OptiVE: An Interactive Platform for the Design and Analysis of Virtual Enterprises

Yun Guo, Alexander Brodsky, and Amihai Motro

Computer Science Department, George Mason University
Fairfax, VA, USA
{yguo7,brodsky,ami}@gmu.edu

Abstract. OptiVE is a platform for designing and analyzing virtual enterprises. OptiVE has two modes of operations. In *composition* mode, business entrepreneurs can define elementary enterprises as well as create complex virtual enterprises from enterprises already registered in the system. In *analysis* mode, business analysts can explore the structure and properties of registered enterprises and ask the system to optimize them: Find a particular combination of participants and a specific production path that will deliver the best outcome (produce the target product at the lowest overall cost). OptiVE is founded on a formal model with rigorous syntax and semantics and uses mixed integer linear programming (MILP) to find its optimal solutions. A prototype implementation of OptiVE is also described. The system insulates its users from technical details, offering an intuitive graphical user interface for operating in either mode.

Keywords: Virtual enterprise, design, analysis, optimization.

1 Introduction and Background

One of the attractive properties of virtual enterprises is their *agility*: The ability to respond quickly to changes in the business environment either by creating new enterprises or by adapting existing ones. Arguably, this particular property has the potential to not only reorient individual enterprises, but to invigorate entire areas of business and industry. An important necessity in this regard are flexible and intuitive tools with which business entrepreneurs and analysts can design and analyze virtual enterprises.

In this paper we describe OptiVE: An interactive platform for the design and analysis of virtual enterprises. OptiVE allows business entrepreneurs and analysts who have a target product they wish to manufacture to conveniently browse a directory of business entities that are available for networked collaborations, look up their supply or manufacturing capabilities, as well as their prices and costs. Once the business entities that can assist in the manufacturing of the target product have been selected, OptiVE automatically interconnects them according to the items they consume and produce. The network thus created is then optimized to find a particular combination of the participants and a specific production path that will deliver the best results (e.g., produce the target product at the lowest overall cost). The resulting enterprise may then be registered

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 199–207, 2013.

in the system to be incorporated in future enterprises. A prototype system of OptiVE was implemented to allow experimentation and gain additional insights.

Several additional contributions of this work are worth mentioning. First, OptiVE is solidly founded on a formal model for which rigorous syntax and semantics have been provided. The model is generic (i.e., domain-independent) and includes only a small number of definitions, yet it can be extended easily to incorporate additional features. Second, OptiVE's focus is to help analysts make optimal decisions. Whereas most design choices are based on intuition and experience, OptiVE models each enterprise design as a mixed integer linear programming (MILP) optimization problem, and suggests an optimal decision. Finally, although the model is mathematically rigorous, the system insulates its users from overly technical details, providing them with an intuitive graphical interface with which their design and analysis tasks may be achieved.

These contributions are described in two sections. Section 2 describes the formal OptiVE model. It includes definitions for three different types of enterprises — manufacturers, suppliers and virtual enterprises — and allows for recursive embedding of virtual enterprises in more complex enterprises. Each enterprise is described by properties such as the items it produces, its production capacity, the items it consumes, its prices and costs, as well as a small set of constraints that govern these parameters. Finally, the semantics of this model are described by defining queries on an enterprise and how they are answered. Section 3 describes a prototype implementation of OptiVE, its software architecture and its functionalities. Essentially, OptiVE operates in two modes: a *composition* mode, in which new enterprises may be defined, and an *analysis* mode, in which existing enterprises may be inspected and optimized.

Before we begin, we provide a brief review of relevant work on the design and formation of new virtual enterprises. The formation of new enterprises is a critical and involved phase in which decisions taken have long-term impact on the eventual success of the enterprise. As proposed in [4], most of the approaches to this phase can be classified in three major categories: manual, agent-based, and service-based. The formation of new virtual enterprises requires a process of partner search and selection. Typically this process is based on traditional "competency" matching rules, and is mostly *manual*, though there have been attempts to take advantage of computer assistance [3]. In a *multi-agent based* approach, initial negotiations among business entities to achieve a common business goal are performed by representative agents [2], [9], [8]. These works often deal with negotiation protocols, auction mechanisms, and distributed matching. In a *service-oriented* approach, the collaborators are published services and enterprise creation involves the selection of services from service directory [10], [5], [7]. Service selection is often based on available quality-of-service (QoS) parameters. Yet there are other contributions that elude this classification. For example, [6] describes a formal process of repetitive refinement that virtual enterprises undergo until they acquire stability.

The work described in this paper can be classified as *computer-assisted enterprise design*. It provides designers with an interactive, graphical tool with which

they can select participants (based on their published properties) and configure them in an optimal supply chain. It is unique in that it supports business decisions that are guided by formal utility functions. The work on collaborating business entities was explored in [1], where entities were represented by an "adaptive trade specification", a specification that can capture the models of suppliers and manufacturers described in this paper. However, [1] did not consider issues such as building a tool for defining virtual enterprises, or constructing new enterprises from a library of previously defined components.

2 The OptiVE Model

OptiVE defines three kinds of enterprises: *manufacturer*, *supplier*, and *virtual*. Manufacturers and suppliers are elementary enterprises; that is, they do not embed other enterprises in their operations; whereas virtual enterprises may embed manufacturers, suppliers, and, recursively, other virtual enterprises.

Items and Enterprises. Each virtual enterprise is associated with a set of *input items* that it consumes and a set of *output items* that it produces. Manufacturers and suppliers are also associated with output items, but whereas a manufacturer consumes input items, a supplier does not.

All input and output items are unique; that is, each item, whether input or output, is assigned a unique identifier *id* (it never appears in more than one item set). Each item is assigned a *type*, and items of the same type are completely interchangeable. Thus, an item can be viewed as an instantiation of a type. Each item is also associated with a variable that indicates a quantity of the item. Altogether, an item is a triple: *item*=<*id*, *type*, *quantity*>. The notation *type*(*id*) and *quantity*(*id*) will denote the type and quantity of a particular item. Note that *type*(*id*) is a constant (typically, a string of characters), whereas *quantity*(*id*) is a numeric variable (typically, an integer).

Each enterprise is described with a quadruple: $ent = <eid, kind, in, out>$, where *eid* is a unique enterprise identifier, *kind* is either "manufacturer", "supplier" or "virtual", and *in* and *out* are sets of item *id*'s. The notation *kind*(*eid*), *in*(*eid*) and *out*(*eid*) will denote the kind and the input and output sets of a particular enterprise. We now define the three different kinds of enterprises.

Manufacturer. A manufacturer is an elementary enterprise. It is defined with a quadruple <*ent*, *bill-of-materials*, *catalog*, *constraints*>, where *kind*(*ent*)= "manufacturer".[1] *bill-of-materials* is a three-column table that describes the composition of each output item produced by this manufacturer. In this table, each output item $o \in out$ is represented with a set of triples, where each triple describes a single input item i that is required to manufacture o, and the corresponding quantity: $< o, i, quantity >$. The notation $quantity(o, i)$ will denote the quantity of input item i required to manufacture one output item o. *catalog* is a two-column table that associates with each output item $i \in out$ a

[1] Recall that *ent* is a quadruple, so a manufacturer is described altogether by 7 parameters.

price-per-unit $price(i)$. Finally, two constraints are associated with each manufacturer. The first (Equation 1) defines the cost of a manufacturer in terms of the quantities and the price-per-unit, for all the items that it manufactures. The second (Equation 2) guarantees that each input item has the quantity necessary to manufacture the output items that require it.

Supplier. A supplier is an elementary enterprise that requires no input items. It is defined with a triple <*ent*, *catalog*, *constraints*>, where $kind(ent)$= "supplier" and $in(ent) = \emptyset$. *catalog* is a three-column table that, for every output item $i \in out$ lists the price-per-unit $price(i)$ and the maximal quantity that can be supplied $capacity(i)$. Finally, two constraints are associated with each supplier. The first (Equation 3) defines the cost of a supplier in terms of the quantities and their price-per-unit, for all the items that it supplies. The second (Equation 4) limits quantities to their corresponding capacities.

Virtual Enterprise. A virtual enterprise is defined with a triple <*ent*, *participants*, *constraints*>, where $kind(ent)$="virtual", and *participants* is a set of *eid*s of the embedded enterprises.[2] Again, two constraints are associated with each virtual enterprise. The first (Equation 5) defines the cost of an enterprise in terms of the costs of its participants. The second (Equation 6) governs the quantity of input items and output items. In this equation, T_{eid} is the set of all item types in the virtual enterprise *eid*. For each type $t \in T_{eid}$, $in(t)$ is the set of item *id*s of all the input items of type t for all the embedded enterprises, plus the item *id*s of the *outputs* of type t of the virtual enterprise; $out(t)$ is the set of item *id*s of all the output items of type t of all the embedded enterprises, plus the item *id*s of the *inputs* of type t of the virtual enterprise. Note that $in(t)$ combines the inputs of the enterprise *eid* with the outputs of its embedded enterprises; this is because both describe commodities that are available to satisfy the inputs of embedded enterprises. Similarly, $out(t)$ combines the outputs of *eid* with the inputs of its embedded enterprises, because both describe commodities that must be satisfied by the outputs of the embedded enterprises.

$$cost(eid) = \sum_{i \in out(eid)} quantity(i) \cdot price(i) \tag{1}$$

$$(\forall i \in in(eid))\ quantity(i) = \sum_{o \in out(eid)} quantity(o) \cdot quantity(o, i) \tag{2}$$

$$cost(eid) = \sum_{i \in out(eid)} quantity(i) \cdot price(i) \tag{3}$$

$$(\forall i \in out)\ quantity(i) \leq capacity(i) \tag{4}$$

[2] The participants embedded in a virtual enterprise must not be involved in cycles. That is, when each participant with output type t is connected to each participant with input type t, the resulting graph should be acyclic.

$$cost(eid) = \sum_{e \in participants} cost(e) \tag{5}$$

$$(\forall t \in T_{eid}) \sum_{i \in in(t)} quantity(i) = \sum_{i \in out(t)} quantity(i) \tag{6}$$

Observe that equations (1), (3) and (5) define enterprise costs, whereas equations (2), (4) and (6) express quantity constraints. Although, the overall cost of manufacturers (Equation 1) and suppliers (Equation 3) are defined similarly, the associated item *price* is slightly different. For a supplier (who does not require input items), it is the overall charge for an item, whereas for a manufacturer, it is a manufacturing cost (on top of the cost of its input items). Note that in each case *enterprise cost* refers to the cost of production, which assumes to incorporate all other enterprise costs.

Semantics. So far we have described the *syntax* of the OptiVE model. To express its *semantics*, we define queries against enterprises and their answers. Recall that each enterprise (whether manufacturer, supplier or virtual) is associated with a definition. The set of definitions for all the enterprises that are embedded in a given enterprise is called the enterprise *library* (it includes the definition of the given enterprise itself). Recall also that each enterprise has *quantity* variables for each of its inputs or outputs.[3] We denote with $var(eid)$ the set of all the quantity variables in all the enterprises in the library of *eid*. Similarly, each enterprise is associated with six constraints. We denote with $con(eid)$ the set of all the constraints in all the enterprises in the library of *eid*. In addition to the constraints in this set, we allow users who present queries to include additional (optional) constraints (for example, to set a limit on a particular quantity).

We now define an OptiVE query on an enterprise *eid* as a pair $< I, C >$. The first argument I is a partial instantiation of the variables in $var(eid)$; that is, I assigns constant values to some (possibly none) of the quantity variables. The constraints that express this instantiation will be denoted C_I. The second argument C is a set of user-defined constraints (it could be empty).

The answer to this query is an instantiation of the variables in $var(eid)$ that minimizes $cost(eid)$ subject to the constraints in $con(eid)$, C and C_I. That is, the free quantity variables are assigned values that will minimize the overall cost of the enterprise, while satisfying all three types of constrains. Formally,

$$\begin{array}{ll} \textbf{argmin}_{var(eid)} & cost(eid) \\ \textbf{subject to} & con(eid) \wedge C \wedge C_I \end{array}$$

3 Development Platform for Virtual Enterprises

A prototype of the OptiVE development framework for virtual enterprises was implemented and tested. The system works in two modes : *composition* mode and

[3] Note that $quantity(o, i)$ used by manufacturers is not a variable, but a constant describing the quantity of items of input item i required to manufacture one output item o.

analysis mode. The system uses Microsoft .NET Framework 4 as the main software platform. SAXON-HE 9.4 edition XQuery processor and Microsoft GLEE (Automatic Graphic Layout) library are the core components in composition mode, and IBM CPLEX Mixed Integer Linear Programming solver is used in analysis mode. The overall architecture of the system is illustrated in Figure 1.

In composition mode, enterprise entrepreneurs manage enterprise configurations. Commands are issued through the graphical user interface; the business logic layer parses each command and stores or retrieves data from a dedicated XML database; and the graphical layout module renders the resulting configuration. In analysis mode, business analysts explore existing enterprises. Commands are issued through the graphical user interface; the business logic layer generates an optimization model and related data are retrieved from XML database; both model and data are then are fed into the solver, which solves the optimization problem and sends its result back to user through the business logic layer.

Details of the two modes are explained in the following two subsections with a simple example.

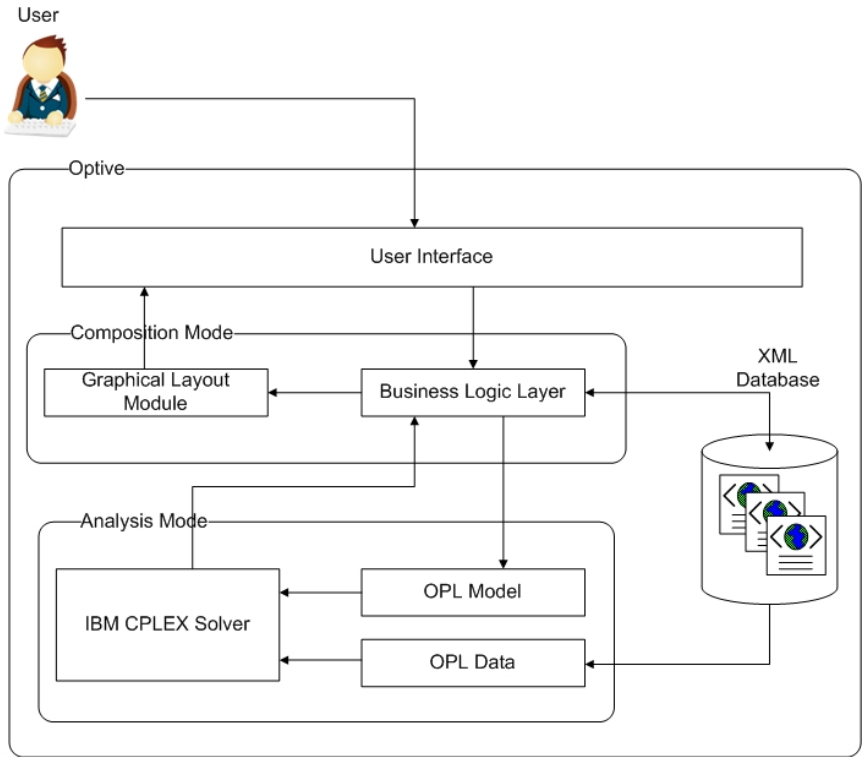

Fig. 1. System architecture of OptiVE

3.1 Composition Mode

In composition mode enterprise entrepreneurs may define elementary enterprises (manufacturers and suppliers) and configure virtual enterprises. Figure 2 illustrates the user interface to OptiVE.[4] Defining elementary enterprises is straightforward: Developers are prompted for the *eid*, *kind*, *in* items, *out* items, and *catalog*, and for manufactures, the *bill-of-materials* (note that constraints need not be specified). Once an elementary enterprise has been defined, it is added to the enterprise library (left frame) making it available for participation in virtual enterprises.

The use of composition mode to design virtual enterprises is demonstrated with the example in Figure 2. After providing the enterprise description consisting of an *eid* (MyFurniture), its *kind* (virtual), its *in* items ({Wood, Leather}) and its *out* items ({Manager Chair}), the developer drags possible participants from the enterprise library in the left frame to the canvas in the middle. To assist the developer in this task, clicking a participant in the library in the left frame, displays its description in the right frame. In the example, the candidate participants are the manufactures M1, M2, and M3, the supplier S2, and the virtual enterprise VE1. Each participant is displayed in different color and shape to reflect its kind.

The system then analyzes the supplier-customer relationships among the participants by matching input and output items according to their types. To assist in the visualization of the flow of items, a pentagon shape, named *aggregator*, is created for each item type. Participants providing a certain type of item as an *out* item are linked to that aggregator with in-coming arrows, and participants requesting that type of item as an *in* item are linked to that aggregator with out-going arrows. Note that the description of M2 shows three input item types (Chair Leg, Chair Back and Chair Seat), and one output item type (Manager Chair), so the graph shows three arrows from the aggregators of the input types to M2, and one arrow from M2 to the aggregator of the output type. Once defined, the new virtual enterprise is added to the library.

It is important to note that the graph (and the virtual enterprise that it models) incorporates all the possible item routings; as such it is an "infrastructure" on which the actual flow will eventually take place. The choice of an actual routing is discussed next.

3.2 Analysis Mode

In analysis mode business analysts can explore existing virtual enterprises. Analysts could present to OptiVE either *data queries* or *optimization queries*. Examples of data queries are "Which participants in a particular enterprise provide items of type chair?" or "What are the output items of manufacturer M2?" Such queries are answered with data extracted from the system databases. All information storage and retrieval is done using XQuery.

[4] While all the essential functionalities have been implemented, some of the features shown in this user interface are not yet available.

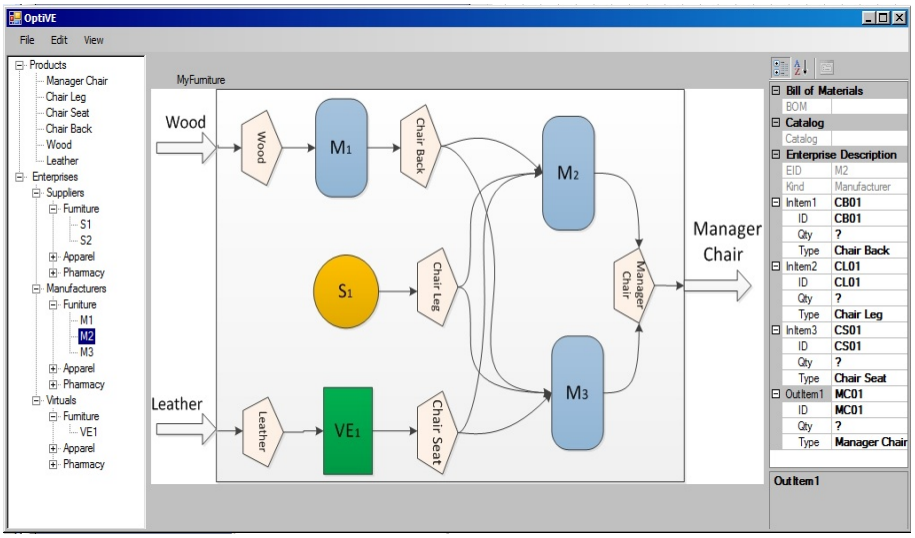

Fig. 2. User interface of OptiVE

Optimization queries are more challenging and are arguably the strongest feature of OptiVE. A typical query might be: "How can I maximize the profit in producing 100 manager chairs?" To answer such questions, OptiVE formulates a query as described at the end of the previous section. This query instantiates the variable *quantity*(Manager Chair) to 100, and sets the optimization target to *cost*(MyFurniture). OptiVE then expresses this query as an optimization problem, solves it, and displays the answer to the analyst. The answer to this query is an instantiation of the unspecified quantity variables, and hence a specification of an actual routing of items among the participants of the virtual enterprise. The description of M2 shown in Figure 2 is prior to the optimization and the quantity variables are given as "?"; when the optimization query concludes these are replaced by specific values. This particular instantiation of the virtual enterprise (the query and its answer) may then be saved as a possible "scenario".

4 Conclusion

We described OptiVE, an interactive system that assists business entrepreneurs and analysts in the creation, evolution, analysis and optimization of virtual enterprises. Both the model and the system are undergoing further research and we mention here two of our directions. First, the OptiVE model described in section 2 is being extended to incorporate (1) additional elementary enterprise types (other than manufacturers or suppliers); for example, *transporters* for moving products from one location to another; (2) additional system variables (other than quantities or prices); for example, time of delivery, reliability, past performance, or risk of failure; and (3) additional optimization targets (other than

overall cost); for example, least risk, shortest delivery time, or shortest supply chain. In addition, the user interface of OptiVE described in Section 3 is being improved to provide features that correspond to the new model, and to complete the implementation of a few features that were illustrated in Figure 2, though not yet fully available.

References

1. Brodsky, A., Zelivinski, S., Katz, M., Gozhansky, A., Karpishpan, S.: System and Method for Adaptive Trade Specification and Match-making Optimization. US patent no. US 6751597 B1
2. Camarinha-Matos, L.M., Afsarmanesh, H.: Virtual Enterprise Modeling and Support Infrastructures: Applying Multi-Agent System Approaches. In: Luck, M., Mařík, V., Štěpánková, O., Trappl, R. (eds.) ACAI 2001 and EASSS 2001. LNCS (LNAI), vol. 2086, pp. 335–364. Springer, Heidelberg (2001)
3. Camarinha-Matos, L.M., Afsarmanesh, H., Osorio, A.L.: Flexibility and Safety in a Web-based Infrastructure for Virtual Enterprises. International Journal of Computer Integrated Manufacturing 14(1), 66–82 (2001)
4. Camarinha-Matos, L.M., Silveri, I., Afsarmanesh, H., Oliveira, H.A.I.: Towards a Framework for Creation of Dynamic Virtual Organizations. In: Proceedings of PRO-VE 2005, 6th IFIP Working Conference on Virtual Enterprises (Collaborative Networks and Their Breeding Environments). IFIP AICT, vol. 186, pp. 69–80. Springer, Heidelberg (2005)
5. Danesh, M.H., Raahemi, B., Kamali, M.A.: A Framework for Process Management in Service Oriented Virtual Organizations. In: Proceedings of NWeSP 2011, 7th International Conference on Next Generation Web Services Practices, pp. 12–17. IEEE Press, New York (2011)
6. D'Atri, A., Motro, A.: Evolving VirtuE. In: Proceedings of PRO-VE 2007, 8th IFIP Working Conference on Virtual Enterprises (Establishing the Foundation of Collaborative Networks). IFIP AICT, vol. 243, pp. 317–326. Springer, Heidelberg (2007)
7. Motro, A., Guo, Y.: The SOAVE Platform: A Service Oriented Architecture for Virtual Enterprises. In: Proceedings of PRO-VE 2012, 13th IFIP Working Conference on Virtual Enterprises (Collaborative Networks in the Internet of Services). IFIP AICT, vol. 380, pp. 216–224. Springer, Heidelberg (2012)
8. Patel, J., Teacy, W.T.L., Jennings, N.R., Luck, M., Chalmers, S., Oren, N., Norman, T.J., Preece, A.D., Gray, P.M.D., Shercliff, G., Stockreisser, P.J., Shao, J., Gray, W.A., Fiddian, N.J., Thompson, S.: Agent-based Virtual Organisations for the Grid. International Journal of Multiagent and Grid Systems 1(4), 237–249 (2005)
9. Petersen, S.A., Divitini, M.: Using Agents to Support the Selection of Virtual Enterprise Teams. In: Proceedings of AOIS 2002 @ AAMAS 2002, 4th International Bi-Conference Workshop on Agent-Oriented Information Systems. CEUR-WS.org, vol. 59 (2002)
10. Zhou, B., Zhi-Jun, H., Tang, J.-F.: An adaptive model of virtual enterprise based on dynamic web service composition. In: Proceedings of CIT 2005, 5th International Conference on Computer and Information Technology, pp. 284–289. IEEE Press, New York (2005)