Applying Multiple Modeling Languages to Large Scale Real-Time Systems Development

Sandeep Neema, Ted Bapty
Shweta Shetty, Steve Nordstrom, Di Yao, Shikha Ahuja
Vanderbilt Univ.
Outline

• Problem Domain
• Motivation/Challenges
• Proposed Solution
• Conclusions/Future Work
High Energy Physics

FermiLab Accelerator

BTeV Experiment
The Trigger

Forward tracker provides:
- Momentum measurement
- Pattern recognition for tracks born in decays downstream of vertex detector
- Projection of tracks into particle ID devices

Problem:
- Determine the set of particle trajectories
- Decide if it is interesting, keep or toss
- Massive amounts of data (Terabytes/Sec)
- Hardware => 2500 DSP’s + 2500 PC’s
- Never Fail (ok to degrade)
BTeV RTES Collaboration
NSF/ITR

- Fermilab
  - Building BTeV Trigger Hardware
  - Domain Experts, Define Goals, Constraints, etc.

- Vanderbilt
  - RTES Lead (Physics)
  - Design Environment, System Synthesis, System Integration, Prototype Hardware

- UIUC
  - ARMOR, Fault Tolerant Middleware
  - Domain Expertise

- Syracuse & Pitt
  - Very Lightweight Agents, Diagnostics
Design Issues

• Complex System
  – Thousands of Processors
  – High Data Rates
  – Real-Time Constraints

• User-Defined Behaviors
  – Domain-Specific Design Tool
  – System-Specific Implementation

• Run-Time Implementation
  – Heterogeneous Architecture
  – Real-Time - Execution & Mitigation
  – Fault-Tolerant
A First Cut MIC-based Solution

- Design a domain-specific modeling language
  - Provision concepts for all the different aspects of the system
  - Express their interactions
  - A “super” system-wide modeling language

- Implement a suite of translators
  - Generate fault-mitigation behaviors
  - Generate system build configurations
  - Link with user code/libraries
A First Cut Solution

Processing & Data Flow

Concepts:
Processes, streams, data channels, Functions, data types, communication

Hardware Resources

Concepts:
Processors, Memory, Topology, Reliability, Failure Modes,…

Hierarchical Fault Management

Concepts:
Recovery Strategies, Modes of Operation, goals/importance

Full
Recov. Mode 1
Recov. Mode 2
Recov. Mode 3
Problems/Challenges

• A single model file for the entire system
• Can not support multiple developers for different aspects
• A minor change in a small portion of the entire model
  – A completely new version
  – Need to regenerate the whole system
  – No partial validation due to the tight coupling
  – Extended build times due to unnecessary compiles
• Lack of modularity in design artifacts
• Where do the models fit in the versioning system?
  – What are the dependencies?
  – Should the generated code be archived in the build system?
  – Should the code-generators be part of the versioning system?
Outline

• Problem Domain

• Motivation/Challenges

• Proposed Solution

• Conclusions/Future Work
Learn from Textual Programming Techniques

- Modularization with multiple files for storing multiple artifacts
- "import" or "include" for referencing/external linkages
- Back-end tools "pre-processors", "linkers" for merging, validating linkages
“Include” applied to MIC

- Multiple “narrowly focused” domain-specific modeling languages
  - System Integration Modeling Language (SIML)
  - Data-Types Modeling Language (DTML)
  - GUI Config Modeling Language (GML)
  - Run Control Modeling Language (RCML)
  - Fault Mitigation Modeling Language (FMML)

- Allow expression of cross-linkages between models in different modeling languages
  - Overlapping concepts derived from a “Link” type
  - Attributes of “Link” type capture linkage specification i.e. type of linked object, location of linked object
“Include” applied to MIC (2)

- Plug-ins developed to facilitate link creation as well as link navigation
- Model interpreters, transformers evolved to navigate the links during synthesis
Data-Type Modeling Language (DTML)

• Modeling of Data Types and Structures
• GME supported sub-typing to model type hierarchies
• Configure marshalling-demarshalling interfaces for communication
Run Control Modeling Language (RCML)

- Modeling of Experiment Run Control
- System startup, shutdown, and run states,
- Transition thru System States and associated Actions
GUI Config Modeling Language (GML)

DataType Link
System Integration Modeling Language (SI ML)

- Model Component Hierarchy and Interactions
- Loosely specified model of computation
- Model information relevant for system configuration
SI ML Model

GUI Component Link
Versioning/Build System

- An equivalent XML representation of GME models is stored in the CVS tree
  - UdmCopy provides forward and backward translation from MGA to XML
- Model transformers developed with UDM
  - Can be built for Windows and Linux platforms
  - Transformer code is also stored in CVS
- Build system is hosted in Linux
  - Multi-stage build
    1. Compiles model transformers
    2. Makefiles invoke model transformers upon the stored models and generates code artifacts (behavior/config code)
    3. Generated codes are compiled, and linked to produce the binaries
Other solutions

• Model Library
  – Matlab Simulink
    • Model containers must specifically be constructed as libraries
  – GME
    • Libraries are inserted completely in the model
    • Updates are user demanded

• Multi-User Capability
  – GME
    • Domain-independent
    • Database backend
    • Stores models in separate files
Conclusions/Future Work

- Addressed scalability and versioning problems with MIC
- A prototype system in place experimenting with the developed concepts
- Good experience with respect to multi-user editing/maintenance of models
- Scalability concepts need to be generalized and formally structured to be applicable for a wider audience
- Consistency is weakly enforced