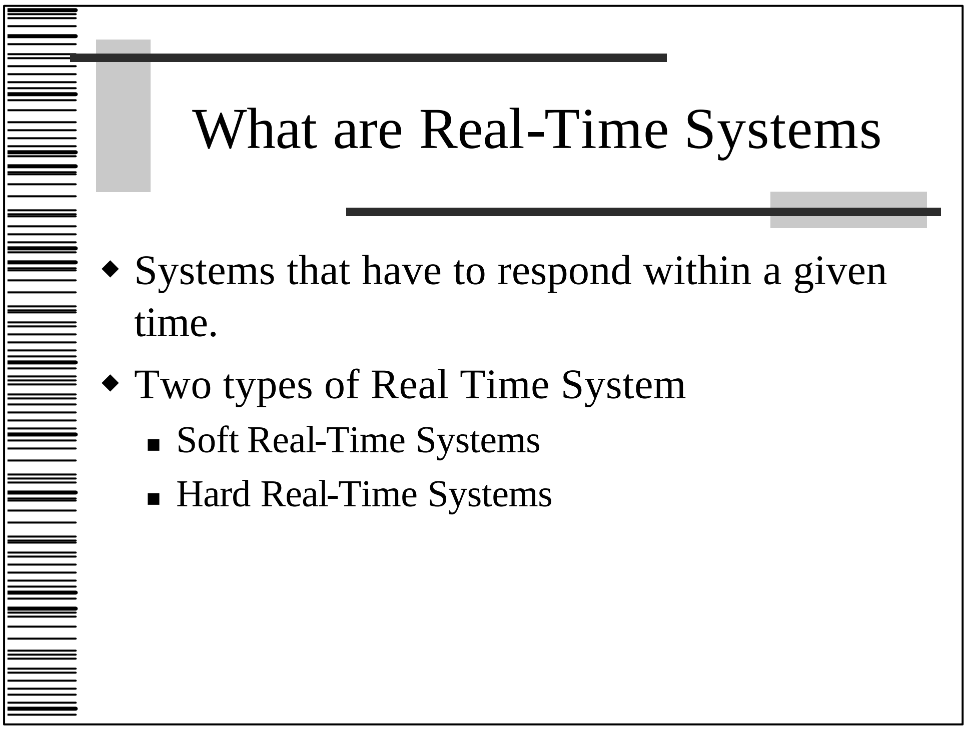
Software Engineering for Real-Time Systems.

- Presented by Andrew Dyer-Smith and Jamie McClelland

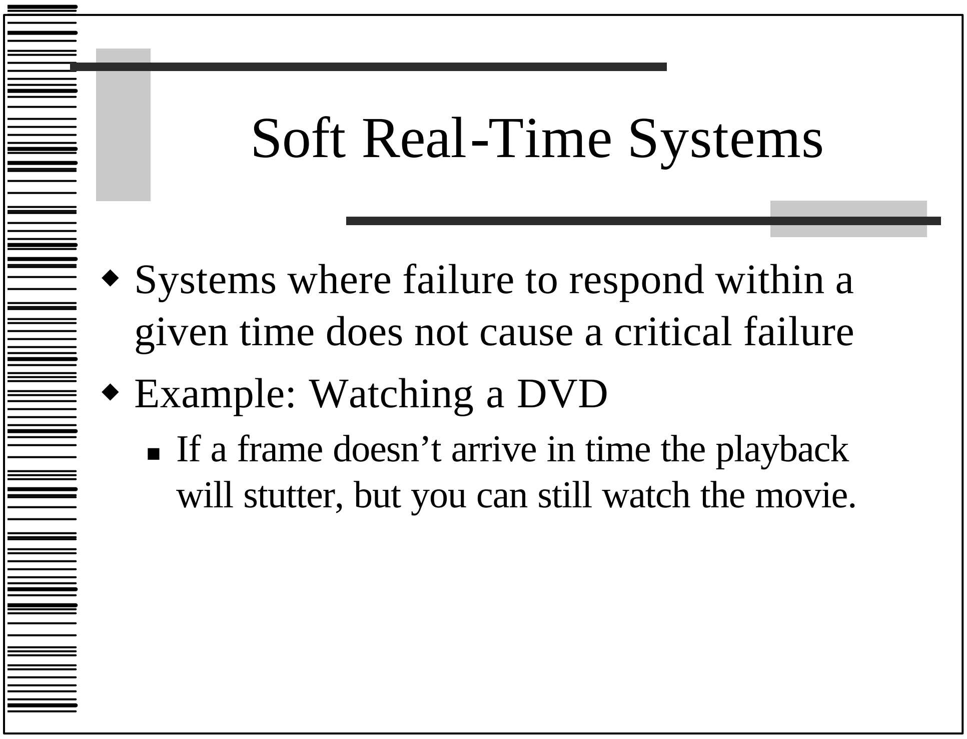
Overview

- What are Real-Time Systems.
- Requirements of Real-Time Systems
- Current Technology
- Construction



What are Real-Time Systems

- Systems that have to respond within a given time.
- Two types of Real Time System
  - Soft Real-Time Systems
  - Hard Real-Time Systems



Soft Real-Time Systems

- Systems where failure to respond within a given time does not cause a critical failure
- Example: Watching a DVD
  - If a frame doesn’t arrive in time the playback will stutter, but you can still watch the movie.

Hard Real-Time Systems

- System where a failure to respond within a given time causes a critical failure.
- Example: Nuclear Power Plant
  - If the system doesn’t notice a problem with the nuclear reactor quick enough it will MELT-DOWN.

Safety Critical Systems

- In safety critical systems failure to respond in time is not an option.
- In the past some form of backup was provided to take over when the software system failed.
- But now systems are wholly software controlled Real-Time systems.
- Therefore we will concentrate on the design of Hard Real-Time systems.

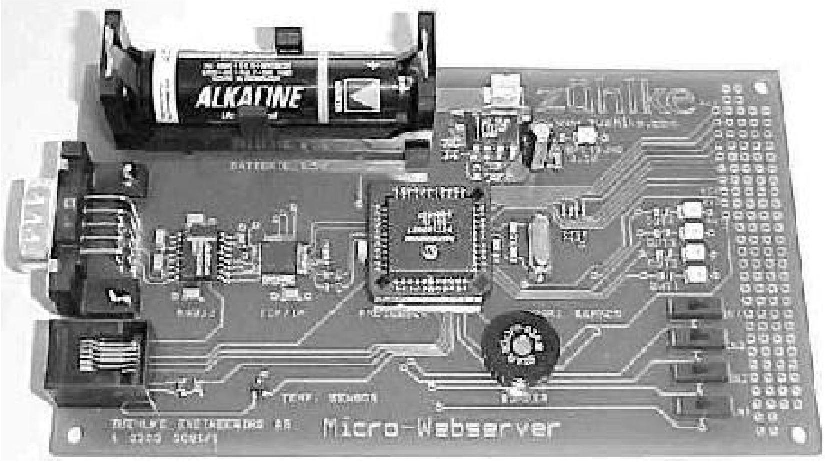
Current Technology

- Current technology is making the design and implementation of Hard Real-Time Systems a lot more feasible.
- We will give four examples.

System on a Chip(SOC)

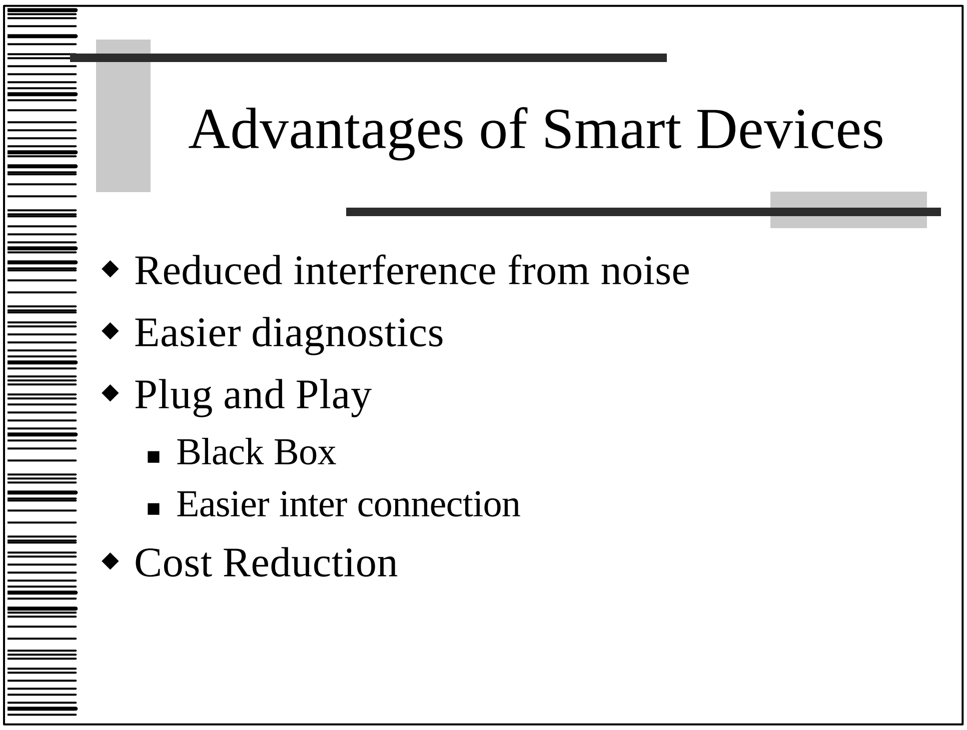
- Nowadays it is possible to build an entire system on one chip including:
  - 32 bit CPU
  - A megabyte of memory
  - I/O and network circuitry
- They can be mass produced and thus have low production cost, although development costs tend to be quite high.

Example: Webserver on a Chip



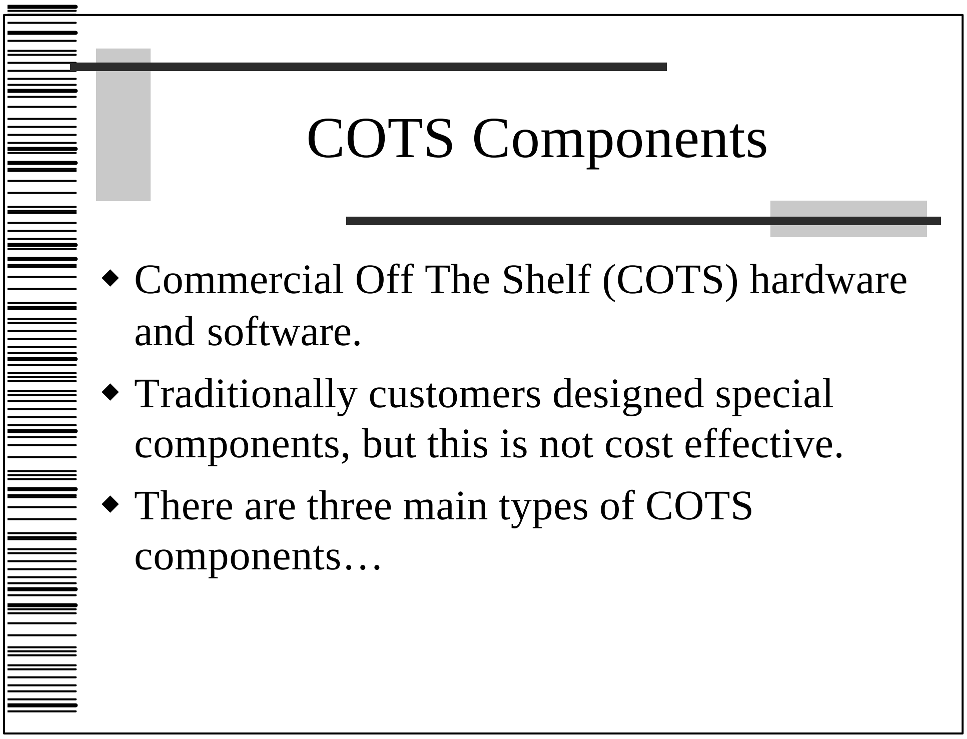
Smart Devices and MEMS Sensors

- Smart devices are the combination of a sensor or actuator and a local microcontroller.
- MicroElectronic Mechanical Systems (MEMS) are sensor elements that can be integrated onto the same silicon die as the microcontroller.



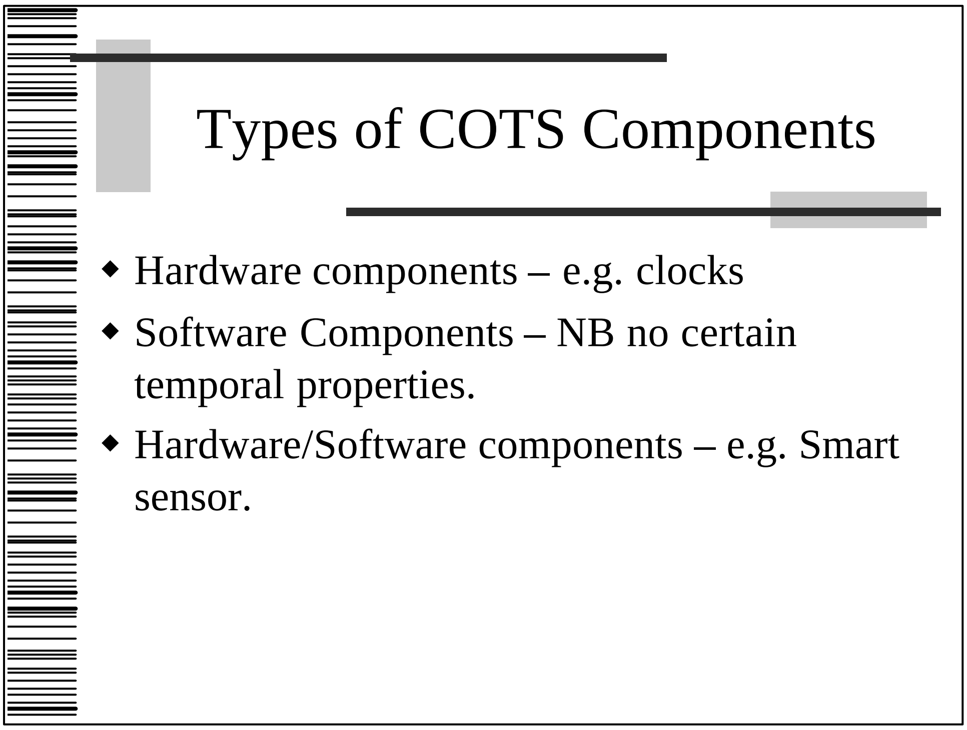
Advantages of Smart Devices

- Reduced interference from noise
- Easier diagnostics
- Plug and Play
  - Black Box
  - Easier inter connection
- Cost Reduction



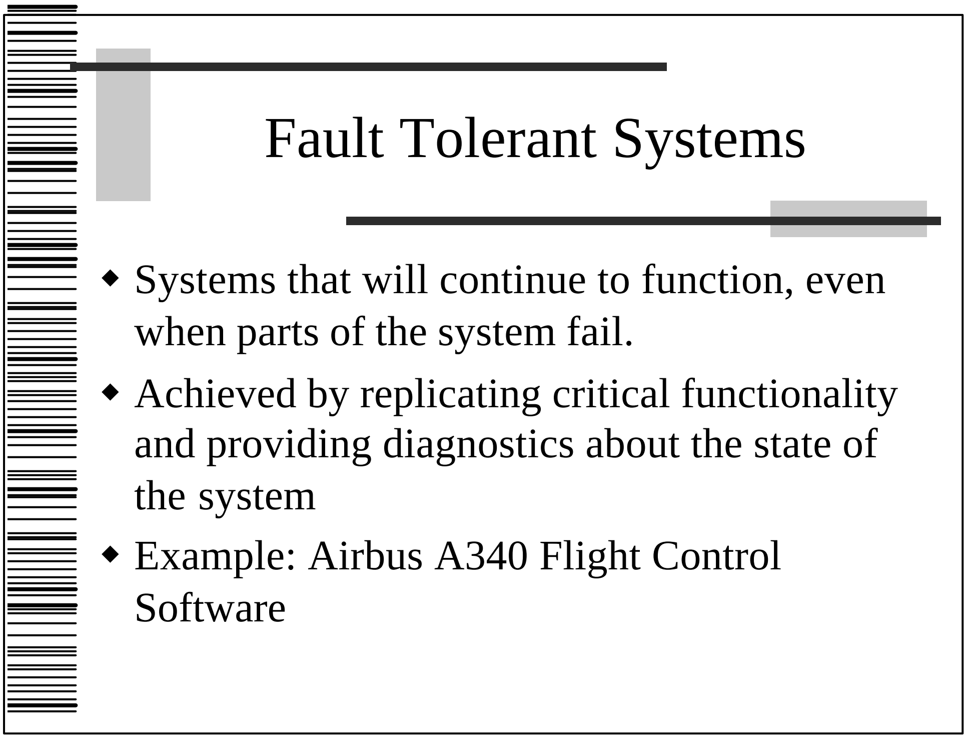
COTS Components

- Commercial Off The Shelf (COTS) hardware and software.
- Traditionally customers designed special components, but this is not cost effective.
- There are three main types of COTS components…



Types of COTS Components

- Hardware components – e.g. clocks
- Software Components – NB no certain temporal properties.
- Hardware/Software components – e.g. Smart sensor.



Fault Tolerant Systems

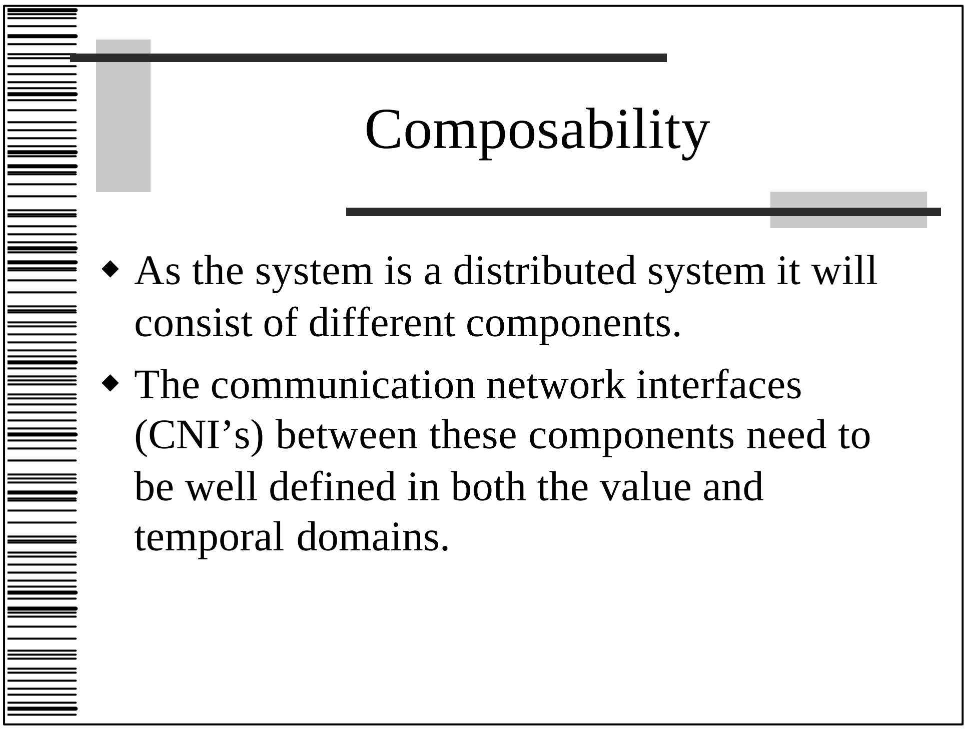
- Systems that will continue to function, even when parts of the system fail.
- Achieved by replicating critical functionality and providing diagnostics about the state of the system
- Example: Airbus A340 Flight Control Software

Construction of Hard Real Time Systems

- From the current technology trends it seems as if the future lies in distributed Real-Time Systems.
- They will consist of a network of nodes, these nodes will either be:
  - Powerful system nodes (SOC’s)
  - Smart sensor nodes

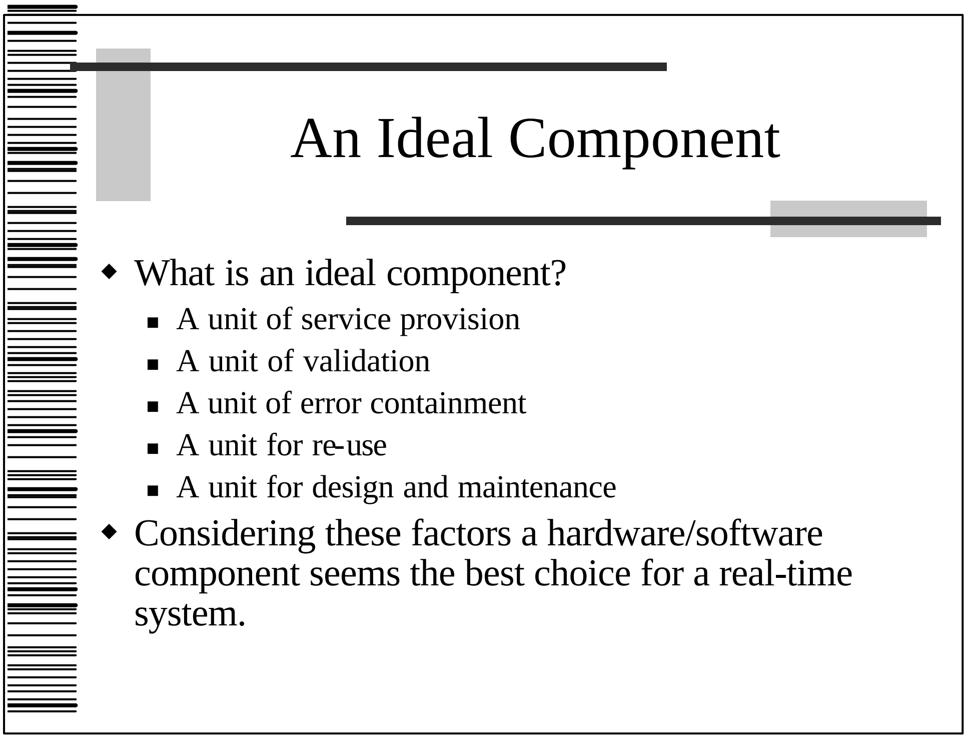
Top Down or Bottom Up

- Real-Time Systems can be designed top-down or bottom-up
  - Top down focuses on the architecture,
  - Bottom up on the individual components



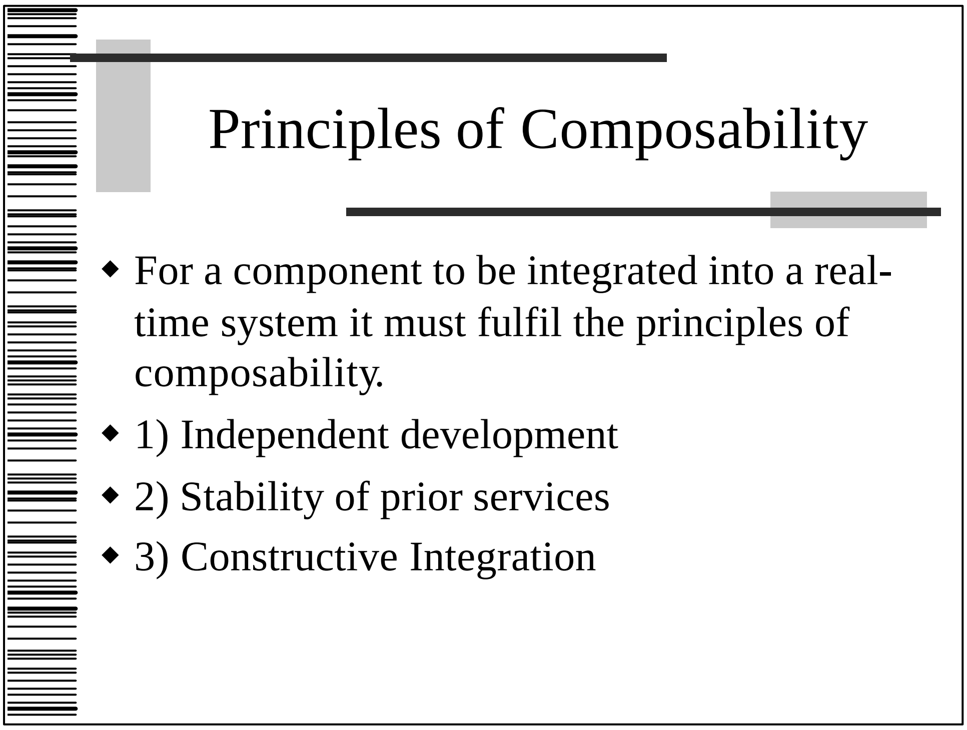
Composability

- As the system is a distributed system it will consist of different components.
- The communication network interfaces (CNI’s) between these components need to be well defined in both the value and temporal domains.



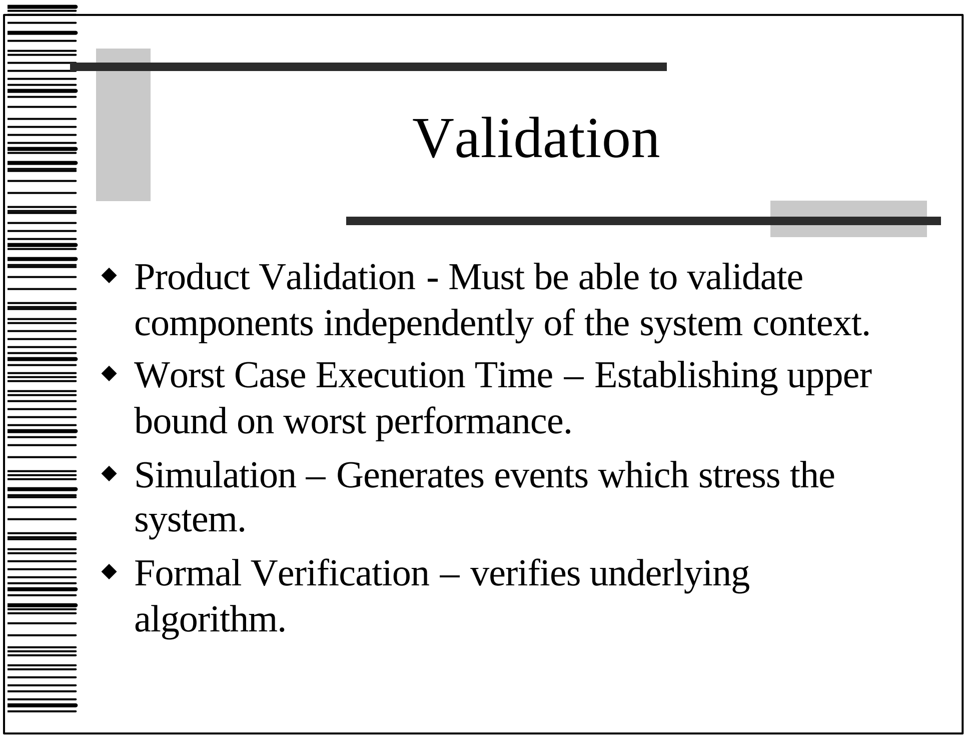
An Ideal Component

- What is an ideal component?
  - A unit of service provision
  - A unit of validation
  - A unit of error containment
  - A unit for re-use
  - A unit for design and maintenance
- Considering these factors a hardware/software component seems the best choice for a real-time system.



Principles of Composability

- For a component to be integrated into a real-time system it must fulfil the principles of composability.
- 1) Independent development
- 2) Stability of prior services
- 3) Constructive Integration



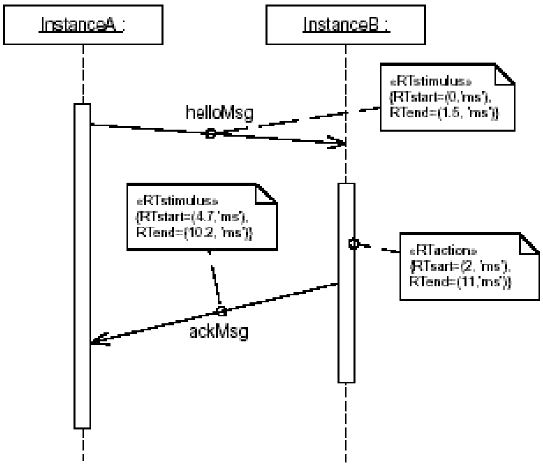
Validation

- Product Validation - Must be able to validate components independently of the system context.
- Worst Case Execution Time – Establishing upper bound on worst performance.
- Simulation – Generates events which stress the system.
- Formal Verification – verifies underlying algorithm.

Real-Time UML

- An extension to UML to provide support for developing Real-Time Systems.
- Provides techniques and notation for modelling and analysing Real-Time Systems.
- Not currently rigorous enough to allow temporal interface specification.

Sequence Diagram



Summary

- Real-Time Software Systems are becoming more viable due to technology advances.
- They require different design and verification techniques to non Real-Time Systems.
- The techniques that currently exist aren’t good enough. When they are there is a very bright future for Real-Time Software Systems.

References

- “Software Engineering for Real-Time: A Roadmap” by Hermann Kopetz
- Micro-Web server - www.zuhlke.co.uk
- “UML Profile for Schedulability, Performance, and Time” by Bran Selic – www.omg.org