The Tao of Software Design

Thus said the Master Programmer:

**“In Matters of Software and All Things,**

**Look Before You Leap, Grasshopper.”**

****The Software Development Life Cycle

A scienceand an aRt

Involves a basic set of overlapping and interrelated phases

“A Goal Without a Plan is Just a Wish”

The **Software Development Life Cycle** (SDLC) describes a classic process, or methodology, for developing a software application in a way that maximizes success, i.e. a product that is cost-efficient, effective, and of high quality.

There are specific phases of the SDLC that are followed in a particular order, with each phase providing critical information that is used in the phase that follows.

The process is *iterative,* meaning that, at any point in the process, one can return to a previous phase to correct mistakes or make modifications (the Agile methodology).

1. **Define the "Problem"**

In this phase, the basic objectives of the application are determined. What is the application supposed to accomplish, in the *simplest* and most *general* sense?

If we do not understand the problem correctly, then no matter how accurate our solution is, it is guaranteed to be a failure.

1. **Establish the Requirements**

In this phase, all possible requirements of the application are gathered by the systems analyst. There are extensive interviews with all major stakeholders (anyone affected by the application), including managers and users. If the new application is to replace an existing one, the existing applicationis studied and analyzed to understand how it works, its strengths and weaknesses. This gives the analyst a much better understanding of what the new system is supposed to accomplish.

* Interview **users** and *everyone* affected by the application.
* Define the Inputs, Outputs, and theProcessing (how to get from the inputs to the outputs).

**The need for Output drives everything**

* Develop a detailed documentof the precise requirements – the *Requirement Specification* document.
* Present to the **Users**, and all stakeholders,and modify accordingly.
* Perform a complete task analysis, i.e. define all tasks necessary to complete the project, and which tasks are dependent on the completion of other tasks.
* Plan and schedule the project.

1. **Work Out the Design**

* Defines how the project will *accomplish* the requirements.
* Provides a reality check between the features of the application and the required completion date.
* Focus on the needs identified in the Requirements phase to develop the **high-level**design of the application.
* Work with potential **users** of the system to define how they use the existing system (if one exists) and what they would like to see in the new system.
* Create an **outline** of what needs to be done. Draw sketches and list the major design points.
* Experiment with a variety of possible solutions. Make rough sketches. Play around. There is no single, “right” answer.
* Review your requirements from the previous phase. Does the design meet all the requirements?

**Now take things to a deeper level:**

* Develop the design at a **lower level**, i.e. more detail.
* Develop a detailed formal design document that others can review – the *Design Specification*.
* Construct a model – a **prototype** – a minimally functioning version of the application that provides the end user with a visual idea of what each screen will look like.
* Present it to the **Users** and all stakeholders, and modify accordingly.

1. **Construct the Program**

Now it's time to start coding.

* Use the Design document as a blueprint to build the application.
* Test the application frequently as a part of the construction process.
* Write a very *small* amount of code, then test it, debug it, and repeat over and over.Sing the Programmer’s Chant: “Code, Test, Debug. Code, Test, Debug,etc…"
* Present to the **Users** and all stakeholders, and modify accordingly.
* Integrate the various code modules together. (Keep singing!)
* Again, present to all involved and modify accordingly.

1. **Deploy & Maintain the Application**

* Set it up on the users' system and insure that it works.
* Make bug fixes and any necessary changes.
* Present to all stakeholders and modify accordingly.
* All revisions follow the same Software Development Life Cycle.

**All Phases: Document Everything**

NOTES:

* The Software Development Life Cycle is also known as the Software Development Process.

Program Design

## Why do programs need to be designed?

* + Most programs are very complex entities. Imagine building a skyscraper or a bridge without a design.
  + Designing programs saves substantial amounts of time in coding, debugging, and maintenance.
  + Design is the nourishment that code feeds upon. A neglected design diet will lead to oversized, bloated, gassy code.Provide your code with a healthy, wellthought-out design, and your code will be lean and mean.

Relative Cost to Fix an Error5

|  |  |
| --- | --- |
| Phase Found | Cost Ratio |
| Requirements | 1 |
| Design | 3 – 6 |
| Coding | 10 |
| Development Testing | 15 – 40 |
| Acceptance Testing | 30 – 70 |
| Operation | 40 – 1000 |

Source: Software Engineering Economics – Barry Boehm

Structured Design /

Structured Programming

## *Top-Down Design/Programming*6

* General solution is outlined first – the skeleton (like the frame of a house).
* Solution is developed gradually from the very general to increasingly lower levels of detail (in the same way that a house is constructed) – Flesh it out.
* Similar to painting (sketching the outline first, then filling in the details), sculpting, making a pizza.
* You DON'T complete one part or section, and then start on another.

## *Object-Oriented Design/Programming*

* **Create components of code that are:**
  + independent of the application and
  + reusable in other,related applications

Think of them as"plug-and-play" modules of code.

* **Analogous to the various components in an entertainment system:**
  + TV, DVD player, amplifier, speakers, digital video recorder,universal remote control, etc.
  + None of these components need to be from the same manufacturer, e.g. Panasonic TV, Samsung DVD player, Sony amplifier, etc. And yet they all work perfectly together, because everyone agrees upon a common interface, a standard way of talking to one another.
  + This is exactly how object-oriented components are designed. They can each be developed by different programmers, but all are agreeing on a common interface.
* **Advantages:**
  + Makes it easierto understand andmodify the software
  + Saves time in creating related applications by re-using pre-written components

**Note:**

**These approaches are NOT mutually exclusive.**

**In other words, both are used together.**

**To put it simply:**

**Structured Programming = Organization**

Goal-centered Design

* Design what the user needs, which is not necessarily what they *think* they need.**8**
* Talk to the users to determine what their *ultimate* goal is. It's possible that you, as the analyst may have a more appropriate solution that better fits their needs.

Algorithms and Pseudocode

## Algorithm

Generally speaking, an algorithm is a"recipe" -- the steps involved in accomplishing a task or solving a problem.

In the world of computers, it's a fancy word for the *basic and general* steps of the solution*,* when writing a computer program.

**An algorithm:**

* must give the correct solution. *(Duh!)*
* must be clear, precise and unambiguous.

**Some everyday examples of algorithms:**

* Driving directions
* Food recipes

## Pseudocode

* Describes a program solution in plain, structured English.
* It is often used *before* actual programming because it focuses more on the *meaning* of the solution, without getting bogged down in the detailed syntax of the programming language.
* Eventually, the programmer will translate the pseudocode into actual code.
* There is no absolute standard, but there are common conventions.
* Example: driving directions

# *Advantages of Pseudocode*

* Reduces coding time
* Makes review easier
* Makes changes easier
* Minimizes commenting effort