Business Process Modeling and Simulation with UML

Part I: Defining a UML Transaction Model That Maps to RUP Business Models

by Pan-Wei Ng
Software Engineering Specialist
Rational Software Singapore

Business modeling is an important activity in software development: For software to be useful, it must directly support the business. Once a sound business model is developed, it is possible to explore different business improvement options such as eliminating redundant tasks, and automating repetitive and mundane tasks or those prone to errors. Although the Rational Unified Process® (RUP®) provides a systematic approach to visually representing a business model, it does not provide a way to derive simulation models that support quantitative comparisons between various improvement options. In this first part of a two-part series, we will define a Unified Modeling Language (UML) profile for a transaction model that fills this gap, and show how it can be mapped from business models in the RUP. Next month, Part II will discuss how to execute the transaction model in a simulation, using the Rational Rose® Process Simulator Add-In.

The business modeling discipline in the Rational Unified Process (RUP) provides a way for project teams to:

- Understand the structure and dynamics of the organization in which a system is to be deployed (the target organization).
- Understand current problems in the target organization and identify potential areas for improvement.
- Ensure that customers, end users, and developers have a common understanding of the target organization.
Derive the system requirements needed to support the target organization.

To achieve these goals, the business modeling discipline describes how to develop a vision of the new target organization, and, based on this vision, define the processes, roles, and responsibilities and their associated organizations within the business models. A key strength of the RUP approach is that business modeling follows the same principles as object modeling for software development; the use of common principles makes the resulting business model easy for stakeholders to understand.

What Is Use-Case-Driven Simulation?

Although business models in UML have a standardized notation that facilitates discussion of existing problems in target organizations and how these problems can be solved, the models lack the quantitative dimension that permit stakeholders to make objective comparisons between the current process and proposed changes, or between different proposals. The RUP counters this inadequacy by recommending Activity Based Costing (ABC) to provide the needed mathematical support. The problem with mathematical computation is its unwieldiness when the model grows in size and complexity, a by-product of the stochastic nature of business processes (i.e., these processes are subject to random variables and guesswork).

Simulation, specifically the technique of Discrete Event Simulation (DEVS), poses an attractive alternative because it is less mathematically demanding in its specification, but it still offers sufficient analytical power to compare different improvement options, such as adding more computing resources or modifying current work processes. In addition, DEVS follows modeling principles similar to those in UML, such as abstraction, modularity, and hierarchy. Instead of manipulating mathematical formulae, one only needs to manipulate simulation entities (i.e., classes) and their collaborations.

One inadequacy of DEVS modeling is the lack of a "use-case-driven" approach that emphasizes business goals and elaborates how collaborating participants contribute to these goals. Instead, DEVS emphasizes organizational structure and individual participant behavior, which consequently produces a larger-than-required model, because even those activities not related to the business goal under investigation are included.

Consequently, it would be highly advantageous to have a UML-based business-process simulation methodology that is "use-case-driven." The benefits would include:

- Common notation based on UML.
- Analytical capability of DEVS.
- Common modeling principles as per the RUP.

In the remainder of this article, we will derive a UML profile for business-
process simulation by walking through a simplified example of an outpatient hospital. First, we'll describe the hospital work processes through RUP business models and then map these models to a transaction model that can be simulated.

**Deriving a Transaction Model**

The RUP specifies two models for business modeling:

- **Business Use-Case Model.** Describes business processes from an external perspective.
- **Business Object Model.** Describes business processes from an internal perspective.

We will refer to these as the *RUP business models*. From these two models, we will derive a *transaction model* to describe the business process from a simulation perspective. We call it a *transaction model* rather than a *simulation model* because it addresses transactions that are handled by the business; simulation is a technique for analyzing these transactions. The transaction model operates at a different level of abstraction from the business object model and has more details to support quantitative analysis.

Transactions are traced from use cases and their realizations, and they are the basis for simulation. The relationship between the RUP business models and the transaction model are depicted in Figure 1.

![Figure 1: Relationship Between RUP Business Models and Transaction Models](image)

**RUP Business Models**

The RUP Business models are comprised of several stereotyped UML elements, namely:

- **Business Actor.** Represents a role played in relation to the business by someone or something in the business environment.
- **Business Use Case.** Defines a set of business use-case instances; each instance is a sequence of actions a business performs that yields an observable result of value to a particular business actor.
- **Business Worker.** Represents an abstraction of a human or
system that acts within the system. A business worker interacts with other business workers and manipulates business entities while participating in business use-case realizations.

- **Business Entity.** Represents passive objects that business workers access, inspect, manipulate, produce, and so on.

- **Business Use-Case Realization.** Describes how a particular business use case is realized within the business object model, in terms of collaborating objects (instances of business workers and business entities).

Our example, an outpatient hospital, illustrates the concepts prescribed in the business modeling stereotypes. The business use-case model comprises a set of business actors and business use cases. In our simplified example, we have a single business actor, the *Patient*, and a single business use case, *Visit Doctor*. See Figure 2.

![Figure 2: Visit Doctor Business Use-Case Model](image)

Within the hospital, we find several business workers, namely:

- **Nurses** who make simple clinical measurements before the patient sees the doctor.
- **Doctors** who make the actual diagnosis and prescribe drugs.
- **Dispensary staff** who collect payments and dispense drugs.

Figure 3 is a sequence diagram of a business use-case realization for a typical visit. The use case begins when the patient walks into the hospital. The nurse retrieves the patient's medical record and takes clinical measurements from the patient, such as temperature and blood pressure. Then the doctor starts to make a diagnosis and may prescribe suitable drugs. The patient subsequently goes to the dispensary to collect the prescribed drugs and pay for them after completing a subsidy form.
Figure 3.: Visit Doctor Business Use-Case Realization: Sequence Diagram

The Sequence Diagram in Figure 3 does not show any business entities. This is because the purpose of simulation is largely to compare transactional durations and human and system resource consumption, which are primarily dependent on active elements (business actors and business workers) in the business process. Business entities are more useful in enterprise modeling that captures business domain details and are often a basis for database design.

The Transaction Model

The active elements in the business models and their collaborations are used to derive elements in the transaction model. The transaction model comprises a set of transactions and participating transaction elements as depicted in Figure 4. Each transaction comprises a number of transaction flows and is visually represented in interaction diagrams, as exemplified in Figures 7 and 8, which show sequence and collaboration diagrams, respectively. Each step in a transaction flow is represented as a message.

Transaction model elements participate in transactions and are further categorized as generators and servers (commonly used terms in simulation literature). Generators are able to initiate transactions, whereas servers are service providers. Each time a service is rendered, a resource unit is held for a period of time and then released after the service is completed. For example, once a doctor has completed a diagnosis and has prescribed drugs, then he or she is available for the next patient. Generators are traced from business actors, whereas servers are traced from business workers. Transactions are traced from business use cases.

Each transaction model element has a set of operations, which are categorized as initiations, services, and pauses. As the name implies, initiations indicate the initiation of a transaction flow. Each step in the
transaction flow may require a transaction model element to provide a service, or it may pause for a while (to allow time to think).

![Figure 4: Transaction Model Meta-Model](image)

The Visit Doctor business use case in the outpatient hospital example is realized by the Visit Doctor transaction depicted in Figure 5.

![Figure 5: Traceability from Business Use Case to Transaction](image)

The traceability from the business model elements to the transaction model elements is depicted in Figure 6. In this example, we have a one-to-one relationship.
The transaction flow is derived from the Visit Doctor business use-case realization and is depicted in Figure 7. Note that, aside from a change in the stereotypes of the participating objects, there is no difference between Figure 7 and Figure 3. This highlights the fact that our transaction modeling approach follows the same paradigm as that of the RUP business modeling approach.

Collaboration Diagrams offer an alternate representation of the transaction flow. The Sequence Diagram in Figure 7 is represented as a Collaboration Diagram in Figure 8. This representation is more familiar to DEVS practitioners who frequently deal with queuing models, in which our icon stereotypes are founded. The icon representation for a server looks like a queue; the one for a generator looks like a signal generator. Links in the Collaboration Diagram appear as possible paths for information flow.
The view of participating classes (VOPC) for the Visit Doctor transaction depicted in Figure 9 is more interesting, because it is here that the differences between business modeling and transaction modeling start to appear. The simulation of the transaction model requires the specification of numerical quantities, such as timing and resource usage mechanisms.

Parameters are required to augment the classes in Figure 9 with numerical quantities required for simulation, including:

- **Server Multiplicity.** Each server has a multiplicity that indicates how many instances it can possibly have. In our example there are two doctors, three nurses, and one dispensary.

- **<<initiates>> Operation Flow Multiplicity.** Each generator <> will generate a number of transaction flows indicated by its multiplicity. In our example 100 typical visits are generated. The notation "Transaction.Transaction Flow" is used to indicate the specific transaction flow initiated.

- **<<initiates>> Operation Inter-Arrival Time.** The transaction flows have a specified inter-arrival rate. In our example the time between visits is fifteen minutes.
● **<<pauses>> Operation Duration.** There is a finite waiting time. In our example it takes two minutes to complete the subsidy form.

● **<<queues>> Operation Duration.** Each service takes a finite time period to complete. In our example a nurse takes one minute to retrieve the patient record, and a doctor takes fifteen minutes per diagnosis.

● **<<queues>> Operation Resource Usage.** Each time a service is processed, it holds a number of resources. In our example, each time a patient is serviced, one resource unit (i.e., a doctor, nurse, or dispensary staff person) is held.

These parameters are represented as tag values in the transaction model classes, attributes, and operations.

Now that we have defined our transaction model, we are ready to execute it in a simulation. In Part II, which will appear in the May issue of *The Rational Edge*, we will show how to perform a simulation with the Rational Rose Process Simulator Add-In.

---

*For more information on the products or services discussed in this article, please click [here](#) and follow the instructions provided. Thank you!*

---

Copyright [Rational Software 2002](#) | [Privacy/Legal Information](#)