Module 03 : The ingredients of computation; Functional decomposition

Notes

The Ingredients of Computation

Three forces are at play when we use software to perform some computations:

- **Processors :** The processors are the computation devices, physical or virtual, that execute instructions.
    - A processor can be an actual processing unit (the CPU of a computer)
    - A process on a conventional operating system
    - A “thread” if the OS is multi-threaded.

- **Actions :** The actions are the operations making up the computation.

  They are
    - Machine language operations at the hardware level
    - Programming language at the hardware software machine level
    - Steps of an algorithm at the software level.

- **Objects :** The objects are the data structures to which the actions apply.
    - Some objects, are internal data structures that exist only while the computation proceeds

- Others are external and may outlive individual computations

Functional Decomposition

Functional decomposition is a method of analysis that dissects a complex process in order to examine its individual elements. A function, in this context, is a task in a larger process whereby decomposition breaks down that process into smaller, easier to comprehend units. It is the process of taking a complex process and decomposing into its smaller, simpler parts based upon the functionality.

It is used to describe a set of steps in which they break down the overall function of a device, system, or process into its smaller parts that describes the problem and or solutions in increasing detail

Top Down Approach

A top-down approach (also known as stepwise design and stepwise refinement and in some cases used as a synonym of decomposition) is essentially the breaking down of a system to gain insight into its compositional sub-systems in a reverse engineering fashion.

- The top-down approach builds a system by stepwise refinement, starting with a definition of its abstract function.
- We start the process by expressing a topmost statement of the function and continue with a sequence of refinement steps.
- Each step must decrease the level of abstraction of the elements obtained.
- It decomposes every operation into a combination of one or more simpler operations, until everything is at a level of abstraction low enough to allow direct implementation.

Problems with the Top Down Approach

- It is has severe scalability issues when it comes to large software.
- There's a certain trade-off over short-term convenience for long term inflexibility.
- Unduly privileging some functions over the others.
- Diversion of attention to minor issues at the expense of more fundamental properties.
- Sacrificing reusability aspect.