Frameworks and Tools for High-Confidence Design of Adaptive, Distributed Embedded Control Systems

- Project Overview -

Janos Sztipanovits
ISIS-Vanderbilt University

AFOSR Dynamics and Control Program Review
Arlington, VA
August 6, 2008
Team

- Vanderbilt
  - Sztipanovits (PI), Karsai, Volgyesi, Kottenstette Porter, Thibodeaux
- UC Berkeley
  - Tomlin (PI), Lee, Sastry, Gonzales, Zhou, Leung
- CMU
  - Krogh (PI), Clarke Jain, Lerda, Bhave, Maka
- Stanford
  - Boyd (PI) Skaf
Objectives

• Development of a theory of deep composition of hybrid control systems with attributes of computational and communication platforms
• Development of foundations for model-based software design for high-confidence, networked embedded systems applications.
• Composable tool architecture that enables tool reusability in domain-specific tool chains
• Experimental research

Long-Term PAYOFF:
Decrease the V&V cost of distributed embedded control systems
Overall Undertaking

Scope of the Project:
- Development of component technologies in selected areas
- Development of model-based design methods
- Incrementally building and refining a tool chain for an experimental domain (micro UAV control)
- Demonstration of control software development with the tool chain
- Experiments
Model-Based Design Challenges

- **Orthogonality among the design layers**
  - Controller design depends on assumptions about implementation
    - The overall design needs to be verified after implementation
    - Changes in any layer require re-verification

- **Decoupling the design layers is very hard**
  - E.g., Time Triggered Architecture
  - Timing and fault tolerance
  - Strict synchrony and static structure
How should we use implementation abstractions in controller design?

- Robust control design (Boyd, Krogh)
  Model precisely effects of implementation as uncertainties and make controller design robust

- Passive control (Kottenstette, Antsaklis, Koutsoukos)
  Develop theories and methods for decoupling key properties (such as stability) from effects of implementation.
Robust Control Design - Boyd

Robust control methods extended to implementation uncertainties. Result of the design is a nominal controller, $C^{\text{nom}}$, a Lyapunov Performance Certificate, $L$, (in LMI form) and an implementation complexity measure, $\phi$. (See Boyd and Skaf, Stanford)

Software architecture and system-level design is decoupled (subject to limits) from controller design and includes implementation platform specific abstractions.

Robust implementation platform designed for decoupling the two design layers.

Code and HW/Network configuration is decoupled (subject to limits) from system-level design and allows composition for other properties (reliability).
Example control design problem:

\[ x(t + 1) = Ax(t) + Bu(t), \quad x(0) = x_0 \]
\[ u(t) = Kx(t) \]

- The nominal controller \( K_{nom} \) is the LQR optimal feedback controller with double precision floating-point coefficients.
- Admissible controllers \( C \) are controller that yield an LQR cost that is, at most, 15% suboptimal.
- The complexity measure \( \Phi \) is the number of bits required to express \( K \).
- The best design, which is 14.9% suboptimal, gives only 1.5 bits/coefficients.
New Extensions

- Upper bound on performance degradation due to timing jitter

- Upper bound on performance degradation due to network delays
  Bhave and Krogh: “Performance bounds on State-Feedback Controllers with Network Delay,”
A passive system only stores and dissipates energy but cannot generate energy of its own.

When connected in either a parallel or negative feedback manner, the overall system remains passive.

Theory applies to linear and non-linear systems, continuous and discrete-time systems.

Provides robustness against time-varying delays and packet loss.

Verification and Testing

- How should we use implementation abstractions in controller design? (Boyd, Krogh, Kottenstette)
- How can we exploit heterogeneous abstractions in verification and test generation? (Krogh, Clarke)
Verification of Dynamics - Krogh

Timing in Networked Control Systems
(Bhave, Krogh)
- impact on performance and stability of feedback loops
- analytical and simulation tools

\[
\left| \frac{P_{\text{dec}(\omega)}C(\omega)}{1 + P_{\text{2OH}(\omega)}C(\omega)} \right| < \frac{1}{N|\omega| - 1}
\]

Verification of Numerical Code
(Maka, Freshe, Krogh)
- verification for target environment
- reachability with polyhedral domains
- widening for iterative computations

- Timing variations in Feedback Control Loops
  - Frequency domain stability analysis
  - Co-simulation using TrueTime Simulink Blockset (Lund)

- Verification of Numerical Code
  - Polyhedral reachability computations
  - Extension to nonlinear computations
Goal: Generate a test suite that is:
- complete
- sound
- minimal
Problem:
- Bisimulation relations between systems used to perform verification of a complex system using a simpler one
- Bisimulation functions used to define bisimulation relations for continuous dynamic systems, but there is no theory regarding when such functions exist

Our Approach:
- We address problem of identifying qualities of systems that permit analysis via bisimulation functions
  - Developed sufficient condition for nonexistence of bisimulation functions
Problem:
- Verification or exhaustive simulation expensive or impossible for many systems

Approach:
- Statistical Hypothesis Testing to estimate the probability that a stochastic system satisfies a property by looking at a sample of its behaviors
- Minimizes the number of simulations needed to guarantee probabilistic bounds on the estimation error
Statistical Model Checking of Hybrid Systems - Clarke

- Application to a Simulink model of a mixed (analog-digital) signal circuit:
  - Definition of a stochastic model adapted to the hybrid case
  - Specification of properties in the frequency domain

- Reference:
  - “Statistical Model Checking of Mixed-Signal Circuits” E.M. Clarke, A. Donzé and A. Legay. Formal Analysis of Circuits, satellite workshop of CAV’08
Model Checking Convex Hybrid Automata
- Clarke

Problem:
- Practical problems require Model Checking of hybrid systems is severely restricted.
  - Small number of continuous variables
  - Simple systems with linear dynamics
- Analysis of high dimensional nonlinear hybrid systems.

Solution:
- Define a suitable analyzable fragment of nonlinear hybrid automata.
- Use convex programming techniques for model checking.
- Can analyze systems 10 times larger than existing approaches.

Reference:
- “Path Oriented Reachability Analysis of Convex Hybrid Automata"
  Lei Bu, Edmund M. Clarke, Sumit Kumar Jha and Xuandong Li.
  (Submitted)
We Develop High-Confidence Code Generators

- How should we use implementation abstractions in controller design? (Boyd, Krogh)
- How can we exploit heterogeneous abstractions in verification and test generation? (Krogh, Clarke, Tomlin, Sastry, )
- How to design high-confidence code generators? (Lee, Karsai)
Model-Based Code Generation

Software architecture and system-level design is decoupled (subject to limits) from controller design and includes implementation platform specific abstractions.

Code and HW/Network configuration is decoupled (subject to limits) from system-level design and allows composition for other properties (reliability).
Principled Design of Embedded Software - Lee

Builds on the extension of the Ptolemy II framework:

- Build a “models to C” lab enabling experiments with
  - Models of concurrency and time
  - Optimization based on partial evaluation
- Create sampled data models and translation to C with
  - Polled I/O
  - Interrupt-driven I/O
- Create event-driven models and translation to C with
  - Model of time
  - Synthesized scheduling of reactions
- Created distributed timed models and translation to C
  - Host, supervisor, and controller interactions
  - Time synchronization

*Emphasis on repeatability and verifiability!*
The EmbeddedCActor in Ptolemy II wraps low-level functionality (written in C) to define an actor. This approach makes it easy to build actor-oriented models and to generate efficient, platform-specific C implementations.
This design of a hill-climbing control algorithm wraps code provided by iRobot as demo code into actors in Ptolemy II for accessing sensors and actuators.

"Helper" classes provide code generators for generic actors and FSM controllers.

"EmbeddedCActor" provides code generator to interface the C templates with the rest of the system.
Ptolemy II approach: partial evaluation for actor-oriented programs

- Model analysis
- Model (actor-oriented program)
- Execution context: data types, buffer sizes, schedules, parameters, model structure, etc.
- Partial evaluator (code generator)
- Highly optimized target code blocks
- Input
- Monolithic and efficient executable
- Output
- Target code execution
In collaborations with TU Munich, we are leveraging OpenArchitectureWare, an open source model-based code generator built on the Eclipse Modeling Framework (EMF) to build a Codegen 2.0.
MIC – based Code Generation - Karsai

Modeling/Simulation Environment (Simulink/Stateflow) → Mdl2Mga → Model Editing Environment (ECSL-DP)

- Dataflow
- Stateflow
- System

- Simulink Code Gen
- Stateflow Code Gen
- Scheduler Conf Gen

- C code
- C code
- TT Schedule Conf

Symbolic verification (TBD)
Code generation using graph transformations

The code generator is formally specified as a programmed graph transformation system. This allows reasoning about the correctness of the transformation itself.

Support for verification: The code generation could insert verification conditions (derived from the models) into the generated ASG.

The result of the transformation is an abstract syntax graph that allows ‘printing’ the executable code in various languages.

C source code
We Build Infrastructure for Reconfigurable Tool Chains

How should we use implementation abstractions in controller design? (Boyd, Krogh)
How can we exploit heterogeneous abstractions in verification and test generation? (Tomlin, Sastry, Clarke, Krogh)
How to design high-confidence code generators? (Lee, Karsai)
How can we design and customize model-based design flows? (Volgyesi, Karsai, Krogh, Lee, Sztipanovits)
Design rationale for prototype toolchain (1)

The connection towards Simulink/Stateflow

- Simulink/Stateflow is the industry standard
- SDF and (restricted) Statechart semantics is well-defined and widely used
- Could be substituted in later stages of the project

The ECSL language

- Software components and architectures and deployment had to be captured in models and integrated with the functional models.
- Not all features of Simulink/Stateflow are supported – only a ‘safe’ subset.
- Dataflow (Simulink/SDF) model: scheduling based on the time-triggered paradigm (t_k is determined by an off-line scheduler)
  - receive(t_k) → execute() → send(t_k+1)
- Extensible towards other models of computation
Platforms

**TTTech**
- MPC 555 micros
- TTP/C comm
- TTTech Software tools
- Fault-tolerance

**Soekris**
- Linux w/ 3xEthernet
- TT Virtual Machine on standard UDP and Linux
- No fault tolerance (yet)

**Gumstix**
We Evaluate Progress Experimentally

- How should we use implementation abstractions in controller design? (Boyd, Krogh)
- How can we exploit heterogeneous abstractions in verification and test generation? (Tomlin, Sastry, Clarke, Krogh)
- How to design high-confidence code generators? (Lee, Karsai)
- How can we design and customize model-based design flows? (Volgyesi, Karsai, Krogh, Lee, Sastry, Sztipanovits)
- How can we evaluate V&V methods experimentally? (Tomlin, Sastry)
Starmac Experimental Platform
Quadrotor aircraft developed by co-PI Claire Tomlin

Requires integration of legacy and custom components.
The project is working on the design of the software, controllers, and models for this platform.
Experimental Set Up

- A mobile sensor network:
  - A set of vehicles, each with a set of sensors for its own navigation and control, as well as for sensing its environment (such as target range or bearing)
  - Computation is distributed, and limited to the processors on board the vehicles (no central computer)
  - Communication between subsets of vehicles (limited by range or geography) available
  - Collision avoidance needed between vehicles
  - Humans share control with automation

- Focus on algorithms for autonomous search:
  - Unexploded ordinance detection
  - Beacon tracking scenarios
  - RFID tracking
  - Survey of disaster areas
  - Search and rescue
  - Biological studies, animal monitoring
Progress

- **Distributed estimation:**
  - Derived conditions under which a set of local Kalman filters with only local information, from nearest neighbors, estimating the state of the environment, converges to a global Kalman filter (with all information available)
  - Integrated automatic collision avoidance algorithms with local estimators
  - In the case of mobile camera networks, used discrete information about occlusions to automatically calibrate cameras
  - Development of Hydra: a networked simulation tool that allows searchers operating under different prior distributions to share control over a limited number of vehicles
Accomplishment Highlights

- Proved feasibility of methods and framework for decoupling (possibly imperfect) controller implementation from controller design/specification (Boyd).
- Developed model-based timing analysis for networked embedded systems, test generation for timed automata and model-based verification of numerical code (Krogh).
- Applied reachable set technologies to the analysis and design of collision avoidance schemes for multiple autonomous quadrotor aircraft, and to the very close formation flying of multiple fixed wing UAVs (Tomlin, Sastry).
- Analyzed the limits of approximation techniques for continuous image computation in model checking hybrid systems. Developed verification algorithms for MATLAB/Simulink models by combining SW model checking with numerical simulation tools. (Clarke)
- Developed model-based code generation algorithm using partial evaluation (Lee).
- Developed model-based code generation algorithm using model transformation (Karsai).
- Developed end-to-end model-based design tool chain prototype for TTP and RTAI Linux platform (Volgyesi, Karsai, Sztipanovits).
- Developed quadrotor UAV experimental platform with rapidly expanding distributed control functions (Tomlin, Sastry).
Transitioning

- Ptolemy II 7.0.1 was released on April 4, 2008. Ptolemy II includes the code generation facility. The Ptolemy source tree is available via CVS. We are actively working with Bosch and National Instruments. In addition we have: Assisted in the transfer of avionics code from B
  - Berkeley HCDDDES team provided consultation and research materials about the IEEE-1588 platform as a possible testbed. Prototyped a vhdl target for the code generation effort. Researched Hybrid Interchange formats and discussed these with researchers in Alberto Sangiovanni-Vincentelli's group and at Cadence Berkeley Labs. Discussed the design of Vanderbilt's code generation

- Vanderbilt’s MIC tool suite (GME, GReAT, UDM, OTIF) has two major releases during the last year. The releases are available through the ESCHER and ISIS download sites.
  - Vanderbilt continued working with GM, Raytheon and BAE Systems research groups on transitioning model-based design technologies into programs.
  - Vanderbilt continued working with Boeing’s FCS program on applying the MIC tools for precise architecture modeling and systems integration.
  - Collaboration with TTTech, University of Vienna.