Model Based Engineering for the support of Models of Computation: The Cometa Approach

MPM 2011 at proceedings of MODELS 2011
October 16th, 2011

Papa Issa DIALLO, Joël CHAMPEAU, Vincent LEILDE
ENSTA Bretagne, 2 rue François VERNY 29806 Brest
papa_issa.diallo@ensta-bretagne.fr
INTRODUCTION

MOTIVATION

COMETA APPROACH

EXPERIMENTATION

CONCLUSION
Introduction (1/3) : Context

• New systems requires the integration of several technologies and tools (video processing, wireless, software) combining Hardware/Software

• Such heterogeneity increases complexity of their design and testing.

• Solutions:
  • High level design of systems (raise the abstraction level of currently used languages in ESL)
  • Modular decomposition of the systems
  • Early validation of properties and performance analysis

• MBE proposes approaches for high level design:
  • Separation of application concerns from platform concerns
  • Definition of several level of abstraction
  • Early property validation
Introduction (2/3) : Existing Approaches

• Several known frameworks for heterogeneous systems Design
  • Ptolemy, Modhel’X, Metropolis, Forsyde, BIP

• MBE Technologies:
  • UML, MARTE
  • SysML
  • DSLs/ DSMLs (AADL, Wright, etc)

• Main Challenges:
  • Adding execution semantics to models
  • More flexibility to the definition of execution semantics
  • Provide support for the interpretation of models (within different interconnected tools).
Introduction (3/3) : MBE Shortcomings

How modules are executed?
Motivation

- **Enrich models with semantics** (MoC) explicitly described
- Solution for describing *concurrency, communication, data, and time* at early design stages

- **Presented solution**: Cometa Metamodel for the description of:
  - Execution semantics (MoC)
  - Structure of systems

- Additional possibilities of Cometa:
  - Flexibility on the description of execution semantics.
  - Portability to different execution environment
Cometa Approach

System

Module 1 (Signal Processing)

Module 2 (Control)

Module 3 (Control)

Cometa Metammodel

SDF MoC

FSM MoC

FSM MoC

Module 1

Module 2

Module 3

Library

Model with Execution semantic

Execution Environment

Ecole Nationale Supérieure de Techniques Avancées Bretagne - 16/10/11 – page 7
• In Cometa, capture of MoC is **based on four concerns:**

  • Capture of **Behavioral properties**
  • Capture of **Communication properties**
  • Capture of **Data properties**
  • Capture of **Time properties**

  • Inspired from the Rugby conceptual model defined by [Axel Jantsch]
Cometa: Behavioral concern

Cometa model for CSP
Cometa: Communication concern
Abstract data types can be refined to current known data types (Integer, String, etc).

Specific Data representations such as Signals or BitVectors

In this representation refinement is used to express more specific data types
Cometa: Time concern

- Identification of general concepts for time expression.
A domain is applied to each semantic container (MoCComponent, MoCPort, MoCConnector)
Cometa: Translator

Methodology to describe translators based on:
- Input MoC semantics to Output MoC semantics
- Classification of Models of Computation (E. Lee, S. Vincentelli)
Translation: MoC Classification

- From less restrictive to more restrictive
- MoC can be classified according to their level of permissiveness

[Ed. Lee S. Vincentelli]
Experiments

- Experiments to represent semantics such as CSP, KPN.

- Definition of an execution semantic for an UML model with Cometa

- Execution with 2 UML tools having different execution environment:
  - Rhapsody (DE based execution environment)
  - TopCased/UML with a java implementation of the framework
Experiments

UML Metamodel

Cometa Metamodel

UML model

Cometa semantics

Library of UML semantic models

Library of Cometa model

Rhapsody
UML model/Rhapsody
Cometa Semantics
Execution Environment

TOPCASED
UML model/ Topcased
Cometa semantics
JVM + Scheduling of elements
Experimentation: portability

Application1

Application2

KPN Connector: KPN Connector

It's Mocc Connector

KPN Connector Interface in

KPN Connector Interface out

KPN Port Reader Interface

KPN Connector Interface in

CSP Connector Interface out

KPN Port Reader Interface

CSP Connector Interface in

CSP Port Writer Interface

CSP Connector Interface out

Cometa Execution
Experimentation: Simulation
Conclusion

• Conclusion
  • MoC properties enrich models with semantics at different level of abstraction
  • Explicitly defining semantics add flexibility to target different execution environments (portability)
  • Composition of MoCs is still difficult to handle due to semantic constraints and number of MoCs possibly definable

• Future Works
  • Improving Cometa Metamodel
  • Model transformation to:
    • TimeSquare Framework for time aspects
    • SPEAR (Thales RT) an Array-OL based Framework – iFEST Project
    • iFEST project (Radar Model/ tool chain Rhapsody-SPEAR)
Thank you for your attention!