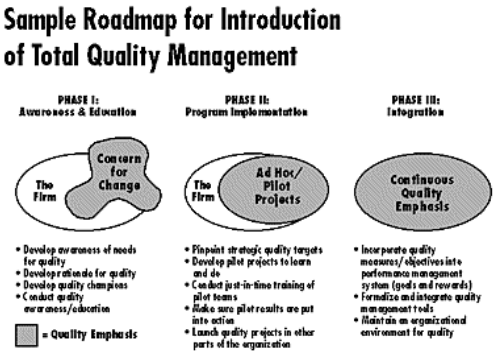
Quality, Success and Failure

Total Quality Management (TQM)

- Total Quality Management (TQM) is a concept that makes quality control a responsibility to be shared by all people in an organization.



Total Quality Management (TQM)

- Everyone is expected to contribute to the overall improvement of quality
  - The **Engineer** who *avoids design errors*
  - The **Production Worker** who *spots defects*
  - The **Sales Representative** who *presents the product properly to potential customers*
  - The **Secretary** who *avoid typing mistakes*
- Quality improvements not only *raise the level of product and service quality*, but they also *lower costs*.

Total Quality Management (TQM) vs. Process Reengineering (BPR)

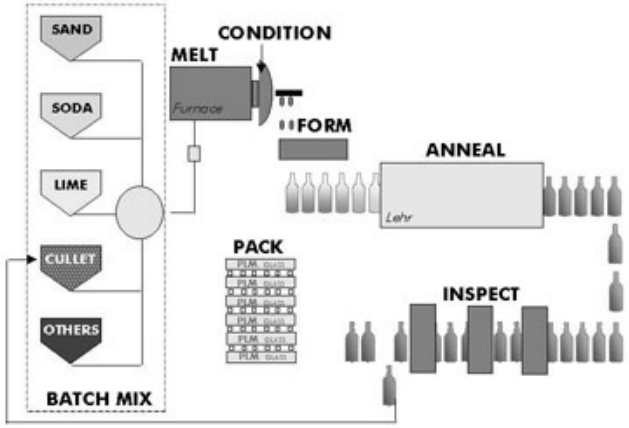
- Total Quality Management (TQM) is more incremental than Business Process Reengineering (BPR) because its efforts focus making a series of continuous improvements rather than dramatic bursts of change.
- Sometimes, however, processes may have to be fully reengineered to achieve a specified level of quality.

How Information System Contribute to Total Quality Management ?

- Information System can help firms achieve their quality goals by helping them
  - Simplify products or processes
  - Meet benchmarking standards
  - Make improvements based on customer demands
  - Reduce cycle time
  - Increase the quality and precision of design and production

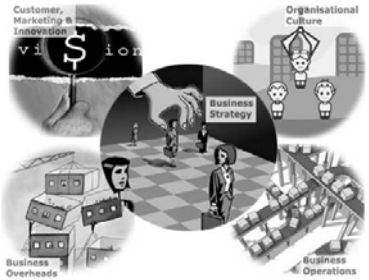
Simplifying the Product or the Production Process

- The fewer steps in a process, the less time and opportunity for an error to occur.



Benchmarking

- Many companies have been effective in achieving quality by setting strict standards for products, services and other activities and then measuring performance against those standards.
- This procedure is called **Benchmarking**.



Use Customer Demands as a Guide to Improve Products and Services

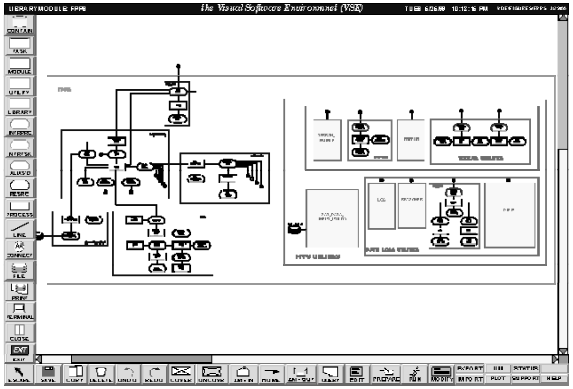
- Improving customer service, making customer service the number one priority, will improve the quality of the product itself.

Reduce Cycle Time

- Reducing the amount of time from the beginning of a process to its end (cycle time) usually results in fewer steps.
- Shorter cycles mean that errors are often caught earlier in production (or logistics or design or whatever), often before the process is complete, eliminating many hidden costs.

Improve the Quality and Precision of the Design

- Computer-Aided Design (CAD) software has dramatic quality improvements possible in a wide range of business.

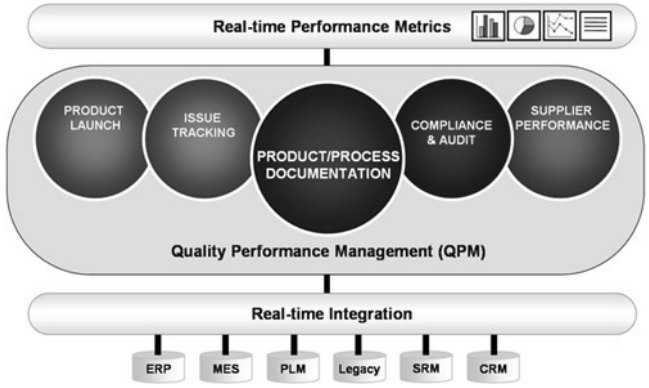


Improve the Precision of Production

- For many products, one key way to achieve quality is to tighten production tolerances.
- CAD software often includes a facility to translate design specifications into specifications both for production tooling and for the production process itself.

Ensuring System Quality

- Organization can improve system quality by using software quality assurance techniques and by improving the quality of their data.



Software Quality Assurance

- Solutions to software quality problems include
  - Using an appropriate systems development methodology
  - Proper resource allocation during system development
  - The use of metrics attention to testing
  - The use of quality tools

Development Methodologies

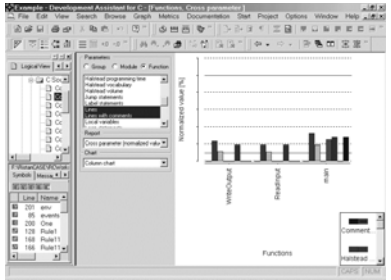
- The primary function of a development methodology is to provide discipline to the entire development process.
- A good development methodology establishes organization-wide standards for requirements gathering, design, programming and testing.
- To produce quality software, organizations must select appropriate methodology and then enforce its use.

Resource Allocation during Systems Development

- Resource Allocation determines the way the costs, time and personnel are assigned to different phases of the project.
- Current literature suggests:
  - 25% resources in specification & analysis
  - 50% on design & programming
  - 25% on installation & post-implementation

Software Metrics

- Software Metrics are objective assessments of the system in the form of quantified measurements.
- Ongoing use of metrics allows the Information System department and the user to measure the performance of the system and identify problems as they occur.



Testing

- Testing begins at the Design Phase.
- Because no coding yet exists, the test normally used a walkthrough – is a review of a specification of design document by a small group of people carefully selected based on the skills needed for the particular objectives being tested.
- Debugging is the process of discovering and eliminating the errors and defects (bugs) in the program code.

Quality Tools

- Many tools have been developed to address every aspect of the systems development process.
- Information System professionals are using project management software to manage their projects.
- Examples:
  - Programming tools
  - Debugging tools
  - Testing tools

Data Quality Audits

- A survey of files and samples of files for accuracy and completeness of data in an Information System.
- They are accomplished by the following methods:
  - Surveying end users for their perceptions of data quality
  - Surveying entire data files
  - Surveying samples from data files
- Unless regular data quality audits are taken, organizations have no way of knowing to what extent their Information System contain inaccurate, incomplete or ambiguous information.

Development Methodology

- A collection of methods, one or more for every activity within every phase of a development project.
  - Structured Methodologies
  - Object-Oriented Software Development
  - Computer-Aided Software Engineering (CASE)
  - Software Reengineering

Structured Methodologies

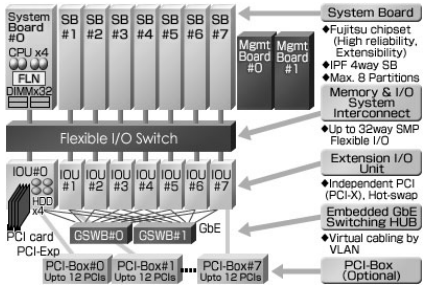
- Used since 1970s.
- Structured means the techniques are step-by-step, with each step building on the previous one.
- Top-down approach.
- Process-Oriented
- Linear, each phase must be completed before the next one start.
- It includes
  - Structured Analysis
  - Structured Design
  - Use of Flowcharts

Structured Analysis

- It is widely used to define system input, processes, and outputs.
- It offers a logical graphic model of information flow, partitioning a system into modules that show manageable levels of details.
- It specifics the processes or transformations that occur within each module and the interfaces between them.
- Primary tools are Data Flow Diagram (DFD), Data Dictionary and Process Specifications.

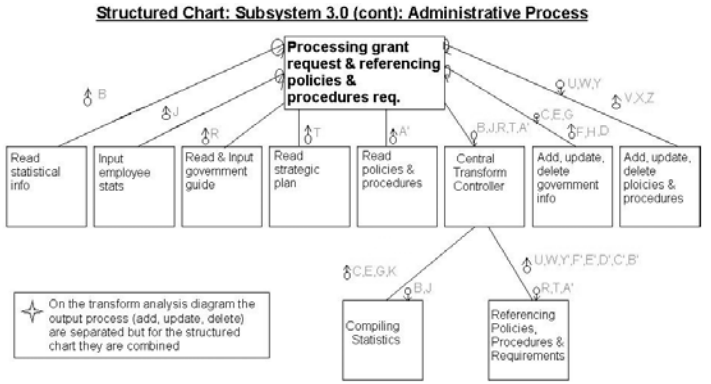
Structured Design

- It encompasses a set of design rules and techniques that promotes program clarity and simplicity.
- Design is in top-down approach.
- Primary tools is Structured Chart.



Structured Chart

- System documentation showing each level of design, the relationship among the levels, and the overall place in the design structure.

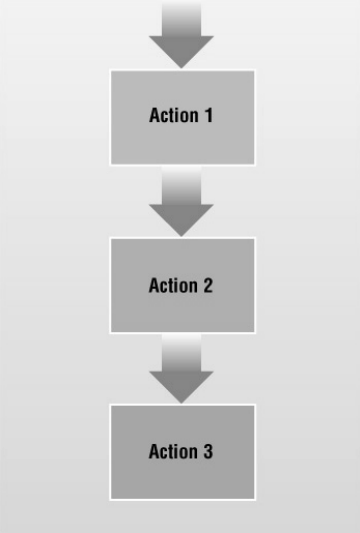


Structured Programming

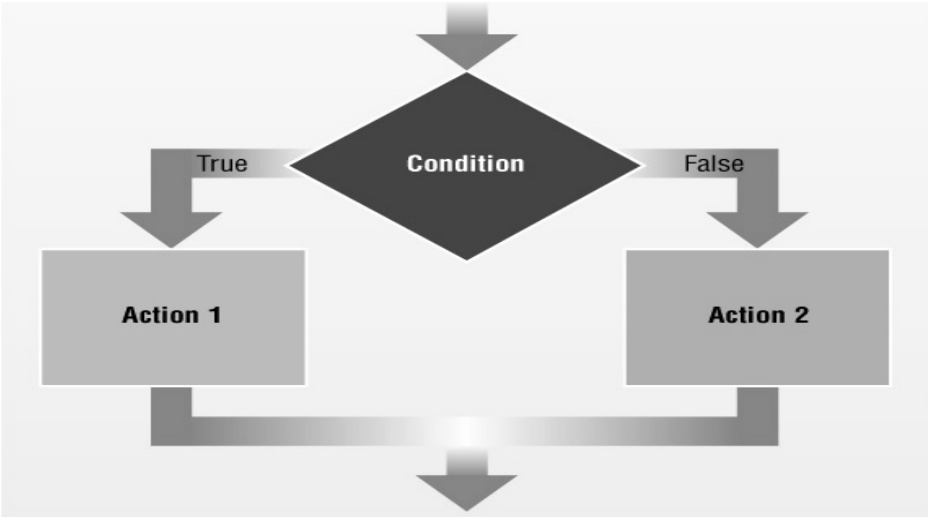
- A discipline for organizing and coding programs that simplifies the control paths so the programs can be easily understood and modified.
- Uses the basic control structures and modules that have only entry point and one exit point.
- Any program can be written using
  - Sequence
  - Selection
  - Iteration

Sequence Control Structure

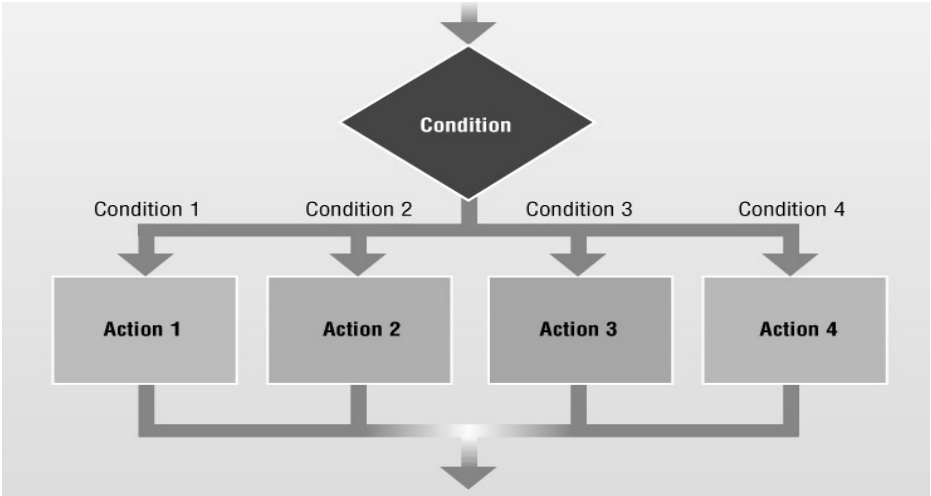
- Shows one or more actions following each other in order
- Actions could be
  - Inputs
  - Processes
  - Outputs



Selection Control Structure: If-then-else

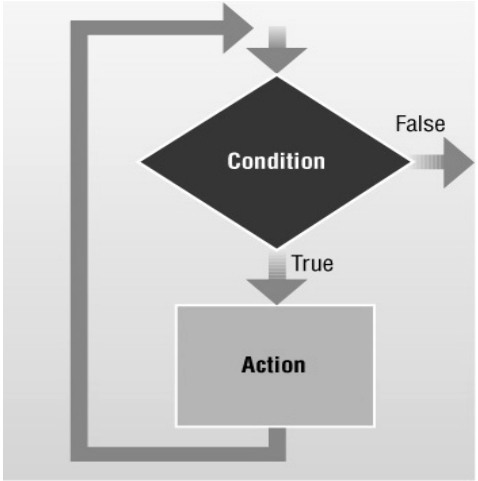


Selection Control Structure: Case



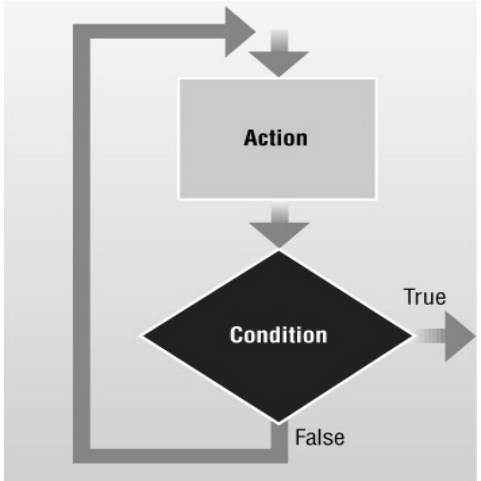
Iteration Control Structure: While Loop

- Repeats one or more times as long as condition is true



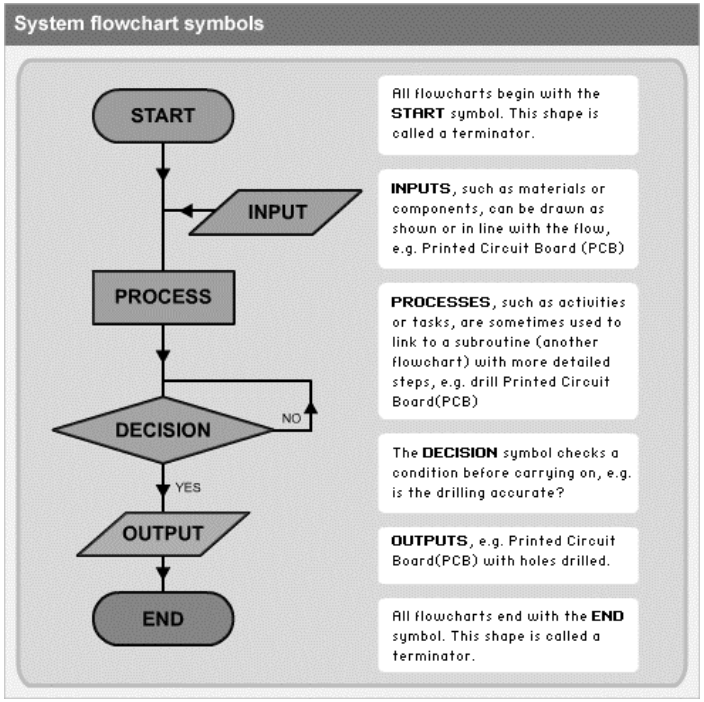
Iteration Control Structure: Do-While Loop

- Tests condition at end of loop



System Flowcharts

- A graphic design tool that depicts the physical media and sequence of processing steps used in an entire Information System.
- They can show all inputs, major files, processing, and outputs for a system, as well as manual procedures.



Program Flowcharts

- It describes the processes taking place within an individual program in the system and the sequence in which they must be executed.

Limitations of Structured Methodologies

- Inflexible
- Time-consuming
- Change in specifications requires analysis & design documents be modified before programs can be changed.
- Function-oriented rather than Data-oriented.

Object-Oriented Software Development

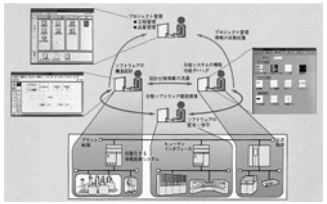
- An approach to software development that deemphasizes procedures and shifts the focus from modeling business processes and data to combining data and procedures to create objects.
- System is viewed as collection of classes, objects and relationships among them.
- Objects are easily reusable and expected to reduce development time & cost.

Computer-Aided Software Engineering (CASE)

- The automation of step-by-step methodologies for software and systems development to reduce the amount of repetitive work the developer needs to do.
- It facilitates the creation of documentation and the coordination of team development efforts.

Functionality for Computer-Aided Software Engineering (CASE)

- CASE tools provide automated graphics facilities for
  - Charts & diagrams
  - Screen & report generators
  - Data dictionaries
  - Analysis & checking tools
  - Code generators
  - Documentation generators



Benefit for Computer-Aided Software Engineering (CASE)

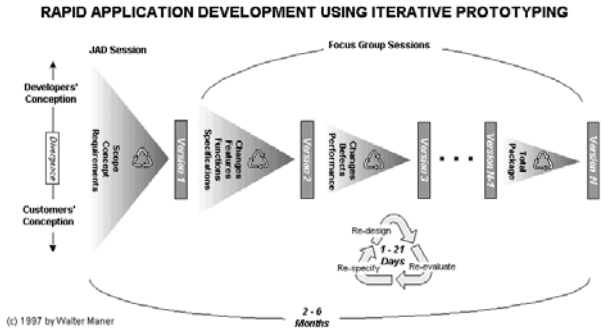
- CASE tools try to increase productivity by
  - Enforce a standard development methodology and design discipline.
  - Improve communication between users and technical specialists.
  - Organize & correlate design components and provide rapid access to them.
  - Automate analysis & design.
  - Automate code generation, testing and control rollout.

CASE Tools Classification

- Front end
  - Focus on capturing analysis and design information in the early stages of system development.
- Back end
  - Address coding, testing and maintenance activities.
  - Also help to convert specifications automatically into program code.

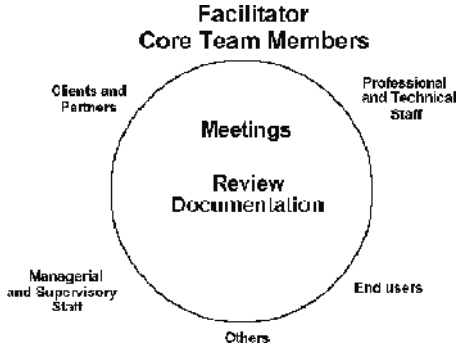
Rapid Application Development (RAD)

- Process for developing systems in very short time period by using prototyping, fourth-generation tools, and close teamwork among users and systems specialists.



Joint Application Design (JAD)

- Process to accelerate the generation of information requirements by having end users and Information System specialists work together in intensive interactive design sessions.



Software Reengineering

- A methodology that addresses the problem of aging software by upgrading it so that the users can avoid a long and expensive replacement project. It involves:
  - Reverse Engineering
    - Converting existing programs, files, and database description into corresponding design-level components that can be used to create new applications.
  - Revision of Design & Program Specifications
  - Forward Engineering
    - Generate new structured program code for a structured and maintainable system.