Canterbury Institute of Management (CIM)

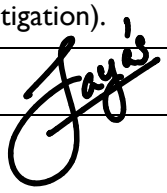
ASSESSMENT COVER SHEET

1. Personal Details			
Student ID	**Given Name(s)**	**Surname**	**Email Address**
1. CIM12137	Ayesh Chathuranga	Jayasekara	cim12137@ciom.edu.au
Campus	Darwin Campus		
Course Title and Code	MSIT401 System Development Methodologies		
Assessment Title	Reflective Journal - *Week 7*		
Due Date & Time	17/11/2024		
Course Lecturer/Tutor Name: Sharad Neupane		Assessment Word Count (if applicable): 579	

2. Student Declaration

By signing and submitting this coversheet, I/we declare that:

- ✓ This assessment submission is my/our own work unless otherwise acknowledged (including the use of generative AI tools) and is in accordance with the Institute's ***Academic Integrity and Honesty Policy*** available on the website.
- ✓ No part of this assessment has been submitted previously for advanced standing or academic credit in this or any other course.
- ✓ I/we certify that we have not given a copy or have shown a copy of this assessment item to another student enrolled in the course, other than members of this group.
- ✓ I/we are aware that the Lecturer/Tutor of this assessment may, for the purpose of assessing this assessment task communicate a copy of this assessment task to a plagiarism checking service to detect possible breaches of academic integrity, for example, plagiarism, recycling, cheating, contract cheating, or unauthorised use of generative AI (which may then retain a copy of the item on its database for the purpose of future investigation).

Signature:		Date:	11/11/24

Canterbury Institute of Management
63 Oxford Street, DARLINGHURST NSW 2010 | 21 Regent Street, REDFERN NSW 2016 | 2 Printers Place, DARWIN CITY NT 0800 | 399 Lonsdale Street, MELBOURNE VIC 3000
Phone: +61 2 8007 5846 | Website: https://ciom.edu.au/
TEQSA Provider ID: PRV14258 | CRICOS Provider Code: 03809A

MSIT401 System Development Methodologies

Reflective Journal - *Week 7*

Ayesh Jayasekara - CIM12137

Decomposing Internal Composition of IT Systems

To describe and define internal components of a software system, number of techniques could be used. These includes,

- Defining classes, methods, attributes (Object Oriented Analysis)
- Drawing class diagrams
- Drawing UML Diagrams such as use case diagrams, class diagrams, sequence diagrams, state transition diagrams, activity diagrams and business process models

Object Oriented Analysis

Object Oriented Analysis is the pretext for Object Oriented Programming. When designing a software solution, it important to understand how we can create a blueprint of real world scenario (i.e a class or collection of classes as a package). As a part of analysis, we can produce class diagrams to understand how we can logically devide the attributes, methods, tasks to avoid repetetion and complexity of the actual solution.

This means we can create classes diagrammatically to include,

- Class Name - Logical name for the object model
- Method - Functions defined to act on messages and produce output or modification to an attribute
- Attribute - Logical atomic piece of information represented by native data type or data structure

These classes then can be generalized (bottom-up) or specialized (top-down) considering their feature (method or attribute) similarity to reduce repetition and redefinition. This process leads to the object oriented concepts such as,

- Abstraction - Generalization of action (method) or class
- Encapsulation - Conceal attributes or methods for a class leading to separation of concern

- Inheritance - Creates parent child relationships among classes enabling generalization or specialization to take place with minimal repetition
- Polymorphism - Enables to define and redifine methods according to the needs

UML Diagrams

UML *(Unified Modelling Language)* is a standardized set of diagrams that includes clearly defined set of symbols and guidelines to make sure every reader comprehend same meaning looking at these diagrams. This rules out ambiguity of describing IT Systems using plain english.

Some of popular UML diagrams are,

- Use case diagrams - Used to define actors (human users or external systems), functions, their interaction and general system boundary of a system
- Class diagrams - Define & describe internal logical components in least technical diagram rather than a program code
- Sequence diagrams - Used to establish on order of events and dependency actions for tasks being performed
- State transition diagrams - Defines the state of the system or the function on every state for every input or result generated from start to end of a process
- Activity diagrams - Illustrates each activity and their general order of execution and conditional alternative execution paths
- Business process models - Used to establish on business processes that is later defined into technical diagrams using one or more diagrams described above

UML also defines how the relationships are generally indicated on diagrams which is noted as cardinality of relationship where applicable. When considering two classes with a relationship it can be in either,

- Zero to Many relationship *(0..*)*
- Zero to One relationship *(0..1)*
- One to One relationship *(1..1)*
- One to Many relationship *(1..*)*

UML and technical descriptions

Further reading according to (Barclay & Savage, 2003), UML does not define how software are developed but how it can be done independent from process of development. Meaning it provides guidelines to divide and conquer large complex solutions.

UML plays a vital role in designing systems and documenting theirs internal functions for both technical and none-technical personals therefore, a must have for any solution project or task regardless of the size or context.

It is critical that these diagrams are include in any system design document from the day one starting from perhaps the simplest context diagram.

Bibliography

Barclay, K., & Savage, J. (2003). *Object-Oriented Design with UML and Java* [https://www.perlego.com/book/1884223(visited 2024-11-11)]. Butterworth-Heinemann.