

Nils Paluch / Achim Schönhoff / Horst Salzwedel

Anwendung von Co-Design für verteilte Echtzeitsysteme

EINLEITUNG

Produkte der Luft- und Raumfahrt enthalten seit einigen Jahrzehnten verteilte und eingebettete Systeme. Zahlreiche Realisierungen dieser eingebetteten Systeme beinhalten kooperierende Hardware- und Softwarekomponenten, die in Großprojekten zum Einsatz kommen. Diese Projekte leiden unter der immer weiter steigenden Komplexität, da hier eine Vielzahl komplexer Einzelsysteme in hochgradiger Vernetzung interagiert. Dennoch gibt es bis heute kein einziges etabliertes Werkzeug, welches den kompletten Entwicklungspfad neuer Systeme zentral verwaltet und aktiv unterstützt.

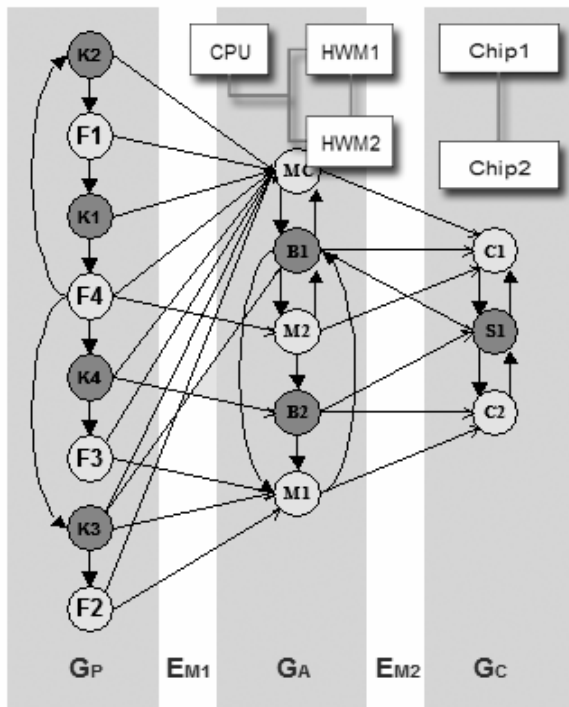
Der Markt wird überschwemmt mit einer Reihe unterschiedlicher Werkzeuge für Spezifikation, Softwareentwicklung, Hardwaredesign und viele andere Bereiche. Damit bleibt es in aller Regel dem Unternehmen überlassen, eine Auswahl aktuell am Markt befindlicher Werkzeuge zu finden und ihren Einsatz sinnvoll zu koordinieren. Dabei entsteht immer eine Vielzahl an Redundanzen und asynchroner Datenbestände.

Co-Design ist ein Schlagwort, das Besserung verspricht. HW/SW-Co-Design bezeichnet ein kombiniertes Design von Hard- und Softwarekomponenten und beinhaltet neben der Spezifikation und der Systemmodellierung auch die Abbildung von Soft- auf Hardware. Mit Hilfe sog. Co-Simulationen ist es dann möglich, ein System zu testen, noch bevor es gebaut wird. In der Praxis ist das Hauptproblem der Co-Simulation fast immer die Hardware, für die sich nur schwer realistische Informationen bzw. Modelle finden lassen, wie sie für eine Simulation benötigt werden. Das liegt vor allem daran, daß die Hardwarehersteller natürlich daran interessiert sind, die Leistungsfähigkeit ihrer Produkte möglichst hoch einzustufen. Auch gibt es bis heute keinen Standard für das HW/SW-Co-Design.

Innerhalb einer Diplomarbeit, die bei der EADS in Ottobrunn entstand, wurde ein Co-Design auf Basis des Modellierungs- und Simulationswerkzeugs MLDesigner der Firma Mission Level Design GmbH zur Modellierung eines Tiefflugsystems des Transportflugzeugs A400M untersucht [1]. Die während der Arbeit entstandenen Modelle und Strategien werden in Auszügen vorgestellt. Die Modellierung wurde auf Basis des Simulationswerkzeugs MLDesigner angefertigt und getestet.

ABBILDUNGSSTRATEGIEN

Die Definition eines Problemgraphen (G_p), eines Architektur- und Chipgraphen (G_A / G_C) stellt eine vollständige Allokation dar. Durch eine Schätzung kann eine Menge möglicher Architekturkomponenten ausgewählt werden, welche die Anforderungen der Ablaufplanung erfüllen könnten. Das können sehr viel mehr und auch redundante Komponenten sein, denn erst durch die Bindung werden sie tatsächlich ausgewählt. Damit eine Bindung durchgeführt werden kann, muß klargestellt werden, auf welche Module



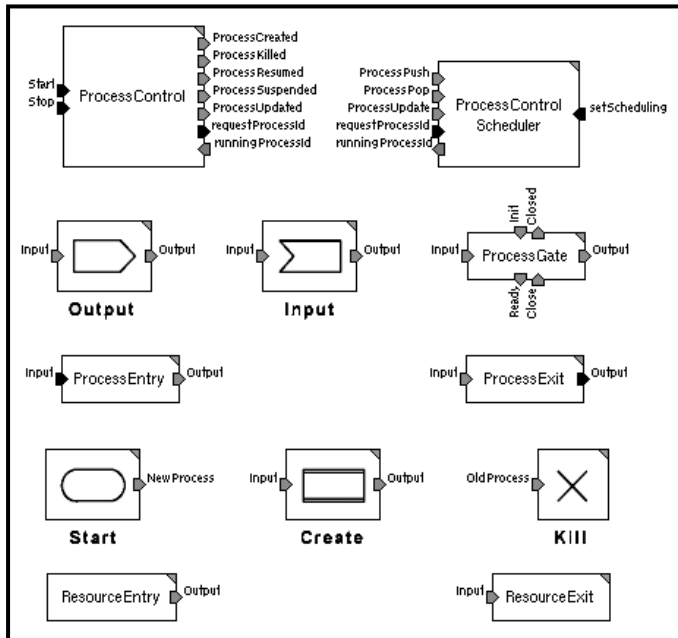
Spezifikationsgraph

und Busse des Architekturgraphen die funktionalen Knoten und Kommunikationsknoten des Problemgraphen abgebildet werden können. Dies ist Teil der Spezifikation des Gesamtsystems und so nennt man den zugehörigen Graphen auch Spezifikationsgraphen [4]. Wie läßt sich dieser Graph nun interpretieren? Der hier beispielhaft gezeigte Spezifikationsgraph läßt erkennen, daß alle vier Funktionen ($F_1 - F_4$) des Problemgraphen mit dem Hardwaremodul MC ausgeführt werden können. Des weiteren kann die Funktion F_4 auch auf das Modul M_2 und die Funktionen F_3 und F_2 auf das Modul M_1 abgebildet werden. In diesem Beispiel ergeben sich die zugehörigen Abbildungen für die Kommunikation implizit. Es kann aber auch sinnvoll sein, sie explizit anzugeben, wenn es mehrere Kommunikationspfade gibt, welche dieselben Hardwaremodule miteinander verbinden. Anforderungen der Ablaufplanung könnten dann dazu herangezogen werden, um einen sinnvollen Kommunikationspfad auszuwählen. Prinzipiell ist es natürlich möglich, die Abbildungskanten rein zufällig zu wählen. Ein funktionierendes System kommt dann natürlich kaum zustande. Es entspricht damit dann auch nicht der Spezifikation des Gesamtsystems. Eine gültige Abbildung funktionaler Knoten auf Hardwaremodule und Busse heißt »Gültige Implementierung«.

DAS PROZESS/RESSOURCEN-MODELL

Das ProcessResourceModel (PRM) wurde vom Autor entwickelt, um eine einfache und direkte Abbildung von Prozessen auf notwendige Ressourcen zu ermöglichen, so wie es im Spezifikationsgraphen modellhaft

dargestellt wird. Das PRM ist eine Weiterentwicklung des InstructionResourceModel (IRM), welches in [2] vorgestellt wird. Die Grundidee des PRM ist mit dem IRM verwandt: Es soll die Ausführung der in einem Datenflußmodell beschriebenen Funktion in einem Server zentralisieren. Im IRM senden dazu alle

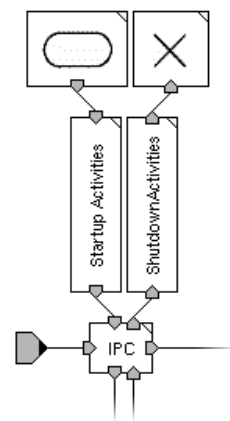


Modulbibliothek des PRM

ausführungsbereiten Funktionselemente des Datenflußmodells eine Anforderung an einen Server, der nach Beendigung der Ausführung der Funktion auf der Ressource eine Bestätigung zurücksendet. Die Modellierung der Funktion erfolgt dabei im Funktionselement. Dieses Vorgehen wurde auch im PRM ermöglicht. Die Abbildungsmöglichkeiten des PRM gehen jedoch weiter. Deshalb ermöglicht das PRM zusätzlich ein Ausführen funktionaler Elemente durch ein komplexes Architekturmodell. Der

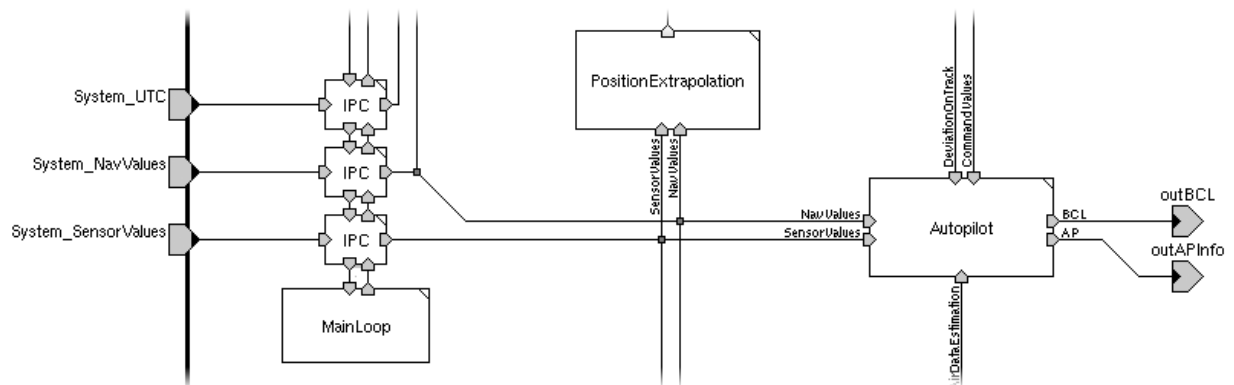
Anwender soll dadurch die Genauigkeit einzelner Komponenten eines verteilten Systems selbst bestimmen können, um eine Ausgewogenheit zwischen sog. Akkuratheit und Simulationsperformanz zu erreichen. Für das PRM wurde eine Bibliothek entwickelt, welche die wesentlichen Grundfunktionen von Prozessen und einem Server, der nun Prozeßkontrolle heißt, sinnvoll zur Verfügung stellt. In Teilen wurde sich dabei an Konzepten der Spezifikations- und Beschreibungssprache SDL orientiert (Module: Start, Create, Kill, Input, Output). Jedoch sollte vor allem eine strukturorientierte Entwicklung ermöglicht werden, die sich am Spezifikationsgraphen orientiert.

Das Modul »ProcessGate« stellt eine vereinfachte Möglichkeit, Kommunikation zu ermöglichen, um die sehr starre zustandsorientierte Implementierung mit Ein- und Ausgangsmodulen zu vereinfachen und komfortabler zu gestalten. Im Beispiel wurden »ProcessGates« für eine einfache Interprozeßkommunikation (IPC) eingesetzt. Sie können aber auch für die Kommunikation über eine Netzwerkarchitektur eingesetzt werden. Das hängt allein davon ab, auf welche Hardwarekomponente das »ProcessGate« abgebildet wird.



Die zustandsorientierte Implementierung hat den Nachteil, daß eine Prozeßschleife modelliert werden muß. Diese kann zwar, wie in SDL, auch für verschiedene Programmzustände unterschieden werden, gestaltet aber den Prozeßverlauf in vielen Fällen sehr unübersichtlich. Sinnvoller ist eine Modellierung der

Kommunikation, die strukturorientiert, und damit auch unmittelbar mit dem Problemgraphen vergleichbar ist. Der Systementwickler kann sich auf diese Weise mehr darauf konzentrieren, welche Komponenten miteinander kommunizieren, und nicht welche Zustände dazu eingenommen werden müssen. Dies hat besonders für transformative Systeme Vorteile.



Modellierung mit »ProcessGates«

Die Prozeßkontrolle hat allein die Aufgabe, funktionale Anforderungen auf Hardwarekomponenten abzubilden. Da nicht alle Anforderungen gleichzeitig abgebildet werden können, muß ein Scheduler der Prozeßkontrolle mitteilen, wann welche Funktion abgebildet werden kann. Es erfolgt also eine harte Trennung der Zuständigkeit der Komponenten für Bindung und Ablaufplanung. Betrachtet man die Prozeßkontrolle zusammen mit dem Scheduling, so kann man vom Betriebssystemkern sprechen. Eine ebenso harte Trennung erfolgt damit bei der Trennung der drei Ebenen »Funktion«, »Architektur« und »Betriebssystem«. Damit wurden die Grundlagen geschaffen, um eine Optimierung der Allokation, Ablaufplanung und Bindung vorzunehmen. In [4] wird eine Optimierung mit evolutionären Algorithmen vorgeschlagen. Eine automatische algorithmengesteuerte Optimierung stellt einen sehr fortschrittlichen Co-Design-Prozeß dar.

SIMULATION UND AUSWERTUNG

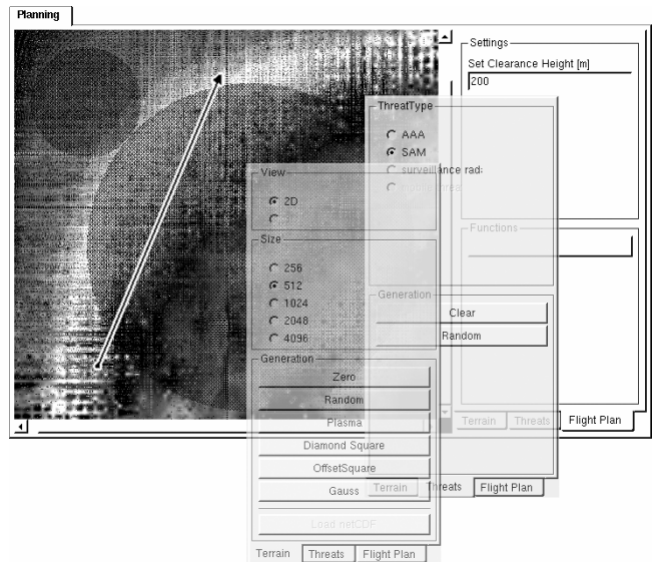
Für die Simulation des Modells für das Tiefflugsystem wurde eine Visualisierungs-Oberfläche auf der Basis von QT entwickelt, welche drei wesentliche Schritte des »Mission Level Design« abdeckt: Zu Beginn einer Simulation muß im Rahmen einer Planung die Operation näher definiert werden, welche simuliert werden soll. Anschließend wird die Simulation selbst durchgeführt und statistische und operative Daten gewonnen. Zum Schluß müssen alle gewonnenen Daten ausgewertet werden.

Die Planung für den Tiefflug teilt sich wiederum in drei Abschnitte auf. Zunächst wird festgelegt, wie das Gelände beschaffen ist, in dem der Tiefflug durchgeführt werden soll. Anschließend können Gefahrenpunkte definiert werden. Sie definieren sich durch ihre Position und den Operationsradius. In einem letzten Schritt werden im Rahmen der Flugplanung Start- und Zielpunkt definiert. Die Geländebeschaffenheit erzeugt sog.

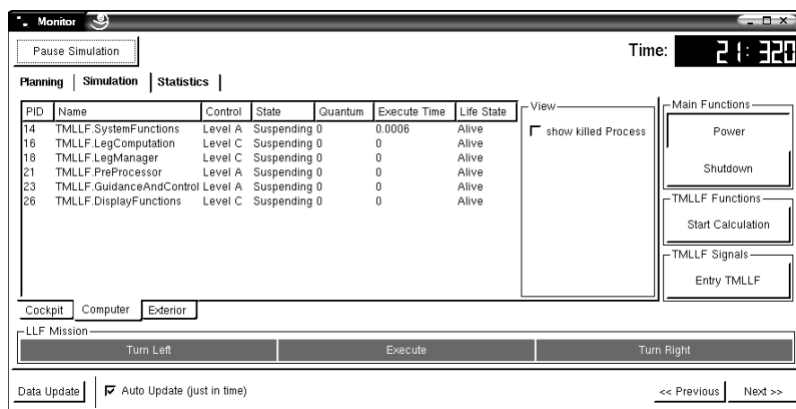
Radarschatten, den das Tiefflugsystem nutzt, um eine möglichst ungefährliche Flugroute (Trajektorie) vom Start zum Ziel zu berechnen. Der Tiefflug kann mit Hilfe der berechneten Trajektorie vollautomatisch durchgeführt werden.

Während der Simulation kann der Prozeßablauf durch eine Prozeßtabelle beobachtet werden. Diese kann Aussagen über den Zustand einzelner Prozesse machen und gibt zugleich Informationen über wichtige Scheduling-Parameter aus.

Sämtliche Veränderungen jedes einzelnen Prozesses werden protokolliert und können nach dem Ende der Simulation in einem zeitlichen Zusammenhang als Gant-Diagramm dargestellt werden. Sinnvoll ist dabei z.B. die zeitsynchrone Darstellung des horizontalen Flugverlaufs, so daß das Verhalten des Flugzeugs unmittelbar mit einzelnen Prozeßzuständen in Verbindung gebracht werden kann.



Missionsplanung



Missionsplanung

Eine möglichst umfassende grafische Darstellung kann bei der Fehlersuche hilfreich sein, was mit dieser Visualisierung ermöglicht werden sollte. Ein Problem liegt in der hohen Spezialisierung. Die Entwicklung von Simulationsoberflächen kann

zuweilen aufwendig werden und steht dann in keinem Zusammenhang mehr zu dem Nutzen. Ein grafisch zu bedienendes Modulkonzept könnte hier Abhilfe schaffen, da einzelne Komponenten immer wieder Verwendung finden können. Es darf nur nicht davon ausgegangen werden, daß jeder Systemdesigner zugleich ein begabter GUI Programmierer ist, soll ein Produkt am Markt breite Zustimmung finden.

AUSBLICK

Für den Co-Design-Prozeß ist ein adäquates Hardwaremodell unabdingbar. Eine Möglichkeit der Modellierung komplexer Architekturen, wie Prozessoren, mit MLDesigner wird in [5] vorgestellt. Dabei wurde eine komplette Architektur manuell anhand technischer Dokumente entwickelt.

Um für ein Co-Design belastbare Ergebnisse zu erhalten, muß ein Hardwaremodell die Zielhardware entweder fehlerfrei simulieren oder zumindest die Abweichungen genau benennen, so daß aussagekräftige Toleranzen für Simulationsergebnisse angegeben werden können. Dies stellt eines der wesentlichen Probleme dar, weil es z.B. keine Zertifizierungen für Hardwaremodelle gibt. In [3] wurde dagegen die Möglichkeit der Simulation mit Instruction-Set-Simulatoren vorgeschlagen, die für den GNU-Debugger für eine Reihe von Prozessoren verfügbar sind. Obwohl es diese Simulatoren bei weitem nicht für alle am Markt verfügbaren Prozessoren gibt, könnte eine Integration dieses Konzeptes in das PRM sinnvoll sein. Doch damit werden die Probleme auch nur zum Teil gelöst werden können, wenn es keine Ansätze für beliebige Architekturen gibt.

Hardwarehersteller sind nicht daran interessiert, HDL-Modelle besonders aktueller Produkte komplett freizugeben. Da diese Modelle aber nicht vollständig für eine Simulation notwendig sind, wäre z.B. die Entwicklung eines zertifizierten Prozesses denkbar, welcher ein HDL-Modell so weit reduziert, daß nur noch Informationen vorhanden sind, die für eine Co-Simulation notwendig sind.

Die aus den HDL-Modellen extrahierten Daten könnten dazu dienen, Simulationsmodelle zu generieren, welche dann die eigentliche Basis für die Co-Simulation bieten. Erst dadurch wäre es möglich, auch einen zertifizierten Co-Design Prozeß zu realisieren.

Literatur- bzw. Quellenhinweise:

- [1] Paluch, Nils: Diplomarbeit: Anwendung von Co-Design für verteilte Echtzeitsysteme; Einführung und Ausblicke am Beispiel des Tiefflugsystems TMLLF an Bord des Airbus A400M. Technische Universität Ilmenau, 2004.
- [2] Zens, Matthias: Diplomarbeit: Entwicklung von Methoden für die Analyse von elektronischen Ventilsteuerungssystemarchitekturen. Technische Universität Ilmenau, 2003.
- [3] Lohfelder, Thomas: Diplomarbeit: Modelling Operating Systems in MLDesigner. Ilmenau: Technische Universität Ilmenau, 2004.
- [4] Teich, Jürgen: Digitale Hardware/Software-Systeme – Synthese und Optimierung. Berlin Heidelberg New York: Springer Verlag, 1997 ISBN: 3-540-62433-3
- [5] Dilakanont, Nang (HSC Research Lab): A High-Fidelity DLX Processor Architecture Model. Gainesville: University of Florida, 2002.

Autorenangabe(n):

Nils Paluch
Dr. Achim Schönhoff
Univ.-Prof. Horst Salzwedel
Technische Universität Ilmenau, Helmholtzring 1
98693, Ilmenau
Tel.: +49 3677 69 13 16
Fax: +49 3677 69 12 18
E-Mail: Nils.Paluch@myxp.de