الجامعة المستنصرية

كلية العلوم

قسم:الحاسوب

عنوان التقرير

How to represent design process in UML

أسم الطالب الرباعي: علي لؤي خلف جلوب

التـــــوقيع:

القسم: الحاسوب المـــرحلة: الثالثة

الشعبـــة : A1 الفـــــرع: CS

البريد الالكتروني: luaya577@gmail.com

التقرير مقدم من ضمن متطلبات درجة الامتحان النهائي

لمادة Object Oriented Software Engineering

يملئ من قبل أستاذ المادة					
اسم الأستاذ:					
الدرجة					
من					

ختم اللجنة الامتحانية

توقيع المدقق

Abstract

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.[1]

INTRODUCTION

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.
It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.[3]

History

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates.[2]

What is the use of UML?[2]

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficacy.

UML itself finds different uses in software development and business process documentation:

Sketch UML diagrams, in this case, are used to communicate different aspects and characteristics of a system. However, this is only a top-level view of the system and will most probably not include all the necessary details to execute the project until the very end.

> Forward Design – The design of the sketch is done before coding the application. This is done to get a better view of the system or workflow that you are trying to create. Many design issues or flaws can be revealed, thus improving the overall project health and well-being.

> Backward Design – After writing the code, the UML diagrams are drawn as a form of documentation for the different activities, roles, actors, and workflows.

Blueprint In such a case, the UML diagram serves as a complete design that requires solely the actual implementation of the system or software. Often, this is done by using CASE tools (Computer Aided Software Engineering Tools). The main drawback of using CASE tools is that they require a certain level of expertise, user training as well as management and staff commitment.

Pseudo Programming Language UML is not a stand-alone programming language like Java, C++ or Python, however, with the right tools, it can turn into a pseudo programming language. In order to achieve this, the whole system needs to be documented in different UML diagrams and, by using the right software, the diagrams can be directly translated into code. This method can only be beneficial if the time it takes to draw the diagrams would take less time than writing the actual code.

Despite UML having been created for modeling software systems, it has found several adoptions in business fields or non-software systems.

Types of UML Diagrams[4]

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing, and deployment.
These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

Structural UML diagrams

- **Class diagram**
- **Package diagram**
- **Object diagram**
- **Component diagram**
- **Composite structure diagram**
- **Deployment diagram**

Behavioral UML diagrams

- **Activity diagram**
- **Sequence diagram**
- **Use case diagram**
- **State diagram**
- **Communication diagram**
- **Interaction overview diagram**
- **Timing diagram**

Class diagrams

are the backbone of almost every object-oriented method, including UML. They describe the static structure of a system.

Package Diagram

Package Diagram are a subset of class diagrams, but developers sometimes treat them as a separate technique. Package diagrams organize elements of a system into related groups to minimize dependencies between packages.

Object Diagram

Object Diagram describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.

Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. The image below shows a component diagram.

Composite Structure Diagram

Composite structure diagrams show the internal part of a class.

Deployment Diagram

A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.

Activity Diagram

Activity diagrams illustrate the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. Learn more

Use Case Diagram

As the most known diagram type of the behavioral UML types, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

State Diagram

State chart diagrams, now known as state machine diagrams and state diagrams describe the dynamic behavior of a system in response to external stimuli. State diagrams are especially useful in modeling reactive objects whose states are triggered by specific events.

Communication Diagram

Communication diagrams model the interactions between objects in sequence. They describe both the static structure and the dynamic behavior of a system. In many ways, a communication diagram is a simplified version of a collaboration diagram introduced in UML 2.0.

Interaction Overview Diagram

Interaction overview diagrams are a combination of activity and sequence diagrams. They model a sequence of actions and let you deconstruct more complex interactions into manageable occurrences. You should use the same notation on interaction overview diagrams that you would see on an activity diagram.

Timing Diagram

A timing diagram is a type of behavioral or interaction UML diagram that focuses on processes that take place during a specific period of time. They're a special instance of a sequence diagram, except time is shown to increase from left to right instead of top down.

UML Diagram Symbols[5]

There are many different types of UML diagrams and each has a slightly different symbol set.

Class diagrams are perhaps one of the most common UML diagrams used and class diagram symbols center around defining attributes of a class. For example, there are symbols for active classes and interfaces. A class symbol can also be divided to show a class's operations, attributes, and responsibilities.

Visibility of any class members are marked by notations of

\+ For Public

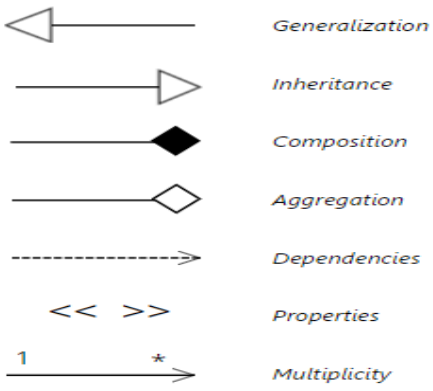
\- For Private

\# For Protected

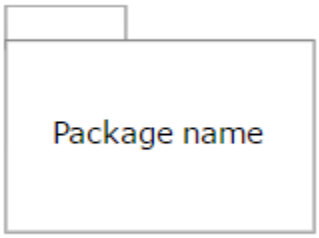
/ For Derived

~ For Package

Lines are also important symbols to denote relationships between components. Generalization and Inheritance are denoted with empty arrowheads. Composition is shown with a filled in diamond. Aggregation is shown with an empty diamond. Dependencies are marked with a dashed line with an arrow. Using << >> allows you to indicate properties of that dependency. Multiplicity is usually shown with a number at one end of the arrow and a * at the other.



Package diagrams have symbols defining a package that look like a folder.



Activity diagrams have symbols for activities, states, including separate symbols for an initial state and a final state. The control flow is usually shown with an arrow and the object flow is shown with a dashed arrow.

Use case diagrams have symbols for actors and use cases.

The use of UML in business process modeling[6]

Business process modeling is the act of documenting the series of steps and actions taken within a business process. This involves:

- **Identifying the actors involved and the roles they play in the process.**
- **Showing how the actors interact with each other through the progression of various activities.**
- **Providing a clear picture of the journey it takes through the organization.**

Tools for drawing UML Diagrams[2]

Just like any other thing in life, in order to get something done properly, you need the right tools. For documenting software, processes or systems, you need the right tools that offer UML annotations and UML diagram templates. There are different software documentation tools that can help you draw a UML diagram. They are generally divided into these main categories:

1.Paper and pen – this one is a no-brainer.

2.Online tools – there are several online applications that can be used to draw a UML diagram.

3.Free Online Tools – these do pretty much the same thing that the paid ones do. The main difference is that the paid ones also offer

tutorials and ready-made templates for specific UML diagrams. A great free tool is draw.io.

4.Desktop application – a typical desktop application to use for UML diagrams and almost any other sort of diagram is Microsoft Visio. It offers advanced options and functionality. The only downside is that you have to pay for it.

main benefits of activity diagrams[6]

1.Activity diagrams are generally far less complicated than other UML diagrams, making them easier for both analysts and stakeholders to fully comprehend.

2.They allow an analyst to display multiple conditions and actors within a workflow through the use of swimlanes.

3.They have the ability to clearly describe the steps performed in a UML use case.

References

1) https://www.sciencedirect.com/science/article/pii/S0065245801800152

2) https://tallyfy.com/uml-diagram/

3) https://www.cs.le.ac.uk/people/rh122/papers/2005/EFHT05PAIS.pdf

4) https://creately.com/blog/diagrams/uml-diagram-types-examples/

5) https://www.smartdraw.com/uml-diagram/

6) https://www.process.st/uml-tutorial/