Advanced Science and Technology Letters
Vol.28 (CIA 2013), pp.88-93
http://dx.doi.org/10.14257/astl.2013.28.17

An Integrated Diagrammatic Model vs. Multiplicity of Models in UML

Sabah Al-Fedaghi

Computer Engineering Department, Kuwait University, Kuwait
sabah.alfedaghi@ku.edu.kw

Abstract. An information system model represents a business establishment and reflects the reality of the organization and its operations. One begins to build such a model by drawing a conceptual picture of the establishment as part of its real-world domain. Object-oriented methods and languages (e.g., UML) are typically used to describe the system at this level. This paper contrasts this approach with a method that provides a single, integrated graphic model that incorporates function, structure, and behavior. This model is demonstrated through a sample that previously applied UML techniques to an object-oriented database analysis workflow.

Keywords: Information system model; UML; use case diagram; multiplicity of models

1 Introduction

An information system model represents a business establishment and reflects the reality of the organization and its operations. Object-oriented methods and languages (e.g., UML) are typically used to describe the system at this level. Researchers have proposed extending object-oriented software design languages such as UML so they can be applied at the conceptual level (e.g., [1]). The remarkable aspect of this approach is the multiplicity of models in UML, a known problem [3] that contrasts with simply providing a single, integrated diagrammatic model that incorporates function, structure, and behavior [4].

These UML representations are completely heterogeneous in form, with several different conceptual bases. Use cases are narratives, use case diagrams are sketchy, sequence diagrams reflect the nature of exchange/communication, and so on. The purpose of this heterogeneity is to provide a wide range of options for expression, depending on the situation. This need for multiple models goes beyond only UML diagrams, e.g., Shoval and Kabeli [5] propose a merger of the data flow diagram, the entity relationship diagram, and object-oriented constructs.

This approach of a multiplicity of models in UML has caused several problems. For example, a problem has arisen in achieving consistency between a UML use case model and its corresponding set of textual descriptions, which are the written explanations of the use case relationships contained in the UML model [6].

This paper proposes that it is time to adopt a new paradigm:

ISSN: 2287-1233 ASTL

Either attempting to go back to the approach of a single, integrated graphic model that incorporates function, structure, and behavior, or attempting to develop this new model that might provide an underlying integrated model for UML representation.

The paper suggests such an integrating model without, at this stage, trying to make a case for either of the two proposals above. To demonstrate the viability of this new conceptual model, called the flowthing model (FM), the paper focuses on a representative example of UML diagrams described by Wang [7] and contrasts it with the methodology proposed in this paper with the aim of comparing the two representations side by side.

2 UML-based Framework

Wang [7] proposes "a framework that applies UML components to ORDBs [Object Oriented Databases] development workflows." It is based on a United Process [8], where UML use cases are utilized to capture the functional requirements and involves an iterative process that "takes a set of use cases from requirements all the way through implementation, test and deployment" [7].

To illustrate how to apply UML techniques to an ORDBs analysis workflow, Wang [7] begins with development of use case narratives. As shown in Figure 1, a "Place order" example is developed based on use case narratives. A use case diagram (Figure 2) is also presented to visually illustrate system functionality. Both sequence (Figure 3) and class diagrams can be derived from this UML use case.

Use case: "Place Order"
Actor: Customer
Precondition "Customer has invoked "Browse catalog"
Flow of Events

1. Customer adds additional item to shopping cart
2. Customer selects "Review order"
3. Customer receives calculation of total cost, tax, and shipping charges
4. Customer selects "Place order"
5. Customer clicks "Submit Order" to update the system
6. Customer 's receipt is created

Post Condition: Customer leaves the screen or Web site

Fig. 1. Use Case Narratives: Place Order

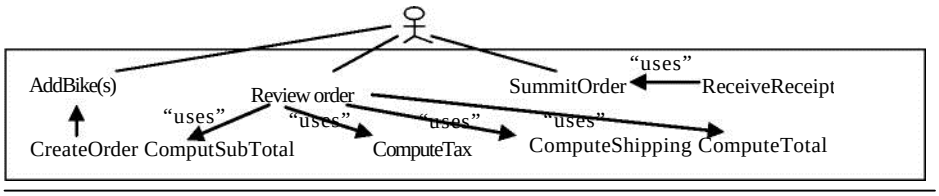

Fig. 2. Use Case Diagram: Place Order

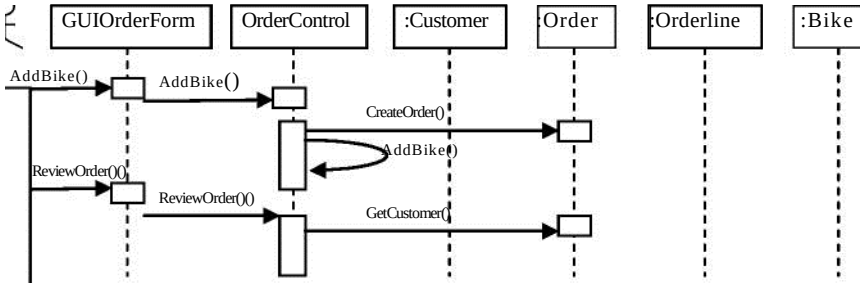

Fig. 3. Partial view of Sequence Diagram: Place Order

Class names can be developed from a list of nouns identified, including actors from use case diagrams that are class names, by checking which use case has what classes and verifying "which classes are responsible for creating, retrieving, updating and deleting methods" [7].

The general claim of this paper is that solving problems related to development of representations of IS requirements ought to be based on a new paradigm based on the notion of flow. The feasibility of the new paradigm is demonstrated by recasting Wang's [7] sample system of use cases in terms of a new proposed methodology.

The next section reviews the FM model as explained in several publications [9–12]. The example constructed at the end of the section is a new contribution.

3 Flowthing Model

The Flowthing Model (FM) was inspired by the many types of flows that exist in diverse fields, such as, for example, supply chain flow, money flow, and data flow in communication models. This model is a diagrammatic schema that uses "flow things" (referred to as *flowthings*) to represent a range of items that can be, for example, data, information, or signals. FM represents processes using "flow systems" (referred to as *flowsystems*) that include six stages: Arrive, Accepted, Processed (changed), Released, Created, and Transferred.

These stages are mutually exclusive, i.e., a flowthing in the *Process* stage cannot be in the *Created* stage or the *Released* stage at the same time. An additional stage of Storage can also be added to any FM model to represent the storage of flowthings; however, storage is not a generic stage, because there can be stored processed flowthings, stored created flowthings, and so on. Hereafter, a *thing* means a flowthing.

Figure 4 shows the structure of a flowsystem. A *flowthing* is a thing that has the capability of being created, released, transferred, arrived, accepted, or processed while flowing within and between systems. A flowsystem depicts the internal flows of a system with the six stages and transactions among them.

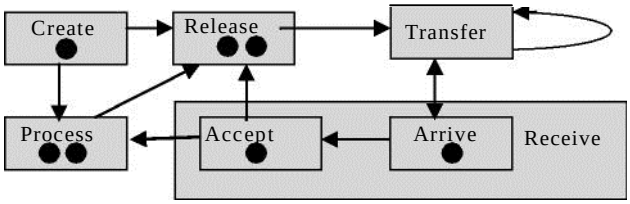

Fig. 4. Flowsystem. The dark dots denote things at different stages of the flowsystem. The figure can be considered like net marking (instantaneous location of all tokens in the net; Petri net terminology).

FM uses the following basic notions:
Flowthings*:* A thing that is created, released, transferred, arrived, accepted, and processed while flowing within and between 'entities' called *spheres*.
Spheres and subspheres: These are the environments (contexts) of the flowthing.
Triggering: Triggering is an activation (denoted in FM diagrams by a dashed arrow) of one flow by another.

4 FM-based conceptual description

To show the FM representation without loss of generality, Wang [7]'s "Place order" example is diagrammed as shown in Figure 5 with an attempt to preserve the semantics of the example as understood in the original paper.

In Figure 5, a catalog is retrieved (circle 1) by the system and released (2) to flow to the customer (3), where it is received and processed (4), e.g., selection and addition of elements ordered from the menu. This processing triggers (5) the creation (6) of ordered elements, an order that flows to the company system (7), where it is processed (8) to trigger (9) a Review of items (10). The Review flows (11) to the customer, where it is processed (12) to trigger (13) the creation of not approved (14) or approved (15). Not approved triggers the process to start again by sending another catalog to the customer (16). An approval note flows to the company (17) where it is processed (18) to trigger the creation of a cost quote (19) that flows to the customer (20). Upon receiving the cost quote (21), the customer may approve (22) or not approve it (23). In case of approval, an indication of that flows to the company (24) where it is processed and triggers creation of the final order (25). This order flows to the Shipping department (26) where it is processed (27) to trigger:
- Retrieval of the ordered items from stock (28) and sending to the customer (29)
- The creation (31) and sending of a receipt to the customer (31).

As shown in Figure 5, the FM representation reflects repeated application of the five generic operations: transfer, receive, process, release, and create. It is characterized by a continuity of events that ensures coherence and completeness in the semantics involved. In Figure 5, several additions are inserted to fill conceptual gaps in the original use case description. For example, approvals have been added in the case of Review and Cost invoice. It is clear that delivering goods is performed by a different entity in the company; thus a Shipping Department has been added.

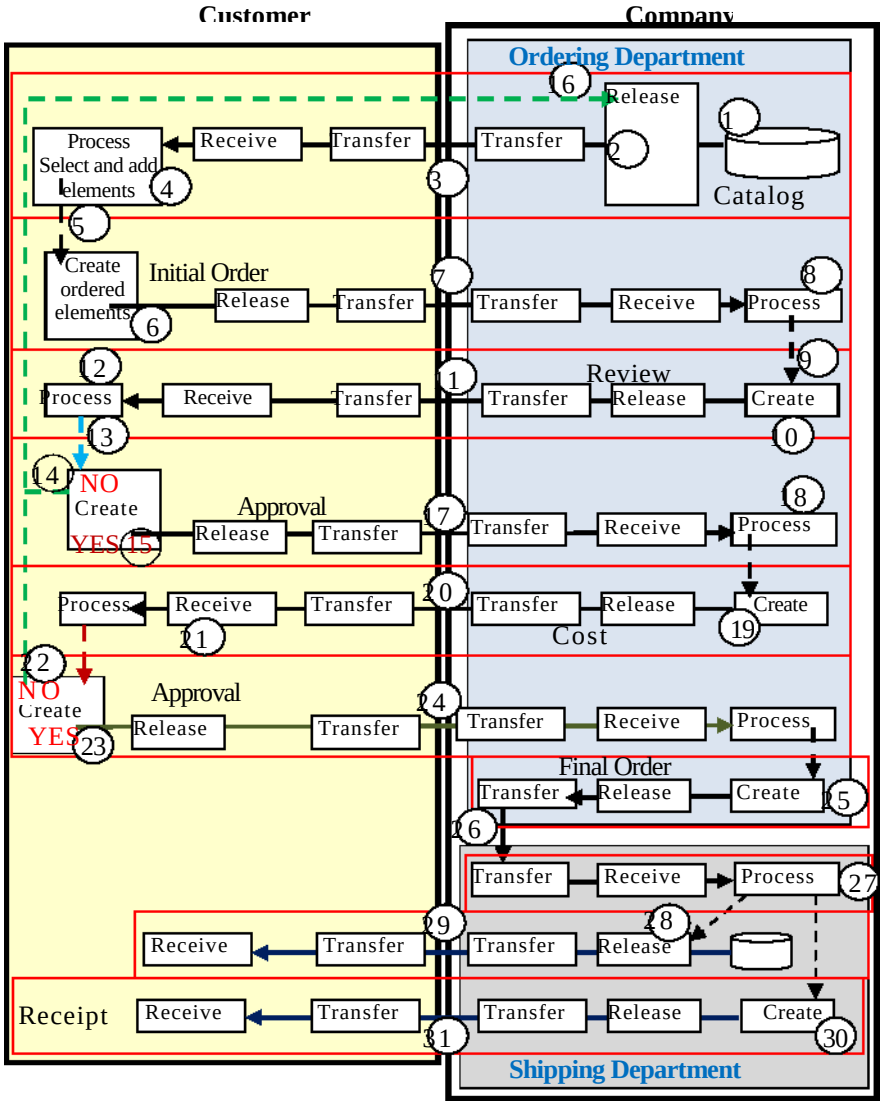

Fig. 5. FM representation of Place Order

References

1. J Evermann and Y Wand. Towards ontologically based semantics for UML constructs. In: H Kunii, S Jajodia, and A. Solvberg (eds.), Proceedings of the 20th International Conference on Conceptual Modeling, Yokohama, Japan. 2001. http://www.mcs.vuw.ac.nz/~jevermann/EvermannWandER01.pdf
2. C Kobryn. Will UML 2.0 be agile or awkward? Communications of the ACM, Vol. 45, No. 1, pp.107–110. 2002.
3. D Dori. Why significant UML change is unlikely. Communications of the ACM, Vol. 45, No. 11, pp.82–85. 2002.
4. D Dori. Object-Process Methodology applied to modeling credit card transactions. Journal of Database Management, Vol. 12, No. 1, pp.4–14. 2001.
5. P Shoval and J Kabeli. FOOM: Functional- and Object-Oriented analysis & design of information systems: An integrated methodology. Journal of Database Management, Vol. 12, No. 1, pp.15–25. 2001.
6. V Hoffmann, H Lichter, A Nyßen, and A Walter. Towards the integration of UML- and textual use case modeling. J. Object Tech., Vol. 8, No. 3, May-June 2009.
7. [7] Ming Wang. Using UML for object-relational database systems development: A framework. Issues in Information Systems. Vol. 9, No. 2, pp. 238-543. 2008.
8. P Kruchten. The Rational Unified Process: An Introduction (3rd Ed.). 2004. ISBN 0-321-19770-4.
9. S Al-Fedaghi. Flow-based enterprise process. International Journal of Database Theory and Application, Vol. 6, No. 3, pp. 59-70. 2013.
10. S Al-Fedaghi. A method for modeling and facilitating understanding of user requirements in software development. Journal of Next Generation Information Technology, Vol. 4, No. 3, pp. 30-38. 2013.
11. S Al-Fedaghi. Toward flow-based semantics of activities. International Journal of Software Engineering and Its Applications, Vol. 7, No. 2, pp. 171-182. 2013.
12. S Al-Fedaghi. Conceptualization of business processes. 2009 IEEE Asia-Pacific Services Computing Conference (IEEE APSCC 2009), Biopolis, Singapore, Dec 7-11, 2009.

