SOFTWARE ENGINEERING NOTES

Software development lifecycle (SLCD)

What is it? – steps to create high quality software followed by a dev team (can consist of Client facing people, Code monkeys, Project management)

Two types

1. WATERFALL MODEL (structured approach)
2. AGILE APPROACH (more flexible)

WATERFALL (in order)

* Define requirements
  1. interviews w/ client
  2. requirements report
     1. Time?
     2. Cost?
     3. Technical requirements
        1. (operating systems) (win11/macOS)
        2. Functionality
* Design Software
  1. UI design (visual)
  2. Logic
     1. Algorithms
     2. Flowchart
* Programming/ Development
  1. GIT – version control
  2. Choosing most suitable language
  3. Team based
* Testing
  1. Detect Bugs
  2. Fix them
* Deployment
  1. Releasing
     1. How do you release it?
     2. Timeline
     3. Phased? (releasing in different phases like only certain countries or areas as once)
* Maintenance
  1. Compatibility (changing to different platforms)
     1. Changing Requirements
  2. New features – could just be to keep software relevant – new feature will probably lead to another process of building software like stated above)
  3. Bug fixes

Pros

* Well- defined project phases
* Clear documentation
* Easier to estimate timeline and costs
* Reduced scope creek (when the client keeps wanting more and more)

Cons

* Requires significant upfront planning
* Difficult to make changes once a phase is completed
* Higher risk due to limited flexibility

People who commonly use the waterfall method to create software include government, large organisations like banks because there are normally legalities that come along with code that doesn’t work. The software must be VERY strong and complete. No mistakes for this.

2nd approach - AGILE APPROACH

“Sprints” In agile, dev teams work in short weekly development sprints, at end of each sprint, they meet together, assess progress show client and GO AGAIN!

Daily check in

Prototypes (and a lot of them)

* Iterative versions

Specifications

* Targets

Requirements

Continuous testing

* Constant testing to fix

FULL STACK DEVELOPER

* Front end (UI)(UX) (interacted space)
* Backend
  1. Python – JS – PHP
     1. Allow an application to access data from Dynamic applications that react according to the user
* DATABASE – SQL (structured query language) database

WHEN USING CHATGPT TO ORDER NOTES

BE SPECIFIC

DATA TYPES

What are they? They are locations in memory that are able to store data

Variables can store a range of different data types

They can be called from memory by computer programs by referring to a variable name(s)

E.g. Print(name) -> “Andrew”

Integer – whole number

(age = 16)(num\_of\_students = 16)

Floating point – decimal numbers

(bank\_balance = -4000.12)

String – text

name = ‘jack’ or name = “jack”

username = thickdickdaddy27

cellphone\_number = ‘0412 312 312’

contents of a string are called a list or array. Each individual letter contains it’s representative number

1234

Jack (1=j, 2=a, 3=c, 4=k)

Boolean

Used a lot in IF statements

male = false

binary 0=false, 1=true

default data types are seen as ‘classes’ in python (str, int, float)

different data types have different methods that are attached to different data types

developers can create their own data types using classes allowing they own types to have their own way to be interacted with.

ARITHMATIC OPERATOR

+ addition

- subtraction

/ division

\* multiplication

\*\* exponentials

// floor division (takes off any decimal place after division)(not rounded)

% remainder division (remainder of 10/3 = 1)(10-9 = 1)

CONTROL STRUCTURE

* Sequences
* Conditionals
  + If else statements
  + Elif statements
* Iteration
  + Repetition (loop(s))
* Counted loop
  + Loop – for loop
  + [For i = 1 to 10] (pseudocode)

[Display i]

[next i]

(python

While x < 11 (pretest)

Print x

x=x+1

x=1

repeat

display x (post-test)(doesn’t exist in python)

x=x+1

until x==10

* + While
  + For
  + Repeat until

PSEUDOCODE

A simplified, informal way of describing an algorithm using plain language and basic programming concepts without adhering to any specific programming syntax.

FLOWCHARTS are visible representation of a path of logic and processes

A blue rectangle with black text

Description automatically generated

**STRUCTURE CHARTS**

**Draw.io**

**Graphical representation of subroutines/ functions in a program.**

**Programs are modular. (multiple functions)**

**Helps us split complex logic into individual functions (decomposition)**

**Structure charts are read from left to right, top to bottom**

**TOP DOWN DESIGN**

**Breaking down a large problem into smaller components. Creates a much more manageable way of seeing a project.**

**Good for teams, - allocating work**

**Reuse functions in other projects.**

**BOTTOM UP DESIGN**

**Literally just the opposite, where you start with the smaller functions and build up from there.**

* All programs have I/P/O – inputs, process, outputs
* Conditions in conditionals / selection
* Use LOGICAL OPERATORS
  + > greater than
  + < less than
  + >= greater than or equal to
  + <= less than or equal to
  + == equal to
  + <>, != not equal to

Investigate procedure, subroutines

Understand functions

parameters

difference between functions and

* Function RETURN values

Blocks of code that can be replicated

* Give a name

START addition (x,y)

Display x + y

End addition

Addition (10,7)

Addition (5,3) #saying attrition now means to do a certain sequence of code

Modular programming

* Main line
* Sub a
* Sub b
* Sub c
* End

DATA STRUCTURES

A data structure is a format for organizing, storing, and processing (search, sort, max, min), data in a computer system.

Array, arrays must store single data type only.

scores = [5,7,9,11]

(5,7,9,11)

Scores = array of int[] (pseudo)

Scores = [] (python)

Arrays have the same indexing like lists.

Scores = [5,3,12,30,9]

Max = 0

Item = 0

while item < LENGTH scores #run until there are no more scores left to sort

A screen shot of a computer program

AI-generated content may be incorrect. IF scores[item] > max THEN

Max = scroes[item]

ENDIF

Item += 1 #increment by 1

#will go through all numbers in an array and change the ‘max’ variable whenever the max is less than a ‘new’ number

Way to test this algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| Scores | Max | Item | Length |
| [5,7,9,31,11] |  |  |  |
|  | 0 |  |  |
|  |  | 0 |  |
|  |  |  | 5 |
|  | 5 |  |  |
|  |  | 1 |  |
|  | 7 |  |  |
|  |  | 2 |  |
|  | 9 |  |  |
|  |  | 3 |  |
|  | 31 |  |  |
|  |  | 4 |  |
|  | 31 |  |  |
|  |  | 5 |  |

**Second type of data structure 2 DIMENTIONAL ARRAY**

(cols) (similar to indexing)(cols 0,1,2 is 3,5,8)

|  |  |  |
| --- | --- | --- |
| 3 | 5 | 8 |
| 1 | 2 | 5 |
| 3 | 3 | 3 |

Nums =

(rows)

Referencing values/cells in a grid, the following notation used(pseudo)

Nums[0,2] >> 8

Nums[2,2] >> 3

Writing out an array looks like this

Cols) (0) (1) (2)

Num = [ [ 3, 5 , 8 ],[ 1, 2, 5 ],[ 3, 3, 3 ] ]

# to display an entire row

For rows = 1 to 3

For col = 1 to 3

Display nums [ row , col ]

**Dictionary**

**Record (tuple)**

CODE TO REMEMBER

|  |  |
| --- | --- |
| A black background with white text  AI-generated content may be incorrect. | Repeat until user enters nothing  (use at start of code) |
|  | RANGE |
|  | ARRAY 3x3 |
|  | RECORDS |
|  | DICTIONARIES |

ASSESMENT TASK 1 WHAT WE KNOW

Creating an application that is a quiz