Service-orientierte Modellierung einer Robotermontagezelle

Alwin Hoffmann, Andreas Angerer, Andreas Schierl, Michael Vistein, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Hoffmann, Alwin, Andreas Angerer, Andreas Schierl, Michael Vistein, and Wolfgang Reif. 2011. "Service-orientierte Modellierung einer Robotermontagezelle." In Tagungsband: Internationales Forum Mechatronik 2011, Cham,21.-22. September 2011, 179–92. Augsburg: Cluster Mechatronik & Automation.

Service-orientierte Modellierung einer Robotermontagezelle

Dipl.-Inf. Alwin Hoffmann

Institut für Software & Systems Engineering

Universität Augsburg

Information zum Referenten:

Dipl.-Inf.
Alwin Hoffmann

Studium

- Studium des Diplomstudiengangs „Informatik“ mit Schwerpunkt Software Engineering an der Universität Augsburg
- Studium des Master-Studiengangs „Finance & Information Management“ an der Technischen Universität München und der Universität Augsburg

Berufliche Karriere

- Wissenschaftlicher Mitarbeiter am Institut für Software & Systems Engineering der Universität Augsburg
- Leitung des Forschungsbereichs „Softwaregetriebene Mechatronik & Robotik “ des Institut für Software & Systems Engineering
- Koordination des Forschungsprojekts *SoftRobot*

Derzeitiger Arbeitgeber / Institut

Institut für Software & Systems Engineering
Universität Augsburg
http://www.isse.de
hoffmann@informatik.uni-augsburg.de

Service-orientierte Modellierung einer Robotermontagezelle

A. Hoffmann, A. Angerer, A. Schierl, M. Vistein, W. Reif

Institut für Software & Systems Engineering

Universität Augsburg

Zusammenfassung

Im Rahmen des Forschungsprojekts *SoftRobot* wurde der Einsatz service-orientierter Architekturen zur Programmierung und Steuerung roboterbasierter Fertigungszellen untersucht. Die Grundlage dafür stellt das objektorientierte *Robotics Application Programming Interface (Robotics API)* dar, welches in der Programmiersprache Java entwickelt wurde. Die Robotics API beinhaltet ein Domänenmodell zur Definition von physikalischen Objekten, Bewegungen oder Schaltaktionen innerhalb einer Roboteranwendung. Dadurch können Bewegungen und weitere Werkzeugaktionen spezifiziert, gegebenenfalls miteinander synchronisiert und anschließend zur echtzeitfähigen Robotersteuerung übertragen. Durch die Verwendung der modernen Standardprogrammiersprache Java können Roboterprogramme fließend in service-orientierte Architekturen integriert werden. Die Vorteile einer service-orientierter Roboterzelle und des in *SoftRobot* entwickelten Serviceansatzes werden in diesem Papier vorgestellt und an einer Montagezelle, die u.a. aus zwei KUKA Leichtbaurobotern besteht, illustriert.

1 Einleitung

Durch ihre mechanische Struktur und die daraus resultierende freie Programmierbarkeit sind Industrieroboter ein wesentlicher Bestandteil flexibler Fertigungssysteme. Ausgestattet mit den passenden Werkzeugen können Roboter eine Vielzahl unterschiedlichster Arbeiten ausführen. Die potentiellen Einsatzgebiete reichen von der industriellen Fertigung über die Qualitätskontrolle bis hin zu medizinischen Anwendungen. Eine entscheidende Hürde stellt bis heute jedoch die einfache und schnelle Integration der Roboter in ihren Kontext (d.h. die weiteren Anlagen und Systeme, mit denen interagiert werden muss) sowie die flexible Anpassbarkeit der Fertigungsprozesse an neue Gegebenheiten dar. Aus der Sicht der Informatik kann eine Roboteranlage als ein inhärent verteiltes System betrachtet werden.

Für Unternehmensanwendungen, die vor ähnlichen Herausforderungen und Problemen stehen, haben sich Service-orientierte Architekturen (Erl, 2005) als Softwarearchitektur der Wahl etabliert. Bei der Modellierung service-orientierter Architekturen nehmen die zugrundeliegenden Geschäftsprozesse des Unternehmens eine zentrale Rolle ein, da Services bzw. Dienste fachliche Funktionalität kapseln und damit einen Teil des Geschäftsprozesses repräsentieren. Per Definition sind Dienste in sich abgeschlossen und besitzen eine wohldefinierte Schnittstelle, über die sie eigenständig benutzt werden können. Ein wesentliches Ziel ist eine hohe Wiederverwendbarkeit von Diensten, um Geschäftsprozesse flexibel und kostengünstig anpassen bzw. neu aufsetzen zu können. Einzelne Dienste wiederum können kombiniert werden und so, entsprechend orchestriert, komplexere fachliche Funktionalität repräsentieren. Darüber hinaus ist es für die Nutzung eines Dienstes ausreichend, nur die Schnittstelle, den Dienstvertrag, zu kennen. Die konkrete Implementierung dagegen muss nicht bekannt sein und ist austauschbar. Service-orientierte Architekturen sind verteilte Systeme, die Mechanismen zur transparenten Kommunikation der einzelnen Dienste anbieten. Dadurch sind Dienste in der Regel plattformunabhängig; die Anbieter und Nutzer von Diensten können in unterschiedlichen Programmiersprachen und auf verschiedenen Plattformen realisiert sein.

Im Bereich der industriellen Automatisierung haben Jammes und Smit (2005) die Vorteile service-orientierter Architekturen für die Integration unterschiedlicher Anlagen und Systeme bereits hervorgehoben. Betrachtet man flexible Fertigungssysteme, insbesondere unter Einbeziehung mehrerer Roboter, so bieten service-orientierte Architekturen noch weitere Vorteile. Vor allem eine Orientierung an den Fertigungsprozessen bei der Modellierung der Dienste kann, analog zu Unternehmensanwendungen, eine Kapselung fachlicher Funktionalität mit hohen Wiederverwendungsgrad zur Folge haben. Die Kapselung von Funktionalität und Bereitstellung von selbiger durch Dienstverträge sind die Voraussetzung für Plug-and-Produce-Zellensteuerungen (Naumann et al., 2007; Veiga und Pires, 2007). Des Weiteren erlauben service-orientierte Architekturen (SOA) den Austausch von Dienstimplementierungen zur Laufzeit durch neuere und evtl. effizientere Implementierungen, aber

auch durch Implementierungen, die andere Roboter- bzw. Anlagenkomponenten steuern.

Im Rahmen des Forschungsprojekts *SoftRobot* wurde der Einsatz service-orientierter Architekturen zur Programmierung und Steuerung roboterbasierter Fertigungszellen untersucht. Die Grundlage dafür stellt das objektorientierte *Robotics Application Programming Interface (Robotics API)* dar, das alle notwendigen Mittel zur Programmierung von Industrierobotern zur Verfügung stellt. Mit Hilfe der Robotics API werden Bewegungen und weitere Werkzeugaktionen spezifiziert, gegebenenfalls miteinander synchronisiert und anschließend zur echtzeitfähigen Robotersteuerung übertragen. Die Robotics API wurde in das Komponenten- bzw. Servicemodell der OSGi-Softwareplattform (OSGi Alliance, 2009a) integriert. OSGi kann als lokale SOA betrachtet werden, wobei zusätzlich die Verteilung und Plattformunabhängigkeit mit Hilfe von Webservices (OSGi Alliance, 2009b) realisiert werden kann. Die Vorteile service-orientierter Architekturen und des in *SoftRobot* entwickelten Serviceansatzes werden in diesem Papier vorgestellt und an einer Montagezelle, die u.a. aus zwei KUKA Leichtbaurobotern besteht, illustriert.

Das Forschungsprojekt *SoftRobot* wird gemeinsam vom Institut für Software & Systems Engineering der Universität Augsburg, der KUKA Roboter GmbH und der MRK-Systeme GmbH durchgeführt und wird von der Europäischen Union und der High-Tech-Offensive Bayern der Bayerischen Staatsregierung gefördert. In ihrer Funktion als Projektträger unterstützt die VDI-VDE-IT GmbH das Projekt.

2 Die SoftRobot-Architektur

Im Forschungsprojekt *SoftRobot* wurde eine neue Softwarearchitektur für die Programmierung von Industrieroboten entwickelt (Hoffmann et al., 2009; Angerer et al., 2010b). Zentraler Bestandteil der in Abbildung 1 dargestellten Architektur ist das *Robotics Application Programming Interface* (Robotics API), eine objektorientierte Programmierschnittstelle, die alle notwendigen Mittel zur Programmierung von Industrierobotern zur Verfügung stellt. Roboterbefehle, die mit den Mitteln der Robotics API definiert wurden, werden automatisch zur echtzeitfähigen Robotersteuerung übertragen, wo die Ansteuerung und Regelung der Hardware stattfindet. Die Robotersteuerung verfügt ebenfalls über eine spezielle, erweiterbare Schnittstelle, das *Realtime Primitive Interface* (RPI). Damit ist es möglich, echtzeitkritische Aufgaben für einen oder mehrere Roboter (z.B. Bewegungen, Interaktion mit Werkzeugen und Sensoren) zu spezifizieren und auf der Robotersteuerung auszuführen.

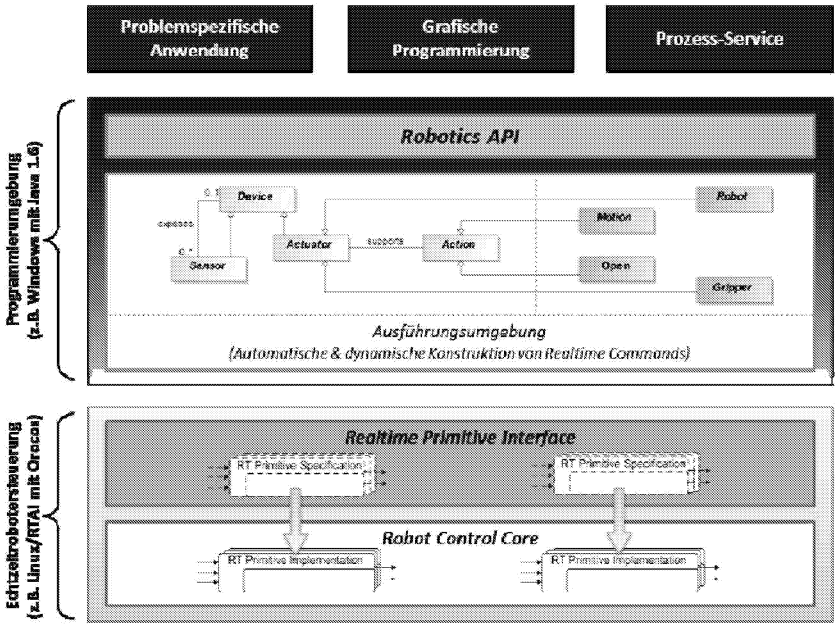

Abbildung 1: Die im Forschungsprojekt „SoftRobot" entwickelte Architektur zur Trennung von echtzeitkritischer Robotersteuerung und objektorientierter Programmierung

1.1 Realtime Primitive Interface

Das *Realtime Primitive Interface* (Vistein et al., 2010) erlaubt die Spezifikation von echtzeitkritischen Aufgaben mittels einer Datenflusssprache und definiert eine Schnittstelle, um solche Aufgaben an eine kompatible Robotersteuerung zu übertragen. Dort werden diese Aufgaben als Transaktionen unter Echtzeitbedingungen ausgeführt. Dazu existieren *Realtime Primitives* und *Links*. Während ein Realtime Primitive ein atomarer Funktionsblock mit Ein- und Ausgängen ist, der eine in der echtzeitfähigen Robotersteuerung implementierte Funktionalität repräsentiert, stellt ein Link eine Verbindung zwischen Ein- und Ausgängen von Instanzen von Realtime Primitives dar. Primitive und deren Links bilden einen Graph, der – sofern er keine offenen Ein- und Ausgänge beinhaltet – als sogenanntes *Realtime Command* ausgeführt werden kann. Dem Transaktionskonzept folgend ist ein *Realtime Command* in sich abgeschlossen, d.h. das System befindet sich vor und nach dessen Ausführung in einem stabilen und konsistenten Zustand. Die Ausführung erfolgt zyklisch in einer kompatiblen Robotersteuerung, d.h. in jedem Takt wird der ganze Graph einmal ausgewertet. Für die Erweiterung der unterstützten Hardware und Steuerungskonzepte erlaubt RPI die Definition und Integration neuer Primitive.

Eine kompatible Robotersteuerung, ein sogenannter *Robot Control Core*, stellt einen Interpreter für Realtime Commands und Implementierungen der benötigten Primitive bereit. Die Robotersteuerung ist für die Steuerung der Robotermechanik und der weiteren Geräte zuständig, d.h. sie muss mit dem Roboter kommunizieren, um diesem neue Sollwerte für seine Achsen zu übermitteln. Sie stellt die echtzeitkritische Funk-

tionalität des gesamten Systems zur Verfügung und muss daher auf einem Echtzeitbetriebssystem implementiert sein.

1.2 Robotics API

Das objektorientierte *Robotics Application Programming Interface* (Angerer et al. 2010a) stellt die Programmierschnittstelle für die Entwicklung von Robotikanwendungen dar. Für die Programmierung von Industrierobotern und deren Werkzeugen bietet die Robotics API eine umfangreiche Klassenhierarchie, die grundlegende Funktionalität bereitstellt und gleichzeitig für eigene Applikationen und Einsatzbereiche um neue Robotertypen und Werkzeuge bzw. Bewegungen und Aktionen erweiterbar ist.

Im Rahmen der objektorientierten Gestaltung der Robotics API wurde die Domäne der Industrierobotik so detailliert wie möglich durch (Software-)Objekte und Assoziationen abgebildet. Dieses Objektmodell ist in Abbildung 2 dargestellt und besteht aus drei verschiedenen Bereichen, die die Betrachtungsweise widerspiegeln, dass ein Roboter (inkl. Werkzeuge und Sensoren) mit seiner (Arbeits-)Umgebung interagiert:

- Die allgemein in der Robotersteuerung verfügbaren Geräte werden durch die Klasse *Device* repräsentiert. Eine Spezialisierung stellt die Klasse *Actuator* dar, die eine steuerbare Einheit repräsentiert und von der weiter abgeleitet werden kann, um konkrete Ausprägungen zu definieren.
- Außerdem ist es möglich, die physikalischen Eigenschaften von Geräten und weiteren, real vorhandenen Bestandteilen (*PhysicalObjects*) der Arbeitsumgebung (z.B. Werkstücke) explizit in Software darzustellen und somit die für die Aufgabe wichtigen, applikationsspezifischen Informationen zu modellieren.

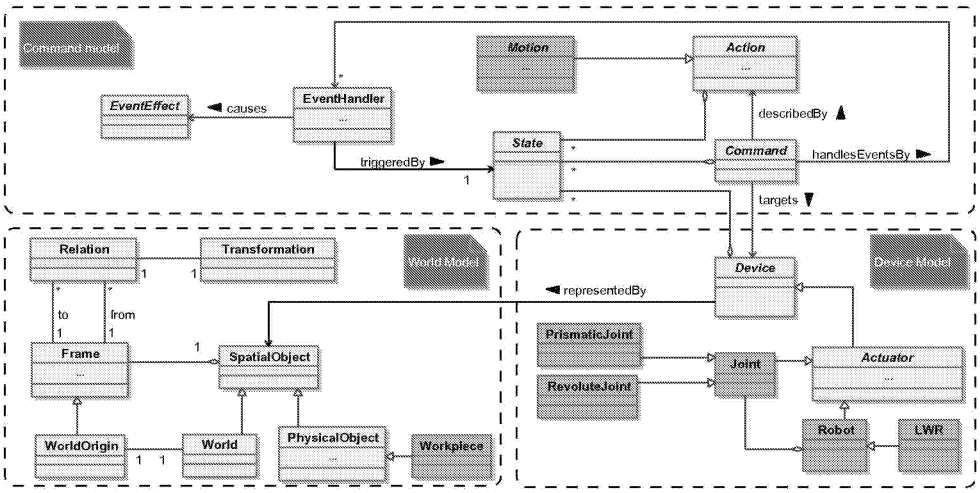

Abbildung 2: Die Klassenhierarchie der Robotics API

- Aktionen (*Actions*) sind abstrakte Beschreibungen einer Fähigkeit und werden über ein Kommando (Command) an ein konkretes steuerbares Gerät gebunden. Durch die Modellierung der Fähigkeiten als einzelne Objekte ist es möglich, aus einzelnen Fähigkeiten größere und komplexere Fähigkeiten zu komponieren, die dem Transaktionskonzept folgend atomar als *Realtime Command* in der Robotersteuerung ausgeführt werden. Da jede Aktion bzw. jedes Gerät ihm spezifische Zustände (*States*) ausweist, deren Betreten bzw. Verlassen jeweils ein Ereignis darstellt, können durch *EventHandler* diese Ereignisse mit einer Reaktion (*EventEffect*) verknüpft werden. Beispiele dafür sind das Starten eines neuen bzw. das Abbrechen eines laufenden Kommandos.

Die Robotics API wurde im Projekt *SoftRobot* prototypisch in Java implementiert und in das OSGi-Framework (OSGi Alliance, 2009a) integriert. Da die Robotics API als modulares und erweiterbares Framework konzipiert ist, besteht sie aus einer Menge einzelner Komponenten[1]. Eine zentrale Komponente definiert das abstrakte, oben beschriebene Domänenmodell und enthält die wichtigsten Klassen und Schnittstellen, wie bspw. *Device*, *Action* oder *State*, definiert. Diese sind in Abbildung 2 hell dargestellt und formen den Kern der Robotics API.

Darüber hinaus gibt es Komponenten, welche die Robotics API um neue Gerätedefinitionen und neue Aktionen erweitern. Durch die in der Objektorientierung übliche Vererbung können neue Klassen und Schnittstellen für weitere steuerbare Geräte (inkl. physikalisches Modell und Aktionen) von geeigneten Oberklassen abgeleitet und in einer eigenständigen Komponente verfügbar gemacht werden. Die Definition von Werkzeugen (z.B. ein elektrischer Greifer) ist ebenso Teil einer solchen Erweiterung wie die Definition der Klasse *Robot* und der dazugehörigen Bewegungen. Der KUKA Leichtbauroboter wiederum ist eine Spezialisierung eines Roboters und ebenfalls in einer eigenständigen Komponente definiert. Diese Erweiterungen der Applikationsschnittstelle sind in Abbildung 2 dunkel dargestellt. Während die oben genannten Komponenten die Applikationsschnittstelle darstellen, beinhalten andere Komponenten die spezifischen Implementierungen für die konkret verwendete RPI-kompatible Robotersteuerung, insbesondere die Transformationsregeln der Aktionen auf *Realtime Commands*. Daneben existieren weitere Komponenten mit Infrastrukturdiensten, z.B. für die Konfiguration und die Bereitstellung von Objekten der verfügbaren Roboter, Werkzeuge und Sensoren.

Durch diese Struktur ist die Robotics API ein erweiterbares Framework für die High-Level-Programmierung von Anwendungen für Industrieroboter. Durch die feingranulare Aufteilung der Komponenten kann der Anwendungsentwickler die für ihn notwendigen Komponenten auswählen und darauf aufbauend seine Applikation entwickeln. Zudem unterstützen Infrastrukturdienste und Hilfsklassen bei der Programmierung.

[1] in OSGi Bundles genannt.

2 Modellierung einer Montagezelle

Um Erfahrungen mit der service-orientierten Modellierung einer Roboterzelle zu gewinnen, wurde ein beispielhafter Montageprozess implementiert und experimentell validiert. Bei der Auswahl des Szenarios waren folgende Aspekte wichtig:

- Die Zelle sollte mindestens zwei Roboter beinhalten, die miteinander interagieren und später kooperieren können.
- Es sollten unterschiedliche, evtl. parallel stattfindende Arbeitsabläufe in der Zelle existieren. Zudem sollten unterschiedliche Variationen in den zu bearbeitenden Werkstücken bzw. Aufgaben möglich sein.
- Die Zelle sollte aus mehreren, heterogene Subsysteme zusammengestellt sein, die miteinander kombiniert werden müssen.

Auf dieser Basis wurde eine Robotermontagezelle konzipiert und aufgebaut, die in Abschnitt 3.1 detailliert vorgestellt wird. Die service-orientierte Modellierung dieser Zelle wird anschließend in Abschnitt 3.2 kurz skizziert.

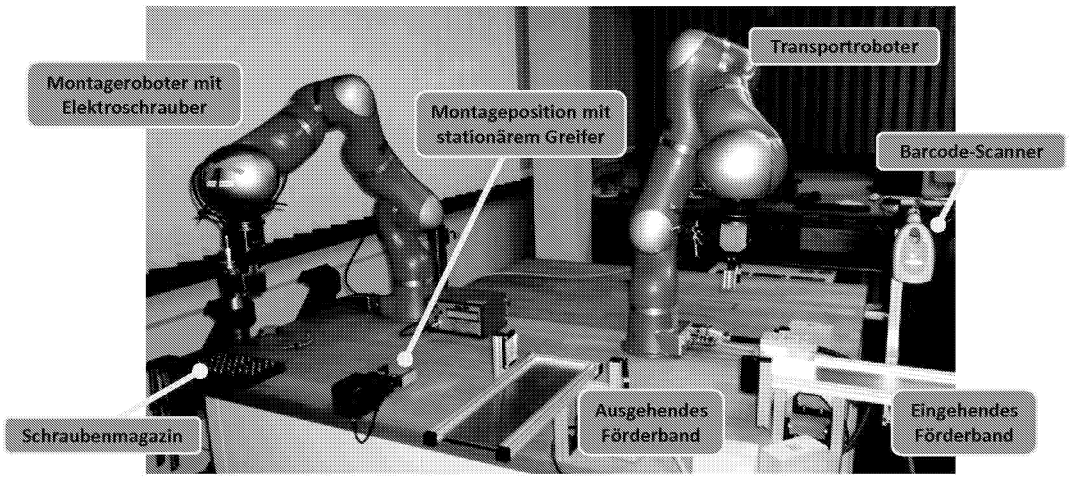

Abbildung 3: Aufbau der Montagezelle im Versuchslabor

2.1 Vorstellung der Montagezelle

In der Montagezelle (vgl. Abbildung 3) kommen zwei KUKA Leichtbauroboter (LBR) zum Einsatz, die auf einer gemeinsamen Werkbank montiert sind. Der rechte Roboter, der Transportroboter, ist mit einem Greifer ausgestattet, um in der Zelle ankommende Werkstücke zu transportieren. Am Flansch des zweiten, linken Roboters, dem Montageroboter, ist ein Elektroschrauber mit Drehmomentabschaltung montiert, mit dem Schrauben aus einem Magazin kraftgesteuert in die angelieferten Werkstücke geschraubt werden können. Für den Transport der Werkstücke in die Zelle hinein bzw. aus der Zelle heraus sind zwei Förderbänder vorgesehen, die im Versuchsaufbau durch zwei Profilkonstruktionen simuliert werden. Durch eine Lichtschrank kann

am eingehenden Förderband das Eintreffen neuer Werkstücke registriert werden. Am ausgehenden Förderband wird dadurch der Abtransport eines abgelegten Werkstückes angezeigt. Um Werkstücke beim Verschrauben zu fixieren, ist an der Montageposition zudem ein stationärer Greifer befestigt. An der linken vorderen Seite der Abbildung ist das Schraubenmagazin zu sehen.

In der Montagezelle können zwei unterschiedliche Werkstücktypen auftreten:

- Einfache Werkstücke sind bereits fertig montiert und müssen nicht weiter bearbeitet werden. Sie werden vom Transportroboter direkt vom eingehenden Förderband zum ausgehenden Förderband transportiert. Der Transport dieser Werkstücke kann parallel zur Bearbeitung eines Werkstückes am Montageroboter durchgeführt werden.
- Noch zu montierende Werkstücke bestehen aus zwei lose miteinander verbundenen Teilen, die vor dem Abtransport mit mehreren Schrauben fixiert werden müssen. Die Anzahl, Art und Größe der Schrauben und der Schraubenlöcher kann dabei von Werkstück zu Werkstück variieren. Für die Bearbeitung werden diese Werkstücke zuerst durch den Transportroboter vom eingehenden Förderband zur Montageposition transportiert und dort in den stationären Greifer eingespannt. Anschließend verschraubt der Montageroboter die Werkstücke entsprechend der Werkstückbeschreibung. Sobald das Werkstück komplett montiert ist, wird es von Transportroboter zum ausgehenden Förderband transportiert.

Durch die beiden, oben beschriebenen Werkstücktypen werden die beiden unterschiedlichen und parallel stattfindenden Arbeitsabläufe der Zelle definiert. Zur Identifikation der Werkstücktypen und -varianten sind die Werkstücke mit Barcodes versehen, welche beim eingehenden Förderband ausgelesen werden. Die Werte der Barcodes werden vom Barcodescanner über einen Webservice zur Verfügung gestellt.

2.2 Modellierung der Dienste

Analog zu Unternehmensanwendungen und Geschäftsprozessen waren die beiden Automatisierungsprozesse der Montagezelle Grundlage für die Modellierung der notwendigen Services und ihrer Kontrakte, d.h. ihrer Schnittstelle. Bei der Identifikation der Dienste hat es sich als sinnvoll und nützlich herausgestellt, die realen Komponenten der Zelle, d.h. die Roboter, die beiden Förderbänder und die Montagestation mit dem stationären Greifer als eigene Services zu konzipieren und sie mit den entsprechenden Fähigkeiten in Form eines geeigneten Kontrakts auszustatten.

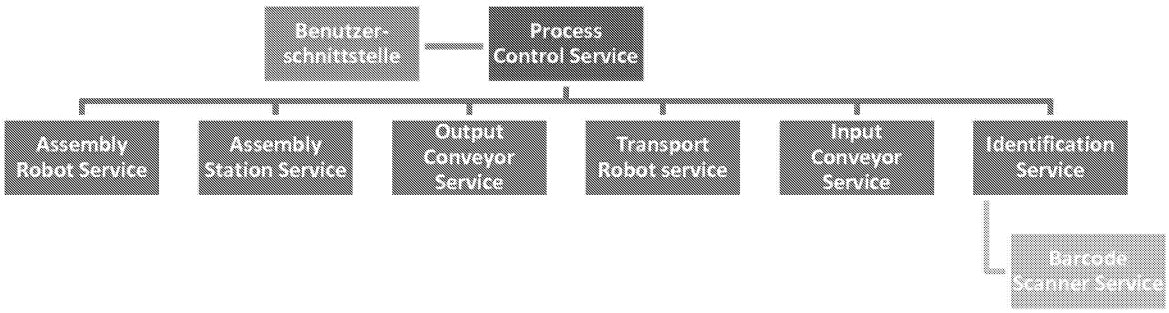

Abbildung 4: Aufbau der Services in der Montagezelle

Die Montagezelle wurde mit Hilfe von Webservices umgesetzt. Da sich zustandslose Webservices nicht eignen, zustandsbehaftete Ressourcen wie Roboter oder andere Peripheriegeräte adäquat zu repräsentieren, wurde mit *Apache Muse*[2] eine Implementierung der Webservice Spezifikationen *WS-ResourceFramework* (OASIS, 2006a) und *WS-Notification* (OASIS, 2006b) ausgewählt. Durch diese Spezifikationen können Webservices sowohl um zustandsbehaftete Eigenschaften als auch um Event-basierte Kommunikation erweitert werden.

Die Services der Montagezelle bilden eine Hierarchie, die in Abbildung 4 vereinfacht dargestellt ist. Der zentrale Dienst ist die Prozesssteuerung (*Process Control Service*), die alle anderen Dienste kontrolliert bzw. orchestriert (vgl. Erl, 2005). In diesem Dienst sind die beiden Prozessabläufe der Zelle unter Verwendung der untergeordneten Dienste implementiert. Der Prozesssteuerungsdienst überwacht und protokolliert die aktuellen Prozessabläufe und kann diese zudem auf Benutzerwunsch abbrechen oder pausieren. Die Kommunikation mit der graphischen Benutzerschnittstelle wird ebenfalls komplett über diesen Dienst abgewickelt.

Die Dienste der darunterliegenden Ebene repräsentieren die realen Komponenten der Montagezelle. Jeder dieser Dienste wurde als eigenständiger Webservice implementiert, mit jeweils einer eigenen, auf die Bedürfnisse angepassten Instanz der Robotics API. Eine Ausnahme bildet der Identifikationsdienst (*Identification Service*), der direkt auf dem darunterliegenden Webservice des Barcode-Scanners zugreift und dem gelesenen Barcode die entsprechende Werkstückbeschreibung zuordnet. Die Dienste dieser Ebene kommunizieren nicht untereinander, sondern ausschließlich über die Prozesssteuerung. Dazu wird sowohl synchrone Kommunikation (z.B. beim Anstoßen von Prozessschritten) als auch Event-basierte Kommunikation (z.B. für die Benachrichtigung der Prozesssteuerung beim Eintreffen eines neuen Werkstücks) verwendet.

[2] http://ws.apache.org/muse/

3 Fazit

In der vorliegenden Arbeit wurde der Einsatz service-orientierter Architekturen für die Steuerung von roboterbasierten Montagezellen untersucht. Die Basis bildet dabei – analog zu Unternehmensanwendungen – der Geschäftsprozess bzw. im Kontext der Fertigung der Automatisierungsprozess. Die Umsetzung automatisierter Roboterzellen durch service-orientierte Prinzipien bietet mehrere Vorteile:

- Services stellen abgeschlossene Komponenten mit eindeutig definierten Schnittstellen dar. Sie bilden damit Basisbausteine für eine Automatisierungslösung mit einer klar spezifizierten Aufgabe. Zudem lassen sich die Service-Implementierungen gegen neue oder verbesserte Implementierungen austauschen.
- Die Services können als Basisbaustein immer wieder neu zusammengesetzt und kombiniert werden, um komplexere Aufgaben, d.h. einen Automatisierungsprozess, abzubilden. Man spricht in diesem Zusammenhang von Orchestrierung.
- Durch Webservices können verteilte Systeme verhältnismäßig einfach und plattformunabhängig realisiert werden.

Die Vorteile service-orientierter Architekturen lassen sich jedoch nur so einfach erreichen, da durch die Verwendung der Robotics API und die Programmiersprache Java eine direkte Integration der Roboter, Werkzeuge und der weiteren Bestandteile der Zelle in eine Webservice-Landschaft ermöglicht wird.

Jedoch stellt die in dieser Arbeit vorgestellte Modellierung erst den Anfang dar. Die identifizierten Services sind sehr grobgranular und weisen nur einen geringen Grad an Wiederverwendbarkeit auf. Daher bestehen die nächsten Schritte in einer feingranularen Modellierung der oben beschriebenen Montagezelle unter besonderer Berücksichtigung von Wiederverwendbarkeit, aber auch Taktzeit und Robustheit. Ein weiterer Punkt auf der Agenda ist der Einsatz von (graphischen) Prozessbeschreibungssprachen (z.B. BPMN 2.0) und entsprechenden Ausführungsumgebungen (z.B. jBPM[3]) in der Roboterautomatisierung.

[3] http://www.jboss.org/jbpm/

Literatur

Angerer, A., Hoffmann, A., Schierl, A., Vistein, M., Reif, W.: *The Robotics API: An Object-Oriented Framework for Modeling Industrial Robotics Applications*; Proc. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipeh, Taiwan: October 2010

Angerer, A., Bischof, M., Chekler, A., Hoffmann, A., Reif, W., Schierl, A., Tarragona, C., Vistein, M.: *Objektorientierte Programmierung von Industrierobotern*, Internationales Forum Mechatronik 2010, Winterthur, Schweiz: November 2010

Erl, T.: *Service-Oriented Architecture: Concepts, Technology, and Design;* Prentice Hall, Upper Saddle River, NJ, USA: 2005

Hoffmann, A., Angerer, A., Ortmeier, F., Vistein, M., Reif, W.: *Hiding Real-Time: A new Approach for the Software Development of Industrial Robots;* Proc. 2009 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, St. Louis, MO, USA: Oktober 2009, S. 2108-2113

Jammes, F., Smit, H.: *Service-oriented Paradigms in Industrial Automation*; IEEE Transactions on Industrial Informatics, Jgg. 1 (1), Februar 2005, S. 62-70

Naumann. M., Wegener K., Schraft, R. D.: *Control Architecture for Robot Cells to Enable Plug and Produce*; Proc. 2007 IEEE International Conference on Robotics and Automation, Rom, Italien: April 2007, S. 287-292.

OASIS: *Web Services Resource 1.2*; Spezifikation, Version 1.2, April 2006

OASIS: *Web Services Base Notification 1.3*; Spezifikation, Version 1.3, Oktober 2006

OSGi Alliance: *OSGi Service Platform – Core Specification, Release 4*; Version 4.2, Juni 2009

OSGi Alliance: *OSGi Service Platform – Service Compendium, Release 4*; Version 4.2, August 2009

Veiga, G.; Pires, J.N.; Nilsson, N.: *On the Use of Service Oriented Software Platforms for Industrial Robotic Cells*; Proc. IFAC International Workshop Intelligent Manufacturing Systems, Alicante, Spanien: Mai 2007.

Vistein, M., Angerer, A., Hoffmann, A., Schierl, A., Reif, W.: *Interfacing Industrial Robots using Real-time Primitives*; Proc. 2010 IEEE International Conference on Automation and Logistics (ICAL), Hong Kong: August 2010