**modular decomposition styles**

styles of decomposing sub-systems into modules

no rigid distinction between system organisation and modular decomposition

**sub-systems and modules**

a sub-system is a system in its own right whose operation is independent of the services provided by other sub-systems

a module is a system component that provides services to other components but would not normally be considered as a seperate system

**modular decomposition**

another structural level where sub-systems are decomposed into modules

two modular decomposition models covered

an object model where the system is decomposed into interacting object

a pipeline or data-flow model where the system is decomposed into functional modules which transform inputs to outputs'

if possible,decisions about concurrency should be delayed until modules are implemented

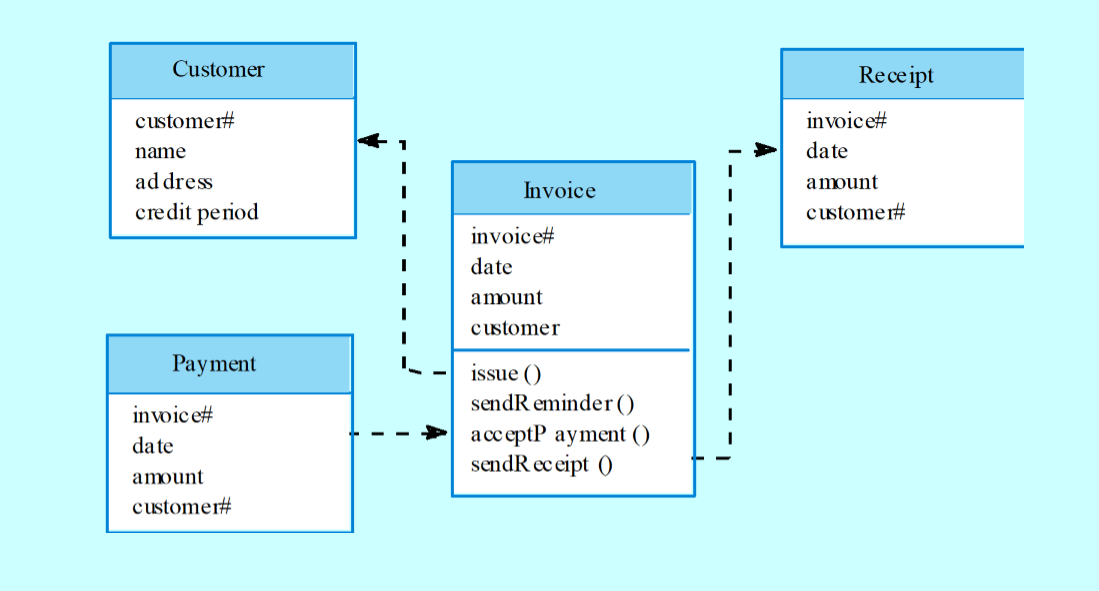
**object models**

structure the system into a set of loosely coupled objects with well-defined interfaces.

object-oriented decomposition is concerned with identifying object classes,their attributes and operations

when implemented, objects are created from these classes and some control model used to coordinate object operations.

eg. invoice processing system



**object model advantages**

objects are loosely coupled so their implementation can be modified without affecting other objects

the objects may reflect real-world entities.

oo implementation languages are widely used

however,object interface changes may cause problems and complex entities may be hard to represent as objects

**function-oriented pipelining**

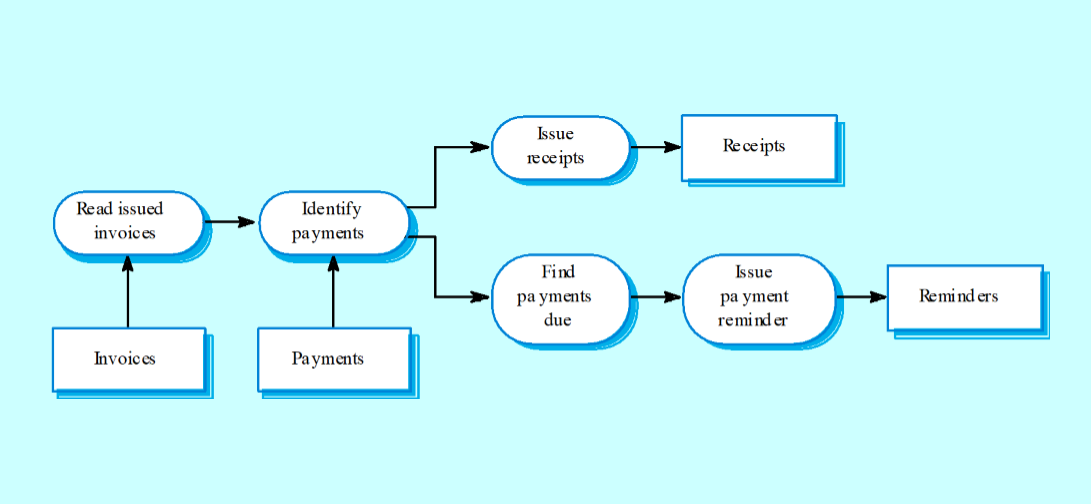
functional transformations process their inputs to produce outputs

may be referred to as a pipe and filter model(as in UNIX shell).

variants of this approach are very common.when transformations are sequential,this is a batch sequential model which is extensively used in data processing systems.

not really suitable for interactive systems.

eg. invoice processing system



**pipeline model advantages**

supports transformation reuse

intuitive organisation for stakeholder communication

easy to ass new transformations

relatively simple to implement as either a concurrent or sequential system.

however, requires a common format for data transfer along the pipeline and difficult to support event-based interaction

**control styles**

are concerned with the control flow between sub-systems.distinct from the system decomposition model.

**centralised control**

one sub-system has overall responsibility for control and starts and stops other sub-systems.

**event-based control**

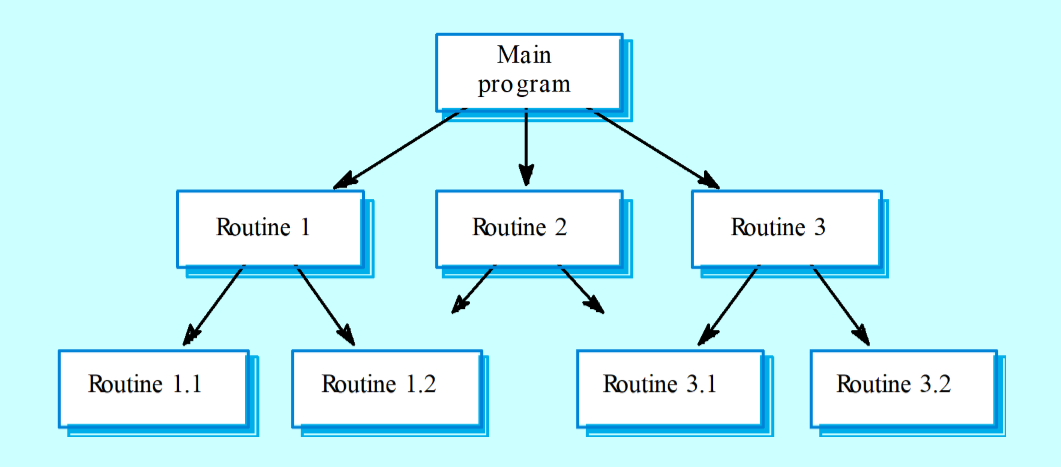
each sub-system can respond to externally generated events from other sub-systems or the system's environment

**centralised control**

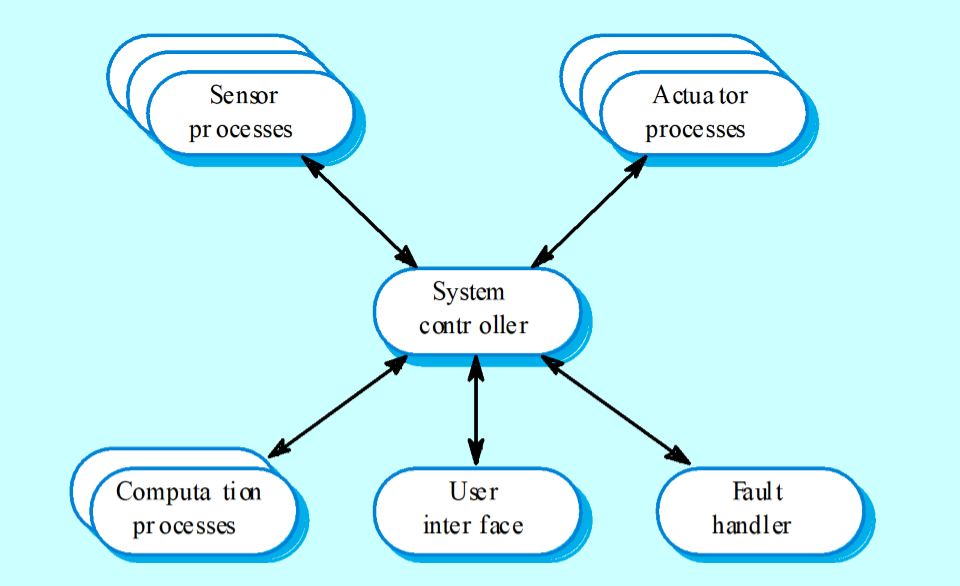
a control sub-system takes responsibility for managing the execution of other sub-systems.

**call-return model**

top-down subroutine model where control starts at the top of a subroutine hierarchy and moves downwards.applicable to sequential systems.



eg. real-time system control



**manager model**

applicable to concurrent systems.one system component controls the stopping,starting and coordination of other system processes.can be implemented in sequential systems as a case statement.

**event-driven systems**

driven by externally generated events where the timing of the event is outwith the control of the sub-systems which process the event.

two principal event-driven models

broadcast models.an event is broadcast to all sub-systems.any sub-system which can handle the event may do so;

interrupt-driven models.used in real-time systems where interrupts are detected by an interrupt handler and passed to some other component for processing.

other event driven models include spreadsheets and production systems.

**broadcast model**

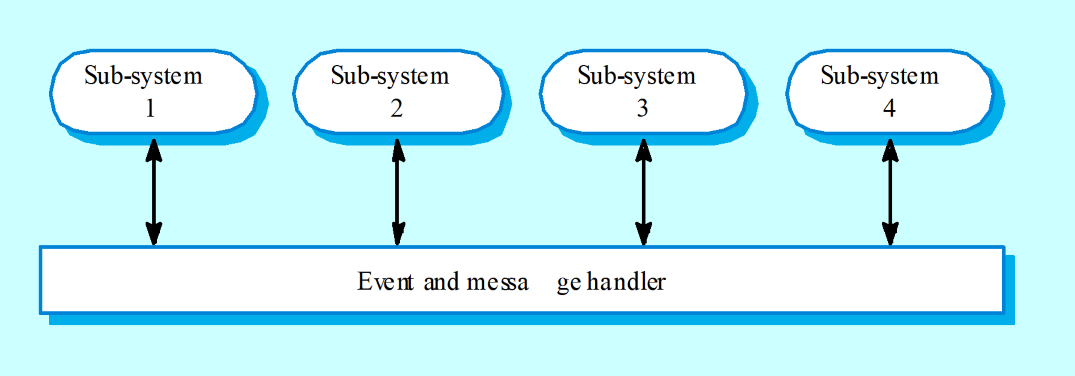
effective in integrating sub-systems on different computers in a network

sub-systems register an interest in specific events.when these occur,control is transferred to the sub-system which can handle the event.

control policy is not embedded in the event and message handler.sub-systems decide on events of interest to them.

however, sub-systems don't know if or when an event will be handled.

eg.selective broadcasting



**interrupt-driven systems**

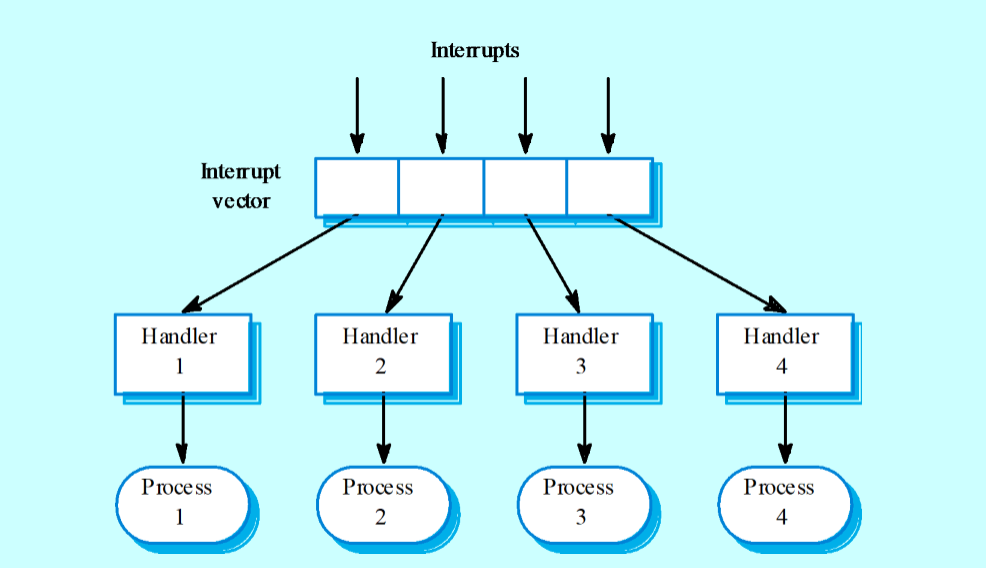
used in real-time systems where fast response to an event is essential

there are knoen interrupt types with a handler defined for each type

each type is associated with a memory location and a hard ware switch causes transfer to its handler.

allows fast response but complex to program and difficult to validate.

eg.interrupt-driven control



**reference architectures**

architectural models may be specific to some application domain.

two types of domain-specific model

generic models which are abstractions from a number of real systems and which encapsulate the principal characteristics of these systems.covered in chapter 13

reference models which are more abstract,idealised model.provied a means of information about that class of system and of comparing different architectures.

generic models are usually bottom-up models;reference models are top-down models.

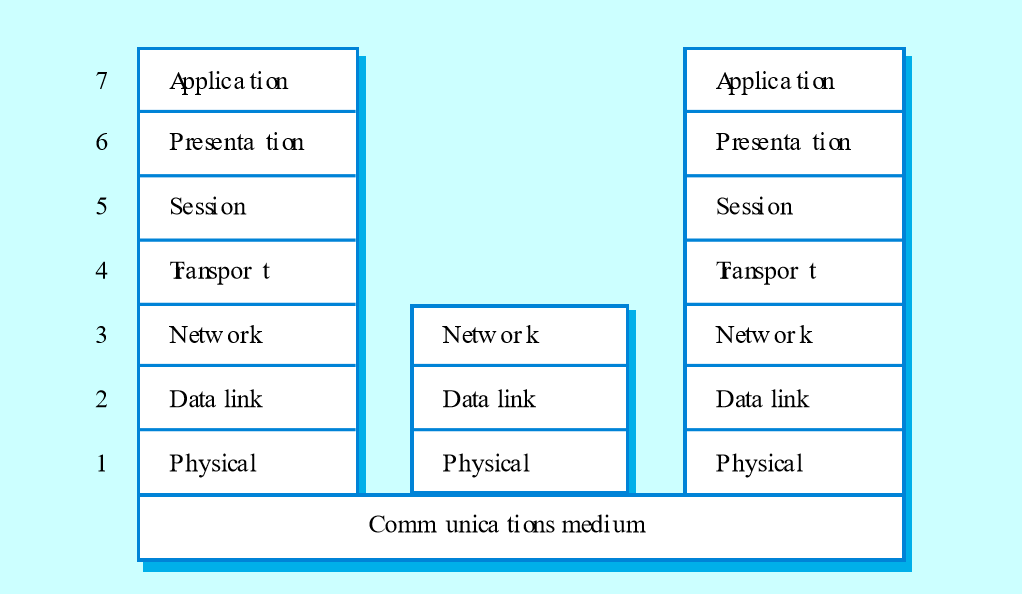
**reference architectures**

reference models are derived from a study of the application domain rather than from existing systems

may be used as a basis for system implementation or to compare different systems.it acts as a standard againts which systems can be evaluated.

OSI model is a layered model for communication systems.

eg.OSI reference model



case reference model

data repository services: storage and management of data items

data integration services: managing groups of entities.

task management services: definition and enaction of process models

messaging services: tool-tool and tool-environment communication

user interface services: user interface development.

eg. the ECMA reference model

