CIS 700: Integration of Embedded System Components: Principles and Practice

CIS 700, Fall 2005

- Class: Towne 307, 3-4:30 MW
- Instructor: Insup Lee
- Email: lee@cis.upenn.edu
- Office: 602 Levine
- Office Hours: 2-2:55 TTh
- Course web: tbd

Challenges and Opportunities for Embedded Systems

Insup Lee
Department of Computer and Information Science
University of Pennsylvania

September 7, 2005

Embedded Systems

- Embedded system are
  - devices used to control, monitor or assist the operation of appliances, gadgets, equipment, machinery or plant;
  - an integral part of the system.
- The next frontier
  - Mainframe computing (60's-70's)
    - Large computers to execute big data processing applications
  - Desktop computing (80's-90's)
    - One computer at every desk to do business/personal activities
  - Ubiquitous computing (00's-?)
    - Numerous computing devices in every room/person
    - "Invisible" part of the environment

A Variety of Application Domains

- Hybrid and embedded systems
  - Aerospace, automobiles, robotics, process control, sensor networks, smart spaces
- Multimedia
  - Virtual reality, immersive environment
- Consumer electronics
  - Mobile phones, office electronics, digital appliances
- Network components
  - Bridges, routers, switches, hubs
- Medical devices and instruments
  - Patient monitoring, MRI, infusion pumps, artificial organs
- E-business
  - ATM, vending machines
- Distributed and grid computing
  - Critical infrastructure defense system, air traffic control, intelligent highway systems, emergence response system

Characteristics of Embedded Systems

- Tightly coupled to the physical world; i.e., interacts with (or reacts to) its environment
- Correct operation is subject to
  - Physical constraints imposed by the environment
  - Resource constraints of the device
- Heterogeneity, networked at larger scale
- Sociological and ethical requirements
  - Users are not system experts
  - Security and privacy

Key Trends and Economic Impact

- Growing importance of software
- Great variety of component types
- Increasing complexity
- Increasing number of non-functional constraints
- Open standards
- Shortening time to market
- Increasing integration and networking
- Dependability
- Reuse of existing hardware and software components

Example: Automotive Telematics

- In 2005, 30-90 processors per car
  - Engine control, Break system, Airbag deployment system
  - Windshield wiper, door locks, entertainment systems
  - Example: BMW 745i
    - 2,000,000 LOC
    - Window CE OS
    - Over 60 microprocessors
      - 53 8-bit, 11 32-bit, 7 16-bit
    - Multiple networks
    - Buggy?

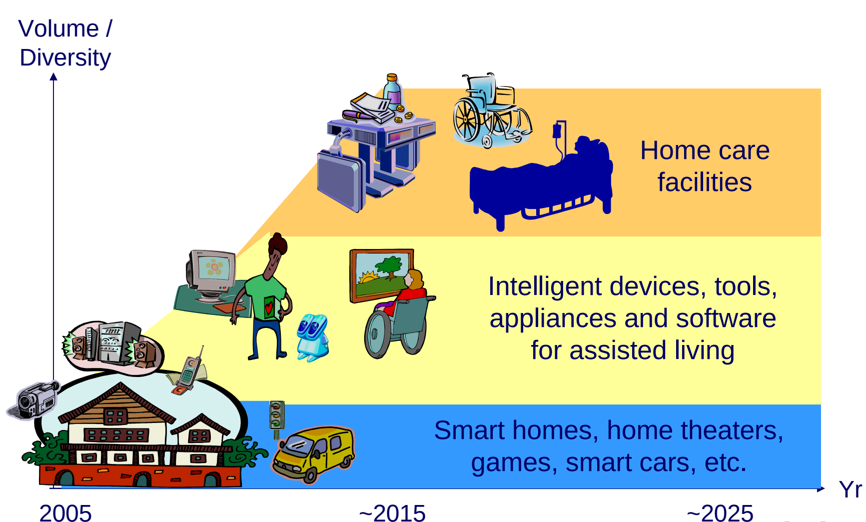
- Problems
  - Disparity between the design cycle of a car and the design cycle of embedded components
  - Difficult to upgrade
  - Not possible to integrate the user's own devices into a car

Challenges

- Three aspects of embedded system development
  - Embedding for smart control
  - Creating new computing gadgets
  - Connecting the physical world to the computing infrastructure

- The goal is to make them invisible cost-effectively!
  - Trustworthy: should not fail (or gracefully degrade), and safe to use. The existence of embedded software becomes apparent only when an embedded system fails.
  - Context Aware: should be able to sense people, environment, and threats and to plan/notify/actuate responses to provide real-time interaction with the dynamically changing physical environment with limited resources.
  - Seamless Integration: should be invisible at multiple levels of a hierarchy: home systems, metropolitan systems, regional systems, and national systems.

Example: Home and Personal Appliances

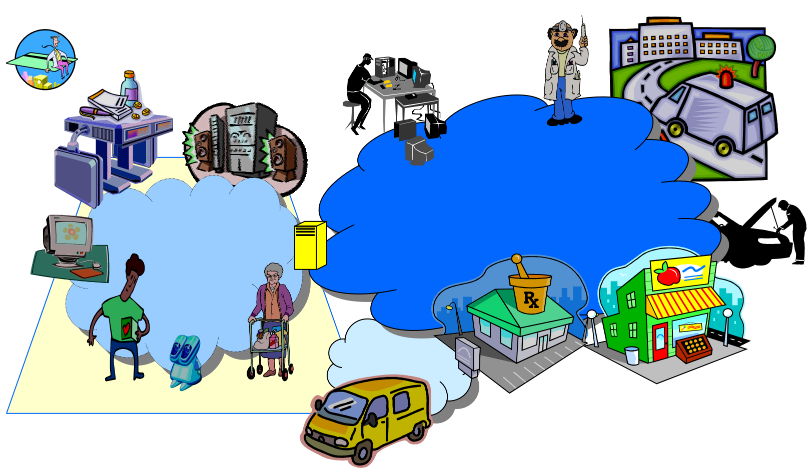


[Liu]

Justifications

- Rapid advances in component technologies, e.g.,
  - Smart gadgets, wearable sensors and actuators, robotic helpers, mobile devices
  - Wireless, wideband interconnects
- Increasing critical needs due to
  - Aging baby-boom generation
  - Long life expectancy
  - New safety, security, and privacy concerns

Embedded Home Environment

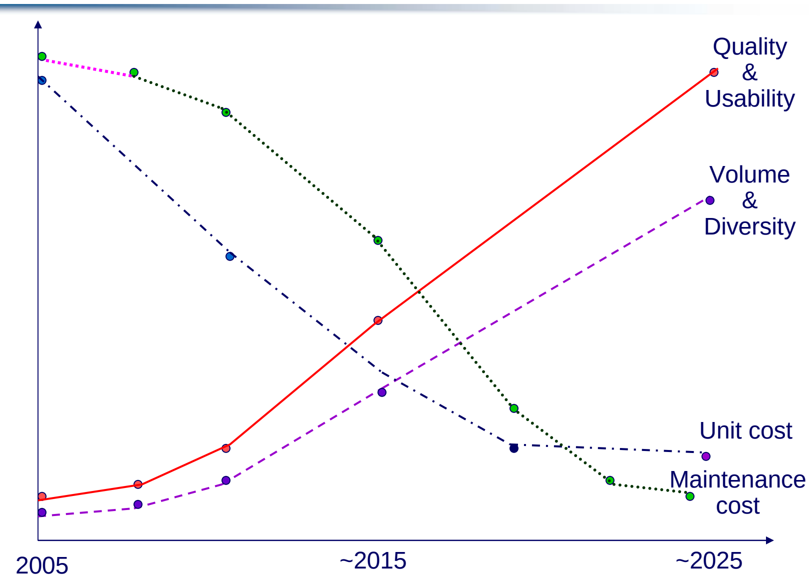


Observations

- Number of users: 10 – 1000 million
- Types of sensors and actuators: 100's
- Number of suppliers: 10 – 100's
- Required reliability: <10,000 recalls/year
- User tolerance to glitches: minimum
- Product life cycles: 3 – 20 yrs
- Tolerable upgrade effort: minimum

The environment must be open and evolvable, & capable of self diagnosis, healing, maintenance

Desired Trends



R&D Needs

- Predictability and manageability
- Self-configuration and adaptive coordination
  - Monitoring and system health
- New abstraction and computation models
- Incorporation of network geometry
- Interoperability for system integration
- Integration of technical, social, ethical, and public policy issues

"Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers," National Research Council, U.S.A.

Embedded Software

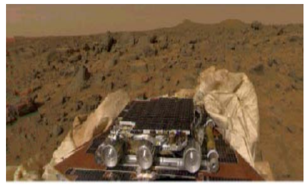
- The impact of information technology on embedded systems is exploding.
- Software development stands for 70-80 % of the overall development cost for some embedded systems.
- The development of embedded software components is needed
  - To help structured system design and system development
  - To reduce the cost of overall system development and maintenance efforts
  - To support the reuse of components within product families

Unexpected interactions

Implicit and inconsistent assumptions and abstractions

Incompatible assumptions of HW & SW regarding the operation of legs led to the loss of the Mars Polar Lander

Incompatible Cross Domain Protocols



Pathological Interaction between RT and sync. protocols Pathfinder caused repeated resets, nearly doomed the mission

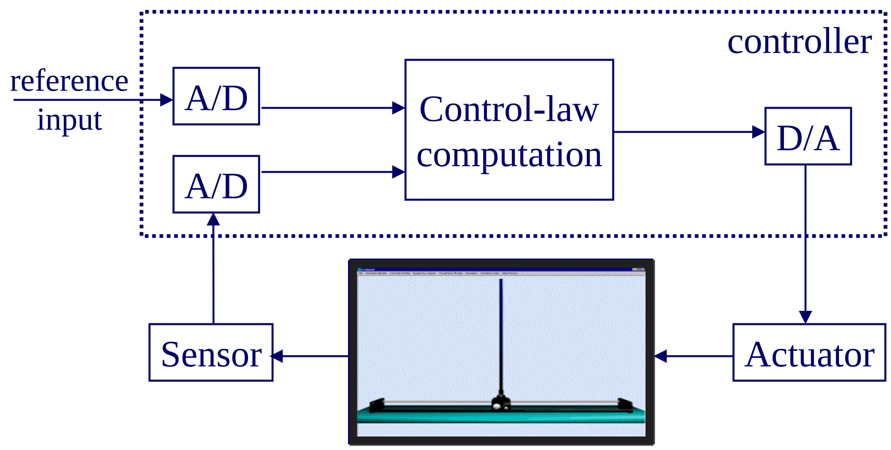
[Sha]

Sources of difficulties

- *Unsound compositionality*
  - incompatible abstractions, incorrect or implicit assumptions in system interfaces.
  - incompatible real time, fault tolerance, and security protocols.
  - combination of components do not preserve functional and para-functional properties; *unexpected feature interactions.*
- *Inadequate development infrastructure*
  - the lack of domain specific-reference architectures, tools, and design patterns with known and parameterized real time, robustness, and security properties.
- *System instabilities*
  - faults and failures in one component cascade along complex and unexpected dependency graphs resulting in catastrophic failures in a large part or even an entire system.

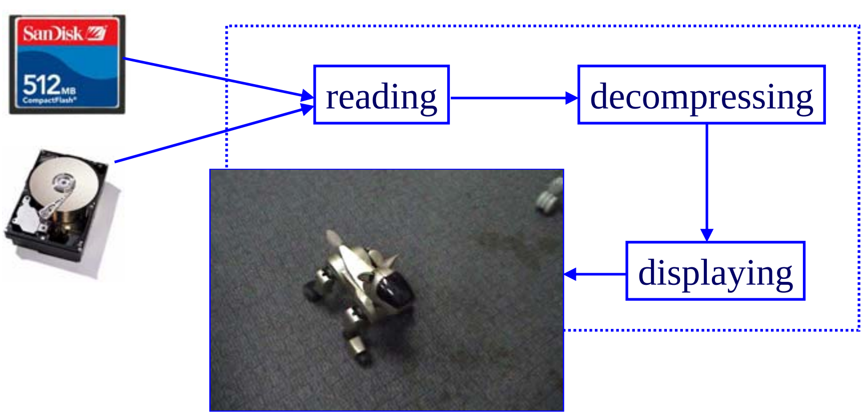
A real-time composition framework

- Digital controller



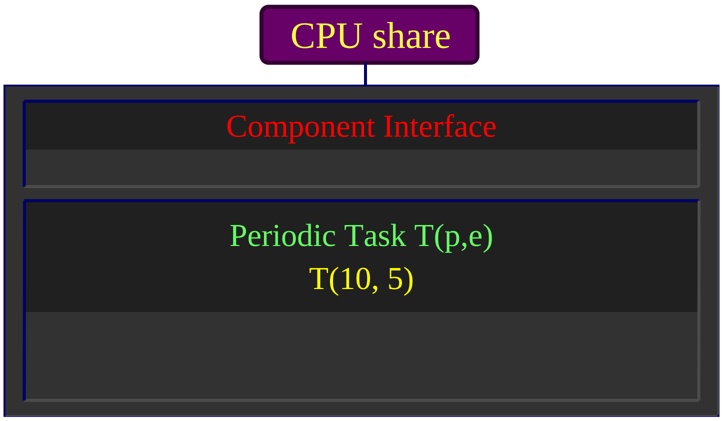
A real-time composition framework

- Multimedia application



A real-time composition framework

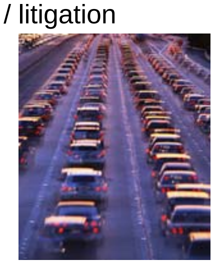
- Desirable to abstract component properties
  - Timing, resource (memory, energy)



Fall '05 CIS 700 21

Embedded Software: An Automotive Perspective

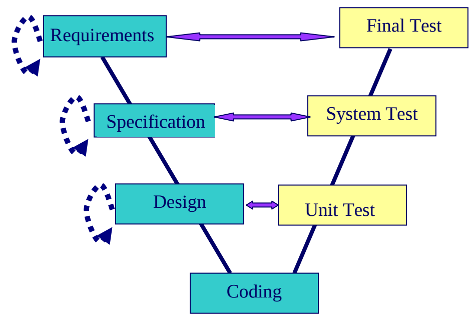
- The impact of information technology on embedded systems is exploding
- 4% of vehicle cost in 2000; 18% in 2010
  (GM: $7 billion in 2000; $40 billion in 2010)
- Testing: 50% of embedded software costs
  - Safety, fear of warranty costs / recalls / liability / litigation
  - **Lack of tool support**
- Efficiencies?
  - Hardware:
    Electronic Design Automation
  - Software:
    Model-Based Development



Fall '05 CIS 700 22

Model-based Development Process

- Requirements capture and analysis
  - Informal to formal
  - Consistency and completeness
  - Assumptions and interfaces between system components
  - Application-specific properties
- Design specifications and analysis
  - Formal modeling notations
  - Analysis techniques
  - Abstractions
- Implementation Generation & Validation
  - Testing
  - Model extraction and verification
  - Run-time monitoring and checking



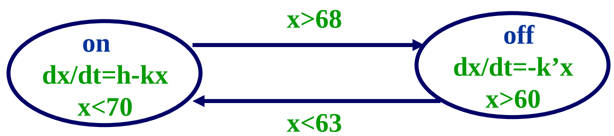
CHARON: Hybrid Modeling Framework

- Hybrid modeling of embedded systems
  - Physical plant / environment: **continuous dynamics**
  - Control software: **finite state machine**
  - Subject to formal verification
- CHARON language features
  - *Agents* and *modes* for architectural and behavioral modeling
  - *Analog* variables and *differential* / *algebraic* equations for modeling of continuous behaviors
  - *Transitions* and *guards* for describing switching of continuous behaviors
- CHARON toolkit
  - GUI for model composition
  - Simulation, verification, and code generation
- Case study in a robotic platform (AIBO)
  - Code generation for fairly complicated systems

What are Hybrid Systems?

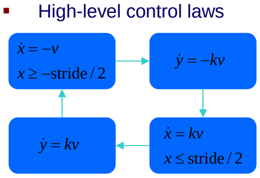
An embedded system consisting of sensors, actuators, plant, and control software is best viewed as a hybrid system.

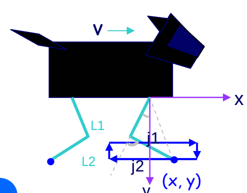
State machines + Dynamical systems



Example: Four Legged Robot

- Control objective
  - o $v = c$
- High-level control laws





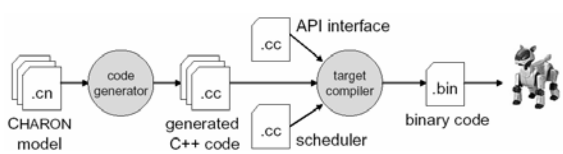
*[LCTES 2003] R. Alur, F. Ivancic, J. Kim, I. Lee, and O. Sokolsky. Generating embedded software from hierarchical hybrid models.

- Low-level control laws

$$j_1 = \arctan(x/y) - \arccos\left(\frac{x^2 + y^2 + L_1{}^2 - L_2{}^2}{2L_1\sqrt{x^2+y^2}}\right)$$

$$j_2 = \arccos\left(\frac{x^2 + y^2 + L_1{}^2 - L_2{}^2}{2L_1L_2}\right)$$

- Code generation



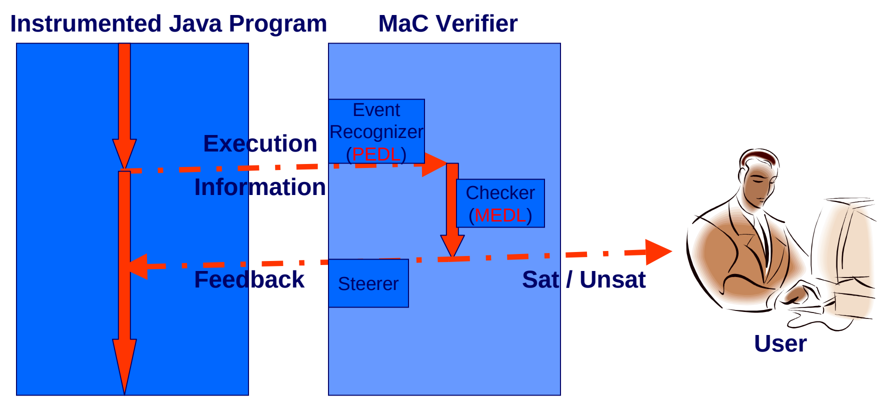
Model-based development R&D

- DARPA MoBIES (Model-Based Integration of Embedded Software)
  - Avionics, automobiles, software radios
  - Tech transfer by ISIS-Escher at ISIS, an non-profit consortium
- OMG standardization efforts
- Commercial tools

Run-time verification

- Run-time monitoring and checking w.r.t. formal specification
- Ensures the runtime compliance of the current execution of a system with its formal requirement
- Steps
  1. **Specify** formal requirements
  2. **Extract** information from current executing program
  3. **Check** the execution against formal requirements
  4. **Steer** the computation to a safe state
- Complementary methodology to formal verification and program testing
  - Validate implementation
  - Not complete: guarantee for current execution
  - Prevention, avoidance, and detection & recovery
- Used by NASA, etc.

Java-MaC (Monitoring and Checking)



[Kim, Viswanathan, Kannan, Lee, Sokolsky, FMSD 2004]

Validation and Certification

- To ensure the quality of embedded systems
  - We need sound scientific foundations for validation and certification of embedded systems.
  - Certification in two steps:
    - Design has the right properties
    - Implementation confirms to the design
- To achieve
  - Eliciting formal models from informal requirements
  - Model validation
  - Implementation validation
- Insurable embedded systems, which requires quantifiable reliability, liability and risk.

Critical System Integration Technologies

- Problems
  - Proliferation of sensors/actuator networks
  - Large scale; e.g., millions of GPS and mobile communication devices in automobiles and traffic management systems in major cities for performing traffic control and emergency response functions
  - Cannot have full knowledge of all the systems that may have to interface in the future
  - Need something more powerful than Internet to integrate the sensor-management-actuator system of systems in the future
  - The challenge is the scale and the stringent requirements on timely information, security and fault tolerance
- Needs
  - Interface engineering technologies based on machine-checkable interface specifications
  - System integration supports
  - Robust software architecture
  - Open system integration standards
  - Model-based development methods for component usability, robustness, etc.

Conclusions

- We have been successful in many ways in the past.
- There are many interesting and promising research and development activities in embedded systems.
  - Not discussed: Programming Languages/Paradigms, RTOS/Middleware, Architecture/Hardware, Stream Data Management, Security and Privacy
- The future has many exciting new challenges and opportunities.

Topics for the course

- Embedded system development process
  - Quality assurance, Metrics
- Software architecture
  - AADL
- Modeling and analysis
  - Hybrid systems
  - Timed automata
- Programming languages for RTES
- RTOS & Middleware
- Testing & Validation
- Certification
- Integration & Composition
  - Compositional RT scheduling framework
  - Interface theory and tools
  - Integration of components
  - PnP (for OR of the future)
- Security & Privacy
- Applications/Case studies
  - Sensor networks, RFID
  - High-confidence medical devices: Infusion pump
  - Reconfigurable robots