The Design of Software Architecture

- What is it?
- Why study it?
- How do you do it?
  - Design the high-level system structure
  - Design the control model
  - Perform modular decomposition
- Use a domain-specific architecture if one exists.

What is software architecture?

- It is a description of the overall structure of the software system
- It is a description of the sub-systems that make up a system and the framework for sub-system control and communication.

What is architectural design?

- It is the process of figuring out the architecture.
- It is an early stage of the system design process.
- It involves identifying major system components and their communications.

Why study software architectures?

- There are several advantages to designing and documenting an explicit software architecture....
- Stakeholder communication
  - Architecture can focus discussion by system stakeholders
- System analysis
  - Means that analysis of whether the system can meet its non-functional requirements is possible (performance, reliability, maintainability, and usability).
- Large-scale reuse
  - The architecture may be reusable across a range of systems

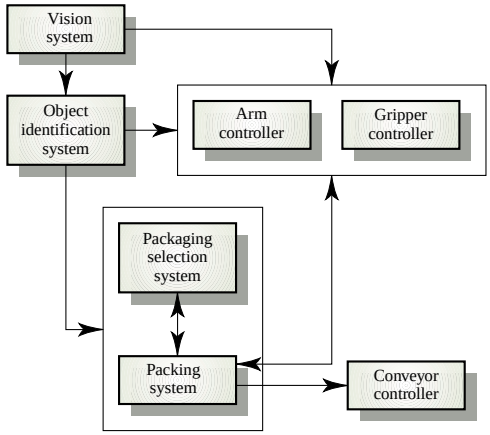
Three high-level activities in the architectural design of a large system

- Design the high-level system structure.
  - Decompose the system into the principal subsystems, and identify the communication and data flow that would be necessary among the subsystems.
- Design the high-level control model.
  - Establish a model of the control relationships among the various subsystems.
- Further decompose each subsystem.
  - Perform modular decomposition, and design the architecture of each subsystem.
- (Large systems rarely conform to a single architectural model.)

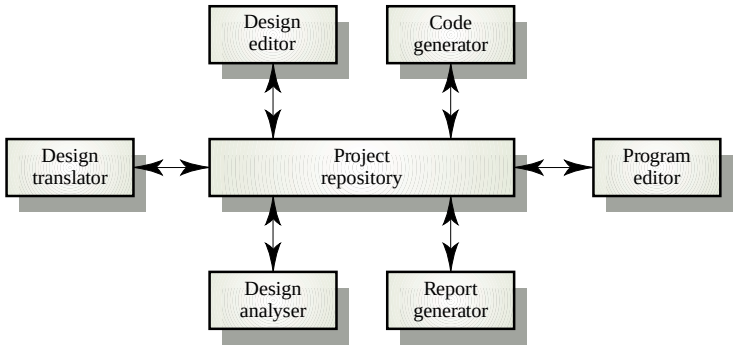
Design the system structure

- Decompose the system identifying the various necessary interacting sub-systems
- The architectural design is normally expressed as a block diagram presenting an overview of the system structure
- More specific models showing how sub-systems share data, are distributed and interface with each other may also be developed

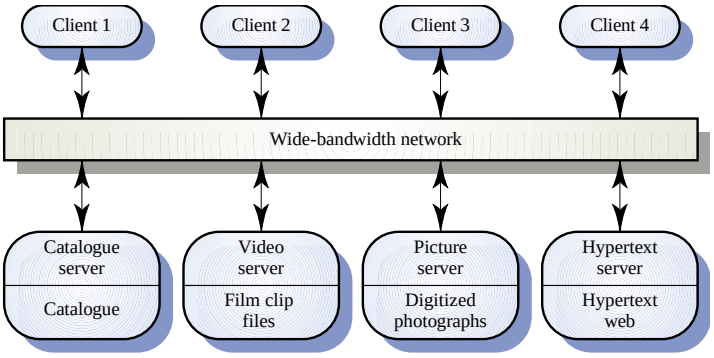
The system structure of a packing robot control system



The system structure of a CASE toolset that uses a data repository model



The system structure of a film and picture library that uses a client-server model



Question

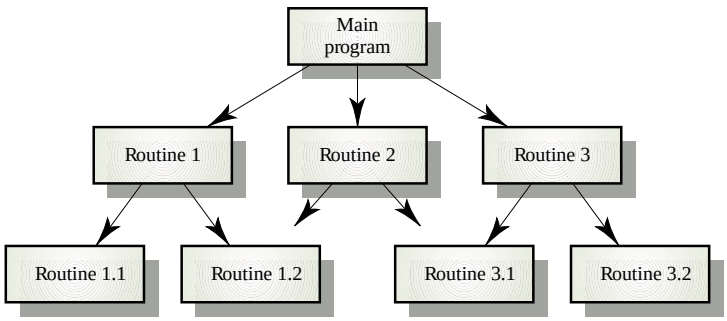
1. (Sommerville 10.3) Suggest an appropriate structural model for the following systems:
   - An automated ticket issuing system used by passengers at a railway station
   - A computer-controlled video conferencing system which allows video, audio, and computer data to be visible to several participants at the same time.
   - A robot floor cleaner that cleans relatively clear spaces such as corridors. The cleaner must be able to sense walls and other obstructions.

Design the control model

1. Identify the control flow between subsystems. Distinct from the system decomposition model
1. Centralized control
   - One subsystem has overall responsibility for control and starts and stops other sub-systems
1. Event-based control
   - Each subsystem can respond to externally generated events from other subsystems or the system's environment

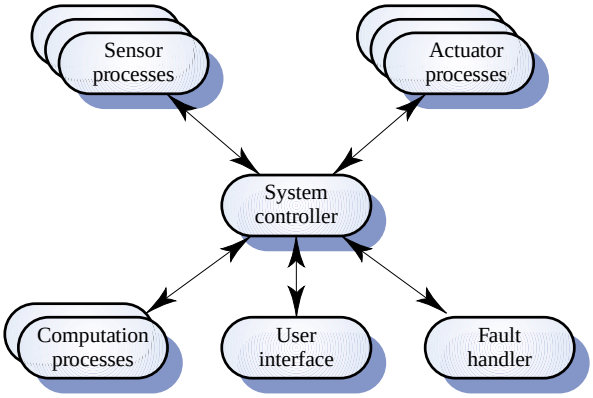
A call-return centralized control model

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



A “manager” centralized control model

In this model, for a real-time system. One system component controls the stopping, starting and coordination of other system processes.

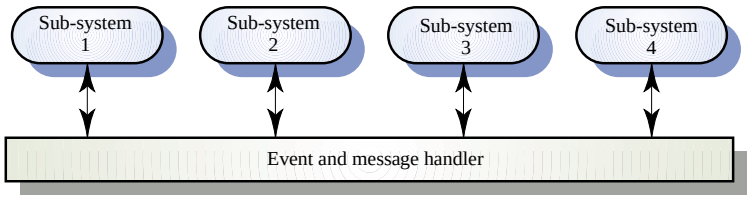


Question

- (Sommerville 10.5) Why is a call-return model of control usually not suitable for real-time systems?

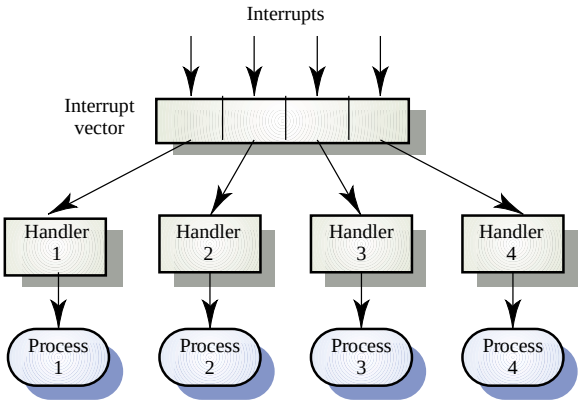
An broadcast event-based control model

Events are broadcasted to all subsystems. Subsystems register an interest in specific events, and receive control when those events occur.



An interrupt-driven event-based control model

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



Question

- (Sommerville 10.6) Suggest an appropriate control model for the following systems:
  - A batch processing system which takes information about hours worked and pay rates and prints salary slips and bank credit transfer information
  - A set of software tools which are produced by different vendors but which must work together
  - A television controller which responds to signals from a remote control unit

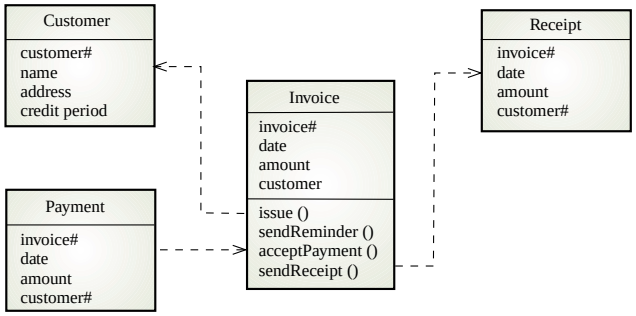
Modular decomposition

- Another structural level where subsystems are decomposed into modules
- Two modular decomposition models will be discussed
  - An object model where the system is decomposed into interacting objects
  - A data-flow model where the system is decomposed into functional modules which transform inputs to outputs. Also known as the pipeline model

Object model modular decompostion

- 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 from a control model used to coordinate object operations

Object model of an invoice processing system

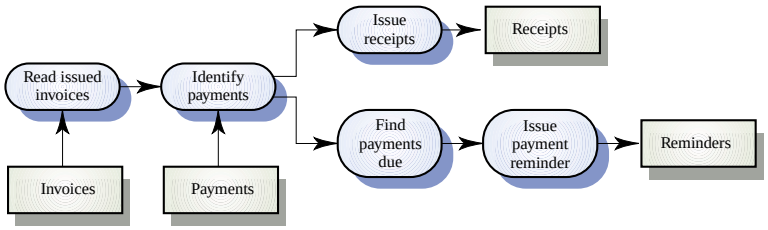


Data-flow model modular decompostion

- 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

Data-flow model of an invoice processing system

This sort of diagram is also known as a data-flow diagram (DFD)



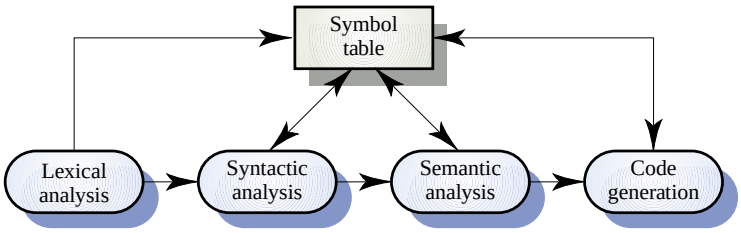
Three high-level activities in the architectural design of a large system

- Design the high-level system structure.
- Design the high-level control model.
- Further decompose each subsystem, performing modular decomposition.

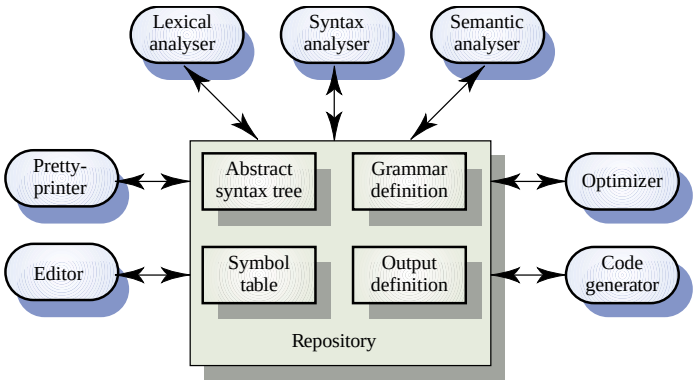
Domain-specific architectures

- Architectural models which are specific to some application domain
- Use them whenever you can
- “Generic models” are abstractions from a number of real systems, abstractions that encapsulate the principal characteristics of these systems.

Generic compiler architecture



Generic language processing system



Summary

- The software architecture is a description of the overall structure of the software system.
- It is an important component of system design.
- Major activities include
  - Designing the system structure
  - Designing the control model
  - Performing further modular decomposition.
- Use domain-specific architectures when you can.