Role and Request Based Conceptual Modeling – A Methodology and a CASE Tool

Yair Wand, Carson Woo, and Ohad Wand

Sauder School of Business,
The University of British Columbia, Vancouver, Canada
yair.wand@ubc.ca, carson.woo@ubc.ca, ohad.wand@gmail.com

Abstract. This paper contains a brief description of the R^2M (Role and Request Modeling) method and its supporting visual modeling CASE (Computer Assisted Software Engineering) tool. R^2M is a modeling method for creating Conceptual Models of work systems using a combination of ontological and object-oriented concepts. Ontological principles serve to define the meaning of modeling constructs in terms of domain semantics, and to derive rules guiding the modeling process. The CASE tool is a graphical software tool that supports the creation of models according to the R^2M method. Guided by the principles of R^2M, the tool helps assure the semantic integrity of models, and enables management of complex models via decomposition (i.e. more details at decreasing abstraction levels). The tool can help ensure consistency between different modelers and completeness of models.

Keywords: conceptual modeling, business analysis, CASE tool

A Conceptual Model - in the context of information systems analysis - can be described in simple terms as a formal representation of the organizational domain for which an information system is being developed. The importance of Conceptual Modeling as a tool in systems analysis and requirements determination has been widely recognized. Four purposes have been identified for conceptual models: supporting an analyst's understanding of an application domain, communicating with stakeholders, communicating with implementers, and documenting system rationale for future needs.

The object-oriented approach is arguably the most common software design and implementation paradigm now in use. This is evidenced by the popularity of UML (the *Unified Modeling Language*)[1]. However, the use of object-concepts in Conceptual Modeling has not been widely adapted. A main reason is that there are no generally accepted semantics of these concepts as conceptual modeling elements.

To address the issue of assigning domain semantics to object-oriented constructs we have used ontological concepts and principles [2,3]. The ontological concepts can be used to define the meaning of object-oriented concepts and the principles can serve to suggest rules to guide ontologically-sound modeling. Specifically, we propose that objects represent active things (actors) in an application domain and object classes represent organizational roles. The dynamics of a modeled domain can then be represented in terms of state changes of individual actors and of interactions between

actors that assume certain roles. This view led us to develop a set of modeling rules which address two issues: first - the mapping of domain phenomena to a model; and second - semantic integrity constraints that can be applied to constructed models. Based on these rules, we developed a modeling procedure that assures the ontological validity of constructed models. The procedure can identify situations where the modeler needs to clarify domain aspects with stakeholders.

The modeling approach – termed *Role and Request Modeling* (R^2M) – has been implemented in a CASE Tool. This tool embeds data structures that reflect the fundamental ontological concepts and principles (that in turn guide the semantic integrity rules). As well, the tool provides checks for the adherence of constructed models to the modeling rules.

R^2M is graphic notation-independent. However, the user interface of the R^2M software (shown in Figure 1 below) uses an intuitive representation of the modeled domain. The information about the model appears in several visible panes:

- The *Role Explorer* (left side) displays all roles in the model for easy navigation.
- The *Modeling Canvas* (main portion) in which the model is created by the user.
- The *Property Details* (lower portion) where details about the role currently selected in the *Modeling Canvas* are displayed and manipulated.
- An additional pane showing errors in the model (the *Semantic Errors* pane, described below) can be visible or hidden.

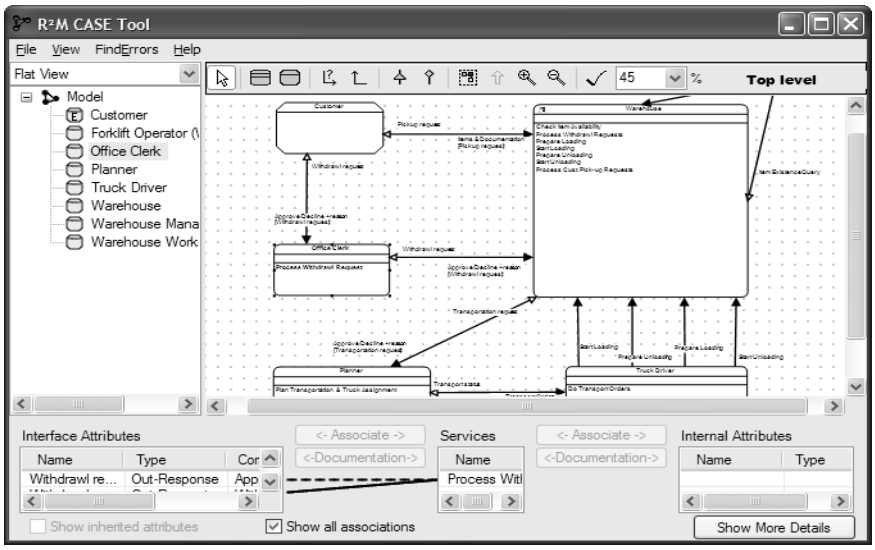

Fig. 1. The *R^2M CASE Tool* user interface

To enable construction of models of a complex environment R^2M supports a well-formalized method of and rules for decomposition. The rules assure that models at any level will be ontologically and syntactically consistent with higher and lower level models of the same domain.

As an example of decomposition using R^{2}M, Figure 2 shows part of a domain model within the *Modeling Canvas*. The view shown is the *top level* model – i.e. the highest level of abstraction. At this level in the example, both the "Customer" and the "Office Clerk" roles communicate with the "Warehouse" role.

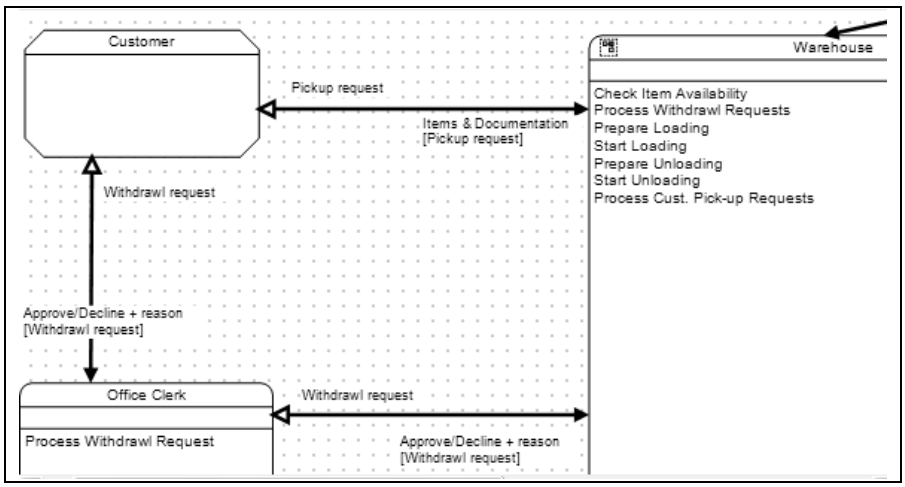

Fig. 2. Sample model – top level

Figure 3 shows part of the decomposition model of the "Warehouse" role of the same domain. This view shows roles and communications that are internal to the "Warehouse" (i.e. the "Warehouse Manager" and "Warehouse Worker" roles, and the communications between them) as well as the communications between these and the roles that – at the *top level* – appear to communicate with the "Warehouse" role. As can be seen in this example, the "Office Clerk" communicates with "Warehouse Manager" and the "Customer" communicates with the "Warehouse Worker". R^{2}M supports decomposition to any level and ensures consistency between the levels.

Figure 4 presents an example of the *Semantic Errors* pane (lower part of the figure). This pane lists all errors present in the model and can be hidden or visible as required. When visible, items on the *Modeling Canvas* relating to the listed errors are highlighted (in red). Clicking on an error brings the item in error into view on the *Modeling Canvas*. As the model is corrected, the *Semantic Errors* pane is automatically updated to reflect the current error state of the model.

We have experimented with the R^{2}M method and tool both in teaching situations and in practical (and realistic size) cases. The results have shown that the use of the method led to consistency of models across modelers. Furthermore, semantic errors identified by the tool were often an indication for the analysts to seek additional information about the modeled domain, thus leading to more complete and accurate models.

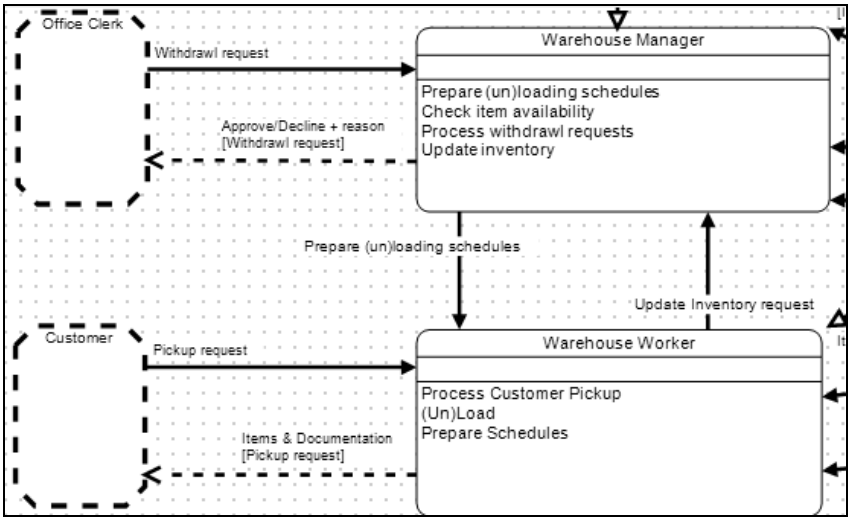

Fig. 3. Sample model – decomposition of the "Warehouse" role

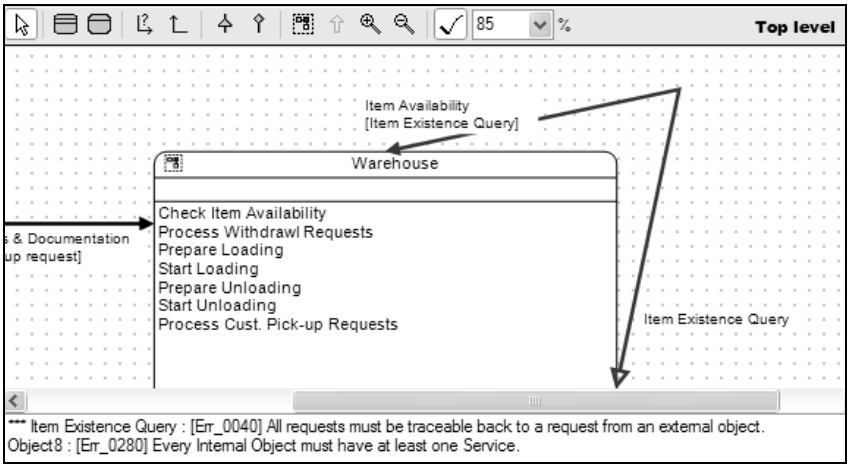

Fig. 4. Semantic checking

References

1. Object Management Group, OMG Unified Modeling Language Specification, Version 2.1.2, http://www.omg.org/spec/UML/2.1.2/
2. Wand, Y., Woo, C.: Object-Oriented Analysis - Is It Really that Simple? In: *Proceedings of the Third Workshop on Information Technologies and Systems* (WITS'93), Orlando, Florida (1993), pp. 186--195.
3. Wand, Y., Woo, C., Hui, S.: Developing Business Models to Support Information System Evolution. In: *Proceedings of the Ninth Workshop on Information Technologies and Systems* (WITS'99), Charlotte, North Carolina (1999), pp. 137—142.