Component Tools:
Anwendung und Integration Formaler Methoden

Ekkart Kindler, Vladimir Rubin, Robert Wagner
Institut für Informatik, Universität Paderborn
Warburger Str. 100, D-33098 Paderborn
Tel.: 05251/60-3312, Fax.: 05251/60-3530
Email: [kindler|vroubine|wagner]@upb.de

Zusammenfassung

Der Einsatz Formaler Methoden beschleunigt den Entwurf des Aufbaus, die Entwicklung der Steuerungssoftware, sowie die Inbetriebnahme und Wartung von Flexiblen Fertigungssystemen. Die unterschiedlichen Aufgaben erfordern jedoch den Einsatz verschiedener Formalismen mit unterschiedlichen Konzepten, Notationen und Verfahren, die zudem häufig nur durch zueinander inkompatible Werkzeuge unterstützt werden. Die daraus resultierende mehrfache Spezifikation und Modellierung desselben Systems mit verschiedenen Formalismen ist nicht nur mühsam, zeitintensiv und damit unwirtschaftlich, sondern auch fehleranfällig. Dies führt dazu, dass Formale Methoden in der Praxis trotz ihrer Vorzüge oft nicht eingesetzt werden.

Ziel des Projektes *Component Tools* ist die Vereinfachung der Integration unterschiedlicher Formaler Methoden für einen bestimmten Anwendungsbereich unter einer einheitlichen Benutzeroberfläche. Der vorliegende Beitrag stellt die Idee, die wesentlichen Konzepte und die Hauptbestandteile von *Component Tools* anhand eines Beispiels aus dem Bereich der Flexiblen Fertigungssysteme vor.

Schlüsselwörter

Formale Methoden, Werkzeugintegration, Fertigungssysteme, Mechatronik

1 Einleitung

Auf dem Gebiet der Formalen Methoden stehen viele mächtige Techniken zur Verfügung, die dazu beitragen den Entwurf, die Analyse, die Verifikation, die Validierung, den Aufbau und die Inbetriebnahme von Systemen zu beschleunigen. Um das Potential der Formalen Methoden voll auszuschöpfen, muss der Ingenieur jedoch verschiedene Techniken miteinander kombinieren, die typischerweise ver-

schiedene Notationen und Formalismen benutzen, und oft nur von untereinander inkompatiblen Werkzeugen unterstützt werden. Eine einzelne Technik hilft meist in einer Phase der Entwicklung, muss aber in späteren Phasen von einer anderen Technik abgelöst werden. Dies führt dazu, dass für dasselbe System immer wieder neue Modelle in unterschiedlichen Formalismen und Notationen von Hand erstellt werden müssen, was zeitaufwendig und fehleranfällig ist. Aus diesem Grunde werden Formale Methoden oft gar nicht eingesetzt – trotz ihres hohen Potentials zur Qualitäts- und Effizienzsteigerung des Entwicklungsprozesses.

Natürlich gibt es eine Reihe von Werkzeugen, die verschiedene Formalismen unterstützen. Beispielsweise erlauben Simulink von The MathWorks, SimOffice™ von MSCSoftware, AutoMod™ [Bro] und eMPower [Tec] Modelle zu konstruieren und dann zu simulieren und zu analysieren. Sie besitzen eine Vielzahl von Bibliotheken für die unterschiedlichsten Anwendungsgebiete. LONTROL [FAS] ist ein Werkzeug, das speziell für die Entwicklung von Materialflusssystemen entwickelt wurde. Es benutzt eine Bibliothek von Komponenten, die sowohl die Anlage als auch die Steuerungshardware und -software betreffen. Das Forschungswerkzeug d^{3}FACT INSIGHT [DML+04] unterstützt ebenfalls die Analyse und Simulation; dabei liegt der Schwerpunkt auf der Simulation komplexer Modelle. SEA [RST98] unterstützt Methoden zur Analyse und Simulation von eingebetteten Systemen und Echtzeitsystemen und benutzt dazu verschiedene Werkzeuge.

Allerdings sind diese Werkzeuge – und viele mehr – meist auf die Simulation oder spezielle Analysen zugeschnitten und es erfordert viel Implementierungsaufwand, neue Techniken zu integrieren. Darüber hinaus ist es sehr schwer, die Ergebnisse der Analysen unter der ursprünglichen Oberfläche des Werkzeuges darzustellen und Resultate zwischen den verschiedenen Werkzeugen auszutauschen.

Das Projekt *Component Tools* und das im Projekt implementierte Werkzeug zielen darauf ab, diese Situation zu verbessern. Der wesentliche Unterschied von *Component Tools* zu den oben genannten Werkzeugen ist, dass *Component Tools* nicht auf eine bestimmte Formale Methode festgelegt ist und auf einfache Weise neue Methoden und Werkzeuge in *Component Tools* integriert werden können. Insbesondere können aufgrund bidirektionaler Transformationen Analyseresultate im ursprünglichen Modell angezeigt werden und als Eingabe für andere integrierte Werkzeuge dienen. Der Schwerpunkt von *Component Tools* liegt also auf der Möglichkeit zur einfachen Integration weiterer Formaler Methoden.

In diesem Beitrag stellen wir die Ideen und Konzepte von *Component Tools* anhand des Beispiels einer einfachen Spielzeugeisenbahn vor. Weitere Details zu den Konzepten und den zugrunde liegenden Techniken und zur technischen Realisierung von *Component Tools* sind in [KRW05] dargestellt. In Abb. 1 ist ein einfacher *Bauplan* der Spielzeugeisenbahn dargestellt. Die Anlage besteht aus vier verschiedenen *Komponententypen*: Geraden (c1, c5), Kurven (c2, c7), Weichen

(c3, c6) und Signalen (c4, c8). Die jeweiligen *Komponenteninstanzen* sind über *Verbindungen* an den entsprechenden *Anschlüssen* miteinander verschaltet. In diesem Beispiel sind dies mechanische Anschlüsse; dementsprechend sind diese Anschlüsse eins-zu-eins miteinander verbunden.

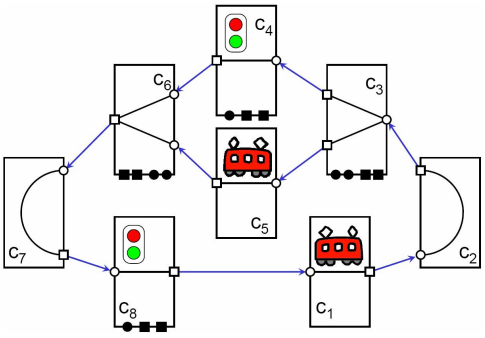

Abbildung 1: Ein Bauplan

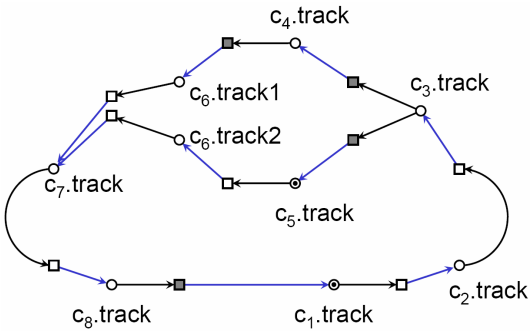

Abbildung 2: Generiertes Petrinetz

Wir nehmen hier an, dass die Züge nur an Signalen gestoppt werden können und sie ansonsten immer weiterfahren. Diese Annahme ist durch das Monoschienen-Transportsystem Montrac der MONTECH AG inspiriert. Der Kurs der Züge wird über die Weichen gesteuert. Die Signale und Weichen besitzen zu ihrer Steuerung weitere Anschlüsse: *Aktuatoren* zum Schalten der Signale und Weichen; außerdem besitzen die Signale *Sensoren*, um festzustellen, ob sich ein Zug vor dem Signal befindet; die Weichen besitzt einen Sensor, der die aktuelle Richtung der Weiche angibt. Diese elektrischen Anschlüsse sind in Abb. 1 durch schwarze Quadrate und Kreise dargestellt. Diese müssen mit einer Steuerung verbunden werden, die jedoch in Abb. 1 aus Platzgründen nicht dargestellt ist. In Abb. 2 ist ein Petrinetz dargestellt, das von *Component Tools* später automatisch generiert wird, um beispielsweise das Verhalten der Anlage zu visualisieren oder die Steuerungssoftware zu generieren. Die schattierten Transitionen des Petrinetzes stellen dabei die Übergänge der Anlage dar, die von der Steuerung kontrolliert werden können.

2 Konzepte und Hauptbestandteile

Damit *Component Tools* in einem bestimmten Anwendungsgebiet verwendet werden kann, muss zum einen eine *Komponentenbibliothek* mit verschiedenen *Sichten* definiert werden; zum anderen müssen die benötigten *Modelle, Transformationen* und *Werkzeugadapter* erstellt werden. In diesem Abschnitt stellen wir diese Konzepte genauer vor.

2.1 Komponentenbibliothek

Die für einen bestimmten Anwendungsbereich benötigten Definitionen werden in einer Komponentenbibliothek hinterlegt. Dazu zählen unterschiedliche *Komponententypen*, *Verbindungstypen* und *Anschlusstypen*.

Die *Anschlusstypen* werden benutzt, um Instanzen der Komponententypen miteinander zu verbinden. Da derselbe Anschlusstyp in verschiedenen Komponententypen vorkommen kann, werden Anschlusstypen unabhängig von den Komponenten definiert, in denen sie später verwendet werden. Eine Komponentenbibliothek kann eine beliebige Anzahl von Anschlusstypen enthalten. In Abb. 1 sind beispielsweise mechanische und elektrische Anschlüsse abgebildet. Die mechanischen Anschlüsse werden als weiße Quadrate und Kreise dargestellt; die elektrischen Anschlüsse hingegen werden durch schwarze Quadrate und Kreise dargestellt.

Die Anschlüsse können über *Verbindungen* miteinander gekoppelt werden. Auch hier können verschiedene *Verbindungstypen* definiert werden. In unserem Beispiel gibt es mechanische Verbindungen und elektrische Verbindungen. Für jeden *Verbindungstyp* wird festgelegt, welche Anschlusstypen damit verbunden werden können. Die Definition dieser Verbindungstypen und der zulässigen Anschlüsse nennen wir *Verbindungsparadigma*. Durch die Möglichkeit, die Anzahl der Verbindungen einzuschränken, können beispielsweise eins-zu-eins Verbindungen definiert werden. Mit Hilfe des Verbindungsparadigmas kann *Component Tools* sicherstellen, dass ein Bauplan niemals gegen strukturelle Einschränkungen und Regeln einer bestimmten Anwendungsdomäne verstößt.

Die *Komponententypen* sind die wichtigsten Bausteine einer Komponentenbibliothek. Jede Definition eines Komponententyps besteht aus einer Menge von Anschluss- und Parameterdefinitionen. Eine Anschlussdefinition setzt sich aus einem eindeutigen Namen, einer Beschreibung und einem Anschlusstyp zusammen; eine Parameterdefinition besteht aus einem Namen und einem Datentypen, der den Wertebereich des Parameters festlegt. In unserem Beispiel besitzt die Komponente „straight track“ einen Parameter „number“, der die Anzahl der Züge angibt, die sich initial auf diesem Gleisstück befinden. Außerdem enthält diese Komponente zwei weitere Parameter, die den geometrischen Start- und Endpunkt dieses Gleisstücks angeben. Wenn eine neue Komponenteninstanz in einem Bauplan verwendet wird, müssen die Parameter mit konkreten Werten initialisiert werden.

Jeder in der Komponentenbibliothek definierte Anschluss-, Verbindungs- und Komponententyp besitzt einen eindeutigen Namen und eine informelle Beschreibung, die die Funktion und den Zweck dieses Elementtyps näher erläutert. Diese Beschreibung wird im Werkzeug als Hilfestellung – auch Tooltip genannt – eingeblendet, sobald der Benutzer seinen Mauszeiger über einer Instanz dieses Ele-

menttyps positioniert. Zusätzlich muss für jeden Elementtyp die graphische Darstellung festgelegt werden. Für Komponenten- und Anschlusstypen wird die graphische Darstellung durch die Linienfarbe, die Füllfarbe und die Gestalt, d.h. die äußere Form, festgelegt. Im Fall von Verbindungstypen kann die graphische Darstellung durch die Pfeilspitzen und die verwendete Linienfarbe festgelegt werden. Die graphische Darstellung wird zur Anzeige der Elemente in einem graphischen Standardeditor von *Component Tools* verwendet. Die graphische Darstellung kann jedoch mit Hilfe von *Sichten* an verschiedene Benutzergruppen angepasst werden.

2.2 Sichten

Die Entwicklung eines mechatronischen Systems erfolgt interdisziplinär, so dass Ingenieure aus den Bereichen der Elektrotechnik und des Maschinenbaus mit Informatikern an einem gemeinsamen Projekt arbeiten. Jeder betrachtet dabei das zu entwickelnde System aus einem eigenen Blickwinkel und unter einem anderen Gesichtspunkt. Um den unterschiedlichen Benutzergruppen zu erlauben, sich auf einen für diese Benutzergruppe wichtigen Aspekt zu fokussieren, unterstützt *Component Tools* die Definition unterschiedlicher *Sichten*.

Jede Definition einer Sicht besteht aus einem Namen, einer Beschreibung des Zwecks und der Benutzergruppen, für die diese Sicht gedacht ist. Zusätzlich werden die Anschlusstypen, die Verbindungstypen und die Komponententypen festgelegt, die in dieser Sicht angezeigt werden. Außerdem kann in einer Sicht eine andere Darstellung definiert werden.

Wenn ein Ingenieur eine spezielle Sicht auswählt, werden ihm nur die für diese Sicht definierten Komponenten und Anschlüsse angezeigt. Außerdem werden diese Komponenten und Anschlüsse in der für diese Sicht spezifizierten Darstellung angezeigt. Die Konsistenz zwischen den unterschiedlichen Sichten gewährleistet *Component Tools* durch ein gemeinsames Modell.

Die unterschiedlichen Sichten können mit dem Standardeditor von *Component Tools* bearbeitet und angezeigt werden. Zusätzlich kann für bestimmte Sichten jeweils ein an diese Sicht angepasster Editor zur Verfügung gestellt werden. Zum Beispiel könnte es einen Editor geben, in dem die mechanischen Eigenschaften von Komponenten genauer angezeigt werden. Abbildung 3 zeigt beispielhaft, wie eine alternative Darstellung des Bauplans aus dem vorangegangenen Beispiel aussehen könnte. In dieser Sicht steht die Geometrie der Anlage im Vordergrund. Daher werden in dieser Sicht lediglich die mechanischen Anschlüsse angezeigt. Zusätzlich ist es in diesem Editor möglich, die zueinander kompatiblen Anschlüsse durch Einrasten direkt miteinander zu verbinden.

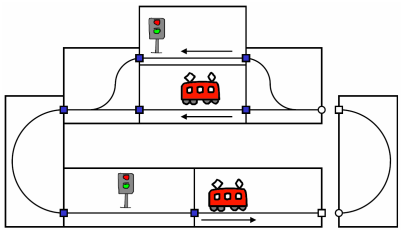

Abbildung 3: Geometrische Sicht auf die Anlage

2.3 Modelle und Modelltransformationen

Die in einer Komponentenbibliothek definierten Komponenten können von einem Ingenieur zu einem Bauplan der Anlage verschaltet werden. Außerdem kann jeder Ingenieur den Bauplan in seiner bevorzugten Sicht bearbeiten. Die zu jeder Komponente hinterlegten Beschreibungen können benutzt werden, um das Verhalten einer jeden Komponente und damit der gesamten Anlage besser zu verstehen. Die wesentliche Funktion von *Component Tools* ist jedoch die Anwendung Formaler Methoden bei der Entwicklung, Analyse, Validierung und Inbetriebnahme der Anlage.

Hierzu müssen zu jedem Komponententypen Formale Modelle hinterlegt werden, die das Verhalten der entsprechenden Komponente festlegen. Die Definition eines Modells besteht aus einem Namen mit einer Referenz zu einem Modell in einem beliebigen Format. Zusätzlich kann eine Beschreibung angegeben werden, die den Zweck dieses Modells erklärt. Tatsächlich können einem Komponententyp sogar mehrere unterschiedliche Modelle zugeordnet werden. Dabei ist *Component Tools* nicht auf bestimmte Formalismen und Notationen eingeschränkt. In unserem Beispiel haben wir zwei verschiedene Varianten von Petrinetzen für eine sehr einfache Modellierung des Verhaltens der Komponenten benutzt und eine einfache Notation zur Beschreibung der geometrischen Anordnung der Komponenten der Anlage. Abbildung 4 zeigt zwei dieser Petrinetze, die das Verhalten der Komponenten Signal und Weiche modellieren.

Zur Anwendung von Formalen Methoden müssen die einzelnen Modelle in ein Gesamtmodell in einer für externe Werkzeuge verständliche Notation überführt werden. Dies geschieht auf Basis des vom Ingenieur erstellten Bauplans und der hinterlegten Modelle der Komponententypen. Die Übersetzung ist mit *Tripel-Graph-Grammatiken* (TGGs) [Sch95] realisiert. Der Vorteil dieser Transformationstechnik besteht darin, dass die Übersetzung in beide Richtungen möglich ist und dadurch Änderungen im Gesamtmodell und Analyseresultate zurückübersetzt und somit im Bauplan angezeigt werden können. Die verwendete Transformationstechnik und die konkreten Regeln werden in [KRW05] genauer erläutert.

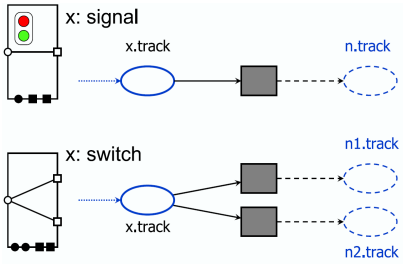

Abbildung 4 : Modelle der Komponenten Signal und Weiche

Das durch die automatische Übersetzung gewonnenen Gesamtmodell dient als Eingabe für ein Werkzeug zur Anwendung einer Formalen Methode. Vor ihrem Einsatz muss die Anwendung in *Component Tools* integriert werden.

2.4 Anwendungen und Werkzeugadapter

Die Definition einer *Anwendung* besteht aus zwei Teilen: (1) einer Tripel-Graph-Grammatik zur Transformation eines Bauplans in ein Gesamtmodell einer bestimmten Notation, und (2) einem Werkzeugadapter, der nach der Transformation gestartet wird und das automatisch generierte Gesamtmodell als Eingabe erhält. Da (1) bereits in Abschnitt 2.3 erläutert wurde, gehen wir hier nur noch auf (2) näher ein.

Ein Werkzeugadapter darf das ihm übergebene Modell verändern, analysieren, speichern, oder an ein externes Werkzeug weitergeben. Hierzu muss jeder Werkzeugadapter eine vorgegebene Schnittstelle implementieren. Diese Schnittstelle besteht aus einer Methode zum Starten des Werkzeugadapters und einer Methode, die als Parameter das Modell entgegennimmt.

Wir unterscheiden zwei Arten von Werkzeugadaptern: Zum einen sind dies Werkzeugadapter, die mit dem Ursprungsmodell auch nach dem Start verbunden bleiben, um beispielsweise Änderungen im transformierten Modell an das Ursprungsmodell zurückpropagieren. Zum anderen handelt es sich um Werkzeugadapter, die keine Verbindung zum Ursprungsmodell aufrechterhalten.

Tatsächlich werden von uns Werkzeugadapter bevorzugt, die auch nach dem Starten weiterhin mit dem Ursprungsmodell verbunden bleiben. Dies liegt daran, dass durch die Aufrecherhaltung dieser Verbindung die im transformierten Modell erzielten Analyseergebnisse und durchgeführten Modelländerungen dem Ingenieur wieder im Bauplan angezeigt werden können. Damit muss der Ingenieur sich nicht mit den verwendeten Formalismen auseinandersetzten – er kann sich die Ergebnisse in seiner bevorzugten Sicht anschauen.

2.5 Szenarien

Die bidirektionalen Transformationen erlauben es insbesondere, Analyseergebnisse einer Anwendung in Form von *Szenarien* in *Component Tools* zurück zu transformieren und dort anzuzeigen. Abbildung 5 zeigt ein Beispiel eines Szenarios, das durch eine Anwendung von der Anlage aufgezeichnet worden sein könnte.

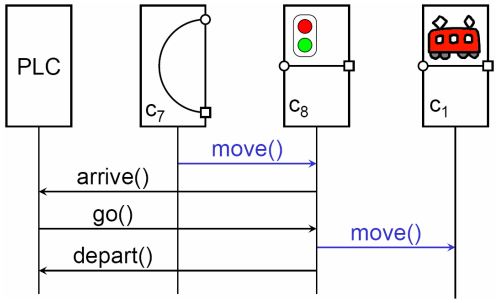

Abbildung 5 : Szenarien in Component Tools

In diesem Szenario fährt zunächst ein Zug vom Gleissegment c_7 auf das Signalsegment c_8; über den Sensor wird die Ankunft eines Zuges am Signal an die Steuerung gemeldet. Daraufhin gibt die Steuerung irgendwann über einen Aktuator die Fahrt für den Zug frei, der dann auf das nächste Gleissegment c_1 weiterfährt. Über den entsprechenden Sensor wird das Verlassen des Gleissegments c_8 an die Steuerung gemeldet. Die Notation für die Szenarien ist vollkommen unabhängig von den zugrunde liegenden Formalen Methoden. Deshalb können Szenarien benutzt werden, um dem Ingenieur Analyseergebnisse in einer einheitlichen Notation darzustellen, und Ergebnisse zwischen den verschiedenen Anwendungen auszutauschen. Beispielsweise können Szenarien zur Spezifikation der Steuerungssoftware benutzt werden, um daraus dann Steuerungssoftware automatisch zu generieren. Oder eine von Hand erstellte Steuerungssoftware kann verifiziert werden; ein mögliches Fehlverhalten könnte dann als Szenario ausgegeben werden; dieses Szenario kann dann dem Ingenieur mit Hilfe einer 3-dimensionalen Visualisierung gezeigt werden (siehe Abb. 6).

3 Zusammenfassung und Ausblick

In diesem Papier haben wir die wesentlichen Ideen von *Component Tools* anhand eines einfachen Beispiels vorgestellt. Ein Prototyp des Werkzeuges wurde im Rahmen einer einjährigen Projektarbeit von einer Gruppe von zehn Studenten als Erweiterungen der frei verfügbaren Softwareentwicklungsumgebung *Eclipse* (http://www.eclipse.org) implementiert. Abbildung 6 zeigt einen Screenshot von *Component Tools* [Gre03, GGK+04] mit einer Anwendung zur 3-dimensionalen Visualisierung des Verhaltens mit Hilfe von *PNVis* [KP04].

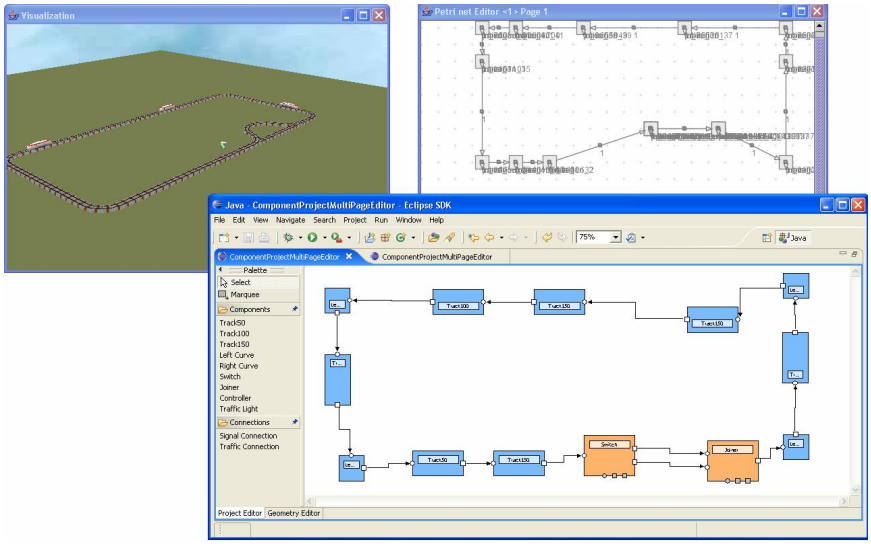

Abbildung 6 : Screenshot von Component Tools mit der Anwendung PNVis

Ziel von *Component Tools* ist die Integration vieler verschiedener Formaler Methoden für den ingenieurmäßigen Einsatz unter einer einheitlichen Benutzeroberfläche. Der Ingenieur muss sich dabei nicht mit den verschiedenen Notationen der verschiedenen Formalen Methoden auseinander setzen; insbesondere kann er Anlagen unabhängig von der benutzten Methode modellieren und immer denselben Mechanismus zur Komposition von Komponenten benutzen. Die Einbindung Formaler Methoden (bzw. der entsprechenden Werkzeuge) übernimmt *Component Tools* mit Hilfe von entsprechenden Transformationen.

Als Beispiel haben wir eine Spielzeugeisenbahn benutzt, die von Anwendungen im Bereich der Flexiblen Fertigungssysteme inspiriert ist. Für dieses Beispiel wurden mehrere Transformationen in verschiedene Varianten von Petrinetzen und eine Beschreibung der Geometrie der Anlage definiert und Adapter zum Petrinetzkern und PNVis implementiert. *Component Tools* hat jedoch ein viel breiteres Einsatzspektrum. Da die Transformationen und die Anwendungen ohne zu programmieren in *Component Tools* eingebunden werden können, ist der Einsatzbereich nur von der Bereitstellung der entsprechenden Formalen Methoden und der zugehörigen Transformationen zu den entsprechenden Werkzeugen abhängig.

Zurzeit werden im Rahmen verschiedener Diplomarbeiten Techniken zur automatischen Synthese und Verifikation der Steuerungssoftware, zur Fehlerdiagnose bei der Inbetriebnahme, sowie zur Visualisierung des Verhaltens und möglichen Fehlverhaltens von Anlagen implementiert. All diese Techniken werden in *Component Tools* integriert und arbeiten somit auf demselben Bauplan der zu entwickelnden Anlage. In diesem Beitrag können wir auf diese und weitere Anwendungen nicht näher eingehen; [GKK+05] gibt einen Überblick über diese

Techniken und eine Vision für den zukünftigen Einsatz einer Kombination verschiedener Formaler Methoden beim Systementwurf.

Danksagung Wir bedanken uns bei den Teilnehmern der Projektgruppe "Component Tools" im Diplomstudiengang Informatik an der Universität Paderborn: A. Gepting, J. Greenyer, A. Maas, S. Munkelt, C. Páles, T. Pivl, O. Rohe, M. Sanders, A. Scholand, und C. Wagner. Sie haben durch viele Diskussionen signifikant zu den Konzepten von *Component Tools* in ihrer heutigen Form beigetragen; darüber hinaus haben sie mit großem Engagement *Component Tools* implementiert.

Literatur

[Bro] Brooks Automation Corporate. AutoMod Suite. http://www.automod.com/products/products.asp.

[DML+04] Dangelmaier, W.; Mueck, B.; Laroque, C.; Mahajan, K.: d3fact insight: A simulation-tool for multiresolution material flow models. In István Lipovszki, György; Molnár (Hrsg.) Simulation in Industry - 16th European Simulation Symposium (ESS2004) SCS - Europe, Seiten 17 – 22, 2004.

[FAS] FASTEC GmbH. Lontrol. http://www.fastec.de/.

[GGK+04] Gepting, A.; Greenyer, J.; Kindler, E.; Maas, A.; Munkelt, S.; Páles, C.; Pivl, T.; Rohe, O.; Rubin, V.; Sanders, M.; Scholand, A.; Wagner, C.; Wagner, R.: Component tools: A vision of a tool. In Proc. of the 11th Workshop on Algorithms and Tools for Petri Nets (AWPN), Tech. Rep. tr-ri-04-251, Seiten 37–42, Paderborn, Deutschland, September 2004.

[GKK+05] Giese, H.; Kindler, E.; Klein, F.; Wagner, R: Reconciling scenario-centered controller design with statebased system models. In 4th International Workshop on Scenarios and State Machines: Models, Algorithms and Tools (SCESM'05), Satellite event of ICSE '05, Mai 2005.

[Gre03] Greenyer, J.: Maintaining and using component libraries for the design of material flow systems: Concept and prototypical implementation. Studienarbeit, Institut für Informatik, Universität Paderborn, Oktober 2003.

[KP04] Kindler, E.; Páles, C.: 3D-visualization of Petri net models: Concept and realization. In J. Cortadella and W. Reisig (Hrsg.) Application and Theory of Petri Nets 2004, 25th International Conference, LNCS 3099, Seiten 464–473. Springer, Juni 2004.

[KRW05] Kindler, E.; Rubin, V.; Wagner, R.: Component Tools: Application and Integration of Formal Methods. Im elektronischen Tagungsband des Workshops Objektorientiertes Software Engineering 2005 (OOSE '05), im Rahmen der Net.ObjectDays 2005, Erfurt, Deutschland, September 2005.

[RST98] Rust, C.; Stroop, J.; Tacken, J.: The Design of Embedded Real-Time Systems using the SEA Environment. In Proc. Of the 5th Annual Australasian Conference on Parallel And Real-Time Systems (PART '98), Adelaide, Australia, September 1998.

[Sch95] Schürr, A.: Specification of graph translators with triple graph grammars. In E. W. Mayr, G. Schmidt, and G. Tinhofer (Hrsg.) Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG '94, LNCS 903, Seiten 151–163, Herrsching, Deutschland, Juni 1995.

[Tec] Tecnomatix Technologies Ltd. eMPower Solutions. http://www.tecnomatix.com/.