The major forms of architectural style

1 Call and return

Characterised by:

- Order of computation (sequencing of control)
- Only a single thread of control (no concurrency)
- Structures organised around computational tasks

Type of reasoning:

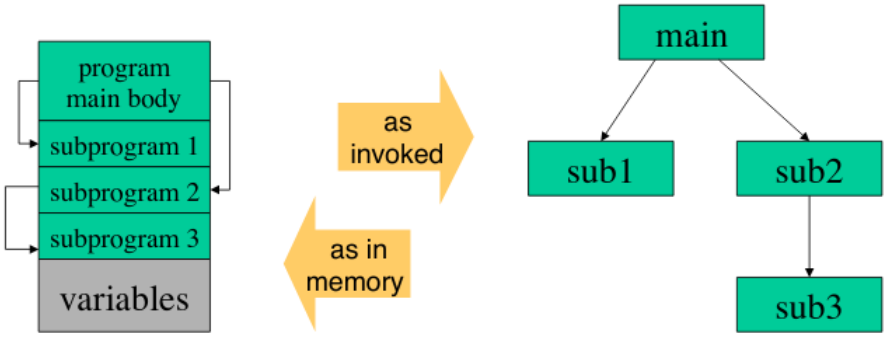
- Hierarchical, organised around control being passed from ”higher” to ”lower” items
- Task performed by a set of sub-tasks
- Explicit control linkages between the subtasks and the algorithm of the program to determine calling order

1.1 Example 1

Main program/sub-programs:

- Components are subprograms
- Connectors are the invocation links (including parameters)

This is the form that is embodied in most non-OO imperative programming languages

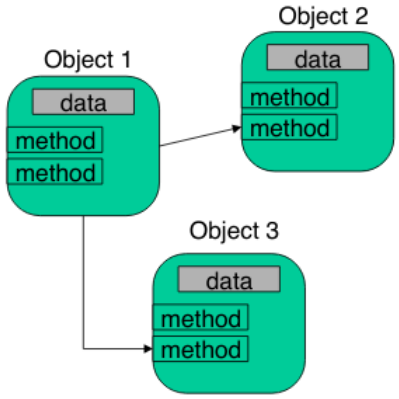


1.2 Example 2

Classical objects - here the methods are part of the objects:

- Components are both the objects and also their methods/data
- Connectors are the method calls and parameters

Supported by object-based and object-oriented programming languages. May use run-time bindings



2 Interacting Processes

Characterised by:

- Communication patterns among independent, usually concurrent, processes
- Independent elements can be objects, lightweight processes, services etc
- Context is more complex as emphasis is upon ”interaction”

Type of reasoning

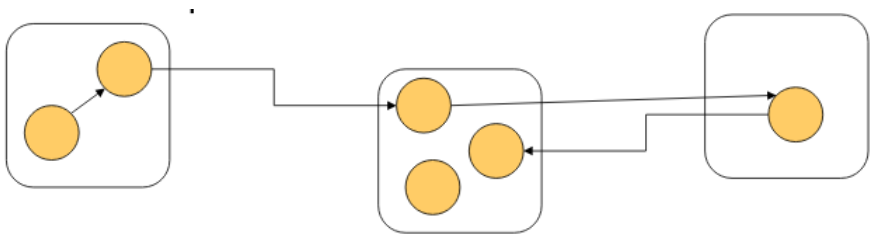
- Non-deterministic - scheduling of system elements is performed by separate, independent computers
- Design based around these independent actions and needs to specifically address any requirements that impose the need for particular sequences or interactions

2.1 Communicating processes

Based around a network of processes running on different machines and linked via Remote Procedure Calls (RPC)

- Components are the processes
- Connectors are the message protocols via RPCs

Binding time can be at each construction or when the system is started. Each system runs in its own address space - no direct access to shared data

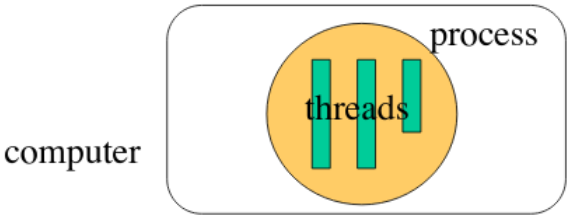


2.2 Lightweight processes (threads)

A thread is a block of code that can be executed independently of other threads within the program

- Components are the threads
- Connectors are provided largely by using shared data

Because threads are part of a single process, they have a shared data space, needing care with synchronisation of access



3 Data-Centred Repository

Characterised by:

- A dominant central data store that is manipulated by independent communications
- Centred around the issues relating to data access
- Data is at the focus

Type of reasoning:

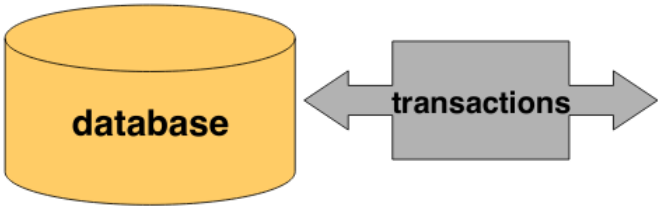
- Can depend on the form of the system
- For databases it is the ACID properties
- Includes modern compilers

3.1 Transactional Database

The classical idea of the data-centred repository

- Components - Memory and computations within the database
- Connectors - Transaction streams (queries)

Style is not concerned with the internal form of the database, only with its role



3.2 Client-Server

A ”distributed” form using a ”thin” client process to interact with the end-users. While the server often incorporates a database, this is not a necessity

- Components - Data managers and computations
- Connectors - Transaction operations with history