ECE 3311 Software engineering 1, HomeWork 2, Northeastern Univ. Boston, Massachusetts

Nasser M. Abbasi

11/6/93 Compiled on November 16, 2018 at 11:30am [public]

1 Problem 10.3

Question:

Discuss the differences between object-oriented design and function-oriented design.

Answer:

In functional design, the system is composed of a collection of function, in hierarchical fashion, the system is designed from a functional viewpoint, the high level view is progressively broken into more detailed design, this results in top level functions calling lower level function.

Also in functional design, the collection of functions making up the system share a central state of the system, since the system state is centralized. little data hiding exist, with functions having knowledge of the internal structures of the data being passed between the functions either via global sharing or via parameter passing.

In object-oriented design, the system is viewed as a collection of objects, each independent of the other object in that each maintain its own state within the object, this state is not shared with other objects, this means OO design will lead to more loosely coupled design, also in OO design, each object has a set of attributes associated with it, an object can be created from a given object that will inherit some or all of its attributes.

In OO design, object conceptually communication by exchanging messages between each others. also, in OO design, data hiding and data abstraction is used more frequently than in functional design, this is due to that most languages used for OO implementations facilitate this by having constructs in the language itself that allows having parts or all of the data to be private to the owner of the data, this leads to more maintainable software, since the design is more loosely coupled than in functional oriented design.

2 Problem 10.6

Question

What do you understand by the terms cohesion, coupling and adaptability. Explain why maximizing cohesion and minimizing coupling leads to more maintainable systems.

Answer

Cohesion of a component is a measure of how well the functions within this component fit together . if the all the functions within the one component all work together to implement one single logical function then we have high cohesion.

Coupling is an indication of the strength of the interconnections between the program units that make up the system , highly coupled systems have strong interactions between different program units, and have inter-dependability between the program units. A loosely coupled system is one which is made up of units which are independent or almost independent from each others.

Adaptability is a measure of how easy it is to incorporate changes to the design in all design documents. It means that the design should be well documented, the component documentation should be understandable and consistent with the implementation, and the implementations should be easily readable. A design that is adaptable is one that has low level of coupling between the program units, the reason for this is that it is easier to make design changes when the program units are not interdependent on each others, so that changes in one unit will not have unforeseen or hard to detect side changes in other parts of the system design due to coupling. For optimum adaptability, a component should be self-contained as much as possible.

When the system have high cohesion this means the functions that implement one logical function are located in one component in the system, this makes maintenance easier because if changes have to be made in the way this logical functionality is implemented, changes will all be contained in this one component, changes are localized, and since the functions that implement one logical function are in one component, this means the maintainer do not have to understand the over all system design to be able to make changes to one of the logical functions the system does, the maintainer will only need to acquire knowledge of this system component, this makes it faster for a new programmer to maintain this system, and makes the chances of inducing errors to the system when making changes to it less likely than with a system who has less cohesion.

When the system have low coupling, this will also lead to better maintainable system. this is because low coupled system means that the program unit have little inter-connections between themselves, less data sharing and less control inter-dependence, so changes to one program unit will have no or minimal impact on the other parts of the system, this means that to maintain this system, the maintainer will not have to understand the overall details of the whole system design before making changes to a small part of it as the case would be in highly coupled system. So low coupled system design will lead to an easier to maintain system, and a system that is easier to understand for the maintainers than otherwise.

3 Problem 12.4

4 Problem 13.3D

5 Problem 14.4

Question:

Discuss the advantages of graphical information display and suggest four applications where it would be more appropriate to use graphical rather than digital display of numeric information.

Answer

In looking at information, many people are interested in trends or anomalous figures rather than just looking at the values themselves, so graphical display of such information as in a histogram or a pie charts will make the trends or the anomalous appear more clearly .

In the case of dynamically varying numeric information it is better to use graphical display, this will give the user some sense of the relative values of the numeric information, which can be easier shown in graphical display such as a dial with needle, or a moving horizontal bar than by just displaying the information and the user having to manually deduce the relative values of this data.

In the case when precise text information need to be presented to get the user attention, graphical display can be used and accompanied with either highlighting of the text or by drawing a box around the text information, this will make the information more visible and easier for the user to notice. Colors also can be used in graphical display to add an extra dimension to help in displaying complex information.

6 Problem 14.7

Question:

Consider the error messages produced by MS-DOS, UNIX or some other OS. suggest how these might be improved.

Answer:

MS-DOS has simple error messages. these are the main 3 types of error messages on MS-DOS, and how they can be improved:

user input:

```
$>DIR foo
```

system response: *Bad command or file name*

This is somewhat a negative error message, instead it can be improved like this:

improved error message: *System unable to locate this file or command, Please type HELP for information about commands available on the system or DIR to find what files available in your directory and try again.*

user input:

```
$>HELP foo
```

system response: *help not available for this command*

the system should indicate to the user to try help to find what commands available on the system. the user could be an unexperienced user and not aware that they can get a list of what commands the system recognizes by ask HELP to list these commands.

user input

```
$>A:
```

here the user wanted to use drive A, assuming that no floppy disk was inserted in the floppy drive before issuing this command, MS-DOS will issue this command.

system response:

Not ready reading drive A

Abort,Retry,Fail?_

here the system is awaiting user input after the "?" , this error message is confusing since no explanation is given to the user as to the meaning of each option, the user does not know what will happen if they choose say Abort instead of Fail, it is also not clear if the user is asked to type only the first letter of each option given or the whole word. Many new users end up trying the first option "Abort", then the next option "Retry" then "Fail", until something happens. This error message should be made more clear what each option means in simple terms so that new users to the system can react .

After typing "fail" to the above system error message, another system message come up:

system response

```
Current drive in no longer valid_
```

Here this message is even more confusing that the original error message , the system is waiting to the user to input some response after the "> " prompt at the end of the above message, but

no indication is given as what the user is supposed to type, after experimenting with this command, the system was expecting the user to type in the name of a drive that is valid and on line, such as drive C, which is the hard disk drive by default. this message can be improved like this:

improved system response:

```
Current drive can't be used, Please
type a name of a drive you wish to use
```

as a result of inadequate user interface and system error messages on MS-DOS, the window interface was developed to make using MS-DOS easier for the general public. MS-windows is the result of this effort.

7 Problem 15.1

8 Problem 16.2

Question:

Write a set of guidelines for the Pascal or C programmers when writing functions or procedures which describe how to make these functions or procedures more reusable.

Answer:

I'll look at the C language. To make the function or the procedure more reusable, one must keep in mind the wide range of programming languages available and try to write the function or the procedure so that it can be used and called from a different function written in different language other than C. the set of guidelines are:

1. A reusable C function or procedure that accepts an array as one of its parameter, must also accept the array size as another parameter. this will apply to both character arrays (strings) and non-character arrays.

2. A reusable C function or procedure must not assume that a string being passed to it is NULL terminated as the case would be in most C programming software, but instead, the C unit must use the array size passed as a separate parameter to show the size of the array.

3.A prototype function or procedure definition must be defined to show clearly how the interface is designed.

4.Complete details on how to use this function or procedure must accompany it, this documentation must clearly show what the actual parameters are, to describe them, and describe the access mechanism by which they are being passed, such as by value or by reference, this way the user of the unit knows clearly what method to use to pass the argument to the C reusable function or procedure.

5.The C reusable unit should NOT use any platform dependent system services, but only use standard C ANSI run time library routines that are guaranteed to exist in other platforms. the documentation should indicate what C language standard revision is being used.

6.Only use ANSI C language construct in writing the reusable C unit. This guarantees that the unit can be ported to other platforms and used on that platform.

7.Write the C reusable unit to implement one simple logical function, do not write a unit that implement many functions at the same time, as an example, when writing a file I/O reusable C units, write one function to implement an OPEN file operation, and another function for CLOSE file operation. this allows the user to use these functions as building blocks to build larger systems with.

8.For functions, the return values should be made clear as the their Values, do not return platform dependent error or status codes, and if the function is returning a constant, this constant values must clearly be indicated in the documentation section of the unit.

9.Do not use variable arguments for the C function or procedure, in some C implementations, one can define a function or a procedure that accepts variable number of arguments (this is the case for example with printf, it accepts variable number of arguments), but the C reusable unit should have fixed number of arguments, this is because the caller might be using a language other than C that does not allow variable arguments procedure r functions to be called.

10.the reusable C unit should not make assumptions as to how many bytes it takes to represent a pointer or an integer, on some platforms that uses 16 bit hardware, a pointer is 2 bytes long, and an integer is 2 bytes long as well, while on 32 bit hardware, the pointer is 4 bytes longs, and so fourth, instead, the reusable C unit must use library macros that expand at compile time to the actual values of these hardware constants, this way when the C reusable unit is taken to a different platform to be used and compiled over on that platform the correct values are used. This also applies to other platform dependent constants such as the largest value of

signed integer, and the smallest value of float values, and other such constants.