Entwurf und Simulation mechatronischer Echtzeitsysteme

Wie Mechanik, Elektronik und Software schnell unter einen Hut kommt

Hendrik Reddehase, Dr. Robert Wagner
Solunar GmbH

Neue Produkte müssen immer ausgefeilter und intelligenter werden. Sie müssen miteinander kommunizieren und kooperieren, um sich flexibel an geänderte Situationen anpassen zu können. Um möglichst kurze Time-To-Market Zeiten zu erzielen, muss im Bereich der mechatronischen Systeme zudem die Software sehr häufig parallel zur Hardware entwickelt werden, so dass das Zusammenspiel erst sehr spät überprüft werden kann. Mit dem hier vorgestellten Ansatz können die Software und die Hardware zunächst modelliert und anschließend zusammen simuliert werden. Dadurch wird eine Überprüfung des Zusammenspiels von Mechanik, Elektronik und Software bereits in einer sehr frühen Entwicklungsphase möglich.

Einleitung

Ein vielversprechender Ansatz für den disziplinübergreifenden Entwurf von Systemen stellt die modellgetriebene Systementwicklung dar, mit derer Hilfe die Zusammenarbeit der verschiedenen Disziplinen (Maschinenbau, Elektrotechnik, Regelungstechnik und Informatik) durch innovative Prozesse, Methoden und Werkzeuge vereinfacht wird. Hierbei werden die zu entwickelnden Systeme mit Modellen beschrieben. Diese Modelle können anschließend für eine gemeinsame Simulation verwendet werden.

Ein Problem stellt die Simulation diskreter Softwaremodelle für Echtzeitsysteme dar, die von gängigen Simulationswerkzeugen bisher kaum berücksichtigt wird. Zur Lösung dieses Problems wird in diesem Beitrag ein Ansatz vorgestellt, mit dem eine fachgebietsübergreifende Simulation eines zu entwickelnden mechatronischen Systems realisiert wurde. Der Ansatz wird anhand eines (vereinfachten) Beispiels von autonom fahrenden Miniaturrobotern erläutert. Ein solcher Miniaturroboter ist in Abbildung 1 dargestellt.

Abbildung 1: Miniaturroboter

Als Miniaturroboter wird ein so genannter BeBot genutzt. Hierbei handelt es sich um eine am Heinz Nixdorf Institut entwickelte Testplattform [GSD+11]. Der BeBot besitzt verschiedene Aktoren und Sensoren, um mit der Umgebung sowie anderen Miniaturrobotern zu interagieren. In diesem Beitrag liegt der Fokus auf dem Softwareentwurf, der gemeinsam mit einem Modell der Hardware simuliert wird.

Konzipierung und Entwurf

Um mechatronische Echtzeitsysteme erfolgreich, sicher und schnell entwickeln zu können bedarf es einer Methodik, mit der Systeme fachgebietsübergreifend konzipiert, modelliert und anschließend ganzheitlich simuliert werden können. Für die fachgebietsübergreifende Konzipierung und den Entwurf wird die Sprache „CONSENS“ (CONceptual design Specification technique for the ENgineering of complex Systems) verwendet [GFD+09]. Für fachgebietsspezifische Anteile werden die für die jeweilige Domäne und Aufgabe geeigneten Sprachen verwendet.

Die fachgebietsübergreifende Spezifikation besteht aus sieben kohärenten Partialmodellen. Die Partialmodelle beschreiben das Umfeld, die Anwendungsszenarien, die Anforderungen, die Funktionen, die Wirkstruktur, das Verhalten sowie die Gestalt des zu entwickelnden mechatronischen Systems. Zusammen bilden die Modelle die sogenannte Prinziplösung.

In Abbildung 2 ist ein Ausschnitt aus der Wirkstruktur des Miniaturroboters dargestellt. Die Wirkstruktur beschreibt die prinzipielle Wirkungsweise des zu entwickelnden mechatronischen Systems. Hierin finden sich die beteiligten Fachdisziplinen wieder. Systemelemente werden durch Sechsecke dargestellt und repräsentieren Systeme, Module, Bauteile, und Softwarekomponenten. Die Struktur eines größeren Systems ist in aller Regel eine Hierarchie von Systemelementen. Wechselwirkungen zwischen den Systemelementen werden als Verbindungslinien dargestellt und können Stoff-, Energie und Informationsflüsse sein.

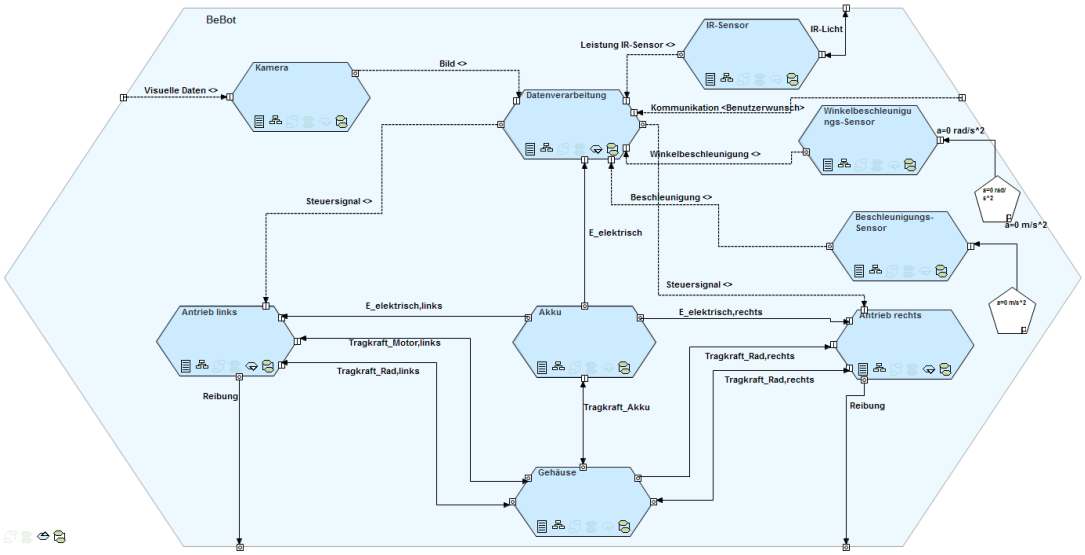

Abbildung 2: Ausschnitt aus der Wirkstruktur eines Miniaturroboters

Neben der Struktur bildet das Verhalten eine wichtige Rolle. Dieses wird immer häufiger durch Software umgesetzt. Die Software sollte allerdings nicht losgelöst von den regelungstechnischen und mechanischen Eigenschaften betrachtet werden, da es sonst später zu Problemen kommen könnte, die nur mit sehr hohem Aufwand behoben werden können. In der frühen Phase der Konzipierung wird die Software daher bereits durch einfache Zustandsdiagramme beschrieben.

Für einen detaillierten Softwareentwurf hingegen wird die Wirkstruktur und die dazugehörigen Zustandsdiagramme in die MechatronicUML überführt [BBB+12]. Die MechatronicUML stellt eine Erweiterung und Verfeinerung der UML dar. Sie verfügt über eine formale Semantik und kann daher formal verifiziert und simuliert werden [PSR+12].

Simulation

Zur Modellierung der mechanischen, regelungstechnischen und softwaretechnischen Aspekte werden verschiedene Werkzeuge eingesetzt. Die Verwendung unterschiedlicher Werkzeuge führt bei der gemeinsamen Simulation der Modelle zu Problemen, da die Modelle nur schwer miteinander integriert werden können. Einen möglichen Ausweg aus diesem Dilemma bietet das Functional Mock-Up Interface (FMI) [MOD12]. Auf dieser Grundlage können Modelle zwischen den Werkzeugen ausgetauscht und simuliert werden, d.h. auf diese Art und Weise ist es möglich, in unterschiedlichen Werkzeugen erstellte Modelle in einem einzigen Werkzeug zu integrieren und gemeinsam zu simulieren.

Hierzu muss eine sogenannte Functional Mock-up Unit (FMU) erstellt werden. Eine FMU nutzt die im FMI-Standard definierten Schnittstellen, um ein Verhaltensmodell zu realisieren. Physikalisch ist eine FMU ein komprimiertes ZIP-Archiv, in dem neben der Implementierung alle zur Simulation benötigten Informationen in einer XML-Datei abgelegt werden. So enthält die XML-Datei beispielsweise eine Liste aller Variablen, deren Werte während der Simulation zwischen dem Simulator und der FMU ausgetauscht und aktualisiert werden. Der FMI-Standard definiert darüber hinaus Funktionen, die für die Interaktion zwischen Modell und Simulator benötigt werden. Diese Funktionen müssen innerhalb einer FMU in der Programmiersprache C implementiert und zu einer Dynamic Link Library (DLL) übersetzt werden.

Eine manuelle Implementierung und Überführung von Softwaremodellen in eine FMU zu Zwecken der Simulation ist allein aus zeitlichen Gründen einem Anwender bzw. Produktentwickler nicht zumutbar. Daher wurde ein Code-Generator entwickelt, der die Modelle der MechatronicUML automatisch in die Programmiersprache C überführt. Abbildung 3 zeigt einen Überblick der Integration von Software- und Hardwaremodellen.

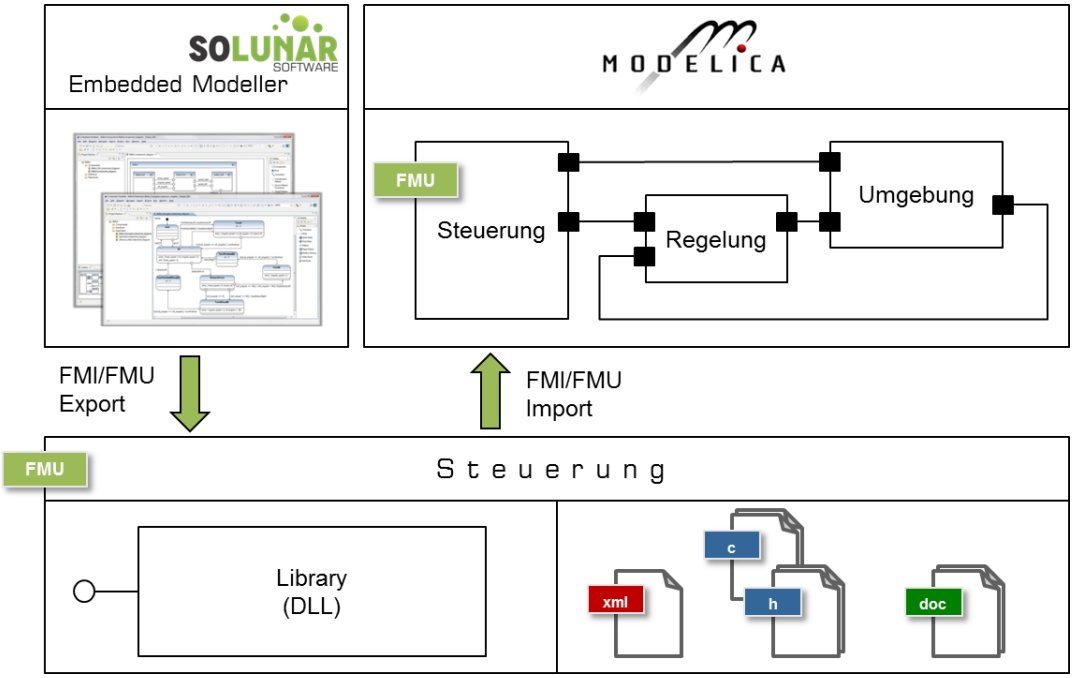

Abbildung 3: Überblick zur Integration von Software- und Hardwaremodellen zur Simulation am Beispiel der Simulationswerkzeuge Modelica/Dymola

Die Softwaremodelle werden im Software-Werkezeug ‚Embedded Modeller' mit Hilfe von Komponenten- und Zustandsdiagrammen der MechatronicUML konkretisiert. Ein Komponentendiagramm bildet dabei eine Hierarchie von zusammengesetzten Komponenten, wobei Komponenten auf der untersteten Hierarchiestufe ein Echtzeit-Zustandsdiagramm besitzen können. Die Kommunikation zwischen Komponenten erfolgt über das Versenden und Empfangen diskreter Nachrichten. Hierzu sind die Komponenten über Kommunikationskanäle miteinander verbunden. Darüber hinaus können Komponenten über kontinuierliche Ein- und Ausgänge verfügen, über die Signale von Sensoren empfangen und an Aktuatoren gesendet werden können. Die Ein- und Ausgänge für kontinuierliche Signale werden zur Simulation mit dem Regler- und Umgebungsmodell verbunden. Anschließend kann das Softwaremodell zusammen mit den Hardwaremodellen simuliert werden. In dem hier vorgestellten Projekt wurde zur Simulation das Werkzeug Modelica/Dymola verwendet.

Fazit

In diesem Beitrag wurde eine Methode vorgestellt, mit der eine Prinziplösung erarbeitet, konkretisiert und anschließend simuliert und überprüft werden kann, noch bevor physikalischer Prototypen erstellt werden. Hierbei wurde anhand der domänenspezifischen Modellierungssprache MechatronicUML gezeigt, wie diskrete Softwaremodelle zusammen mit Hardwaremodellen integriert und simuliert werden können, ohne dass physikalische Prototypen realisiert werden müssen. Der in diesem Beitrag dargestellte Ansatz ist nicht auf die MechatronicUML beschränkt, d.h. es lässt sich auf andere Modellierungssprachen, denen eine formalen Semantik zugrunde liegt, wie z.B. Petri-Netze, übertragen.

Danksagung

Diese Arbeit ist im Rahmen des Verbundprojekts „ENTIME: Entwurfstechnik Intelligente Mechatronik“ entstanden. Das Projekt ENTIME wird vom Land NRW sowie der EUROPÄISCHEN UNION, Europäischer Fonds für regionale Entwicklung, „Investition in unsere Zukunft“ gefördert.

Literatur und Quellenverzeichnis

[BBB+12] Becker, S.; Brenner, C.; Brink, C.; Dziwok, S.; Loeffler R.; Heinzemann, C.; Pohlmann, U.; Schäfer, W.; Suck, J.; Sudmann, O.: The MechatronicUML Method - Process, Syntax, and Semantics, 2012

[GFD+09] Gausemeier, J.; Frank, U.; Donoth, J.; Kahl, S: Specification technique for the description of self-optimizing mechatronic systems. In: Research in Engineering Design vol. 20, Springer, London 2009, Nr. 4, S. 201–223.

[GSD+11] Gausemeier J.; Schierbaum T.; Dumitrescu R.; Herbrechtsmeier S.; Jungmann A.: Miniature robot bebot: Mechatronic test platform for self-x properties. In Proc. Of the 9th IEEE International Conference on Industrial Informatics (INDIN 2011), S. 451-456, 2011

[MOD12] MODELISAR Consortium. Functional mock-up interface for model exchange. Version 1.0, 2010. www.functional-mockup-interface.org

[PSR+12] Pohlmann, U.; Schäfer, W.; Reddehase, H.; Röckemann, J.; Wagner, R: Generating Functional Mockup Units from Software Specifications. In: Proc. of the 9th International Modelica Conference 2012, 2012

Autoren

Dr. Robert Wagner ist Geschäftsführer der Solunar GmbH und seit vielen Jahren in der modellgetriebenen Softwareentwicklung tätig. Hier beschäftigt er sich mit der Modellierung und Simulation eingebetteter Systeme sowie mit der Entwicklung innovativer Techniken zur Modellsynchronisation und Codegenerierung.

Hendrik Reddehase arbeitet bei der Solunar GmbH als Software-Entwickler. Sein Schwerpunkt liegt auf der modellgetriebenen Softwareentwicklung auf Basis von Eclipse, wo er grafische Editoren, Codegeneratoren und Simulationswerkzeuge für eingebettete mechatronische Systeme entwickelt.

Kontakt

Internet: www.solunar.de

Email: info@solunar.de