Neo-waterfall Outline

Kurtis Lloyd

1. Requirements(What does the Program do?)

Format of Requirements:

Introduction

Charter(Explain functions, attributes, constraints, and Preferences)

List Functions[attributes][constraints] / Preferences

(Brainstorm to gain input)

Functions: Sentence saying what the system should do. (Verb)

Add attributes to the function(Attributes need constraints) Example(-Fly –High[>5000ft])

Preference(Function that cannot be constrained)

1. External Specifications( What does it look like when it its doing it?)

Format of External Specs

Introduction

Charter(Explain interface vocabulary)

Table of Contents(List of interfaces)

Section for each interface

Compare requirement with external spec.

1. Design(How will it work)

Abstraction is extremely important for design.

Take the system and decompose it.

Start with top Level object:

Start with main: design your entire program in 5-7 lines

(Circle the nouns)

(Underline the Verbs)

Examples:

Nouns:

Game

Player

Players

Verbs:

Setup(game)

Construction

Create Board

Create Pieces

Place Pieces on Board

Create Players

REPEAT with next nouns and verbs

Rules for pseudocode

1. Avoid passive voice ( No subject in the sentence)
2. Loop… use … to show loop.
3. No Programming
4. If\_\_\_ , (comma at the end
5. Otherwise, (else statement)
6. Implementation

Translate PsuedoCode to Code

1. Verification/Testing (Does it work like the external spec or match the spec)

Axioms for Testing:

Guiding Philosophy: Guilty until proven innocent.

We are trying to:

Build the right thing

Build the thing right

-Try all boundary values for all inputs

-exercise all conditions in code

Boundary value analysis

Make sure all necessary code exists

For each boundary value of input type

Test software with the boundary value

Test extremes and norms

Condition Coverage

Make sure all the code works as expected

Make sure to test true and false conditions to make sure its correct

(White box testing)

Done after every boundary and condition is covered.

Practice for:

Worry free software

Finite test cycle to estimate

Safety new when refactoring

Improved design skills