Model-based Design

182.721 VO Embedded Systems Engineering
Armin Wasicek
6.12.2011
Overview

Introduction
System Theory
Model Driven Architecture (MDA)
MBD Development Environments
Summary
History

General Systems Theory (GST) was the fundamental underpinning of most commercial software design techniques by the 1970s. Computer-Aided Software Engineering (CASE) tools developed in the 1980s. Creating the Unified Modeling Language (UML) in the 1990s. Model Driven Architecture (MDA) launched in 2001.
Challenges

Exploding code sizes and complexity of software
Increase the productivity of the individual engineer
Agility of applications, technologies, platforms
Coordinate the resources of people with expertise in a wide range of disciplines
When software engineers build systems...

Software-centric design methodologies:
- Treat software as main entity
- Overlook domain knowledge
- Dominate the functional design

UML is not the answer to system design
Overview

Introduction

System Theory

Model Driven Architecture (MDA)

Models of Computation

Summary
Introduction to System Theory

System theory is an approach to a complex structure that abstracts away from the particular physical, chemical, or biological nature of its components and simply considers the structure they together implement, in terms of the functional role of individual parts and their contribution to the functioning of the whole.

Systems theory first originated in biology in the 1920s out of the need to explain the interrelatedness of organisms in ecosystems.

Systems as interacting components: Purpose, Environment, Interrelations, Boundary/Interfaces, Input/Output

Emergence is the way complex systems and patterns arise out of a multiplicity of relatively simple interactions.
Which models are there?

Scale models
Partial models
Process models
State models
Etc.

Substitute for direct measurement and experimentation
Model Definition

Representation
A model represents some *thing*.
Model and *thing* are connected by a *morphism*.

Abstraction
The model suppresses irrelevant detail and focuses on important aspects.

Pragmatics
The model is created for a purpose.

[Herbert Stachowiak, *Allgemeine Modelltheorie*]
“What’s a model?“  "The code is the model“  "Manage code and model“  "The model is the code“  "Let’s talk models!“
Model representations

Textual

```
mttype = {REQ,ACK};
typedef Msg {
  byte a[2];
mtype tp;
};
chan toR = [1] of {Msg};
bool flag;
proctype Sender() {
  Msg m;
  ...m.a[0]=2; m.a[1]=7; m.tp = REQ;
toR ! m;
}
proctype Receiver(byte n) {
  Msg m;
  ...
toR ? m;
}
init {
  run Sender();
  run Receiver(2);
}
```

Hierarchical

Classic Hierarchical Model

Graphical

```
Graphical modeling tools reduce the complexity of model designs by breaking them into hierarchies of individual design blocks.
```

10.12.2010 Model-based Design 11
Model Driven X

Model Driven Architecture (MDA)
Model Driven Development (MDD) (e.g. UML)
Model Driven Engineering (MDE)
Model Based Engineering (MBE)
Overview

Introduction
System Theory
Model Driven Architecture (MDA)
MBD Development Environments
Summary
Traditional approach for a controller

Development strategy for a digital control system:
- Design controller on paper with block diagrams
- Implement the design in a programming language
- Integrate controller with the plant model
- Adjust parameters of the controller through simulation
- Trash or take: reiterate the design until satisfactory

Rapid prototyping is the **application of productivity tools** to develop working prototypes of control systems in the minimum amount of time.
Model-Based Design

- From Paper-based approach to Executable Model
- Model captures all information about concept, design, implementation
- Model is used in all development stages (Research, Design, Implementation and Verification&Validation)
- Model is continuously updated and elaborated
Applications of System Models

Models reflecting all different aspects of the required system are developed and used throughout all stages of the system development process:

Simulation, Visualization
Static System Analysis, (e.g., predict energy consumption)
Formal Verification
Virtual Fault Injection
Synthesis of Implementations & Test Suites
Documentation, Presentation
Key steps in the MBD approach

(1) **System identification** (modeling the plant) is an iterative process. By acquiring and processing raw data from a real-world system and choosing a mathematical model to represent the plant behavior.

(2) **Controller analysis and synthesis.** Identify dynamic characteristics of the plant model and synthesize an appropriate controller.

(3) **Offline simulation.** Simulation allows specification, requirements, and modeling errors to be found early.

(4) **Deployment**
MBD: System Simulation

- **Simulation Goals**
  - Verification and Validation
  - Collect Data to Improve Models

- **Software in the Loop**
  - Non real-time simulation, e.g. using SimuLink
  - Using plant model and implementation of controller model

- **Hardware in the Loop**
  - Real-time simulation of plant model
  - Using plant model and controller implemented on target
Software-in-the-Loop (SiL) Simulation

**Goal:** verify that code generated from a model will function identically to the model

Generate code from models (e.g., c-code)

Run code together with model of plant on the development platform

**Benefits:**
- Low cost
- Target platform must not be specified
Hardware-in-the-Loop (HIL)-Simulation

Form of real-time simulation
Purpose of a HiL is to provide all of the electrical stimuli to fully exercise the ECU
Connect actual ECU to simulator
Plant is simulated, ECU is real

1. Model of the plant
2. Sensor models
3. Real-time target
4. Real or simulated loads
5. Fault insertion relay matrix
6. Host PC to
   • Control simulation
   • Collect diagnostics
Key benefits of the MBD approach

MBD provides a **common design environment**, which facilitates general communication, data analysis, and system verification between development groups.

Engineers can **locate and correct errors early** in system design, when the time and financial impact of system modification are minimized.

**Design reuse**, for upgrades and for derivative systems with expanded capabilities, is facilitated.
Shift from 'V' to 'Y' Model

**Domain Engineering:**
identify potentially reusable components

**Frameworking:**
generic components and their relationships

**Assembly:**
specialization of reusable components

**Archiving:**
make a reusable component available

Model Driven Architecture (MDA)

OMG Standard: Model Driven Architecture (MDA).

**Application of models and generators** to improve the software development process

The objective is **NO total automation**, but a reasonable proportion

**Goals**

- Speed up the development process: „automation through formalization“
- Tackle the complexity challenge through abstraction
MDA Process

a. **Computation Independent Model (CIM):** informal specification

b. **Platform Independent Model (PIM):** model which is independent from a specific computational platform (for business processes)

c. **Platform Specific Model (PSM):** model which is dependent on a particular architecture, service definitions

d. Code model and target platform

MDA defines conceptual separation between models and the transformation between models:

- Model-to-model transformation (MOF QVT)
- Model-to-code transformation
Architecture Analysis and Design Language (AADL)

Textual and graphic language (SAE standard)
Design and analyze the software and hardware architectures of embedded and real-time systems
  - End-to-end latency
  - Schedulability
  - Reliability, …

Describe the structure as an assembly of software components
Mapping to an execution platform
AADL - Context Diagram for a Set of Vehicle Control Systems

AADL - Context Diagram of the Cruise Control

AADL - Software Component View of the Cruise Control
AADL - System Hierarchy for the Cruise Control
Basic Comparison of UML and AADL

<table>
<thead>
<tr>
<th></th>
<th>UML</th>
<th>AADL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Diagrams tradition</td>
<td>Language tradition</td>
</tr>
<tr>
<td>Purpose</td>
<td>Depict functional structures</td>
<td>Define runtime behavior</td>
</tr>
<tr>
<td>Representation</td>
<td>Diagrams; graphic</td>
<td>Textual and graphic</td>
</tr>
<tr>
<td>Verification</td>
<td>---</td>
<td>Automated analysis</td>
</tr>
<tr>
<td>Current Domains of</td>
<td>Software, business processes, and many</td>
<td>Embedded and real-time software system</td>
</tr>
<tr>
<td>Use</td>
<td>others</td>
<td></td>
</tr>
</tbody>
</table>

System designers can use UML to diagram functional structures, and AADL to define runtime behavior.

SysML and the MARTE profile extend UML for embedded and real-time systems.
# UML & AADL Focus Areas

<table>
<thead>
<tr>
<th>UML or AADL</th>
<th>Core Area</th>
<th>Extended Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>UML</td>
<td>Functional structures</td>
<td>Analysis through SysML and MARTE</td>
</tr>
<tr>
<td>AADL</td>
<td>Runtime architecture modeling and analysis</td>
<td>Error handling through error model annex; formal verification through behavioral annex</td>
</tr>
</tbody>
</table>
Overview

Introduction
System Theory
Model Driven Architecture (MDA)
**MBD Development Environments**
Summary
MatLab/SimuLink

Development Environment for multi-domain simulation and model-based design of dynamic systems
Basic notation are blocks and data flows between them
Provides domain-specific toolboxes (e.g., automotive)
Provides code generation facilities
dSPACE

Leading developer of MBD tools for automotive applications

dSPACE products help to reduce complexity of today’s embedded software engineering

Integrated Development Environment features:

- system design, rapid control prototyping, automatic production code generation, and HIL testing
Esterel SCADE

SCADE addresses mission and safety-critical embedded applications.
SCADE is certified/qualified

Features
- Graphical and text editor
- Simulation
- Formal verification
- Code generation
- Model Test Coverage
- Gateways: Simulink, DOORS, Altia, UML/SysML, etc.
Eclipse Modeling Framework

Eclipse-based **modeling framework** and **code generation facility** for building tools and other applications based on a **structured data model**

Ecore is a language to define meta-models

“Your models instantiate the meta-model, and your meta model instantiates Ecore”

**Xtext** can be used to create a domain-specific language

People use Xtext-based languages to drive code generators that target Java, C, C++, C#, Objective C, Python, or Ruby code
Overview

Introduction
System Theory
Model Driven Architecture (MDA)
MBD Development Environments
Summary
MBD Increase of productivity

The MBD approach can dramatically increase a system designer’s productivity

<table>
<thead>
<tr>
<th>SCADE Reductions of efforts in percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software development cost</td>
</tr>
<tr>
<td>Software certification cost</td>
</tr>
<tr>
<td>Software update cycle time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MatLab Reductions of efforts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development time</td>
</tr>
<tr>
<td>Design reuse</td>
</tr>
<tr>
<td>Debugging process</td>
</tr>
<tr>
<td>Code efficiency</td>
</tr>
<tr>
<td>Minimized documentation</td>
</tr>
<tr>
<td>Collaboration simplified</td>
</tr>
</tbody>
</table>
Summary

Models and their application
MBD offers a systematic engineering approach for embedded systems design
Industry-grade software development is not imaginable without model-based testing
Key benefits of MBD:
  - Design reuse
  - Faster development cycle
  - Early detection of faults
Danke für die Aufmerksamkeit!