Simulatorkopplung durch Übersetzung
SMI - Simulation Module Interface

Ronald Ruzicka
SIMUTECH
Hadikgasse 150
A-1140 Wien

Abstract: Der Beitrag zeigt zwei Beispiele der Kopplung von Simulationssprachen durch Übersetzung. Die Methode der Übersetzung ergibt insbesondere dann Vorteile, wenn man die Probleme der Taktung und der Kommunikation leicht in den Griff bekommen möchte. Im ersten Beispiel wird unter Verwendung des Bondgraphtools BAPS ein SIMUL_R/BAPS Modell in ACSL übersetzt und dort simuliert. Einen wesentlich eleganteren Ansatz stellt das vorzustellende SMI dar, welches die Modulbildung in Teilmodellen erlaubt. SMI-Module können unter-einander beliebig gekoppelt werden, im Idealfall aus jedem Simulationssprachenmodell erzeugt und in jede Simulationssprache eingebettet werden. Vorteile, Entwicklungsstand und noch zu lösende Probleme werden angeführt.

1. Motivation

Im Sinne der auch in der Simulationswelt immer weiter um sich greifenden ganzheitlichen Sichtweise der Realität ist Simulatorkopplung derzeit ein sehr aktuelles Thema. Durch die Vielfalt der verwendeten Simulatoren ergeben sich jedoch viele Stolpersteine bei der integrativen Simulation eines Gesamtmodells.

Das große Problem bei der Kopplung von digitalen Simulationssprachen - und darunter wird i.a. die Simulatorbackplane verstanden [HESS95] - besteht neben der (EDV-technischen) Schnittstellenproblematik in

- der Synchronisation zu bestimmten Kommunikationszeitpunkten und
- der richtigen Taktung der Rechnung.

Zwischen zwei Kommunikationen der unterschiedlichen Simulatoren bzw. Modelle mit Zeitschrittweite h werden autonom (d.h. für jedes Modell separat) mehrere Rechenschritte durchgeführt. Unabhängig davon, wie exakt diese Rechnung auch sein mag, läßt sich einfach zeigen, daß für das - verteilte - Gesamtmodell immer ein

$$\text{Rechenfehler } O(h) = f*h$$

zu veranschlagen ist. Dies entspricht dem Euler-Verfahren! Je nach dem verwendeten Verfahren kann der Vorfaktor f kleiner oder größer ausfallen.

Ein zweites Problem entsteht, wenn durch die Kopplung von Teilmodellen algebraische Schleifen, also implizite Modelle, auftreten. Diese würden zur Lösung im Prinzip zumindest einen den Teilmodellen übergelagerten Algorithmus benötigen - über diese Tatsache schwindelt man sich meist numerisch hinweg. Beide numerischen Probleme gelten im übrigen auch für die Parallelisierung von Simulationsmodellen auf Teilmodellebene [RUZI93]!

Die Kopplung führt also zwangsläufig zu ungenauer Rechnung:

☹ viele Gesamtmodelle werden dadurch **unsimulierbar**, oder

☹ man muß die Schrittweite so herabsetzen, daß es zu erheblichen **Rechenzeitproblemen** kommt (als Richtwert bei Modellen aus der Praxis: Faktor 100)

2. Lösungsansätze

Obige Probleme lassen sich nur durch gemeinsame Rechnung der Modelle vermeiden. Zieht man weiters in Betracht, daß meist die Teilmodelle schon vorhanden sind und sie also in ein Gesamtmodell eingebracht werden müssen, so bleibt nur ein Weg: die Teilmodelle - aus unterschiedlichen Simulatoren - müssen in irgendeiner Form **übersetzt** werden. Zwei Wege bieten sich an: die

1. Verkopplung der Teilmodelle in einer Zielsprache durch Übersetzung (Abb. 1)
2. Modularisierung als *Component Ware* (Abb. 3)

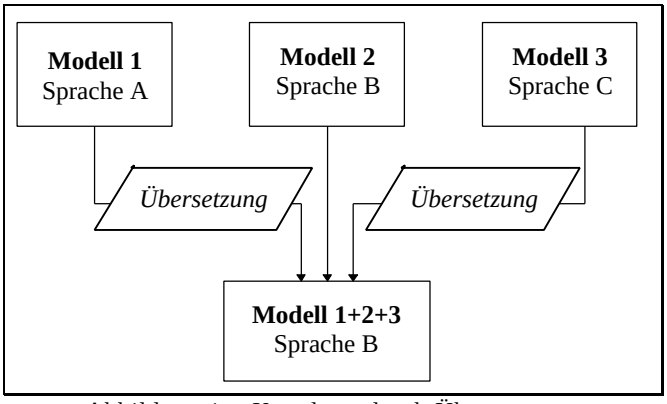

Abbildung 1: Kopplung durch Übersetzung.

2.1 Übersetzung

Bei der Übersetzung wird eine Zielsprache ausgewählt, die

- notwendigerweise die Modelleigenschaften aller Teilmodelle darstellen kann
- sinnvollerweise eine der Teilmodellsprachen ist (hier also Sprache B)

Als Beispiel sei hier das Bondgraphtool BAPS [RUZI88] genannt, das u.a. auch ein Subset von SIMUL_R [SIMU95] (Sprache A) versteht. Das BAPS/SIMUL_R Modell wurde in ACSL als Zielsprache (Sprache B) übersetzt und in dieses Modell ein bestehendes ACSL Modell inkludiert. Abb. 2 zeigt die Struktur von BAPS.

Die Vor- und Nachteile dieses Verfahrens sind rasch erklärt:

+ man erhält prinzipiell die Möglichkeit, zwei Modelle unterschiedlicher Simulationssprachen ohne numerische Probleme zu verknüpfen
+ der Zwischencode ist leicht übersetzbar (neue *Linker* für andere Sprachen können relativ rasch erstellt werden)

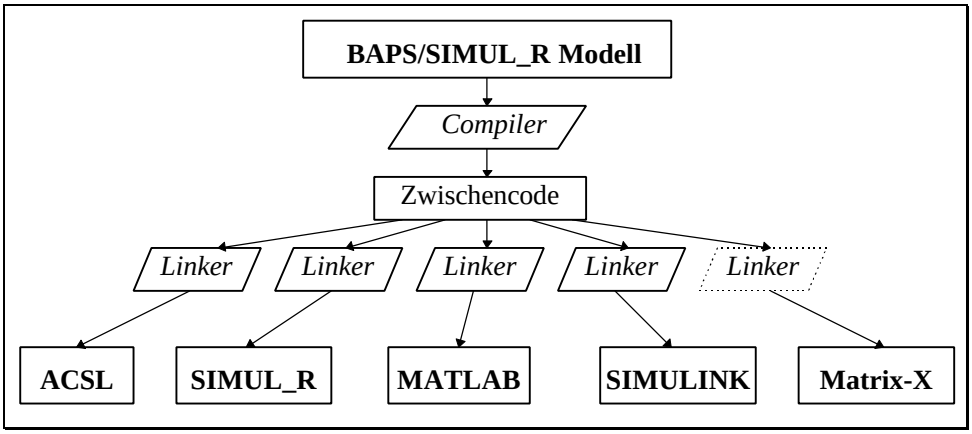

Abbildung 2: Die Struktur von BAPS.

- es bestehen starke Einschränkungen im Modellbereich:
  - ♦ keine Programmtexte (C, Fortran) direkt verwendbar (Subroutinen)
  - ♦ spezielle Funktionen haben unterschiedliche Argumente (z.B. Impuls)
  - ♦ gewisse sprachliche Elemente nicht portierbar:

    in ACSL - SIMUL_R's PDE
    in Matlab - SIMUL_R's discrete sections
    in SIMUL_R - Matlab Module
- Zusammenführen von Hand nötig, da Teilmodellbildung zu wenig unterstützt
- Programmtexte müssen vorhanden sein

Gerade der letzte Punkt - das Vorhandensein der Programmtexte - ist im industriellen Alltag oft ein Problem: in den Teilmodellen steckt unbezahlbar viel Know-How, das im Detail nicht an andere weitergegeben werden kann. Sehr wohl wäre es möglich die Funktionalität des Teilmodells als Blackbox zu übergeben - aber dann kann der Anwender nicht selbst übersetzen ...

Die Lösung kann die Modulbildung sein.

2.2 Modulbildung

ComponentWare ist ein Modebegriff aus der Informatik, der teilweise den Begriff Objektorientierung ersetzt bzw. ergänzt. Beispielsweise bietet die Programmierumgebung *VisualBasic* von Microsoft die Möglichkeit der einfachen Ergänzung von Visual-Basic Programmen durch zukaufbare Komponenten anderer Hersteller: z.B. für Datenbankzugriffe, graphische Editoren, Multimedia.

Auf die Simulationstechnik übertragen bedeutet dies die Modularisierung von Teilmodellen. Bei Bearbeitungsmodulen wird dies bereits verwendet: z.B. Matlab Toolboxes, SIMUL_R userdefined functions.

Unter Modulbildung im Modellbereich versteht man also die Möglichkeit, Teilmodelle in Module zu verpacken, die dann - idealerweise - von jedem Simulator als Teilmodelle eines Gesamtsystems simuliert werden können; und, was noch wich-tiger ist: diese Module können aus Modellen beliebiger Simulatoren abgeleitet werden. Soweit die Idee.

Der Vorgang bei der Modulbildung sieht folgendermaßen aus: Module (ohne Rechenalgorithmen!) mit genormten Schnittstellen (Abb. 3) werden zu einem gemeinsamen Modell zusammengefaßt (Abb. 4) und von einem Simulator gerechnet.

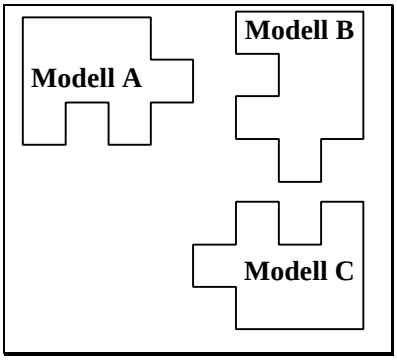

Abbildung 3: Teilmodelle als Module

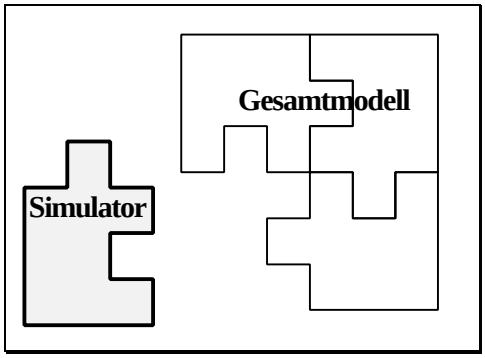

Abbildung 4: Gemeinsame Simulation von Modulen

☺ alle Simulatoren können Module nach der Schnittstellendefinition erzeugen
☺ alle Simulatoren, die die Schnittstellen beinhalten, können die Modelle rechnen

3. SMI - Das Simulation Module Interface

3.1 Definition

Um die Modulbildung überhaupt zu ermöglichen, ist eine Schnittstellendefinition von Nöten. Hierzu wurde das ***S***imulation ***M***odule ***I***nterface eingeführt, eine Sammlung von genormten Funktionsaufrufen und Strukturen (in C, via Interfaces auch in Fortran). Folgende Komponenten sind darin enthalten:

- **Modellstruktur:**
  - Integranden (als System 1. Ordnung)
  - algebraische Gleichungen (implizit oder explizit)
  - Zeit- und logische Ereignisse (z.B. als discrete sections)

- **Modellgrößen:**
  - Eingänge
  - Ausgänge
  - algebraische Abhängigkeiten
  - Informationen; z.B.Anzahl der Zustandsgrößen, implizit/explizit

Wichtig ist hierbei insbesondere, daß nicht nur implizite Teilmodelle gerechnet werden können, sondern auch Vorkehrungen getroffen sind, die die Lösung eines *impliziten Gesamtmodells* ermöglichen - informatisch gesprochen also die Umsortierung der Gleichungen innerhalb des Moduls erlaubt.

SMI ist keine neue Simulationssprache, sondern enthält nur die notwendigen Informationen in komprimierter Form. Daher ist das Interface für unterschiedlichste Systeme rasch implementierbar!

Prinzipiell wurde SMI dafür entwickelt, compilierten Code weiterzugeben - also etwa DLLs unter MS-Windows. Jedoch bleibt es dem Ersteller vorbehalten, auch die hinter den Funktionsaufrufen steckenden SMI-Funktionen - also die Modellstruktur und die Modellvariablen - an Anwender weiterzugeben.

3.2 Einordnung

Wie ist SMI mit anderen Schnittstellendefinitionen zu vergleichen?

Matlab/Simulink Mex-Files	SMI-ähnlich, eine proprietäre Schnittstelle, enthält aber keine Vorkehrungen für DAEs (implizite Modelle) und Ereignisse [MATH92]
Simulink SimStruct	SMI-ähnlich, nicht proprietär, enthält aber keine Vorkehrungen für Ereignisse [MATH94]
Matrix-X/SystemBuild Block-Script; User Code Block	eine proprietäre Schnittstelle, kann in C-Code übersetzt werden [INTE92]
DSblock	SMI-ähnlich, nicht proprietär, implizite Modelle und Ereignisverwaltung möglich, aber keine impliziten Gesamtmodelle [OTTE95]
VHDL-A	neue, im Stadium der Normierung befindliche Simulationssprache (entspricht eher Variante 2.1); der Sourcecode wird benötigt; relativ aufwendige Erstellung von Compilern aus anderen Simulationssprachen in VHDL-A [VACH95]

Vor- und Nachteile der Modulbildung sind:

* praktisch jeder derzeit am Markt befindliche Simulator könnte als Modul-generator verwendet werden
* als Simulator können verwendet werden, die
  ⇒ die Schnittstellendefinition erfüllen
  ⇒ algebraische Schleifen (im Gesamtmodell) erkennen und lösen können
  ⇒ diskrete Ereignisse behandeln können
– Simulatorspezifische-, nicht-Standard-C-Funktionen müssen als Bibliothek mitgegeben oder emuliert werden
\+ klare Modul- und Bibliotheksbildung
\+ einfache Austauschbarkeit (Arbeitsgruppen)
\+ kein "O(h)" Problem mehr
\+ einfache Modulkopplung (*connect* in parallelen Programmiersprachen)
\+ jeder Anwender kann mit *seinem Simulator modellieren* (Module generieren)
\+ jeder Anwender kann mit *seinem Simulator simulieren* (Module benützen)
\+ Modelle können ohne Source weitergegeben werden (Wissensschutz)
\+ ein neuer Markt für mehrfach verwendbare Module entsteht

3.3 Stand der Entwicklung

Der derzeitige bzw. in Kürze implementierte Stand der Entwicklung stellt sich wie folgt dar (ein Pfeil vom SMI Symbol in eine Sprache bedeutet, diese kann SMI simulieren; ein Pfeil in SMI hinein: die Modelle der Simulationssprache können in SMI Module umgewandelt werden), siehe Abbildung 5:

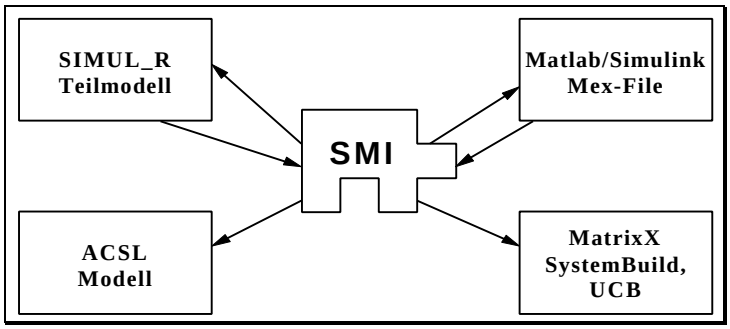

Abbildung 5: Entwicklungsstand SMI

4. Zusammenfassung

Simulatorkopplung durch Übersetzung ist der einzig gangbare Weg, um durch die Modellverbindung entstehende numerische Probleme zu lösen. SMI bietet einen umfassenden Ansatz zur Modularisierung und simulatorunabhängigen Erstellung und Simulation von Teilmodellen.

5. Literatur

[HESS95] E. Hessel: *Model exchange - illusion or future reality?*, Proc. 1995 Eurosim Conference, Wien, Elsevier

[INTE92] Integrated Systems: *SystemBuild Block Library Reference Guide*, Santa Clara, CA, 1992

[MATH92] The MathWorks Inc.: *SIMULINK User's Guide*, Natick, Massachusetts, 1992

[MATH94] The MathWorks Inc.: *SIMULINK Release Notes Version 1.3*, Natick, Massachusetts, 1994

[OTTE95] M.Otter: *The DSblock model interface for exchanging model components*, Proc. 1995 Eurosim Conference, Wien, Elsevier

[RUZI88] R.Ruzicka: *SIMUL_R - eine Simulationssprache mit speziellen Befeh-len zur Modelldarstellung und -analyse*, Informatik Fachberichte 179, Proc. 5. Symp. Simulationstechnik Aachen, Springer, 1988

[RUZI93] R.Ruzicka: *SIMUL_R PARALLEL - hardware-in-the-loop Simulation mit Transputern unter Windows*, Fortschritte in der Simulationstechnik 6, Proc. 8. Symp. Simulationtechnik, Vieweg, Berlin, 1993

[SIMU95] SIMUTECH: *SIMUL_R - User's Guide 2.50*, Wien, 1995

[VACH95] A. Vachoux: *Analog and Mixed-Mode Extensions to VHDL*, Proc. 1995 Eurosim Conference, Wien, Elsevier