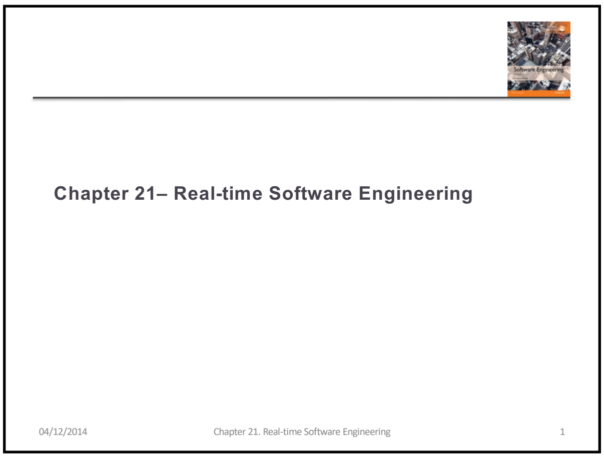
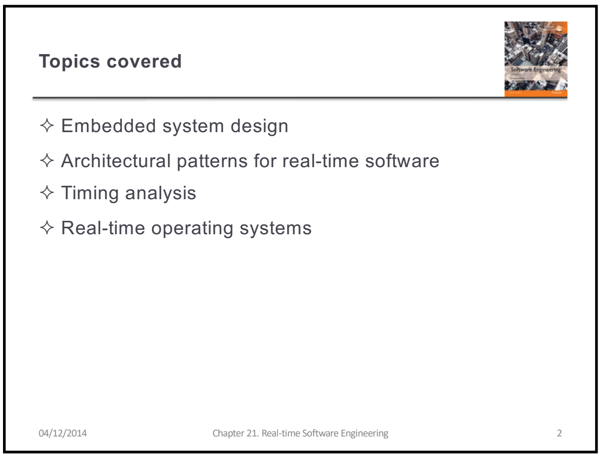


Chapter 21– Real-time Software Engineering

1



Topics covered

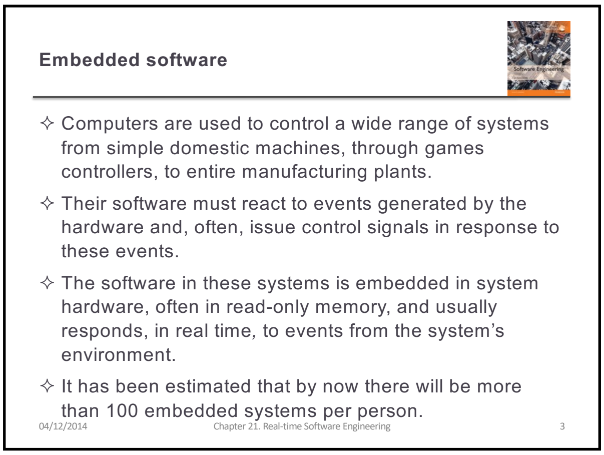
✧ Embedded system design

✧ Architectural patterns for real-time software

✧ Timing analysis

✧ Real-time operating systems

2



Embedded software

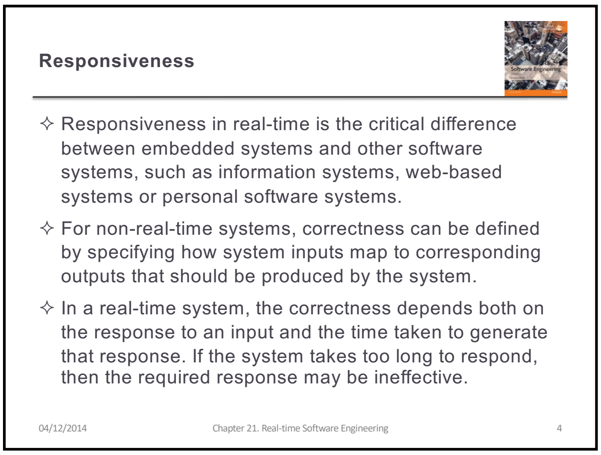
✧ Computers are used to control a wide range of systems from simple domestic machines, through games controllers, to entire manufacturing plants.

✧ 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.

✧ It has been estimated that by now there will be more than 100 embedded systems per person.

3



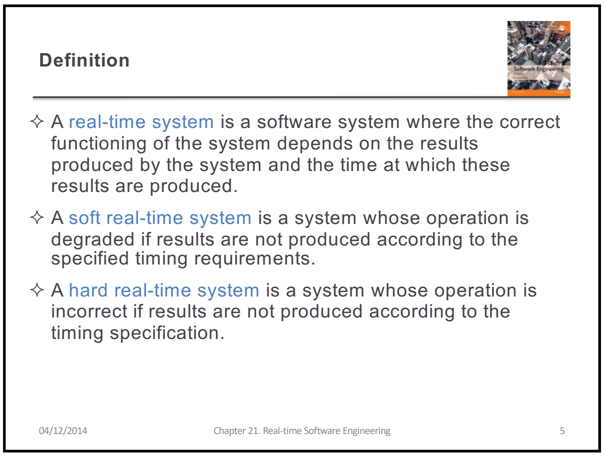
Responsiveness

✧ 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.

4



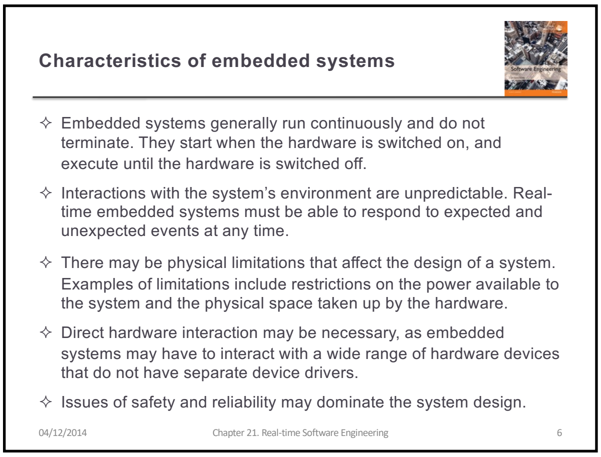
Definition

✧ 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.

5



Characteristics of embedded systems

✧ Embedded systems generally run continuously and do not terminate. They start when the hardware is switched on, and execute until the hardware is switched off.

✧ Interactions with the system's environment are unpredictable. Real-time embedded systems must be able to respond to expected and unexpected events at any time.

✧ There may be physical limitations that affect the design of a system. Examples of limitations include restrictions on the power available to the system and the physical space taken up by the hardware.

✧ Direct hardware interaction may be necessary, as embedded systems may have to interact with a wide range of hardware devices that do not have separate device drivers.

✧ Issues of safety and reliability may dominate the system design.

6

Embedded system design

7

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.

8

Reactive systems

- ✧ Real-time systems are often considered to be 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.

9

Stimuli and responses for a burglar alarm system

Stimulus	Response
Clear alarms	Switch off all active alarms; switch off all lights that have been switched on.
Console panic button positive	Initiate alarm; turn on lights around console; call police.
Power supply failure	Call service technician.
Sensor failure	Call service technician.
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.

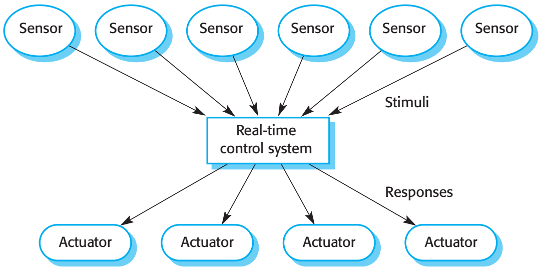
10

Types of stimuli

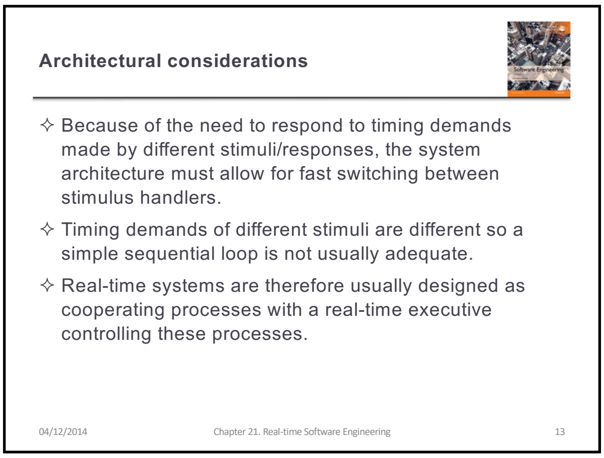
- ✧ Stimuli come from sensors in the systems environment and from actuators controlled by the system.
  - *Periodic stimuli* These occur at predictable time intervals.
    - For example, the system may examine a sensor every 50 milliseconds and take action (respond) depending on that sensor value (the stimulus).
  - *Aperiodic stimuli* These occur irregularly and unpredictably and may be signalled using the computer's interrupt mechanism.
    - An example of such a stimulus would be an interrupt indicating that an I/O transfer was complete and that data was available in a buffer.

11

A general model of an embedded real-time system



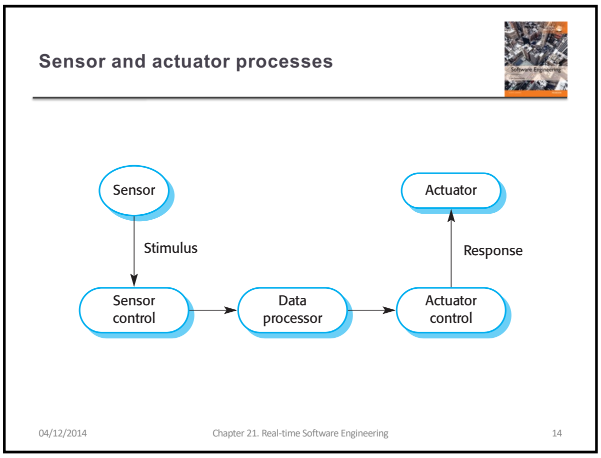
12



Architectural considerations

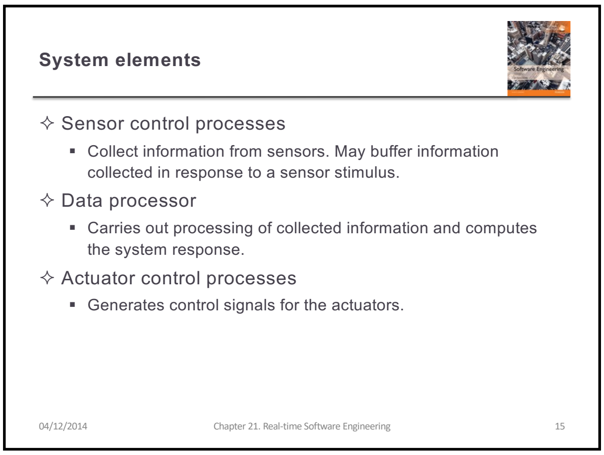
- ✧ Because of the need to respond to timing demands made by different stimuli/responses, the system architecture must allow for fast switching between stimulus handlers.
- ✧ Timing demands of different stimuli are different so a simple sequential loop is not usually adequate.
- ✧ Real-time systems are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

13



Sensor and actuator processes

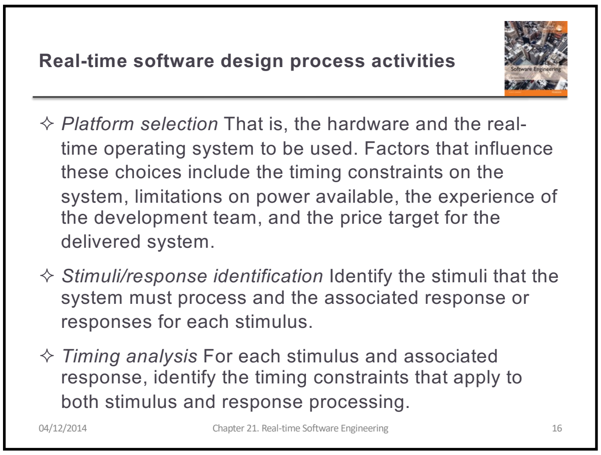
14



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.

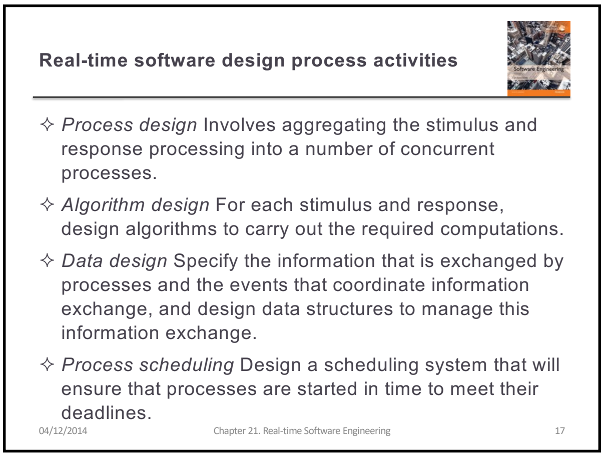
15



Real-time software design process activities

- ✧ *Platform selection* That is, the hardware and the real-time operating system to be used. Factors that influence these choices include the timing constraints on the system, limitations on power available, the experience of the development team, and the price target for the delivered system.
- ✧ *Stimuli/response identification* Identify the stimuli that the system must process and the associated response or responses for each stimulus.
- ✧ *Timing analysis* For each stimulus and associated response, identify the timing constraints that apply to both stimulus and response processing.

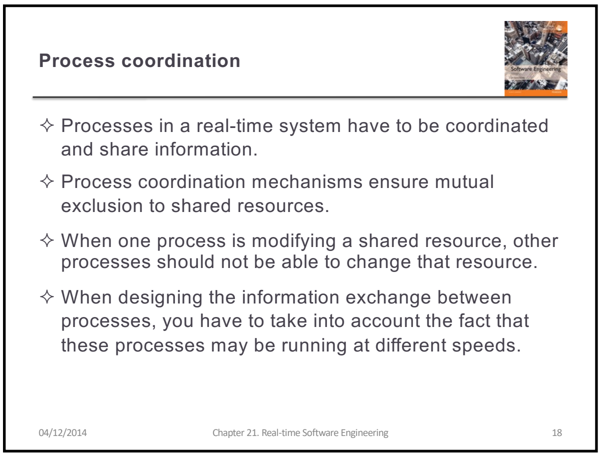
16



Real-time software design process activities

- ✧ *Process design* Involves aggregating the stimulus and response processing into a number of concurrent processes.
- ✧ *Algorithm design* For each stimulus and response, design algorithms to carry out the required computations.
- ✧ *Data design* Specify the information that is exchanged by processes and the events that coordinate information exchange, and design data structures to manage this information exchange.
- ✧ *Process scheduling* Design a scheduling system that will ensure that processes are started in time to meet their deadlines.

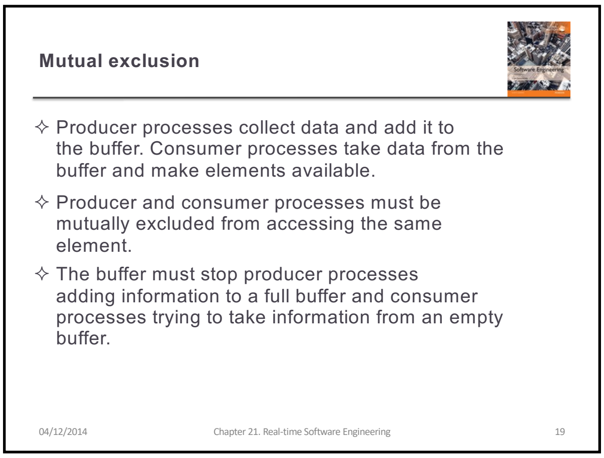
17



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.

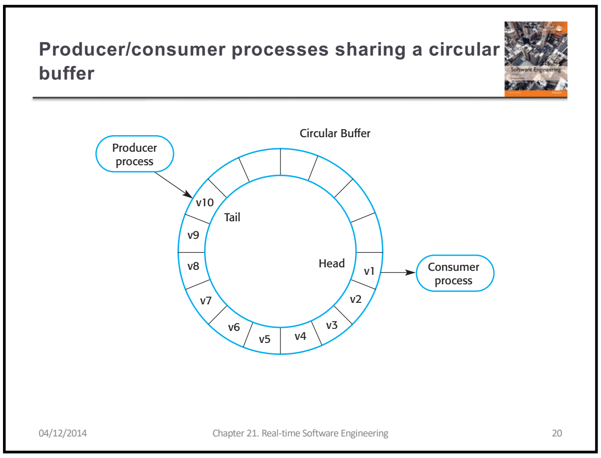
18



Mutual exclusion

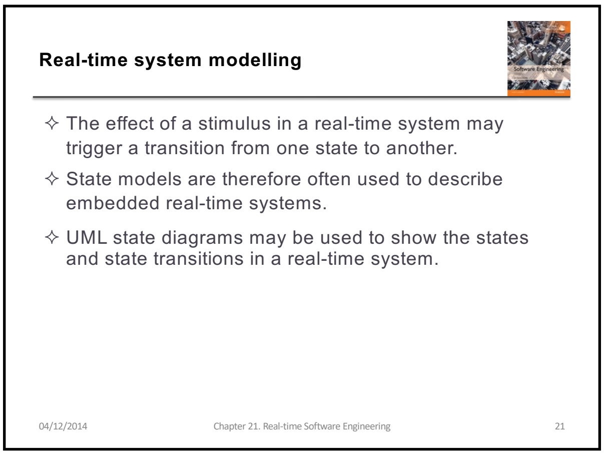
- ✧ Producer processes collect data and add it to the buffer. Consumer processes take data from the buffer and make elements available.
- ✧ Producer and consumer processes must be mutually excluded from accessing the same element.
- ✧ The buffer must stop producer processes adding information to a full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

19



Producer/consumer processes sharing a circular buffer

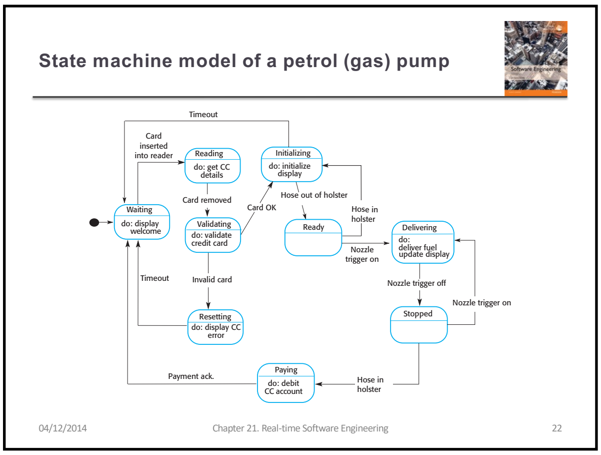
20



Real-time system modelling

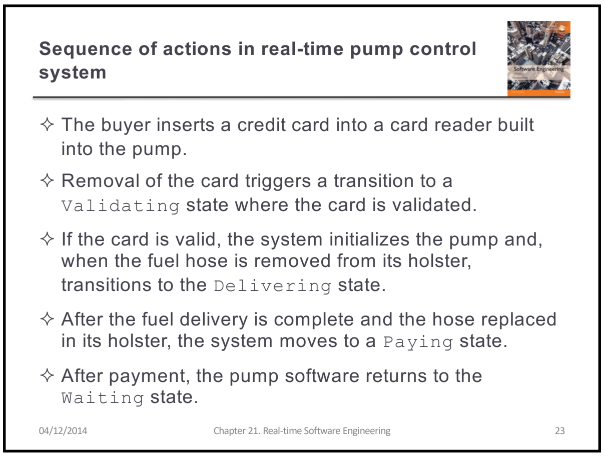
- ✧ The effect of a stimulus in a real-time system may trigger a transition from one state to another.
- ✧ State models are therefore often used to describe embedded real-time systems.
- ✧ UML state diagrams may be used to show the states and state transitions in a real-time system.

21



State machine model of a petrol (gas) pump

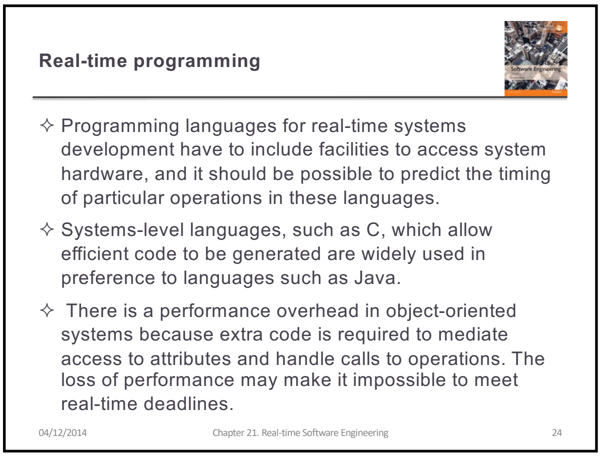
22



Sequence of actions in real-time pump control system

- ✧ The buyer inserts a credit card into a card reader built into the pump.
- ✧ Removal of the card triggers a transition to a `Validating` state where the card is validated.
- ✧ If the card is valid, the system initializes the pump and, when the fuel hose is removed from its holster, transitions to the `Delivering` state.
- ✧ After the fuel delivery is complete and the hose replaced in its holster, the system moves to a `Paying` state.
- ✧ After payment, the pump software returns to the `Waiting` state.

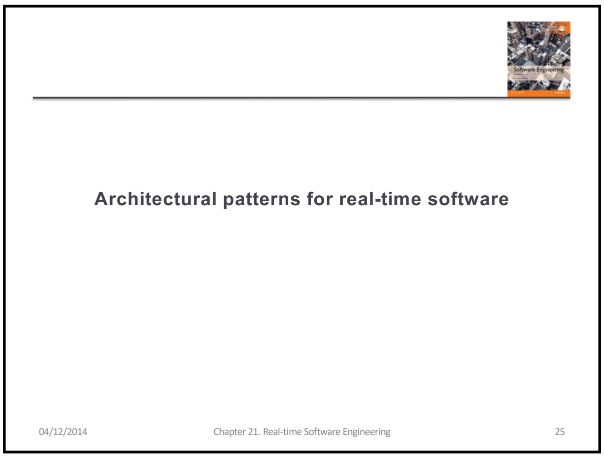
23



Real-time programming

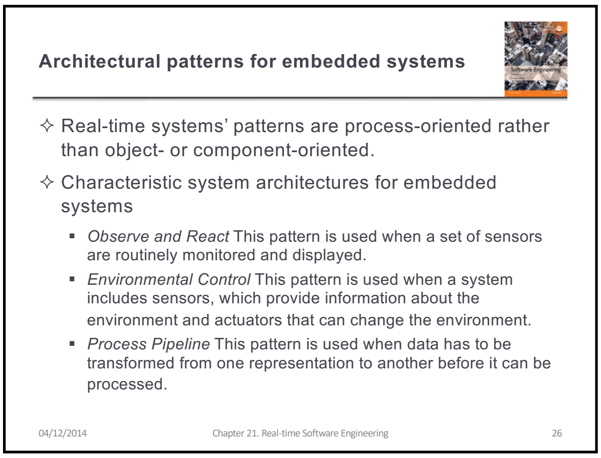
- ✧ Programming languages for real-time systems development have to include facilities to access system hardware, and it should be possible to predict the timing of particular operations in these languages.
- ✧ Systems-level languages, such as C, which allow efficient code to be generated are widely used in preference to languages such as Java.
- ✧ There is a performance overhead in object-oriented systems because extra code is required to mediate access to attributes and handle calls to operations. The loss of performance may make it impossible to meet real-time deadlines.

24



Architectural patterns for real-time software

25



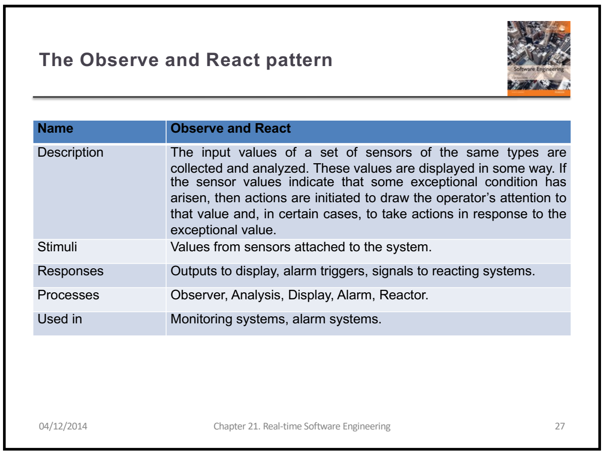
Architectural patterns for embedded systems

✧ Real-time systems' patterns are process-oriented rather than object- or component-oriented.

✧ Characteristic system architectures for embedded systems

- *Observe and React* This pattern is used when a set of sensors are routinely monitored and displayed.
- *Environmental Control* This pattern is used when a system includes sensors, which provide information about the environment and actuators that can change the environment.
- *Process Pipeline* This pattern is used when data has to be transformed from one representation to another before it can be processed.

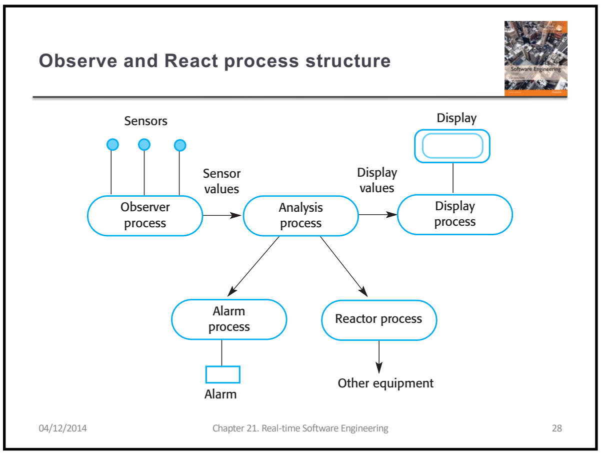
26



The Observe and React pattern

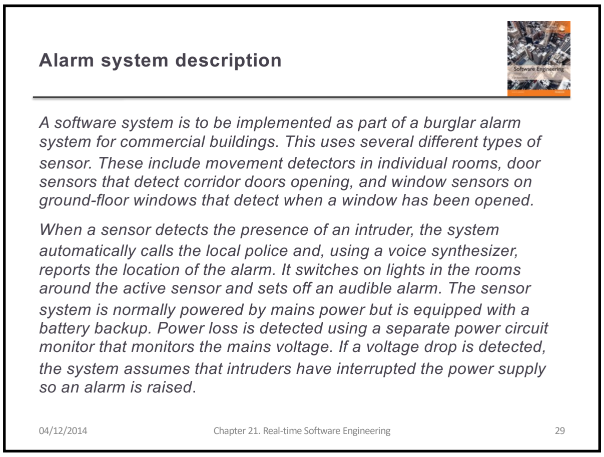
Name	Observe and React
Description	The input values of a set of sensors of the same types are collected and analyzed. These values are displayed in some way. If the sensor values indicate that some exceptional condition has arisen, then actions are initiated to draw the operator's attention to that value 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.

27



Observe and React process structure

28

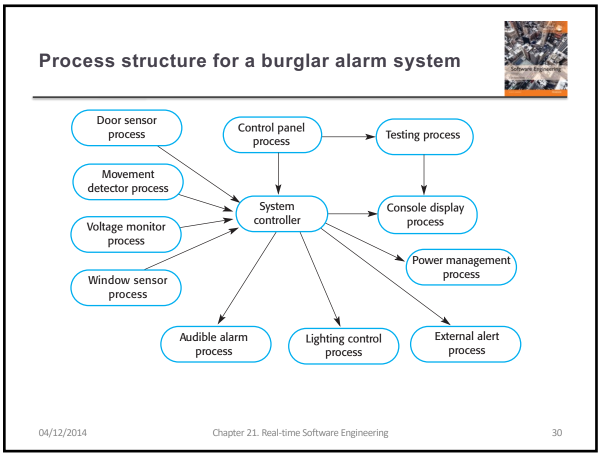


Alarm system description

A software system is to be implemented as part of a burglar alarm system for commercial buildings. This uses several different types of sensor. These include movement detectors in individual rooms, door sensors that detect corridor doors opening, and window sensors on ground-floor windows that detect when a window has been opened.

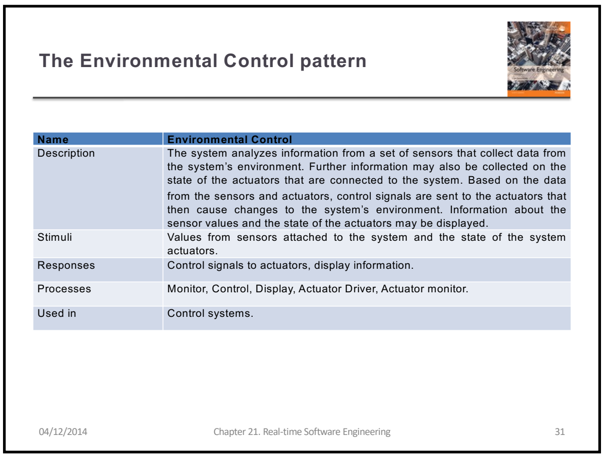
When a sensor detects the presence of an intruder, the system automatically calls the local police and, using a voice synthesizer, reports the location of the alarm. It switches on lights in the rooms around the active sensor and sets off an audible alarm. The sensor system is normally powered by mains power but is equipped with a battery backup. Power loss is detected using a separate power circuit monitor that monitors the mains voltage. If a voltage drop is detected, the system assumes that intruders have interrupted the power supply so an alarm is raised.

29



Process structure for a burglar alarm system

30

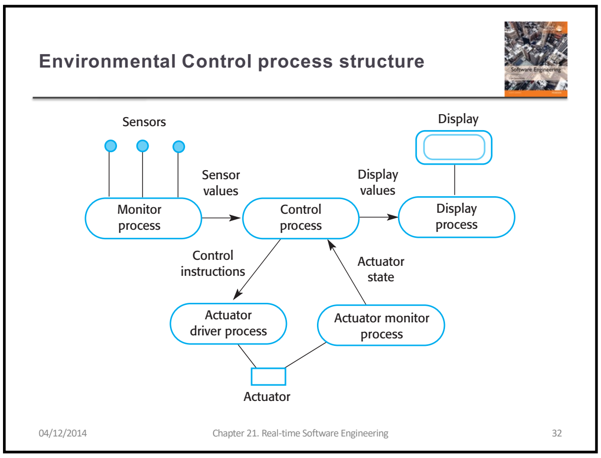


The Environmental Control pattern

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.

04/12/2014 Chapter 21. Real-time Software Engineering 31

31



Environmental Control process structure

Sensors

Display

Sensor values

Display values

Monitor process

Control process

Display process

Control instructions

Actuator state

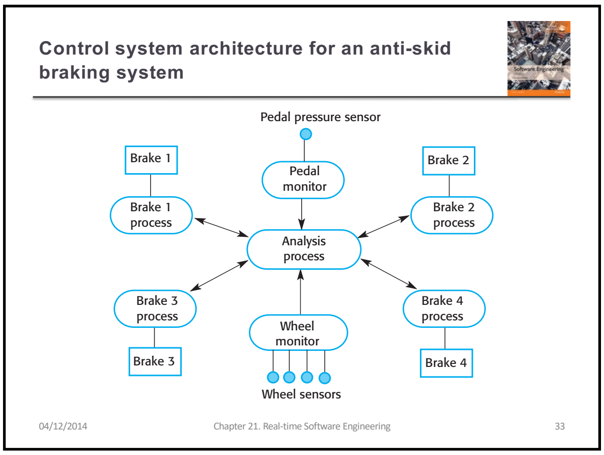
Actuator driver process

Actuator monitor process

Actuator

04/12/2014 Chapter 21. Real-time Software Engineering 32

32



Control system architecture for an anti-skid braking system

Pedal pressure sensor

Brake 1

Pedal monitor

Brake 2

Brake 1 process

Brake 2 process

Analysis process

Brake 3 process

Brake 4 process

Wheel monitor

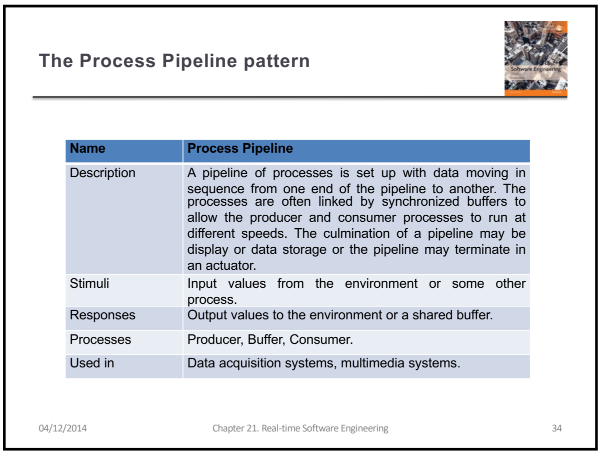
Brake 3

Brake 4

Wheel sensors

04/12/2014 Chapter 21. Real-time Software Engineering 33

33

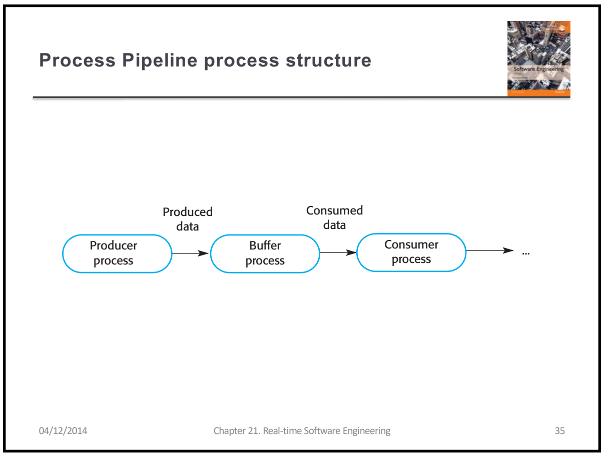


The Process Pipeline pattern

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.

04/12/2014 Chapter 21. Real-time Software Engineering 34

34



Process Pipeline process structure

Produced data

Consumed data

Producer process

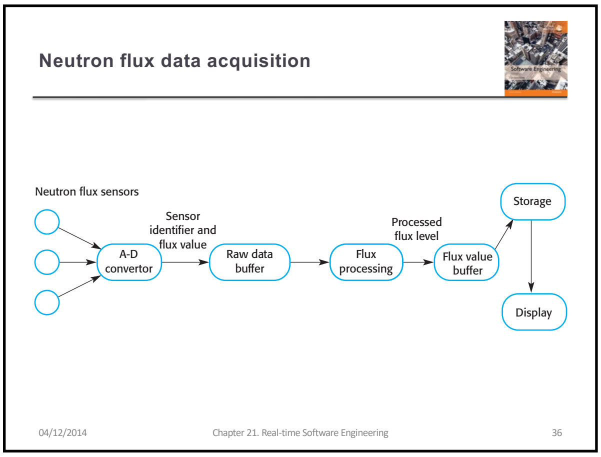
Buffer process

Consumer process

...

04/12/2014 Chapter 21. Real-time Software Engineering 35

35



Neutron flux data acquisition

Neutron flux sensors

Sensor identifier and flux value

Processed flux level

Storage

A-D convertor

Raw data buffer

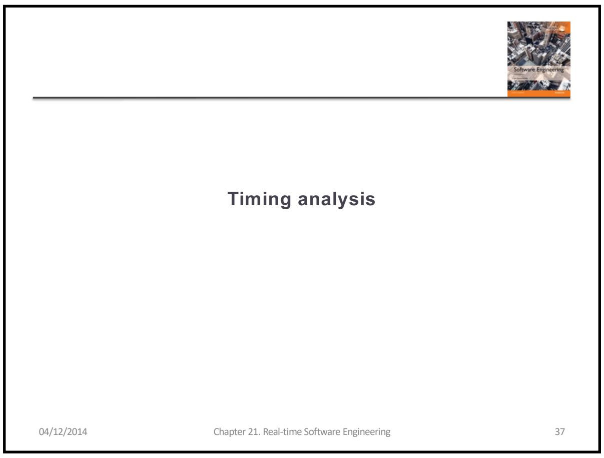
Flux processing

Flux value buffer

Display

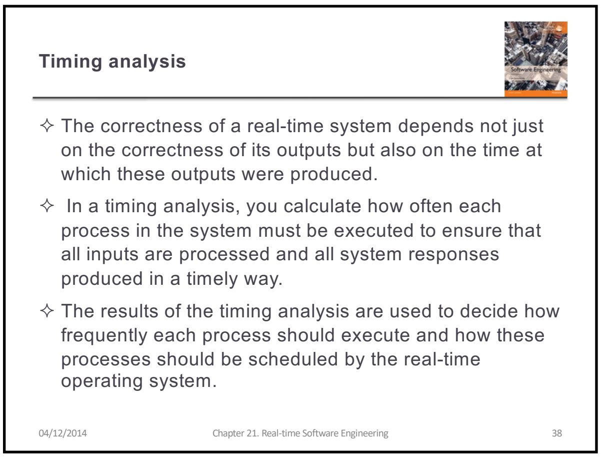
04/12/2014 Chapter 21. Real-time Software Engineering 36

36



Timing analysis

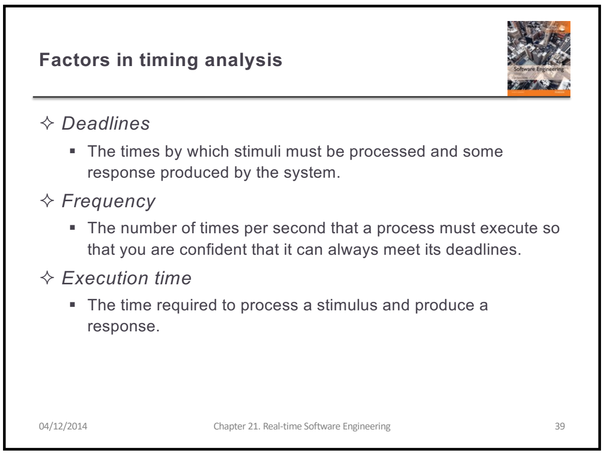
37



Timing analysis

- ✧ The correctness of a real-time system depends not just on the correctness of its outputs but also on the time at which these outputs were produced.
- ✧ In a timing analysis, you calculate how often each process in the system must be executed to ensure that all inputs are processed and all system responses produced in a timely way.
- ✧ The results of the timing analysis are used to decide how frequently each process should execute and how these processes should be scheduled by the real-time operating system.

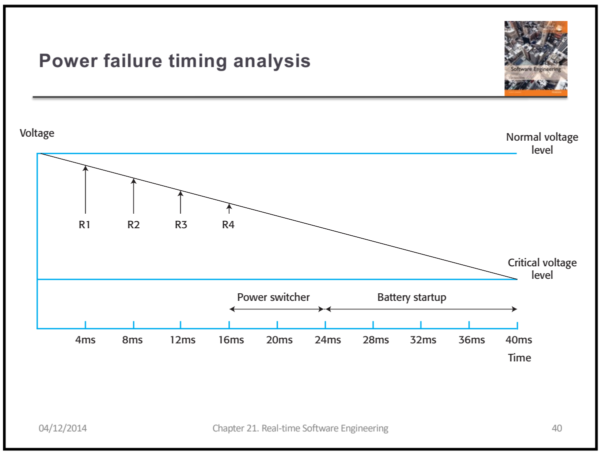
38



Factors in timing analysis

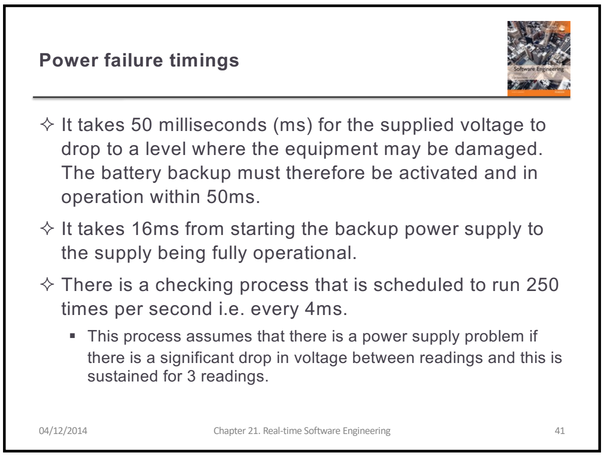
- ✧ *Deadlines*
  - The times by which stimuli must be processed and some response produced by the system.
- ✧ *Frequency*
  - The number of times per second that a process must execute so that you are confident that it can always meet its deadlines.
- ✧ *Execution time*
  - The time required to process a stimulus and produce a response.

39



Power failure timing analysis

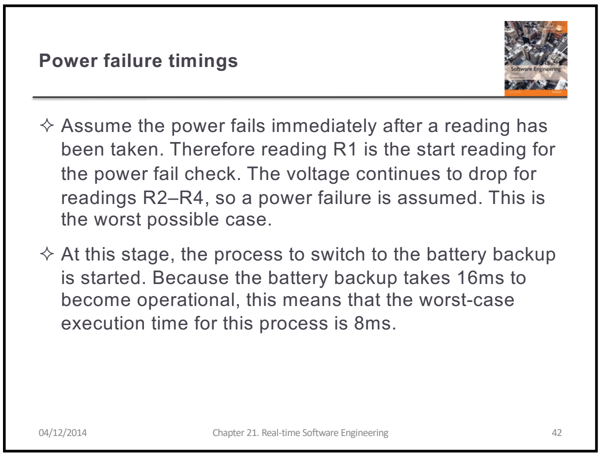
40



Power failure timings

- ✧ It takes 50 milliseconds (ms) for the supplied voltage to drop to a level where the equipment may be damaged. The battery backup must therefore be activated and in operation within 50ms.
- ✧ It takes 16ms from starting the backup power supply to the supply being fully operational.
- ✧ There is a checking process that is scheduled to run 250 times per second i.e. every 4ms.
  - This process assumes that there is a power supply problem if there is a significant drop in voltage between readings and this is sustained for 3 readings.

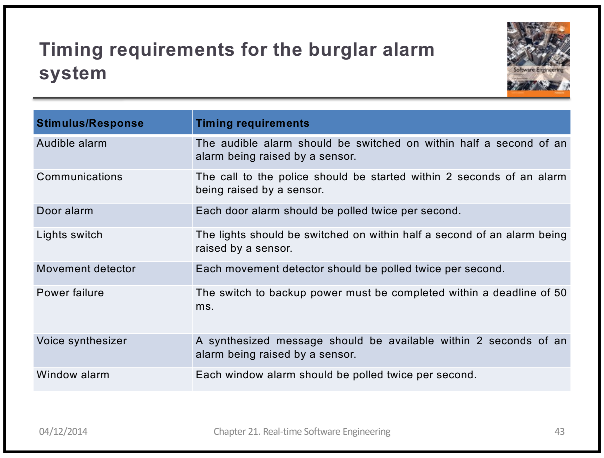
41



Power failure timings

- ✧ Assume the power fails immediately after a reading has been taken. Therefore reading R1 is the start reading for the power fail check. The voltage continues to drop for readings R2–R4, so a power failure is assumed. This is the worst possible case.
- ✧ At this stage, the process to switch to the battery backup is started. Because the battery backup takes 16ms to become operational, this means that the worst-case execution time for this process is 8ms.

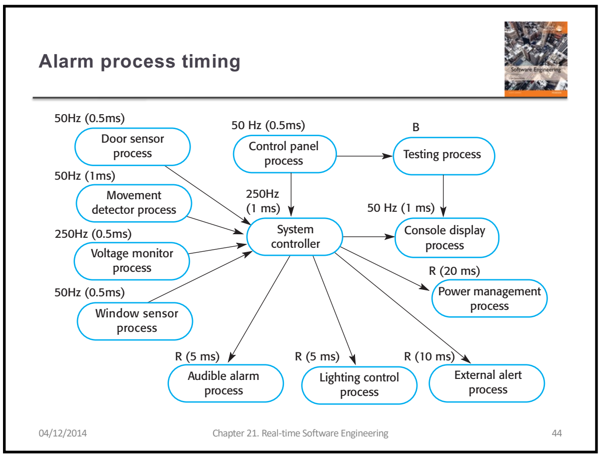
42



Timing requirements for the burglar alarm system

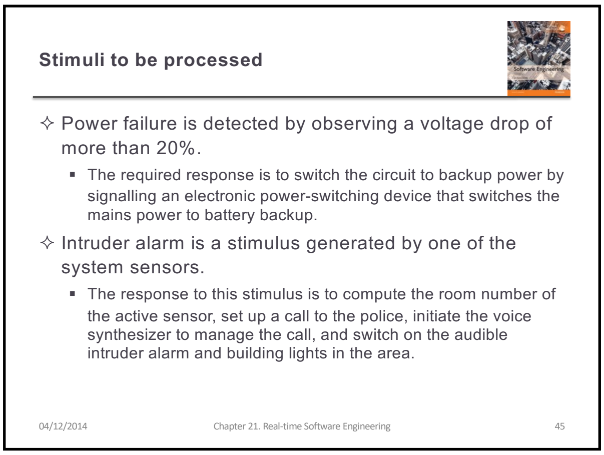
Stimulus/Response	Timing requirements
Audible alarm	The audible alarm should be switched on within half a second of an alarm being raised by a sensor.
Communications	The call to the police should be started within 2 seconds of an alarm being raised by a sensor.
Door alarm	Each door alarm should be polled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

43



Alarm process timing

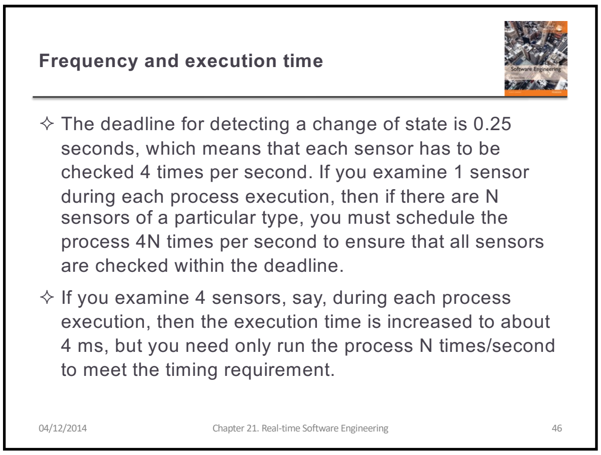
44



Stimuli to be processed

- ✧ Power failure is detected by observing a voltage drop of more than 20%.
  - The required response is to switch the circuit to backup power by signalling an electronic power-switching device that switches the mains power to battery backup.
- ✧ Intruder alarm is a stimulus generated by one of the system sensors.
  - The response to this stimulus is to compute the room number of the active sensor, set up a call to the police, initiate the voice synthesizer to manage the call, and switch on the audible intruder alarm and building lights in the area.

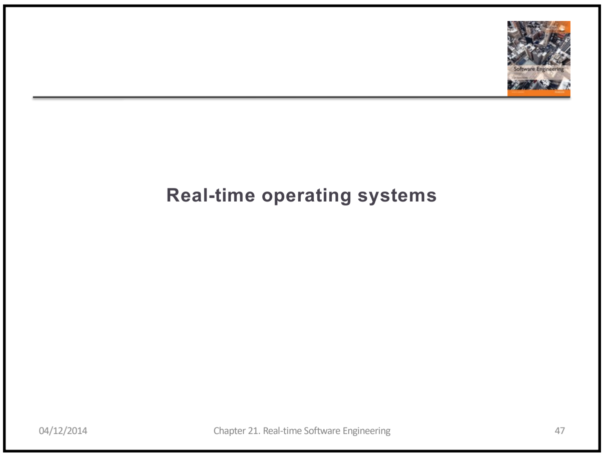
45



Frequency and execution time

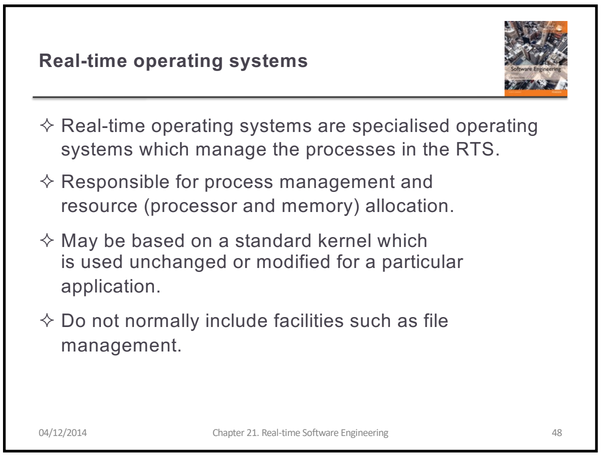
- ✧ The deadline for detecting a change of state is 0.25 seconds, which means that each sensor has to be checked 4 times per second. If you examine 1 sensor during each process execution, then if there are N sensors of a particular type, you must schedule the process 4N times per second to ensure that all sensors are checked within the deadline.
- ✧ If you examine 4 sensors, say, during each process execution, then the execution time is increased to about 4 ms, but you need only run the process N times/second to meet the timing requirement.

46



Real-time operating systems

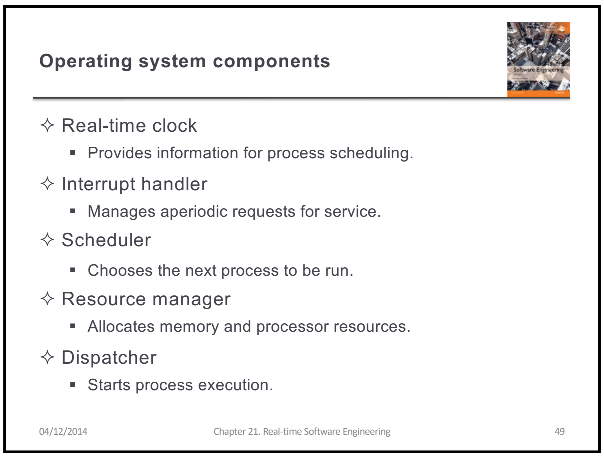
47



Real-time operating systems

- ✧ Real-time operating systems are specialised operating systems which manage the processes in the RTS.
- ✧ Responsible for process management and resource (processor and memory) allocation.
- ✧ May be based on a standard kernel which is used unchanged or modified for a particular application.
- ✧ Do not normally include facilities such as file management.

48



Operating system components

✧ Real-time clock
- Provides information for process scheduling.

✧ Interrupt handler
- Manages aperiodic requests for service.

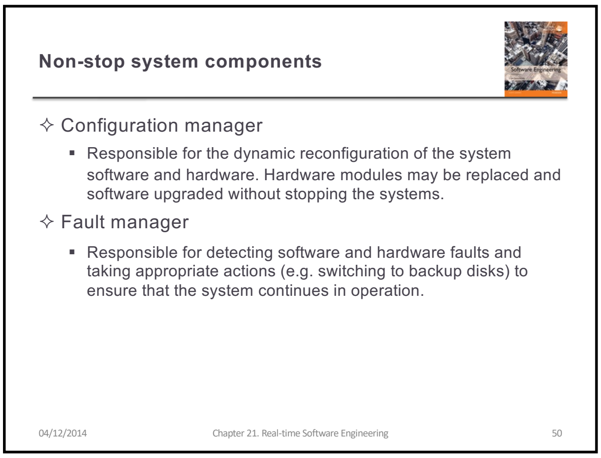
✧ Scheduler
- Chooses the next process to be run.

✧ Resource manager
- Allocates memory and processor resources.

✧ Dispatcher
- Starts process execution.

04/12/2014 Chapter 21. Real-time Software Engineering 49

49



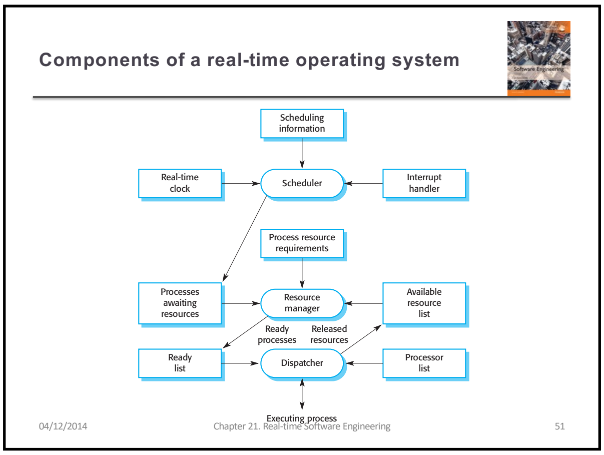
Non-stop system components

✧ 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.

04/12/2014 Chapter 21. Real-time Software Engineering 50

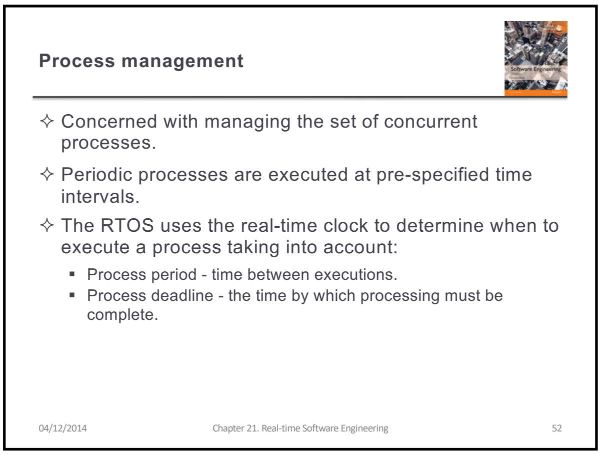
50



Components of a real-time operating system

04/12/2014 Chapter 21. Real-time Software Engineering 51

51



Process management

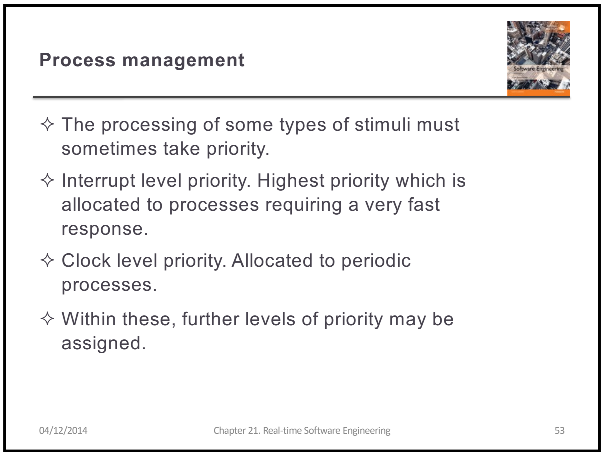
✧ Concerned with managing the set of concurrent processes.

✧ 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.

04/12/2014 Chapter 21. Real-time Software Engineering 52

52



Process management

✧ The processing of some types of stimuli must sometimes take priority.

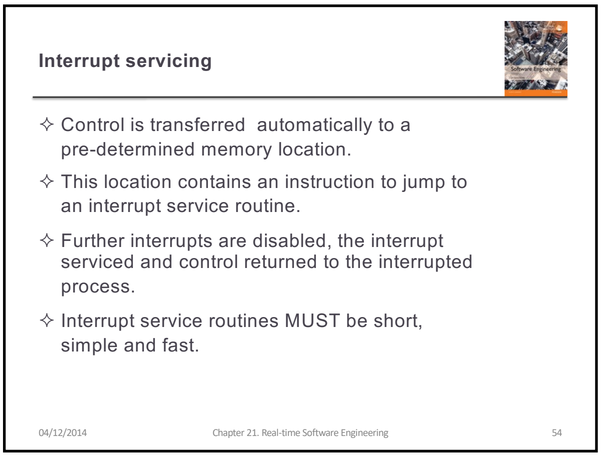
✧ Interrupt level priority. Highest priority which is allocated to processes requiring a very fast response.

✧ Clock level priority. Allocated to periodic processes.

✧ Within these, further levels of priority may be assigned.

04/12/2014 Chapter 21. Real-time Software Engineering 53

53



Interrupt servicing

✧ Control is transferred automatically to a pre-determined memory location.

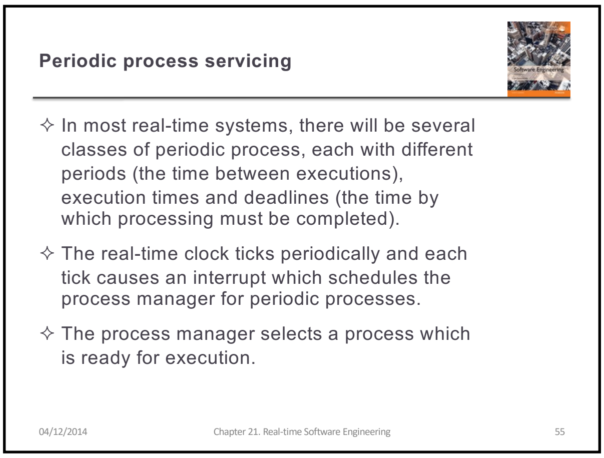
✧ This location contains an instruction to jump to an interrupt service routine.

✧ Further interrupts are disabled, the interrupt serviced and control returned to the interrupted process.

✧ Interrupt service routines MUST be short, simple and fast.

04/12/2014 Chapter 21. Real-time Software Engineering 54

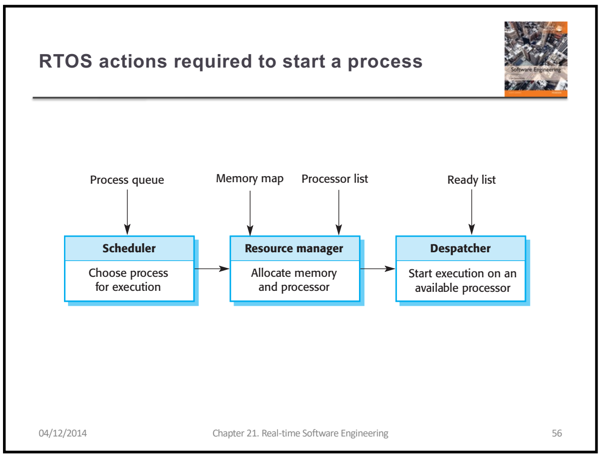
54



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.

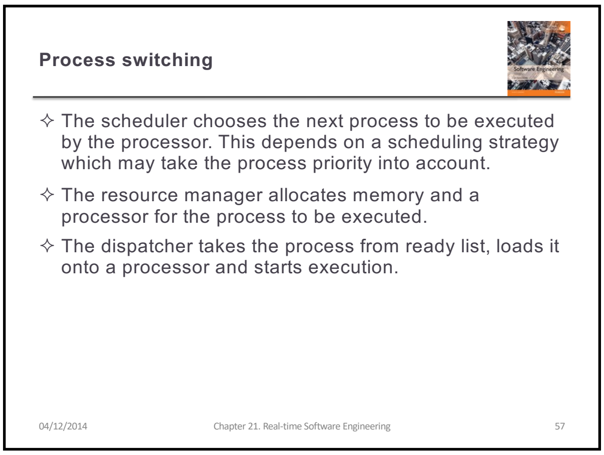
55



RTOS actions required to start a process

Process queue
Memory map
Processor list
Ready list
Scheduler
Choose process for execution
Resource manager
Allocate memory and processor
Despatcher
Start execution on an available processor

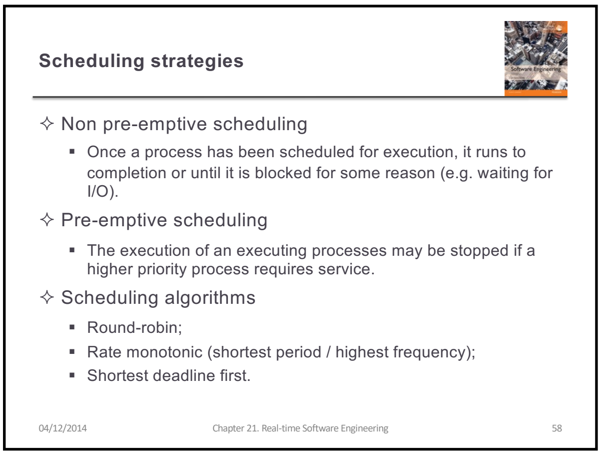
56



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.

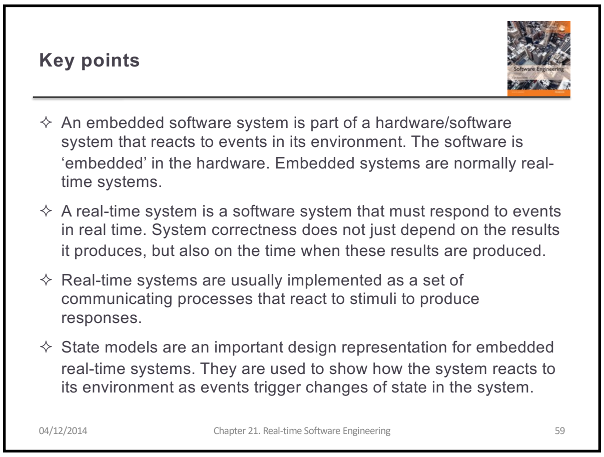
57



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 period / highest frequency);
  - Shortest deadline first.

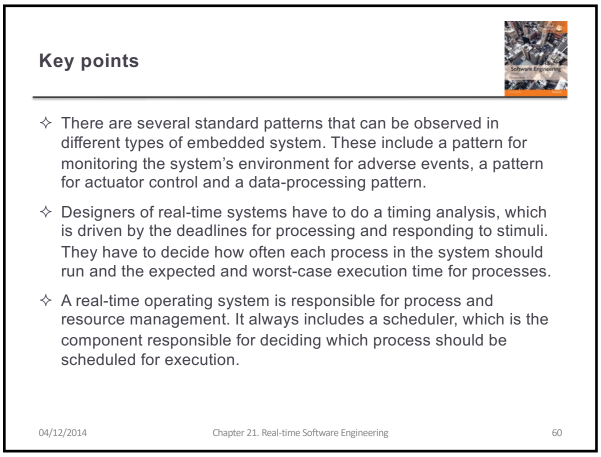
58



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.

59



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.

60