New Developments in Model-Integrated Development of High-Confidence Software

Joe Porter, Graham Hemingway, Nicholas Kottenstette, Harmon Nine, Chris vanBuskirk, Gabor Karsai, and Janos Sztipanovits

Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37205
Big Picture: High-Confidence Embedded Software Design

Layered Design Concerns

Our Goals

Certifiable implementations

- Model-based Design Flows (GME)
- Domain-Specificity (ESMoL)
- Support certification processes

Integrating verification and validation into tools

- Concurrency/Deadlock
- Schedulability
- Value domain issues

Exploring compositional techniques

- Emphasize correct-by-construction
- Passive control design
- Verification
- Scheduling
Big Picture: High-Confidence Embedded Software Design

Layered Design Concerns
- Plant Dynamics Models
- Controller Models
  - Controller design
- Software Architecture Models
- Software Component Code
- Software design
- System Architecture Models
- Resource Management Models
  - System Platform Design

AFOSR HCDDES MURI Team

https://wiki.isis.vanderbilt.edu/hcddes

Sponsored by the AFOSR MURI
FA9550-06-0312
and by NSF CPS contract
NSF-CCF-0820088
High-confidence development involves many activities in different domains of expertise.
Control designers create Simulink and Stateflow models to capture and simulate the physical behavior as well as the engineering design. Design verification takes the form of scripts to assess controller performance (e.g. stability, settling time, overshoot) and adjust controller gains.
The ESMoL domain-specific modeling language (DSML) includes a sublanguage which fully represents Simulink and Stateflow model structures. The tools include a fully automated model importer.
Software and hardware designers manually enter software designs in GME to describe the software architecture of the Simulink design models, network topology, and deployment of the software components to the hardware.
Workflow: Software Analysis

- **CONTROL DESIGN**
  - Simulink Simulation
  - Requirements

- **SOFTWARE IMPLEMENTATION**
  - Software Modeling (Arch/Deployment)

- **SOFTWARE ANALYSIS**
  - Scheduling
  - Esched Tool
  - BIP Spec
  - DFinder

- **GENERATION & EXECUTION**
  - Platform/HIL Simulation
  - Testing

**Model interpreters generate scheduling specifications and import results. BIP analysis is under development.**

**Round-trip analysis is integrated into the tool environment.**
Workflow: Generation & Execution

CONTROL DESIGN

SOFTWARE IMPLEMENTATION

SOFTWARE ANALYSIS

GENERATION & EXECUTION

Simulink Simulation

Requirements

Model interpreters synthesize C code for controller functions and for platform-specific task/messaging wrappers.

Software Modeling (Arch/Deployment)

ESMoL Modeling Language

Software Generator

Platform Design

A platform-independent time-triggered virtual machine provides a synchronous distributed execution environment.

Platform/HIL Simulation

Control Functions

Task/Msg Wrappers

FRODO VM

TrueTime (Simulink)

xPC Target (HIL)

Testing

TrueTime provides platform-specific simulation, and the xPC target enables hardware-in-the-loop.
Control designers can use the same tests to assess controller stability and performance, closing the loop on the design flow.

In the TrueTime and HIL execution environments we can measure the effects of platform uncertainty on controller performance.
Workflow: Constructive Methods

Constructive methods rely on component behavior conditions and interconnection rules to guarantee correct behavior, reducing verification burdens.

- CONTROL DESIGN
- SOFTWARE IMPLEMENTATION
- SOFTWARE ANALYSIS
- GENERATION & EXECUTION

PASSIVE CONTROL TECHNIQUES
- Simulink Simulation
- Requirements

SOFTWARE MODELING
- Software Modeling
- Platform Design
- Platform/HIL Simulation

SCHEDULING
- Scheduling
- Deadlock
- Testing

COMPOSITIONAL ANALYSIS

SYNCHRONOUS EXECUTION/TIME-TRIGGERED ARCHITECTURE

Constructive design assumptions must be maintained throughout the implementation process.
Quadrotor Control Architecture

Quadrotor Design Example

CONTROL DESIGN
Quadrotor: Simulink

Quadrotor

Inner Loop Controller (Attitude)

Outer Loop Controller (Inertial Position) and Trajectory Generator

CONTROL DESIGN
Quadrotor Software Design: GME & ESMoL

Software Deployment

Node Ports == Comm Channels

Processor

Bus

Platform Design Model

Block == Component Instance

Port == Message Instance

Logical Architecture (Dataflow)

SOFTWARE IMPLEMENTATION
Quadrotor Platform Simulation with TrueTime

SOFTWARE IMPLEMENTATION

Component Timing Parameters:
- TT Schedule – start times
- ExecPeriod
- WC Duration

SOFTWARE ANALYSIS

Schedule Spec

Calculated Schedule

GENERATION & EXECUTION

Time-Triggered Network

Time-Triggered Nodes

Generated Task Execution Code

ESMoL Model File (mga)

Generator

Scheduler Spec (.scs)

Scheduler

Scheduler Result (.rslt)

TrueTime Simulink Model (mdl)

Task Execution Code (C)
Quadrotor: TrueTime Execution and Schedule

- Time-triggered execution schedule for one Node
- Tracking performance is close to ideal...
- ...but platform effects are successfully simulated
- Primary execution loop is an embedded Time-Triggered scheduler
Fast quadrotor dynamics introduce a small amount of active behavior. Sector search relates control gain to an interval behavior bound abstraction.

Example: Quadrotor position tracking uses a passive PD controller, and we validate the position gain using sector search around the gain loop (Kx < -1/a).

\[ \|yT\|^2 - (a + b)(y, u)T + ab\|uT\|^2 \leq 0 \]

See Kottenstette [1] for details.
We extended the Simulink Quadrotor model with a random (Poisson-distributed in time and duration) sensor fault. When the fault occurs, all of the sensor outputs return zero.

Statistical model checking sees the system as a black box, so we used an LTL error condition on the measured trajectory. “For 500 seconds of the trace, it will never be true that the error of either x, y, or z will exceed the specified bound for more than 50 seconds.”

\[ \neg F[500] G[50] ( ex > 400 \mid ey > 400 \mid ez > 100 ) \]

How did we do? Well, it shouldn’t crash more than 10% of the time under these fault conditions...

Work In Progress:
Fixed Wing Aircraft Example

We wanted to try our hand at a more complex control design...

Multicore Experiment: Deploying multiple control loops to the Cell processor.

Questions to be answered:
- Design coverage
- Online stability assessment for backstepping designs
Work in Progress: Test and Tap

We want to compare simulated data traces to those collected from the platform for various test cases and tap points. Our prototype test generator automates data generation, execution, data collection, and comparison. It’s still fairly limited...

One problem with complex systems is assessing operational data values inside the system (tap the data streams). This is a job for... grad students!

Control Design
Software Analysis
Generation & Execution
Where would we like to go?

• Modeling Concepts
  – Requirements modeling and management
• Design examples
  – Coordinated flight
• Synthesis tools
  – Automated controller design (IDA-PBC for Port Hamiltonian systems)
  – Platform-specific function templates
• Modeling Semantics
  – Asynchronous execution models
  – Characterization of the interactions and conflicts between layered design domains
• Analysis
  – Reducing response time
  – Performance optimization
Questions?

Those interested in trying our tools can visit

https://wiki.isis.vanderbilt.edu/hcddes/index.php/The_ESMoL_Tool
References


