Model-based Design and Implementation of Distributed Real-time Embedded Systems in Cadena

http://www.cis.ksu.edu/cadena

Gurdip Singh
John Hatcliff
Matt Dwyer

Venkatesh Ranganath
Xianghua Deng
Prashant Shanti Kumar
Qiang Deng

Support
US Army Research Office (ARO)
US National Science Foundation (NSF)
US National Aeronautics and Space Agency (NASA)
US Department of Defense
Advanced Research Projects Agency (DARPA)

Rockwell-Collins ATC
Honeywell Technology Center and NASA Langley
Sun Microsystems
Intel
Distributed Components

Java

C++

C

Network

C

Java

C++
Distributed Components

Middleware (e.g. CORBA)

Event Service
Transaction Service
Naming Service
Synchronization Service
**Real-time Embedded Systems**

**Issues**

- These systems are huge!
- Extensive use of OO patterns & software layering
- What are appropriate abstractions for formal reasoning?
- How can we help developers write them?
- How to auto-generate aspects?
- How can we configure middleware?
Example DRE Systems

- Mission-control software for Boeing military aircraft
- Boeing’s Bold Stroke Avionics Middleware
  - built on top of ACE/TAO RT CORBA

- Provided with an Open Experimental Platform (OEP) from Boeing
  - a sanitized version of the real system
  - 100,000+ lines of C++ code (including RT CORBA middleware)
  - 50+ page document that outline development process and describe challenge problems

- Must provide...
  - tool-based solutions that can be applied by Boeing research team to realistic systems
  - solutions that fit within current development process, code base, etc.
  - metrics for that allow Boeing research team to evaluate tool performance and ease of use
Cadena - CCM Development

High-level specification of abstract component behavior

Visualization and design-level reasoning

CCM Interface Definition Language
RT Aspect Specs
State Transitions
System Configuration

Eclipse Plug-In

Integrated Development Environment

High-level Specification Language

Configuration and Deployment Information

Code generation functions (via OpenCCM) produces code amenable to conformance checking and certification
Theme ...
COTS/Industry standard component models/frameworks...
Component developer’s activities (interface & behavior specs) are centered at the CCM IDL level.
Overall Themes

DRE Systems in a CCM Framework

Themes

Component Development

Scenario Specification

Scenario Analysis

Scenario Deployment

Middleware Library

.. Specify and reason about component connections and dependencies independently of middleware services
DRE Systems in a CCM Framework

Middleware Library

Component Development

Scenario Specification

Scenario Analysis

Scenario Deployment

Themes

...Choose particular middleware services and synthesize QoS attributes.

...Check invariants, dependencies, and temporal properties.
..Autogenerate configuration and component container code for chosen middleware services.

..Deploy based on deployment parameters.
Outline

1. Component Interface
   - Interface (Data) Ports
     - facet (interface provided)
     - receptacle (interface used)
   - Event Ports
     - event
   - Event Correlation
     - correlation

2. Component Connections
   - BM.PushDataSource
     - Navigator
   - BM.Passive
     - NavSteeringPoints
   - BM.Mode12
     - NavSteering
       - disabled-enabled
     - BM.Display
       - Display
     - BM.ModeSource
       - PilotControl

3. Dependence Information
   - Timer[5]
   - Timer[20]
   - Timer[1]
Component IDL

CORBA 3
CCM IDL
ModalSP Components
Component IDL

```
component BMLazyActive {
    provides  ReadData outData;
    uses      ReadData inData;
    publishes DataAvailable outDataAvailable;
    consumes  DataAvailable inDataAvailable;
    attribute LazyActiveMode dataStatus;
};
```
Component IDL

```plaintext
component BMLazyActive {
  provides ReadData outData;
  uses ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

output data port (facet)

CORBA 3
CCM IDL
ModalSP Components
Component IDL

```java
component BMLazyActive {
  provides ReadData outData;
  uses ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

CORBA 3
CCM IDL
ModalSP Components
Leverage CORBA IDL

component BMLazyActive {
    provides ReadData outData;
    uses ReadData inData;
    publishes DataAvailable outDataAvailable;
    consumes DataAvailable inDataAvailable;
    attribute LazyActiveMode dataStatus;
};

dependencydefault
== none;== none;== none;== none;
dependencies {
    inDataAvailable
    outDataAvailable
};

behavior {
    if (mode==enabled) {
        push outDataAvailable;
    } else {
        ...
    }
}

Component Implementation
Stubs & Skeletons

IDL Compiler

Model Builder

Dependency Analysis
and Model-checking Engine

Dependency Annotations
Transition System Semantics
Outline

2. Component Connections

- Interface (Data) Ports
  - facet (interface provided)
  - receptacle (interface used)

- Event Ports

- Event Correlation

- timer[5]

- timer[20]

- BM.PushDataSource: Navigator
- BM.Passive: NavSteeringPoints
- BM.Model2: NavSteering
  - disabled
  - enabled
- BM.Device: GPS
- BM.Legacy: AirFrame
  - state: crash
- BM.Display: Display
- BM.ModeSource: PilotControl
- BM.Model1: TacticalSteering
  - enabled
  - disabled
Three Synchronized Views

Scenario Description

Graphical View

Spreadsheet View

Textual View

Single Internal Representation
Instance AirFrame implements BMLazyActive on Board2 {
    connect this.inDataAvailable to GPS.outDataAvailable;
    connect this.dataIn to GPS.dataOut;
}

Instance EventChannel implements EventChannel on Board1 {
    Instance GSM implements BMDevice on Board1 {
        connect this.timeOut to EventChannel.timeOut20 atRate 20;
    }
    Instance AirFrame implements BMLazyActive on Board2 {
        connect this.inDataAvailable to GPS.outDataAvailable;
        connect this.dataIn to GPS.dataOut;
    }
    Instance DisplayCorrelator implements Correlator on Board3 {
        connect this.outDataAvailable to DisplayCorrelator.outDataAvailable;
        connect this.outDataAvailable to TacticalSteering.outDataAvailable;
    }
    Instance Display implements BMDisplay on Board3 {
        connect this.dataIn to AirFrame.dataOut;
        connect this.dataIn to TacticalSteering.dataOut;
    }
    Instance PilotControl implements BMModeSource on Board4 {
        connect this.timeOut to EventChannel.timeOut1 atRate 1;
        connect this.modeToggle1 to TacticalSteering.modeChange;
    }
    InstanceNavSteering implements BMModeSource on Board5 {
    }
Graphical View
### Spreadsheet View

#### Ports for Component Type
- **timeOut** < ::modalsp::TimeOut, 20, < timeOut20, EventChannel
- **outDataAvailable** > ::modalsp::DataAvailable, 20, > inDataAvailable, AirFrame
- **dataOut** > ::modalsp::ReadData, 0, > dataIn, AirFrame

#### Port Types

#### RT Attributes

#### Port Connections

---

#### Distribution Sites

#### Rate Group
Aspects Synthesis

- Analysis algorithms
- Synthesis of QoS and real-time aspects
Aspect Synthesis

Dependency-driven rate assignment to event consumers
Automatic detection of optimization opportunities

Asynchronous message delivery to synchronous method calls (must be co-located and run at same rate)
Spreadsheet View

Results of automatic rate group synthesis are fed back into spreadsheet
Mode-based Projections

Scenario Diagram w/ Complete Connectivity

NavSteering enabled
TacticalSteering disabled

NavSteering disabled
TacticalSteering enabled

Enabled Connectivity for Different Modes
Mode-based Projections

Multiple mode-based views are automatically created and synchronized through the design process.

...possible values for mode variables
Model-Checking Infrastructure

- Infrastructure dedicated to event-based middleware
- Complete model derived from CCM IDL, component-assembly description and annotations
- Model-checking for global system properties
Complete Model

System Components

Transition System Specs

RT CORBA Event Channel

Configuration Info
Leverage CORBA IDL

component BMLazyActive {
  provides ReadData outData;
  uses ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};

dependencydefault == none;
dependencies {
inDataAvailable outDataAvailable ;
};

behavior {
  if (mode==enabled) {
    push outDataAvailable;
  } else {
    ...
  }
}

Dependency Annotations
Transition System Semantics

IDL Compiler
Component Implementation Stubs & Skeletons
Model Builder
Dependency Analysis and Model-checking Engine
Incremental Specification

Specifications

- port action dependencies
- state-based dependencies
- component transition semantics

Component Structure

- ...only in mode Y
- ...state machines give abstract behavior

Increasing Effort & Strength of Verification
Code generation and Configuration

- OpenCCM CCM IDL to Java compiler to component stub and skeleton code generation
- XML files to container and middleware configuration
Extending COTS OpenCCM system to include different middleware services
Experiments: Mode-aware Event Service

Modal Scenario (modified MediumSP) → Scenario Analysis → Mode information → Mode-aware Event service → Configured Event Service

- BMDevice MonolithicImpl
  - BMDeviceCCM (context)
  - BMDevice MonolithicWrapper
  - HomeCCM
  - CCMObjectImpl

Synch_Module
Conclusions

- Cadena: model-driven design and implementation of DRE systems
- Light-weight specifications
- Component assembly description forms
- Dependency analysis tools
- Model-check infrastructure
- Code generation and configuration
Project Web Site

http://www.cis.ksu.edu/cadena