hochschule mannheim

Methode zur Entwicklung sicherheitskritischer eingebetteter Systeme mittels deterministischer UML-Modelle

Workshop "Entwicklung zuverlässiger Software-Systeme"

MSc Dipl.- Ing. Zamira Daw

Regensburg, 18. Juni 2009

Agenda

- Motivation
- Stand der Technik
- DMOSES - ***D****eterministische* ***MO****delle für* ***S****ignalverarbeitende* ***E****ingebettete* ***S****ysteme*
- *DMOSES-Profil*
- *Die fundamentale ausführbare Einheit in DMOSES*
- DMOSES-Toolchain
- Einsatz von DMOSES für die Entwicklung sicherheitsrelevanter eingebetteter Systeme
- Ausblick

Motivation

- Steigende Anforderungen
- Leistungsfähige Systeme durch Einsatz von parallelen Bearbeitungen (Multi-Threading bzw. Multicore)
- Nicht-deterministisches Verhalten bei parallelen Bearbeitungen
- Unterstützung von modellbasierten Methoden

Stand der Technik

SySML: *Systems Modeling Language*

- Modellierung von Anforderungen
- Verbindung zwischen Spezifikationen, Anforderungen und Systemen auf Modellebene

MARTE: *A UML Profile for Modeling and Analysis of Real-Time Emdedded Systems*

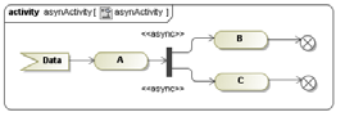
- Konzept von Zeit innerhalb von UML-Modellen
- Detaillierte Modellierung von Hardware-Ressourcen

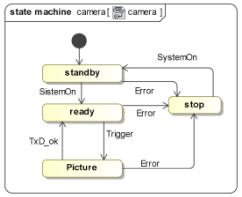
DMOSES - Deterministische MOdelle für Signalverarbeitende Eingebettete Systeme

Nicht-deterministische Systemen sind instabil und nicht vorhersehbar

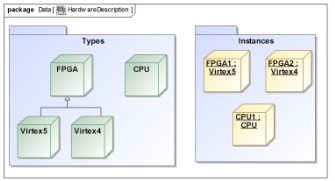
Gewährleistung von einem deterministischen Systemverhalten unabhängig von Hardware-Plattform

Verhaltensbeschreibung



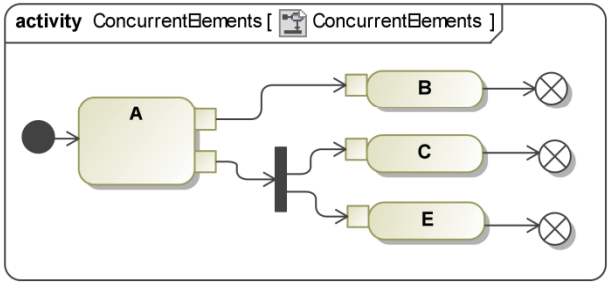


Hardware-Plattform Modellierung



Nicht-deterministische Modelle innerhalb der UML

Mangelhafte Definition von parallelen Bearbeitungen bildet nicht-deterministisches Verhalten

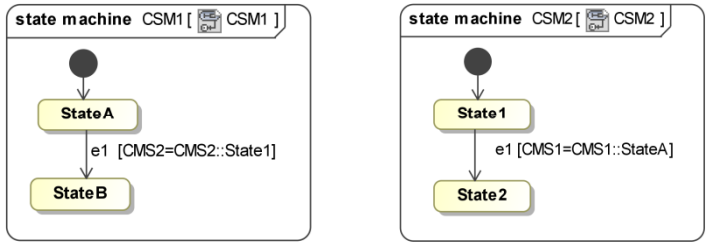


- Unabhängigkeit
- Möglichkeit von gleichzeitiger Ausführung

Mit nur einer Ressource: Welche muss als erstes ausgeführt werden B, C oder E?

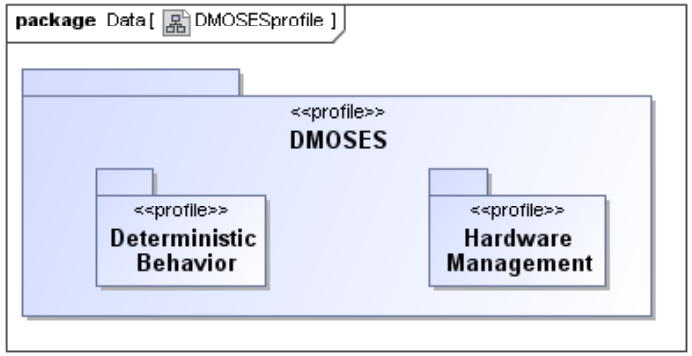
Nicht-deterministische Modelle innerhalb der UML

Mangelhafte Definition von parallelen Bearbeitungen bildet nicht-deterministisches Verhalten



Endergebnisse abhängig von der Ausführungsreihenfolge

DMOSES-Profil



Deterministic Behavior:

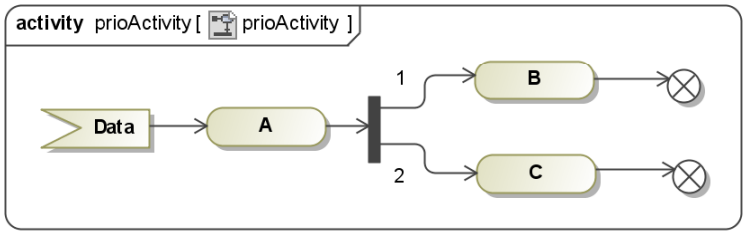
- Sicherstellung von deterministischem Systemverhalten

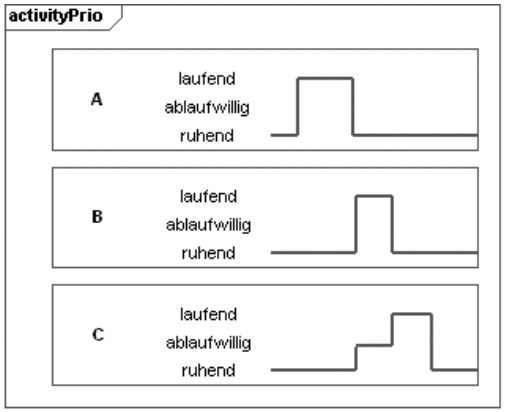
Hardware Management:

- Erweiterung der Beschreibung von Hardware-Plattformen
- Verbindung zwischen den Hardware-Elementen und der Funktionalität des Modells

Sicherstellung von deterministischen Modellen

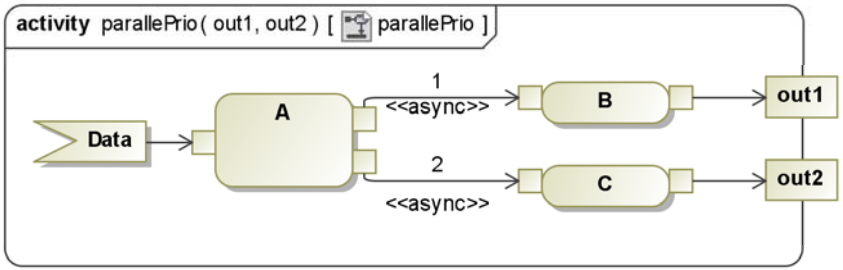
Ungewissheit über Ausführungsreihenfolge werden durch Anbringen von Prioritäten an die Kanten ausgeschlossen

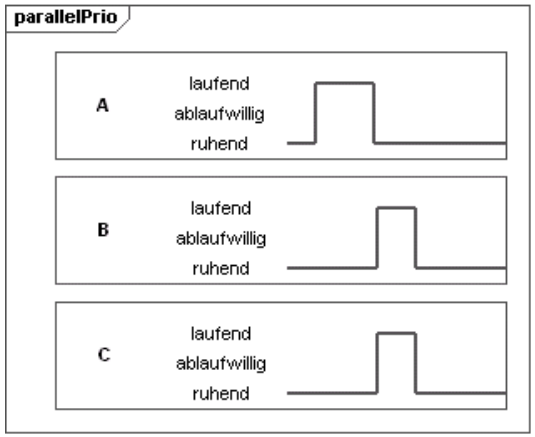




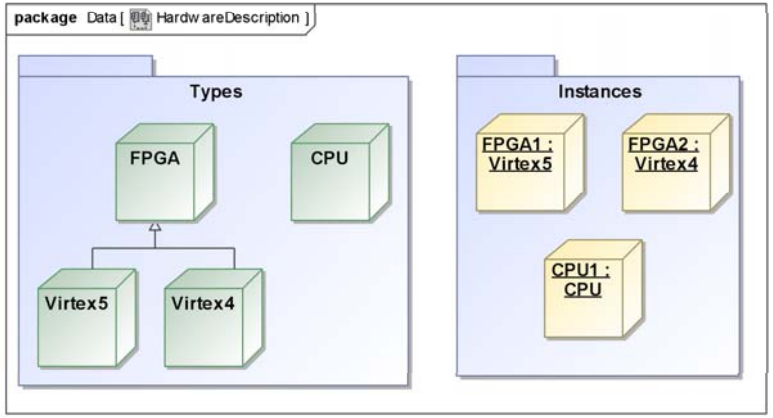
Sicherstellung von deterministischen Modellen

Der Stereotyp *async* trennt im Modell die Funktionalität von der Ausführung und ermöglicht so die Verwaltung der Ressource auf der Modellebene



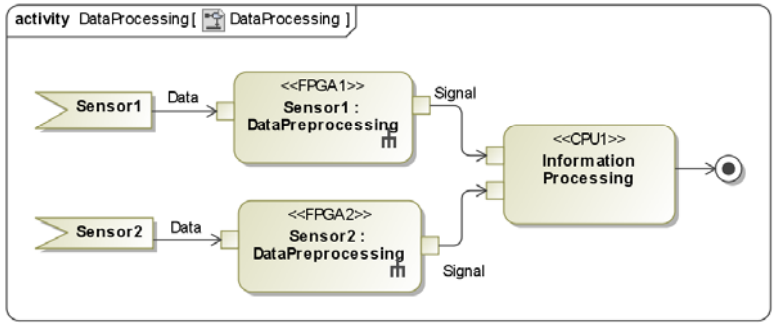


DMOSES-Profil -> Hardware Package



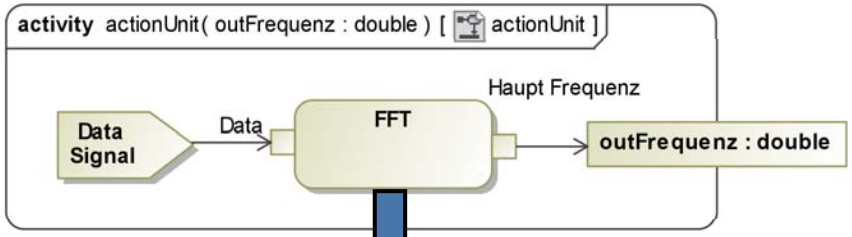
Beschreibung der Hardware-Plattform

Aktionen und Zustandsautomaten werden durch Stereotypen mit den Instanzen der Plattformen verbunden



Die fundamentale ausführbare Einheit in DMOSES

Aktion ist die kleinste Einheit ausführbarer Funktionalität im Modell



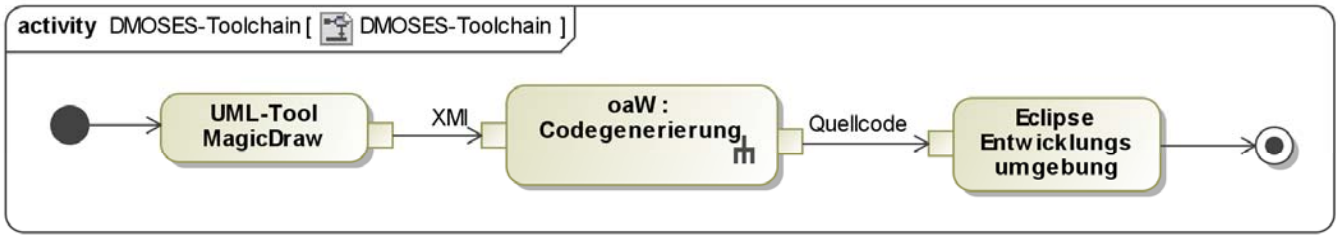

```
/**
 *
 */
bool calculate ()
{
    //----ToDo----

    return true;
}
```

Implementierung

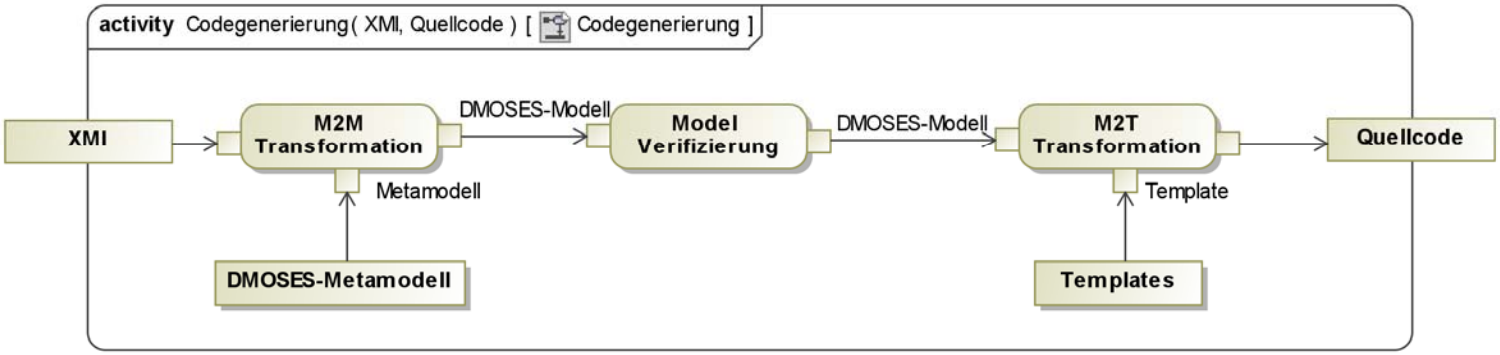
- Darstellung von einfachen oder komplexen Funktionalitäten
- Implementierung auf der Codeebene
- Geschlossenes Element
- Pins dienen als Schnittstelle

DMOSES-Toolchain

- Unterstützung der Entwicklung von eingebetteten Systemen von Modellierung bis zur Realisierung
- Gewährleistung von deterministischen Verhalten auf der Modell- und Codeebene

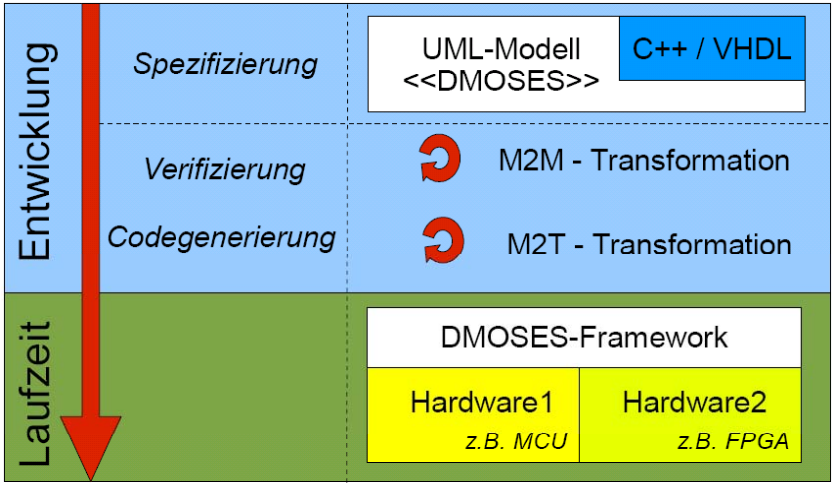


DMOSES-Toolchain: Codegenerierung



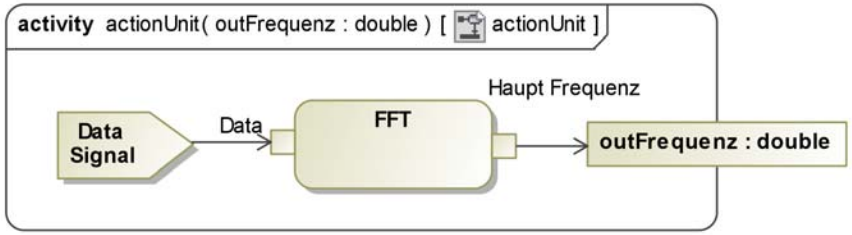
- **M2M Transformation:** Wandelt ein UML-Modell in einem DMOSES-Modell um
- **Modell-Verifizierung:** Modellierungsfehler werden identifiziert und behoben (z.B. Datenintegrität)
- **M2T Transformation:** Aus DMOSES-Modellen wird Code durch definierte Templates erzeugt. Automatische Erzeugung von Tests:
  - Model → Graphenanalyse
  - Contract → Äquivalenzklassenbildung → Grenzwertanalyse

Einsatz von DMOSES für die Entwicklung sicherheitsrelevanter eingebetteter Systeme



- Modularität
- Wartbarkeit
- Deterministisches Verhalten
- Modellverifizierung
- Automatische Code- und Testgenerierung

Einsatz von DMOSES für die Entwicklung sicherheitsrelevanter eingebetteter Systeme



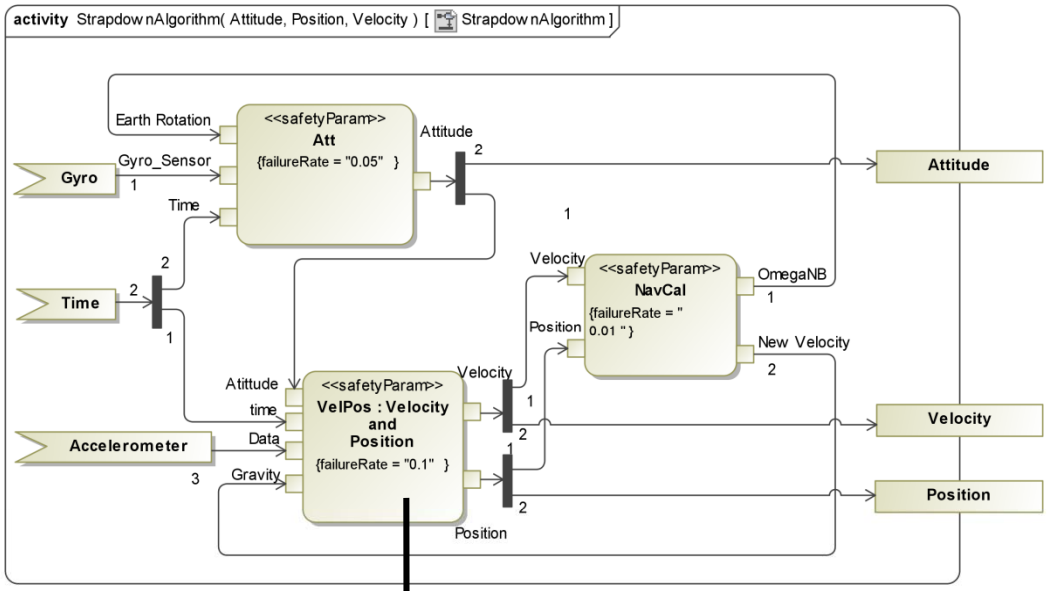
Pin:

- Datenintegrität
- Contracts

Aktion:

- Vor- und Nachbedingungen
- Freiheit für die Implementierung der Aktion

Einsatz von DMOSES für die Entwicklung sicherheitsrelevanter eingebetteter Systeme




```
 */
bool calculate ()
{
    //----ToDo----

    return true;
}
```

Innerhalb der *Calculate*-Methode werden die Metriken berechnet

DMOSES-Technologien

C++, VHDL, Haskel, C#, C,
(Java ?, Ada ?)

Nativ (single Thread)	ARM-RTX	Win CE/Win 32/64	POSIX		
			QNX	RT-Linux	…

Ergebnisse

- Codegenerierung von C++, C#, VHDL, Haskell
- Eclipse Plug-in von DMOSES-Tool

  Download *www.dmoses.org* Release1.0 ab November 2009

Ausblick

- Implementierung von unterschiedlichen Softwaremetriken
- Verbindung zwischen dem Modell und zeitlichen Werten aus Simulatoren der entsprechenden Hardware-Plattformen oder SystemC

DMOSES-Team

Projektsleiter
Prof. Marcus Vetter

Modellierung von SES
Zamira Daw

Entwicklung Eclipse Pluing
Flor Alvarez

Entwicklung C++ Framework
Christian Englert

Entwicklung VHDL Framework
Rüdiger Willenberg

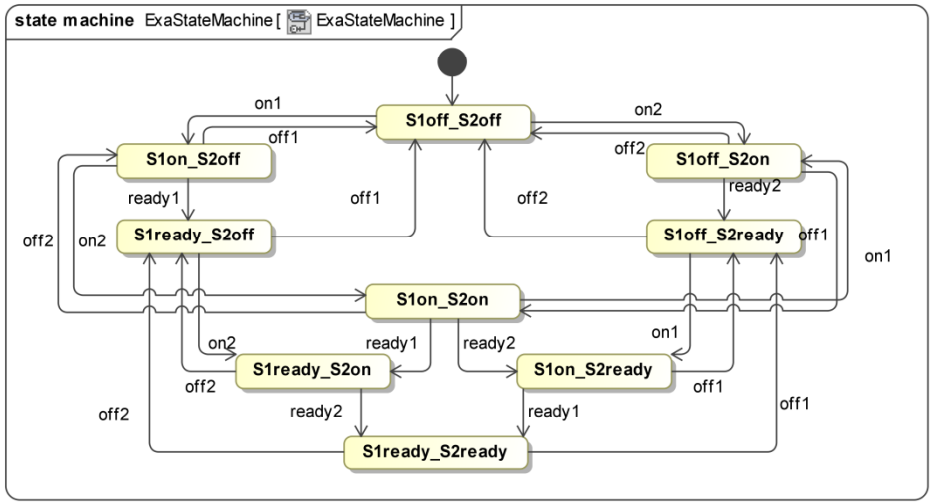
Vielen Dank für Ihre Aufmerksamkeit

Workshop "Entwicklung zuverlässiger Software-Systeme"
MSc Dipl.- Ing. Zamira Daw
Regensburg, 18. Juni 2009

DMOSES-Tool Vorführung

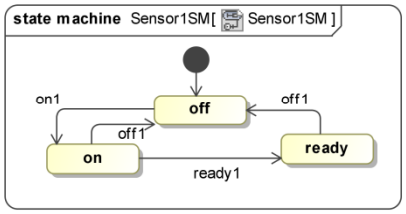
Nicht-deterministische Modelle innerhalb der UML

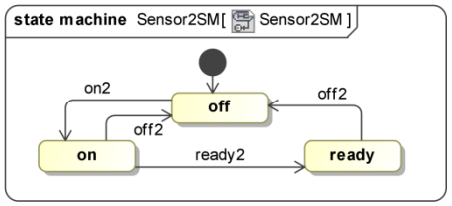
Nebenläufige Zustandsautomaten machen das Modell überschaubarer



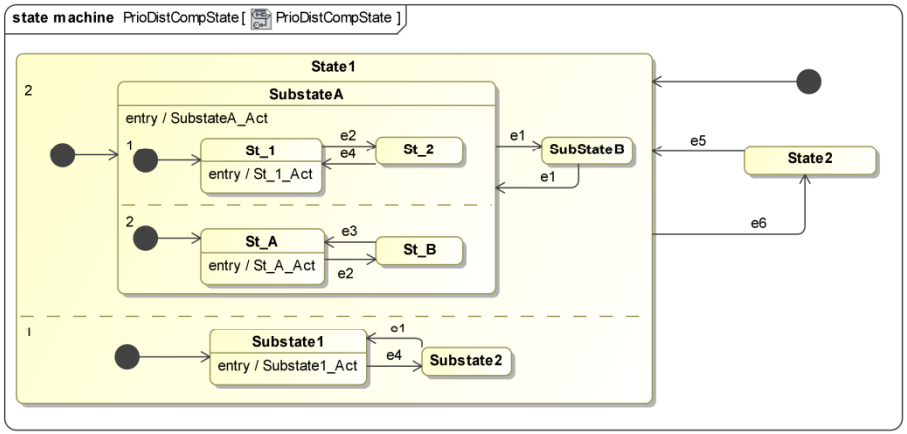
Nicht-deterministische Modelle innerhalb der UML

Nebenläufige Zustandsautomaten machen das Modell überschaubarer





Sicherstellung des deterministischen Verhaltes



Orthogonale Regionen innerhalb zusammengesetzter Zustände modellieren Unabhängigkeit und parallelisierbare Ausführung