EECE 276
Embedded Systems

Design:
Software properties
Engineering principles
Properties of Software

- Reliability: measure of dependability
- RT systems:
  - “Downtime is lower than 1 hr per year”
  - “Accuracy is within 1E-3”
  - “Deadlines are met consistently”
- Statistical metric: (T: time of fault)
  \[ r(t) = P(T > t) \]
  The probability that the SW system will operate without a failure for specified period of time.
  \[ r(t) = 1 \Rightarrow \text{never fails}, \]
  \[ 10E-9/\text{hour} \Rightarrow r(t) = (0.99999999)^t : \text{“Manned systems”} \]
Properties of Software

- “Failure functions” (physical systems)
  - Inverse exponential (decreasing)
  - “Bathtub” curve
- For SW: not a good model!
- Correctness:
  - Compliance with all (incl. real-time) requirements
- Performance
  - Algorithmic complexity (scaling)
  - Temporal domain performance
Properties of Software

- Usability
  » “Human factors”
- Interoperability
  » Working together with other systems
- Maintainability
  » Evolvability & repairability
- Portability
  » How easy is it to port it to other systems/platforms
- Verifiability
  » Can we verify that it performs as expected?
# Properties of Software

<table>
<thead>
<tr>
<th>Quality metric</th>
<th>Measurement Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>Probabilistic measures, MTBF, MTFF</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Compliance with accepted standards</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Anecdotal observation of resources spent</td>
</tr>
<tr>
<td>Performance</td>
<td>Algorithmic complexity analysis, direct measurement, simulations</td>
</tr>
<tr>
<td>Portability</td>
<td>Anecdotal observations</td>
</tr>
<tr>
<td>Reliability</td>
<td>Probabilistic measures, MTBF, MTFF</td>
</tr>
<tr>
<td>Usability</td>
<td>User feedback from surveys, problem reports</td>
</tr>
<tr>
<td>Verifiability</td>
<td>Software monitors</td>
</tr>
</tbody>
</table>
Software Engineering Principles

- Rigor and formality
  » Precision and (possibly) mathematical techniques in all phases of the project
  » Real-time systems: interactions with the physical world and human users

- Separation of concerns
  » Modularization of code (see below)
  » Aspect-oriented programming (see AOP)

- Modularity
  » Cohesion: *intramodule* connectivity
  » Coupling: *intermodule* connectivity
Modularity

Right way: high cohesion, low coupling
Wrong way: low cohesion, high coupling
Software Engineering Principles

- Anticipation of change
  - Real-time systems often have a long life cycle.
  - Changes in HW/SW infrastructure, specifications, etc. must be anticipated

- Generality
  - “General” solutions through modularization

- Incrementality
  - Iterative/rapid prototyping development style

- Traceability
  - Linking requirements to design artifacts, etc.
Summary: The Design Process

- Hardware/software trade-off analysis
- Designing interfaces to external components
- Designing interfaces between components
- Deciding between centralize and distributed processing schemes
- Defining concurrency execution
- Designing control strategies
- Designing data storage, maintenance, and allocation strategies
- Designing database structures and handling routines
- Designing the startup and shutdown processing
- Designing algorithms for functional processing
- Designing error processing and handling
- Conducting performance analyses
- Specifying the physical location of components and data
- Designing test software
- Creating documentation for the system: Operator/User/Programmer’s Manual(s)
- Conducting internal reviews
- Developing a detailed design for the components of the software architecture
- Developing test cases and procedures for formal acceptance testing
- Documenting the software architecture in a design document
- Presenting the design details in formal design reviews