Advanced Tool Architectures

Edited and Presented by
Edward A. Lee, Co-PI
UC Berkeley

Tool Projects

- Concurrent model-based design
  - Giotto (Henzinger)
  - E machine & S machine (Henzinger)
  - NP-Click (Keutzer)
  - Streambit (Bodik)
  - Metropolis (Sangiovanni-Vincentelli)
  - Ptolemy II (Lee)

- Meta modeling
  - GME (Sztipanovits, Vanderbilt)
  - GREAT=Language,Engine,C/G,Debugger (Karsai, Vanderbilt)
  - MOF-based Metamodeling (Sztipanovits, Vanderbilt)

- Verification
  - Blast (Henzinger)
  - CCured (Necula)
  - Chic (Henzinger)
  - SMoLES (Karsai, Vanderbilt)
Emphasis of Each Tool

<table>
<thead>
<tr>
<th>Concurrent model-based design</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Giotto</td>
<td>O/S neutral run-time virtual machine</td>
</tr>
<tr>
<td>E/S machines</td>
<td>Programming model for network processors</td>
</tr>
<tr>
<td>NP-Click</td>
<td>Domain-specific language for bit stream processing</td>
</tr>
<tr>
<td>Streambit</td>
<td>Design refinement and mapping.</td>
</tr>
<tr>
<td>Metropolis</td>
<td>MoCs and visualization of design.</td>
</tr>
<tr>
<td>Ptolemy II</td>
<td>Checking interface compatibility in component compositions.</td>
</tr>
</tbody>
</table>

UML-OCL/GME | Modeling the modeling languages using UML class diagrams and OCL
GREAT/GME, GRE/C/G | Modeling Model transformations using the GREAT Language, Transformation engine, Code Generator, Debugger
MOF/GME | Metamodeling using the Meta-Object Facility
Blast | Model checking C programs.
CCured | Making C programs more reliable.
SMOLES | Simple Modeling Language for Embedded Systems with Timing Analysis (using Timed Automata)
Chic | Checking interface compatibility in component compositions.

A Common Approach in Many of These: Actor-Oriented Design

Object orientation:

<table>
<thead>
<tr>
<th>class name</th>
<th>data</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
<td></td>
<td>return</td>
</tr>
</tbody>
</table>

What flows through an object is sequential control

Actor orientation:

<table>
<thead>
<tr>
<th>actor name</th>
<th>data (state)</th>
<th>parameters</th>
<th>ports</th>
</tr>
</thead>
</table>

What flows through an object is streams of data

A key theme is to identify the fundamentals of actor-oriented design and to show how to leverage a solid actor-oriented foundation to create domain-specific modeling languages (DSMLs)
Actor-Oriented Platforms

Actor oriented models compose concurrent components according to a model of computation.

Time and concurrency become key parts of the programming model.

Actor Interfaces: Ports and Parameters

Example:

- **input ports**: $p_1$, $p_2$, $p_3$
- **output port**:

Parameters:
- $a_1 = \text{value}$
- $a_2 = \text{value}$
Our Actor-Oriented Domain-Specific Modeling Languages: DSMLs

- **Giotto**
  - time-triggered
  - hard-real-time periodic tasks
  - deterministic mode switching
- **NP-Click**
  - push/pull semantics
  - combines application modeling language with target architecture abstraction
  - targeted towards network processors
- **HyVisual**
  - continuous-time semantics with mode changes
  - intended for hybrid systems modeling
- **GME-generated languages (Vanderbilt)**
  - meta modeling
  - synthesis of domain-specific visual languages

Giotto and xGiotto: Time-Driven Languages

- **Giotto**
  - Periodic hard-real-time tasks with precise mode changes.
- **xGiotto**
  - Extended to support event triggering
  - Deterministic task interaction.

- Giotto compiler targets the E Machine/S Machine
- Giotto model of computation also implemented in Ptolemy II

Chess Review, May 10, 2004
Giotto with a Visual Syntax

The Giotto Director in Ptolemy II gives the diagram Giotto semantics.

tasks defined using another MoC

Distributed E and S Machines

A major challenge with embedded real-time systems is preserving predictability in distributed implementations. The E & S machine platforms provide such predictability.
NP-Click: A Programming Model for Network Processors

- Concurrent composition of elements for packet processing
- Elements communicate via ports that pass packets
  - push: initiated by source element
  - pull: initiated by sink element
- Salient features of target architecture
  - threads
  - data layout
  - arbitration of shared resources

Click Router Application

The Click director in Ptolemy II gives the diagram push/pull semantics.
StreamBit: Sketching bitstream programs

• What is StreamBit?
  - language for developing bit-stream programs (e.g., crypto)
  - but mainly, a case study in programming by sketching

• What is programming by sketching?
  - separate Specification and Implementation
  - Specification (correctness): an SDF-based DSL
  - Implementation (performance): implementation strategy is only sketched; the compiler will derive the missing details

• What does this mean for the programmer?
  - First, program without concern for performance
  - Next, experiment with implementations w/out introducing bugs

• Sketching used to implement DES:
  - a brief sketch achieved 600% performance increase over naïve implementation (within \( \frac{3}{4} \) of a hand-tuned library)

Frameworks

• Metropolis
  - Unified model of computation (MoC)
  - Formal semantics given w.r.t. this MoC
  - Emphasis on joint modeling of hardware architecture and application
  - Modeling of design constraints and assertions

• Ptolemy II
  - Framework for experimentation with MoCs
  - Many MoCs have been implemented
    • discrete-event, continuous-time, dataflow, push/pull, process networks, CSP, FSMs, ...
  - Emphasis on heterogeneous mixing of domain-specific MoCs.
Joint modeling of function, constraints, and implementation platform.

Synthesis/Refinement
- Compile-time scheduling of concurrency
- Communication-driven hardware synthesis
- Protocol interface generation

Analysis/Verification
- Static timing analysis of reactive systems
- Invariant analysis of sequential programs
- Refinement verification
- Formal verification of embedded software

Metropolis Framework
- Design methodology
- Meta model of computation
- Base tools
  - Design imports
  - Meta-model compiler
  - Simulation

Metropolis Infrastructure
- Design methodology
- Meta model of computation
  - Architecture (Platform)
- Specification

Ptolemy II Framework
- Director from a library defines component interaction semantics
- Large, domain-polymorphic component library.
- Hierarchical components
- Visual editor

Type system

Basic Ptolemy II infrastructure:
**Hierarchical Heterogeneity**

Hierarchy allows distinct models of computation at distinct levels of the hierarchy. An "abstract semantics" defines the interaction across levels of hierarchy where the semantics differ.

**Actor-Oriented Classes**

- Objective is to bring modern modularity mechanisms to actor-oriented design:
  - classes
  - subclasses
  - inheritance
  - interfaces
  - subtypes
  - aspects
- Early, highly experimental prototype in Ptolemy II includes:
  - classes, inheritance, inner classes
  - interactive editing of class relationships
Example Using Actor-Oriented Classes

This model illustrates the mechanisms in Ptolemy II for defining classes and subclasses with inheritance.

Meta Modeling
Specifying and Integrating Tool Chains

Meta-Models
Transformation Meta-Models
Valid Meta-Models
Model Transformation Tools

DSML Meta-Modeling
Transformation Meta-Modeling
Validation & Verification
Meta-Generators
Domain-Specific Tool Chains

Modeling
Controller Synthesis
System Analysis
Code Generation
Validation Verification
Target Analysis
Platform

Domain-Specific Design Flows and Tool Chains

Ptolemy II
GME
DESERT

GME
Ptolemy II
Ptolemy II

DESERT

CCured

Blast

E/S machines
O/S
Processor
Platform Models
Valid Code/Model
Valid Code/Model
Valid Model
Valid Model
Valid Model
Valid Model
Valid Model
Download

Chess Review, May 10, 2004 19
**GReAT: Model Transformation Tool Chain**

Tools: UMT Language, GRE (engine), C/G, GR-DEBUG

---

**Verification**

- **Blast**
  - Scalable verification through lazy refinement of the state space.

- **CCured**
  - Static analysis of C programs for type safety.
  - Where static analysis fails, run-time checks are added automatically.

- **Chic**
  - Expresses assumptions made by module about environment, and guarantees made by module if assumptions are satisfied.
  - Checks compatibility of interfaces of composed components.

Both based on **CIL** (C Intermediate Language), a representation of C code along with a set of tools that permit easy analysis and source-to-source transformation.

---

http://www.isis.vanderbilt.edu
Many In-Depth Talks have a Tools Element

10:10-11:10 a.m.

- Event-driven Real-Time Programming (Arkadeb Ghosal and Tom Henzinger)
- A Comparison of Network Processor Programming Environments (William Plishker and Kurt Keutzer)
- Classes and Inheritance in Actor-Oriented Models (Stephen Neuendorffer and Edward A. Lee)

2:00-3:00 p.m.

- Metropolis, an Environment for System-level Design (Abhijit Davare and Alberto Sangiovanni-Vincentelli)
- StreamBit: Sketching Implementations for Bitstream Programs (Armando Solar-Lezama and Ras Bodik)
- Verifying Data Structure Invariants in Device Drivers (Scott McPeak and George Necula)

Many Posters have a Tools Element

- Wireless Sensor Network Design Methodology
- galsC: A Language for Event-Driven Embedded Systems
- Metamodelling Infrastructure for Model-Integrated Computing
- Distributed Diagnosis Algorithm
- Multiple Aspect Modeling Front-End for SIGNAL
- Online supervisory control
- Platform-based Design for Mixed Analog-Digital Designs
- Distributing Giotto
- Hierarchical Reconfiguration of Dataflow Models
- Fault Tolerant Design of Distributed Automotive Systems
- Modeling of Sensor Nets in Ptolemy II