International Journal of Advanced Research in IT and Engineering

ISSN: 2278-6244
Impact Factor: 6.111

AN OBJECT ORIENTED DESIGN METHOD FOR DATABASE SYSTEMS AND COMPUTER APPLICATIONS

Magilyn A. Racacho, Assistant Professor II, Apayao State College, San Isidro Sur, Luna, Apayao

Abstract: *This study presents the concepts of database systems as well as the overview of the use of Unified Modeling Language (UML) as a standard notation of real-world objects in developing object-oriented design methodology for computer applications. The UML is an instrument for specifying software systems that include standardized diagrams to describe, demonstrate and visually map or model a software system's design and structure. UML diagrams include the use case diagram, class diagram, sequence diagram, state chart diagram, activity diagram, component diagram, and deployment diagram. The use of diagrams to different software processes has been integrated in this study.*

1. INTRODUCTION

Developing an object-oriented design methodology for computer applications as well as for database systems is the use of Unified Modeling Language (UML) as a standard notation for the modeling of real-world objects. Software systems designers and architects are given many choices for providing reliable, flexible and efficient object persistence for computer applications and database systems. They could choose between object-oriented, object-relational hybrids, pure relational and custom solutions based on open or proprietary file formats.

In the Unified Modeling Language (UML), one of the key tools for behaviour modeling is the *Use Case* model, originated from the Object-Oriented Software Engineering (OOSE). The key concepts associated with the use case model are *actors* and *use cases*. The users and any other systems that may interact with the system are represented as actors. The required behaviour of the system is specified by one or more uses cases, which are defined according to the needs of the actors. Each use case specifies some behaviour, possibly including variants that the system can perform in collaboration with one or more actors.

This study presents the layering of an object-oriented class model on top of a purely relational database. The techniques and issues involved in mapping from the class model to the database model have been illustrated, including object persistence, object behaviour,

relationships between objects and object identity. Among the concepts of modeling that UML specifies how to describe are: class (of objects), object, association, responsibility, activity, interface, use case, package, sequence, collaboration, and state [5].

The overview of the database systems, the concepts of Unified Modeling Language (UML) and the diagrams associated has been discussed in this paper. The different UML diagrams for database application are also illustrated that could be integrated to different software development processes.

On the other hand, showing how an existing object oriented design method has been modified and improved to include database systems and computer applications. The objective is to construct an object oriented design methodology where system objects can be seamlessly implemented in both software and hardware.

2. RELATED WORK

A use case is defined in the Unified Modeling Language (UML) specification as "the specification of a sequence of actions, including variants that a system (or a subsystem) can perform, interacting with actors of the system" [1]. A use case is a specification of interactions involving a software system and external actors of that software system. A use case describes the response of a software system to an actor's request [2]. A use case is a kind of system behaviors, which will produce some measurable results for performance [3]. Different works related to use cases are based on a variety of use case notations with formal semantics. In [4, 5] use cases are defined in relation to formal pre/post-conditions. The formalization allows derivation of conceptual models in [4], and test generation in [5]. In [6] a formalization of textual use cases is proposed, an UML meta-model and a restricted-form of natural language are defined for use case description, use cases execution semantics are proposed as a set of Mapping Rules from well-formed use cases to Basic Petri nets. In [7], a lightweight formalism based on the event sequence for use case description is proposed. In [8], a method is proposed to model use case from multi-angle views.

3. DATABASE SYSTEMS

A database is an ordered collection of related data elements intended to meet the information needs of an organization and designed to be shared by multiple users [16]. Database systems reduce data redundancy, integrate corporate data, and enable information sharing among the various groups in the organization. It is a term that is

typically used to encapsulate the constructs of a data model, Database Management System (DBMS) and database [11].

A database is an organized pool of logically-related data. Data is stored within the data structures of the database. A DBMS is a suite of computer software providing the interface between users and a database or databases. A DBMS is a shell which surrounds a database or series of databases and through which all interactions take place with the database. The interactions catered for by most existing DBMS fall into four main groups:

- Data Definition. Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- Data Maintenance. Inserting new data into existing data structures, updating data in existing data structures, deleting data from existing data structures.
- Data Retrieval. Querying existing data by end-users and extracting data for use by application programs.
- Data Control. Creating and monitoring users of the database, restricting access to data in the database and monitoring the performance of databases.

A database and its DBMS are both conform to the principles of a particular data model. Data models include the hierarchical data model, the network data model, the relational data model and the object-oriented data model [11].

4. UNIFIED MODELING LANGUAGE (UML) FOR DATABASE SYSTEMS

Software designers and developers have been provided by Unified Modeling Language (UML) with a stable and common design language that could be used to develop and build database systems and computer applications [9].

It was in the 1990s when the UML has first appeared and become the industry standard for software modeling and design, as well as the modeling of other processes in the scientific and business worlds [12].

4.1. What is Unified Modeling Language (UML)?

Unified Modeling Language (UML) is defined as a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created, by the Object Management Group (OMG). It was first added to the list of OMG adopted technologies in 1997, and has since become the industry standard for modeling

software-intensive systems [10]. It includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems.

The Unified Modeling Language (UML) is a tool for specifying and visualizing software systems. It includes standardized diagram types that describe and visually map a computer application or a database systems design and structure. The use of UML as a tool for defining the structure of a system is a very useful way to manage large, complex systems. Having a clearly visible structure makes it easy to introduce new people to an existing project [12].

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. It offers a standard way to visualize a system's architectural blueprints, including elements such as [10]: activities, actors, business processes, database schemas, (logical) components, programming language statements and reusable software components.

The Unified Modeling Language (UML) combines techniques and processes from the data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies [10].

4.2. UML Diagrams for Database Systems

The Unified Modeling Language (UML) standard diagrams are: use case diagram, class diagram, sequence diagram, state chart diagram, activity diagram, component diagram, and deployment diagram [9].

4.2.1 Use Case diagram

The functionality provided by a database system or a computer application can be illustrated by a use case diagram. Its main purpose is to visualize the functional requirements of a system, including the relationship of "actors" (human beings who will interact with the system) to essential processes, as well as the relationships among different use cases. It is a list of steps, typically defining interactions between a role (known in UML as an "actor") and a system, to achieve a goal.

ISSN: 2278-6244
Impact Factor: 6.111

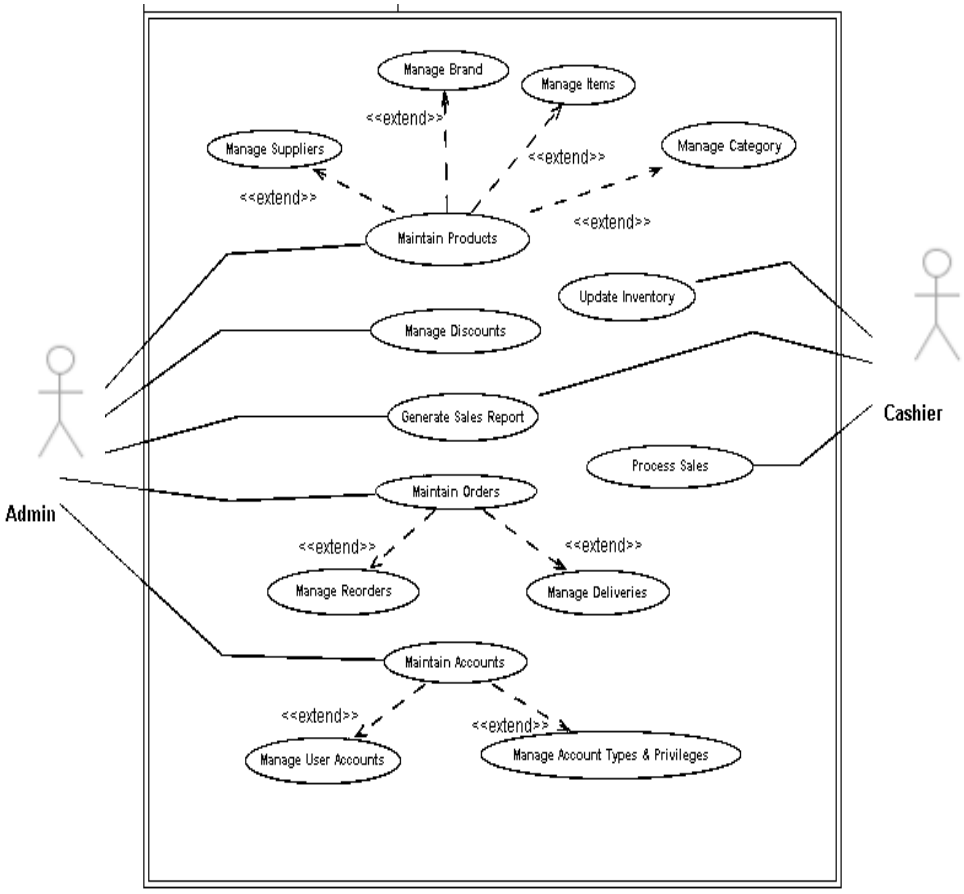

Fig 1. Sample Use Case Diagram of a Point of Sale

A use case diagram is typically used to communicate the high-level functions of the system and the system's scope. Fig 1 shows a use case diagram of a simple point of sale (POS). This diagram can easily tell the functions that a point of sale provides.

4.2.2 Class diagram

The static structures of a computer application or a database station are shown in a class diagram. It also shows how the different entities (people, things, and data) relate to each other. It can be used to display logical classes, and implementation classes. It is the main building block of object oriented modeling. It is a type of static structure diagram in UML that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes [15].

4.2.3 Sequence diagram

The sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. It shows the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects [9].

ISSN: 2278-6244
Impact Factor: 6.111

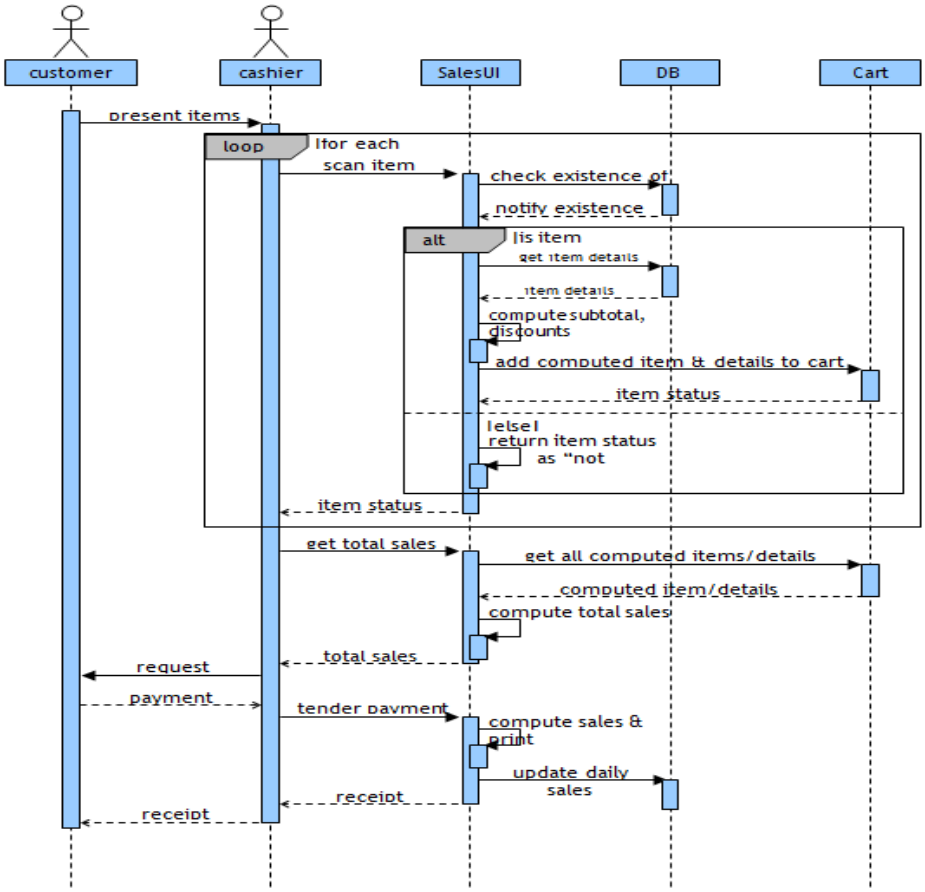

Fig 2. Sample Sequence Diagram of a Point of Sale (Processing Sales)

A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent. Fig.2 illustrates an example of a sequence diagram of processing sales in a Point of Sale (POS) system.

4.2.4 StateChart diagram

StateCharts [13] is a modelling technique supporting a hierarchy for finite state machines. The model hierarchy consists of super-states and substates. Sub-states of a super-state can be parallel (for AND-super-states) or exclusive (for OR-superstates). The technique supports also timers via timeout transitions.

4.2.5 Activity diagram

The procedural flow of control between two or more class objects while processing an activity can be shown with an activity diagrams. It can be used to model higher-level business process at the business unit level, or to model low-level internal class actions [9].

ISSN: 2278-6244
Impact Factor: 6.111

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. Fig. 3 illustrates an example of an activity diagram of processing order in a Point of Sale (POS) system.

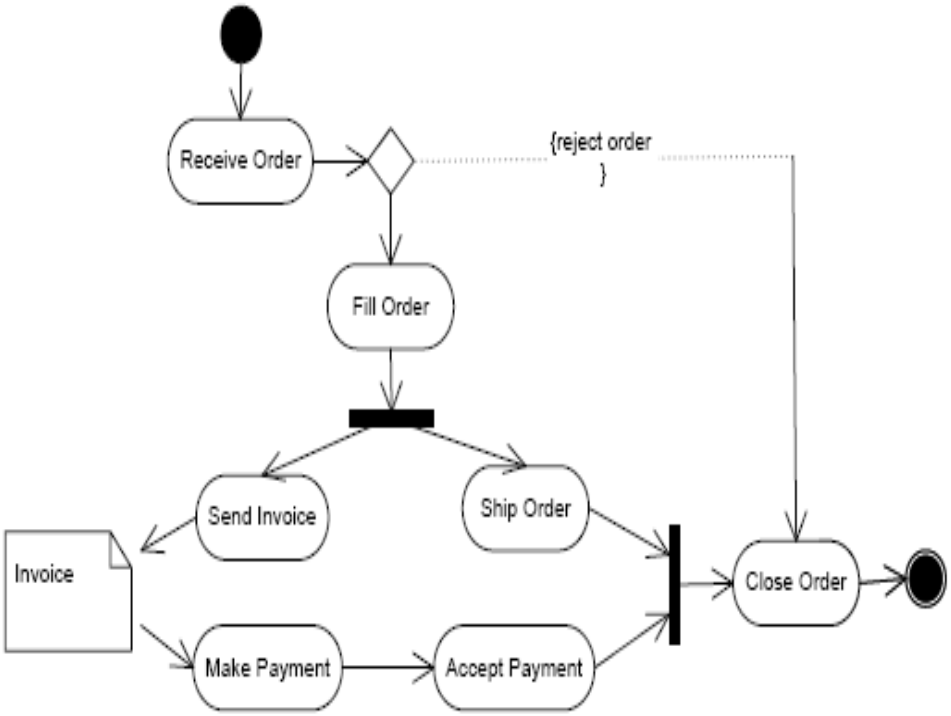

Fig. 3 Sample Activity Diagram of a Point of Sale (Processing Order)

4.2.6 Component diagram

A component diagram provides a physical view of the system. It depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. Its purpose is to show the dependencies that the software has on the other software components (e.g., software libraries) in the system.

4.2.7 Deployment diagram

The deployment diagram shows how a system will be physically deployed in the hardware environment. Its purpose is to show where the different components of the system will physically run and how they will communicate with each other. Since the diagram models the physical runtime, a system's production staff will make considerable use of this diagram. It models the physical deployment of artifacts on nodes. The notation in a deployment

diagram includes the notation elements used in a component diagram, with a couple of additions, including the concept of a node. A node represents either a physical machine or a virtual machine node (e.g., a mainframe node) [9].

4.2.8 Package diagram

Package diagrams are used to reflect the organization of packages and their elements. When used to represent class elements, package diagrams provide a visualization of the namespaces. The most common use for package diagrams is to organize use case diagrams and class diagrams, although the use of package diagrams is not limited to these UML elements [12].

5. CONCLUSIONS

As a future work, we firstly plan to extend our work to deal with the <<extends>> relationship of use cases. Secondly, we would like to incorporate our methodology in a CASE tool in order to automate it. And finally, we would like to integrate our technique in the whole development process.

The Unified Modeling Language (UML) is a tool for specifying software systems that include standardized diagrams to define, illustrate and visually map or model a software system's design and structure. UML diagrams include the use case diagram, class diagram, sequence diagram, state chart diagram, activity diagram, component diagram, and deployment diagram.

This paper outlined the use of Unified Modeling Language (UML) as a standard notation of real-world objects in developing object-oriented design methodology for computer applications and database systems.

REFERENCES

[1] OMG. UML 2.0 Superstructure, 2003. Object Management Group.

[2] A. Cockburn. Writing Effective Use Cases. Addison Wesley, 2001.

[3] G. Schneider, J. Winters. Applying use case: A practical guide. Addison Wesley, 2001.

[4] Xiaoshan Li, Zhiming Liu, and Jifeng He. Formal and use-case driven requirement analysis in UML. Research Report 230, UNU/IIST, Macau, March 2001.

[5] Wuwei Shen, Mohsen Guizani, Zijiang Yang, Kevin J. Compton, and James Huggins. Execution of A Requirement Model in Software Development. In Proceedings of

the ISCA 13th International Conference on Intelligent and Adaptive Systems and Software Engineering, pages 203-208. ISCA, July 2004.

[6] Somé, S. Petri Nets Based Formalization of Texual Use Cases. Tech, Report in SITE, TR2007-11, Uni. Of Ottawa, 2007.

[7] Ren SB, Chen SQ, Y SY. Use Case Description Formalization and Analysis Based on Event Sequence. Computer Engineering and Application, 2004(23):12-14.

[8] Chen X, Li XD. Design calculus based approach to modeling use case. Journal of Software, 2008, 19(10):2539-2549.

[9] Bell D, UML basics: An introduction to the Unified Modeling Language, IBM Developer Works, http://www.ibm.com/developerworks/rational/library/769.html, (2003) June 15.

[10] http://en.wikipedia.org/wiki/Unified_Modeling_Language.

[11] http://en.wikipedia.org/wiki/Database_system.

[12] http://www.sparxsystems.com.au/platforms/uml.html.

[13] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231 – 274, 1987.

[14] http://martinfowler.com/.

[15] http://en.wikipedia.org/wiki/Class_diagram.

[16] Ponniah P, "Database Design and Development: An Essential Guide for IT Professionals", ISBN 0-471-21877-4 Copyright © 2003 by John Wiley and Sons, Inc, (2003).