen mit ATL eingegangen. Hierzu wird zunächst die Anwendung der Brücke und anschließend die Syntaxüberprüfung demonstriert.

Für die Anwendung der ARIS-EMF-Brücke ist zunächst ein entsprechender Methodenfilter auszuwählen bzw. zu definieren, der es ermöglicht, EPK-Modelle zu erstellen. Dieser Methodenfilter besteht aus den in Tabelle 1 aufgelisteten Objekttypen, Symbolen und Linien. Zur Vereinfachung und besseren Beschreibung wurde auf die Symbole UND-Regel, ODER-Regel und XOR-Regel verzichtet. Dieser Methodenfilter wird mit dem ARIS-Skript „ArisFilter-EmfMetamodell-Transformation“ zu einem EMF-Metamodell transformiert. Ein Ausschnitt aus dem erzeugten EMF-Metamodell, welcher die Beziehungen zwischen den Symbolen darstellt, ist in Abbildung 6 abgebildet. Als nächstes können ARIS-EPK-Modelle mit der „Aris-EmfModell-Transformation“ zu einem entsprechenden EMF-Modell transformiert werden. Zur Demonstration der Transformation ist in Abbildung 7(a) eine syntaktisch nicht korrekte EPK in ARIS und die dazugehörige EPK in EMF in Abbildung 7(b) dargstellt. Ausgehend von den Regeln für korrekte EPKs, die in [NR02] beschrieben sind, werden die folgenden zwei Regeln verletzt:

1. Funktionen besitzen genau eine eingehende und genau eine ausgehende Kante.
2. Es gibt mindestens ein Start- und mindestens ein Endereignis.

Die Implementierung der beiden Regeln in der ATLAS Transformation Language wird im Folgenden beschrieben. Eine Transformationsdefinition in ATL wird als Modul bezeichnet. Dieses enthält einen Header, Import-Anweisungen, Helper und Regeln. Der Header deklariert das Quell- und Zielmodell, wobei OUT das Zielmodell und IN das Quellmodell bezeichnet. Das Quellmodell und das dazugehörige Metamodell sind das nach EMF



(a) Eine EPK als ARIS-Modell (b) Eine EPK als EMF-Modell

Abbildung 7: Eine syntaktisch nicht korrekte EPK in ARIS und EMF

überführte ARIS-EPK-Modell und der EPK-Methodenfilter. Das Zielmodell wird während der Transformation erstellt. Das dazugehörige Metamodell „Problem“ besteht im Wesentlichen aus der EClass „Problem“ und den drei Attributen „description“, „severity“ und „location“. Mit Import-Anweisungen können beispielsweise andere Module importiert werden. Dies wird allerdings nicht benötigt. Helper sind Hilfsoperationen, die in einer Regel verwendet werden können. In Abbildung 8 Zeile 3 ist ein Helper beschrieben, der aus einem ARIS-Modell alle Symbole selektiert und als Menge zurückliefert. Für die vollständige Implementierung werden weitere Helper benötigt, die alle Linien, Zielsymbole, Quellsymbole, Ereignis-Symbole, Startereignis-Symbole, Endereignis-Symbole, ausgehende Linien von Symbolen und eingehende Linien von Symbolen selektieren. Diese sind analog dem ersten Helper und werden deswegen nicht näher beschrieben.

In ATL gibt es zwei Arten von Transformationsregeln, zum einen die so genannte Called-Rule und zum anderen die Matched-Rule. Called-Rules können manuell aufgerufen werden. Sie werden in diesem Beispiel nicht benötigt und deswegen nicht weiter beschrieben. Hingegen wird bei einer Matched-Rule eine Bedingung spezifiziert, die festlegt, wann eine Regel ausgeführt wird. Die Regelausführungsstrategie wird dabei von ATL selbst übernommen. Die Bedingung einer Regel wird im ”from“-Teil spezifiziert, wobei zunächst festgelegt wird, auf welches Metamodellelement die Regel zutrifft. Anschließend kann optional durch einen OCL-ähnlichen Ausdruck diese Bedingung verfeinert werden. Im „to“-Teil einer Regel können dann entsprechend die Elemente im Zielmodell erstellt, wobei Attribute des Quellmodellelements an die Attribute des Zielmodellelements gebunden werden können.

In Abbildung 8 ab Zeile 8 sind die implementierten Regeln dargestellt. Die erste Regel „NoEndEvent“ wird angewendet, wenn die Menge der Endsymbole in einem EPK-Modell leer ist. Es wird ein Problemelement mit einer Beschreibung, dem Schwierigkeitsgrad und einer Stellenbeschreibung erstellt. Die danach folgenden drei Regeln sind analog der ersten Regel. „NoStartEvent“ wird bei einer leeren Menge von Startsymbolen in einem EPK-Modell angewendet. Die Dritte wird angewendet, wenn ein Funktionssymbol mehrere Eingangskanten besitzt und die Vierte, wenn ein Funktionssymbol mehrere Ausgangskanten hat. Bei der Ausführung der Transformationsdefinition finden die Regeln entsprechend

```
1  create OUT : Problem  from IN : Aris;

3  helper context Aris!Model def : getAllSymbols() :
     Set(Aris!Symbol) = self.containSymbols -> asSet();
5
   ...
7
   nodefault rule NoEndEvent {
9    from i : Aris!M_EPK_13(i.getAllEndEventSymbols()->isEmpty())
     to o : Problem!Problem(
11       description <- 'No end event exists.',
         severity <- #error,
13       location <- 'model name: ' + i.name + ' --- model id: ' + i.id)
   }
15
   nodefault rule NoStartEvent {
17   from i : Aris!M_EPK_13(i.getAllStartEventSymbols()->isEmpty())
     to -- ähnlich wie bei NoEndEvent
19 }

21 nodefault rule FunctionInput {
     from i : Aris!S_Funktion_335(i.getInputEdgesOfSymbol()->size() <> 1)
23   to -- ähnlich wie bei NoEndEvent
   }
25
   nodefault rule FunctionOutput {
27   from i : Aris!S_Funktion_335(i.getOutputEdgesOfSymbol()->size() <> 1)
     to -- ähnlich wie bei NoEndEvent
29 }
```

Abbildung 8: Implementierung in ATL

Anwendung und es entsteht ein Problemmodell mit den zwei Problemen, dass es kein Endereignis und eine Funktion mit mehr als einer Eingangskante gibt.

7 Zusammenfassung

In diesem Beitrag wurde eine Brücke zwischen den technischen Räumen ARIS und EMF vorgestellt. Hierzu wurden die Sprachdefinitionsmöglichkeiten beider Räume untersucht und zueinander in Beziehung gesetzt. Darauf aufbauend konnten Transformationen auf Metamodell- und Modellebene definiert werden. Die resultierende Brücke erhöht die Interoperabilität zwischen beiden Räumen und ermöglicht die Verarbeitung von ARIS-Modellen im EMF-Raum.

Am Beispiel der EPK wurde das Anwendungspotenzial von EMF-Werkzeugen skizziert und am Beispiel der Syntaxüberprüfung demonstriert. Der Raum-Wechsel von ARIS nach EMF lohnt sich genau dann, wenn auf Basis zur Verfügung stehender Werkzeuge Modellverarbeitungsaufgaben im EMF-Raum effizienter umgesetzt werden können. Die unter-

schiedlichen Transformationsmöglichkeiten bieten Verbesserungspotenziale hinsichtlich Kompaktheit, Modularisierbarkeit, Anpassbarkeit und Komponierbarkeit der Implementierungen.

Die in diesem Beitrag vorgestellte Brücke demonstriert die prinzipielle Funktionsfähigkeit des gewählten Ansatzes. Die Brücke lässt sich in verschiedener Hinsicht verbessern und erweitern. Beispielsweise wurde die Brücke hinsichtlich der Generierung graphischer Modellierungswerkzeuge (auf Basis des Eclipse Graphical Modeling Frameworks) erweitert, was die graphische Visualisierung von ARIS-Modellen im EMF-Raum ermöglicht.

Die Autoren sehen Anwendungsmöglichkeiten der ARIS-EMF-Brücke sowohl in wissenschaftlicher als auch in praktischer Hinsicht. Für konkrete Aussagen, welche EMF-Werkzeuge sich für welche EPK-Verarbeitungsaufgaben eignen, bedarf es jedoch eingehenderer Untersuchungen.

Literatur

[BBC+05] Jean Bézivin, Christian Brunette, Régis Chevrel, Frédéric Jouault und Ivan Kurtev. Bridging the Generic Modeling Environment (GME) and the Eclipse Modeling Framework (EMF). In *Proceedings of the Best Practices for Model Driven Software Development at OOPSLA'05*, San Diego, California, USA, 2005.

[BDD+05] Jean Bézivin, Vladan Devedzic, Dragan Djuric, Jean-Marie Favreau, Dragan Gasevic und Frédéric Jouault. An M3-Neutral infrastructure for bridging model engineering and ontology engineering. In *Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'2005)*, Seiten 159–171. Springer-Verlag, 2005.

[BEK+06] Enrico Biermann, Karsten Ehrig, Christian Köhler, Günter Kuhns, Gabriele Taentzer und Eduard Weiss. Graphical Definition of In-Place Transformations in the Eclipse Modeling Framework. In Oscar Nierstrasz, Jon Whittle, David Harel und Gianna Reggio, Hrsg., *Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006*, Jgg. 4199 of *LNCS*, Seiten 425–439. Springer, 2006.

[BHJ+05] Jean Bézivin, Guillaume Hillairet, Frédéric Jouault, Ivan Kurtev und William Piers. Bridging the MS/DSL Tools and the Eclipse Modeling Framework. In *Proceedings of the International Workshop on Software Factories at OOPSLA 2005*, San Diego, California, USA, 2005.

[BMT07] Cedric Brun, Jonathan Musset und Antoine Toulme. EMF Compare. `http://wiki.eclipse.org/index.php/EMF_Compare`, 2007.

[BSM+04] Frank Budinsky, David Steinberg, Ed Merks, Ray Ellersick und Timothy J. Grose. *Eclipse Modeling Framework*. the eclipse series. Addison Wesley, 2004.

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

[Eff06] Sven Efftinge. OpenArchitectureWare 4.1: Extend Language Reference. Bericht, openArchitectureWare.org, `http://www.eclipse.org/gmt/oaw/doc/4.1/r25_extendReference.pdf`, 2006.

[FBJ+05] Marcos Didonet Del Fabro, Jean Bézivin, Frédéric Jouault, Erwan Breton und Guillaume Gueltas. AMW: a generic model weaver. In *Proceedings of the 1ère Journée sur l'Ingénierie Dirigée par les Modèles (IDM05)*, 2005.

[IBM04] IBM alphaWorks. Model Transformation Framework (MTF). http://www.alphaworks.ibm.com/tech/mtf, 2004.

[IDS07] IDS Scheer AG. ARIS Platform. http://www.ids-scheer.com/products, 2007.

[JK05] Frédéric Jouault und Ivan Kurtev. Transforming Models with ATL. In *Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005*, Montego Bay, Jamaica, 2005.

[KBA02] Ivan Kurtev, Jean Bézivin und Mehmet Aksit. Technological Spaces: an Initial Appraisal. In *CoopIS, DOA'2002*, 2002.

[KK07] Heiko Kern und Stefan Kühne. Model Interchange between ARIS and Eclipse EMF. In Juha-Pekka Tolvanen, Jeff Gray, Matti Rossi und Jonathan Sprinkle, Hrsg., *7th OOPSLA Workshop on Domain-Specific Modeling at OOPSLA 2007*, 2007.

[KNS92] G. Keller, M. Nüttgens und A.-W. Scheer. Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Bericht Heft 89, Institut für Wirtschaftsinformatik, Universität des Saarlandes, Saarbrücken, 1992.

[NR02] Markus Nüttgens und Frank J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In J. Desel und M. Weske, Hrsg., *Promise 2002 – Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen, Proceedings des GI-Workshops und Fachgruppentreffens (Potsdam, Oktober 2002)*, Jgg. P-21 of *LNI*, Seiten 64–77, Bonn, 2002.

[Sch91] August-Wilhelm Scheer. *Architektur integrierter Informationssysteme – Grundlagen der Unternehmensmodellierung*. Springer, Berlin, Heidelberg, 1991.

[Sch98a] August-Wilhelm Scheer. *ARIS – Modellierungsmethoden, Metamodelle, Anwendungen*. Springer, 1998. (German).

[Sch98b] August-Wilhelm Scheer. *ARIS – Vom Geschäftsprozess zum Anwendungssystem*. Springer, 1998. (German).

[STA05] August-Wilhelm Scheer, Oliver Thomas und Otmar Adam. Process Modeling Using Event-Driven Process Chains. In Marlon Dumas, Wil M. P. van der Aalst und Arthur H. M. ter Hofstede, Hrsg., *Process-Aware Information Systems – Bridging People and Software through Process Technology*. Wiley, 2005.

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

[Völ07] Markus Völter. openArchitectureWare 4 – the flexible open source tool platform for model-driven software development. Bericht, openArchitectureWare, 2007.