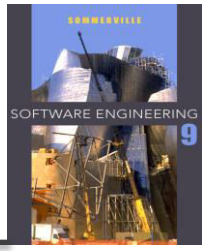
Chapter 20- Embedded Systems

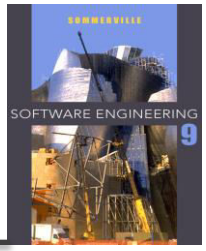
Lecture 1

Topics covered



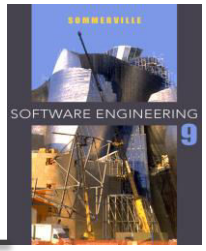
- ✧ Embedded systems design
- ✧ Architectural patterns
- ✧ Timing analysis
- ✧ Real-time operating systems

Embedded software



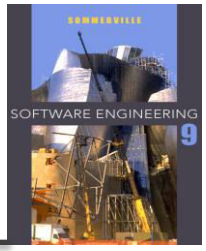
- Computers are used to control a wide range of systems from simple domestic machines, through games controllers, to entire manufacturing plants.
- Their software must react to events generated by the hardware and, often, issue control signals in response to these events.
- The software in these systems is embedded in system hardware, often in read-only memory, and usually responds, in real time, to events from the system's environment.

Responsiveness



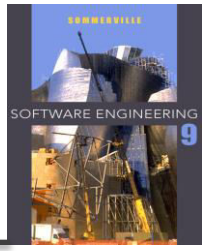
- Responsiveness in real-time is the critical difference between embedded systems and other software systems, such as information systems, web-based systems or personal software systems.
- For non-real-time systems, correctness can be defined by specifying how system inputs map to corresponding outputs that should be produced by the system.
- In a real-time system, the correctness depends both on the response to an input and the time taken to generate that response. If the system takes too long to respond, then the required response may be ineffective.

Definition



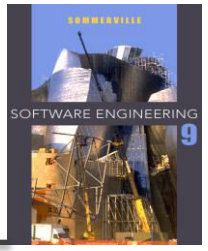
- A real-time system is a software system where the correct functioning of the system depends on the results produced by the system and the time at which these results are produced.
- A soft real-time system is a system whose operation is degraded if results are not produced according to the specified timing requirements.
- A hard real-time system is a system whose operation is incorrect if results are not produced according to the timing specification.

Embedded system characteristics



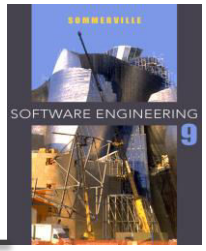
- Embedded systems generally run continuously and do not terminate.
- Interactions with the system's environment are uncontrollable and unpredictable.
- There may be physical limitations (e.g. power) that affect the design of a system.
- Direct hardware interaction may be necessary.
- Issues of safety and reliability may dominate the system design.

Embedded system design



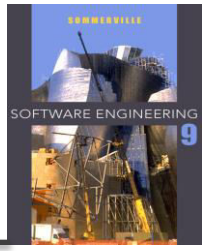
- The design process for embedded systems is a systems engineering process that has to consider, in detail, the design and performance of the system hardware.
- Part of the design process may involve deciding which system capabilities are to be implemented in software and which in hardware.
- Low-level decisions on hardware, support software and system timing must be considered early in the process.
- These may mean that additional software functionality, such as battery and power management, has to be included in the system.

Reactive systems



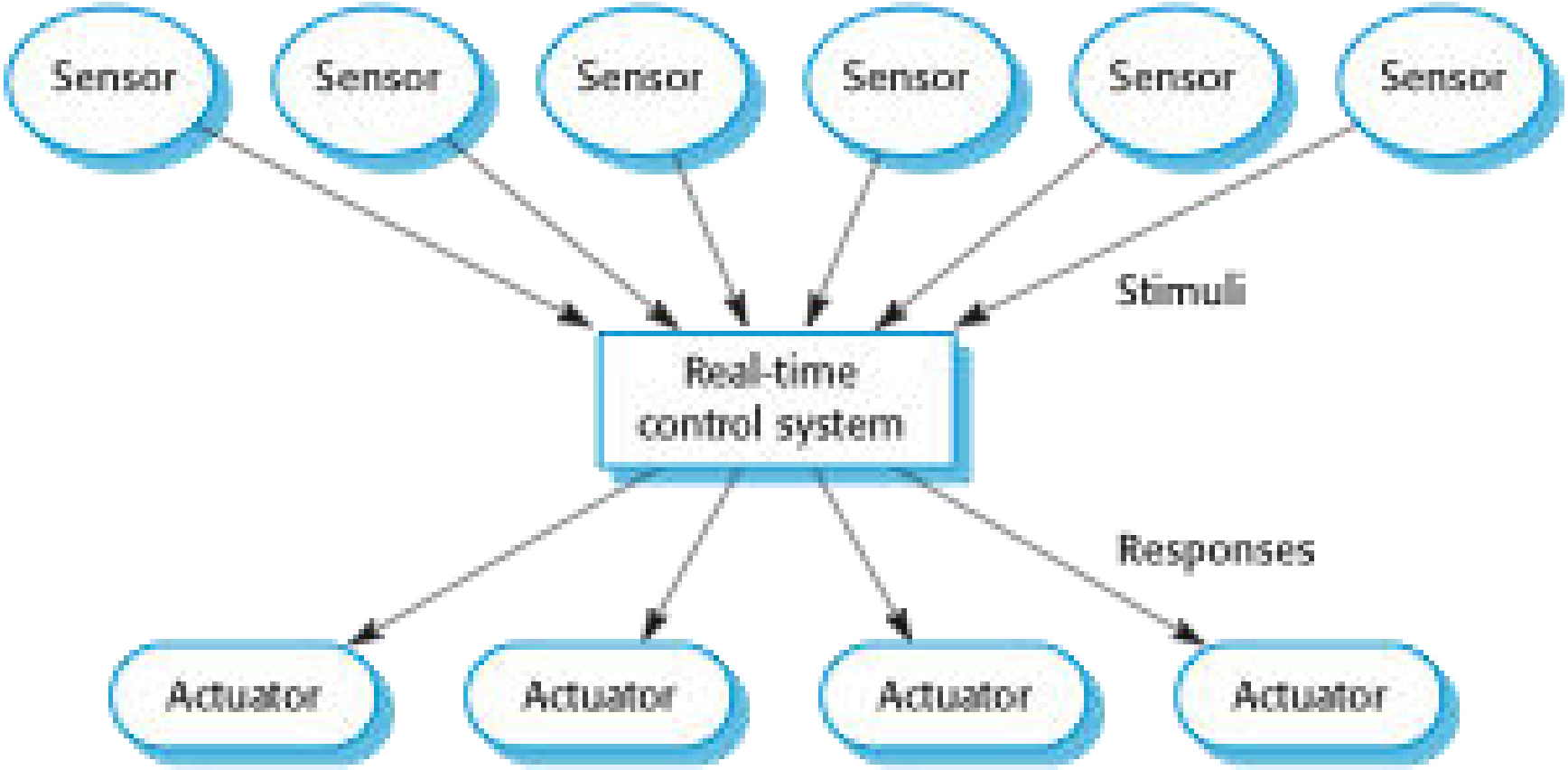
- Given a stimulus, the system must produce a reaction or response within a specified time.
- Periodic stimuli. Stimuli which occur at predictable time intervals
  - For example, a temperature sensor may be polled 10 times per second.
- Aperiodic stimuli. Stimuli which occur at unpredictable times
  - For example, a system power failure may trigger an interrupt which must be processed by the system.

Stimuli and responses for a burglar alarm system

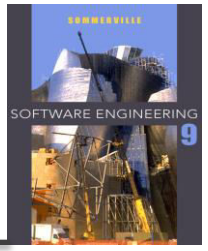


Stimulus	Response
Single sensor positive	Initiate alarm; turn on lights around site of positive sensor.
Two or more sensors positive	Initiate alarm; turn on lights around sites of positive sensors; call police with location of suspected break-in.
Voltage drop of between 10% and 20%	Switch to battery backup; run power supply test.
Voltage drop of more than 20%	Switch to battery backup; initiate alarm; call police; run power supply test.
Power supply failure	Call service technician.
Sensor failure	Call service technician.
Console panic button positive	Initiate alarm; turn on lights around console; call police.
Clear alarms	Switch off all active alarms; switch off all lights that have been switched on.

A general model of an embedded real-time system

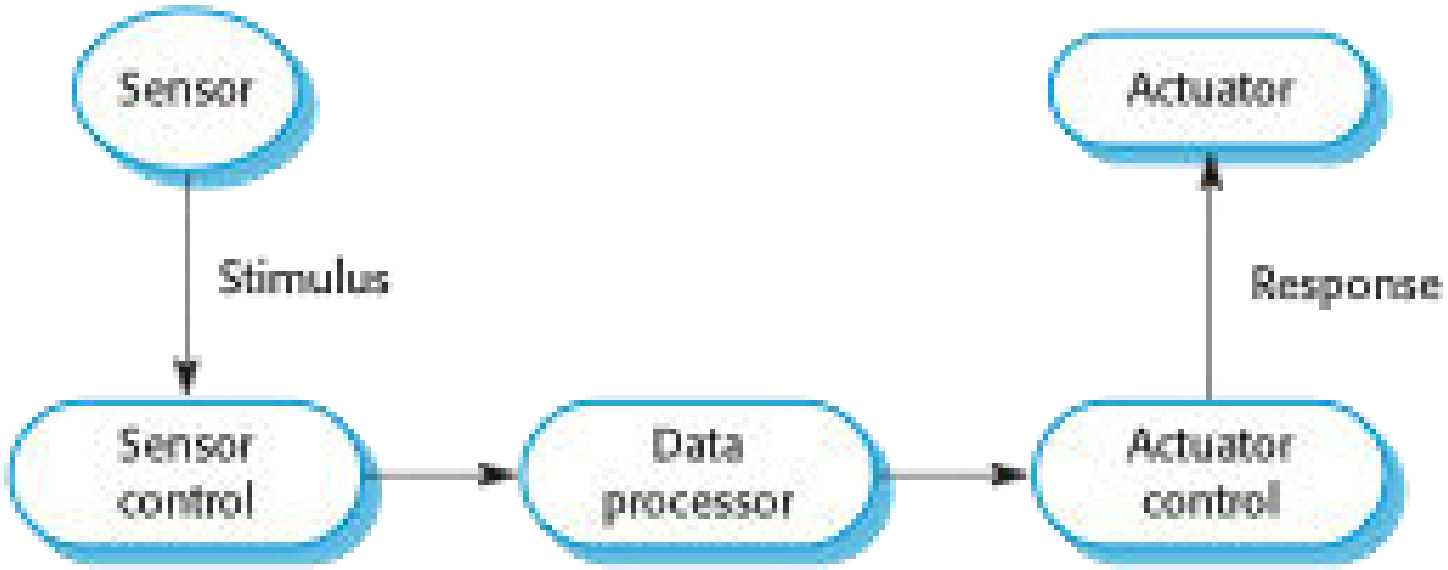


Architectural considerations

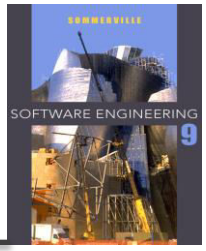


- Because of the need to respond to timing demands made by different stimuli/responses, the system architecture must allow for fast switching between stimulus handlers.
- Timing demands of different stimuli are different so a simple sequential loop is not usually adequate.
- Real-time systems are therefore usually designed as cooperating processes with a real-time executive controlling these processes.

Sensor and actuator processes

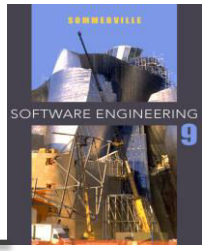


System elements



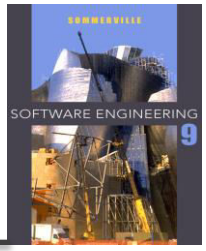
- Sensor control processes
  - Collect information from sensors. May buffer information collected in response to a sensor stimulus.
- Data processor
  - Carries out processing of collected information and computes the system response.
- Actuator control processes
  - Generates control signals for the actuators.

Design process activities



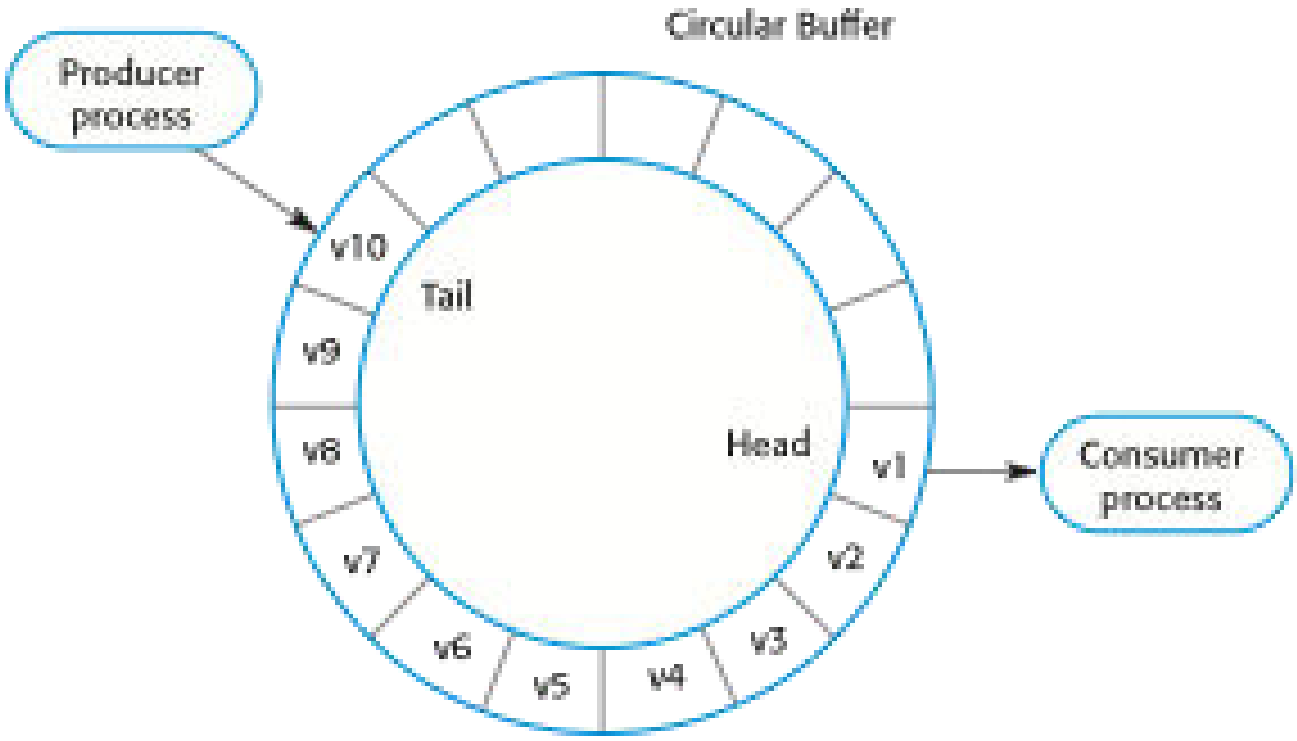
- Platform selection
- Stimuli/response identification
- Timing analysis
- Process design
- Algorithm design
- Data design
- Process scheduling

Process coordination

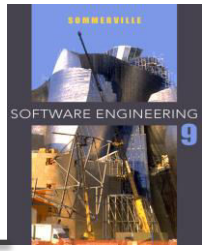


- Processes in a real-time system have to be coordinated and share information.
- Process coordination mechanisms ensure mutual exclusion to shared resources.
- When one process is modifying a shared resource, other processes should not be able to change that resource.
- When designing the information exchange between processes, you have to take into account the fact that these processes may be running at different speeds.

Producer/consumer processes sharing a circular buffer

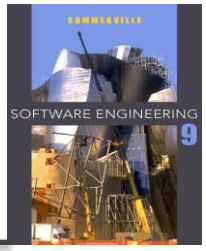


Mutual exclusion



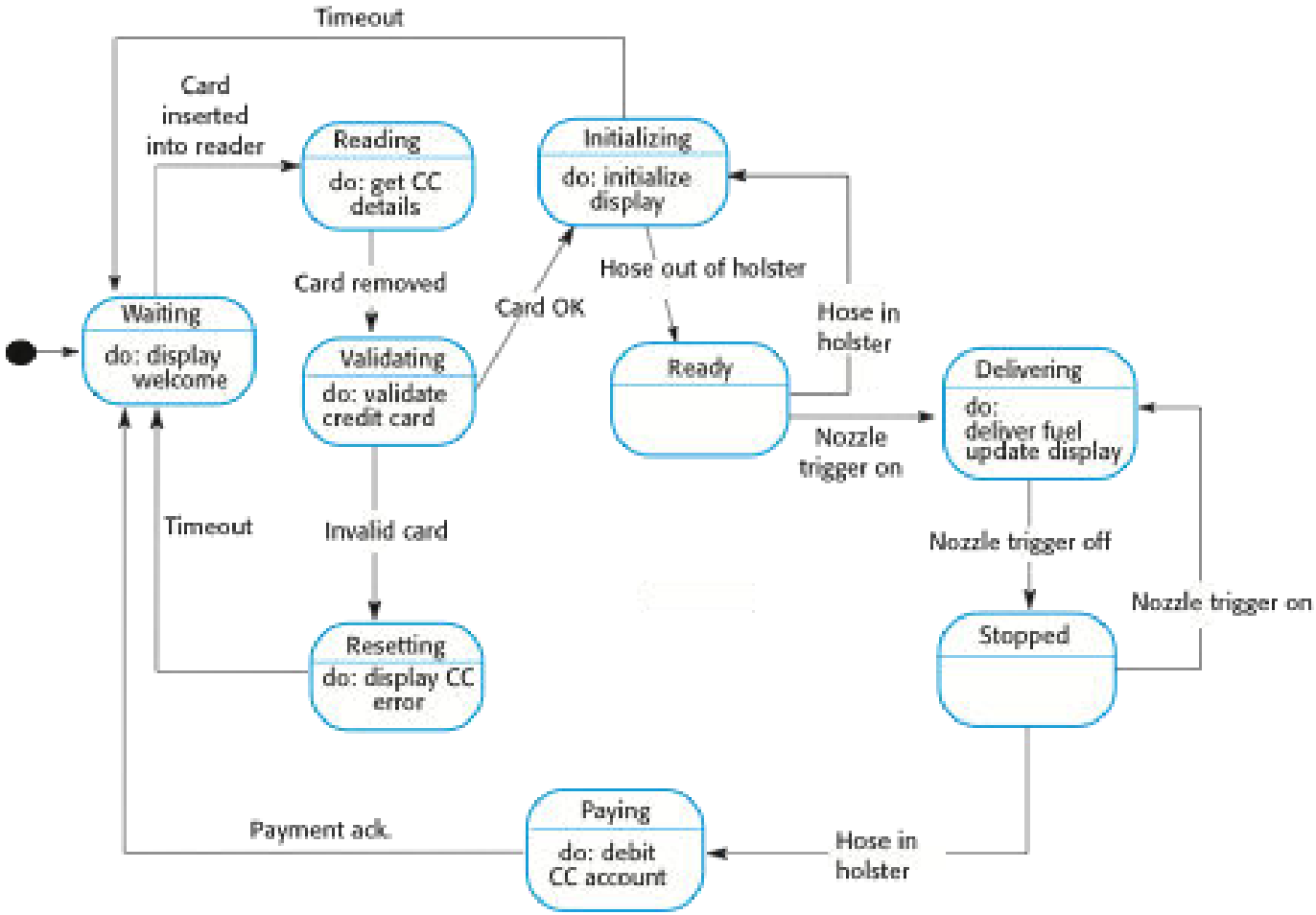
- Producer processes collect data and add it to the buffer. Consumer processes take data from the buffer and make elements available.
- Producer and consumer processes must be mutually excluded from accessing the same element.
- The buffer must stop producer processes adding information to a full buffer and consumer processes trying to take information from an empty buffer.

Real-time system modelling

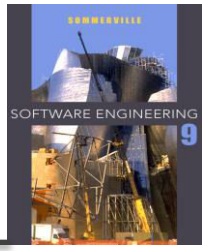


- The effect of a stimulus in a real-time system may trigger a transition from one state to another.
- State models are therefore often used to describe embedded real-time systems.
- UML state diagrams may be used to show the states and state transitions in a real-time system.

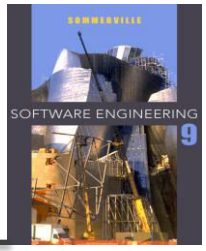
State machine model of a petrol (gas) pump



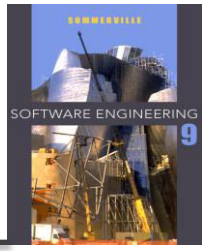
Real-time programming



Architectural patterns for embedded systems

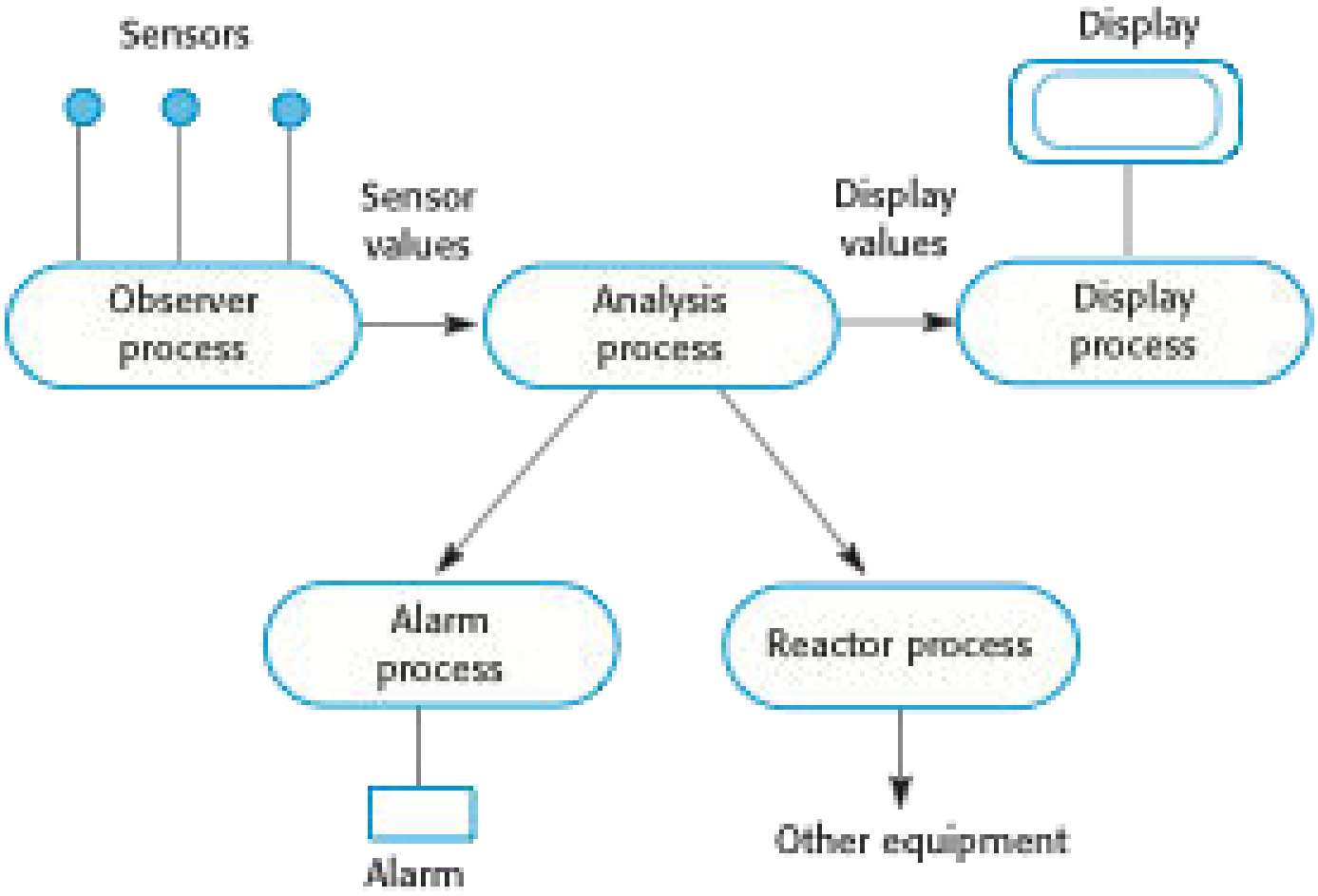


The Observe and React pattern

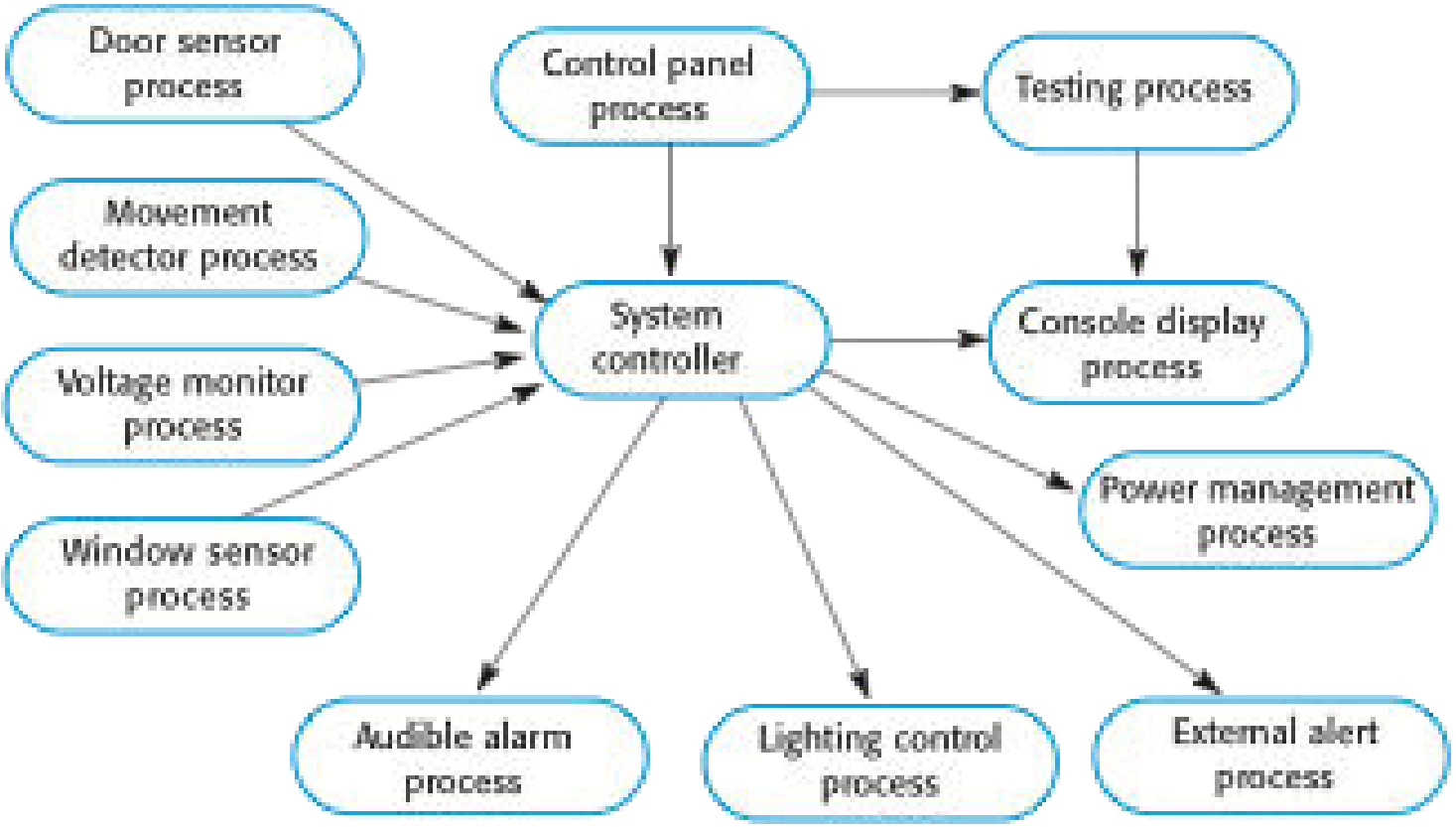


Name	Observe and React
Description	The input values of a set of sensors of the same types are collected and analyzed. These values are displayed in some way. If the sensor values indicate that some exceptional condition has arisen, then actions are initiated to draw the operator's attention to that value and, in certain cases, to take actions in response to the exceptional value.
Stimuli	Values from sensors attached to the system.
Responses	Outputs to display, alarm triggers, signals to reacting systems.
Processes	Observer, Analysis, Display, Alarm, Reactor.
Used in	Monitoring systems, alarm systems.

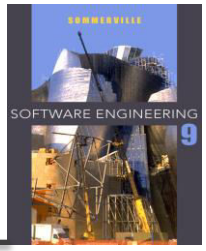
Observe and React process structure



Process structure for a burglar alarm system

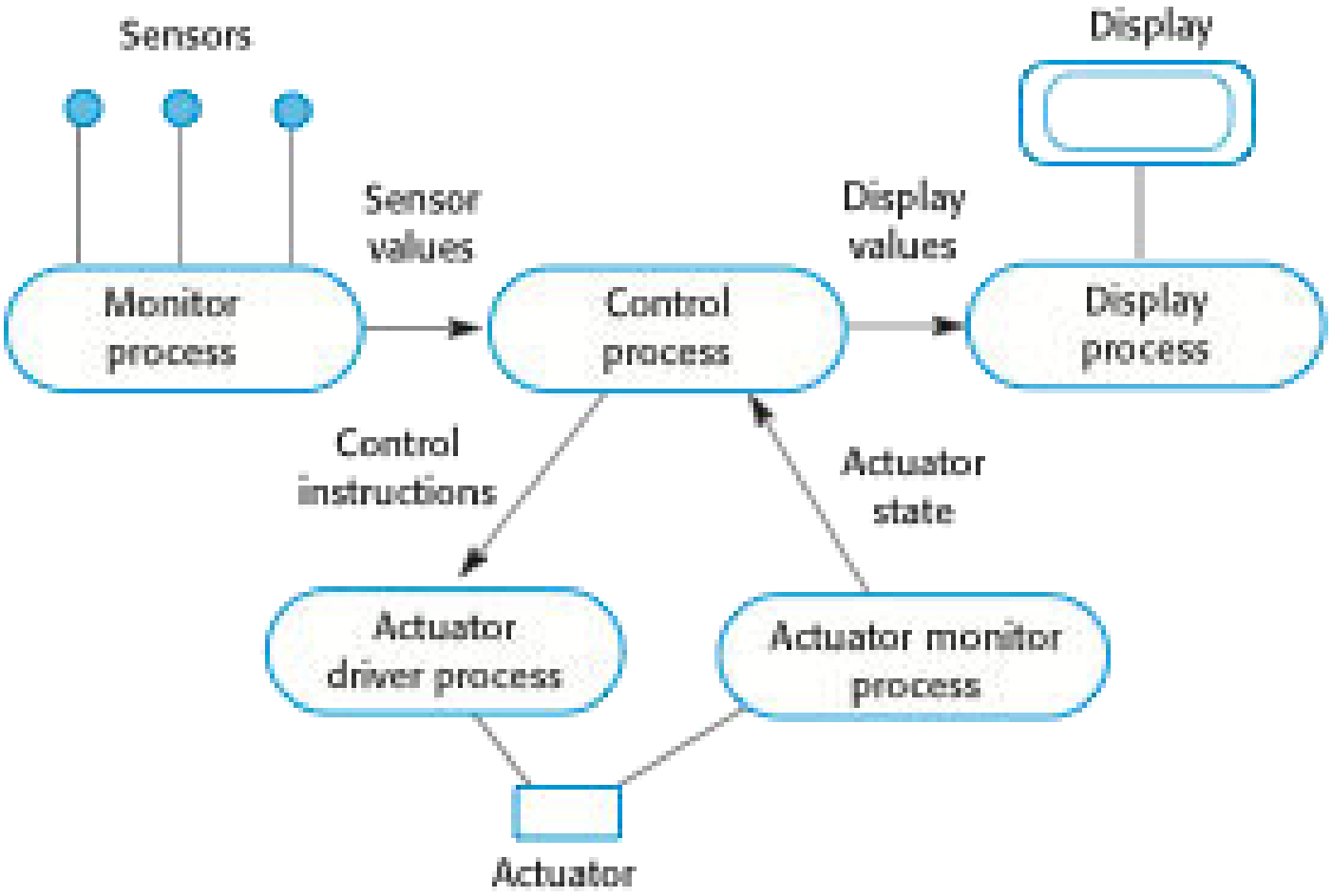


The Environmental Control pattern

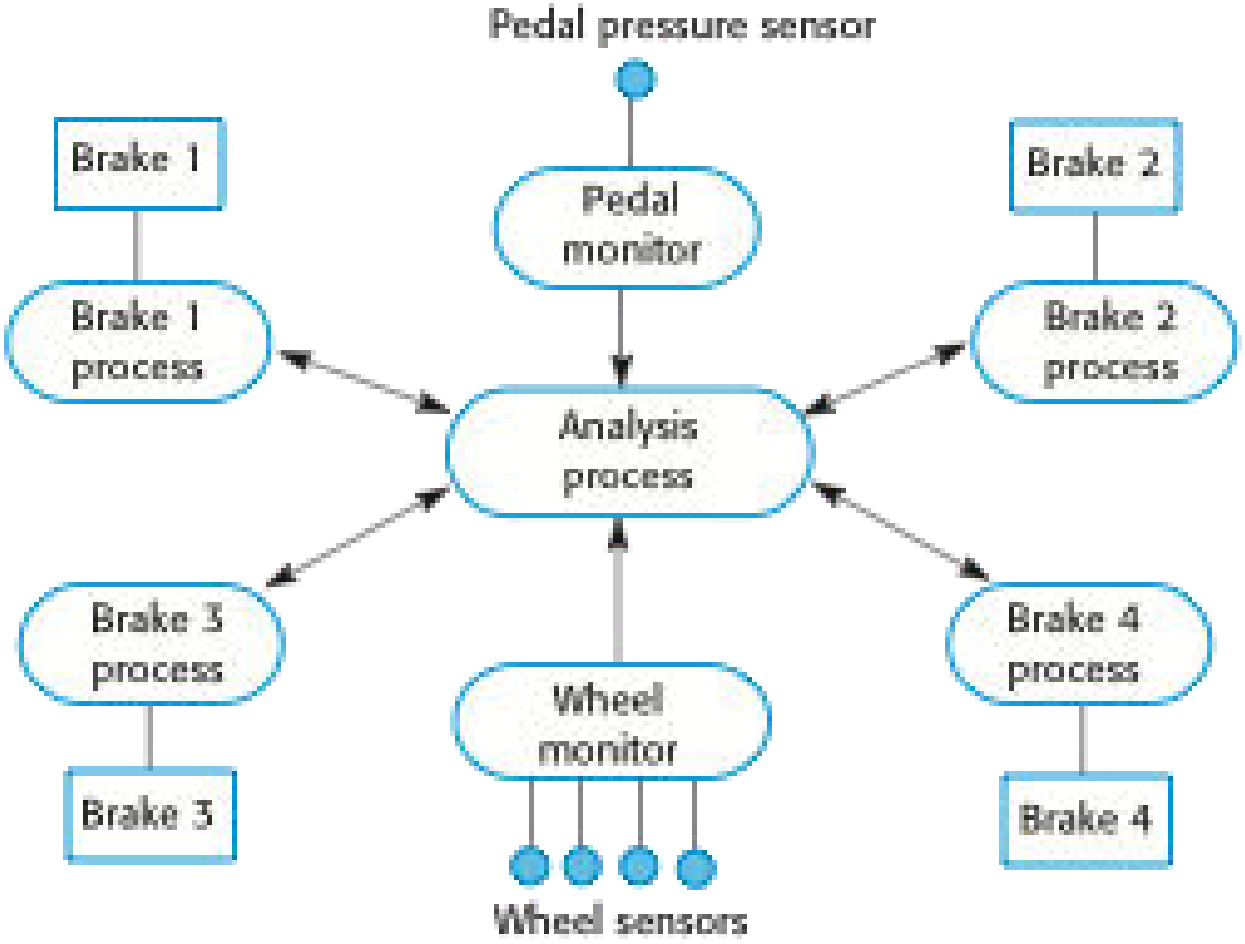


Name	Environmental Control
Description	The system analyzes information from a set of sensors that collect data from the system's environment. Further information may also be collected on the state of the actuators that are connected to the system. Based on the data from the sensors and actuators, control signals are sent to the actuators that then cause changes to the system's environment. Information about the sensor values and the state of the actuators may be displayed.
Stimuli	Values from sensors attached to the system and the state of the system actuators.
Responses	Control signals to actuators, display information.
Processes	Monitor, Control, Display, Actuator Driver, Actuator monitor.
Used in	Control systems.

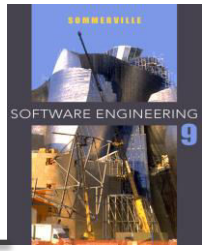
Environmental Control process structure



Control system architecture for an anti-skid braking system



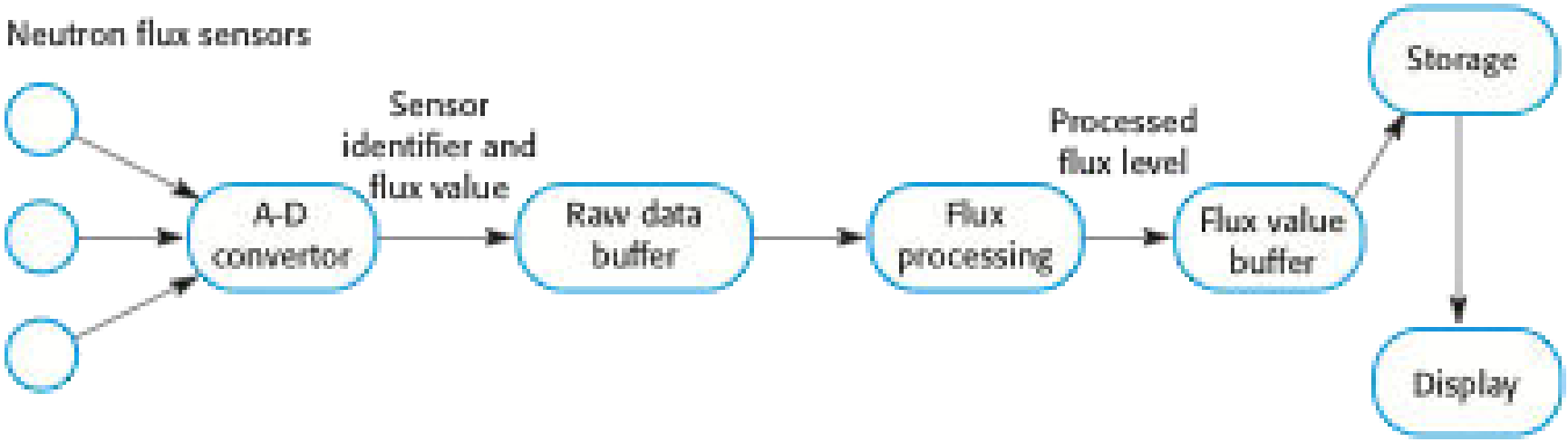
The Process Pipeline pattern



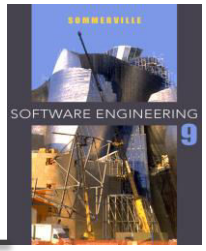
Name	Process Pipeline
Description	A pipeline of processes is set up with data moving in sequence from one end of the pipeline to another. The processes are often linked by synchronized buffers to allow the producer and consumer processes to run at different speeds. The culmination of a pipeline may be display or data storage or the pipeline may terminate in an actuator.
Stimuli	Input values from the environment or some other process
Responses	Output values to the environment or a shared buffer
Processes	Producer, Buffer, Consumer
Used in	Data acquisition systems, multimedia systems

Process Pipeline process structure

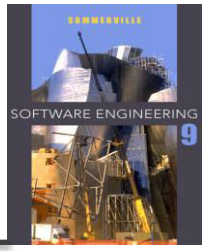
Neutron flux data acquisition



Timing analysis

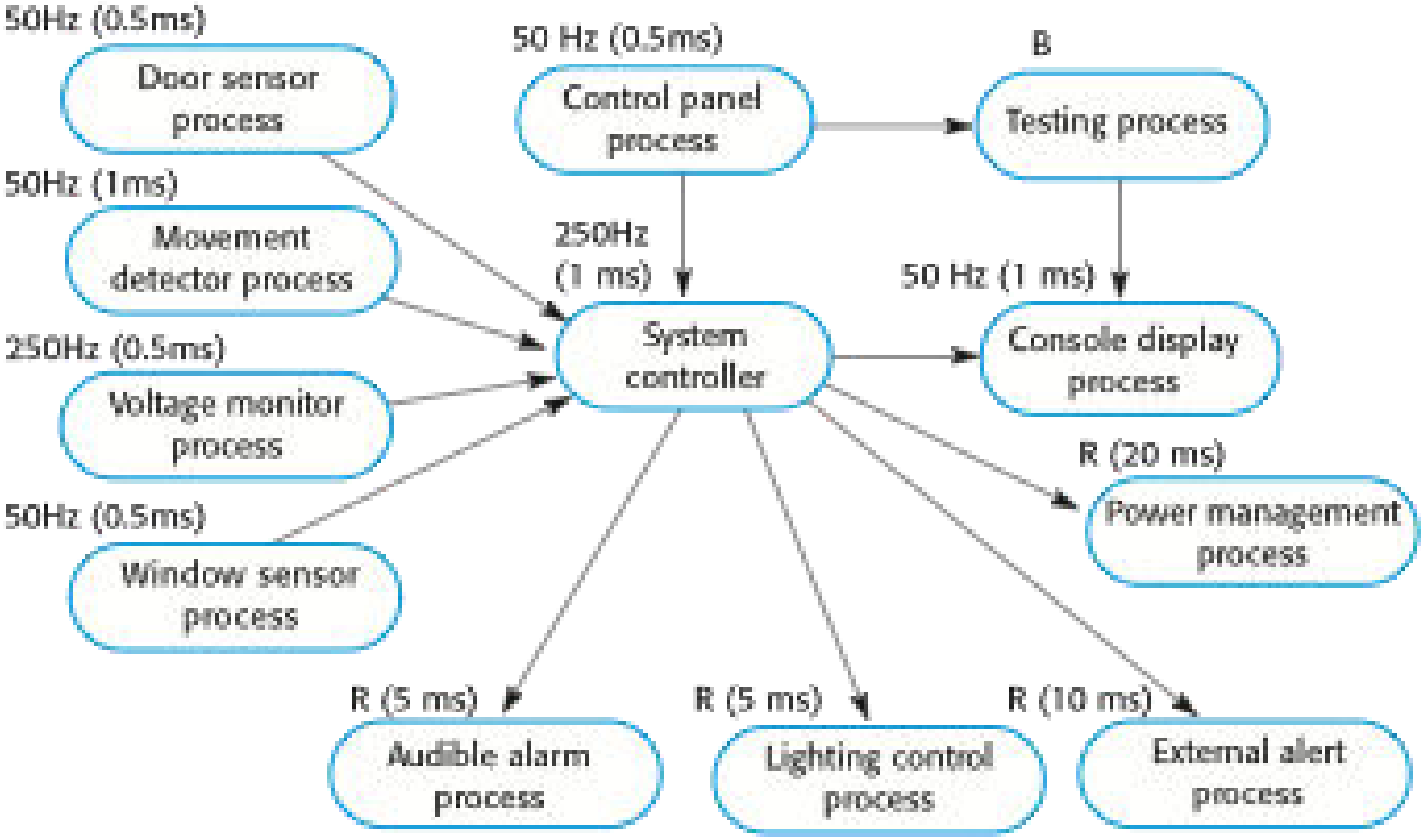


Timing requirements for the burglar alarm system

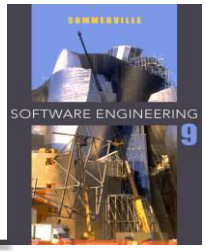


Stimulus/Response	Timing requirements
Power failure	The switch to backup power must be completed within a deadline of 50 ms.
Door alarm	Each door alarm should be polled twice per second.
Window alarm	Each window alarm should be polled twice per second.
Movement detector	Each movement detector should be polled twice per second.
Audible alarm	The audible alarm should be switched on within half a second of an alarm being raised by a sensor.
Lights switch	The lights should be switched on within half a second of an alarm being raised by a sensor.
Communications	The call to the police should be started within 2 seconds of an alarm being raised by a sensor.
Voice synthesizer	A synthesized message should be available within 2 seconds of an alarm being raised by a sensor.

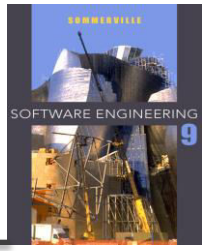
Alarm process timing



Real-time operating systems

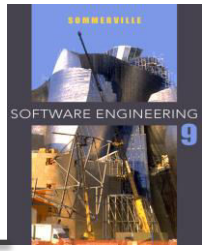


Operating system components



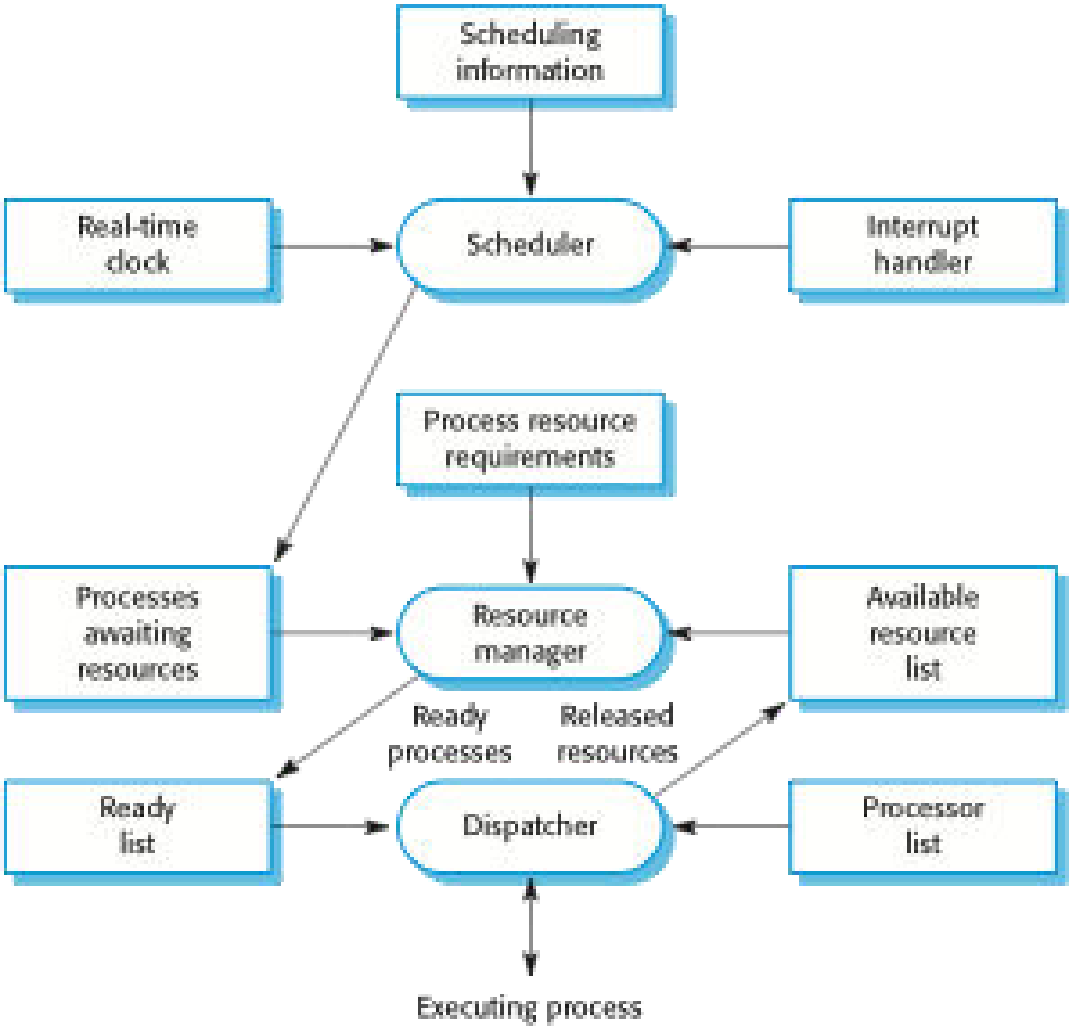
- Real-time clock
  - Provides information for process scheduling.
- Interrupt handler
  - Manages aperiodic requests for service.
- Scheduler
  - Chooses the next process to be run.
- Resource manager
  - Allocates memory and processor resources.
- Dispatcher
  - Starts process execution.

Non-stop system components

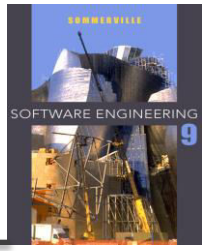


- Configuration manager
  - Responsible for the dynamic reconfiguration of the system software and hardware. Hardware modules may be replaced and software upgraded without stopping the systems.
- Fault manager
  - Responsible for detecting software and hardware faults and taking appropriate actions (e.g. switching to backup disks) to ensure that the system continues in operation.

Components of a real-time operating system

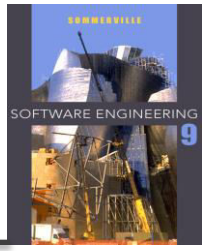


Process priority



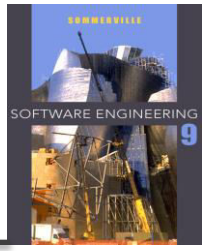
- The processing of some types of stimuli must sometimes take priority.
- Interrupt level priority. Highest priority which is allocated to processes requiring a very fast response.
- Clock level priority. Allocated to periodic processes.
- Within these, further levels of priority may be assigned.

Interrupt servicing



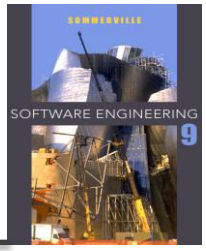
- Control is transferred automatically to a pre-determined memory location.
- This location contains an instruction to jump to an interrupt service routine.
- Further interrupts are disabled, the interrupt serviced and control returned to the interrupted process.
- Interrupt service routines MUST be short, simple and fast.

Periodic process servicing



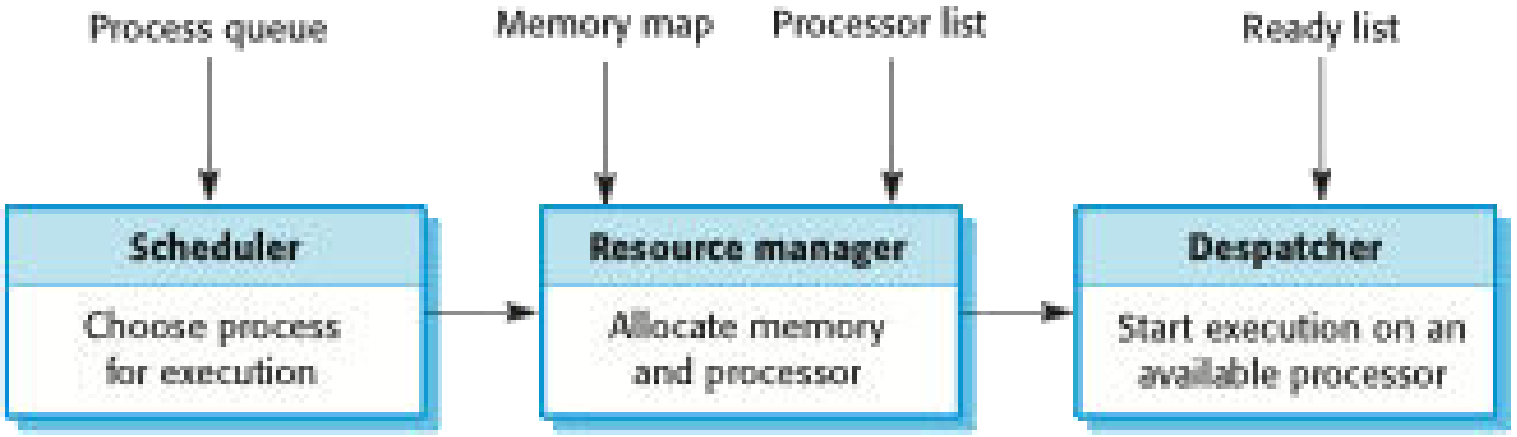
- In most real-time systems, there will be several classes of periodic process, each with different periods (the time between executions), execution times and deadlines (the time by which processing must be completed).
- The real-time clock ticks periodically and each tick causes an interrupt which schedules the process manager for periodic processes.
- The process manager selects a process which is ready for execution.

Process management

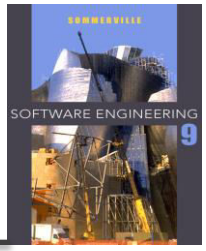


- Concerned with managing the set of concurrent processes.
- Periodic processes are executed at pre-specified time intervals.
- The RTOS uses the real-time clock to determine when to execute a process taking into account:
  - Process period - time between executions.
  - Process deadline - the time by which processing must be complete.

Figure 20.18 RTOS actions required to start a process

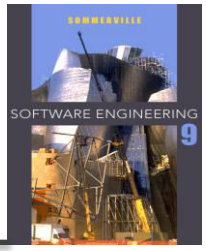


Process switching



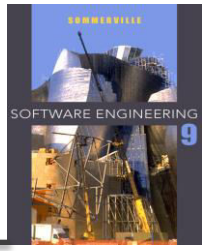
- The scheduler chooses the next process to be executed by the processor. This depends on a scheduling strategy which may take the process priority into account.
- The resource manager allocates memory and a processor for the process to be executed.
- The dispatcher takes the process from ready list, loads it onto a processor and starts execution.

Scheduling strategies



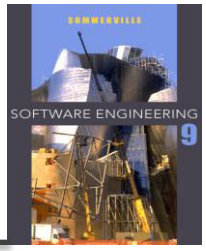
- Non pre-emptive scheduling
  - Once a process has been scheduled for execution, it runs to completion or until it is blocked for some reason (e.g. waiting for I/O).
- Pre-emptive scheduling
  - The execution of an executing processes may be stopped if a higher priority process requires service.
- Scheduling algorithms
  - Round-robin;
  - Rate monotonic;
  - Shortest deadline first.

Key points



- An embedded software system is part of a hardware/software system that reacts to events in its environment. The software is 'embedded' in the hardware. Embedded systems are normally real-time systems.
- A real-time system is a software system that must respond to events in real time. System correctness does not just depend on the results it produces, but also on the time when these results are produced.
- Real-time systems are usually implemented as a set of communicating processes that react to stimuli to produce responses.
- State models are an important design representation for embedded real-time systems. They are used to show how the system reacts to its environment as events trigger changes of state in the system.

Key points



- There are several standard patterns that can be observed in different types of embedded system. These include a pattern for monitoring the system's environment for adverse events, a pattern for actuator control and a data-processing pattern.
- Designers of real-time systems have to do a timing analysis, which is driven by the deadlines for processing and responding to stimuli. They have to decide how often each process in the system should run and the expected and worst-case execution time for processes.
- A real-time operating system is responsible for process and resource management. It always includes a scheduler, which is the component responsible for deciding which process should be scheduled for execution.