CBSE for ERTS

Barry R. Pekilis
PhD Candidate
ECE750-11: Component-based Software Systems
July 21, 2005

Presentation Outline

- Introduction
- Definitions & Concepts
- Design & Deployment Issues
- Implications for Components
- Development For ERTS
- Research Needs Overview
- Conclusions
Introduction

- COTS well established for electronic & mechanical components
- CBSE for Embedded Real-Time Systems must deal with:
  - timing issues
    - average execution time (AET)
    - worst-case execution time (WCET)
  - resource constraints (e.g., limited hardware capacity)
  - dependability
  - portability & mobility
  - demand for low production costs

What Is A Real-Time System?

- Real-Time (RT)
  - pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process (IEEE 1990)

- Real-Time System (RTS)
  - a system in which correctness depends not only on the logical result of the computations performed, but also on time factors (Stankovic 1998)
Real-Time Domain

- Requirements
  - behavioral
  - response time

- Interaction with environment
  - sensors (inputs)
  - actuators (outputs)

Real-Time Domain (cont’d)

- Quality of Service (QoS) levels
  - hard RTS
    - may never miss deadline e.g., avionic control
    - missed deadline = catastrophic failure
  - soft RTS
    - may miss deadline e.g., multimedia
    - missed deadline = wait or response performance failure
Real-Time Domain (cont’d)

- Resources
  - dedicated
    - always enough e.g., hard RT, safety critical
  - shared
    - limited availability e.g., soft RT, telephone switches

- Activation
  - event-triggered
    - asynchronous interrupts e.g., automated bank machine
  - time-triggered
    - synchronous schedule e.g., token ring network

What Are Embedded Systems?

- Embedded Computer System:
  - *a computer system that is part of a larger system and performs some of the requirements of that system (IEEE 1990)*
  - 99% of computing units today are ERTS (DARPA 2000)

- Embedded Real-Time System (ERTS):
  - part of larger system & interact with external devices
    - automobile control & safety
    - medical equipment
    - process control
    - consumer electronics
    - consumer appliances
ERTS Design Considerations

- Real-time requirements
- Resource consumption
  - CPU, memory, power, space, cost
- Dependability
  - safety, reliability, availability
- Life-cycle properties
  - maintainability, extensibility, platform portability
- Interoperability
  - e.g., bluetooth

Implications for ERTS Components
(I. Crnkovic)

- Contractually specified interfaces
- Unit of composition & independent deployment
- Explicit context dependencies
- Component granularity
- Reusability
- Architecture & framework
- Portability & platform independence
Contractually Specified Interfaces

- Nonfunctional specifications equally important as functional behavior
- Interface specification must include:
  - memory consumption
  - method WCET
  - expected power consumption under certain execution schedules
- Use of procedural interfaces both with & without polymorphism
  - avoid performance penalty due to polymorphism
    - (i.e., dynamic binding)

Unit of Composition & Independent Deployment

- Focus on design-time component composition
- Source code components rather than binary component
- Full model & environment support too expensive
  - strip away or degrade unnecessary functionality
- Configuration (static) composition at design time
  - capable of generating monolithic firmware from designs
  - optimization to use direct references & function calls
Explicit Context Dependencies

- Focus is on run-time environment (i.e., RTOS) dependencies & support
- Abandon programming language independence in return for higher, more predictable performance
- Source language components must be tailored for specific deployment environments

Component Granularity

- Finding right-sized components critical optimal design decision
- Fine-grained components limited to one device type or specific class of application
- Light-weight components should be pruned of unneeded functionality
- Components should be resource usage aware
Reusability

- Whitebox or greybox reuse impacts component substitution & requires partial implementation access
- Need clear conventions for knowledge about ERTS component implementation
  - capture knowledge in architecture (e.g., connectors)
  - composition scripts belonging to component
  - composition environment has implementation knowledge

Architecture & Framework

- Common architecture needed for family of ERTS
  - guide component usage & interactions
- Framework for family of ERTS needed
  - standard interfaces
  - architectural styles
  - connectors
  - extensibility
- Define points where framework open for new, independent components to be developed & plugged in
Portability & Platform Independence

- Micro-controller programming language specific compiler-extensions cause manual porting effort & version explosion
- Source level portability requires implementation language agreement between devices within a given class
- Need agreement on:
  - implementation language
  - available libraries
  - proper abstractions e.g., RTOS API

CBSE Development for ERTS

Component model
Component-based architecture for field devices
Component repository
Composition environment
Run-time environment
Component Technology For Embedded Devices
ERTS Component Model

- Nonfunctional properties & constraints
  - AET, WCET, memory consumption
- Efficient functional (procedural) interfaces
- Architectural styles describing connections & relations
- Code generation & controlled component adaptation
  - source language or generative components

ERTS Component-Based Architecture

- Framework for ERTS
  - standard interfaces
  - components
  - architectural styles
- Compile-time optimization abilities applied during target code preparations
ERTS Component Repository

- Storage & retrieval of components for analysis, design, implementation & composition
- Store components & architectural styles according to component model
  - interface descriptions
  - nonfunctional properties
  - scripts for composition
- Maintaining component versioning

ERTS Composition Environment

- Composition techniques supported
  - visual tools
  - scripts
- Definition of component composition rules
- Component composition rule checking
  - based on architectural style
  - verify component configuration meets specified constraints
- Component adaptation & application code generation
ERTS Run-Time Environment

- Efficient implementation model for components
- Constraints for specific ERTS-based devices
- Approach for compiling component-based design into optimal firmware for target device
- Hardware- & RTOS-independent implementation of components

CBSE for ERTS Research Needs

- Must develop widely adopted component technology standards
- Must optimize composed system for performance & consumption
- Must adequately specify & analyze extra-functional properties
  - composition theories for certain extra-functional properties
  - predictability theories of system properties from component properties
- Runtime platforms
  - must provide services that use limited resources
  - platform & vendor independence
- Tools
  - component certification
  - component noninterference
M. de Jonge, J. Muskens, M. Chaudron
Dept. of Math & CS, Eindhoven University of Technology, Netherlands

Scenario-Based Prediction Of Run-Time Resource Consumption In Component-Based Software Systems

- Resources typically expensive & not extensible
- Scenario-based resource predictions of consumed resources to reduce state explosion
  - analyze designs and/or implementations
- Component-based applications complicate resource predictions since consumption spread across components
- Need way to express resource consumption per component then compose results to do prediction
- Propose model for combining individual component resource estimations which are used in runtime scenarios to predict memory consumption of applications

A. Möller, J. Fröberg, M. Nolin
Mälardalen RT Research Centre (MTRC), Mälardalen University, Sweden

Industrial Requirements On Component Technologies For Embedded Systems

- Component-based technologies for ERTS in heavy vehicular systems
  - Volvo construction equipment
- Presents set of requirements based on industrial needs aimed at evaluating existing component technologies
- Results find that for component technology to become valuable in an industrial context:
  - must address CBSE impact on overall development process & life cycle management
  - must support ability to gradually migrate into new technologies
K. Sandström, J. Fredriksson, M. Åkerholm
Mälardalen RT Research Centre (MTRC), Mälardalen University, Sweden
Introducing A Component Technology For Safety Critical Embedded Real-Time Systems

- Focus on development of component model
- Low footprint systems with very high demands on safe & reliable behavior
- Key concept
  - provide expressive design time models for effective resource usage & timing through powerful compile time techniques
- Results in a component technology for resource effective & temporally verified mapping of a component model to a commercial real-time operating system

Scuola Superiore Sant'Anna
A Hierarchical Framework For Component-Based Real-Time Systems

- Describes a methodology for design & development of component-based RTS using dataflow paradigm
- Component modeled as set of concurrent runtime threads that communicate by means of synchronized operations
- Each component specifies own scheduling algorithm
- Support at the O/S level is discussed
- Implementation presented in the Shark RTOS
W. Maydl
University of Passau, Germany
Design Accompanying Analysis Of Component-Based Embedded Software

- Analysis techniques to detect early design errors
- Components models as functions on streams of signal data which allows describing behavior of dataflow components precisely by constraints
- Static constraints – may be complex multi-variable polynomials enforced by a new interface type system
- Dynamic constraints – checked using a novel model checking technique based on FIFO automata
- Component model is compositional resulting in well-defined hierarchical abstraction

University of Passau, Germany
Dept. of Languages & Computer Science, University of Malaga, Spain

- New predictable component model for ERTS
- Set of tools to support proposed component model
- Environment for real-time analysis at component & application level using:
  - component meta-information
  - abstract behavioral model
  - WCET model
- Developer view specifies basic component elements for primitive & generic components plus how they interact using IDL from CORBA
- User view deals with external definition of components including externally visible interfaces and configuration parameters
  - declares component instances, concrete configurations, interfaces, real-time constraints
- Analysis model abstracts away component behavior, leaving only details needed for RT analysis, based on real-time extensions to SDL
Robust Open Component Based Software Architecture for Configurable Devices Project (ROBOCOP)

- Funded by Information Technology for European Advancement (ITEA)
- Define component-based software architecture for middleware layer of high volume embedded appliances
- Consumer Appliances includes: mobile phone, set-top boxes, dvd-players & network gateways
- Consortium includes:
  - Switzerland: CSEM, SAIA – Burgess
  - Spain: ESI, FAGOR ELECTRODOMESTICOS, IKERLAN, Technical University of Madrid; Visual Tools
  - Finland: Nokia
  - The Netherlands: Philips; Technical University Eindhoven
  - France: Philips

Conclusions

- Dynamic component deployment infeasible for ERTS
  - predictability & performance considerations
  - done statically at design time rather than run time
- Composition tools generate monolithic firmware to achieve predictable behavior & better performance
- Optimization of static component composition
  - connections between components translated into direct function calls
- Verification of system requirements presently done using static analysis
References

- I. Crnkovic, Component-based software engineering for embedded systems (tutorial, ICSE 2005, pp. 712-713.