UML-Based Modeling for What-If Analysis

Matteo Golfarelli and Stefano Rizzi

DEIS, University of Bologna, Viale Risorgimento 2, Bologna, Italy
golfare@csr.unibo.it, stefano.rizzi@unibo.it

Abstract. In order to be able to evaluate beforehand the impact of a strategical or tactical move, decision makers need reliable previsional systems. What-if analysis satisfies this need by enabling users to simulate and inspect the behavior of a complex system under some given hypotheses. A crucial issue in the design of what-if applications in the context of business intelligence is to find an adequate formalism to conceptually express the underlying simulation model. In this experience paper we report on how this can be accomplished by extending UML 2 with a set of stereotypes. Our proposal is centered on the use of activity diagrams enriched with object flows, aimed at expressing functional, dynamic, and static aspects in an integrated fashion. The paper is completed by examples taken from a real case study in the commercial area.

Keywords: What-if analysis, data warehouse, UML, simulation.

1 Introduction

An increasing number of enterprises feel the need for obtaining relevant information about their future business, aimed at planning optimal strategies to reach their goals. In particular, in order to be able to evaluate beforehand the impact of a strategical or tactical move, decision makers need reliable previsional systems. Data warehouses, that have been playing a lead role within business intelligence (BI) platforms in supporting the decision process over the last decade, are aimed at enabling analysis of past data, and are not capable of giving anticipations of future trends. That's where what-if analysis comes into play.

What-if analysis can be described as a data-intensive simulation whose goal is to inspect the behavior of a complex system (i.e., the enterprise business or a part of it) under some given hypotheses (called *scenarios*). More pragmatically, what-if analysis measures how changes in a set of independent variables impact on a set of dependent variables with reference to a *simulation model* offering a simplified representation of the business, designed to display significant features of the business and tuned according to the historical enterprise data [7].

Example 1. A simple example of what-if query in the marketing domain is: *How would my profits change if I run a* 3×2 *(pay 2 – take 3) promotion for one week on all audio products on sale?* Answering this query requires a simulation model to be built. This model, that must be capable of expressing the complex

I.-Y. Song, J. Eder, and T.M. Nguyen (Eds.): DaWaK 2008, LNCS 5182, pp. 1–12, 2008.

relationships between the business variables that determine the impact of promotions on product sales, is then run against the historical sale data in order to determine a reliable forecast for future sales.

Surprisingly, though a few commercial tools are already capable of performing forecasting and what-if analysis, very few attempts have been made so far outside the simulation community to address methodological and modeling issues in this field (e.g., see [6]). On the other hand, facing a what-if project without the support of a methodology is very time-consuming, and does not adequately protect the designer and her customers against the risk of failure.

From this point of view, a crucial issue is to find an adequate formalism to conceptually express the simulation model. Such formalism, by providing a set of diagrams that can be discussed and agreed upon with the users, could facilitate the transition from the requirements informally expressed by users to their implementation on the chosen platform. Besides, as stated by [3], it could positively affect the accuracy in formulating the simulation problem and help the designer to detect errors as early as possible in the life-cycle of the project. Unfortunately, no suggestion to this end is given in the literature, and commercial tools do not offer any general modeling support.

In this paper we show how, in the light of our experience with real case studies, an effective conceptual description of the simulation model for a what-if application in the context of BI can be accomplished by extending UML 2 with a set of stereotypes. As concerns static aspects we adopt as a reference the multidimensional model, used to describe both the source historical data and the prediction; the YAM^2 [1] UML extension for modeling multidimensional cubes is adopted to this end. From the functional and dynamic point of view, our proposal is centered on the use of activity diagrams enriched with object flows. In particular, while the control flow allows sequential, concurrent, and alternative computations to be effectively represented, the object flow is used to describe how business variables and cubes are transformed during simulation.

The paper is structured as follows. In Section 2 we survey the literature on modeling and design for what-if analysis. In Section 3 we outline the methodological framework that provides the context for our proposal. Section 4 discusses how we employed UML 2 for simulation modeling; it proposes some examples taken from a case study concerning branch profitability and explains how we built the simulation model. Finally, Section 7 draws the conclusion.

2 Related Literature

In the literature about simulation, different formalisms for describing simulation models are used, ranging from colored Petri nets [10] to event graphs [8] and flow charts [2]. The common trait of these formalisms is that they mainly represent the dynamic aspects of the simulation, almost completely neglecting the functional (how are data transformed during the simulation?) and static (what data are involved and how are they structured?) aspects that are so relevant for data-intensive simulations like those at the core of what-if analysis in BI.

A few related works can be found in the database literature. [5] uses constraint formulae to create hypothetical scenarios for what-if analysis, while [9] explores the relationships between what-if analysis and multidimensional modeling. [4] presents the SESAME system for formulating and efficiently evaluating what-if queries on data warehouses; here, scenarios are defined as ordered sets of hypothetical modifications on multidimensional data. In all cases, no emphasis is placed on modeling and design issues.

In the context of data warehousing, there are relevant similarities between simulation modeling for what-if analysis and the modeling of ETL (Extraction, Transformation and Loading) applications; in fact, both ETL and what-if analysis can be seen as a combination of elemental processes each transforming an input data flow into an output. [15] proposes an ad hoc graphical formalism for conceptual modeling of ETL processes. While such proposal is not based on any standard formalisms, other proposals extend UML by explicitly modeling the typical ETL mechanisms. For example, [14] represents ETL processes by a class diagram where each operation (e.g., conversion, log, loader, merge) is modeled as a stereotyped class. All these proposals cannot be considered as feasible alternatives to ours, since the expressiveness they introduce is specifically oriented to ETL modeling. On the other hand, they strengthen our claim that extending UML is a promising direction for achieving a better support of the design activities in the area of BI.

Finally, we mention two relevant approaches for UML-based multidimensional modeling [1,12]. Both define a UML profile for multidimensional modeling based on a set of specific stereotypes, and represent cubes at three different abstraction levels. On the other hand, [1] is preferred to [12] for the purpose of this work since it allows for easily modeling different aggregation levels over the base cube, which is essential in simulation modeling for what-if analysis.

3 Methodological Framework

A what-if application is centered on a *simulation model*. The simulation model establishes a set of complex relationships between some *business variables* corresponding to significant entities in to the business domain (e.g., products, branches, customers, costs, revenues, etc.). In order to simplify the specification of the simulation model and encourage its understanding by users, we functionally decompose it into *scenarios*, each describing one or more alternative ways to construct a *prediction* of interest for the user. The prediction typically takes the form of a multidimensional cube, meant as a set of cells of a given type, whose dimensions and measures correspond to business variables, to be interactively explored by the user by means of any OLAP front-end. A scenario is characterized by a subset of business variables, called *source variables*, and by a set of additional parameters, called *scenario parameters*, that the user has to value in order to execute the model and obtain the prediction. While business variables are related to the business domain, scenario parameters convey information technically related to the simulation, such as the type of regression

adopted for forecasting and the number of past years to be considered for regression. Distinguishing source variables among business variables is important since it enables the user to understand which are the "levers" that she can independently adjust to drive the simulation; also non-source business variables are involved in scenarios, where they are used to store simulation results. Each scenario may give rise to different simulations, one for each assignment of values to the source variables and of the scenario parameters.

Example 2. In the promotion domain of Example 1, the source variables for the scenario are the type of promotion, its duration, and the product category it is applied to; possible scenario parameters are the forecasting algorithm and its tuning parameters. The specific simulation expressed by the what-if query reported in the text is determined by giving values "3 × 2", "one week" and "audio", respectively, to the three source variables. The prediction is a Sales cube with dimensions week and product and measures revenue, cost and profit, which the user could effectively analyze by means of any OLAP front-end.

Designing a what-if application requires a methodological framework; the one we consider, presented by [6], relies on the seven phases sketched in the following:

1. *Goal analysis* aims at determining which business phenomena are to be simulated, and how they will be characterized. The goals are expressed by (i) identifying the set of business variables the user wants to monitor and their granularity; and (ii) outlining the relevant scenarios in terms of source variables the user wants to control.
2. *Business modeling* builds a simplified model of the application domain in order to help the designer understand the business phenomenon, enable her to refine scenarios, and give her some preliminary indications about which aspects can be neglected or simplified for simulation.
3. *Data source analysis* aims at understanding what information is available to drive the simulation, how it is structured and how it has been physically deployed, with particular regard to the cube(s) that store historical data.
4. *Multidimensional modeling* structurally describes the prediction by taking into account the static part of the business model produced at phase 2 and respecting the requirements expressed at phase 1. Very often, the structure of the prediction is a coarse-grain view of the historical cube(s).
5. *Simulation modeling* defines, based on the business model, the simulation model allowing the prediction to be constructed, for each given scenario, from the source data available.
6. *Data design and implementation*, during which the cube type of the prediction and the simulation model are implemented on the chosen platform, to create a prototype for testing.
7. *Validation* evaluates, together with the users, how faithful the simulation model is to the real business model and how reliable the prediction is. If the approximation introduced by the simulation model is considered to be unacceptable, phases 4 to 7 are iterated to produce a new prototype.

The five analysis/modeling phases (1 to 5) require a supporting formalism. Standard UML can be used for phases 1 (use case diagrams), 2 (a class diagram coupled with activity and state diagrams) and 3 (class and deployment diagrams), while any formalism for conceptual modeling of multidimensional databases can be effectively adopted for phase 4 (e.g., [1] or [12]). On the other hand, finding in the literature a suitable formalism to give broad conceptual support to phase 5 is much harder.

4 A Wish List

Phase 5, simulation modeling, is the core phase of design. In the light of our experience with real case studies of what-if analysis in the BI context, we enunciate a wish list for a conceptual formalism to support it:

♯1 The formalism should be capable of coherently expressing the simulation model according to three perspectives: functional, that describes how business variables are transformed and derived from each other during simulation; dynamic, required to define the simulation workflow in terms of sequential, concurrent and alternative tasks; static, to explicitly represent how business variables are aggregated during simulation.
♯2 It should provide constructs for expressing the specific concepts of what-if analysis, such as business variables, scenario parameters, predictions, etc.
♯3 It should support hierarchical decomposition, in order to provide multiple views of the simulation model at different levels of abstraction.
♯4 It should be extensible so that the designer can effectively model the peculiarities of the specific application domain she is describing.
♯5 It should be easy to understand, to encourage the dialogue between the designer and the final users.
♯6 It should rely on some standard notation to minimize the learning effort.

UML perfectly fits requirements ♯4 and ♯6, and requirement ♯5 to some extent. In particular, it is well known that the stereotyping mechanism allows UML to be easily extended. As to requirements ♯1 and ♯3, the UML diagrams that best achieve integration of functional, dynamic and static aspects while allowing hierarchical decomposition are *activity diagrams*. Within UML 2, activity diagrams take a new semantics inspired by Petri nets, which makes them more flexible and precise than in UML 1 [13]. Their most relevant features for the purpose of this work are summarized below:

- An *activity* is a graph of *activity nodes* (that can be action, control or object nodes) connected by *activity edges* (either control flows or object flows).
- An *action node* represents a task within an activity; it can be decorated by the rake symbol to denote that the action is described by a more detailed activity diagram.
- A *control node* manages the control flow within an activity; control nodes for modeling decision points, fork and synchronization points are provided.

- An *object node* denotes that one or more instances of a given class are available within an activity, possibly in a given state. Input and output objects for activities are denoted by overlapping them to the activity borderline. A datastore stereotype can be used to represent an object node that stores non-transient information.
- *Control flows* connect action nodes and control nodes; they are used to denote the flow of control within the activity.
- *Object flows* connect action nodes to object nodes and vice versa; they are used to denote that objects are produced or consumed by tasks.
- *Selection* and *transformation* behaviors can be applied to object nodes and flows to express selection and projection queries on object flows.

Though activity diagrams are a nice starting point for simulation modeling since they already support advanced functional and dynamic modeling, some extensions are required in order to attain the desired expressiveness as suggested by requirement ♯2. In particular, it is necessary to define an extension allowing basic multidimensional modeling of objects in order to express how simulation activities are performed on data at different levels of aggregation.

5 Expressing Simulation Models in UML 2

In our proposal, the core of simulation modeling is a set of UML 2 diagrams organized as follows:

1. A use case diagram that reports a what-if analysis use case including one or more scenario use cases.
2. One or more class diagrams that statically represent scenarios and multidimensional cubes. A scenario is a class whose attributes are scenario parameters; it is related via an aggregation to the business variables that act as source variables for the scenario. Cubes are represented in terms of their dimensions, levels and measures.
3. An activity diagram (called *scenario diagram*) for each scenario use case. Each scenario diagram is hierarchically exploded into activity diagrams at increasing level of detail. All activity diagrams represent, as object nodes, the business variables, the scenario parameters and the cubes that are produced and consumed by tasks.

Representation of cubes is supported by YAM^2 [1], a UML extension for conceptual multidimensional modeling. YAM^2 models concepts at three different detail levels: *upper*, *intermediate*, and *lower*. At the upper level, *stars* are described in terms of *facts* and *dimensions*. At the intermediate level, a fact is exploded into *cells* at different aggregation granularities, and the aggregation *levels* for each dimension are shown. Finally, at the lower level, *measures* of cells and *descriptors* of levels are represented.

In our approach, the intermediate and lower levels are considered. The intermediate level is used to model, through the cell stereotype, the aggregation

granularities at which data are processed by activities, and to show the combinations of dimension levels (level stereotype) that define those granularities. The lower level allows single measures of cells to be described as attributes of cells, and their type to be separately modeled through the KindOfMeasure stereotype.

In order to effectively use YAM^2 for simulation modeling, three additional stereotypes are introduced for modeling, respectively, scenarios, business variables and scenario parameters:

name:	scenario
base class:	class
description:	classes of this stereotype represent scenarios
constraints:	a scenario class is an aggregation of business variable classes (that represent its source variables)

name:	business variable
base class:	class
description:	classes of this stereotype represent business variables
tagged values:	· isNumerical (type Boolean, indicates whether the business variable can be used as a measure) · isDiscrete (type Boolean, indicates whether the business variable can be used as a dimensions)

name:	scenario parameter
icon:	SP
base class:	attribute
description:	attributes of this stereotype represent parameters that model user settings concerning scenarios
constraints:	a scenario parameter attribute belongs to a scenario class

Besides these basic stereotypes, and considering the characteristics of each specific application domain, the designer may define some ad hoc activity and dependency stereotypes to model, respectively, recurrent types of activities and specific roles of object flows within such activities. In Section 6, in the context of the case study, we will see some examples of ad hoc stereotyping.

6 A Case Study

Orogel S.p.A. is a large Italian company in the area of deep-frozen food. It has a number of branches scattered on the national territory, each typically entrusted with selling and/or distribution of products. Its data warehouse includes a number of data marts, one of which dedicated to commercial analysis and centered on a Sales cube with dimensions Month, Product, Customer, and Branch.

The managers of Orogel are willing to carry out an in-depth analysis on the *profitability* (i.e., the net revenue) of branches. More precisely, they wish to know if, and to what extent, it is convenient for a given branch to invest on either selling or distribution, with particular regard to the possibility of taking new

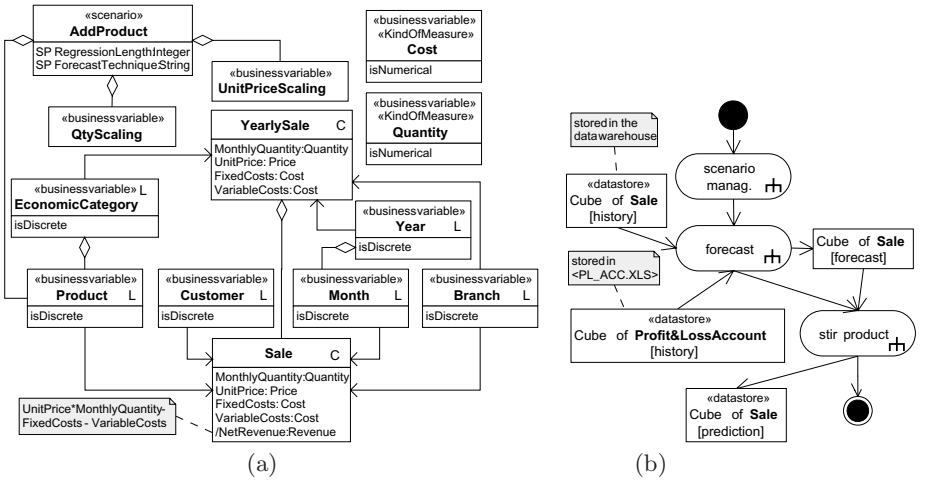

Fig. 1. Class diagram (a) and scenario diagram (b)

customers or new products. Thus, the four scenarios chosen for prototyping are: (i) analyze profitability during next 12 months in case one or more new products were taken/dropped by a branch; and (ii) analyze profitability during next 12 months in case one or more new customers were taken/dropped by a branch. Decision makers ask for analyzing profitability at different levels of detail; the finest granularity required for the prediction is the same of the Sales cube.

The main issue in simulation modeling is to achieve a good compromise between reliability and complexity. To this end, in constructing the simulation model we adopted a two-step approach that consists in first forecasting past data, then "stirring" the forecasted data according to the events (new product or new customer) expressed by the scenarios. We mainly adopted statistical techniques for both the forecasting and the stirring steps; in particular, linear regression is employed to forecast unit prices, quantities and costs starting from historical data taken from the commercial data mart and from the profit and loss account during a past period taken as a reference. Based on the decision makers' experience, and aimed at avoiding irrelevant statistical fluctuations while capturing significant trends, we adopted different granularities for forecasting the different measures of the prediction cube [6].

6.1 Representing the Simulation Model

The four what-if use cases (one for each scenario) are part of a use case diagram – not reported here for space reasons– that, as suggested by [11], expresses how the different organization roles take advantage of BI in the context of sales analysis.

As to static aspects, the class diagram shown in Figure 1.a gives a (partial) specification of the multidimensional structure of the cubes involved. Sale is the base cell; its measures are MonthlyQuantity, UnitPrice, FixedCosts, VariableCosts and NetRevenue (the latter is derived from the others), while the dimensions

are Product, Customer, Month and Branch. Aggregations within dimensions represent roll-up hierarchies (e.g., products roll up to economic categories). Both dimension levels and measure types are further stereotyped as business variables. YearlySale is a cell derived from Sale by aggregation on EconomicCategory, Year and Branch. Finally, the top section of the diagram statically represents the AddProduct scenario in terms of its parameters (RegressionLength and ForecastTechnique) and source variables (UnitPriceScaling, QtyScaling and Product).

As to dynamic aspects, the add product use case is expanded in the scenario diagram reported in Figure 1.b, that provides a high-level overview of the whole simulation process. The rake symbol denotes the activities that will be further detailed in subdiagrams. Object nodes whose instances are cubes of cells of class <Sale> are named as Cube of <Sale>. The state in the object node is used to express the current state of the objects being processed (e.g., [forecast]).

The activity nodes of the context diagram are exploded into a set of hierarchical activity diagrams whose level of abstraction may be pushed down to describing tasks that can be regarded as atomic. Some of them are reported here in a simplified form and briefly discussed below:

- Activity forecast (Figure 2.a) is aimed at extrapolating sale data for the next twelve months. This is done separately for the single measures. In particular, forecasting general costs requires to extrapolate the future fixed and variable costs from the past profit and loss accounts, and scale variable costs based on the forecasted quantities. Input and output objects for forecast are emphasized by placing them on the activity borderline. The transformation stereotype expresses which measure(s) are selected from an object flow.
- The quantity forecast for next year (Figure 2.b) can be done, depending on the value taken by parameter ForecastTechnique, either by judgement (the total quantities for next year are directly specified by the user) or by regression (based on the total quantities sold during the last RegressionLength years); in both cases, the total quantity is then apportioned on the single months, products and customers proportionally to the quantities sold during the last 12 months. The selection stereotype expresses which objects are selected from an object flow. Note also the use of dependency stereotypes to specify the roles taken by objects flows within standard activities such as regression and apportion; for instance, with reference to a regression activity, length and history denote the input flows that provide, respectively, the temporal interval and the historical data to be used as a basis for regression.
- Finally, Figure 2.c explodes the stir product activity, that simulates the effects of adding a new product of a given type by reproducing the sales events related to a representative product of the same type in the same branch. First, the past sales of the reference product are scaled according to the user-specified percentages stored in QtyScaling and UnitPriceScaling. Then, cannibalization[1] on forecasted sales for the other products is simulated by

[1] *Cannibalization* is the process by which a new product gains sales by diverting sales from existing products, which may deeply impact the overall profitability.

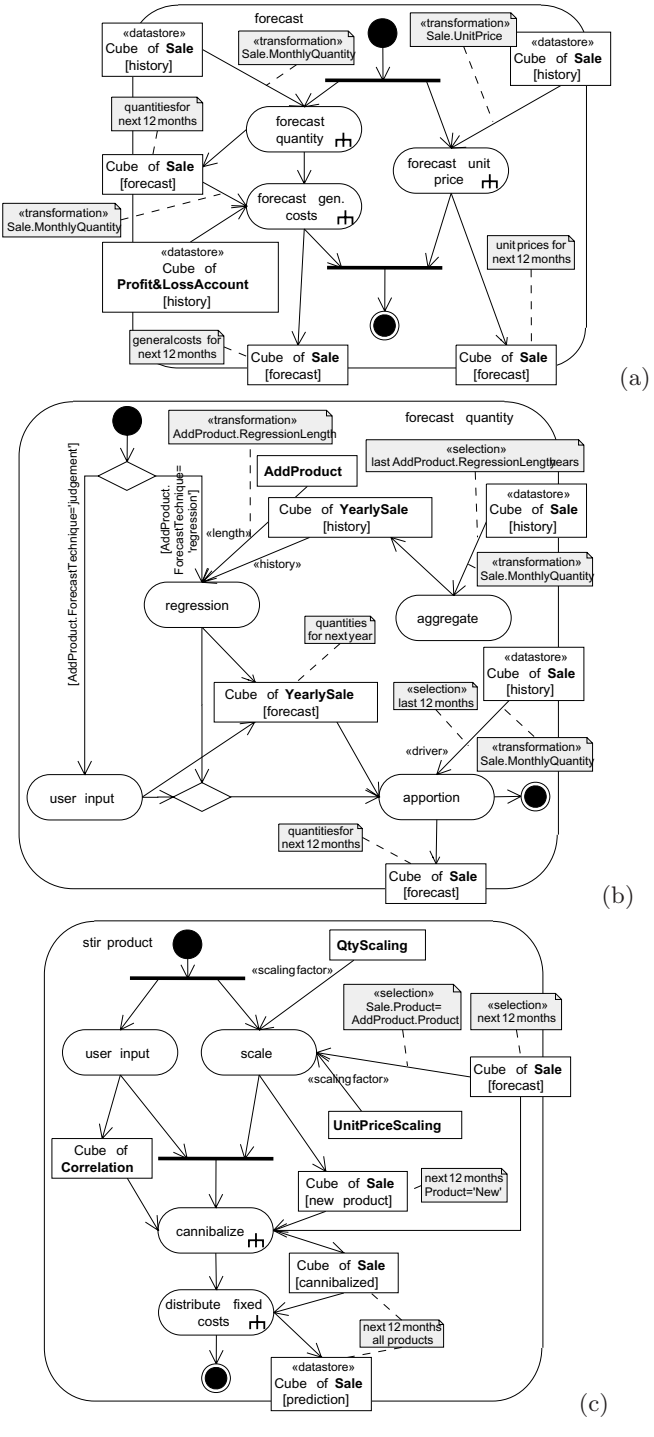

Fig. 2. Activity diagrams for forecast (a), forecast quantity (b) and stir product (c)

applying a product correlation matrix built by judgmental techniques. Finally, fixed costs are properly redistributed on the single forecasted sales.

6.2 Building the Simulation Model

In this section we give an overview of the approach we pursued to build the UML simulation model for Orogel. The starting points are the use case diagram, the business model and the multidimensional model obtained, respectively, from phases 1, 2 and 4 of the methodology outlined in Section 3. For simplicity, we assume that the multidimensional model is already coded in YAM^2.

1. The class diagram is created first, by extending the multidimensional model that describes the prediction with the statical specification of scenarios, source variables and scenario parameters.
2. For each scenario reported in the use case diagram, a high-level scenario diagram is created. This diagram should show the macro-phases of simulation, the main data sources and the prediction. The object nodes should be named consistently with the classes diagram.
3. Each activity in each scenario diagram is iteratively exploded and refined into additional activity diagrams. As new, more detailed activities emerge, business variables and scenario parameters from the class diagram may be included in activity diagrams. Relevant aggregation levels for processing business variables within activities may be identified, in which case they are increasingly reported on the class diagram. Refinement goes on until the activities are found that are elemental enough to be understood by an executive designer/programmer.

7 Conclusion

To sum up, our approach to simulation modeling fulfills the wish list proposed in Section 4 as follows: (♯1) Static, functional and dynamic aspects are modeled in an integrated fashion by combining use case, class and activity diagrams; (♯2) Specific constructs of what-if analysis are modeled through the UML stereotyping mechanism; (♯3) Multiple levels of abstraction are provided by both activity diagrams, through hierarchical decomposition, and class diagrams, through the three detail levels provided by YAM^2; (♯4) Extensibility is provided by applying the stereotyping mechanism; (♯5) Though completely understanding the implications of a UML diagram is not always easy for business users, the precision and methodological rigor encouraged by UML let them more fruitfully interact with designers, thus allowing solutions to simulation problems to emerge easily and clearly during analysis even when, in the beginning, users have little or no idea about how the basic laws that rule their business world should be coded; (♯6) UML is a standard. In the practice, the approach proved successful in making the design process fast, well-structured and transparent.

A critical evaluation of the proposed approach against its possible alternatives unveils that the decisive factor is the choice of adopting UML as the modeling language rather than devising an ad hoc formalism. Indeed, adopting UML poses

some constraints in the syntax of diagrams (for instance, the difficulty of directly showing on activity diagrams the aggregation level at which cells are processed); on the other hand it brings some undoubted advantages to the designer, namely, the fact of relying on a standard and widespread formalism. Besides, using hierarchical decomposition of activity diagrams to break down the complexity of modeling increases the scalability of the approach.

We remark that the proposed formalism is oriented to support simulation modeling at the *conceptual* level, which in our opinion will play a crucial role in reducing the overall effort for design and in simplifying its reuse and maintenance. Devising a formalism capable of adequately expressing the simulation model at the *logical* level, so that it can be directly translated into an implementation, is a subject for our future work.

References

1. Abelló, A., Samos, J., Saltor, F.: YAM^2: a multidimensional conceptual model extending UML. Information Systems 31(6), 541–567 (2006)
2. Atkinson, W.D., Shorrocks, B.: Competition on a divided and ephemeral resource: A simulation model. Journal of Animal Ecology 50, 461–471 (1981)
3. Balci, O.: Principles and techniques of simulation validation, verification, and testing. In: Proc. Winter Simulation Conf., Arlington, VA, pp. 147–154 (1995)
4. Balmin, A., Papadimitriou, T., Papakonstantinou, Y.: Hypothetical queries in an OLAP environment. In: Proc. VLDB, Cairo, Egypt, pp. 220–231 (2000)
5. Dang, L., Embury, S.M.: What-if analysis with constraint databases. In: Proc. British National Conf. on Databases, Edinburgh, Scotland, pp. 307–320 (2004)
6. Golfarelli, M., Rizzi, S., Proli, A.: Designing what-if analysis: Towards a methodology. In: Proc. DOLAP, pp. 51–58 (2006)
7. Kellner, M., Madachy, R., Raffo, D.: Software process simulation modeling: Why? what? how? Journal of Systems and Software 46(2-3), 91–105 (1999)
8. Kotz, D., Toh, S.B., Radhakrishnan, S.: A detailed simulation model of the HP 97560 disk drive. Technical report, Dartmouth College, Hanover, NH, USA (1994)
9. Koutsoukis, N.-S., Mitra, G., Lucas, C.: Adapting on-line analytical processing for decision modelling: the interaction of information and decision technologies. Decision Support Systems 26(1), 1–30 (1999)
10. Lee, C., Huang, H.C., Liu, B., Xu, Z.: Development of timed colour Petri net simulation models for air cargo terminal operations. Computers and Industrial Engineering 51(1), 102–110 (2006)
11. List, B., Schiefer, J., Tjoa, A.M.: Process-oriented requirement analysis supporting the data warehouse design process - a use case driven approach. In: Ibrahim, M., Küng, J., Revell, N. (eds.) DEXA 2000. LNCS, vol. 1873, pp. 593–603. Springer, Heidelberg (2000)
12. Luján-Mora, S., Trujillo, J., Song, I.-Y.: A UML profile for multidimensional modeling in data warehouses. Data & Knowledge Engineering 59(3), 725–769 (2006)
13. OMG. UML: Superstructure, version 2.0 (2007), `www.omg.org`
14. Trujillo, J., Luján-Mora, S.: A UML based approach for modelling ETL processes in data warehouses. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 307–320. Springer, Heidelberg (2003)
15. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. In: Proc. DOLAP, McLean, VA, pp. 14–21 (2002)