Unified Modeling Language (UML) for Database Systems and Computer Applications

Sunguk Lee*

Research Institute of Industrial Science and Technology
Pohang, Korea
sunguk@rist.re.kr
**Correspondent Author: Sunguk Lee* (sunguk@rist.re.kr)*

Abstract

This paper 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 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, statechart diagram, activity diagram, component diagram, and deployment diagram. The integration of these diagrams to different software processes have been discussed.

Keywords: *android, android platform, SQLite database*

1. Introduction

The first step in developing an object-oriented design methodology for computer applications as well as for database systems is the use of UML (Unified Modeling Language) 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.

UML is considered an industry standard modeling language with a rich graphical notation, and comprehensive set of diagrams and elements. It is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development.

This paper 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 have been discussed in this paper. The different UML diagrams for database application are also illustrated that could be integrated to different software development processes.

The rest of this paper is organized as follows: Section 2 explains the Database systems overview; Section 3 outlines the concepts of Unified Modeling Language (UML) and the integration of its diagrams for Database systems; and the concluding remarks in Section 4.

2. 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 [7]. 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 [3].

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.

Both a database and its DBMS 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 [3].

3. 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 [1].

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 [4].

3.1 What is UML?

Unified Modeling Language or 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 [2]. It includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems.

The 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 [4].

The 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 [2]:

- activities
- actors
- business processes
- database schemas
- (logical) components
- programming language statements
- reusable software components.

The 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 [2].

3.2 UML Diagrams for DB Systems

The UML standard diagrams are: use case diagram, class diagram, sequence diagram, statechart diagram, activity diagram, component diagram, and deployment diagram [1].

3.2.1 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.

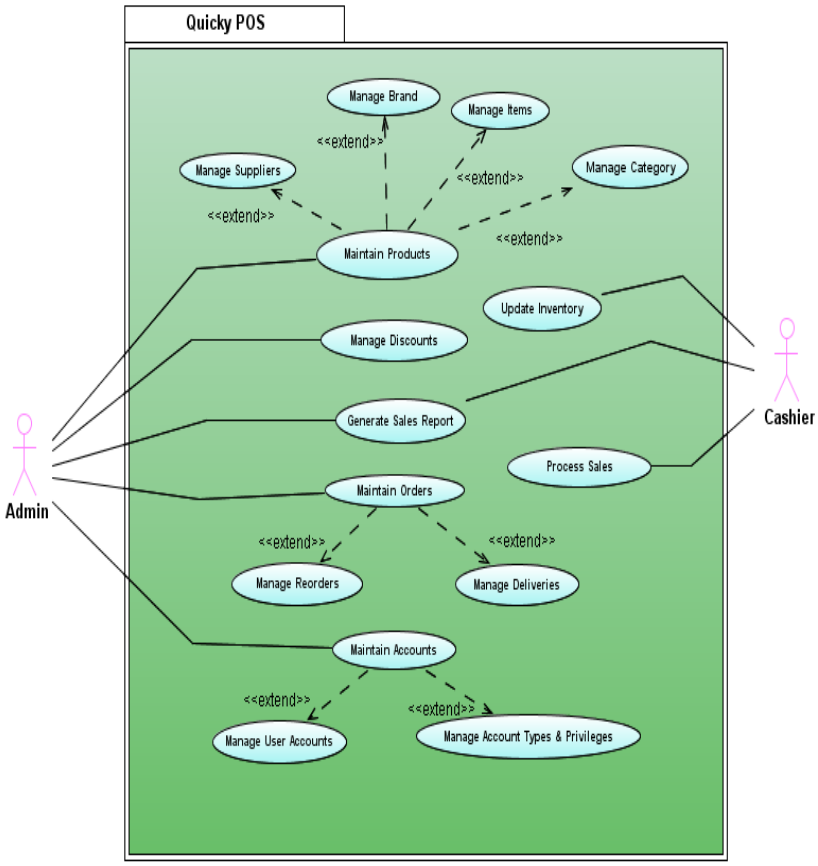

Figure 1. Sample Use Case Diagram of a Point of Sale

A use-case diagram is typically used to communicate the high-level functions of the system and the system's scope. Figure 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.

3.2.2 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 [6].

3.2.3 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 [1].

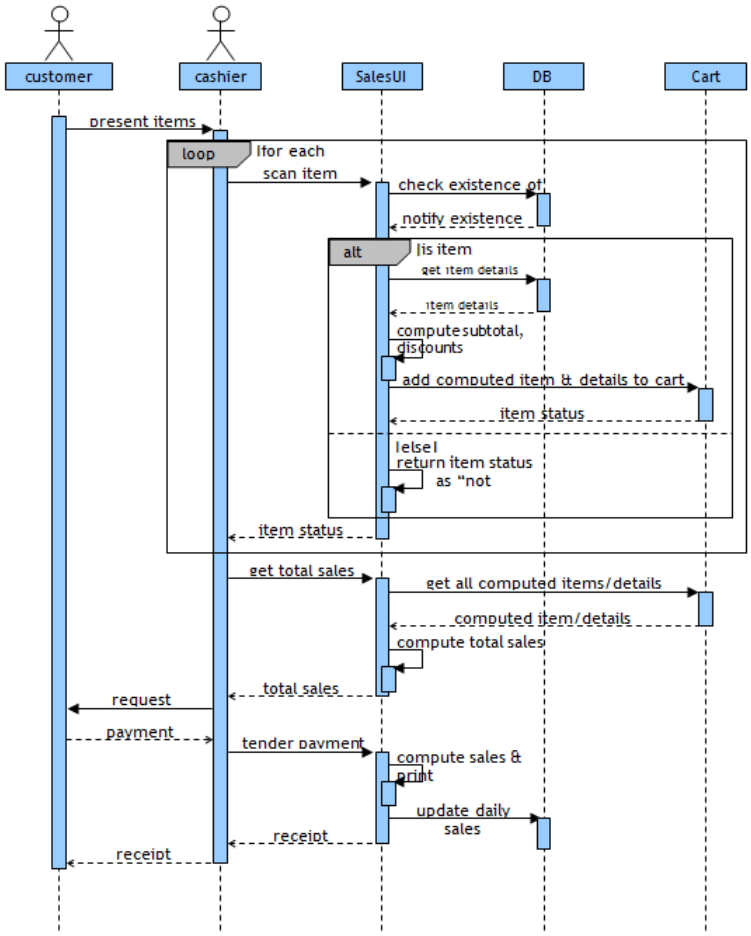

Figure 2. Sample Sequence Diagram of a Point of Sale (Processing Sales)

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. Figure 2 illustrates an example of a sequence diagram of processing sales in a Point of Sale (POS) system.

3.2.4 Statechart Diagram: The statechart diagram is used to visualize and model the different states that a class can be in and how that class transitions from state to state. It describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. Its purpose is to model life time of an object from creation to termination. It has five basic elements: the initial starting point, which is drawn using a solid circle; a transition between states, which is drawn using a line with an open arrowhead; a state, which is drawn using a rectangle with rounded corners; a decision point, which is drawn as an open circle; and one or more termination points, which are drawn using a circle with a solid circle inside it.

3.2.5 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 [1].

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. Figure 3 illustrates an example of an activity diagram of processing order in a Point of Sale (POS) system.

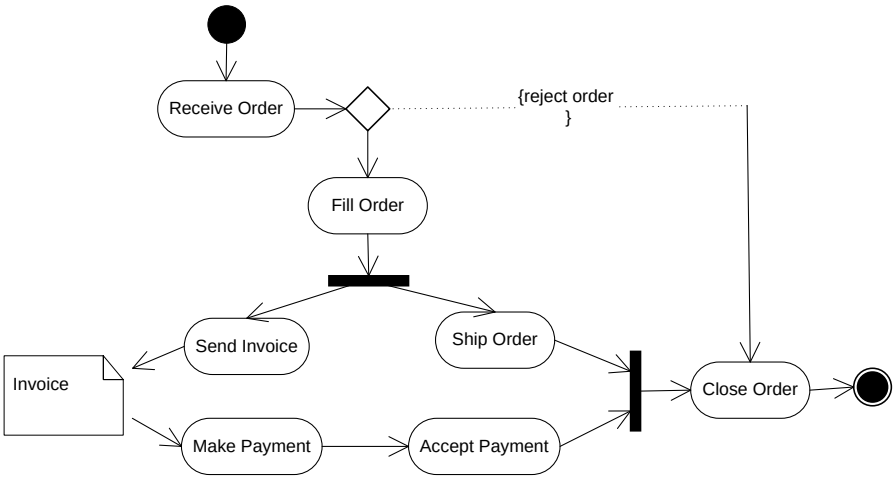

Figure 3. Sample Activity Diagram of a Point of Sale (Processing Order)

3.2.6 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.

3.2.7 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) [1].

3.2.8 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 [4].

4. Conclusions

The 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, statechart 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] 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.

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

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

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

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

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

[7] 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)**.