Structured programming design concepts

Programming Fundamentals (Monash University)

Top-down design

A top-down approach (also known as stepwise design) is essentially the breaking down of a system to gain insight into the sub-systems that make it up. In a top-down approach an overview of the system is formulated, specifying but not detailing any first-level subsystems.

Advantages

- Your organization realizes a focused use of resources from the individual managed application.
- The first implementation becomes a showcase for the identity management solution.
- When the phases are completed for the managed application, you have implemented a deeper, more mature implementation of the identity management solution.
- Operation and maintenance resources are not initially impacted as severely as with the bottom-up approach.

Disadvantages

- The solution provides limited coverage in the first phases.
- A minimal percentage of user accounts are managed in the first phases.
- You might have to develop custom adapters at an early stage.
- The support and overall business will not realize the benefit of the solution as rapidly.
- The implementation cost is likely to be higher.

Bottom up design

Design methodology in which you complete the lowest-level portion of your design first. Only after the low-level building blocks are complete do you finish higher level hierarchical blocks in your design. This methodology is used with schematic capture programs.

Advantages

- User and business awareness of the product. Benefits are realized in the early phases.
- You can replace many manual processes with early automation.
- You can implement password management for a large number of users.
- You do not have to develop custom adapters in the early phases.
- Your organization broadens identity management skills and understanding during the first phase.
- Tivoli Identity Manager is introduced to your business with less intrusion to your operations.

Disadvantages

- The organizational structure you establish might have to be changed in a later roll-out phase.
- Because of the immediate changes to repository owners and the user population, the roll-out will have a higher impact earlier and require greater cooperation.
- This strategy is driven by the existing infrastructure instead of the business processes.

Modular design

Modular design or modularity in design is a design approach that subdivides a system into smaller parts called modules or skids, which can be independently created and then used in different systems.

Advantages

- Fewer bug because each set of programming commands is shorter
- Algorithm is more easily understood
- Many programmers can be employed, one on each of the modules
- Programmers can use their expertise on particular techniques
- Testing can be more thorough on each of the modules
- Allows library programs to be inserted
- Saves time and means the finished program can be completed more quickly

Disadvantages

- Can lead to problems with variable names
- Means documentation of modules must be thorough
- Can lead to problems when modules are linked because link must thoroughly tested

Monolithic design

A software system is called monolithic if it has a monolithic architecture, in which functionally distinguishable aspects (for example data input and output, data processing, error handling, and the user interface) are not architecturally separate components but are all interwoven.

Monolithic programming is the act of creating an application or software that is comprised of a single tier. Everything within the program is combined into one single state and operates independently of any other neighbour programs.

Mainframe computers used a monolithic architecture with considerable success. Monolithic architectures implemented on DOS and earlier Windows based PCs often worked poorly with multiple users. This performance degradation is mainly due to poor mechanisms for record locking and file handling across local area networks.

Advantages

Disadvantages

- Problematic due to the fact that the all the functions of the program are compiled into one single tier. This means the size of that single program would be considerably larger than the modules created through modular programming, which makes the monolithic program less stable.
- Problematic due to the fact that if a bug or virus is installed into a monolithic program then it cannot be isolated in the same way as if the same error was present in a modular program. This is because a programmer could isolate the infected module from the rest of the system with a modular program, whereas with a monolithic program this is not possible.
- Finding faults and bugs is not easy

Control flow structure

In computer science, control flow (or alternatively, flow of control) is the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated. The emphasis on explicit control flow distinguishes an imperative programming language from a declarative programming language. Within an imperative programming language, a control flow statement is a statement whose execution results in a choice being made as to which of two or more paths should be followed. For non-strict functional languages, functions and language constructs exist to achieve the same result, but they are not necessarily called control flow statements.