1. Software Engineering

Young W. Lim

2016-03-11 Thr

Outline

Based on

"Software Engineering for Embedded Systems. . .", R Oshana and M Kraeling, 2013

A1

- near-optimal performance
- roubstness
- distribution
- dynamism
- mobility

A1

- mechanical eng
- civil eng
- chemical eng
- electrical engnu
- nuclear eng
- aeronautical eng

A1

- methods/methodologies/techniques
- languages
- tools
- processes

A1

- software
- malleable
- human-sensitive construction
- intangible
- unexpectedly complex problems involved
- hardware dependent
- unusual rigor
- discontinuous operational nature

Waterfall software development modl

- Requirements
- Design
- Implementation
- Integration
- Validation
- Deployment

Spiral software development model

- Determine objectives, alternatives, constraints
- Evaluate alternatives, identit, resolve risks, develop prototypes
- Develop, verify, next-level product
- Plan next phases

Agile software development model

- Feature Set
- Budget Approval
- Feature Prioritzation
- Sprint
- Customer acceptance and testing
- Potential shippable product

A1

1. Problem defintion
2. Architecture design
3. Implementation
4. Verification and Validation

Foundational software engineering principles

- rigor and formality
- separation of concerns
  - modularity and decomposition
  - abstraction
- anticipation of change
- generality
- incrementality
- scalability
- compositionality
- heterogeneity
- from principles to tools

Software engineering principles

tools -> methodologies -> method and techniques -> principles

Typical embedded system components

- Processor core
- Analog I/O
- Sensors and actuators
- User interfaces
- Applicaton-specific gates (ASIC, FPGA)
- Software
- Memory
- Emulation and diagnostics (JTAG)

Embedded System as a reactive system

- sensors (energy conversion, signal conditioning)
- computer (decision making)
- actuation (power modulation)
- physical system (mechanical electrical, etc)

Several key characteristics of embedded systems

a. monitoring and reacting to the environment b. controlling the environment c. processing of information d. application-specific e. optimized for the application f. resource constrained g. real-time h. multi-rate

Real-time systems

types of real-time system

- soft real-time
- hard real-time

Difference between real-time and time-shared systems

- high degree of schedulability
- worst case latency
- stability under transient overload

Time-shared systems

- system capcity : high throughput
- responsiveness : fast average response time
- overload : fairness to all

Real-time System

- system capcity : schedulability and ability of system tasks to meet all deadlines
- responsiveness : ensured worst case latency which is the worst-case response time to events
- overload : stability; when the system is overloaded important tasks must meet deadlines while others may be starved

the feasibility and costs of hard real-time computing depends on

- timeliness parameters (arriaval period, upper bounds)
- deadlines
- worst case execution times
- ready and suspension times
- resource utilization profiles
- precedence and exclusion constraints
- relative importance

system characteristics

- system loading
- resource interactions
- queuing disciplines
- arbitration mechanisms
- interrupt priorities and timing
- caching

Real-time characteristics

real-time event categories

- asynchronous events are entirely upredictable
- synchronous events are predictable events with regularity
- isosynchronous events occur with requlrity within a given time window

Challenges in real-time system design

- response time
- recovering from failures

embedded system software build process

- compiling / assembling using an optimizing compiler
- linking using a linker
- relocaing using a locator

embedded system vs desk-top systems

1. energy efficiency
2. custom voltage/power requiremens
3. security
4. reliability
5. environment
6. efficient iteraction with user
7. integrated with design in hw/sw co-design approach

A1

- Distributed and multi-processor architectures
- initialization of the system
- processor interfaces
- load distribution
- centralized resource allocation and management

software for embedded systems

- super loop architecture
- power-save super loop

HAL (hardware abstraction layers) for embedded systems

- Low level drivers CAN driver, ADC driver, Timer Driver, GPIO driver, Special driver
- High level drivers Communication Protocol encapsulation
- Custom drivers Custom component (UDP encapsulation)
- Application layer Application
- Modelling layer Modelling in MATLAB, etc

A1

- benefits of a HAL
- easy migraton between embedded processors
- leverages existing processor knowledge base
- creats code compliant with a defined programming interface API