Program Design

Part of design phase of SDLC
Determine what code modules are required
Determine how code modules will work together to form the program Develop specific instructions for development team on how code modules should be written
Techniques in program design Structure chart
Program specifications
Designing Programs Planning before coding generally improves the development process Never begin writing code without having a complete understanding of what code must do Mitigates risk of:
Inefficient programs Incompatible programs
Programs missing key functionality--- Program design goal Program design that is modular and flexible
Top-down modular approach to design Determine major program components first,then detailed sub-components supporting them Eases understanding and communication of design Process of program decomposition
Designing Programs Modular design advantages Code that is easier to understand
Code that is reusable Code that has less redundancy Code that is easier to maintain Program modules should be decomposed until they perform a single function---Program design document Prepared at completion of program design Components Program structure chart Program specificationsStructure Chart Hierarchical, high-level depiction of all modular components of an program Emphasizes structure and reusability Modules should perform a single function Details sequence, selection, and iteration of modules--- Structure Chart Syntax Modules Control module Higher-level module that contains logic for calling and controlling sub-component modules Subordinate module Low-level module contains logic for performing

Program Design (cont)

specific function when called by the control module Library module Generic module used by multiple control module. Loops Iteration – execution of subordinate modules repeated Conditional line Selection – subordinate execution based on condition.Couples
Shows information passed between modules
Data couples Passing data
Control couples Passing parameters, flags, system messages---Building a structure chart --Types of processes Afferent Provide inputs into the system Central Perform critical functions in the system operation Efferent Produce outputs of the system---Module structures 1.Transaction structure Control module calls subordinate modules based upon a condition or selection that handle specific transactions Correspond to higher levels of DFD and structure chart 2.Transform Structure Control module calls several subordinate modules in sequence to create a specific output Forms a process to transform inputs into an output Correspond to lower levels of DFD and structure char--Program design guidelines 1.High Cohesion-- Modules should perform a single function efficiently Modules easy to understand and develop 2.Loose coupling. Modules are independent from one another Data passed between modules is limited Modules easy to modify without impacting others 3.High fan-in
Multiple control modules should call a single subordinate module Sign of proper module reuse 4.Low fan-out Single control module has few subordinate modules Maximum of seven subordinate modules per control Control modules are not overly complex. Program specification -- Explicit instructions for development team on how code modules should be written Develop program specification for each module in structure chart. Program information Events that trigger functionality in the program Inputs and outputs
Structured English and pseudocode.

System Acquisitions and Arch

System acquisition strategies Approaches to system acquisition Influences on acquisition strategy Selecting an acquisition strategy---Architecture design
Elements of an architecture design Creating an architecture design--Approaches to System Acquisition Three ways to approach creation of a system Custom development Packaged software Outsourcing--Systems Acquisition and Architecture System acquisition strategies Approaches to system acquisition
Influences on acquisition strategy Selecting an acquisition strategy Architecture design Elements of an architecture design
Creating an architecture design---Approaches to System Acquisition Three ways to approach creation of a system1.Custom development 2.Packaged software: 3.Outsourcing---Influences on Acquisition Strategy 1.Business need2.Time frame3.Inhouse experience---Selecting an Acquisition Strategy. Alternative matrix Evaluate pros and cons of design alternatives Weight criteria according to importance Avoid subjective bias toward preferences
Independent ratings by each analyst--Architecture Design Information systems are distributed Architecture design determines how system will be distributed across computers Non-functional requirements play key role---Virtualization Server virtualization Partitioning physical server into several independent virtual servers Reduced hardware requirements Storage virtualization Combining multiple storage devices into single virtual storage space Improved storage and retrieval speed--Creating an Architecture Design Non-functional requirements used to guide architecture design decisions Operational requirements Performance requirements
Security requirements Cultural and political requirements

Implementation

Implementation Construction of the new system Thorough job in analysis and design phases lead to a successful implementation of the system

Developing the system's software Largest component of SDLC in time and cost Generally presents the fewest problems Design and execution of system testing Develop system documentation---Managing the Programming Process

Coordinating programming activities Regular meetings Programming standards and guidelines Software development areas Development, testing, and production Code management systems Version control Program log Program check-in / check-out---Testing Writing programs is fun Testing and documentation are not Professional organizations spend more time and money in testing than writing programs Risk associated with system failure is severe Testing is insurance; expenditure justified Testing often performed by systems analyst Four types of tests Unit, integration, system, and acceptance tests Most errors found in integration and system testing---Developing Documentation System documentation Created in analysis and design phases

Enable maintenance of system after installation User documentation

User manuals, training manuals, help systems

Designed to assist users of the system Good documentation takes time Allocate resources to documentation in project plan---Organizational Transition People are generally resistant to change Business processes and computer systems become habitual, people become comfortable Implementation of a new system challenging Managing organizational change Unfreeze, transition, refreeze Prior SDLC phases help users prepare for change

Migration plan guides the transition

Post-implementation establishes new system---Migration plan.

Implementation (cont)

Decisions, plans, procedures guiding transition

Conversion strategy Business contingency plan-- Conversion strategy Conversion style Defines how abruptly the new system is introduced Conversion locations Defines what portion of the organization transitions Conversion style Direct conversion Instant replacement of the old system with the new High risk Any problems have dramatic impact on organization---Business contingency planning

Withstand impact of problems with new system Proper project management and migration planning helps to ensure smooth transition Parallel conversion provides a fallback Plan for worst-case scenario – no system--Post-Implementation Activities Institutionalize use of new system Establish as normal way of doing business Refreeze organization after successful transition Three key post-implementation activities System support Maintenance Project assessment System support Providing assistance to users after. implementation System transferred to operations group Provide online support, guides, FAQs Help desk Level 1 and 2 support staff---System maintenance Refining system after implementation to ensure it continues to meet business needs Substantially more resources devoted to system maintenance than initial development Change request Changes performed through smaller SDLC cycle Bug fixes – highest priority Enhancements – second priority Project assessment Organizational learning Understand successes and failures in system development activities Project team review Focuses on way project team performed activities System review Focuses on extent system provided benefits promised.