Art of creating software

Content

TODO: FOR every topic do google resarch.

From: https://www.tutorialspoint.com/software_engineering/index.htm

Characteristics of good software

- Operational
- Transitional
- Maintenance

Operational

This tells us how well software works in operations. It can be measured on:

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

Transitional

This aspect is important when the software is moved from one platform to another:

- Portability
- Interoperability
- Reusability
- Adaptability

Maintenance

This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:

- Modularity
- Maintainability
- Flexibility
- Scalability

Software Requirements

- Functionalit
- Non- functional

Functional Requirements

Examples

- Search option given to user to search from various invoices.
- User should be able to mail any report to management.
- Users can be divided into groups and groups can be given separate rights.
- Should comply business rules and administrative functions.
- Software is developed keeping downward compatibility intact.

Non-functional requirements

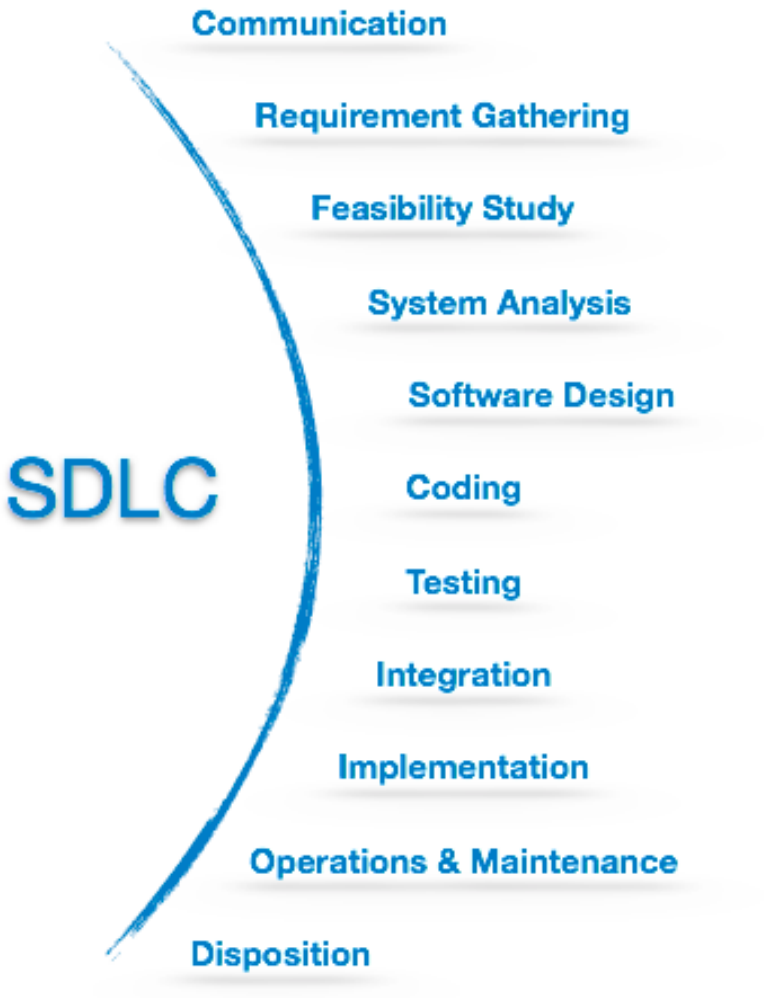
- Security
- Logging
- Storage
- Configuration
- Performance
- Cost
- Interoperability
- Flexibility
- Disaster recovery
- Accessibility

Requirements are categorized logically as

- **Must Have** : Software cannot be said operational without them.
- **Should have** : Enhancing the functionality of software.
- **Could have** : Software can still properly function with these requirements.
- **Wish list** : These requirements do not map to any objectives of software.

Project Management

- Gantt Chart
- PERT Chart
- Resource Histogram
- Critical Path Analysis



PM Paradigms

- Waterfall Model
- Iterative Model
- Spiral Model
- V – model
- Big Bang Model

Software Design Levels

Software design yields three levels of results:

- **Architectural Design -** The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of proposed solution domain.
- **High-level Design-** The high-level design breaks the ‘single entity-multiple component’ concept of architectural design into less-abstracted view of sub-systems and modules and depicts their interaction with each other. High-level design focuses on how the system along with all of its components can be implemented in forms of modules. It recognizes modular structure of each sub-system and their relation and interaction among each other.
- **Detailed Design-** Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations. It defines logical structure of each module and their interfaces to communicate with other modules.

Software Design Approaches

- Top Down Design
- Bottom-up Design
- TODO find more

Software design process can be perceived as series of well-defined steps. Though it varies according to design approach (function oriented or object oriented, yet It may have the following steps involved:

- A solution design is created from requirement or previous used system and/or system sequence diagram.
- Objects are identified and grouped into classes on behalf of similarity in attribute characteristics.
- Class hierarchy and relation among them is defined.
- Application framework is defined.

Structured design is mostly based on ‘divide and conquer’ strategy where a problem is broken into several small problems and each small problem is individually solved until the whole problem is solved.

In function-oriented design, the system is comprised of many smaller sub-systems known as functions. These functions are capable of performing significant task in the system

Object oriented design works around the entities and their characteristics

Metrics

→ **Complexity Metrics -** McCabe’s Cyclomatic complexity quantifies

→ **Quality Metrics -** Defects, their types and causes, consequence, int

Halstead's Complexity Measures

In 1977, Mr. Maurice Howard Halstead introduced metrics to measure software complexity. Halstead’s metrics depends upon the actual implementation of program and its measures, which are computed directly from the operators and operands from source code

Cyclomatic Complexity Measures

Every program encompasses statements to execute in order to perform some task and other decision-making statements that decide, what statements need to be executed

Function Point

It is widely used to measure the size of software. Function Point concentrates on functionality provided by the system. Features and functionality of the system are used to measure the software complexity.

Modularization is a technique to divide a software system into multiple discrete and independent modules-

Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.

There are seven types of cohesion, namely –

- **Co-incidental cohesion -** It is unplanned and random cohesion, which might be the result of breaking the program into smaller modules for the sake of modularization. Because it is unplanned, it may serve confusion to the programmers and is generally not-accepted.
- **Logical cohesion -** When logically categorized elements are put together into a module, it is called logical cohesion.
- **Temporal Cohesion -** When elements of module are organized such that they are processed at a similar point in time, it is called temporal cohesion.
- **Procedural cohesion -** When elements of module are grouped together, which are executed sequentially in order to perform a task, it is called procedural cohesion.
- **Communicational cohesion -** When elements of module are grouped together, which are executed sequentially and work on same data (information), it is called communicational cohesion.

- **Sequential cohesion -** When elements of module are grouped because the output of one element serves as input to another and so on, it is called sequential cohesion.
- **Functional cohesion -** It is considered to be the highest degree of cohesion, and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

Coupling is a measure that defines the level of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the program.

There are five levels of coupling, namely -

- **Content coupling -** When a module can directly access or modify or refer to the content of another module, it is called content level coupling.
- **Common coupling-** When multiple modules have read and write access to some global data, it is called common or global coupling.
- **Control coupling-** Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution.
- **Stamp coupling-** When multiple modules share common data structure and work on different part of it, it is called stamp coupling.
- **Data coupling-** Data coupling is when two modules interact with each other by means of passing data (as parameter). If a module passes data structure as parameter, then the receiving module should use all its components.

Diagrams

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

Structure chart is a chart derived from Data Flow Diagram. It represents the system in more detail than DFD. It breaks down the entire system into lowest functional modules, describes functions and sub-functions of each module of the system to a greater detail than DFD.

HIPO diagram represents the hierarchy of modules in the software system

IPO (Input Process Output) diagram which depicts the flow of control and data in a module

A Decision table represents conditions and the respective actions to be taken to address them, in a structured tabular format.

Entity-Relationship model is a type of database model based on the notion of real world entities and relationship among them.

CASE Tools

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

Case Tools Types

- Diagram tools (Flow Chart Maker)
- Process Modeling Tools (EPF Composer)
- Project Management Tools (For example, Creative Pro Office, Trac Project, Basecamp.)
- Documentation Tools (For example, Doxygen, DrExplain, Adobe RoboHelp)
- Analysis Tools (Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.)
- Design Tools (For example, Animated Software Design)
- Configuration Management tools (Configuration Management tools)
- PROTOTYPING tools Serena prototype composer, Mockup Builder.
- Quality Assurance Tools (For example, SoapTest, AppsWatch, Jmeter. For example, Bugzilla for defect tracking, HP Quality Center)

Programming

Read:
https://www.tutorialspoint.com/software_engineering/software_implementation.htm

Testing

Read:
https://www.tutorialspoint.com/software_engineering/software_testing_overview.htm