Auswahl von Open Source Tools zur modellgetriebenen Simulationserzeugung

Bernd Scholz-Reiter, Daniel Rippel, Steffen Sowade und Torsten Hildebrandt, Universität Bremen

Prof. Dr.-Ing. Bernd Scholz-Reiter ist Professor für Planung und Steuerung produktionstechnischer Systeme am Fachbereich Produktionstechnik der Universität Bremen und Institutsleiter des BIBA - Bremer Institut für Produktion und Logistik GmbH.

Dipl.-Inf. Daniel Rippel arbeitet als wissenschaftlicher Mitarbeiter am BIBA im Bereich Intelligente Produktions- und Logistiksysteme.

Dipl.-Ing. Steffen Sowade arbeitet als wissenschaftlicher Mitarbeiter am BIBA im Bereich Intelligente Produktions- und Logistiksysteme.

Dipl.-Wirtsch.-Inf. Torsten Hildebrandt arbeitet als wissenschaftlicher Mitarbeiter im Fachgebiet Planung und Steuerung produktionslogistischer Systeme an der Universität Bremen.

Die Autonomous Logistic Engineering Methodology (ALEM), die im Rahmen des Sonderforschungsbereichs „Selbststeuerung logistischer Prozesse – ein Paradigmenwechsel und seine Grenzen“ entwickelt wird, bietet verschiedene Werkzeuge zur Modellierung selbststeuernder logistischer Systeme. Zur Auswertung der Modelle wird ALEM um eine Simulation erweitert. Um die ALEM-Modelle simulierbar zu machen, müssen sie aufbereitet werden. Dazu werden die ALEM-Modelle mithilfe von Konzepten der Model Driven Architecture in ausführbare Simulationsmodelle transformiert. Zur Durchführung dieser Transformation wurden verschiedene Open Source Tools untersucht.

Kontakt:
BIBA – Bremer Institut für Produktion und Logistik GmbH
Hochschulring 20
28359 Bremen
Tel.: 0421 / 218-5538
E-Mail: rip@biba.uni-bremen.de
URL: http://www.biba.uni-bremen.de

Heutige Logistiksysteme stehen vor der Herausforderung steigender Komplexität und Dynamik. Eine Möglichkeit damit umzugehen, ist der Einsatz von Selbststeuerung. Diese charakterisiert sich durch die Zerlegung komplexer Planungs- und Entscheidungsprobleme in kleinere, dezentrale Teilprobleme. Der Sonderforschungsbereich „Selbststeuerung logistischer Prozesse – ein Paradigmenwechsel und seine Grenzen“ untersucht die Vorteile und die Einschränkungen des Einsatzes von Selbststeuerung in logistischen Systemen.

Selbststeuerung

Selbststeuerung beruht auf den Konzepten Autonomie und Selbstorganisation und wird in verschiedenen Forschungsgebieten untersucht. Im Kontext logistischer Systeme definieren Hülsmann und Windt Selbststeuerung als „ [...] Prozesse dezentraler Entscheidungsfindung in heterarchischen Strukturen. Sie setzt voraus, dass interagierende Elemente in nichtdeterministischen Systemen die Fähigkeit und Möglichkeit zum autonomen Treffen von Entscheidungen besitzen. Ziel des Einsatzes von Selbststeuerung ist eine höhere Robustheit und positive Emergenz des Gesamtsystems durch eine verteilte, flexible Bewältigung von Dynamik und Komplexität“ [1].

Ein oft untersuchtes Beispiel für ein produktionslogistisches Szenario ist das 3x3-Maschinen-Modell (Bild 1) von Scholz Reiter u.a. [2]. Dabei handelt es sich um eine Werkstattfertigung, in der die Werkstücke drei Produktionsebenen (Fräsen, Bohren, Stanzen) durchlaufen müssen. Auf jeder dieser Ebenen gibt es drei gleichartige Maschinen. Im klassischen Fall übernimmt ein zentrales Planungssystem oder ein Mensch die Belegungsplanung der Maschinen. Jedem Auftrag wird auf jeder Ebene eine Maschine zugewiesen. Sollte auf einer der Maschinen eine Fehlfunktion auftreten, muss die Planung neu durchgeführt werden.

Unter Anwendung der Selbststeuerung wird die Planungskompetenz auf die einzelnen Werkstücke verteilt. Sobald sie das Eingangslager verlassen, können sie selbstständig entscheiden welche der Maschinen einer Produktionsebene sie ansteuern wollen. Jedes Werkstück besitzt ein eigenes Zielsystem, das ihm beispielsweise vorschreibt möglichst schnell oder möglichst preiswert durch die Produktion zu gehen. Tritt bei einer Maschine eine Fehlfunktion auf, können die Werkstücke dynamisch darauf reagieren und die Bearbeitung an einer anderen gleichartigen Maschine anfragen.

Um die Umsetzung autonomer Systeme zu unterstützen wird die Autonomous Logistic Engineering Methodology (ALEM) entwickelt [3]. Das Framework bietet eine auf der Unified Modeling Language (UML) basierende Notation, ein Vorgehensmodell sowie Softwareunterstützung zur Modellierung selbststeuernder logischer Systeme. ALEM wurde mit dem Ziel entworfen, dass Prozessexperten in die Lage versetzt werden, selbststeuernde Systeme abzubilden und zu entwerfen.

Die prototypische Umsetzung basiert auf verschiedenen Open Source Fra-

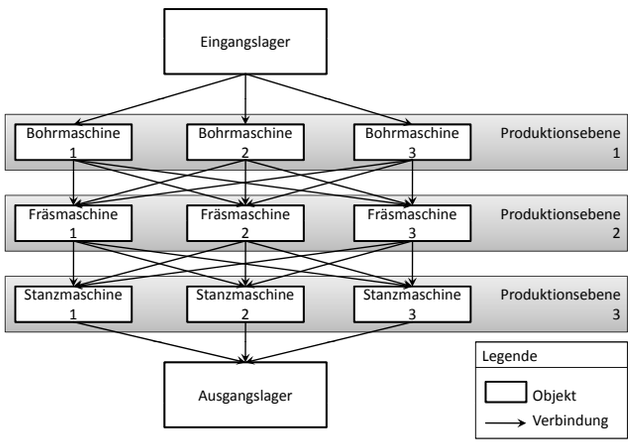

Bild 1: Ein 3x3-Maschinen-Modell (nach [2]).

meworks der Eclipse Plattform (http://www.eclipse.org). Die Beschreibung der ALEM-Modelle wurde mithilfe des Eclipse Modeling Frameworks (EMF) realisiert. Mittels des Graphical Modelling Framework (GMF) wurden grafische Editoren generiert. Diese ermöglichen ein leichtes Erstellen von Modellen in ALEM [4].

Simulation

Um die mit ALEM erstellten Modelle auszuwerten, wird ALEM um eine Simulationskomponente erweitert. Somit kann unabhängig von einer physischen Umsetzung des Systems geprüft werden, wie gut es funktioniert. Fehler in der Konzeption können frühzeitig aufgedeckt werden. Zudem lassen sich mithilfe der Simulation Kennzahlen, wie z.B. die Produktivität eines Systems bestimmen, welche mit den Werten bereits existierender Systeme verglichen werden können. Die Modelle und Simulationsergebnisse erlauben es, Kostenabschätzungen für die Einführung von Selbststeuerung anzufertigen.

Die Struktur eines Simulationsmodells unterscheidet sich grundlegend von der eines ALEM-Modells. Aufgrund der Gleichartigkeit vieler in ALEM verwendeter Diagramme, sieht die Modellstruktur eine semantische Unterteilung der Daten vor. Zum Beispiel werden die Inhalte der unterschiedlichen Klassendiagramme, die z.B. das Wissen von Objekten oder ausgetauschte Nachrichten darstellen, in separaten Untermodellen gehalten und nicht (wie es z.B. in der UML üblich ist) in einem Datenpool abgelegt. Im Gegensatz dazu sind Simulationsmodelle anhand der zu simulierenden Objekte (Prozesse, Ressourcen, usw.) strukturiert. Auch können zusätzliche Informationen notwendig sein, die eventuell nicht im ALEM-Modell vorhanden sind. Die ALEM-Modelle müssen dementsprechend aufbereitet werden um simulierbar zu sein. Hierzu bietet die Model Driven Architecture (MDA) verschiedene Konzepte, die für eine solche Anpassung genutzt werden können.

Modelltransformation

Die MDA verfolgt die Annahme, dass auf Basis von Modellen mittels automatisierter Modelltransformationen ausführbare Programme erzeugt werden können [5]. Zuerst werden allgemeine grafische Beschreibungen des Programms, z.B. in UML, angefertigt. Diese grafischen Modelle werden mithilfe von Transformationen in ausführbare Formen, z.B. in Quelltexte übersetzt. Die Transformationen können mithilfe der im „Query/View/Transformation“ (QVT) Standard [6] spezifizierten Sprachen umgesetzt werden. Diese können verwendet werden, um ALEM-Modelle in ausführbare Simulationsmodelle zu überführen. Da die Struktur eines Simulationsmodells stark von der gewählten Simulationsplattform abhängt, wird ein dreistufiger Transformationsansatz verwendet (Bild 2), der ein konzeptionelles Simulationsmodell mit einbezieht. Dieses entspricht der allgemeinen Form des Zielmodells, spart aber Details der Zielplattform aus. Zum Beispiel kann dieses Modell eine plattformunabhängige Beschreibung der zu simulierenden Agenten enthalten, wenn es sich bei der Zielplattform um eine Multi Agenten Simulation handelt. Im Falle einer Materialflusssimulation könnten Ressourcen und Güter die Hauptelemente sein. Der Vorteil dieses Zwischenschritts ist, dass ein Benutzer sich so mit Konzepten wie Gütern, Bohr- oder Fräsmaschinen auseinander setzen kann und nicht mit simulationsspezifischen Elementen wie z.B. Agenten.

Im ersten Schritt werden die vorhandenen Informationen des ALEM-Modells restrukturiert um der generellen Form der Zielplattform zu entsprechen. Im zweiten Schritt werden fehlende Informationen erhoben und die Simulationsobjekte instanziiert. Schließlich wird dieses Modell weiter verfeinert, um einer speziellen Simulationsplattform zu entsprechen.

Open Source Tools zur Modelltransformation

Die Umsetzung der Transformationsschritte erfordert den Einsatz unterschiedlicher Werkzeuge. Es existiert eine Vielzahl von Open Source Tools, die für die einzelnen Schritte eingesetzt werden können. Hierbei ist es wichtig, dass die Werkzeuge direkt in ALEM integrierbar sind. Daher werden Lösungen bevorzugt, die direkt mit der Implementierung verknüpft werden können, ohne dass weitere Benutzerinteraktionen notwendig sind.

Insbesondere die Restrukturierung und die Verfeinerung können mithilfe von MDA Konzepten und Sprachen realisiert und automatisiert werden. Der zweite Schritt, die Instanziierung, erfordert dagegen ein hohes Maß an Interaktion mit dem Benutzer. Für jeden der Schritte werden im Folgenden Möglichkeiten diskutiert, wie und mithilfe welcher Werkzeuge der jeweilige Schritt umgesetzt werden kann.

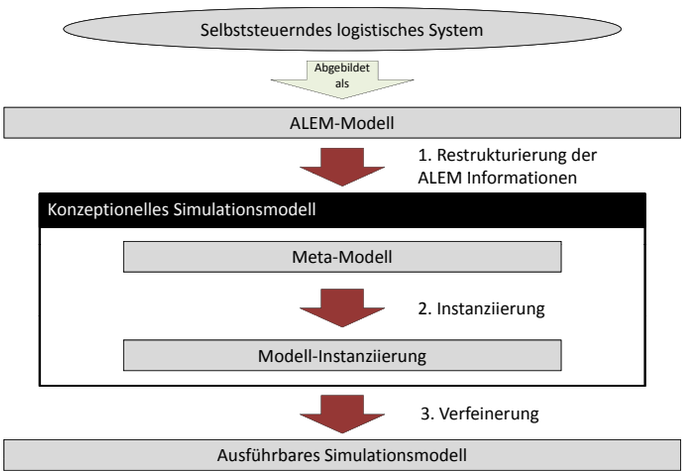

Bild 2: Transformationskonzept.

Restrukturierung

Der erste Transformationsschritt lässt sich mithilfe der QVT Sprachen realisieren, da sowohl das ALEM Quellmodel, als auch das konzeptionelle Zielmodell in derselben Metasprache verfasst sind. Es existieren verschiedene Open Source Implementierungen der QVT-Sprachen für Eclipse. Betrachtet wurden die Sprachen QVT-Procedural, die Atlas Transformation Language (ATL) und MediniQVT. Im Folgenden wird jede dieser Sprachen kurz vorgestellt.

QVT-Procedural basiert auf dem Konzept von Abbildungen zwischen Modellelementen. Für ein Element des Quellmodells wird angegeben welches Element des Zielmodells erzeugt werden soll. Weitere Bedingungen können genutzt werden, um die Anwendung einer Abbildung auf Elemente mit bestimmten Eigenschaften einzuschränken.

Das ALEM-Modell verwendet gleichartige Elemente in verschiedenen semantischen Kontexten. Zum Beispiel werden Klassen genutzt um logistische Objekte, das Wissen von logistischen Objekten oder um Nachrichten darzustellen. Dies macht es notwendig. den jeweiligen Kontext eines Elements, z.B. einer Klasse, zu ermitteln. Aufgrund dessen, dass die zusätzlichen Bedingungen hauptsächlich auf Eigenschaften der Elemente abzielen, ist es schwierig, einen übergeordneten Kontext abzufragen. Auch wenn die Nutzung von QVT-Procedural theoretisch möglich wäre, wird diese Sprache ausgeschlossen.

ATL stellt sowohl eine Ausführungsengine, als auch Editoren für die Modelltransformationen bereit. Die Syntax dieser Sprache ist regelbasiert. Ähnlich wie QVT-Procedural, bildet jede Regel eine Abbildung von einem Element des Quellmodells auf Elemente des Zielmodells. Im Unterschied zu QVT-Procedural erlaubt ATL es, zusätzlich zu den Bedingungen Funktionen zu benutzen, um weitere Informationen zu berechnen. Diese Funktionen können genutzt werden, um den Kontext eines Elements abzufragen und so festzustellen, ob es sich z.B. bei einer Klasse um eine Nachricht oder ein logistisches Objekt handelt. Allerdings führen diese zusätzlichen Funktionen zu schwer nachvollziehbaren Transformationen, die nur unter bestimmten Bedingungen gültig sind. Aus diesem Grund wurde auch ATL ausgeschlossen.

MediniQVT bietet eine Ausführungsengine sowie Editoren für die Transformationen. Das Grundkonstrukt dieser Sprache sind Relationen. Eine Relation beschreibt in diesem Fall die Beziehung zwischen verschiedenen Modellelementen. Der Vorteil dieser Sprache liegt darin, dass eine Relation nicht auf nur ein Element abzielt, sondern dass mehrere Elemente und deren Beziehungen untereinander abgefragt werden können. Dies erlaubt es auf klar strukturierte Weise, Kontextinformationen und Eigenschaften der einzelnen Elemente zu prüfen. Aus diesem Grund wurde MediniQVT ausgewählt, um den ersten Transformationsschritt umzusetzen.

Instanziierung

Der zweite Transformationsschritt, die Instanziierung, dient dazu, fehlende Informationen vom Benutzer abzufragen. Entsprechend muss eine geeignete Schnittstelle geschaffen werden, die es einem Nutzer erlaubt, die fehlenden Informationen auf intuitive Art und Weise hinzuzufügen. Da die ALEM-Modellierungsmethode auf grafischen Diagrammen beruht, wurde auch hier eine grafische Lösung genutzt. Im Gegensatz zu den anderen Schritten standen hier keine unterschiedlichen Tools zur Auswahl.

Wie auch bei anderen ALEM-Diagrammen, wird ein grafischer GMF-Editor genutzt, um das konzeptionelle Simulationsmodell zu instanziieren. Während der Umstrukturierung im ersten Schritt entsteht eine Auswahl von logistischen Objekten. Diese müssen einerseits räumlich in einem Szenario angeordnet werden, andererseits müssen ihre Eigenschaften festgelegt werden.

Bild 3 zeigt die prototypische Implementierung des Editors. Die Palette auf der rechten Seite zeigt alle Objekte, die als selbststeuernde logistische Objekte umgesetzt wurden. In diesem 3x3-Maschinen-Modell wurden Maschinen und Güter als selbststeuernde Objekte definiert. Die Zeichenfläche erlaubt ein räumliches Anordnen dieser Objekte. Im unteren Bereich ist ein Teil des Eigenschaftseditors zu sehen. Hier können die zugewiesenen Eigenschaften instanziiert werden. Für eine Maschine wird z.B. die Eingabe verlangt, welche Arbeitsschritte sie durchführen kann und wie der anfängliche Belegungsplan ist.

Verfeinerung

Die Verfeinerung des instanziierten Modells fügt plattformspezifische Informationen hinzu. Zudem wird der syntaktische Aufbau des Modells der Zielplattform angepasst. Je nachdem welche Form das ausführbare Simulationsmodell haben soll, können andere Tools notwendig sein. Eine EMF basierte Simulation

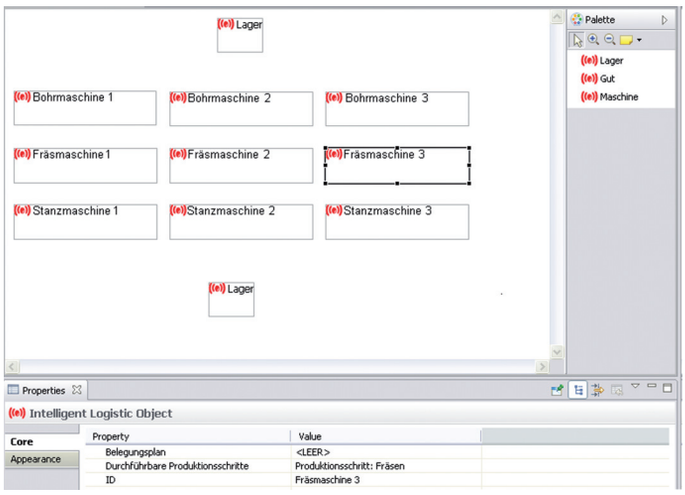

Bild 3: ALEM Layout Editor mit markierter Maschine.

kann dieselben Tools nutzen, die im ersten Transformationsschritt angewendet wurden. Wenn die Zielmodelle als Programmcode oder als XML erzeugt werden müssen, können andere Tools herangezogen werden. Im Rahmen der Eclipse Plattform existieren drei vorlagenbasierte Technologien, die dazu geeignet sind.

Alle drei Technologien basieren auf Skripten, die bestimmte Modellelemente in Text übersetzen. Grundsätzlich werden im Skript definierte Texte erzeugt, die mit dynamischem Inhalt versehen werden können. Diese dynamischen Inhalte werden über in das Skript eingebetteten Programmcode generiert. Die drei Technologien unterscheiden sich in ihrer Syntax, sowie anhand der verwendeten Skriptsprache.

Für die Automatisierung dieses Transformationsschritts wird eine Vorlagenbibliothek in ALEM integriert. Diese enthält Vorlagen für die Umsetzung von Funktionen und Modellelementen für bestimmte Simulationsplattformen. Für spezielle Steuerungsstrategien und Szenarien kann es notwendig sein, dass der Benutzer die Vorlagen anpasst und somit die Bibliothek erweitert. Um diese Anpassung so leicht wie möglich zu halten, wurden Java Emitter Templates (JET) für diesen Schritt ausgewählt. Die von JET genutzte Sprache ist sehr ähnlich zu Java Server Pages (JSP). JSP wird oft eingesetzt, um dynamische Internetseiten zu erstellen und so ist es nicht unwahrscheinlich, dass entsprechendes Knowhow in einem Unternehmen bereits vorhanden ist. Die andren beiden Technologien verwenden eigene Sprachen, die zumeist vorher vom Benutzer erlernt werden müssten.

Fazit

Es existiert eine Vielzahl von Open Source Tools, die zur Transformation von Modellen genutzt werden können. Weil die Software-Unterstützung von ALEM bereits auf Basis von Eclipse konzipiert wurde, wurden nur Tools in diesem Umfeld untersucht. Die meisten der Tools, die im Rahmen dieser Transformation untersucht wurden, bieten vielfältige Einsatzmöglichkeiten und unterscheiden sich untereinander nur in Details. Für die Erweiterung von ALEM wurden für die Restrukturierung MediniQVT, für die Instanziierung ein GMF Editor und für die Verfeinerung JET ausgewählt.

Diese Tools ermöglichen es, ALEM-basierte Modelle mit dem Ziel der Simulation zu transformieren. Ein Großteil der Prozesse kann mithilfe dieser Open Source Tools automatisiert werden. Sie können nahezu nahtlos in die bisherige Implementierung eingebettet werden, sodass nur an einer Stelle eine Interaktion mit dem Benutzer notwendig ist. Als nächste Schritte müssen sowohl das konzeptionelle Simulationsmodell, als auch die Verfeinerungsbibliothek entworfen und im Detail umgesetzt werden.

Literatur

[1] Scholz-Reiter, B.; Jagalski, T.; de Beer, C.: Selbststeuerung logistischer Prozesse in Produktionsnetzen. In: Industrie Management, 23 (2007) 1, S. 19-22.

[2] Scholz-Reiter, B.; Freitag, M.; de Beer, C.; Jagalski, T.: Modelling dynamics of autonomous logistic processes: Discrete-event versus continuous approaches. In: Annals of the CIRP, 55 (2005) 1, S. 413–417.

[3] Scholz-Reiter, B.; Hildebrandt, T.; Kolditz, J.: Modellierung selbststeuernder produktionslogistischer Prozesse - die Modellierungsmethode ALEM. In: Mattfeld, D.C.; Günther, H.-O.; Suhl, L.; Voß, St. (Hrsg): Informations- und Kommunikationssysteme in SCM, Logistik und Transport. Teilkonferenz der Multikonferenz Wirtschaftsinformatik 2008. Paderborn 2008, S. 173-185.

[4] Budinsky, F.; Steinberg, D.; Merks E.; Ellersick, R.; Grose, T.: Eclipse Modeling Frame-work: a developer's guide. Boston 2003.

[5] Object Management Group: OMG Model Driven Architecture. URL: http://www.omg.org/mda/, Abrufdatum 28.01.2010.

[6] Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT) - Specification Version 1.0. URL: http://www.omg.org/spec/QVT/1.0/PDF/, Abrufdatum 28.01.2010.

Schlüsselwörter:
Selbststeuerung, Simulation, Modelltransformation, Model Driven Architecture, Open Source

Dieser Beitrag entstand im Rahmen des Teilprojekts B2 des Sonderforschungsbereichs 637: „Selbststeuerung logistischer Prozesse – Ein Paradigmenwechsel und seine Grenzen", das von der Deutschen Forschungsgemeinschaft gefördert wird.

Selecting Open Source Software for the Model Driven Generation of Simulations

The Autonomous Logistic Engineering Methodology (ALEM), which is developed within the Collaborative Research Center 637, provides several tools for creating models of autonomously controlled logistic systems. To evaluate such models, the ALEM framework is extended to include a simulation component. As the ALEM Models cannot run directly within simulation software, they are transformed using principles of the Model Driven Architecture. To enable the transformation several open source tools can be applied. This article evaluates a selection of such tools with the aim of integrating them into ALEM.

Keywords:
autonomous control, simulation, model transformation, model driven architecture, open source