Simulationsmodellbasierte Steuerung einer Roboterzelle

G. Maletzki[1], T. Pawletta[1], P. Dünow[1], P. Manemann[2]
pawel@mb.hs-wismar.de
[1]Hochschule Wismar, FG Computational Eng. und Automation, PF 1210, 23952 Wismar
[2]IAV GmbH, Nordhoffstraße 5, 38518 Gifhorn

Kurzfassung

Der Beitrag enthält die Beschreibung einer simulationsmodellbasierten Roboterzellensteuerung. Es werden die Vorteile dieses Steuerungsansatzes gegenüber der traditionellen Roboterprogrammierung herausgearbeitet und an einem Anwendungsbeispiel erläutert.

1 Einleitung

Durch die voranschreitende Robotikforschung werden stetig neue Anwendungsfelder für Industrieroboter erschlossen. Aufgrund der großen Einsatzbreite werden die unterschiedlichsten Anforderungen an die Industrieroboter gestellt. Eine einfache Programmierung und die Einbindung unterschiedlicher externer Hardware sind dabei von besonderer Bedeutung. Für die simulationsbasierte Steuerungsprogrammierung von Produktionssystemen wurden bereits unterschiedliche Ansätze untersucht [1], [2], welche bis hin zur Realisierung "intelligenter" Steuerungen reichen [3], [4]. Ziel jeder Steuerungsimplementierung ist, dass der Entwurf und die Inbetriebnahme einer Steuerung möglichst einfach, schnell und kostengünstig umgesetzt werden kann. Um dieser Forderung gerecht zu werden, ist es notwendig Reimplementierungen zu vermeiden. Das kann erreicht werden, indem die bereits implemtierte Software, die für Simulationszwecke in der Steuerungsentwurfsphase genutzt wurde, in der Betriebsphase wiederverwendet wird [5]. Im Beitrag wird ausgehend von der traditionellen Roboterprogrammierung ein allgemeines Konzept zur simulationsmodellbasierten Robotersteuerung aufgezeigt, welches anschließend anhand einer konkreten Realisierung und Applikation vertieft wird.

2 Traditionelle Roboterprogrammierung versus simulationsmodellbasierte Robotersteuerungen

2.1 Traditionelle Roboterprogrammierung

Die Implementierung von Robotersteuerungen beginnt mit einer Spezifikation der Steuerungsaufgaben. Anschließend werden die Steuerungsaufgaben in einer Roboterprogram-

miersprache umgesetzt und mit speziellen Robotersimulationsumgebungen getestet bevor die Inbetriebnahme der Steuerung erfolgt.

Die Implementierung komplexer Robotersteuerungen erfordert dagegen einen größeren Aufwand. Nachdem die Spezifikation der Steuerungsaufgaben abgeschlossen ist, erfolgt in der Planungsphase die Bestimmung "optimaler" Steuerungsstrategien und die Dimensionierung der Roboterzelle mittels diskret-ereignisorientierter Simulationsmodelle. Im Anschluss wird die textuelle und formale Spezifikation geeigneter Steuerungsstrategien vorgenommen. Die ermittelten Strategien werden daraufhin von einem Steuerungsspezialisten in einer Roboterprogrammiersprache implementiert, getestet und in Betrieb genommen.

Ein Nachteil des traditionellen Ansatzes besteht im hohen Programmieraufwand. Die Steuerungsstrategien werden im Rahmen der Planungsphase mit Simulationswerkzeugen entwickelt und anschließend in der Realisierungsphase mit einer speziellen Roboterprogrammiersprache reimplementiert (Abb. 1). Ein weiteres Problem stellt die Einbindung unterschiedlicher externer Hardware dar. Die speziellen Roboterprogrammiersprachen unterstützen in der Regel nur die Integration von Hardwarekomponenten, die direkt vom Hersteller angeboten werden.

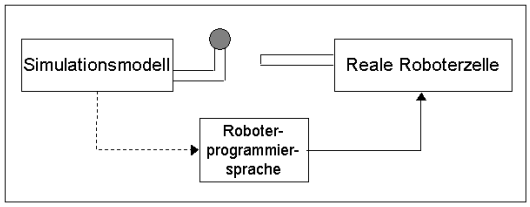

Abbildung 1: Entwurfssimulation & traditionelle Roboterprogrammierung

2.2 Simulationsmodellbasierte Robotersteuerungen

Das Konzept der simulationsmodellbasierten Robotersteuerung wurde entwickelt, um die Nachteile der traditionellen Roboterprogrammierung zu kompensieren. Die in der Planungsphase entwickelten Simulationsmodelle werden so aufgebaut, dass die Modellkomponenten direkt zu Prozesssteuerungsmodulen erweiterbar sind.

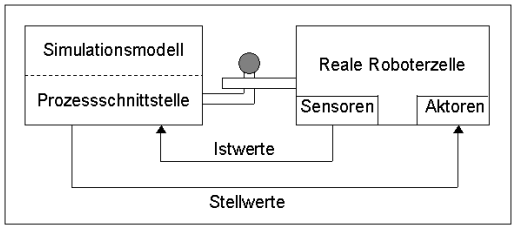

Abbildung 2: Simulationsmodellbasierte Robotersteuerung

Jede reale Materialflusskomponente, wie z.B. Puffer, Roboter, Maschine etc. wird als separate Modellkomponente (High-Level Komponente) abgebildet. Weiterhin werden im Modell der Material- und Informationsfluss strikt voneinander getrennt. In der Realisierungsphase werden die materialflussorientierten Modellkomponenten um eine Prozessschnittstelle (Low-Level Komponente) erweitert (Abb. 2) und das gesamte Simulationsmodell um eine Real-Time Synchronisation ergänzt. Abbildung 3 zeigt ausschnittsweise die Grundstruktur einer simulationsmodellbasierten Roboterzellensteuerung.

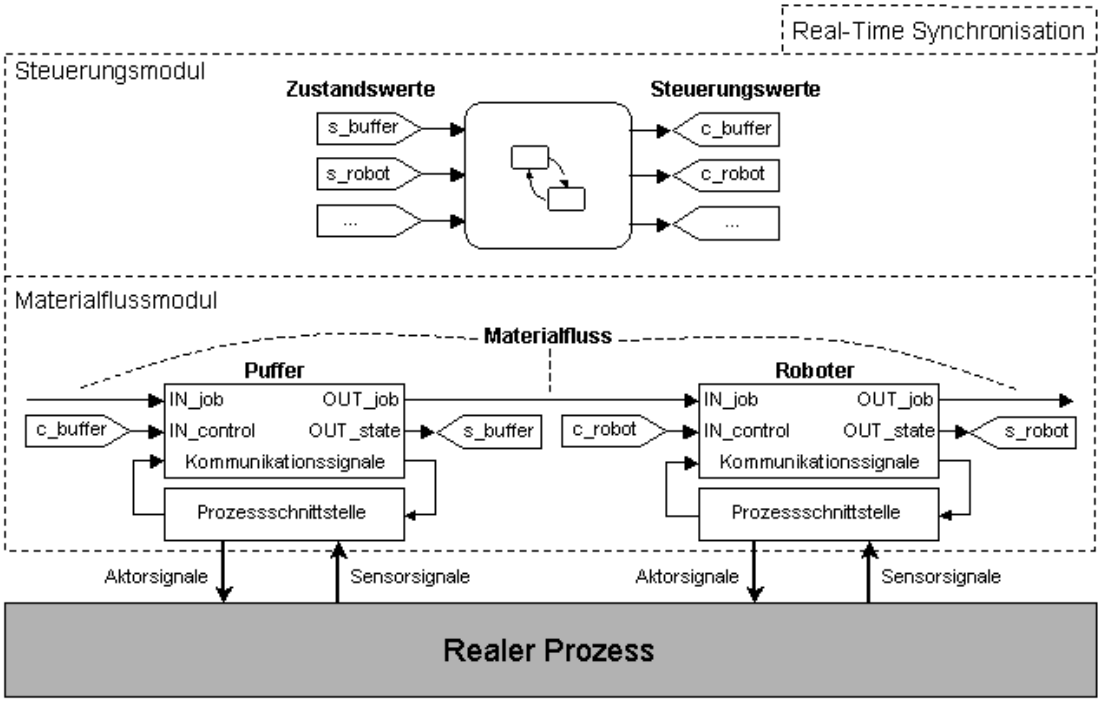

Abbildung 3: Struktur der simulationsmodellbasierten Steuerung einer Roboterzelle

Die unmittelbare Realisierung der Steuerungssoftware in einer Simulationsumgebung bietet als weiteren Vorteil eine einfache Umsetzung der in [4] beschriebenen Online-Integration von prädiktiven Simulationen während des Roboterbetriebes, um z.B. bei Prozessstörungen die Steuerung geeignet anzupassen.

3 KRL-Matlab-Stateflow Schnittstelle

An der Hochschule Wismar wird die Steuerungsprogrammierung von Industrierobotern, z.B. der Firma KUKA, untersucht. Für die Roboterprogrammierung wird von KUKA eine spezielle Roboterprogrammiersprache (KRL[1]) zur Verfügung gestellt. Die KRL ähnelt am

[1] **K**UKA **R**obot **L**anguage

ehesten der Programmierung mit C, jedoch mit deutlich geringerem Funktionsumfang. Die Möglichkeit, Visualisierungen zu erzeugen oder eine effektive Fehlersuche mittels Debugger durchzuführen, wird z.B. von der KRL nicht unterstützt. Des Weiteren ist die Integration zusätzlicher Hardwarekomponenten auf Produkte beschränkt, die direkt von KUKA angeboten werden.

Zur Kompensation dieser Nachteile und zur Realisierung simulationsmodellbasierter Steuerungskonzepte wurde die Roboterzelle um einen PC mit Matlab/Stateflow ergänzt. Für die Kommunikation zwischen dem KUKA Controller KR C3 und dem Matlab/Stateflow-PC wurde ein Interpreter in der KRL implementiert (Abb. 4). Der Interpreter ist für die Identifikation und Ausführung der Anweisungen verantwortlich, die vom Matlab/Stateflow-PC über die serielle Schnittstelle übermittelt werden. Entscheidende Vorteile dieser Lösung sind: *(i)* eine gemeinsame Softwarebasis von der Entwurfs- bis zur Realisierungsphase, *(ii)* eine simulationsgestützte Hard-Software-in-the-Loop Inbetriebnahme, *(iii)* eine verbesserte Integration externer Hardware sowie *(iv)* die Verwendung weiterer Matlab-Toolboxen vom Entwurf bis zum Betrieb.

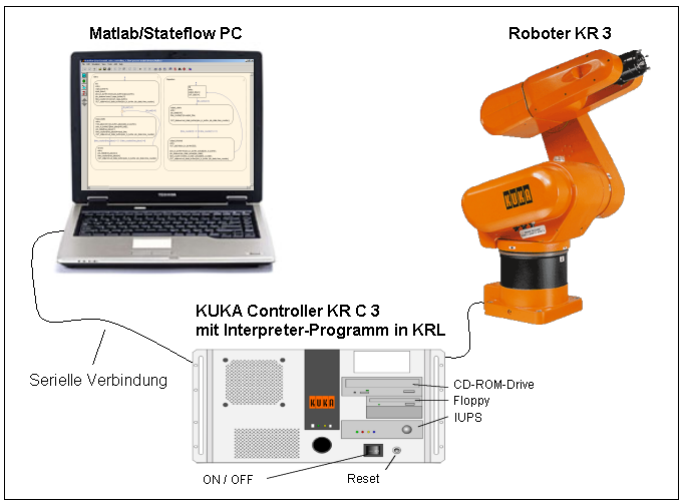

Abbildung 4: Integration von Matlab/Stateflow-PC, KRL-Controller und Roboter

4 Anwendungsbeispiel

Zur Validierung des aufgezeigten Steuerungskonzeptes wurden verschiedene Anwendungsbeispiele umgesetzt. An dieser Stelle soll ein materialtechnisches Bestückungsproblem betrachtet werden. Der Roboter hat die Aufgabe, unterschiedliche Fördereinheiten aus einem Eingangspuffer zu entnehmen und anschließend sortiert in drei Ausgangspuffer abzulegen.

Für die Identifikation der im Eingangspuffer befindlichen Fördereinheiten ist ein bildgebender Sensor erforderlich. Der bildgebende Sensor ist als Funkkamera ausgeführt, die

am TCP[2] des Roboters befestigt ist. Für die Übergabe der Kamerabilder an Matlab wird eine Matlab-Mex-Funktion genutzt. Die Auswertung der Bildinformationen erfolgt mit Hilfe der Image Processing Toolbox von Matlab. Die Aufnahme und Abgabe der Fördereinheiten wird durch einen schaltbaren Elektromagneten, der ebenfalls am TCP fixiert ist, realisiert.

Das simulationsmodellbasierte Steuerungsprogramm der Roboterzelle besteht aus einem Steuerungsmodul, das die Steuerungslogik enthält, und einem Materialflussmodul, wie in Abb. 5 dargestellt. Das Steuerungsmodul aus der Planungsphase wird direkt wiederverwendet und kann während der Betriebsphase beliebig ausgetauscht werden.

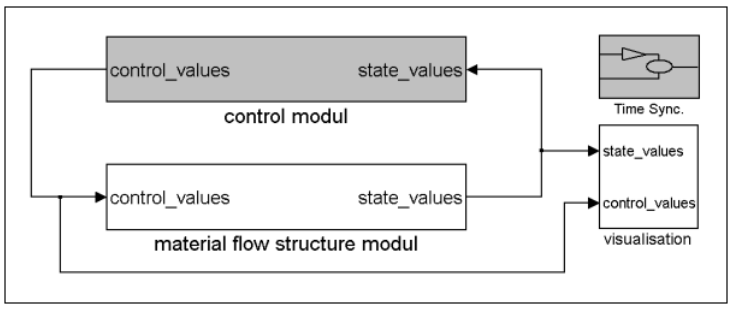

Abbildung 5: Grundstruktur des simulationsmodellbasierten Steuerungsprogramms

Das Materialflussmodul ist komponentenorientiert, gemäß den realen Elementen der Roboterzelle, aufgebaut. Jede Modellkomponente besteht aus einer High-Level- und einer Low-Level Komponente (Abb. 6). Alle Komponenten werden graphisch in Form von erweiterten Zustandsautomaten (Stateflow-Diagrammen) implementiert.

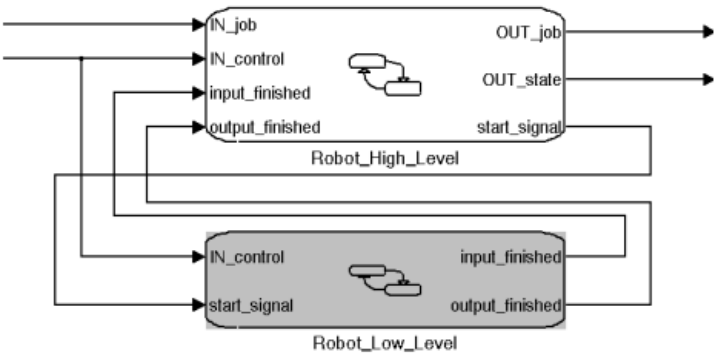

Abbildung 6: Struktur der Roboterkomponente

[2] **T**ool **C**enter **P**oint

5 Zusammenfassung

Durch die unmittelbare Erweiterung von Simulationsmodellen aus der Planungsphase zu Robotersteuerungssoftware kann der Entwurf und die Inbetriebnahme einer Roboterzellensteuerung einfacher, kostengünstiger und schneller erfolgen. Die aufwendige Umsetzung der entworfenen Steuerungsstrategien mit einer speziellen Roboterprogrammiersprache entfällt. Weiterhin unterstützt das aufgezeigte Konzept direkt die in [4] beschriebene Online-Integration von prädiktiven Simulationen im Rahmen der Prozesssteuerung und bietet damit eine Basis zur Realisierung hoch flexibler Prozesssteuerungen.

Durch die Erweiterung der Standardkomponenten einer KUKA Roboterzelle um einen PC mit Matlab/Stateflow und eine KRL-Matlab-Stateflow Schnittstelle werden die Vorteile der unterschiedlichen Systeme vereint. Die von der KRL gegebene Sicherheit bezüglich Arbeitsraumüberwachung, Endschalterkontrolle der einzelnen Roboterachsen etc. bleibt in der hochflexiblen Programmierumgebung Matlab/Stateflow erhalten. Darüber hinaus ist die Integration externer Hardwarekomponenten mit geringem Aufwand realisierbar. Die Fehlersuche wird durch die interaktive Arbeitsweise, graphische Programmierung und komfortablen Debugging-Möglichkeiten deutlich vereinfacht.

Literatur

[1] *H. Praehofer, G. Jahn, W. Jacak, G. Haider:* Supervising manufacturing system operation by DEVS-based intelligent control, IEEE Computer Society Press, 1994, S. 221-228

[2] *M. Kim, Y.-D. Kim:* Simulation-based Real-Time scheduling in a flexible manufacturing system, Journal of Manufacturing Systems 13/2, 1994, S. 85-93

[3] *F. Gonzales:* Intelligent control of flexible manufacturing systems using a distributed architecture, Proc. of the 17th International Conf. on Productio Research, 2003, S.12

[4] *T. Pawletta, M. Kremp, S. Pawletta, G. Colquhoun:* Optimisation of manufacturing control strategies using on-line simulation, Modeling and Simulation in Practice and Theory, Elsevier Science Publisher, Netherlands, (submitted for publication, 2005/02, 16 pages)

[5] *M. Kremp, T. Pawletta, G. Colquhoun:* The re-use of design phase simulation models for process control, International Journal of Production Research, Taylor and Francis LTD, (submitted for publication, 2004/12, 11 pages)