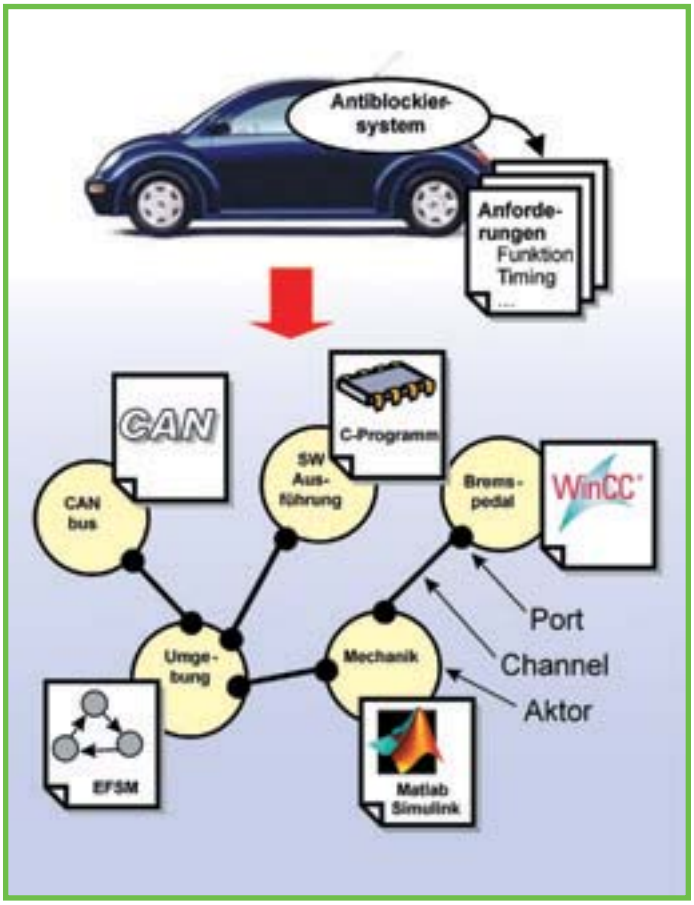
Simulation und Realität flexibel verbinden

EIN SIMULATOR MIT ANPASSBAREN SCHNITTSTELLEN, ECHTZEITFÄHIGKEIT UND DER MÖGLICHKEIT, SICHERHEITSKRITISCHE SYSTEME ZU MODELLIEREN

Die Entwicklung technischer Systeme anhand realer Prototypen ist heute kaum noch finanzierbar. Aus diesem Grund findet ein Großteil der Arbeit auf dem Weg zum einsatzfähigen Produkt am Rechner statt, wobei die Simulation ein wesentliches Werkzeug ist. Dieses Werkzeug muss leistungsfähig, flexibel und anpassbar sein sowie verlässliche Ergebnisse liefern. Derzeit verfolgt das Institut für Technische Informatik – Rechnerstrukturen und Betriebssysteme (IRB) das Ziel, Simulation und Realität nahtlos miteinander zu verknüpfen, was zur Verbesserung und Beschleunigung des Entwicklungsprozesses beitragen wird.



Die Simulation ist ein unentbehrliches Werkzeug bei der Entwicklung vieler technischer Systeme geworden. Die Einsatzmöglichkeiten erstrecken sich über alle ingenieurwissenschaftlichen Gebiete.

Dabei sind im Laufe der Zeit hochspezialisierte Simulationswerkzeuge entstanden, die exakt auf einen bestimmten Problembereich zugeschnitten sind. Mit steigendem Umfang und wachsender Komplexität der zu simulierenden Systeme stellt sich nun immer öfter das Problem, dass unterschiedliche Simulationswerkzeuge zur Abbildung des Gesamtsystems gekoppelt werden müssen.

Am Institut für Technische Informatik – Rechnerstrukturen und Betriebssysteme wurde im Rahmen mehrerer internationaler Förderprojekte und Industriekooperationen der Simulator ClearSim-Multi-Domain entwickelt.

Das primäre Einsatzgebiet dieses Simulators ist derzeit die Simulation von eingebetteten Systemen, bestehend aus Mikrocontroller(n), zugehöriger Software und der nötigen Peripherie.

Im Rahmen dieser Arbeiten hat es sich als nützlich erwiesen, die Modelle in einzelne Module, so genannte Aktoren, zu zerlegen.

Aktoren sind generische Aktivitätsträger, die für den Datenaustausch untereinander Schnittstellen (Ports) aufweisen. Ports werden durch Channels verbunden.

Dieses Konzept ist äußerst flexibel und erlaubt die Beschreibung und Kopplung von Subsystemen unterschiedlichster Domänen. Es ist auch nutzbar für die Kopplung unterschiedlicher Simulatoren. Weiterhin wird damit die Basis für eine Verbindung zwischen Simulator und realen Komponenten gelegt. Dadurch steht ein sehr leistungsfähiges Simulationstool zur Verfügung, das ständig weiterentwickelt wird, um es an neue Aufgabenstellungen anzupassen und um weitere Anwendungsgebiete für ClearSim bzw. die Simulation allgemein zu erschließen.

Aktuelle Arbeitsfelder

1. Strukturorientierte Modelldefinition

Im Bereich der Softwareentwicklung setzt sich in zunehmendem Maße eine Technik durch, bei der ein Programm mit seinen Funktionen vollständig grafisch objektorientiert entworfen wird. Dabei entstehen Diagramme der Programmstruktur, aus denen dann direkt der ausführbare Programmcode generiert werden kann. Der »Programmierer« beschäftigt sich also ausschließlich mit der übergeordneten Struktur, die Fleißarbeit des Umsetzens dieser Struktur in Codes ist vollkommen automatisiert.

Für diese Aufgabe wird heute zunehmend die Unified Modelling Language (UML) verwendet. UML ist jedoch vom Ansatz her nicht auf die Softwareentwicklung beschränkt. Daher wird an der Möglichkeit gearbeitet, UML zur Beschreibung der Modelle für den Simulator ClearSim zu verwenden.

Dadurch lässt sich der Umgang mit dem Simulator deutlich effektiver gestalten, da die Fehleranfälligkeit sinkt, viele Routineaufgaben automatisiert sind und beispielsweise die Dokumentation eines Projekts weitgehend automatisch aus der Struktur generiert wird.

Für den Bereich des Entwurfs von Software für eingebettete Systeme existiert bereits ein Tool auf der Basis von SDL (Specification and Description Language), das am Institut entwickelt wurde und einen vergleichbaren Ansatz verfolgt.

2. Echtzeitfähigkeit und inkrementeller Entwurf

Ziel einer Simulation ist im Allgemeinen die Entwicklung eines realen Systems.

Häufig muss im Laufe dieser Entwicklung ein großer Schritt vom simulierten (virtuellen) Prototypen in einer ebenfalls simulierten Umgebung zu einem realen Prototyp in realer Umgebung gemacht werden, weil eine Verknüpfung von Simulation und Realität schwer möglich ist. Hier wäre es sehr vorteilhaft, wenn dieser Übergang inkrementell, also schrittweise, erfolgen könnte.

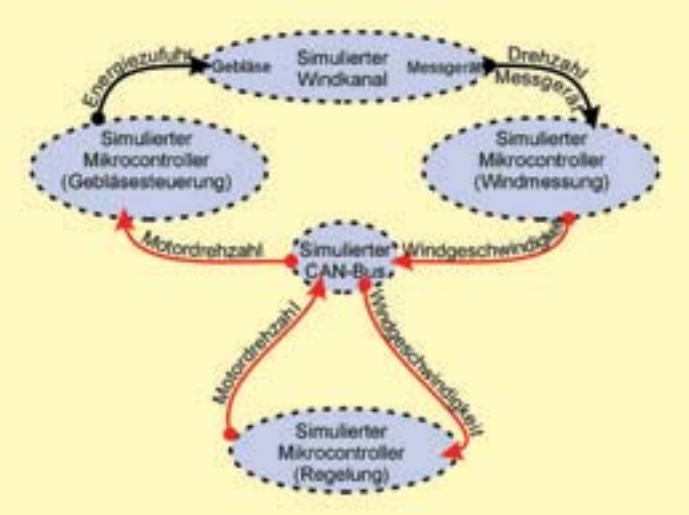
Dies soll am Beispiel des in den Bildern 2 und 3 gezeigten sehr einfachen Systems erläutert werden.

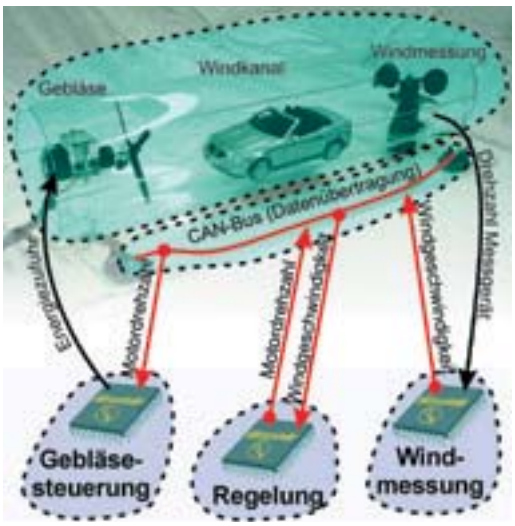
Es besteht aus einem Windkanal, einem Gebläse und einem Messgerät für die Windgeschwindigkeit. Weiterhin sind drei Mikrocontroller enthalten, von denen jeweils einer die Daten des Messgeräts aufbereitet (Windmessung) und das Gebläse ansteuert (Gebläsesteuerung). Der dritte Controller übernimmt die eigentliche Regelung des Gesamtsystems. Diese Mikrocontroller tauschen über einen CAN-Bus (Controller Area Network) Daten untereinander aus. Bild 2 zeigt, wie dieses System für die Simulation in einzelne virtuelle Komponenten aufgeteilt wird.

Wenn nun beispielsweise die Peripherie, bestehend aus Windkanal mit Gebläse und Messgerät, Gebläsesteuerung und Windmessung, durch reale Komponenten ersetzt werden könnte, während der zentrale für die Regelung zuständige Controller in der Simulation verbleibt, ließe sich die Komplexität des Modells reduzieren und gleichzeitig die Genauigkeit verbessern, da Modelle durch reale Teile ersetzt werden. Die Verbindung zwischen realen und virtuellen Komponenten wird durch den CAN-Bus hergestellt, der um eine Schnittstelle zur Simulation erweitert wird.

Besonders interessant ist dies bei der Entwicklung von Software für eingebettete Systeme.

Wenn die Software auf einem simulierten Mikrocontroller läuft, lässt sie sich wesentlich leichter beobachten. Der Wert einer beliebigen Va-





riablen ist zur Laufzeit von außen zugänglich, was die Fehlersuche erleichtert.

Derartige Schnittstellen, die eine Überwachung zur Laufzeit ermöglichen, sind mit einem realen Mikrocontroller nur mit verhältnismäßig großem Aufwand machbar. Außerdem beeinflussen sie das Verhalten des Gesamtsystems, da sie Ressourcen in Form von Speicherplatz und Rechenleistung beanspruchen.

Voraussetzung für diese Verknüpfung von Simulation und Realität ist, dass jede virtuelle Komponente nicht nur hinsichtlich ihrer Funktionalität, sondern auch ihrer Verarbeitungsgeschwindigkeit (Timing) exakt durch die Simulation abgebildet wird.

Im einfachsten Fall sind relative Aussagen über das Timing möglich: »Vorgang A benötigt 3mal so viel Zeit wie Vorgang B«. Wenn die Leistungsfähigkeit des Bauteils von Interesse ist, muss zusätzlich das Verhältnis von simulierter zu realer Zeit bekannt sein: »4 Sekunden in der Simulation entsprechen 1 Sekunde in der Realität«.

Wird eine Verbindung von virtuellen und realen Komponenten benötigt, muss ein simulierter Vorgang auch genauso lange dauern wie ein entsprechender realer Vorgang.

Die Simulation muss in Echtzeit ablaufen, da sie sonst das Zusammenspiel der einzelnen Komponenten verzerren würde.

Bisher sind derartige Simulationen, wenn überhaupt, nahezu ausschließlich unter Verwendung von sehr leistungsfähiger und damit teurer Hardware möglich.

Am IRB laufen derzeit Arbeiten mit dem Ziel, eine Echtzeit-fähige Simulation auf einem preiswerten Standard-PC unter Verwendung des kostenlos erhältlichen Betriebssystems Linux zu realisieren.

Nicht nur die Geschwindigkeit der Simulation, auch der Aufbau des verwendeten Modells muss an die Anforderungen der Echtzeit-Fähigkeit angepasst werden.

Bisherige Simulationsverfahren sind überwiegend Ereignis-basierend. Das heißt, das Gesamtsystem wird in einzelne Knoten oder Module (zum Beispiel Bremspedal, ABS-Regler …) zerlegt, die untereinander mittels Ereignissen wie »Modul Y empfängt Information A von Modul X« kommunizieren. Diese Ereignisse werden in einer Liste verwaltet und der Reihe nach abgearbeitet. Für das Beispiel folgt auf das Ereignis die Bestimmung der Reaktion von Modul Y auf die Daten von Modul X, anschließend wird dann das nächste Ereignis in der Liste abgearbeitet.

Nun wäre es möglich, dass in der Zeit, die ein Modul für die Verarbeitung von Daten benötigt, weitere Daten für das selbe Modul eintreffen. Bei einer Ereignis-basierenden Simulation fällt dies nicht weiter auf, es wird einfach ein weiteres Ereignis in der Liste eingetragen.

Eine zeitgenaue Simulation bietet hingegen die Möglichkeit, diese Kollision von Daten zu erkennen und entsprechend darauf zu reagieren.

Bei dieser Art der Simulation wird der zu simulierende Zeitraum in einzelne kleine (diskrete) Intervalle zerlegt. Dann wird für jedes Teilmodul des Gesamtsystems ermittelt, ob in diesem speziellen Intervall etwas zu tun ist. Dies geschieht, bis alle Module abgearbeitet sind und das nächste Intervall betrachtet werden kann. Dabei kann direkt berücksichtigt werden, dass ein Modul beispielsweise den Zeitraum von drei Intervallen benötigt, um bestimmte Informationen zu verarbeiten, in der Zwischenzeit anfallende Daten also anderweitig zwischengespeichert werden müssen.

3. Sicherheit und Simulationsqualität

Die immer weiter voranschreitende Automatisierung führt dazu, dass technische Systeme immer komplexer werden. Leistungsfähige Mikrocontroller bieten vielfältige Einsatzmöglichkeiten.

Dies führt aber auch dazu, dass es eines steigenden Aufwandes bedarf, um die Fehlerfreiheit eines in der Entwicklung befindlichen Systems aus Hardware (Mikrocontroller) und zugehöriger Software sicherzustellen. Denn durch die zunehmende Anzahl von Funktionen ergeben sich auch mehr Möglichkeiten der gegenseitigen Störung der einzelnen Systeme.

Da diese Systeme aber auch sicherheitskritische Aufgaben übernehmen (beispielsweise im Bereich des Bremssystems eines Autos: ABS), müssen Möglichkeiten geschaffen werden, bereits während der Entwicklung zu gewährleisten, dass alle Parameter des Gesamtsystems unter allen denkbaren Betriebszuständen innerhalb definierter unkritischer Wertebereiche verbleiben.

Der Ansatz, der dazu am IRB verfolgt wird, besteht aus der Vorgabe sogenannter Assertions (Feststellungen) für beliebige Parameter des Systems.

Eine Assertion beinhaltet eine Einschränkung des zulässigen Wertebereichs einer Variablen oder sonstigen Größe, sowie eine Festlegung, wie zu reagieren ist, wenn dieser Gültigkeitsbereich verlassen wird (zum Beispiel Korrektur auf den nächstliegenden zulässigen Wert, Zurücksetzen des Gesamtsystems, Abschalten des Systems).

Die beobachteten Parameter können dabei ganz einfach Skalare sein, die beispielsweise eine elektrische Spannung, eine Drehzahl oder eine Temperatur repräsentieren. Oder aber, und das ist in dieser Art neu, auch Bedingungen für das Zeitverhalten.

So ist es möglich, die maximale und/oder minimale Dauer eines Vorgangs festzulegen.

Hier kann in hervorragender Weise von den exakten Informationen zum Timing einer Simulation, die der Simulator ClearSim-MultiDomain liefert, profitiert werden.

Diese Assertions können dabei entweder nur für die ursprüngliche Testphase eingefügt werden und dem Debugging dienen, oder aber auch in der Software für das fertige System verbleiben und dort die Basis für Überwachungs- und Diagnosesysteme bilden.

Während der Simulation können Parameter für alle simulierten Komponenten überwacht werden, im fertigen System selbstverständlich nur Werte, die der Software eines Controllers über Variablen zugänglich sind.

Um das Einfügen der Assertions in die Modelle für ClearSim zu vereinfachen, sollen diese ebenfalls unter Verwendung von UML definiert werden können. Dadurch ist eine nahtlose Integration in die zukünftige rein objektorientierte/strukturorientierte Modellgenerierung von ClearSim möglich.

Das Vorgehen für das Verwenden von Assertions in mittels UML generierten Modellen ist in Bild 4 dargestellt.

Bereits im ersten Schritt auf der höchsten Abstraktionsebene wird die Modellstruktur mit Assertions instrumentiert. Diese werden vor der Erzeugung des virtuellen Prototyps aus der Modellbeschreibung zusammen mit allen notwendigen Informationen zum Kontext extrahiert und in entsprechenden C-Code umgewandelt. Mit diesem Code wird dann der automatisch generierte virtuelle Prototyp (ebenfalls C-Code) instrumentiert.

Mit »Instrumentieren« ist hierbei gemeint, dass die automatisch generierten Funktionen des Prototyps nicht verändert, sondern die Assertions lediglich in den Programmablauf integriert werden, also eine zusätzliche Überprüfung der an eine Funktion übergebenen Parameter erfolgt.

Treten bei der anschließenden Simulation des Prototypen Fehler auf, werden entsprechende Veränderungen an den Definitionen vorgenommen. Treten keine Fehler auf und werden alle Assertions eingehalten, kann mit der Entwicklung fortgefahren werden.

Assertions stellen ein mächtiges Werkzeug für die Gewährleistung einer hohen Simulationsqualität dar.

Durch die Integration in den UML-basierten Modellbildungsprozess wird ihre Handhabung erheblich vereinfacht.

Prof. Dr.-Ing.
Christian Müller-Schloer
Jahrgang 1950, Leiter des Instituts für Technische Informatik – Rechnerstrukturen und Betriebssysteme (IRB) seit der Gründung des Instituts im Jahre 1991