**NAME: Carlos Minimah**

**MATRIC NUMBER: 18/sci01/102**

**COURSE CODE: CSC308**

**COURSE TITLE: Formal Methods And Software Development**

**ASSIGNMENT**

**Question**

1. Briefly distinguish between the axiomatic and model oriented approach to Formal Methods.
2. You are on a team that is working on a mission-critical system. You have a simple task of suggesting one of the two approaches to use to verify your specifications. Which of the two approaches are you going to advise them to use and why?
3. Using the Z Specification, model the process of using a library.

ANS:

Number 1:

|  |  |
| --- | --- |
| AXIOMATIC ORIENTED APPROACH | MODEL ORIENTED APPROACH |
| The axiomatic approach focuses on the properties that the proposed system is to satisfy | The model oriented approach is the specification or abstraction of a model/entity of the real world that contains the essential details. |
| It has the advantage that the implementer is not constrained to a particular choice of implementation | They serve to explain the behavior of a particular entity |
| It does not support future behavior predictions | It may be used to predict the future behaviors of the entity  |

Number 2:

I would suggest the use of the model-oriented approach to the system in the question above. This is due to the fact that the model-oriented approach will first; produce solutions that can be used again. And also allows simplification and abstraction of more complex systems.

Number 3:

SOLUTION:

* Assume the following types for library users and library books:

 [USER, BOOK]

* Define MESSAGE as follow:

MESSAGE :: = ‘OK’

| ‘Book not available’

| ‘Invalid return

* Define the abstract state of the library system

LibSys

available: P BOOK

borrowed: BOOK 7→ USER

available ∪ dom borrowed = BOOK

available ∩ dom borrowed = ∅

* Define ∆LibSys

∆LibSys

available, available ′ : P BOOK

borrowed, borrowed ′ : BOOK 7→ USER

available ∪ dom borrowed = BOOK

available ∩ dom borrowed = ∅

available ′ ∪ dom borrowed ′ = BOOK

available ′ ∩ dom borrowed ′ = ∅

* Define ΞLibSys

ΞLibSys =b

[∆LibSys | borrowed ′ = borrowed ∧ available ′ = available ′ ]

* Define the initial abstract state

InitLibSys ′

LibSys ′

available ′ = BOOK

borrowed ′ = ∅

* Defining CheckOut operation (successful case)

CheckOutOK

∆LibSys

u? : USER

b? : BOOK

rep! : MESSAGE

b ? ∈ available

available ′ = available \ { b ? }

borrowed ′ = borrowed ∪ { b ? ֏ u ? }

rep! = ‘OK’

* Error case: b ? 6∈ available

NotAvailable

ΞLibSys b? : BOOK

rep! : MESSAGE

b ? 6∈ available

rep! = ‘Book not available’

* A total definition of CheckOut

CheckOut =b CheckOutOK ∨ NotAvailable

* Define ReturnOK for returning a book (successful case)

ReturnOK

∆LibSys

u? : USER

b? : BOOK

rep! : MESSAGE

(b ? ֏ u ?) ∈ borrowed

available ′ = available ∪ { b ? }

borrowed ′ = borrowed \ { b ? ֏ u ? }

rep! = ‘OK

* Error case: recording an incorrect return

InvalidReturn

ΞLibSys

u? : USER

b? : BLOCK

rep! : MESSAGE

(b ? ֏ u ?) 6∈

borrowed

rep! = ‘Invalid return’

* A total definition for Return

Return =b ReturnOK ∨ InvalidReturn