Learning Log

Shazil Arif

March 12, 2020

The purpose of the learning log is to reflect upon your progress in learning the content of SE 2AA4/ CS 2ME3. This is a personal journal. The intention is for you to be aware of your progress by means of recording and reflecting. A template is provided for each week. You should fill in the question marks. You are also free to add your own subsections.

1 Week 1 Intro to Course

Dates

Jan 6 to Jan 10

Lecture 1 Introduction to Course

Discuss Course administrative details, marking scheme, material and content to be taught

Lecture 2 Software Engineering Profession

Discussed the differences between studying Computer Science and Software Engineering, History of Software Engineering and some important figures such as Parnas

Tutorial 1 Git, Doxygen and A1

Learnt how to install Doxygen, Tex, Git and set up necessary tools and development environment to complete assignments

Textbook Reading (Ghezzi, H&S or other)

I did not read the book this week

Assignment Progress

Finished the coding and testing portions

Midterm/Final Review Progress

Have not started

Reflection Relating Course Topics, Other Courses, Other Experiences

The discussion board is quite helpful. I find that this course really helps you understand deeply what "Software Engineering" really is. It focuses on the practical aspects of the profession rather than just programming.

2 Week 2 Software Qualities, Software Engineering Principles

Dates

Jan 13 to Jan 17

Lecture 3 Software Qualities

Discussed Software qualities such as correctness, robustness, reliability, portability, maintainibility, etc.. Comparing and contrasting different terminology

Lecture 4 Software Engineering principles

Discussed key SE principles including abstraction, information hiding, designing for change, seperation of concerns and more

Tutorial 2 Basics of Latex and PEP8 convention for Python

Learnt the basics of latex, syntax, different editors etc. Disucssed the PEP8 standards

Textbook Reading (Ghezzi, H&S or other)

Have not started :-(

Assignment Progress

Nearly complete. missing a few tests for pos adt

Midterm/Final Review Progress

Have not started reviewing

Reflection Relating Course Topics, Other Courses, Other Experiences

It is nice to see many different software engineering principles, methods and practices being defined in detail and how to apply them

3 Week 3 Introduction to modules and Mathematics for MIS

Dates

Jan 20 to Jan 24

Lecture 5 Introduction to Modules

Important goals to keep in mind when developing software such as Design for change and Product families. Discussed The module interface, module implementation

Information Hiding: Basis for design Implementation secrets are hidden from clients Encapsulate changeable design decisions as implementation secrets within module implementations Encapsulate changeable design decisions as implementation secrets with module implementations

The WRONG ANS: HAS NOTHING TO DO WITH Security and HIDING DATA, VARIABLES

Important for midterm! internalize it

Discussed examples of modules such as record, library, abstract data type, generic modules Note: follow precise terminology from Ghezzi textbook

Difference between a library and module

Library: Has no state information or record of any stored data. E.g a Math library that has functions that take inputs and gives outputs

Module: Has state information and some record of data (a ADT module?)

When implementing a specification must match it, not look like it

Lecture 6 Mathematics for MIS

Worked through an example of balancing chemical equations to demonstrate how we can take a problem, describe it in mathematical terms and syntax and from there translate to an actual program/code

Lecture 7 Module Interface Specification

Worked through an example of defining an abstract data type for a circle
Note : MIS is not giving the implementation, it only defines the interface! e.g MIS may give a specification isbalanced() that returns whether an equation is balanced..But it is not specified how to achieve this, it us up to the developer to figure out the implementation!

An abstract object is programming terms is a module where this only one instance/singleton pattern

Tutorial 3 Math Review

Reviewed mathematical operators, unary and binary operators ad their precedences

Discussed what a set is: 1) Distinct elements (i.e no elements are repeated) 2) All elements are of the same type

Operations on sets: Union: essentially combine two sets Intersection: elements in both sets Set Difference: Take first set and remove any elements that are common with other set e.g if we have S= 1,2 and T = 2,3,4 then we have S - T = 1 . 3 and 4 not included cuz not in both sets

Subset

Cartesian product: all possible pairs

A set can be described in two ways: set enumeration: List out all elements in a set

Set comprehension: $S = x : t | R : E$ This means S is a set where its elements are of type t and satisfy a property R and E is some defining expression for a set element e.g $S = x : N | 1 <= x < 5 : x^2$ then $S = 1, 4, 9, 16$

Types: A set of values e.g a value of type integer belongs to the set S = ...-1,-2,0,1,2....

We can have custom types: Such as a PointT type which can be a tuple(x : R, y : R)

Quantifiers (Shorthand for applying the same operator many times)

(*x : X — R : P) x is an element of type X R is a range (usually a boolean condition indicating which elements to include/consider) P - the values to apply the operator "*" to. * may be +,-, / etc. e.g +x : N — 1¡= x ¡ 5 : x^2 means to sum up the square of the terms from 1 to 5 (including 1 but not 5)

Quantifiers for conjunction and disjunction for logical and, we use the universal quantifier forall, since "and"

for logical or, we use existential quantifier, exists since "or"

Textbook Reading (Ghezzi, H&S or other)

Did not read

Assignment Progress

Complete

Midterm/Final Review Progress

Reviewing principles

Reflection Relating Course Topics, Other Courses, Other Experiences

Discussing modular design and information hiding tied in with what is currently being taught in our 2XB3 course, modular design and object oriented programming with Java. The overlapping material helps build a deeper understanding!

The Math review helps tie in with other courses such as Discrete Mathematics (2FA3). It is very interesting to see how to take a high level specification given in mathematical terminology and syntax and actually translating it to a program.

4 Week 4 Abstract Data Types

Dates

Jan 27 to Jan 31

Lecture 8 Implementing ADT's in Python

H and S notation $s[0:0]$ includes the upper bound. Python notation $s[0:n]$ goes up to n - 1

Looked at different examples of implementations of ADT's such as a (x,y) point, a line ADT

A mutator function will have a transition state (since they modify the state)

can use x,y,z point in 3d space to define latitude longitude positions, and use complex math to transform it into 2d coords

@staticmethod?

Implementation and specification are very different! Keep this in mind
Abstract object: Singleton, Abstract data type: Instantiable, can have multiple copies

Lecture 9 More on ADT's and Assignment Discussion

Reviewed the difference between an abstract object and abstract data type.

Difference between a reference and actual value. Aliasing and pointers.

Generic Modules. What if we want a stack module for booleans, integers, stacks, strings?...

A state invariant is a proposition/expression that always evaluate to true. the state variable may change but the invariant holds true. Example for a generic stack module the invariant is size <= MAXSIZE.

Module state machine

Lecture 10 A2 discussion and Classes, Interfaces in Python, UML

Discussed Assignment 2 details and how to work with classes, interfaces and inherited classes in python.

UML.
Difference between type and a class. Types are known at compile time Class of an object may only be known at run time

Tutorial 4 Mathematical notation, MIS

Discussed the MIS.
Worked through last years Assignment 2 to aid in our assignment and translating MIS specification to code.

Textbook Reading (Ghezzi, H&S or other)

Did not read

Assignment Progress

Finished Assignment 1 + report. Started working on Assignment 2

Midterm/Final Review Progress

Did not review

Reflection Relating Course Topics, Other Courses, Other Experiences

Not much to comment, understood what a invariant is which tied in with discrete mathematics concepts also understood a bit of referencing/aliasing in python and pointers which tied in with the C programming course.

5 Week 5 Functional Programming and More Abstraction, MIS stuff

Dates

Feb 3 to Feb 7

Lecture 11 Functional programming in Python

More on UML, MIS, interfaces and abstraction.
Discussed more about generic types, answered some MC questions Discussed some functional programming, same some live code examples of functional programming in python

Lecture 12 ?

Did not attend

Lecture 13 Modules with External Interaction

reduce and lambda (anonymous functions) in Python

Tutorial 5 Unit testing in python with pytest

Blackbox vs whitebox testing.

Unit testing allows to identifying bugs in code

pytest

all tests are in a class
each test has its own method
setup method(self,method):
do something before every test. e.g. define some state variables
teardown method(self,method)
do something after every test e.g.reset state variables
assert pytest.approx pytest.raise(exception name)

Have acceptable code coverage

Textbook Reading (Ghezzi, H&S or other)

Did not read

Assignment Progress

Almost complete Assignment 2

Midterm/Final Review Progress

None

Reflection Relating Course Topics, Other Courses, Other Experiences

Interesting to see the concepts of interfaces and generic modules tie in with other courses such as 2XB3

6 Week 6 Assumptions, Exceptions, Module Interface Design

Dates

Feb 10 to Feb 14

Lecture 14 Module Interface Design

Assumptions vs Exceptions
For assignment 2 report refer to slide 7/20 in Lec 16 for "critique the design"

Quality Criteria of a modules *interface*
Essential - Omit unnecessary features, functions etc, no redundant features
If asked if a module interface is essential, need to answer "can i remove features and still meet the specifications", whether or not a feature is a good idea/design choice is not the question.

Slide 10/20 Lec 16 , Tuple Idiom Version - too many setters
Minimal - more than one state transition -¿ not minimal. state transition and printing something. Avoid independent services .

General - e.g. functional programming makes interfaces more general.
Opaque - Satisfies information hiding.

Lecture 15 ?

Did not attend due to a busy week. midterms + assignments

Tutorial 6 ?

Did not attend, same reason as lecture above

Textbook Reading (Ghezzi, H&S or other)

Read about object oriented design

Assignment Progress

Working on the report for assignment 2!

Midterm/Final Review Progress

Have not started

Reflection Relating Course Topics, Other Courses, Other Experiences

Not much to reflect this week since I did not attend 2/3 lectures and the tutorial. Not something that I will continue tho!

7 Midterm Break

Dates

Feb 17 to Feb 21

8 Week 7

Dates

Feb 24 to Feb 28

Lecture 16 Maze Tracing Robot Example

Steps in design:

1. Anticipated changes
2. No template keyword = not an ADT but an astract object
3. Some access routines are redundant because they will be only called once (e.g reading file and populating the maze)
4. slide 16/23 (Maze Tracing Robot Example) the exception statement is an "if-else statement"
5. Potential Midterm question slide 19/23 (Maze Tracing Robot Example)

Lecture 17 Specification via UML

1. Generics and interfaces in Java
2. extends keyword
3. objects are compared via their reference (location in memory)

Lecture 18 Specification via UML

1. More on UML diagram
2. Measurable interface can be implemented and used by BankAccountT and PointInterfaceT
3. strategy is something you can change at run time

Tutorial 7 A1 solution and Midterm Review

1. Discussed A1 and common mistakes
2. worked through 2017 midterm with a polling setup to vote for answers.//

   2017 Midterm answers and explanations

   (a) false because x was mutated and then the UPDATED value was used when calculating y

   (b) False

   (c) B

   (d) B, key idea is that we want to keep the same external interface so only getters need to change

   (e) D, option A and C do the same thing

   (f) B, Always false

   (g) Q7 ?

   (h) Q8, Answer: E, reduce takes two arguments: 1) a function with two parameters(binary function) 2:) a list that the binary function is applied to

   (i) B, False

   (j) Q10, True, If any access programs behavior can be achieved by a combination of any other program, invariant etc. then we can remove it (probably)

   (k) C, D is wrong because every implementation should be independent of the interface

   (l) C, it is possible to keep the same number of access routines, e.g. if we split up one but dont add a routine for all the independent services

   (m) D, fast for set membership because we can do binary search. C is false because it is an implementation detail does not concern with the fact order in a set matters

(n) Answer: E, Read up on Parnas Style Diagram?

(o) Lookup Book notation for list slicing, Answer: C

(p) Book slice notation includes the last element A[1:n] includes n. Set comprehension $x : T|Condition : Expressionsointhiscasewareaddingjbasedongivencondtion, so$

(q) A, Q19, env variable is used when something external to the software is changed, like the output monitor or speaker etc.

(r) True, Because the "uses" module is used within maze

(s) Q22, D because B and C are the same thing

(t) D is true, why is C false?

(u) Q24, Answer: E

(v) Q25, Equality defined if two objects point to same location in memory, Answer: A

(w) Q26, Read carefully "return set of cells" thus C and D are wrong

Textbook Reading (Ghezzi, H&S or other)

Reviewing parnas papers, lectures slides and Ghezzi textbook chapters for midterm

Assignment Progress

Working on the A3 specification

Midterm/Final Review Progress

Worked on 2017 midterm, reviewing lecture and topics covered for Thursday March 5

Reflection Relating Course Topics, Other Courses, Other Experiences

Not much to reflect, interfaces and classes were covered in the 2xb3 course, which was the focus of this course for this week, so it was a nice reinforcement of what was already learned. Also interesting to compare and contrast python and Java and differences when implementing generic interfaces due to dynamic vs static typing

9 Week 8 Midterm Exam Week

Dates

Mar 2 to Mar 6

Lecture 19 Midterm Review

1. Passive vs active learning
2. Local functions do not have to be implemented in code
3. Ghezzi Chapter 1 to 4, Hoffman and Strooper (Ch 3,6,7, Ch1,2)
4. lectures and tutorials
5. Emphasis on lecture concepts
6. Faked rational design process (important)
7. Dynamic binding Some types are unknown and can change at runtime

Lecture 20 Design pattern

1. In UML italicized method names are abstract
2. Arrow indicates inheritance relation
3. Can add fly() so all ducks can inherit it
4. Can be a bad solution because fly() needs to be modified for many subclasses
5. Can potentially add a interface
6. Dotted line is implements (for a interface), solid is extends/inherits (superclass)
7. Inheritance is usually a "is a" relationship. E.g. Car is a Vehicle
8. Composition instead is "has a" relationship. E.g. Duck has a Fly method

Lecture 21 Design Patterns continued

1. WE are not reusing a design, we are reusing a pattern
2. Abstract object, Generic abstract objects, ADT, Generic ADT's, Interface, Libraries, Records
3. Slide 7/37 Lecture ? Exam question. Ans: True
4. Strategy Design pattern
5. MVC. Seperate computational elements from I/O elements.

(a) The UI/ User Interface. Displaying data etc. User interacts with view

(b) Handles changes/actions , the application /business logic

(c) Model changes/updates data

6. 3 types of Design patterns

   (a) Creational Design pattern

   (b) Structural Design pattern

   (c) Behavioral Design pattern

7. Proxy Pattern. There is a middle man that gets the required data/result for a client but pretend as if it is being accessed directly. Example getting data from a embedded device from web interface through some IoT/http protocol.

8. a REST API is proxy pattern?

9. Client => Proxy => Real Subject

10. Adapter Pattern

    (a) Motivated by the problem of different wall outlets, changing adapter. Only the physical interface matters. Power is not an issue

    (b)

11. "Singleton" Pattern. For abstract objects

Tutorial 8 Java concepts

1. Doxygen comments in java

```
/**
* Author: Name
* Revised : Data
*
* Description: text
*/

/**
* @brief
```

```
* @param
* @return
* ...
*/
```

2. Exceptions are thrown with throw new exceptionname(string Description)
3. Worked through last years excercises
4. module called Test* in file named Test*.java
5. import org.junit.*
6. import static org.junit.Assert.*;
7. can create global variables to be used in test
8. setUp() is JUnit equivalent to setup method, set global variables here

```
@before
setUp(){
}
```

9. Every method begins with @Test
10. assertTrue(boolean value)
11. assertFalse(boolean value)
12. assertEquals(t1,t2);
13. JUnit documentation (T08 slide 17/22)
14. assertEquals(t1,t2, TOLERANCE)
15.

Textbook Reading (Ghezzi, H&S or other)

Did not read

Assignment Progress

Completed Part 1 the Spec and Critique Questions

Midterm/Final Review Progress

Reviewing for midterm

Reflection Relating Course Topics, Other Courses, Other Experiences

?

10 Week 9 ?

Dates

Mar 9 to Mar 13

Lecture 22 Intro to specification

1. Adapter pattern?
2. Specification is a broad term meaning definition
3. Specification document should be maintainable, reusable and usable
4. Specifications should be:
   (a) Clear, unambigious and understandable
   (b) Consistent, complete (internal and external)
   (c) Incremental
   (d) Validatable
   (e) Abstract

Lecture 23 Intro to spec continued

1. problems with specification
2. External and Internal completeness
3.

Lecture 24 Intro to verification

1. Verification can help build confidence in our programs
2. Verification: Are we building the product right? Implementing requirements correctly
3. Validation: Are we building the right product? Getting right requirements?
4. e.g. given a equation are we solving it right? (Verification), is this the right equation for this scenario (validation)
5.

Tutorial 9 MVC and Observer pattern

1. Benefits of MVC allows for parallel development for rapid application development
2. Model can includes schemas, classes, templates, ADT modules
3. View (Output)
4. Controller
5. Observer Pattern:
   (a) There is a Observer (example a display) that responds to changes in a "subject"
   (b) Subject's state changes (maybe data changes, example WeatherData object whose temperature, humidity changes)
   (c) The subject pulls data and allowed the observer to update. Slides T08

Textbook Reading (Ghezzi, H&S or other)

?

Assignment Progress

?

Midterm/Final Review Progress

?

Reflection Relating Course Topics, Other Courses, Other Experiences

?

11 Week 10 ?

Dates

Mar 16 to Mar 20

Lecture 25 ?

?

Lecture 26 ?

?

Tutorial 10 ?

?

Textbook Reading (Ghezzi, H&S or other)

?

Assignment Progress

?

Midterm/Final Review Progress

?

Reflection Relating Course Topics, Other Courses, Other Experiences

?

12 Week 11 ?

Dates

Mar 23 to Mar 27

Lecture 27 ?

?

Lecture 28 ?

?

Tutorial 11 ?

?

Textbook Reading (Ghezzi, H&S or other)

?

Assignment Progress

?

Midterm/Final Review Progress

?

Reflection Relating Course Topics, Other Courses, Other Experiences

?

13 Week 12 ?

Dates

Mar 30 to Apr 3

Lecture 29 ?

?

Lecture 30 ?

?

Tutorial 12 ?

NA

Textbook Reading (Ghezzi, H&S or other)

?

Assignment Progress

?

Midterm/Final Review Progress

?

Reflection Relating Course Topics, Other Courses, Other Experiences

?

14 Week 13 ?

Dates

Apr 6 to Apr 7

Lecture 31 ?

?

Tutorial 13 ?

NA

Textbook Reading (Ghezzi, H&S or other)

?

Assignment Progress

?

Midterm/Final Review Progress

?

Reflection Relating Course Topics, Other Courses, Other Experiences

?