Model-Based Engineering mit Industriesteuerungen

Andreas Hofmann, Bosch Rexroth, 97816 Lohr am Main, Deutschland,
andreas.hofmann7@boschrexroth.de
Nils Menager, Bosch Rexroth, 97816 Lohr am Main, Deutschland,
nils.menager@boschrexroth.de
Stephan Schweig, Uni Duisburg-Essen, Lehrstuhl für Mechatronik, 47057 Duisburg,
Deutschland, schweig@mechatronik.uni-duisburg.de
Lars Mikelsons, Bosch Rexroth, 97816 Lohr am Main, Deutschland,
lars.mikelsons@boschrexroth.de

Kurzfassung

Das durchgängige Engineering über den gesamten Lebenszyklus ist neben der horizontalen und vertikalen Vernetzung die dritte Säule von Industrie 4.0. Durchgängigkeit im Engineering bedeutet dabei insbesondere Wiederverwendung von Modellen aus vorherigen Entwicklungsphasen. Beispiele hierfür sind die virtuelle Inbetriebnahme sowie die Codegenerierung. Dieser Beitrag stellt dar, wie diese modernen Engineering-Methoden bei der Verwendung von Rexroth Komponenten angewendet werden können. Die Kosten für die Inbetriebnahme neuer technischer Systeme beanspruchen heute einen erheblichen Anteil des Projektbudgets. Insbesondere die Optimierung des Steuerungscodes der Anlage erfordert einen hohen Zeitaufwand. So werden
bis zu 70% der Zeit, die für die Inbetriebnahme der Steuerungstechnik benötigt wird, für das Finden und Beheben von Softwarefehlern aufgewendet. Um die Anzahl der Prozessdurchläufe beim Kunden zu reduzieren, kann heute ein großer Anteil dieser Aufgaben virtuell gelöst werden. In diesem Beitrag wird ein Software-Framework für die virtuelle Inbetriebnahme vorgestellt. Dieses Framework kann für mechatronische Systeme, die von Rexroth Komponenten angetrieben werden, eingesetzt werden. Exemplarisch wird dies am Beispiel eines Deltaroboters aus dem Packaging-Bereich präsentiert. Dafür wird die reale SPS-Steuerung der Anlage mit einem Simulationsmodell des Deltaroboters gekoppelt. Zur einfacheren Interpretation der Simulationsergebnisse
werden diese mittels einer 3D-Visualisierung graphisch dargestellt. Die Modellierung des Deltaroboters erfolgt in der Modellierungssprache Modelica. Diese eignet sich besonders für domänenübergreifende Systeme. Außerdem erlaubt der objektorientierte Ansatz eine hohe Wiederverwendbarkeit der verwendeten Modelle. Die Kopplung von Rexroth Industriesteuergeräten und der Simulation wurde mit Hilfe des OpenCore-Interfaces

realisiert. Die entwickelten Modelle wurden durch Messungen mit verschiedenen Bewegungsabläufen des Deltaroboters bestätigt. Somit ist es möglich, zukünftige Anlagen dieses Typs virtuell in Betrieb zu nehmen und die Anzahl der Iterationen, während der Inbetriebnahme, deutlich zu reduzieren. Dies setzt voraus, dass das Simulationsmodell das dynamische Verhalten der Anlage ausreichend genau abbildet. Neben der virtuellen Inbetriebnahme ist die Codegenerierung eine wichtige Technologie, um vorhandene Modelle in weitere Entwicklungsphasen zu übernehmen. Insbesondere erlaubt die Codegenerierung die Übernahme von Modellen in die Betriebsphase des Systems. Der wohl bekannteste Anwendungsfall der Codegenerierung ist das Rapid Control Prototyping. Neben der Möglichkeit zur Generierung von Steuerungscode gibt es jedoch auch Use-Cases, die die Simulation eines physikalischen Systems auf dem Steuergerät voraussetzen. Hierzu zählt beispielsweise die modellbasierte Diagnose, die modellbasierte Regelung oder die virtuelle Inbetriebnahme auf Basis einer Simulation auf dem Steuergerät. In diesem Beitrag wird eine Toolchain zur Generierung von Code aus Modelica, der auf Rexroth Steuergeräten ausgeführt werden kann, vorgestellt.

1 Einleitung

1.1 Motivation

Industrie 4.0 skizziert die Industrielandschaft der Zukunft. Um hoch automatisierte, selbst einstellende und intelligent vernetzte Fertigungsprozesse zu realisieren werden geregelte und gesteuerte Anlagen durch cyber-physikalische Systeme ersetzt, welche sowohl untereinander als auch mit der Umwelt vernetzt sind. Diese horizontale wie vertikale Vernetzung führt zu einem deutlich höheren Wertschöpfungsanteil der Elektronik und Software. Konventionelle Entwicklungsmethoden sind dieser zunehmenden Komplexität nicht gewachsen. Stattdessen muss diesen Herausforderungen mit modellbasierten Entwicklungsmethoden begegnet werden, um die Wettbewerbsfähigkeit deutscher Unternehmen zu erhalten, vgl. [3]. Dabei stellt das durchgängige, modellbasierte Engineering über den gesamten Lebenszyklus neben der horizontalen und vertikalen Vernetzung
die dritte Säule von Industrie 4.0 dar. Durchgängigkeit im Engineering bedeutet dabei insbesondere Wiederverwendung von Modellen aus vorherigen Entwicklungsphasen. Neben dem klassischen Einsatz der Simulation als Entwicklungswerkzeug kann auch die Applikation und der Betrieb von Anlagen durch Simulationsmodelle umfangreich unterstützt werden. Beispiele hierfür sind die virtuelle Inbetriebnahme sowie die Codegenerierung. Die Inbetriebnahme (IBN) technischer Systeme, welche im Folgenden nach [1] als „funktionsgerechtes Einschalten des Systems in Verbindung mit dem zugehörigen Prozess“

verstanden werden soll, stellt einen wesentlichen Bestandteil der Entwicklung eines technischen Produktes dar. Hierbei beansprucht die IBN einer Anlage bis zu einem Viertel der kompletten Projektdauer, vgl. **Bild 1**. Innerhalb des technischen Systems wird diese Zeit vor allem für die Einrichtung der Steuerungstechnik und die Beseitigung von Softwarefehlern benötigt. Sind heute schon ein Großteil der Probleme der IBN einer technischen Anlage softwareseitig verursacht, so ist anzunehmen, dass die Inbetriebnahmezeiten durch die vierte industrielle Revolution (I 4.0) auf Basis cyber-physikalischer Systeme steigen wird.

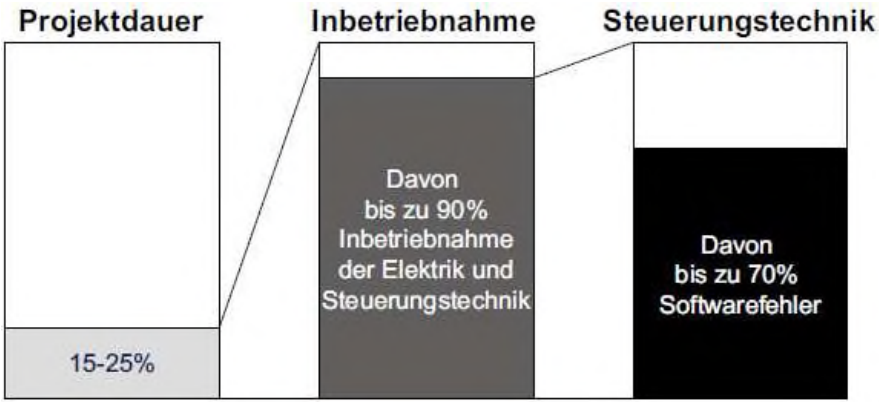

Bild 1 Anteil der Inbetriebnahmezeit an der Projektdauer, [2]

Dies wird zum einen durch den deutlich ansteigenden Anteil an Elektronik und Software im Maschinenbau und die horizontale wie vertikale Vernetzung, vgl. [3] und **Bild 2**, verursacht, aber auch durch die steigende Komplexität der Softwareprojekte selbst.

Bild 2 Anstieg der Produktkomplexität und Leistungsfähigkeit disziplinspezifischer Entwicklungsmethoden, [4]

Dementsprechend müssen auch Methoden, die schon jetzt für eine deutliche Reduzierung von Inbetriebnahmezeiten eingesetzt werden, für diese neuen Herausforderungen weiterentwickelt werden. Besonders die virtuelle Inbetriebnahme bietet hier gute

Einsparpotentiale bei der Inbetriebnahmezeit, aber darüber hinaus auch beim Finden von Fehlern und der Steigerung der Softwarequalität [5], vgl. **Bild 3**.

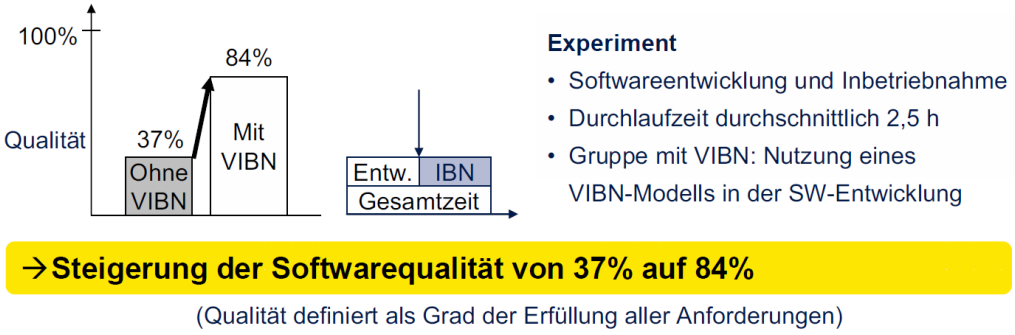

Bild 3 Steigerung der Softwarequalität durch virtuelle Inbetriebnahme, [5]

Allerdings müssen auch die Verfahren der virtuellen Inbetriebnahme, wie beispielsweise die Hardware-In-The-Loop-Simulation, an die neuen Herausforderungen angepasst werden. Eine weitere wesentliche Ausprägung des modellbasierten Engineerings ist die Codegenerierung, also die Erzeugung von Code aus Simulations- bzw. Engineeringtools heraus. Der generierte Code kann für eine Vielzahl von Anwendungen verwendet werden. Ein klassisches Einsatzfeld ist das *Rapid Control Prototyping*. Während des Entwicklungsprozesses einer neuen Anlage wird in der Regel ein Simulationsmodell der Anlage innerhalb einer Simulationsumgebung erstellt. Für eine Untersuchung der Dynamik des Systems ist innerhalb der Simulationsumgebung zusätzlich ein Modell des Reglers erforderlich. In einer späteren Phase des Entwicklungsprozesses ist es notwendig, den Regler unter realen Bedingungen, d.h. auf echter Regelungshardware, zu testen, um beispielsweise die Echtzeitfähigkeit des Regelungs-Algorithmus zu validieren. An dieser Stelle wird bisher die bereits vorhandene Implementierung in Form des Simulationsmodells nicht weiter verwendet. Stattdessen wird innerhalb des Engineeringtools der Steuerung der Regelungs-Algorithmus komplett neu implementiert. Häufig wird dabei auf einen bereits implementierten Standard-Regler zurückgegriffen, auch wenn dieser unter Umständen weniger performant ist. Dieses Vorgehen bringt insgesamt einige wesentliche Nachteile mit sich. Zunächst bedeutet eine Re-Implementierung bereits vorhandener Teile immer zusätzlichen Zeit- und damit Kostenaufwand sowie eine potentielle Fehlerquelle. Da der Regler bei der Re-Implementierung in der Regel in einer anderen Sprache (zumeist SPS Programmiersprachen nach IEC 61131) implementiert wird als innerhalb der Simulationsumgebung, kann nicht garantiert werden, dass sich beide Regler gleich verhalten. Aus diesem Grund ist es wünschenswert, den bereits vorhandenen Regler aus der

Simulationsumgebung auf der Regelungs-Hardware weiterverwenden zu können. Ein Lösungsansatz ist es, aus dem Simulationsmodell des Reglers ausführbaren Code zu generieren, der auf der Steuerung in Echtzeit ausgeführt wird. Neben der Weiterverwendung des Simulationsmodells des Reglers als Regelungs-Algorithmus auf der Steuerung ist es ebenso möglich, das Simulationsmodell der Regelstrecke weiterzuverwenden und den daraus generierten Code auf der Steuerung auszuführen. Dies kann beispielsweise zu Diagnosezwecken eingesetzt werden. Hierbei wird mit Hilfe des Simulationsmodells das erwartete Verhalten des Systems vorausberechnet und mit dem tatsächlichen, von Sensoren gemessenen Verhalten, verglichen. Abweichungen zwischen erwarteten und gemessenen Ergebnissen lassen auf einen Fehlerfall schließen und ermöglichen, bei entsprechend genauen Fehlermodellen, unter Umständen sogar eine Identifikation der Ursache des Fehlers. Zusätzlich ist es denkbar, auch zukünftig auftretende Fehler frühzeitig detektieren zu können, was zu einer Reduzierung der Stillstands Zeiten führen kann.

1.2 Aufbau der Arbeit

In Kapitel 2 wird der Begriff virtuelle Inbetriebnahme näher beleuchtet, Grundprinzip und Anforderungen untersucht und auf geläufige Verfahren eingegangen. Anschließend wird die virtuelle Inbetriebnahme konkret für Rexroth Industriesteuerungen und ihre Besonderheiten, welche eine Erweiterung der Hardware-In-The-Loop-Simulation darstellt, präsentiert. Die virtuelle Inbetriebnahme mit Rexroth Steuerungen wird am konkreten Beispiel eines Delta-Roboters dargestellt. In Kapitel 3 wird eine auf offenen Standards basierende Toolchain für die Codegenerierung vorgestellt. Dabei werden zunächst die Anforderungen an die Toolchain abgeleitet und anschließend die einzelnen Komponenten beschrieben, die darin zum Einsatz kommen. Anschließend werden diese Komponenten miteinander verknüpft und die Gesamtstruktur der Toolchain präsentiert. Im letzten Kapitel wird der Inhalt des Beitrages noch einmal zusammengefasst und ein Ausblick gegeben.

2 Virtuelle Inbetriebnahme

2.1 Grundlagen der virtuellen Inbetriebnahme

2.1.1 Definition, Grundprinzip und Anforderungen

Unter virtueller Inbetriebnahme (VIBN) wird im Folgenden die Überprüfung der Funktionsfähigkeit bzw. des Funktionsablaufes einer Anlage mit Hilfe eines virtuellen Simulationsmodells der Anlage bezeichnet. Dabei wird davon ausgegangen, dass die Steuerung analog zur Anlage als virtuelles Modell vorliegt oder als reale Steuerungshardware mit darauf laufender Steuerungsapplikation zur Verfügung steht.

Grundprinzip der VIBN ist die Kopplung einer virtuellen Anlage mit einer virtuellen oder realen Steuerung. Durch die Simulation kann bereits in frühen Stadien der Entwicklung die Funktionsfähigkeit von Steuerungsapplikationen überprüft werden. Wenn die virtuelle Inbetriebnahme parallel zur Fertigung der Anlage erfolgt, können nach [5] speziell Fehler der Steuerungsapplikation gefunden und behoben werden. Dadurch können bei der IBN des technischen Systems erhebliche Inbetriebnahmezeiten und dadurch Kosten eingespart werden, vgl. **Bild 4**. Darüber hinaus hilft die VIBN zusätzlich die Qualität der Steuerungsprogramme zu verbessern. Wird die Steuerungsapplikation erst während der tatsächlichen IBN entwickelt und getestet, werden üblicherweise nicht ausgereifte Steuerungsapplikationen auf der Steuerung implementiert, was zu unerwartetem Verhalten führen kann.

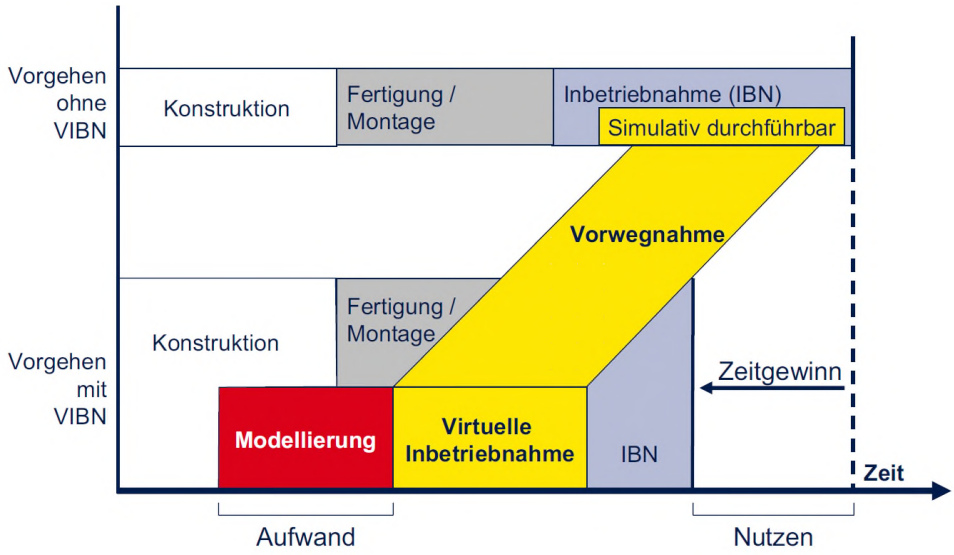

Bild 4 Grundidee der virtuellen Inbetriebnahme nach Wünsch und Zäh, [5]

Als zentrales Ziel der VIBN soll daher im Folgenden das frühzeitige Auffinden von Fehlern in der Steuerungsapplikation eines technischen Systems definiert werden, um Kosten zu sparen und die Qualität des Anlagenverhaltens zu verbessern. Durch die Integration von Simulation in den Prozess der (virtuellen) Inbetriebnahme ergeben sich zusätzlich noch weitere Möglichkeiten. Dies beinhaltet zum einen die Durchführung gefahrloser Tests unter einstellbaren äußeren Einflüssen, die mit geringen Kosten verbunden sind.
Zum anderen können problemlos Fehler injiziert werden und so die Reaktion der Anlage und der Steuerung auf Fehler untersucht werden, lange bevor die echte Anlage überhaupt aufgebaut ist. Allerdings ist für die VIBN zusätzlicher Aufwand in Form der Modellbildung der Anlage erforderlich. Dies stellt vor allem für viele kleine und mittelständische Unternehmen

einen unbekannten Schritt dar, der ein Umdenken in der Entwicklung erfordert. Im Zuge des Zukunftsprojektes Industrie 4.0 des Bundesministeriums für Bildung und Forschung ist dieses Umdenken zur modellbasierten Entwicklung aber unabhängig von der VIBN nötig, um die Wettbewerbsfähigkeit deutscher Unternehmen zu erhalten, vgl. [3]. Nichtsdestotrotz ergibt sich daraus eine Anforderung an die VIBN technischer Systeme, sich möglichst nahtlos in die durchgängige modellbasierte Entwicklung zu integrieren, damit keine neuen Modelle erstellt, sondern virtuelle Abbilder der Anlage aus früheren Produktentstehungsphasen verwendet werden können.

2.1.2 Kopplungsstrategien der VIBN

Die Kopplung zwischen Steuerung und virtueller Anlage stellt einen zentralen Punkt der VIBN dar und hat wesentlichen Einfluss auf die Aussagekraft der Ergebnisse. Die beiden Kopplungsstrategien Hardware-In-The-Loop (HiL) und Software-In-The-Loop (SiL) sind dabei die wichtigsten Verfahren und sollen näher beschrieben werden. Da die Begriffe HiL und SiL nicht zwangsweise eindeutig definiert sind, wird an dieser Stelle zunächst eine kurze Definition gegeben. Des Weiteren werden die Verfahren für die eingangs gestellten Anforderungen, das Finden von Fehlern in der Steuerungsapplikation und die Integration in eine durchgängige modellbasierte Entwicklung, bewertet.

Software-In-The-Loop-Kopplung

Unter einer SiL-Kopplung wird im Folgenden die Strategie der Verknüpfung des virtuellen Modells der Anlage mit einem virtuellen Abbild der Steuerung verstanden. Dadurch, dass die Steuerung nur in Form eines Modells vorliegt, kann das komplette System auf dem gleichen PC und häufig in der gleichen Software simuliert werden. Da keine reale Steuerungshardware verwendet wird, sind folglich auch keine Echtzeitanforderungen zu erfüllen. Daraus ergibt sich, dass beliebig komplexe virtuelle Modelle der Anlage für die VIBN verwendet werden können und somit eine hohe Durchgängigkeit durch die Verwendung von Modellen aus früheren Entwicklungsphasen ermöglicht ist. Da bei der SiL-Kopplung aber keine realen Steuerungsapplikationen verwendet werden, kann oftmals nicht das reale Verhalten der Anlage abgebildet werden und speziell das Finden von Fehlern ist nicht möglich.

Hardware-In-The-Loop-Kopplung

Die HiL-Kopplung wird im Folgenden als Kopplungsverfahren definiert, bei der eine reale Steuerungshardware, auf welcher die echte Steuerungsapplikation der Anlage läuft, mit dem Simulationsmodell der Anlage gekoppelt wird. Aufgrund dessen, dass die Steuerung in harter Echtzeit (d.h. mit festem Zeittakt, welcher nicht überschritten werden darf) arbeitet, muss die Kopplung in der Regel mittels echtzeitfähigem Bus (z.B. Sercos, Profibus, etc.) und

Computer mit einem in Echtzeit laufenden Betriebssystem erfolgen. Daraus ergeben sich ein hoher Aufwand für die Infrastruktur und zusätzlich erhebliche Einschränkungen an das Simulationsmodell der Anlage aufgrund der Echtzeitanforderungen der Steuerungshardware. Vor dem Hintergrund der durchgängigen modellbasierten Entwicklung ist der Ansatz der HiL-Kopplung im Allgemeinen wenig geeignet, da hierfür in der Regel spezielle Anlagenmodelle erstellt werden müssen, um die Echtzeitanforderungen zu erfüllen. Das Verwenden von Modellen aus vorherigen Entwicklungsphasen ist daher kaum möglich. Für das Finden von Fehlern in der Steuerungsapplikation ist das Verfahren der HiL-Kopplung dagegen gut einsetzbar, da die echte Steuerungsapplikation auf der echten Steuerungshardware direkt in die VIBN einbezogen wird.

2.2 VIBN mit Rexroth Industriesteuerungen

2.2.1 Modelica

Die Modellbildung und Simulation technischer Systeme, die im vorherigen Kapitel immer nur als Blackbox behandelt wurde, stellt einen integralen Bestandteil der VIBN dar. Speziell unter dem Gesichtspunkt hoch automatisierter, selbst einstellender und intelligent vernetzter Produktionsanlagen steigt die Nachfrage nach cyberphysikalischen Industrieanlagen und daher der Bedarf nach detaillierten Simulationsmodellen, die über einzelne Domänen hinaus gehen, deutlich. Bei Bosch Rexroth wird für die Entwicklung dieser detaillierten, domänübergreifenden Systemmodelle Modelica verwendet, vgl. [6], [7]. Modelica [8], [9], als universelle Modellierungssprache, ist der fortschrittlichste Ansatz um derartige domänübergreifende Modelle zu entwickeln. Im Gegensatz zu üblicher Systemengineering-Software ermöglicht Modelica dem Anwender, Systeme mittels akausaler Gleichungen zu beschreiben, ohne sich Gedanken über die mathematischen Details machen zu müssen. Die Transformation in ein lösbares mathematisches Modell wird automatisch durch einen Modelica Compiler realisiert. Die verschiedenen kommerziellen und freien Modelica-basierten Modellierungs- und Simulationsprogramme sind bereits heute Stand der Technik bei Entwurf und Optimierung von Systemen und Arbeitsabläufen. Da Modelica ein nichtproprietärer Sprachstandard ist, arbeitet eine große Community bestehend aus akademischen und industriellen Vertretern an der Weiterentwicklung der Kapazitäten von Modelica, z.B. im Bereich von modellprädiktiver Regelung oder cloud-basierter Simulation. Auch der weit verbreitete Modellaustauschstandard FMI wird von der Modelica-Community gepflegt und weiterentwickelt.

2.2.2 Das OpenCore-Interface – Softwareschnittstelle zur Rexroth Industriesteuerung

Das Rexroth OpenCore-Interface [10] ist eine universelle Schnittstelle, mit der ein direkter Funktionszugriff auf den Steuerungs- und Antriebskern von Rexroth Industriesteuerungen ermöglicht wird. Mit diesem können mit modernen Hochsprachen wie C++ oder Java Softwareprogramme geschrieben werden, die es ermöglichen, Antriebs- und Steuerungssysteme mit klassischen IT-Systemen zu kombinieren, vgl. [11]. Für die VIBN von Rexroth Industriesteuergeräten wird das OpenCore-Interface in der Modelica-Bibliothek mlpi4Modelica implementiert. Diese Bibliothek, die Kernfunktionen des Rexroth OpenCore Software Development Kits aufruft, ermöglicht aus einer Modelica-basierten Simulationsumgebung direkt auf die Steuerung zuzugreifen.

2.2.3 Erweiterung der HiL-Kopplung mit dem OpenCore-Interface

Die VIBN mit Rexroth Industriesteuerungen basiert auf dem Ansatz der HiL-Kopplung, erweitert diesen aber erheblich, um die beiden zentralen Anforderungen der durchgängigen modellbasierten Entwicklung und dem Auffinden von Fehlern in der Steuerungssoftware zu ermitteln. Erreicht wird dies dadurch, dass die limitierenden Echtzeitanforderungen mit Hilfe des OpenCore-Interfaces aufgehoben werden, indem die Steuerung in einen Simulationsmodus geschaltet wird. In diesem wird der Motionkernel der Steuerung, der für den Aufruf der einzelnen Motion- und Sercostasks zuständig ist, in einen Zustand versetzt, in welchem die Tasks nicht mehr synchron zum internen Takt der Steuerung, sondern durch externe Vorgabe aufgerufen werden, vgl. **Bild 5**.

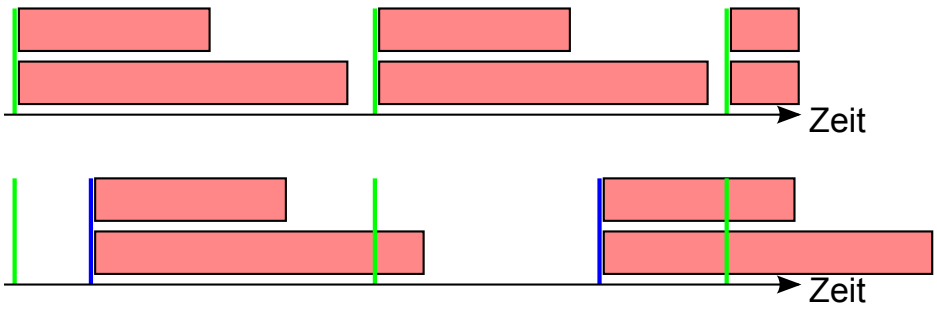

Bild 5 Modi der Steuerung
oben: zyklisch Abarbeitung zweier Motiontasks (rot) mit dem internen Takt (grün)
unten: Abarbeitung der Motiontasks durch externe Benutzervorgabe (blau)

Dennoch wird innerhalb der Firmware der Steuerung sichergestellt, dass die Steuerung konsistent und deterministisch arbeitet. Das manuelle Ansprechen der Motion- und Sercostasks ermöglicht es, beliebig komplexe virtuelle Modelle der Anlage, deren Simulation

in der Regel langsamer als der Echtzeittakt der Steuerung abläuft, mit der Steuerung zu kombinieren. Somit ist über diese Technik neben der guten Fehleranalyse der Steuerungsapplikation auch eine vollständig durchgängige modellbasierte Entwicklung gewährleistet. Aufgrund des Entfallens der Echtzeitanforderungen wird die komplizierte Infrastruktur aus echtzeitfähigem Bus und Realtime-PC, welche üblicherweise für die HiL-Kopplung erforderlich ist, durch einen Standardcomputer und eine Ethernet-Verbindung ersetzt. Die Synchronisierung der Daten und das Anstoßen der
Tasks der Steuerungsapplikation während der Simulation wird von der Modelica-Bibliothek mlpi4Modelica übernommen, welche die Funktionalität des OpenCore-Interfaces für Modelica-basierte Modellierungs- und Simulationstools zur Verfügung stellt und Teil des Rexroth OpenCore Software Development Kits ist. Zu Beginn des definierten Austauschzeitpunktes (üblicherweise der Motion- bzw. Sercostakt des Steuerungsapplikation auf der Industriesteuerung) werden von der Simulation alle relevanten Daten an die Steuerung übertragen. Anschließend sendet die Simulation dem Motionkernel der Steuerung das Signal, die Motion- und Sercostasks abzuarbeiten. Im letzten Schritt der Kommunikation werden alle relevanten Daten von der Steuerung abgefragt und zur Simulation transferiert. Die Zeitschritte zwischen den Austauschzeitpunkten rechnet nur die Simulation, wobei die Daten der Steuerung konstant gehalten werden, was auch dem realen Verhalten während des Betriebs an der echten Anlage entspricht.

2.2.4 Visualisierung technischer Systeme für dieVIBN

Die Auswertung von Simulationsläufen einer Anlage stellt eine anspruchsvolle Aufgabe dar. Gerade im Bereich Robotik sind Graphen mit dem Bewegungsverhalten der Anlage oft nur schwer interpretierbar. Daher wurde bei Bosch Rexroth parallel zur Entwicklung der VIBN-Schnittstelle für Industriesteuerungen damit begonnen, eine Visualisierung zu entwickeln, die es ermöglicht, das Bewegungsverhalten einer Anlage nachzuvollziehen. Obwohl die meisten Modelica-basierten Modellierungs- und Simulationstools eine Visualisierung der Mechanik aus dem Modell erzeugen können, sind diese in ihrem Umfang eingeschränkt. Diese eigenständige Visualisierung ist in der Lage, ebenfalls unter Verwendung des OpenCore-Interfaces, anlagenrelevante Daten aus der Steuerung zu extrahieren. So kann beispielsweise die komplette Geometrie von Robotern zur Laufzeit direkt aus der Steuerung abgefragt werden und zusätzlich der Arbeitsraum berechnet und dargestellt sowie der Arbeitsbereich und Hindernisse visualisiert werden. Dabei können speziell letzte genutzt werden, um das Eindringen der Anlage in den Bereich eines anderen Systems zu detektieren. In **Bild 6** ist die Visualisierung für einen Delta-Roboter aus Steuerungsdaten gezeigt.

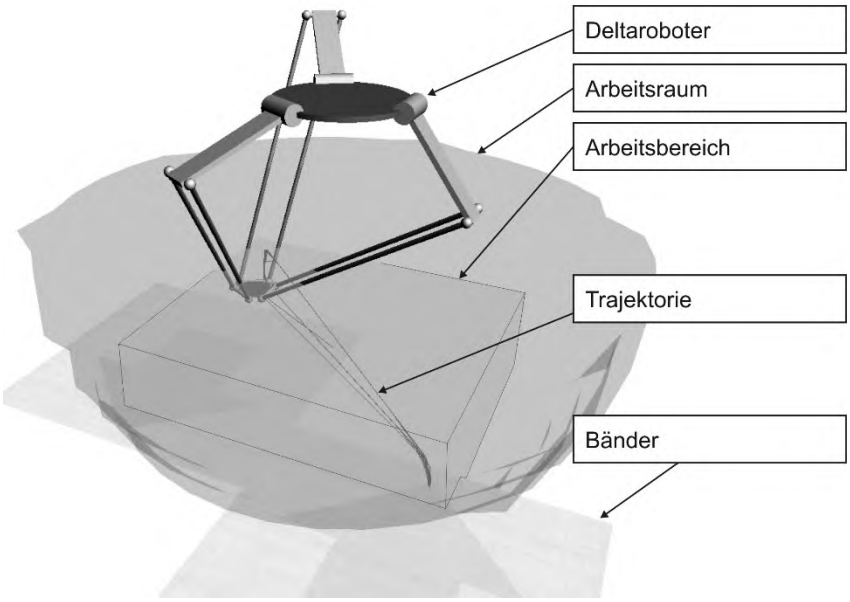

Bild 6 Visualisierung eines Deltaroboters

Die Visualisierung erfolgt mit der OpenSource-Bibliothek *Object Oriented Graphics Rendering Engine*. Technische Details können unter [13] eingesehen werden. Ein Teilprogramm der Visualisierung kann im Live-Betrieb zusätzlich verwendet werden, um Daten an realen Anlagen für die Analyse aufzuzeichnen. Dadurch können die vorgestellten Methoden nicht nur zur Inbetriebnahme, sondern auch zur modellgestützten Diagnose verwendet werden.

2.3 Durchführung der VIBN eines Delta-Roboters

Delta-Roboter werden in der Industrie heute vielfach eingesetzt. Aufgrund ihrer parallelkinematischen Bauart erreichen sie hohe Geschwindigkeiten und Beschleunigungen und einen hohen Durchsatz bei einem geringen Energiebedarf. Diesen Vorteilen gegenüber stehen allerdings eine geringe Lastaufnahme und ein erhöhter Aufwand für die Steuerung, vgl. [14]. Für die Programmierung von Delta-Robotern an Rexroth Industriesteuerungen sind die Transformationen zwischen den Motoren und dem Endeffektor des Roboters bereits in der Entwicklungsumgebungen für Steuerungsapplikationen hinterlegt. Innerhalb der Steuerungsapplikation werden die Algorithmen für die Generierung der Sollpositionen, beispielsweise aus den Input-Daten einer Kamera, implementiert. Diese Sollwerte werden anschließend an den Antriebsregler übertragen, der daraufhin den Motor entsprechend

ansteuert. Da allerdings keine Rückführung der Istwerte des Motors an die Steuerung erfolgt, kann nicht garantiert werden, dass der Roboter die vorgegebenen Sollpositionen erreicht. Ziel der VIBN war es, für eine vorgegebene Steuerungsapplikation zu prüfen, ob der Delta-Roboter die geforderten Sollpositionen erreicht. Zur Überprüfung der Steuerungsapplikation wurde im Modelica-Tool Dymola [15] ein Mechanikmodell des Roboters erstellt. Dieses wurde über vorhandene Modelica-Modelle von Antriebsregler und Motor mittels Elementen der Bibliothek mlpi4Modelica direkt mit der Steuerungsapplikation gekoppelt, vgl. **Bild 7**.

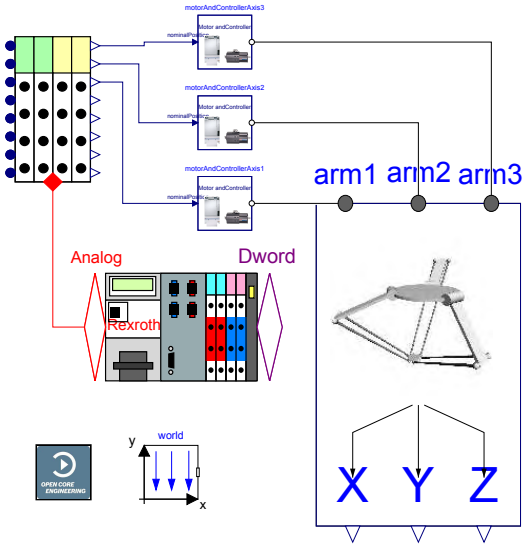

Bild 7 Simulationsmodell des Delta-Roboters mit den Kopplungskomponenten der Modelica-Bibliothek mlpi4Modelica

Obwohl vor allem der Mechanikaufbau des Anlagenmodells sehr einfach gestaltet ist, wäre eine HiL-Kopplung zur Überprüfung der Steuerungsapplikation nicht möglich gewesen, da der mechanische Aufbau des Roboters auf nichtlineare Gleichungssysteme führt, bei denen nicht sichergestellt werden kann, dass die Lösung innerhalb einer festen Anzahl von Iterationen konvergiert. Die Modelle von Antriebsregler und Motor, die für die Anforderung an die Simulation relevant sind, würden die Echtzeitanforderungen der HiL- Simulation, aufgrund ihrer Komplexität, ebenfalls nicht erfüllen. Mit Hilfe der Rexroth HiL-Kopplung war es dennoch möglich, die Anforderung an die Steuerungsapplikation zu überprüfen. Die Kopplung zwischen Simulations-PC und Steuerungshardware erfolgte, wie bereits zuvor beschrieben, mittels Ethernet-Verbindung.

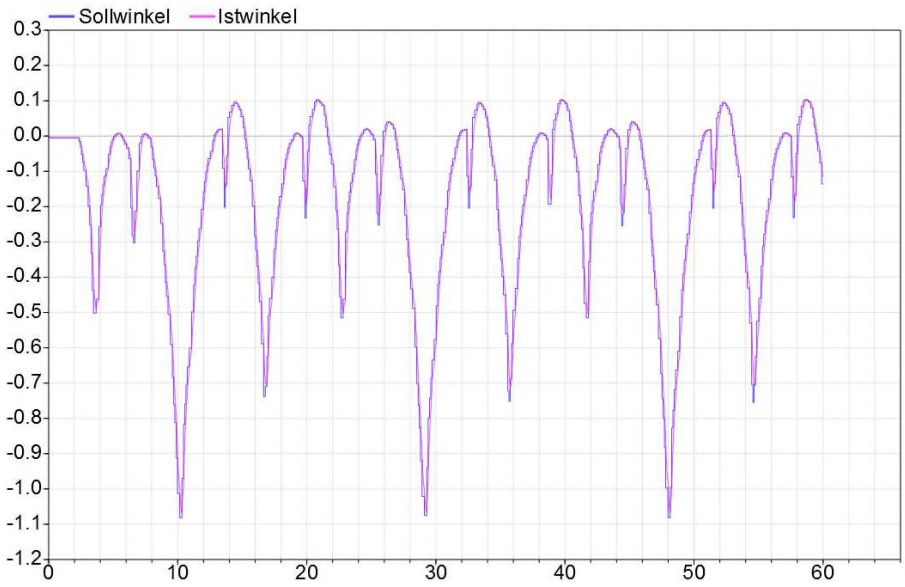

Bild 8 Sollwinkel [Rad] der Steuerung und Istwinkel [Rad] der Simulation eines der Motoren des Delta-Roboters über die Simulationsdauer

Wie in **Bild 8** zu sehen ist, konnte gezeigt werden, dass die Sollpositionen der Steuerungsapplikation vom Roboter erreicht werden können und die Applikation in der realen Anlage verwendet werden kann.

3 Codegenerierung

3.1 Toolchain für modellbasierte Entwicklung

Wie bereits in der Einleitung erwähnt, ist neben der VIBN die Codegenerierung eine wichtige Ausprägung der durchgängigen modellbasierten Entwicklung. Es existieren bereits Möglichkeiten, Code aus Simulationsmodellen zu generieren und diesen auf einer Echtzeit-Hardware auszuführen, beispielsweise bei der Verwendung einer Toolchain auf Basis von MATLAB/Simulink. Für Bosch Rexroth hat diese Toolchain jedoch einige Nachteile. Ein Grund dafür ist, dass die beschriebene Toolchain aufgrund der hohen Lizenzgebühren sehr kostenintensiv ist. Der Großteil der Rexroth-Kunden haben häufig keine Möglichkeit, diese Toolchain zu verwenden, weswegen die Anwendung der Methoden der modellbasierten Entwicklung diesen Kunden nicht möglich ist. Des Weiteren ist das Codegenerierungs-Modul von MATLAB/Simulink eine Blackbox und kann nicht verändert werden. Sollen bereits bestehende Funktionen (z.B. einer API) unmittelbar mit in den generierten Code integriert werden, ist es nicht möglich, das Codegenerierungs-Modul anzupassen. Ein dritter Nachteil ist die Release-Häufigkeit der Software (i.d.R. zweimal jährlich). Da bei einem neuen Release nicht ersichtlich ist, welche Änderungen an der Codegenerierung vorgenommen

wurden, müssen alle bestehenden Modelle mit der Erscheinung einer neuen MATLAB/Simulink Version neu getestet werden. Zusätzlich ist MATLAB/Simulink zwar sehr gut zum Regler-Design geeignet, für die Beschreibung der Regelstrecke ist Simulink jedoch nur mäßig geeignet. Hierfür sind objekt-orienterte Modellierungssprachen wie Modelica deutlich besser geeignet. Aus diesem Grund sind häufig zwei unterschiedliche Simulationsumgebungen zur Beschreibung des Reglers und der Regelstrecke notwendig. Daher wird in diesem Beitrag eine alternative Toolchain vorgestellt, die die zuvor genannten Nachteile nicht aufweist.

Die auf offenen Standards basierende Toolchain kann schließlich verwendet werden, um Code aus Simulationsmodellen zu erzeugen und diesen auf Rexroth-Industriesteuerungen auszuführen. Zur Validierung der Funktionsweise der Toolchain wird ein Beispiel für den Anwendungsfall *Rapid Control Prototyping* gewählt. Aus einer Simulationsumgebung heraus wird automatisiert Regler-Code erzeugt, der anschließend in Echtzeit auf der Steuerung ausgeführt wird. Im Folgenden werden zunächst die Anforderungen an die Toolchain präsentiert.

3.1.1 Anforderungen an die Toolchain

Die Anforderungen ergeben sich unmittelbar aus den Nachteilen der MATLAB/Simulink-Toolchain, die zuvor beschrieben wurden. Die Codegenerierung soll flexibel sein und bei Bedarf angepasst werden können, um beispielsweise bereits vorhandene Funktionen einer Schnittstelle verwenden zu können. Weiterhin soll die Toolchain nicht hauptsächlich auf kommerziellen Tools basieren, sondern stattdessen offene Standards verwenden. In der Folge werden externe Abhängigkeiten vermieden und eine breite Verfügbarkeit sichergestellt. Darüber hinaus ermöglicht die Verwendung von offenen Standards die Integration in bestehene IT-Infrastrukturen. Eine weitere Anforderung ist es, dass die Toolchain einfach und intuitiv zu bedienen ist.

3.1.2 Komponenten der ToolchainModelica/Compiler

Den Ausgangspunkt der entwickelten Toolchain stellen Modelica-Modelle dar. Um Modelica-Modelle simulieren zu können, wird ein Modelica-Compiler benötigt. Es existieren sowohl kommerzielle (z.B. Dymola, SimulationX) als auch Open-Source-Modelica-Compiler (z.B. Open-Modelica, JModelica.org). Für die Entwicklung dieser Toolchain wird der OpenModelica-Compiler verwendet. Ein Teil des Compilers ist das Codegenerierungs-Modul. An dieser Stelle besitzt Bosch Rexroth ein eigenentwickeltes Codegenerierungs-Modul zur Generierung von C++-Code aus den Modelica-Modellen. Aus diesem Grund ist die Codegenerierung komplett flexibel, da die Codegenerierung je nach Bedarf angepasst

werden kann. Für den detaillierten Aufbau des OpenModelica-Compilers und der Funktionsweise der einzelnen Komponenten sei auf [16] verwiesen.

Simulationskern

Der mit Hilfe desModelica-Compilers erzeugte C++-Code beinhaltet die Modellbeschreibung bzw. die zugrunde liegenden mathematischen Gleichungen. Allerdings beinhaltet der Code keine Informationen darüber, wie die Gleichungen gelöst werden. An dieser Stelle wird ein Simulationskern benötigt, der unter anderem die numerischen Verfahren zur Lösung der mathematischen Gleichungen beinhaltet. Die Forderung der Echtzeitfähigkeit der Simulation ist eine harte Anforderung an die numerischen Verfahren, weswegen spezielle Methoden verwendet werden müssen. Weiterhin hat der Simulationskern die Aufgabe, den Ablauf der Simulation zu steuern (z.B. auftretende Events zu behandeln) sowie die benötigten Simulationsergebnisse zu speichern. Der verwendete Simulationskern wird ebenfalls von Bosch Rexroth entwickelt [17]. Dieser ist in C++
geschrieben und unterstützt unmittelbar die mit Hilfe der Codegenerierung des OpenModelica-Compilers generierten Modelle. Sofern Teile der Codegenerierung geändert werden, besteht somit unmittelbar dieMöglichkeit, die entsprechend notwendigen Änderungen an dem Simulationskern ebenfalls vorzunehmen, was eine vollkommene Flexibilität ermöglicht.

Compiler für Zielhardware

Um den Code auf der Zielhardware ausführen zu können, muss dieser mit Hilfe eines Compilers für das Betriebssystem der Zielhardware kompiliert werden. Auf den Bosch Rexroth Industriesteuerungen läuft das Echtzeit-Betriebssystem *VxWorks*. Für diese wird ein spezieller VxWorks-Compiler der Firma *Windriver* verwendet.

Engineering-Umgebung

Der letzte notwendige Schritt ist die Integration des kompilierten, ausführbaren Codes in ein Projekt, welches auf
die Steuerung geladen werden kann. Wünschenswert ist es, dass der Inbetriebnahme-Ingenieur weiterhin die gewohnte Entwicklungsumgebung verwenden kann und die Integration des Codes in das Steuerungsprojekt weitestgehend automatisiert erfolgt. Für Industriesteuerung der Bosch Rexroth AG besteht die Möglichkeit, in der Entwicklungsumgebung der Steuerung (*Rexroth IndraWorks*) wie gewohnt ein Projekt anzulegen, welches aus unterschiedlichen Funktionen und Funktionsbausteinen besteht. Statt die Implementierung des Funktionsbausteins in SPS-Programmiersprachen nach IEC

61131 vorzunehmen, besteht unter anderem die Möglichkeit, eine externe Implementierung zu dem Baustein hinzuzulinken. An dieser Stelle wird das *OpenCore Interface* [11] verwendet. Dies ist eine hauseigene Schnittstelle (verfügbar in verschiedenen Hochsprachen wie C/C++, C#, LabView, Matlab, Modelica), um von außen auf die Steuerung zugreifen zu können. Diese beinhaltet unter anderem Funktionen zum Lesen oder Schreiben von Steuerungsparametern und Variablen, Starten oder Stoppen von Applikationen, Triggern von Tasks oder Ausführen von Motion-Befehlen. Weiterhin bietet das OpenCore Interface die zuvor erwähnte Funktionalität, extern implementierte Funktionen zu Funktionsbausteien hinzuzulinken.

3.1.3 Gesamtstruktur der Toolchain

Die zuvor dargestellten Komponenten werden schließlich miteinander verbunden, sodass eine durchgängige Toolchain vom Modelica-Modell hin zu dem Projekt entsteht, welches auf der jeweiligen Steuerung ausgeführt wird. Die Gesamtstruktur ist in **Bild 9** dargestellt.

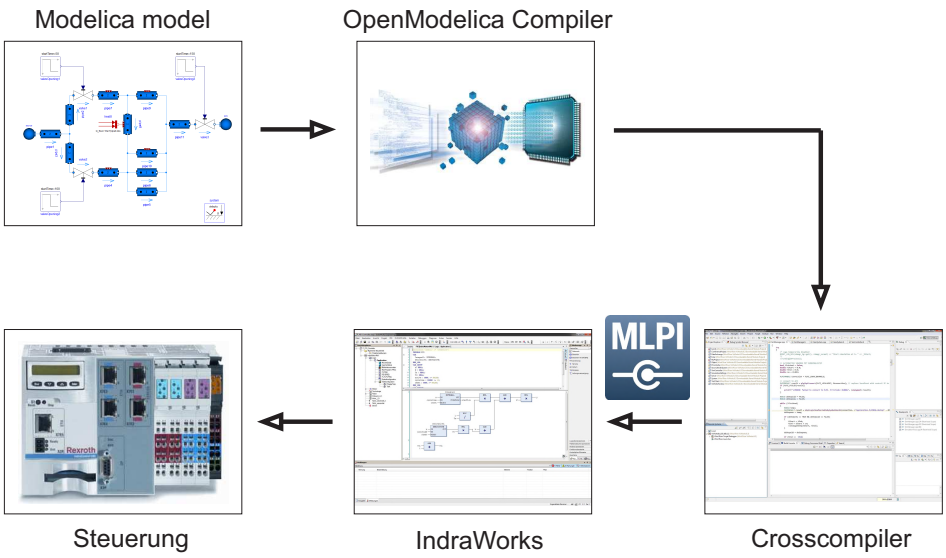

Bild 9 Struktur der Toolchain

Die Abarbeitung der aufeinanderfolgenden Schritte lässt sich vollständig automatisieren, sodass der Anwender lediglich das Modelica-Modell aufbaut und dieses Modelica-Modell einem Skript übergibt. Als Ergebnis kann unmittelbar die Bibliothek erhalten werden, die der Anwender durch Anhängen an einen Funktionsbaustein in sein SPSProjekt einbinden kann. Auf diese Weise ist es möglich, ein Steuerungsprojekt wie gewohnt zu erstellen, wobei einzelne Funktionsbausteine eine Implementierung in SPSSprachen und einzelne Funktionsbausteine in C/C++-Code beinhalten können.

3.2 Anwendungsfall Rapid Control Prototyping

Die in **Kapitel 3.1** vorgestellte Toolchain soll im Folgenden verwendet werden, um einen in Modelica entwickelten Regler auf einer Industriesteuerung auszuführen. Als Beispiel dient die Auslegung eines hydro-mechanischen Systems, konkret die Auslegung einer Presse. Die Struktur des Modelica-Modells ist in **Bild 10** dargestellt. Das Modell besteht aus einer Druckversorgung (innerhalb der Komponente HPU), einem Ventil sowie einem Differentialzylinder. Für die Regelung der Anlage ist ein Soll-Geschwindigkeitsprofil für den Zylinderkolben vorgegeben. Das daraus abgeleitete Positionsprofil ist in **Bild 11** in Form der blauen Kurve gezeigt.

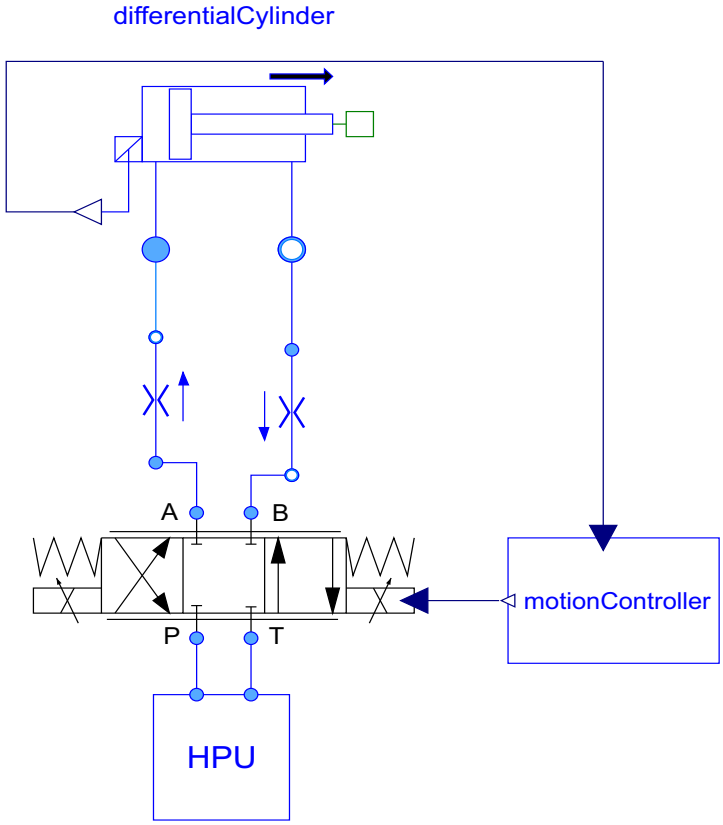

Bild 10 Modell der Regelstrecke

Um die Dynamik des Systems untersuchen zu können, befindet sich innerhalb des *MotionController*-Blockes der Regler. Dabei handelt es sich um einen Positionsregler mit einer Geschwindigkeitsvorsteuerung. Aus diesem Regler wird mit Hilfe der vorgestellten Toolchain Code zur Ausführung auf der Steuerung generiert. Nachdem der OpenModelica-Compiler aus dem Modelica-Modell C++-Code generiert hat und dieser, gemeinsam mit dem Simulationskern, mit Hilfe des Windriver VxWorks Compilers kompiliert wurde, befindet sich der gesamte kompilierte Code in einer Bibliothek. Diese Bibliothek wird auf den internen

Speicher der Industriesteuerung verschoben und wie gewohnt ein SPS-Projekt innerhalb von IndraWorks angelegt. Für den Regler wird innerhalb des Projektes ein neuer Funktionsbaustein angelegt, welcher einen Eingang und einen Ausgang besitzt. Die Anzahl der Ein- und Ausgänge entspricht dabei genau der Anzahl im Modelica-Modell (vgl. **Bild 10**). Innerhalb IndraWorks muss in den Funktionsbaustein-Eigenschaften noch die Option *externe Implementierung* ausgewählt werden. Die Zuordnung zwischen der Bibliothek und dem Funktionsbaustein erfolgt mit Hilfe von OpenCore Engineering-Befehlen. Der benötigte Code wird bereits von der Codegenerierung mit erzeugt und befindet sich innerhalb der Bibliothek. Die Ausführung des Codes erfolgt automatisch beim Starten der Steuerung.

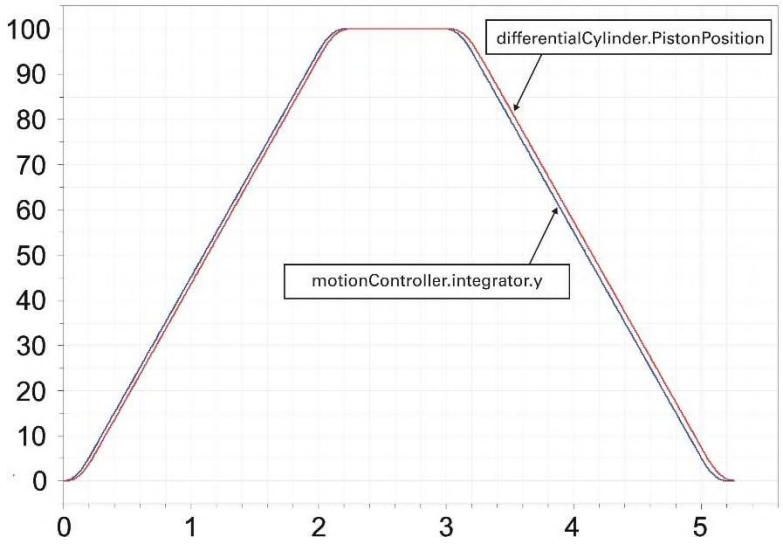

Bild 11 Sollposition (blau) sowie Istposition (rot) des Zylinderkolbens nach der HiL-Simulation

Virtuelle Inbetriebnahme Für die VIBN wird im Simulationsmodell der Block *motionController* durch Elemente der bereits oben vorgestellen Bibliothek mlpi4Modelica ersetzt. Das Simulationsergebnis der Zylinderkolbenposition ist als rote Kurve in **Bild 11** dargestellt. Zwischen der Sollkurve und der Istkurve ist bereits eine gute Übereinstimmung zu erkennen. Für den Betrieb an der finalen Anlage sind die Reglerparameter aufgrund des unterschiedlichen Verhaltens von realer Anlage und Simulationsmodell in der Regel noch in geringem Rahmen nachzujustieren. Insgesamt ergibt sich durch die Verwendung von modellbasierten Entwicklungsmethoden eine deutliche Zeitreduzierung, da der Regler rein virtuell ausgelegt wird und virtuell in Betrieb genommen wird. Es muss für die Inbetriebnahme des Reglers folglich nicht gewartet werden, bis die Anlage verfügbar ist. An der realen Anlage ist lediglich die Nachjustierung vorzunehmen.

4 Zusammenfassung und Ausblick

Im vorliegenden Beitrag wurde ein Überblick über die Möglichkeit zum Einsatz von modellbasierten Engineeringmethoden mit Industriesteuerungen der Bosch Rexroth AG gegeben. Die virtuelle Inbetriebnahme wurde unter dem Aspekt der neuen Herausforderungen, die sich aus Industrie 4.0 ergeben, betrachtet. Darüber hinaus wurde eine Hardware-In-The-Loop-Methode für Rexroth Industriesteuerungen vorgestellt, welche die übliche HiL-Simulation, in Bezug auf die gefundenen Anforderungen an die virtuelle Inbetriebnahme, erweitert. An einem einfachen Beispiel, der Kopplung einer Rexroth-Industriesteuerung mit einem Delta-Robotermodell, wurde diese neue Technik angewendet. Eine häufig zur modellbasierten Entwicklung verwendete Toolchain zur Codegenerierung basiert auf der Verwendung von MATLAB/Simulink. Aufgrund einiger Nachteile, die mit der Verwendung dieser Toolchain einhergehen, wird in diesem Beitrag eine alternative Lösung vorgestellt. Diese verwendet offene Standards (Modelica als offener Sprachstandard zur Beschreibung physikalischer Systeme, OpenSource-Modelica-Compiler) und erlaubt das automatisierte Ausführen von aus Simulationsmodellen generiertem Code direkt auf Bosch Rexroth Steuerungen. Die Methoden der modellbasierten Entwicklung lassen sich auf unterschiedliche Einsatzgebiete anwenden, darunter Rapid Control Prototyping, Model Predictive Control oder die Fehlererkennung im Rahmen der Diagnose. Die Toolchain wird in diesem Beitrag auf einen Anwendungsfall (hydro-mechanische Presse) angewendet, um die Methodik des Rapid Control Prototypings zur modellbasierten Reglerentwicklung anzuwenden. Die Validierung des Reglers erfolgt in diesem Beitrag über eine Hardware-In-The-Loop-Kopplung an einer virtuellen Regelstrecke der Presse, die ebenfalls in Modelica vorhanden ist. In diesem Beitrag wurden zwei Anwendungsfälle des modellbasierten Engineerings imKontext der Applikation vorgestellt. Weitere Anwendungsfelder finden sich beispielsweise im Betrieb der Anlage durch Methoden wie z.B. modellbasierte Diagnose. Auch für die hier vorgestellten Methoden ergeben sich weitere Themen. Die Erstellung der Anlagenmodelle wurde in dieser Arbeit nicht thematisiert, stellt aber einen wesentlichen Aspekt der virtuellen Inbetriebnahme dar. Hierbei spielt speziell die automatisierte Generierung von Modellen aus CADDaten einen interessanten Punkt dar. Da die Validierung des Reglers bisher nur an virtuellen Regelstrecken erfolgt ist, besteht eine Hauptbestrebung momentan darin, den generierten Regler an einer realen Anlage zur Regelung einzusetzen. Ein weiteres Thema für die Zukunft ist die Umsetzung von *Model Predictive Control* für den Einsatz auf Bosch Rexroth Steuerungen. Dazu ist der Simulationskern zusätzlich um die mathematischen Methoden zur Optimierung des Systemeingangs zu erweitern. Diese müssen so effizient implementiert werden, dass die Durchführung der Optimierung innerhalb der vorgegebenen Rechenzeit (Zykluszeit) abgeschlossen ist.

5 Literatur

[1] DIN 19246 1991-6: *Messen, Steuern, Regeln; Abwicklung von Projekten; Begriffe*. Berlin, Beuth Verlag, 1991.

[2] o. Autor: *VDW: Abteilungsübergreifende Projektierung komplexer Maschinen und Anlagen*. Aachen, WZL, 1997.

[3] o. Autor: *Umsetzungsempfehlung für das Zukunftsprojekt Industrie 4.0*. o. Ort, BMBF, 2013.

[4] Dumitrescu, R.: *Systems Engineering für Industrie 4.0*. Mannheim, Vortragsfolien 3DEXPERIENCE Forum, 2014.

[5] Wünsch, G.: *Methoden für die virtuelle Inbetriebnahme automatisierter Produktionssysteme*. München,
Herbert Utz Verlag, 2008.

[6] Mikelsons, L. et al.: *Symbolic Model Reduction Applied to Realtime Simulation of a Construction Machine*. In: Proceedings of the 7th Modelica Conference, September 20-22, 2009, Como, Italy.

[7] Hofmann, A. et al.: *Simulating Collisions within the Modelica MultiBody library*. In: Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden.

[8] Homepage der Modelica Association: www.modelica.org

[9] Fritzson, P.: *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley, 2014.

[10] Homepage zum OpenCore Interface:
http://www.boschrexroth.com/de/de/produkte/engineering/open-core-engineering/die-features-vonopen-core-engineering/open-core-interface/opencore-interface-1 (07. Januar 2015).

[11] Engels, E., Gabler, T.: *Universelle Programmierschnittstelle für Motion-Logic Systeme*. In: Struktur, Funktionen und Anwendung in Forschung und Lehre, Tagungsband AALE 2012.

[12] Homepage des OpenCore Engineering Networks https://www.boschrexroth.com/network (07. Januar2015).

[13] Homepage zu Ogre3d: http://www.ogre3d.org (08. Januar 2015).

[14] Martini, A.: *Delta-Roboter - Aufbau, Arbeitsweise und Anwendung*. o. Ort, o. Jahr, In: Technologien der Fertigungsautomatisierung - Seminarreihe zu ausgewählten Forschungsthemen der industriellen Anwendung.

https://wiki.zimt.unisiegen.de/fertigungsautomatisierung/index.php/Delta-Roboter (07. Januar 2015).
[15] Multi-Engineering Modeling and Simulation - Dymola: www.dynasim.se (08. Januar 2015).
[16] Fritzson, P. et al.: *The open source Modelica project*. In: Proceedings of the 2nd International Modelica Conference, March 18-19, 2002, Oberpfaffenhofen, Germany.
[17] Worschech, N.; Mikelsons, L.: *A Toolchain for Real-Time Simulation using the OpenModelica Compiler*.
In: Proceedings of the 9th International Modelica Conference, September 3-5, 2012,Munich, Germany.