Model-Integrated Computing for Composition of Complex QoS Applications

Sandeep Neema,
Ted Bapty, Jeff Gray, Aniruddha Gokhale
Institute for Software Integrated Systems
Vanderbilt University
http://www.isis.vanderbilt.edu/projects/pces
Roadmap

• Model Integrated Computing (MIC)
• Motivation
• Adaptive QoS Modeling Environment
• Simulation of QoS adaptation
• Synthesis of QoS adaptation
• Case-study (BBN UAV application)
Model Integrated Computing (MIC)

Resulting from 15 years of research on computer-based systems in aerospace, instrumentation, manufacturing and robotics.

Common challenges:

- “Software” and “environment” are inseparable
- Need for adaptability to changing environment and end-user needs
- Complex, heterogeneous applications
- Stringent reliability and dependability requirements
Model Integrated Computing (MIC)

- Represent the information that directly or indirectly determines the structure of computations using domain-specific modeling paradigms

- Capture the relationship between domain models, analysis models and executable models

- Validate, analyze the domain models using generic tools

- Synthesize computations from the domain models
MultiGraph Architecture

**Meta-modeling Interface**
- Formal Specifications

**Meta-Level Translation**

**DSDE**
- DSME
- Models
- Model Interpreters

**Application Domain**
- App. 1
- App. 2
- App. 3

**Model Interpretation**

- Environment Evolution
- Application Evolution

DSDE: Domain-Specific Design Environments
DSME: Domain-Specific Modeling Environments
Typical System Development

End-users Specify their systems

Specifications

Specs, Drawings, Diagrams, Equations etc.

Applications

Simulation and/or analysis data

Domain-specific software apps

Configuration scripts

Programmers Implement Code
The MultiGraph Solution

Specifications

Graphical Modeling Environment

Synthesis

Model Interpreter

Applications

Simulation and/or analysis data

Domain-specific software apps

Configuration scripts

End-users model their systems

Application Evolution
Motivation

- RT Middleware solutions enable highly-complex Distributed Real-time Embedded (DRE) systems
- QoS adaptive frameworks (QuO*) enable abstraction of QoS parameters and QoS adaptation
- Low-level of abstraction for specification of adaptation
  - Difficult to validate/verify
- System behavior not apparent in specification of adaptation
  - QoS adaptation software is equivalent to a discrete controller for a highly non-linear system
  - Could result in instability
  - Need to simulate/verify
- Need: Specifications of behavior separated from software concerns

*QuO: Quality Objects (BBN Tech.)
MIC Approach

• Separation of problem aspects
  – QoS Specs
  – Behavior
  – Middleware Interfaces
  – Application QoS parameters

• Generation of QoS adaptation code
  – Contract Definition Language (CDL) output

• Generation of Simulations
  – Verify behavior over points in the envelope of operation
  – Using same models

• Analysis of system
  – Model Checking over full operational envelope, using same models
Design Flow

Adaptive QoS Modeling Environment (AQME)

End-users model their systems in AQME

Simulations are generated from models

Inputs for model checkers is generated

CDL output is generated when simulation and model checking results are satisfactory

QoS adaptive DRE system is deployed

Matlab Simulink/Stateflow Simulation

Symbolic Model Verifier (SMV) Model Checking

BBN Contract Definition Language

Model Interpreter

= Model Interpreter
Adaptive QoS Modeling Environment (AQME)

- QoS Specification and Adaptation modeling
  - Hierarchical, parallel, finite state machine representation
  - States capture system-wide QoS configurations
  - Transitions represent cause and effect of change in operating conditions
  - Data/Event variables represent the interface to the operating environment
Adaptive QoS Modeling Environment (AQME)

- **Software modeling**
  - Hierarchical dataflow representation
    - Compounds, primitives
  - Parameters for component instrumentation and customization
- **Middleware modeling**
  - Services and system condition objects
  - Parameters for middleware instrumentation and customization
Simulation of QoS adaptation

- Matlab® Simulink®/Stateflow® used as the simulation engine
- Model interpreters generate Matlab M-code to construct Simulink/Stateflow representation
- User provides a network simulation model, that can simulate various load conditions in the network
QuO Contract Generation

- AQME models translated to QuO contracts represented in Contract Definition Language (CDL)
- Hierarchical concurrent state machine models translated to flat state machine representation
- Transition guards and actions expressions over data/event variables translated to calls over Middleware (syscond) objects
Verification of QoS adaptation

- Symbolic Model Verifier (SMV) planned to be used as the model checking engine
- AQME models translated to SMV input specifications
- Can verify properties like reachability, liveness, fairness, etc. of the QoS adaptation specifications

Diagram:

1. UAV Model
2. SMV specification
3. SMV
4. Valid?
5. Counter example
6. yes
7. no
8. reiterate
Case Study: Dataflow Model

- Dataflow model of the UAV video-streaming application
- Sender, Distributor, and Receiver, → distributed application components
- ActualFrameRate, TimeInRegion, … → middleware syscond (parameter) objects
- ResvWithDropping → reference the adaptation behavior
Case Study: Adaptation Model

- NormalLoad, HighLoad, and ExcessLoad → states within the QoS parameter space
- actualFrameRate, frameRate, ... → data variables within state machine interface with the syscond objects

- Duty, Test → sub-states of HighLoad state
- Transition from duty to test enabled when time in duty exceeds 30 ticks
- frameRate variable set to 30 when transitioning from duty to test state
Case Study: Simulation

- Simulation models in Matlab Simulink → automatically generated from AQME models
- Matlab Stateflow model generated from QoS adaptation model in AQME
- Manually created Network model in Simulink captures the transfer function of the distributed network
Case Study: Simulation Results

- Results depict variation in the desired frame rate and the actual frame rate observed on the simulated network.
- Results are obtained against a simulated sinusoidal loading profile on the network.
- Results can be used to fine-tune the adaptation policy against different loading profiles (bursty, sharp transients, gradual persistent, ...).
- Benefit: Rapid prototyping and visualization of the responsiveness and behavior of the QoS adaptation policy.
contract UAVsplitContract (  
callback UAVCallbacks::Sender_Control_Callback senderControl,  
sysocond nowatch quo::ValueSC quo_sc::ValueSCImpl currentRegion,  
sysocond nowatch quo::ValueSC quo_sc::ValueSCImpl negotiatedFrameRate,  
sysocond nowatch quo::ValueSC quo_sc::ValueSCImpl actualFrameRate,  
sysocond quo::ValueSC quo_sc::ValueSCImpl timeInRegion )  
{
  state state_6 (  
    (((long) timeInRegion ) >= 3) -> state_5,  
    (((long) actualFrameRate ) < 27 and (((long) actualFrameRate ) >= 8) -> state_3,  
    (((long) actualFrameRate ) >= 27) -> state_2  
    )
  )
  ...  
transition state_5 -> state_6 {  
  synchronous  
  {  
    senderControl.setFrameRate( 10 );  
    negotiatedFrameRate.longValue( 10 );  
    timeInRegion.longValue( 0 );  
  }
  }
  ...  
};

• Complex CDL contract synthesized from models
• CDL output is post-processed by the BBN QuoGen compiler to generate code for the BBN QuO adaptation framework
• Generated Adaptation code monitors the QoS state of the distributed real-time system, and execute transitions and actions to bring the system in a desired operational state
Conclusions

- MIC enables better design and synthesis of highly complex QoS adaptation contracts by providing a domain-specific, higher level of abstraction.
- The approach shortens the design, implement, test, and iterate cycle by providing early simulation, and analysis capabilities.
- The approach facilitates change maintenance as minimal changes in the models effect large changes in the low-level (textual, code-base) CDL specification.
Future Work

- Enhancement of the middleware and dataflow modeling aspects of the AQME
- Investigating results from control-theory for synthesizing adaptation behaviors
- Extending the QoS adaptation modeling from a single centralized adaptation strategy, to several distributed local adaptation strategies, and their coordination and synchronization