VERTAF: An Application Framework for Design and Verification of Embedded Real-Time Software

Pao-Ann Hsiung, Shang-Wei Lin, Chih-Hao Tseng, Trong-Yen Lee, Jih-Ming Fu and Win-Bin See
Contents

- Introduction
- Design and Verification Flow
  - UML Modeling
  - Real-time Software Scheduling
  - Formal Verification
  - Component Mapping
  - Code generation
- VERTAF Components
- Experimental Results
- Conclusion
Introduction

Available applications
- poor integration of functional & non-functional requirements

New design
- accelerate the real-time embedded software construction
- component reuse, formal synthesis and formal verification
Designing an embedded system

- Model the classes required
- Generate code based on those models
- For real-time systems
  - temporal constraints
- Formal verification
  - performance
  - reliability
  - time constraints
Features of this Framework

- **Formal Modeling:**
  - Well-defined UML semantics

- **Formal Synthesis:**
  - Guarantees satisfaction of temporal & spatial constraints

- **Formal Verification:**
  - Checks if system satisfies user-given or system-defined generic properties

- **Code Generation:**
  - Produce efficient portable code
Software synthesis has two phases

- **Front-end phase (m/c independent)**
  -- UML modeling phase
  -- Scheduling phase
  -- Formal verification phase

- **Back-end phase (m/c dependent)**
  -- Component mapping phase
  -- Code generation phase
UML Modeling

Class Diagrams
- introduce the deployment relationship
- two types of classes
  → software classes
    --- specified from scratch by the designer
    --- reuse a component from the libraries
  → hardware classes
    --- Supported hardware component

![Class Diagram Example]
Timed Statecharts

- Method Types
  - event-triggered
  - time-triggered
    -- deadlines, period
    -- start, stop and restart
Extended Sequence Diagrams

- Used for scheduling different tasks performed by objects
- Show how a user should use the system
- Added state-markers:
  - They relate the sequence diagram to the corresponding state in the timed state chart
Scheduling

- Generate Petri nets from UML diagrams

Algorithms used

- Without RTOS (Quasi Dynamic Scheduling)
  -- Single real-time kernel
- With RTOS (Extended Quasi Static Scheduling)
  -- Schedule multiple threads

VERTAF uses simple RTPN/CCPN models for scheduling purposes.
Petri Nets

- Standard Petri Net (N) : \(<P,T,\emptyset>\)
- RTPN: \(<N,\chi,\pi>\)
  \(\pi\) – indicates period for RTPN
  \(\chi\) - maps transition to worst-case execution time and deadline
- Temporal constraints that appear in sequence diagrams
  - converted into guard constraints on arcs in generated Petri nets
Model based verification

- Static Analysis
  - more suitable (all possible executions)

- Model checking
  - if temporal property is satisfied
  - else show the counterexample

- Verification kernel used:
  - State Graph Manipulator (SGM)
    -- State-graph merger
    -- Dead state checker
    -- State-reduction techniques

- Properties verified
  - Dead states, deadlocks, livelocks
Component Mapping & Code Generation

- Automatically generate make files, header files etc.

Main Issue:
- when a software class is not deployed on any specific hardware component

Solution:
- display a list of available compatible device types to the user

Code generation (3-tier)
- hardware abstraction layer
- OS with middleware layer
- Scheduler
VERTAF Components
Experimental Results

Two Applications:
- Avionics
  -- 24 tasks
  -- 45 objects were found
- AICC (Autonomous Intelligent Cruise Controller)
  -- 12 tasks
  -- 21 objects were found

Time taken to develop
- Avionics
  -- Without VERTAF : 5 weeks
  -- Using VERTAF : 1 week
- AICC
  -- Without VERTAF : 20 days
  -- Using VERTAF : 5 days
Conclusions

- VERTAF integrates 3 different technologies
  - software component reuse
  - formal synthesis
  - formal verification
- New specification languages can be easily integrated into it.
- More advanced features like network delay, network protocols will be considered in future work.
Thank You